Simplex Consensus
A Fast and Simple Consensus Protocol

Benjamin Chan

Cornell Tech

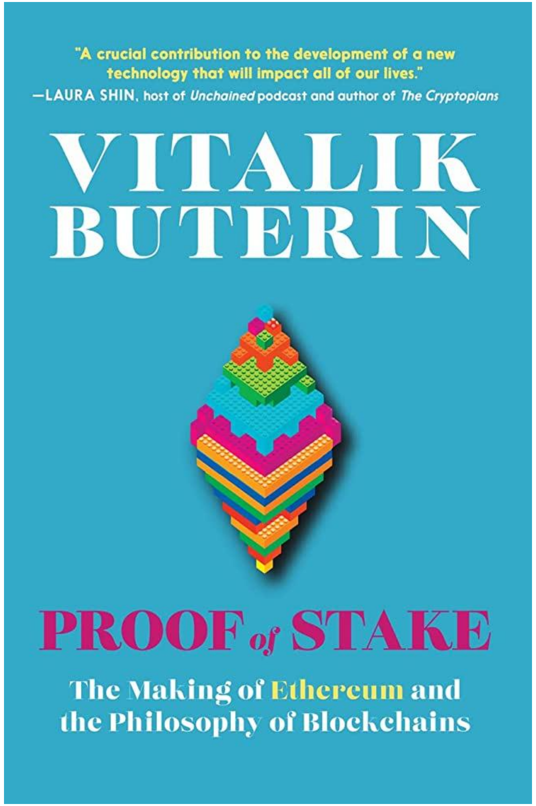
Joint work with Rafael Pass

Consensus Protocols in today's world

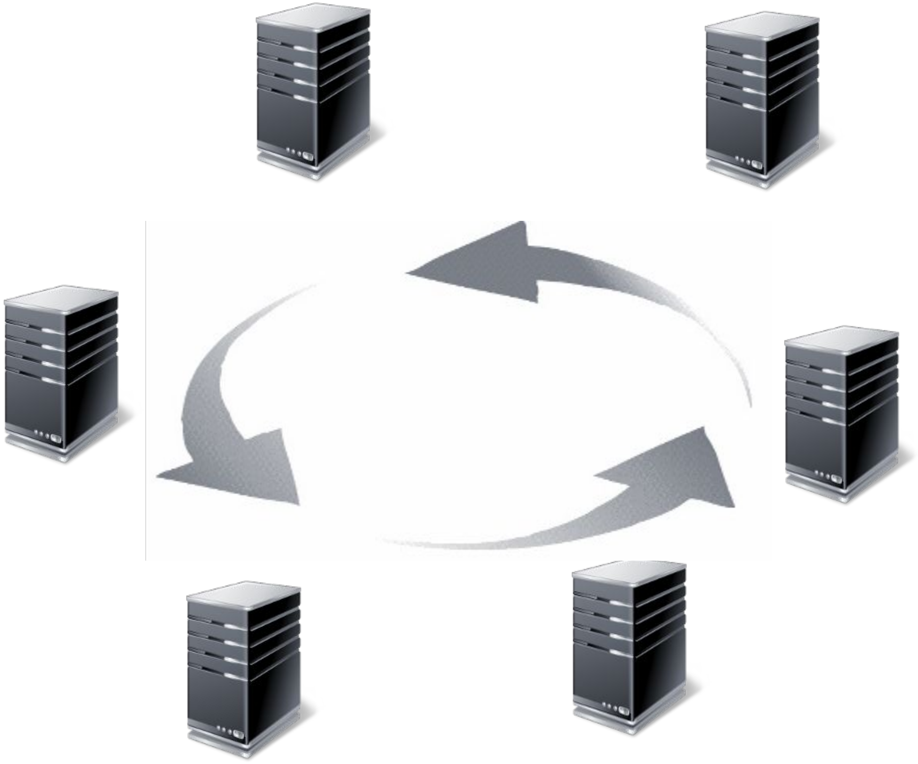
Algorand's Pure Proof of Stake Blockchain

Delivering security, scalability, decentralization and sustainability since 2019.

LEARN MORE



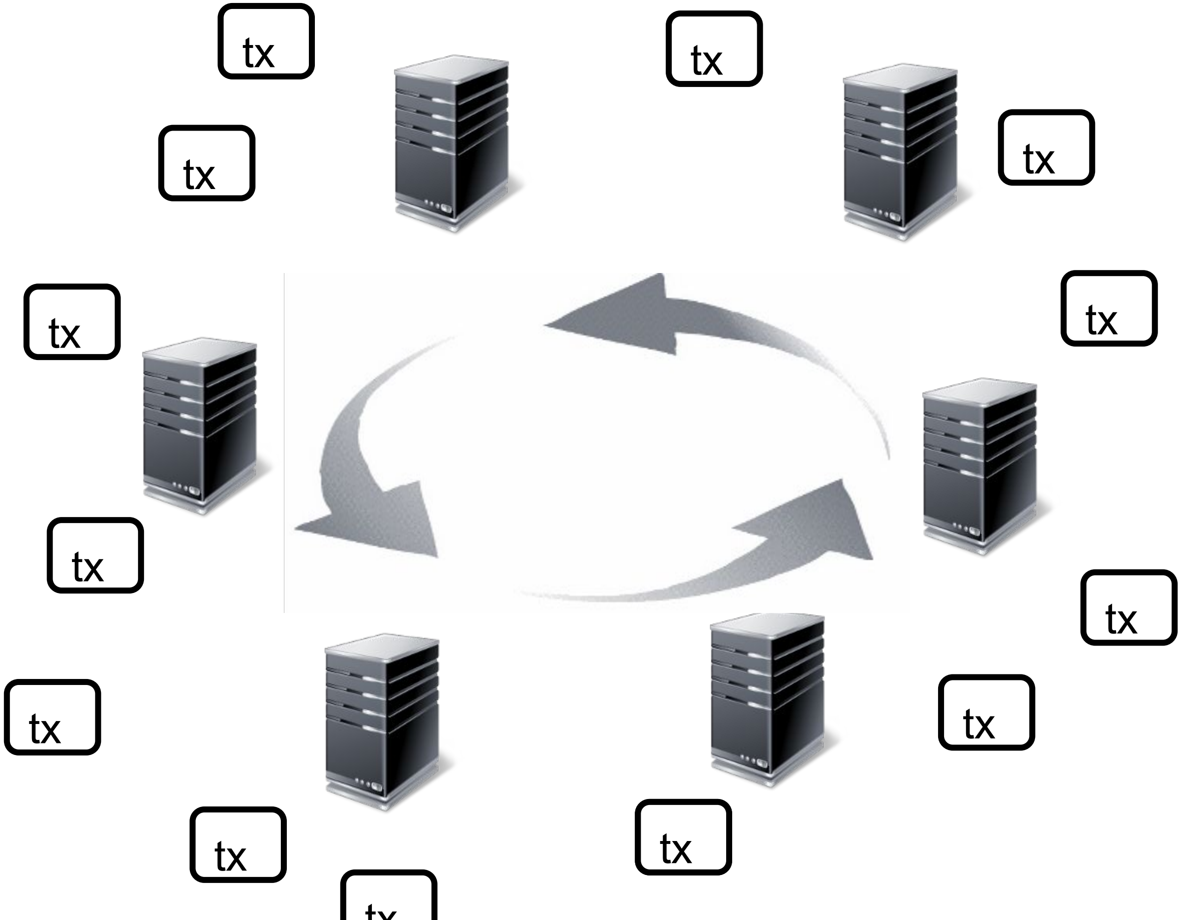
Consensus (a.k.a. state machine replication, public ledger)



- **Consistency**
- **Liveness**

hold even when
some nodes corrupted
(e.g., assume ⅔ honest)

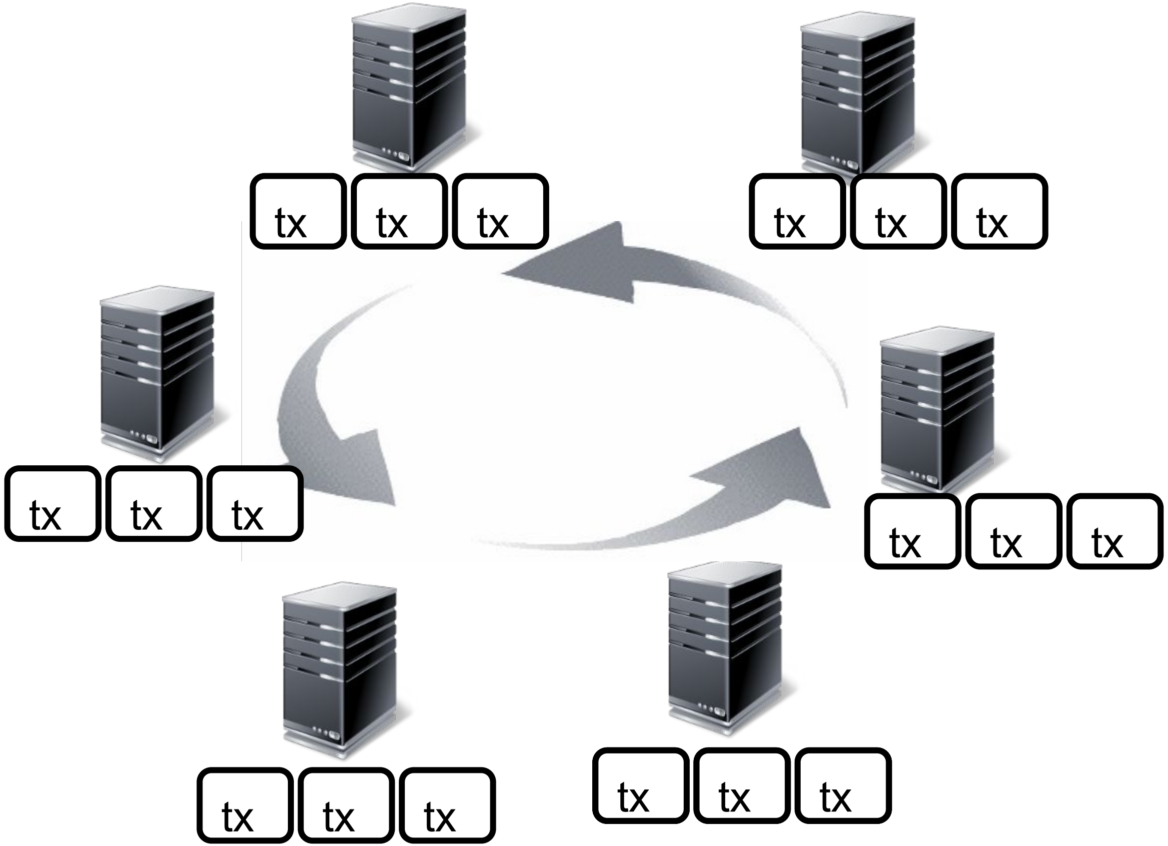
Consensus (a.k.a. state machine replication, public ledger)



- **Consistency**
- **Liveness**

hold even when
some nodes corrupted
(e.g., assume ⅔ honest)

Consensus (a.k.a. state machine replication, public ledger)



- **Consistency**
- **Liveness**

hold even when
some nodes corrupted
(e.g., assume ⅔ honest)

New era, new requirements

- thousands of players
- malicious faults
- unreliable Internet
- fast transaction confirmation time
- **fairness**

Bitcoin and Proof-of-Work

- Amazing protocol, but sub-optimal "performance":
- E.g. **Bitcoin** has
  - Transaction confirmation time: ~60 minutes (6 blocks)
  - Block time: 10 min (7 transactions per second)
- Wastes electricity and computational resources.

> "And Riot Platforms' mine in **Rockdale, Texas**, uses about the same amount of electricity as the nearest 300,000 homes, making it the most power-intensive Bitcoin mining operation in America."

(source: New York Times)

Proof-of-Stake blockchains

- Can be much more performant than **Proof-of-Work** blockchains
- E.g. Ethereum
  - Transaction confirmation time: **15 mins**
  - Block time: **12 sec**
  - Throughput: **350 tps** (assuming block size of 4200 txs)
- E.g. Algorand
  - Transaction confirmation time: **4 sec**
  - Block time: **4 sec**
  - Throughput: **1050 tps** (assuming block size of 4200 txs)
- No computational waste
- **Two different philosophies**
  - Dynamic/sleepy participation [PS'18]: "people come and go"
  - **Partial synchrony**: security even under network partitions, faster.

(sources: ethereum.org, algorand.org)

(Partially-Synchronous) Proof-of-Stake blockchains

Uses **classical permissioned consensus protocols** under-the-hood

- In classical consensus, the set of ***n*** players is known ahead of time.
- Overall latency inherited from underlying consensus protocol.
- Require additional features for “fairness”: **random-leader consensus**

This talk: classical consensus protocols for the proof-of-stake setting

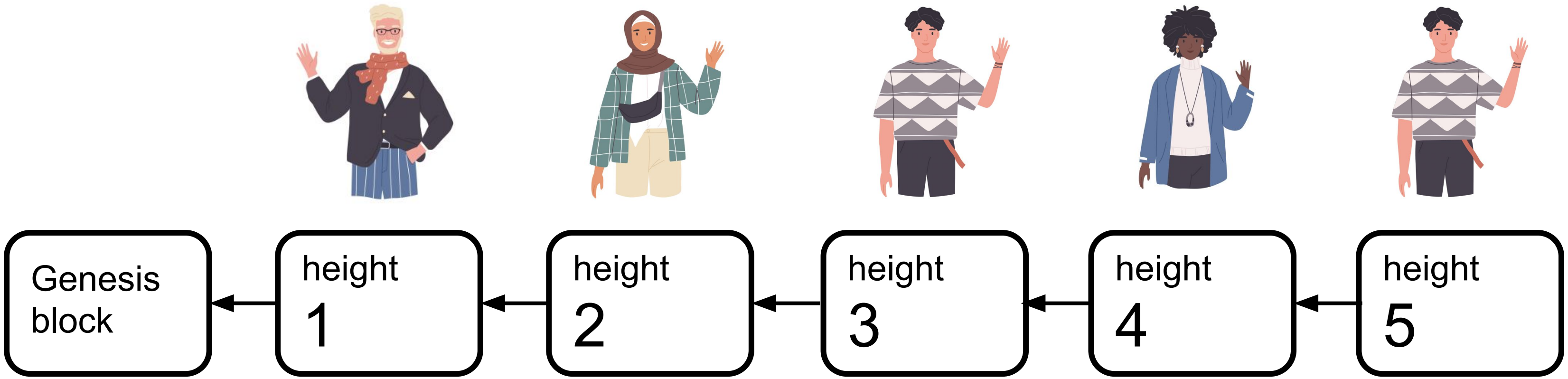
This talk:
Designing a simpler and faster random-leader consensus protocol

What do we look for in a consensus protocol?

1. **Fairness.** Each player should have a fair chance at proposing each block.

    Something like PBFT — where the same leader can propose every block for eternity — is not suitable for a blockchain application.

Random-leader consensus



What do we look for in a consensus protocol?

1. **Fairness.** Each player should have a fair chance at proposing each block.

   Something like PBFT — where the same leader can propose every block for eternity — is not suitable for a blockchain application.

What do we look for in a consensus protocol?

1. **Fairness.** Each player should have a fair chance at proposing each block.

   Something like PBFT — where the same leader can propose every block for eternity — is not suitable for a blockchain application.

2. **Latency.** Specifically, must have fast *transaction confirmation time.*
   a. The *optimistic* case: when every player is honest.
   b. The *pessimistic* case: when some players are faulty. ← Underappreciated!

What do we look for in a consensus protocol?

1. **Fairness.** Each player should have a fair chance at proposing each block.

   Something like PBFT — where the same leader can propose every block for eternity — is not suitable for a blockchain application.

2. **Latency.** Specifically, must have fast *transaction confirmation time.*
   a. The *optimistic* case: when every player is honest.
   b. The *pessimistic* case: when some players are faulty. ← Underappreciated!

3. **Easy-to-understand.** Should be easy to understand *why* the protocol is secure.

Transaction confirmation time

Suppose a transaction **tx** is provided to the protocol by time ***t***. How long does it take for **tx** to be finalized?

- Optimistic Confirmation Time (no faults)
  - **Proposal Confirmation Time**: when a new block is proposed, how long does it take for it to get confirmed?
  - **Optimistic Block Time**: how long does a transaction need to wait before being included in a block proposal?

Transaction confirmation time

Suppose a transaction **tx** is provided to the protocol by time ***t***. How long does it take for **tx** to be finalized?

- Pessimistic Confirmation Time (allowing faults)
  - **Worst-case confirmation time.** How long does it take in the worst case to be finalized?
  - **Expected Liveness**: On average, how long does it take?
    (We assume that the transaction arrives at the beginning of the **i**th block proposal opportunity.)

Partial Synchrony

The network may be unreliable, and even occasionally partitioned in half.

Formally, there is a fixed unknown time **GST**, an unknown time bound $\boldsymbol{\delta}$, and a known time bound $\boldsymbol{\Delta > \delta}$ s.t.

- **Before GST**, messages take arbitrarily long to be delivered
- **After GST**, every message is delivered within **$\boldsymbol{\delta}$ seconds.**

Partial synchrony models a flaky Internet, or implementation bugs that cause players to drop messages.

State-of-the-art

Theoretical latency of partially-synchronous protocols that support random leaders

First “random-leader” partially synchronous

	Proposal Conf. Time	Optimistic Block Time	Pessimistic Liveness ($f = \lceil n/3 \rceil - 1$)
Algorand* [CGMV18]	**3δ**	3δ	$4\delta + 2\Delta$
ICC [CDH+22]	**3δ**	**2δ**	$5.5\delta + 1.5\Delta$
PaLa [CPS18]	4δ	**2δ**	$6.25\delta + 9.25\Delta$
Pipeline Fast-Hotstuff [JNFG20] Jolteon [GKKS+22]	5δ	**2δ**	$10.87\delta + 9.5\Delta$
Chained Hotstuff (v6) [YMR+19]	7δ	**2δ**	$19.31\delta + 12.18\Delta$
Streamlet [CS20a]	10Δ	2Δ	39.56Δ

*Base protocol without sortition.

Table 1: Latency of Popular Consensus Protocols (Random Leaders)

State-of-the-art

Theoretical latency of partially-synchronous protocols that support random leaders

These protocols pipeline their block proposals to achieve **2δ** block time

	Proposal Conf. Time	Optimistic Block Time	Pessimistic Liveness ($f = \lceil n/3 \rceil - 1$)
Algorand* [CGMV18]	$\mathbf{3\delta}$	3δ	$4\delta + 2\Delta$
ICC [CDH+22]	$\mathbf{3\delta}$	$\mathbf{2\delta}$	$5.5\delta + 1.5\Delta$
PaLa [CPS18]	4δ	$\mathbf{2\delta}$	$6.25\delta + 9.25\Delta$
Pipeline Fast-Hotstuff [JNFG20] Jolteon [GKKS+22]	5δ	$\mathbf{2\delta}$	$10.87\delta + 9.5\Delta$
Chained Hotstuff (v6) [YMR+19]	7δ	$\mathbf{2\delta}$	$19.31\delta + 12.18\Delta$
Streamlet [CS20a]	10Δ	2Δ	39.56Δ

*Base protocol without sortition.

Table 1: Latency of Popular Consensus Protocols (Random Leaders)

State-of-the-art

Theoretical latency of partially-synchronous protocols that support random leaders

However, they require multiple honest leaders in-a-row to confirm blocks, which hurts pessimistic liveness.

	Proposal Conf. Time	Optimistic Block Time	Pessimistic Liveness ($f = \lceil n/3 \rceil - 1$)
Algorand* [CGMV18]	$\mathbf{3\delta}$	3δ	$4\delta + 2\Delta$
ICC [CDH+22]	$\mathbf{3\delta}$	$\mathbf{2\delta}$	$5.5\delta + 1.5\Delta$
PaLa [CPS18]	4δ	$\mathbf{2\delta}$	$6.25\delta + 9.25\Delta$
Pipeline Fast-Hotstuff [JNFG20] Jolteon [GKKS+22]	5δ	$\mathbf{2\delta}$	$10.87\delta + 9.5\Delta$
Chained Hotstuff (v6) [YMR+19]	7δ	$\mathbf{2\delta}$	$19.31\delta + 12.18\Delta$
Streamlet [CS20a]	10Δ	2Δ	39.56Δ

*Base protocol without sortition.

Table 1: Latency of Popular Consensus Protocols (Random Leaders)

State-of-the-art

Theoretical latency of partially-synchronous protocols that support random leaders

Protocols that don't pipeline blocks usually sacrifice block time, but get good expected liveness

	Proposal Conf. Time	Optimistic Block Time	Pessimistic Liveness ($f = \lceil n/3 \rceil - 1$)
Algorand* [CGMV18]	$\mathbf{3\delta}$	3δ	$4\delta + 2\Delta$
ICC [CDH+22]	$\mathbf{3\delta}$	$\mathbf{2\delta}$	$5.5\delta + 1.5\Delta$
PaLa [CPS18]	4δ	$\mathbf{2\delta}$	$6.25\delta + 9.25\Delta$
Pipeline Fast-Hotstuff [JNFG20] Jolteon [GKKS+22]	5δ	$\mathbf{2\delta}$	$10.87\delta + 9.5\Delta$
Chained Hotstuff (v6) [YMR+19]	7δ	$\mathbf{2\delta}$	$19.31\delta + 12.18\Delta$
Streamlet [CS20a]	10Δ	2Δ	39.56Δ

*Base protocol without sortition.

Table 1: Latency of Popular Consensus Protocols (Random Leaders)

State-of-the-art

Theoretical latency of partially-synchronous protocols that support random leaders

	Proposal Conf. Time	Optimistic Block Time	Pessimistic Liveness ($f = \lceil n/3 \rceil - 1$)
Algorand* [CGMV18]	$\mathbf{3\delta}$	3δ	$4\delta + 2\Delta$
ICC [CDH+22]	$\mathbf{3\delta}$	$\mathbf{2\delta}$	$5.5\delta + 1.5\Delta$
PaLa [CPS18]	4δ	$\mathbf{2\delta}$	$6.25\delta + 9.25\Delta$
Pipeline Fast-Hotstuff [JNFG20] Jolteon [GKKS+22]	5δ	$\mathbf{2\delta}$	$10.87\delta + 9.5\Delta$
Chained Hotstuff (v6) [YMR+19]	7δ	$\mathbf{2\delta}$	$19.31\delta + 12.18\Delta$
Streamlet [CS20a]	10Δ	2Δ	39.56Δ

*Base protocol without sortition.

Table 1: Latency of Popular Consensus Protocols (Random Leaders)

Easiest protocol description [**C**S20]

This talk

A new consensus protocol, called **Simplex Consensus**

- Partial synchrony, **$f < n/3$** byzantine faults
- In our eyes, easiest security proofs!
- Can get communication efficiency using "sortition" [Algorand]

Thm: Assuming a (Bare) PKI, CRH, there exists a partially synchronous consensus protocol in the "random-leader model" with:

- Proposal confirmation time of **3δ**
- Optimistic block time of **2δ**
- Expected pessimistic liveness of **$3.5\delta + 1.5\Delta$**
- Worst-case liveness of **$4\delta + \omega(\log \lambda) \cdot (3\Delta + \delta)$**

This talk

A new consensus protocol, called **Simplex Consensus**

- Partial synchrony, **$f < n/3$** byzantine faults
- In our eyes, **easiest liveness proof**
- Can get communication efficiency using “sortition” [Algorand]

Thm: Assuming a (Bare) PKI, CRH, there exists a partially synchronous consensus protocol in the “random-leader model” with:

- Proposal confirmation time of **3δ**
- Optimistic block time of **2δ**
- Expected pessimistic liveness of **$3.5\delta + 1.5\Delta$**
- Worst-case liveness of **$4\delta + \omega(\log \lambda) \cdot (3\Delta + \delta)$**

This talk

A new consensus protocol, called **Simplex Consensus**

- Partial synchrony, **$f < n/3$** byzantine faults
- In our eyes, **easiest liveness proof**
- Can get communication efficiency using “sortition” [Algorand]

Essentially all prior work in this model has non-trivial liveness proof

Thm: Assuming a (Bare) PKI, CRH, there exists a partially synchronous consensus protocol in the “random-leader model” with:

- Proposal confirmation time of **3δ**
- Optimistic block time of **2δ**
- Expected pessimistic liveness of **$3.5\delta + 1.5\Delta$**
- Worst-case liveness of **$4\delta + \omega(\log \lambda) \cdot (3\Delta + \delta)$**

Comparisons

Theoretical latency of protocols that support random leaders

Simplex:
The best of both worlds

	Proposal Conf. Time	Optimistic Block Time	Pessimistic Liveness ($f = \lceil n/3 \rceil - 1$)
Simplex	$\mathbf{3\delta}$	$\mathbf{2\delta}$	$\mathbf{3.5\delta + 1.5\Delta}$
Algorand* [CGMV18]	$\mathbf{3\delta}$	3δ	$4\delta + 2\Delta$
ICC [CDH+22]	$\mathbf{3\delta}$	$\mathbf{2\delta}$	$5.5\delta + 1.5\Delta$
PaLa [CPS18]	4δ	$\mathbf{2\delta}$	$6.25\delta + 9.25\Delta$
Pipeline Fast-Hotstuff [JNFG20] Jolteon [GKKS+22]	5δ	$\mathbf{2\delta}$	$10.87\delta + 9.5\Delta$
Chained Hotstuff (v6) [YMR+19]	7δ	$\mathbf{2\delta}$	$19.31\delta + 12.18\Delta$
Streamlet [CS20a]	10Δ	2Δ	39.56Δ

*Base protocol without sortition.

Table 1: Latency of Popular Consensus Protocols (Random Leaders)

Comparisons

Theoretical latency of protocols that support random leaders

Simplex:
The best of both worlds

In our eyes, also easier to understand.

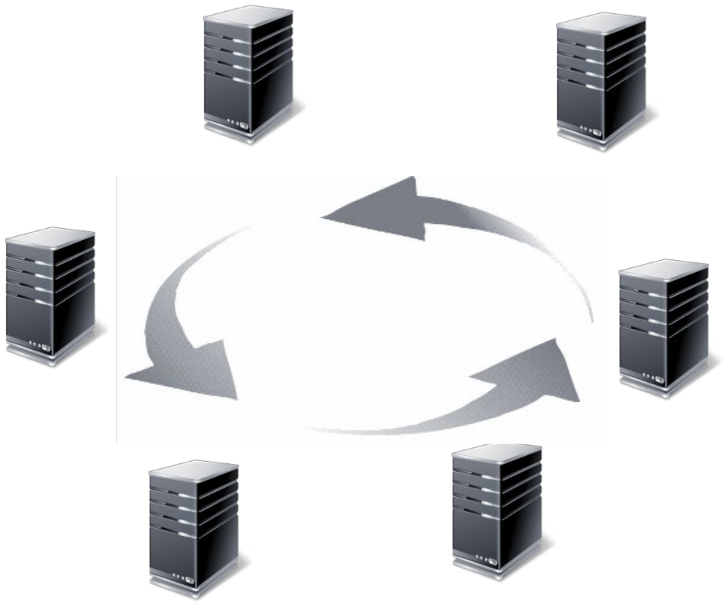
	Proposal Conf. Time	Optimistic Block Time	Pessimistic Liveness ($f = \lceil n/3 \rceil - 1$)
Simplex	$\boldsymbol{3\delta}$	$\boldsymbol{2\delta}$	$\boldsymbol{3.5\delta + 1.5\Delta}$
Algorand* [CGMV18]	$\boldsymbol{3\delta}$	3δ	$4\delta + 2\Delta$
ICC [CDH+22]	$\boldsymbol{3\delta}$	$\boldsymbol{2\delta}$	$5.5\delta + 1.5\Delta$
PaLa [CPS18]	4δ	$\boldsymbol{2\delta}$	$6.25\delta + 9.25\Delta$
Pipeline Fast-Hotstuff [JNFG20] Jolteon [GKKS+22]	5δ	$\boldsymbol{2\delta}$	$10.87\delta + 9.5\Delta$
Chained Hotstuff (v6) [YMR+19]	7δ	$\boldsymbol{2\delta}$	$19.31\delta + 12.18\Delta$
Streamlet [CS20a]	10Δ	2Δ	39.56Δ

*Base protocol without sortition.

Table 1: Latency of Popular Consensus Protocols (Random Leaders)

Protocol Description

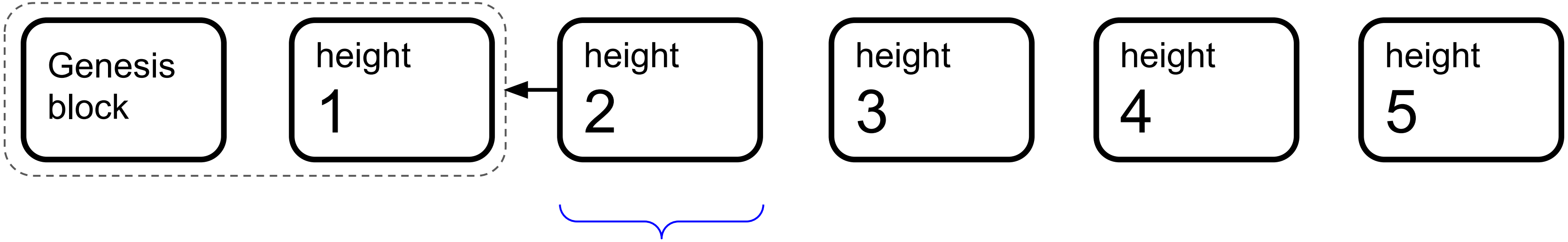
Simplex Consensus



n players, f < n/3 malicious faults.
we know their public keys ahead of time (bare PKI)

Simplex Consensus

Key data structure: **blockchain**

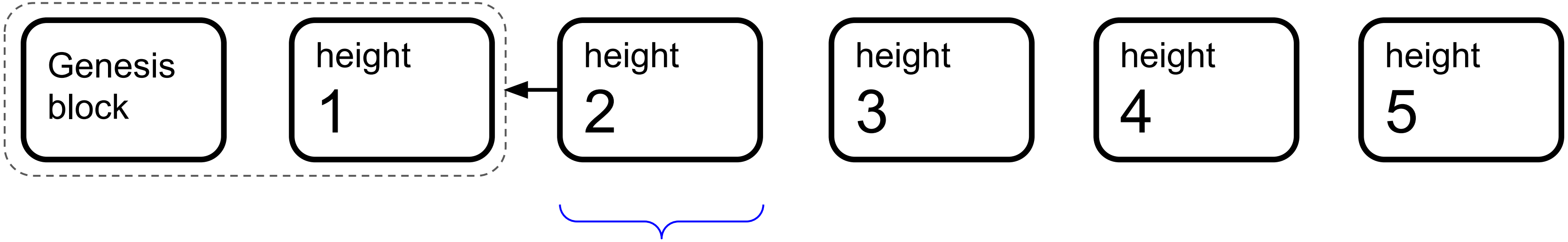


each block of height **h** is a tuple of the form

$\mathbf{b_h}$ **= (h,** *hash of a parent chain*, **txs)**

Simplex Consensus

Key data structure: **blockchain**

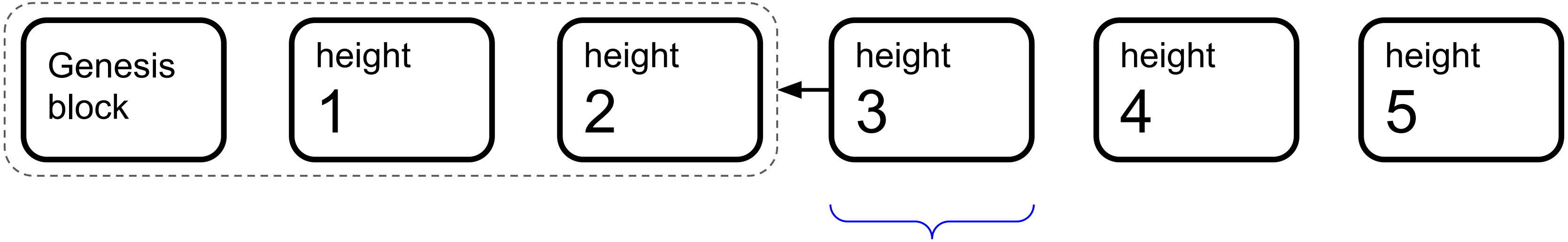


each block of height **h** is a tuple of the form

$$\mathbf{b_h = (h,}\ \mathit{Hash}\mathbf{(b_1 \ldots b_{h-1}), txs)}$$

Simplex Consensus

Key data structure: **blockchain**

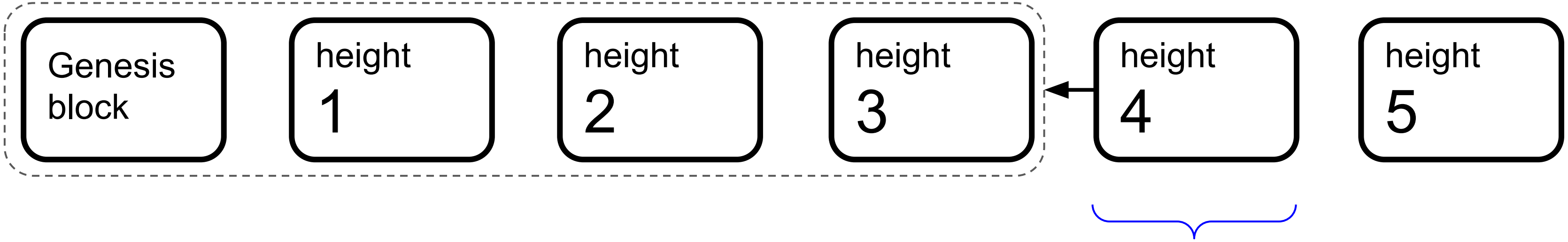


each block of height **h** is a tuple of the form

$$\mathbf{b_h = (h,}\ \mathit{Hash}\mathbf{(b_1 \ldots b_{h-1}), txs)}$$

Simplex Consensus

Key data structure: **blockchain**

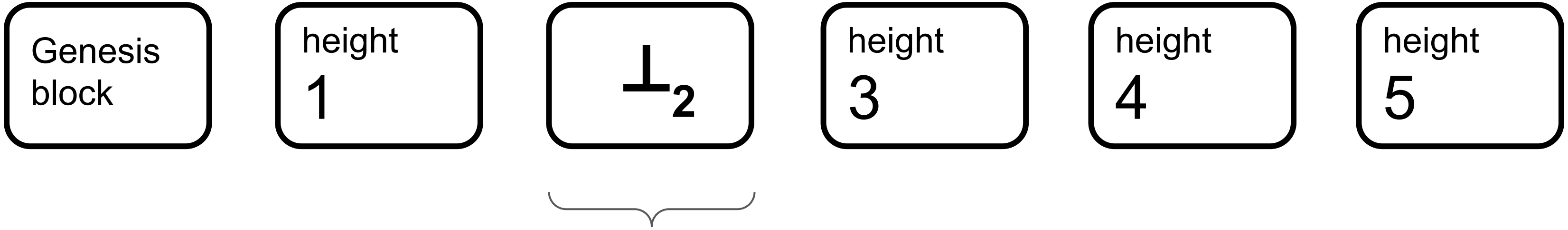


each block of height **h** is a tuple of the form

$$\mathbf{b_h = (h, \mathit{Hash}(b_1 \ldots b_{h-1}), txs)}$$

Dummy blocks

We also allow the blockchain to contain “**dummy blocks**”

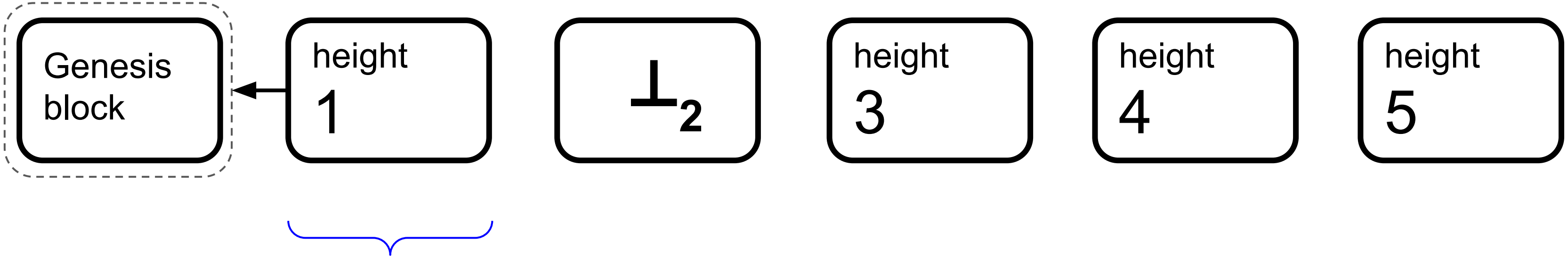


a dummy block of height **h** is the tuple

$$\perp_h = (h, \perp, \perp)$$

Dummy blocks

We also allow the blockchain to contain "**dummy blocks**"

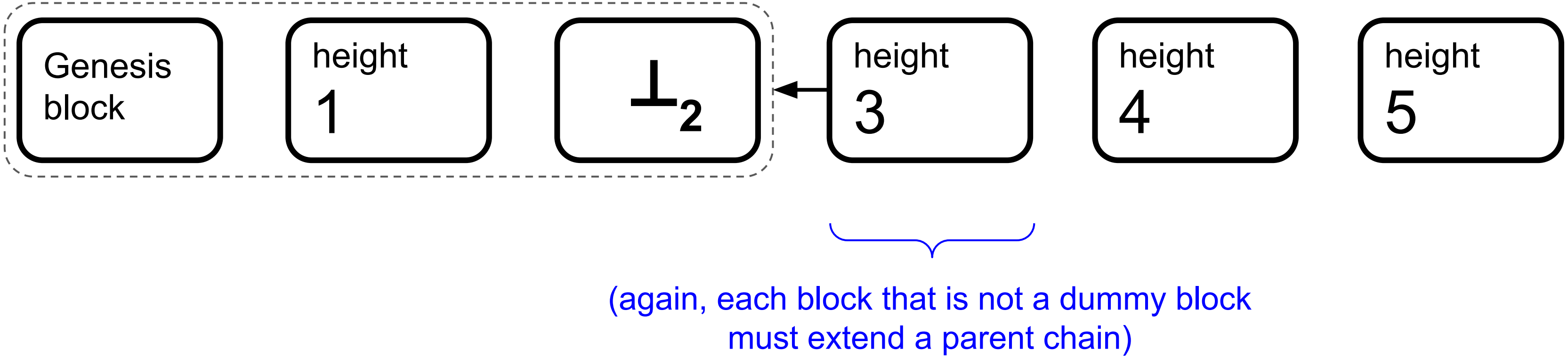


(again, each block that is not a dummy block must extend a parent chain)

$$\mathbf{b_h = (h,}\ \mathit{Hash}\mathbf{(b_1 \ldots b_{h-1}), txs)}$$

Dummy blocks

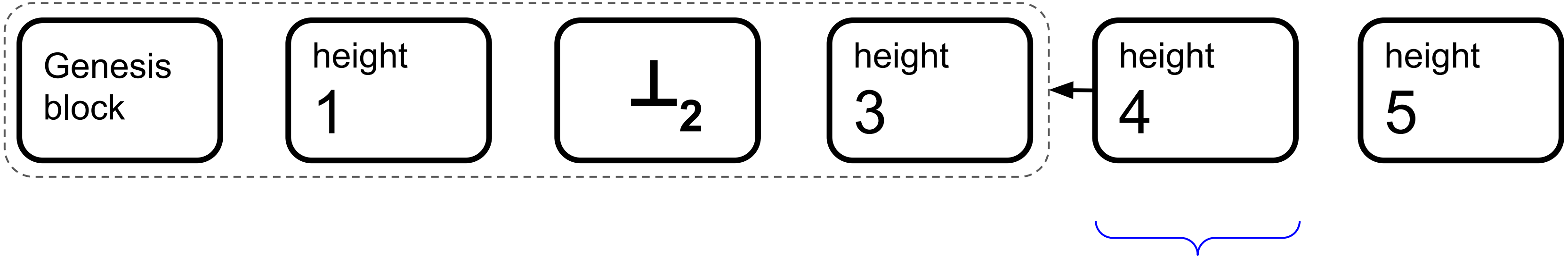
We also allow the blockchain to contain "**dummy blocks**"



$$\mathbf{b_h = (h,\ } \mathit{Hash}\mathbf{(b_1 \ldots b_{h-1}),\ txs)}$$

Dummy blocks

We also allow the blockchain to contain “**dummy blocks**”

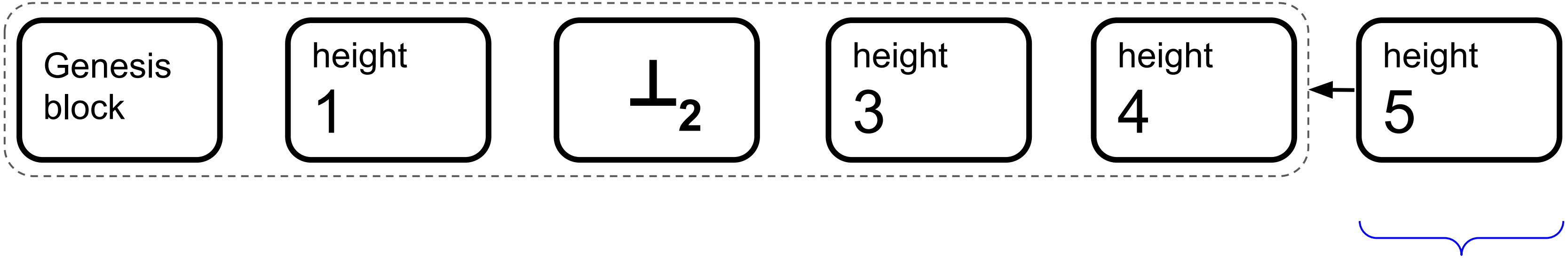


(again, each block that is not a dummy block must extend a parent chain)

$$\mathbf{b_h = (h,}\ \mathit{Hash}\mathbf{(b_1 \dots b_{h-1}), txs)}$$

Dummy blocks

We also allow the blockchain to contain "**dummy blocks**"

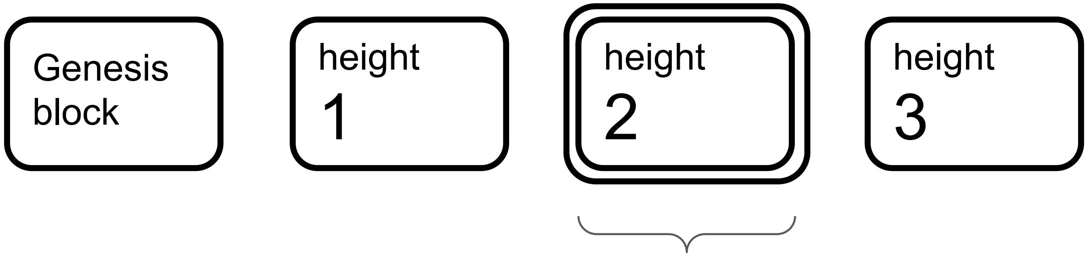


(again, each block that is not a dummy blo
must extend a parent chain)

$$\mathbf{b_h = (h, \mathit{Hash}(b_1 \ldots b_{h-1}), txs}$$

Notarized blocks

Key data structure: **notarized blocks**



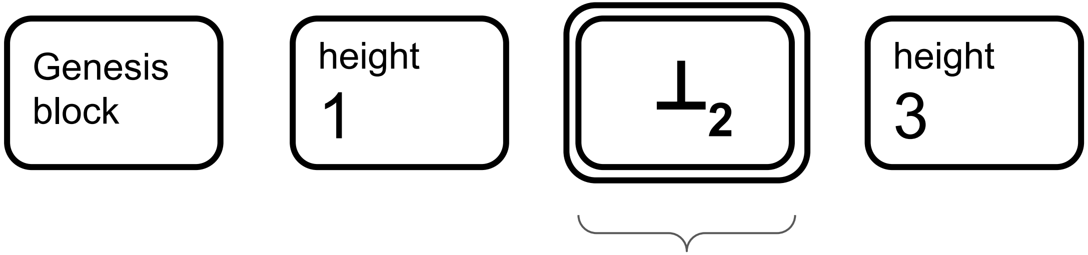
a block is notarized in my view if I've seen

> 2n/3 votes for it

a vote for **b** = a signed message "vote for **b**"

Notarized blocks

Dummy blocks can also be **notarized.**



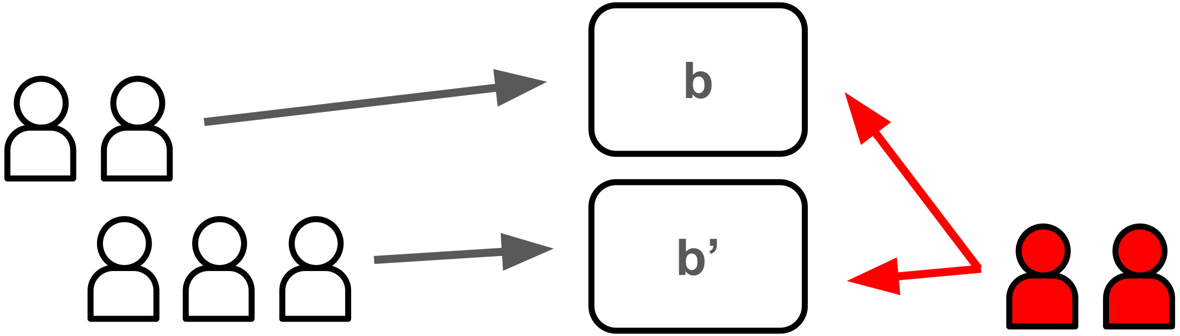
a block is notarized in my view if I've seen

> 2n/3 votes for it

a vote for **b** = a signed message "vote for **b**"

“Quorum intersection”

If honest players only vote for one of **b** or **b’**, then it cannot be that both **2n/3** players voted for **b**, and **2n/3** players voted for **b’**.

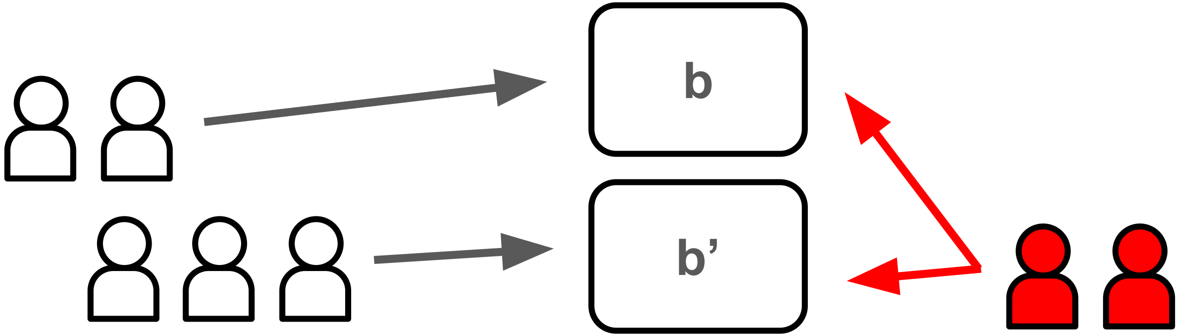


suppose each honest player only votes for one

corrupt players can always vote for both

“Quorum intersection”

If honest players only vote for one of **b** or **b’**, then it cannot be that both **2n/3** players voted for **b**, and **2n/3** players voted for **b’**.



suspose each honest player only votes for one

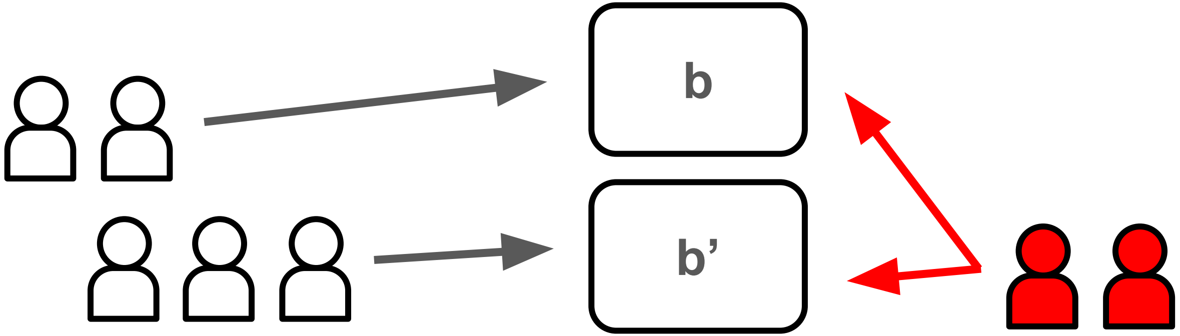
n – f votes

corrupt players can always vote for both

2f votes

“Quorum intersection”

If honest players only vote for one of **b** or **b’**, then it cannot be that both **2n/3** players voted for **b**, and **2n/3** players voted for **b’**.



suppose each honest player only votes for one

n – f votes

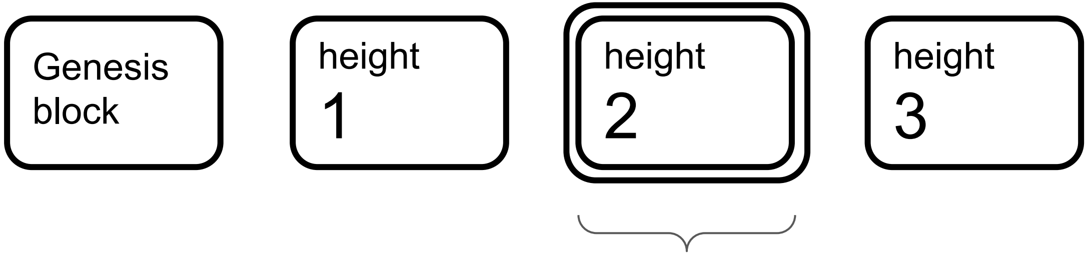
n + f total votes
< 4n/3 since **f < n/3**

corrupt players can always vote for both

2f votes

Notarized blocks

Key data structure: **notarized blocks**



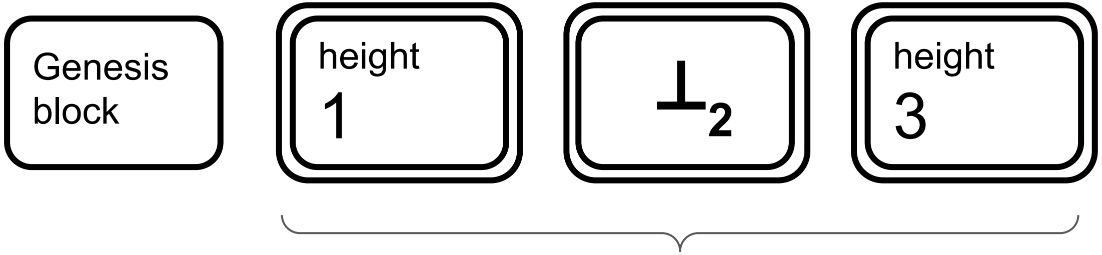
a block is notarized in my view if I’ve seen

> 2n/3 votes for it

a vote for **b** = a signed message “vote for **b**”

Notarized blockchains

Key data structure: **notarized blockchain**

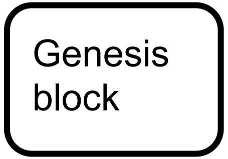


every block of the chain is notarized (except genesis)

The Simplex Consensus Protocol

Proceed in iterations **h = 1, 2, 3, …**

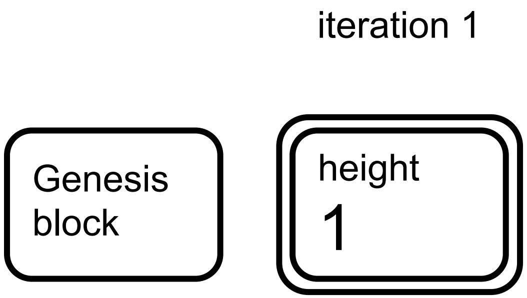
In each iteration **h**, collectively try to build a notarized block of height **h.**



The Simplex Consensus Protocol

Proceed in iterations **h = 1, 2, 3, …**

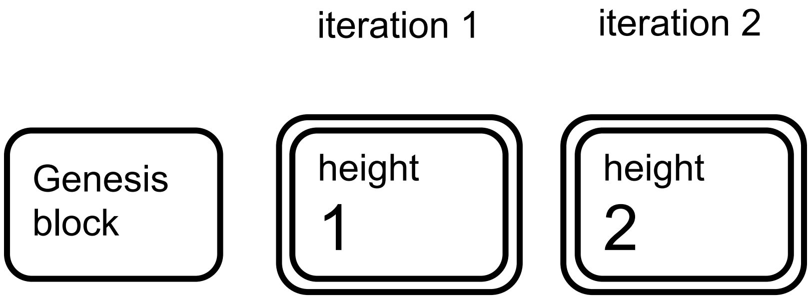
In each iteration **h**, collectively try to build a notarized block of height **h.**



The Simplex Consensus Protocol

Proceed in iterations **h = 1, 2, 3, …**

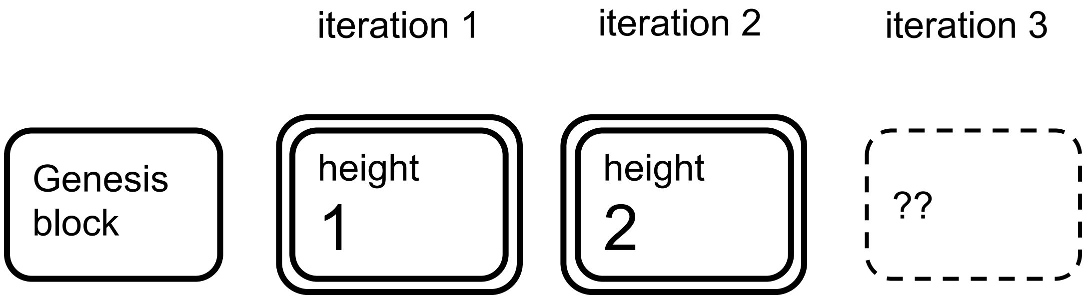
In each iteration **h**, collectively try to build a notarized block of height **h.**



The Simplex Consensus Protocol

Proceed in iterations **h = 1, 2, 3, …**

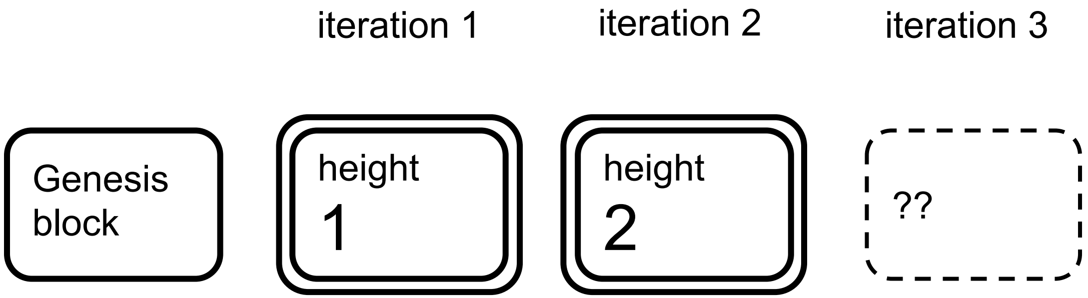
In each iteration **h**, collectively try to build a notarized block of height **h.**



The Simplex Consensus Protocol

Proceed in iterations **h = 1, 2, 3, …**

Only move to the next iteration when I've seen a notarized blockchain of length **h**.

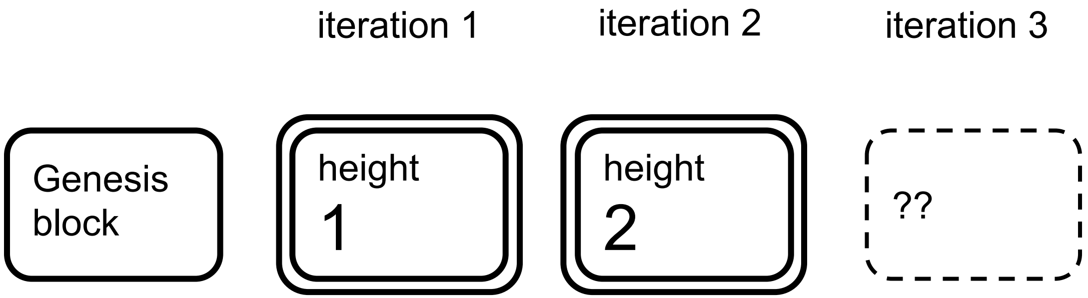


The Simplex Consensus Protocol

Proceed in iterations **h = 1, 2, 3, …**

Only move to the next iteration when I've seen a notarized blockchain of length **h**.

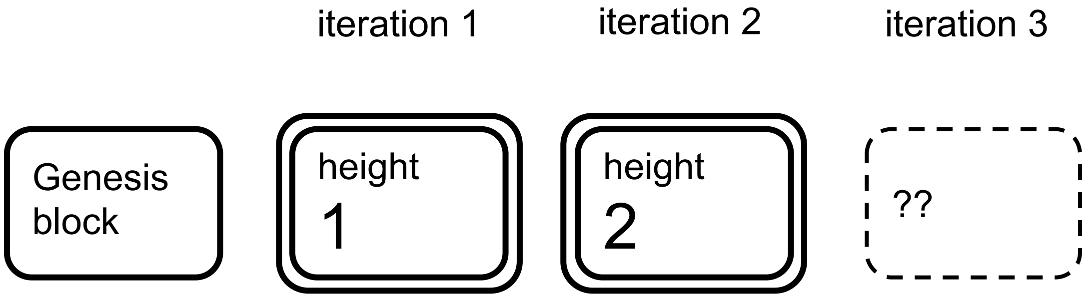
(Also, send this notarized blockchain to everyone else.)



Constructing notarized blocks

Each iteration has a leader player chosen randomly ahead of time.

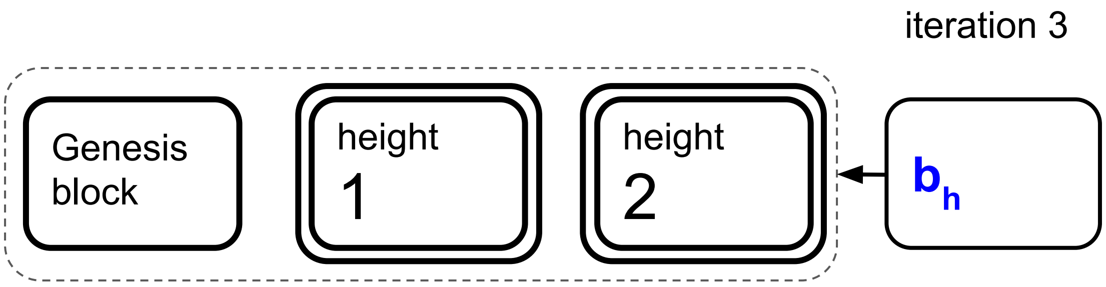
Specifically, the leader of iteration **h** = *H** **(h)** mod **n**, where *H** is a random oracle.



Constructing notarized blocks

Each player **i**, on entering iteration **h**

1. If **i** is the leader**, i** chooses notarized blockchain of length **h-1**, extends it with a new block $\mathbf{b_h}$ and sends everyone a signed message "**propose** $\mathbf{b_h}$".

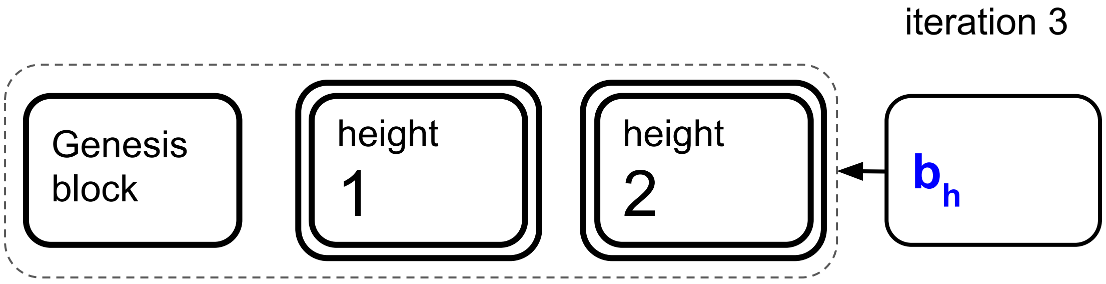


Constructing notarized blocks

Should include all pending transactions.

Each player **i**, on entering iteration **h**

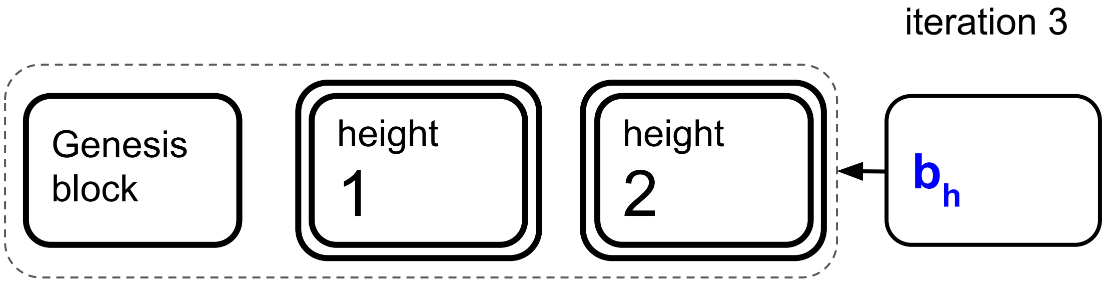
1. If **i** is the leader, **i** chooses notarized blockchain of length **h-1**, extends it with a new block b_h and sends everyone a signed message **"propose b_h"**.



Constructing notarized blocks

Each player **i**, on entering iteration **h**

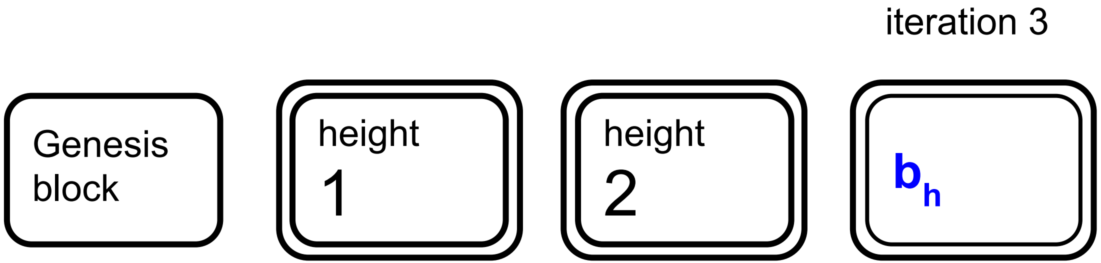
1. If **i** is the leader, **i** chooses notarized blockchain of length **h-1**, extends it with a new block b_h and sends everyone a signed message "**propose** b_h".
2. On seeing the ***first*** valid proposal from the leader, player **i** sends everyone a signed message "**vote** b_h".



Constructing notarized blocks

Each player **i**, on entering iteration **h**

1. If **i** is the leader, **i** chooses notarized blockchain of length **h-1**, extends it with a new block $\mathbf{b_h}$ and sends everyone a signed message "**propose** $\mathbf{b_h}$".
2. On seeing the ***first*** valid proposal from the leader, player **i** sends everyone a signed message "**vote** $\mathbf{b_h}$".

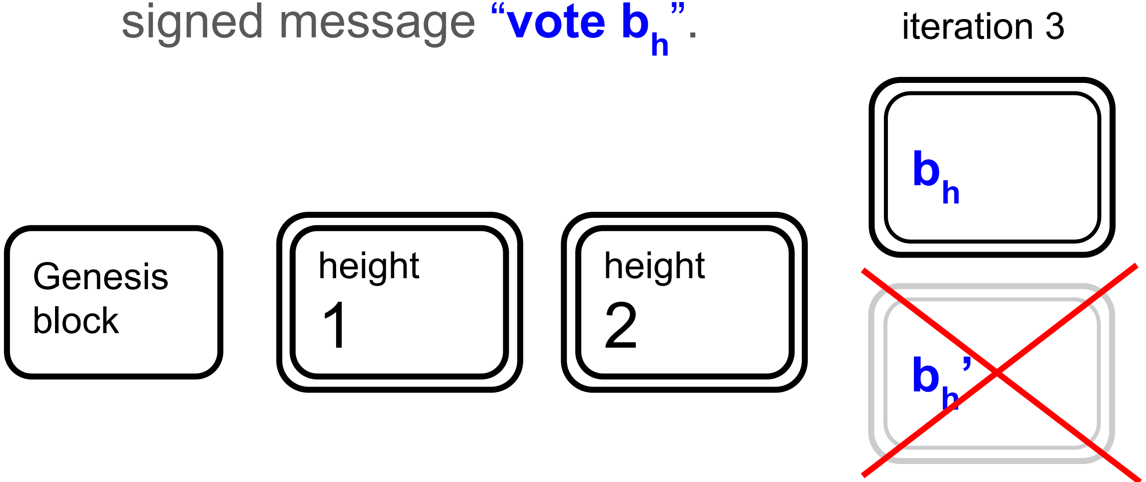


If the network is good and the leader is honest, the block proposal will get notarized!

Constructing notarized blocks

Each player **i**, on entering iteration **h**

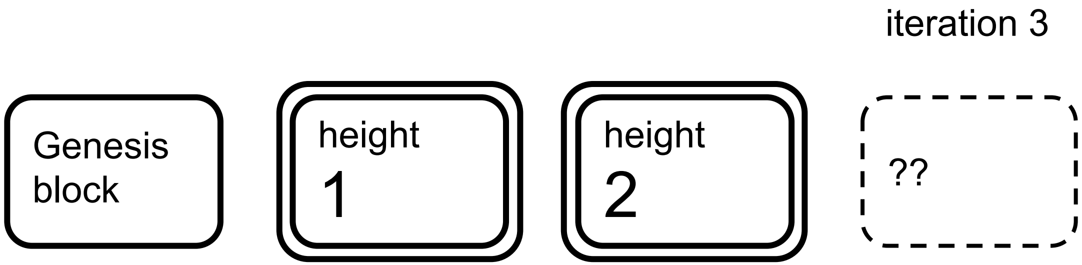
1. If **i** is the leader, **i** chooses notarized blockchain of length **h-1**, extends it with a new block b_h and sends everyone a signed message "**propose** b_h".
2. On seeing the ***first*** valid proposal from the leader, player **i** sends everyone a signed message "**vote** b_h".



At most one block proposal from the leader can be notarized in honest view

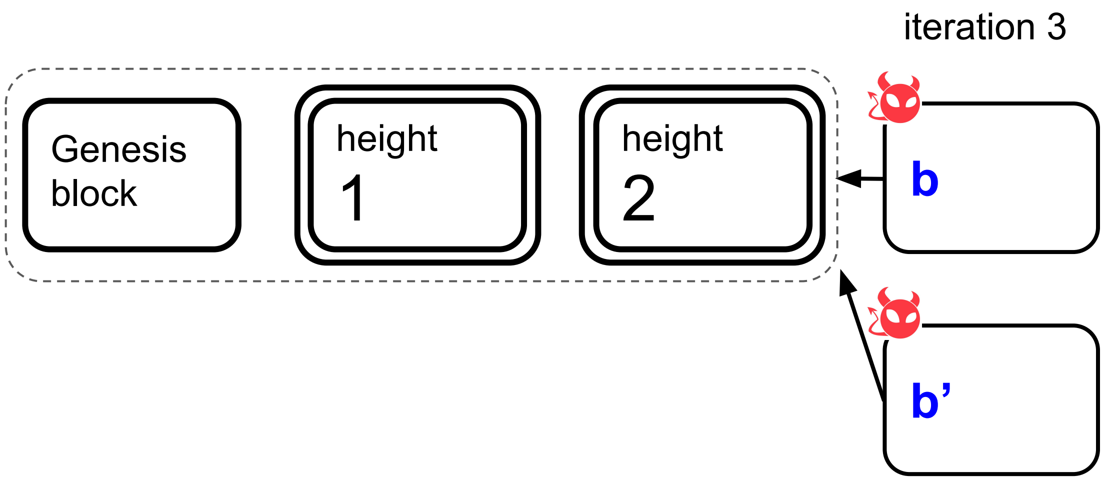
Handling faults

Scenario 1: if the network drops all messages, or leader crashed, maybe players never see a block proposal for that iteration…



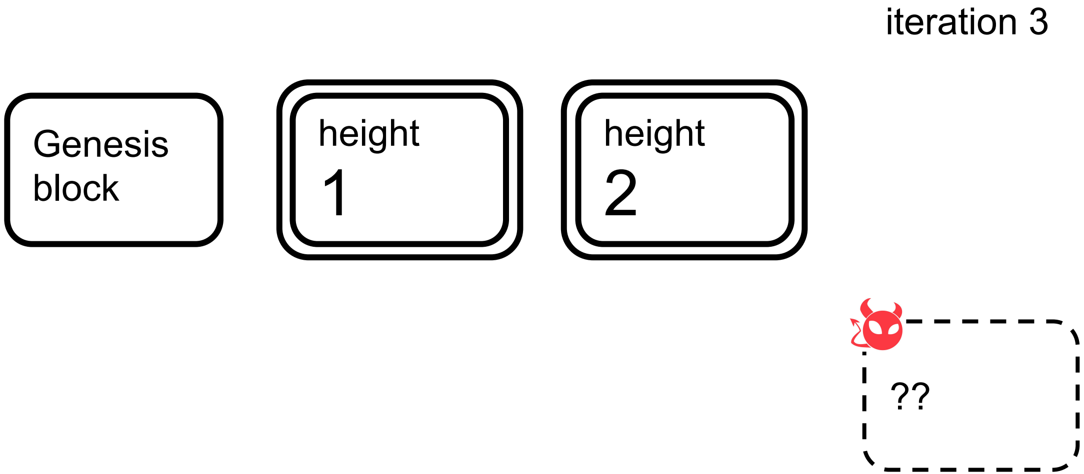
Handling faults

Scenario 2: a faulty leader sends different proposals to different players, and honest players split their vote, so no block proposal gets notarized...



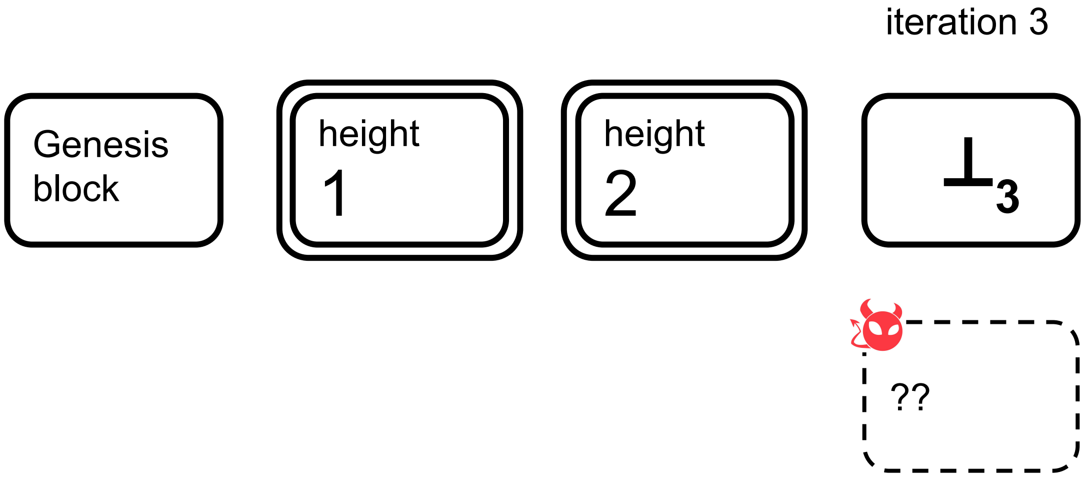
Solution: dummy blocks.

If **3Δ time** has passed since player **i** has entered iteration **h**, and if **i** still has not entered iteration **h+1**, player **i** sends to everyone a signed message "**vote $\perp_h$**".



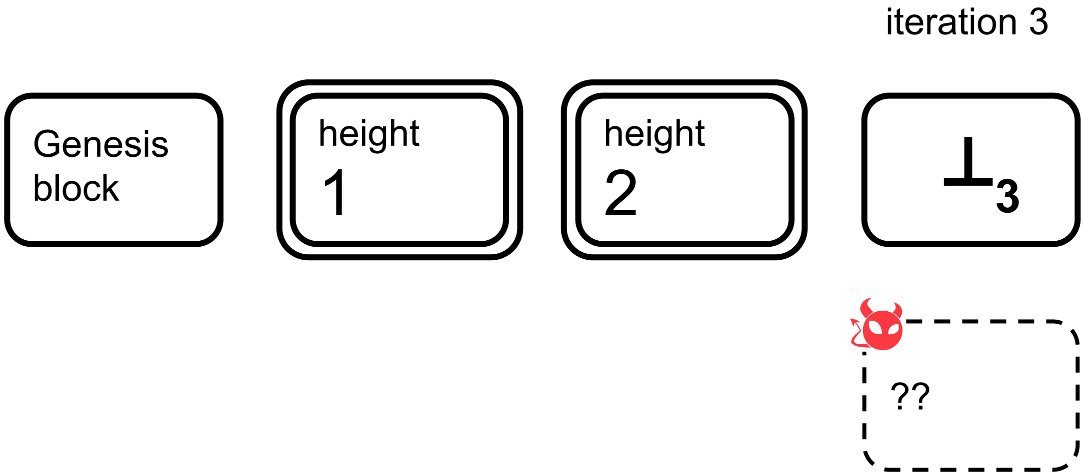
Solution: dummy blocks.

If **3Δ time** has passed since player **i** has entered iteration **h**, and if **i** still has not entered iteration **h+1**, player **i** sends to everyone a signed message "**vote $\perp_h$**".



Solution: dummy blocks.

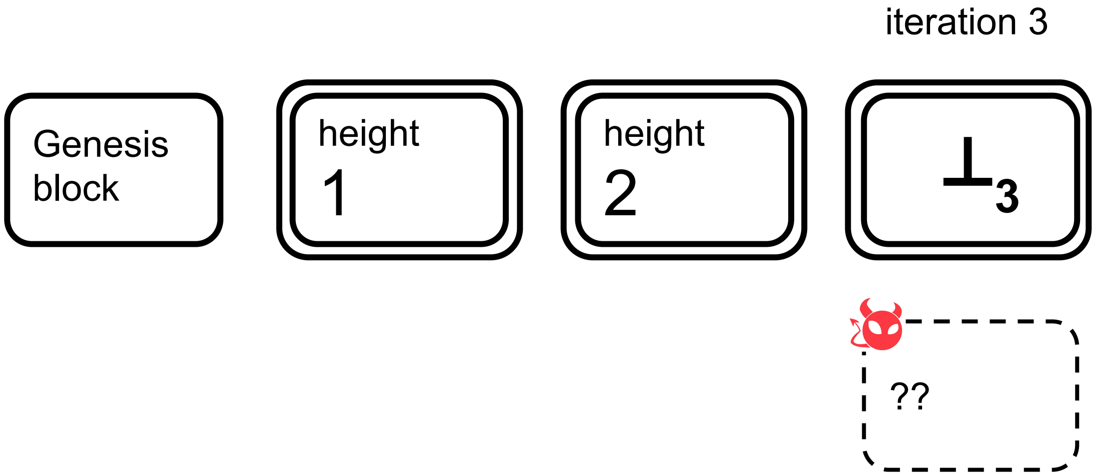
If **3Δ time** has passed since player **i** has entered iteration **h**, and if **i** still has not entered iteration **h+1**, player **i** sends to everyone a signed message "**vote $\perp_h$**".



Recall: the dummy block of height **h** is the tuple **$\perp_h$ = (h, ⊥, ⊥)**

Solution: dummy blocks.

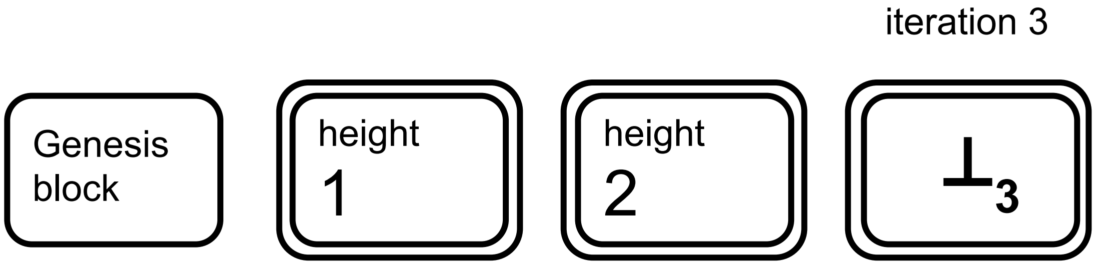
If **3Δ time** has passed since player **i** has entered iteration **h**, and if **i** still has not entered iteration **h+1**, player **i** sends to everyone a signed message “**vote** $\perp_h$”.



Recall: the dummy block of height **h** is the tuple $\perp_h = (h, \perp, \perp)$

Solution: dummy blocks.

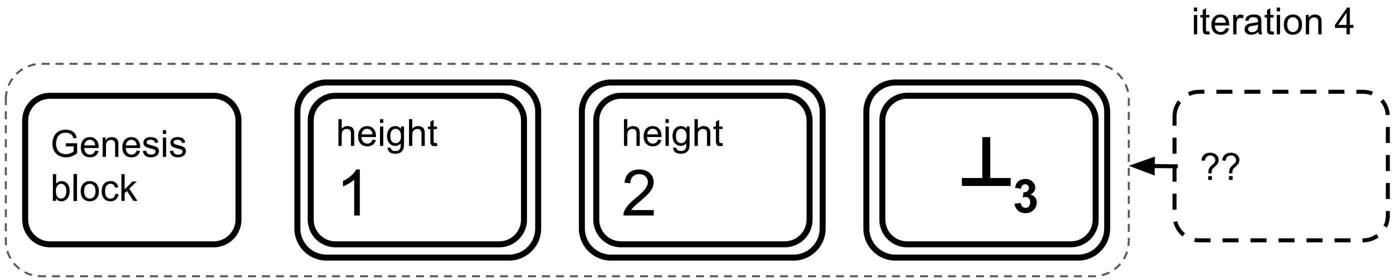
If **3Δ time** has passed since player **i** has entered iteration **h**, and if **i** still has not entered iteration **h+1**, player **i** sends to everyone a signed message "**vote $\perp_h$**".



On seeing notarized dummy block,
can now move on to the next iteration!

Solution: dummy blocks.

If **3Δ time** has passed since player **i** has entered iteration **h**, and if **i** still has not entered iteration **h+1**, player **i** sends to everyone a signed message "**vote $\perp_h$**".



On seeing notarized dummy block,
can now move on to the next iteration!

Intuition: example of an honest view

If there are faults during iteration **h**, there may be ***both***

- a notarized block proposal (for **h**), and
- a notarized dummy block $\perp_h$

in the view of honest players.

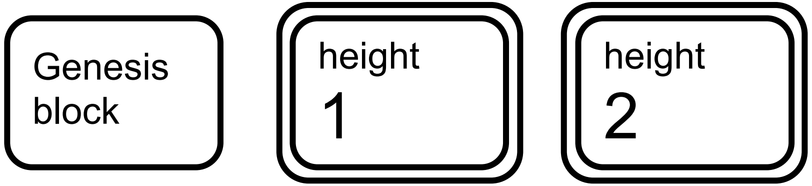
Intuition: example of an honest view

If there are faults during iteration **h**, there may be ***both***

- a notarized block proposal (for **h**), and
- a notarized dummy block $\bot_h$

in the view of honest players.

i.e. Alice sees a notarized block proposal for h=3

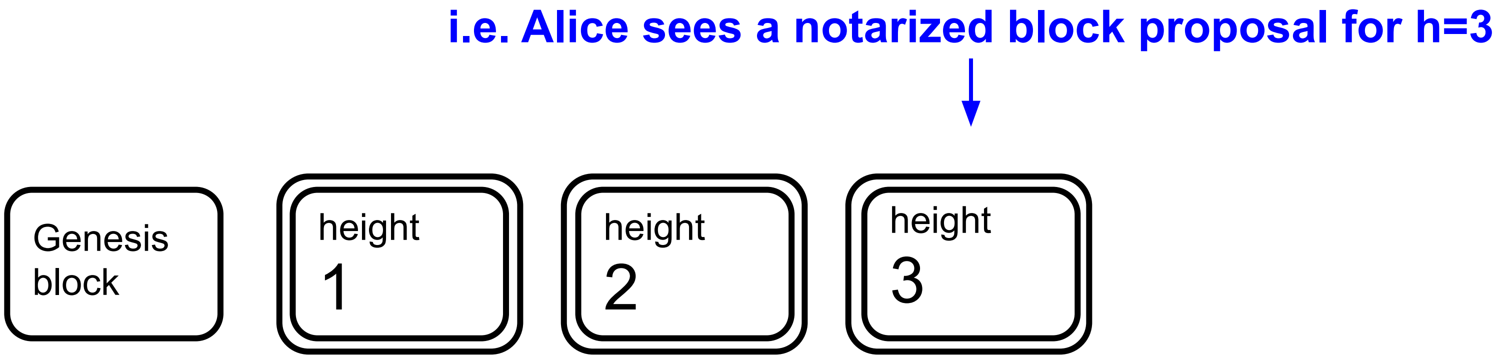


Intuition: example of an honest view

If there are faults during iteration **h**, there may be ***both***

- a notarized block proposal (for **h**), and
- a notarized dummy block $\bot_h$

in the view of honest players.

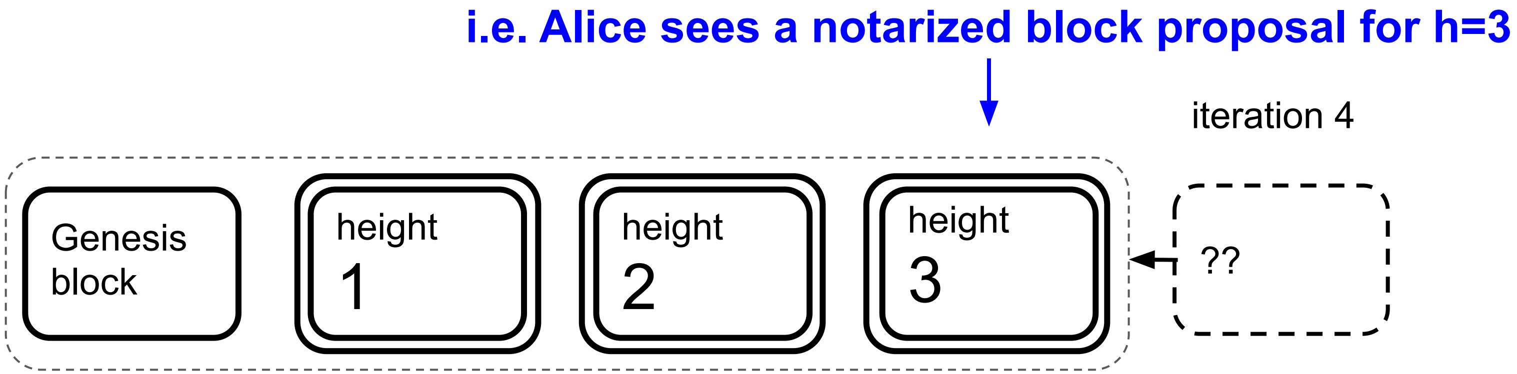


Intuition: example of an honest view

If there are faults during iteration **h**, there may be ***both***

- a notarized block proposal (for **h**), and
- a notarized dummy block $\perp_h$

in the view of honest players.



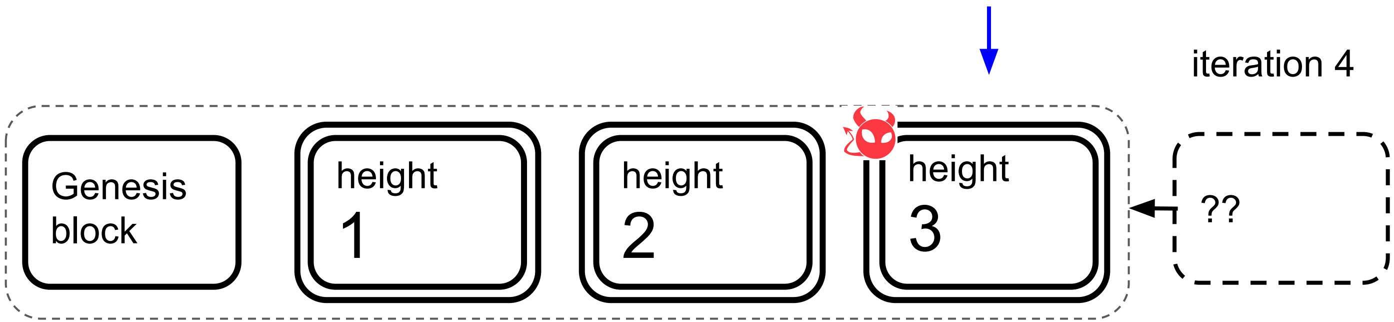
Intuition: example of an honest view

If there are faults during iteration **h**, there may be ***both***

- a notarized block proposal (for **h**), and
- a notarized dummy block $\bot_h$

in the view of honest players.

i.e. Alice sees a notarized block proposal for h=3



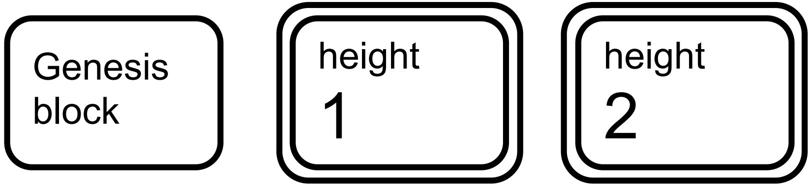
Intuition: example of an honest view

If there are faults during iteration **h**, there may be ***both***

- a notarized block proposal (for **h**), and
- a notarized dummy block $\perp_h$

in the view of honest players.

but everyone else times out
(and votes for $\perp_3$)



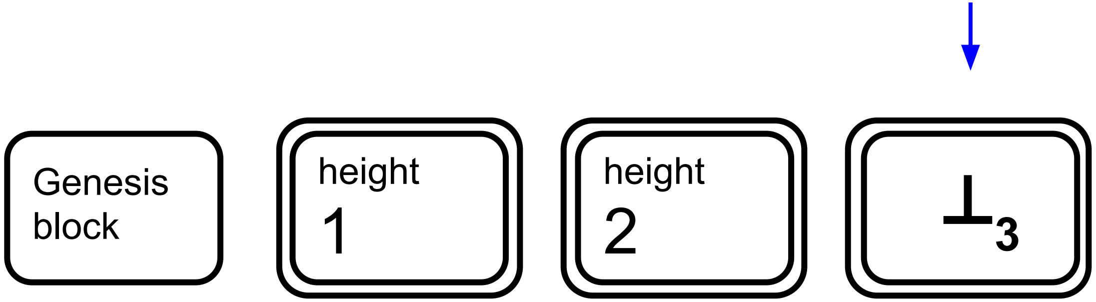
Intuition: example of an honest view

If there are faults during iteration **h**, there may be ***both***

- a notarized block proposal (for **h**), and
- a notarized dummy block $\bot_h$

in the view of honest players.

so Bob sees a notarized dummy block $\bot_3$

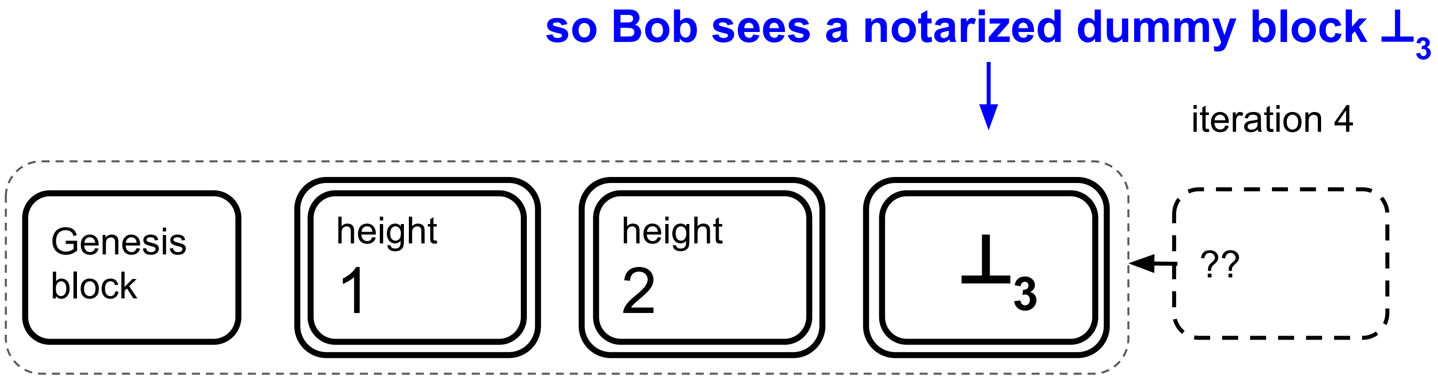


Intuition: example of an honest view

If there are faults during iteration **h**, there may be ***both***

- a notarized block proposal (for **h**), and
- a notarized dummy block $\perp_h$

in the view of honest players.

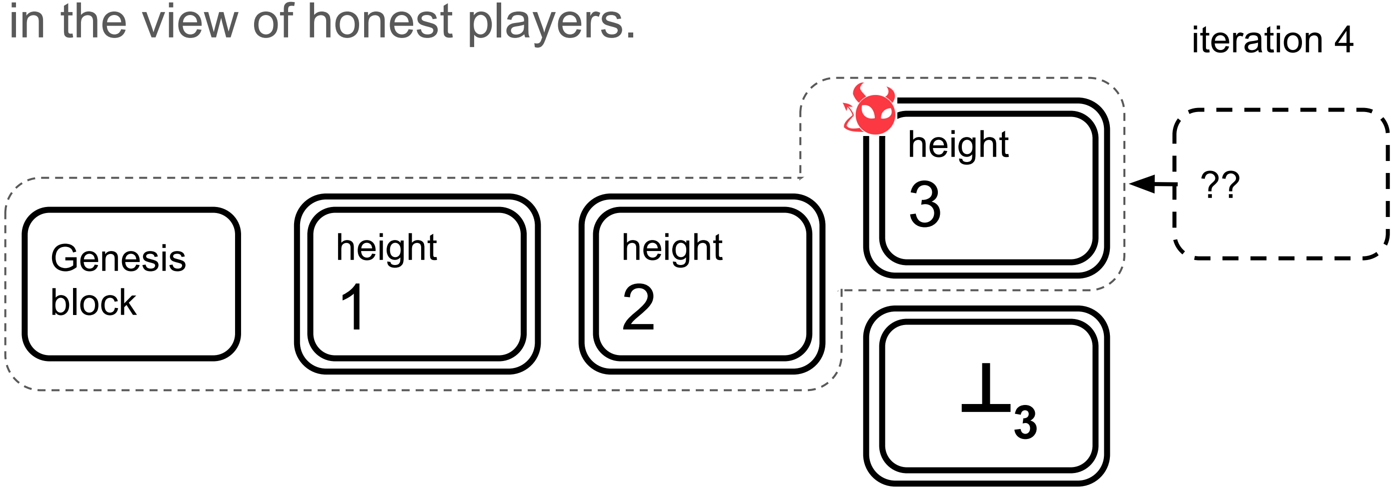


Intuition: example of an honest view

If there are faults during iteration **h**, there may be ***both***

- a notarized block proposal (for **h**), and
- a notarized dummy block $\bot_h$

in the view of honest players.

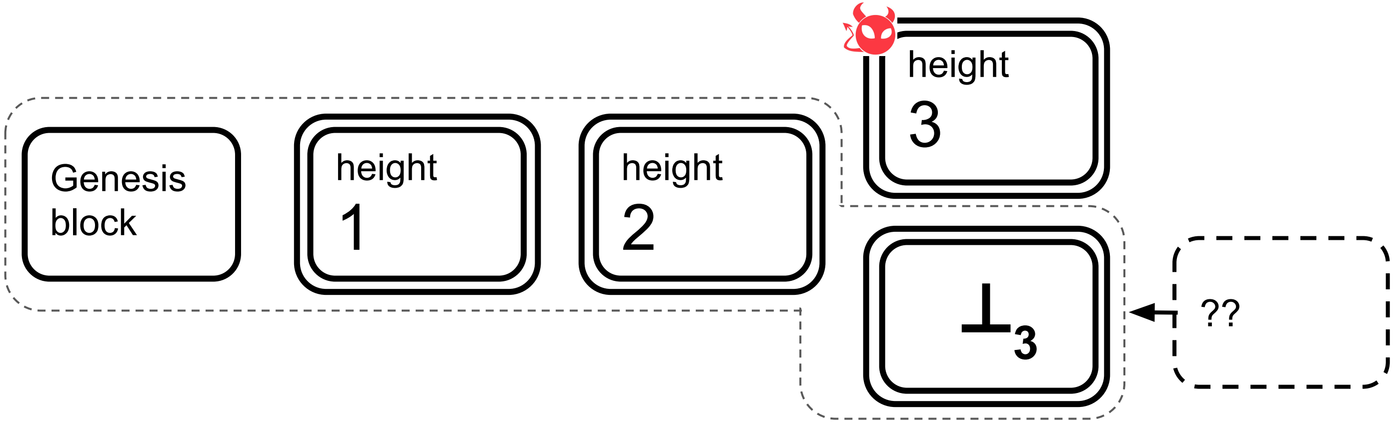


Intuition: example of an honest view

If there are faults during iteration **h**, there may be ***both***

- a notarized block proposal (for **h**), and
- a notarized dummy block $\perp_h$

in the view of honest players.



Finalizing blocks

When player **i** enters iteration **h+1**, if **i** did not time out and vote for the dummy block for **h**, player **i** sends everyone a signed "**finalize h**" message.

Finalizing blocks

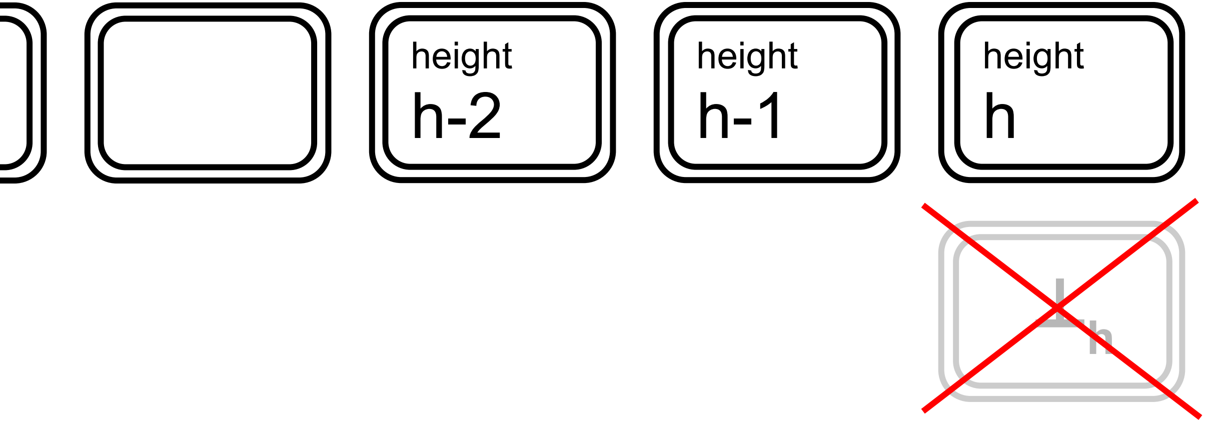
When player **i** enters iteration **h+1**, if **i** did not time out and vote for the dummy block for **h**, player **i** sends everyone a signed "**finalize h**" message.

On seeing **2n/3** "**finalize h**" messages, a player **i** finalizes any notarized blockchain of length **h** that it sees.

Finalizing blocks

When player **i** enters iteration **h+1**, if **i** did not time out and vote for the dummy block for **h**, player **i** sends everyone a signed "**finalize h**" message.

On seeing **2n/3** "**finalize h**" messages, a player **i** finalizes any notarized blockchain of length **h** that it sees.

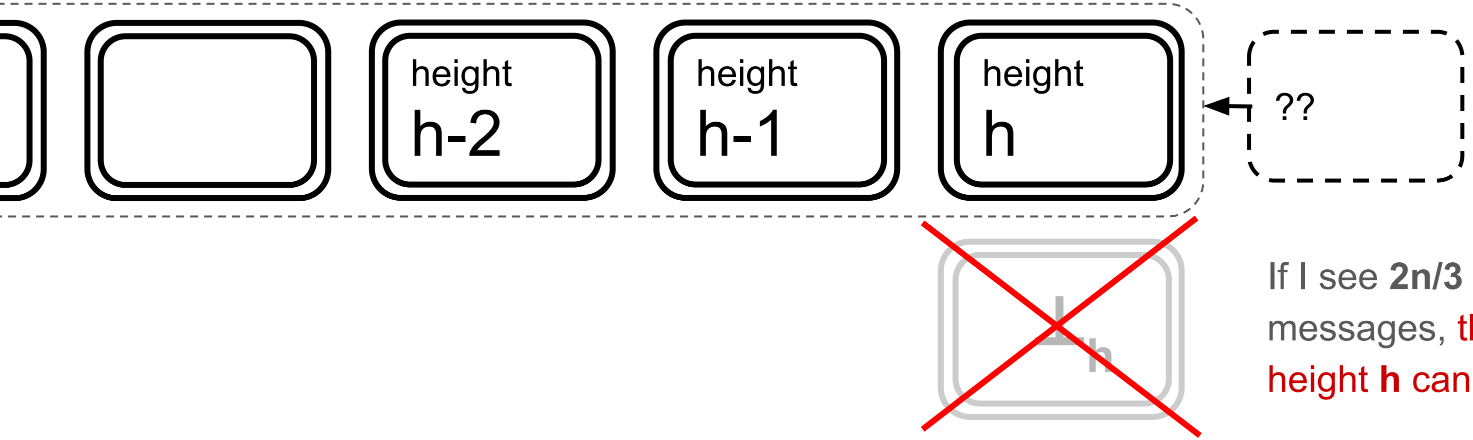


If I see **2n/3** "**finalize h**" messages, the dummy block of height **h** cannot be notarized!

Finalizing blocks

When player **i** enters iteration **h+1**, if **i** did not time out and vote for the dummy block for **h**, player **i** sends everyone a signed “**finalize h**” message.

On seeing **2n/3** “**finalize h**” messages, a player **i** finalizes any notarized blockchain of length **h** that it sees.



If I see **2n/3 “finalize h”** messages, the dummy block of height **h** cannot be notarized!

Protocol Summary

In each iteration **h = 1, 2, 3, …** each player does the following:

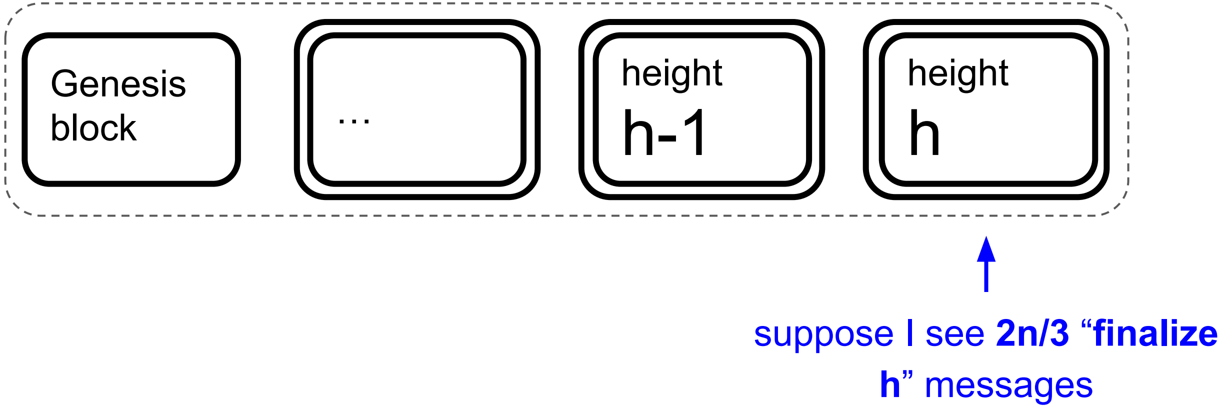
1. The leader proposes a new block of height **h** extending a notarized blockchain of length **h-1**.
2. On seeing the first valid block proposal **b** from the leader, send everyone "**vote b**".
3. (Timeout) After **3Δ time**, if we are still in iteration **h**, send everyone "**vote** $\bot_h$".
4. On seeing a notarized blockchain of length **h**, enter iteration **h+1**. If we did not previously timeout, send everyone "**finalize h**".

At any point, in any iteration

5. On seeing **2n/3 finalize** messages for any **h**, we can finalize any notarized blockchain of length **h**.

Consistency

Thm: Consider two finalized chains LOG, LOG' s.t |LOG| ≤|LOG'|.
Then, LOG ≤ LOG'

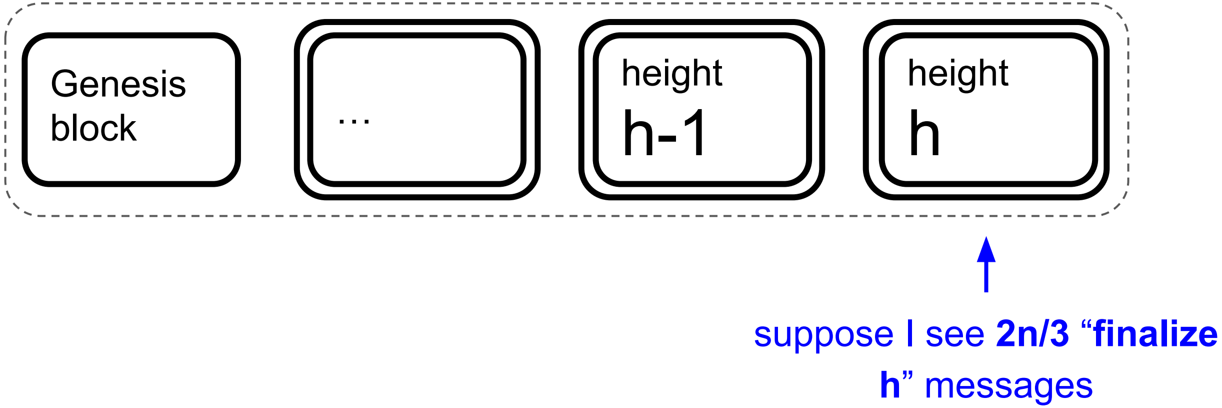


Proof: Consider the shorter one: LOG, let its length be **h**

Consistency

Since LOG is finalized, some honest player sees **2n/3** "**finalize h**" messages.

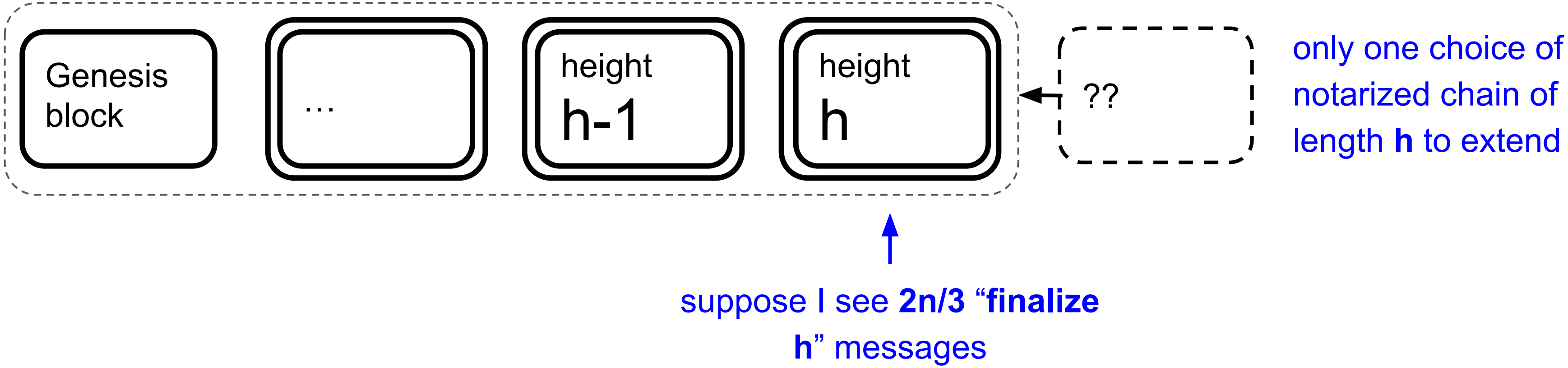
Claim: there can be only one notarized blockchain of length **h**, across all honest views



Consistency

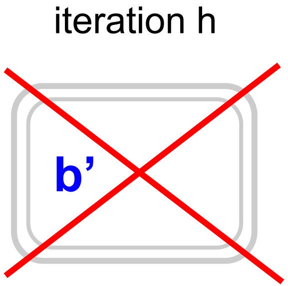
Since LOG is finalized, some honest player sees **2n/3 "finalize h"** messages.

Claim: there can be only one notarized blockchain of length **h**, across all honest views

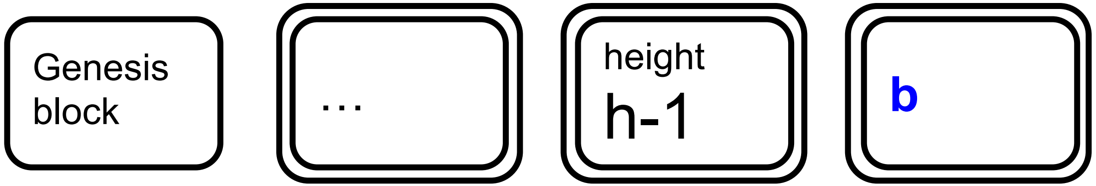


Consistency

Claim: At most one block proposal from the leader can be notarized in honest view

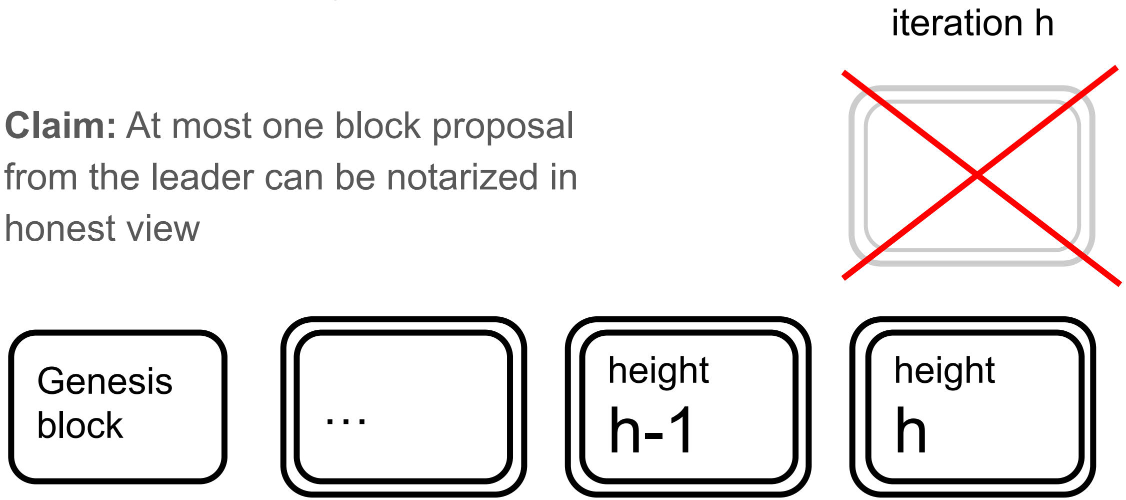


Proof: Each honest player votes for at most one proposal. Quorum intersection.



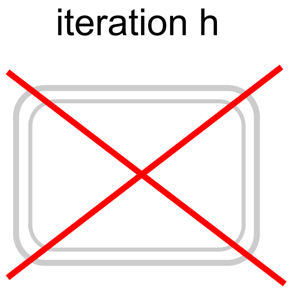
Consistency

Claim: At most one block proposal from the leader can be notarized in honest view



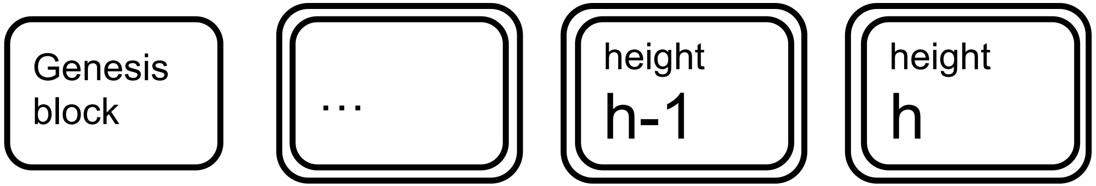
Proof: Each honest player votes for at most one proposal. Quorum intersection.

Consistency

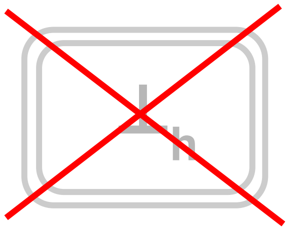


Claim: At most one block proposal from the leader can be notarized in honest view

Proof: Each honest player votes for at most one proposal. Quorum intersection.



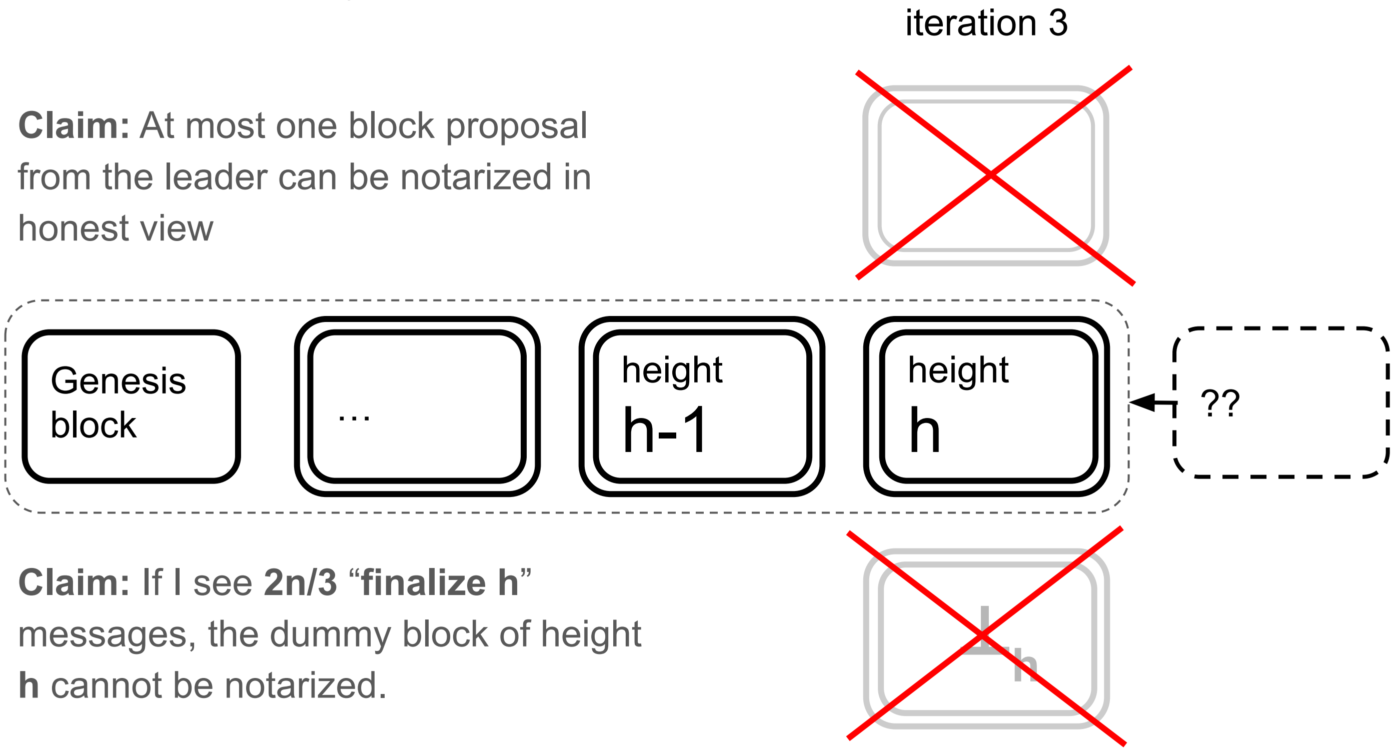
Claim: If I see **2n/3 "finalize h"** messages, the dummy block of height **h** cannot be notarized.



Proof: Each honest player either votes **finalize** or for $\perp_h$. Apply quorum intersection.

Consistency

Claim: At most one block proposal from the leader can be notarized in honest view

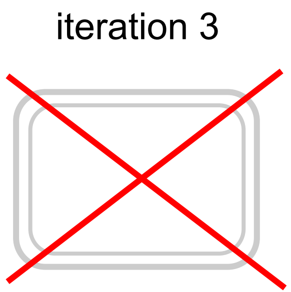


Claim: If I see **2n/3 "finalize h"** messages, the dummy block of height **h** cannot be notarized.

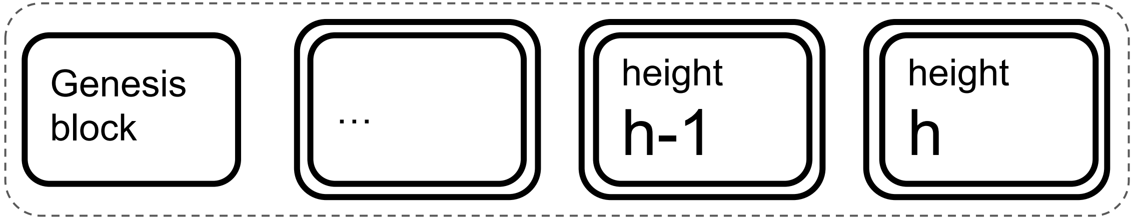
Thus, if someone sees **2n/3 "finalize h"** messages: only one choice of notarized chain of length **h** to extend

LOG ≤ LOG'

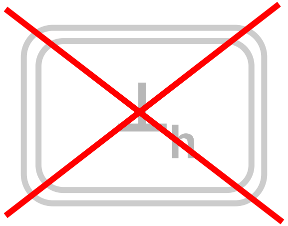
Consistency



Claim: At most one block proposal from the leader can be notarized in honest view



Safe to finalize the transactions in this notarized chain!



Claim: If I see **2n/3** “**finalize h**” messages, the dummy block of height **h** cannot be notarized.

Liveness

Claim: if the network is good (after GST), an honest leader can always get its block proposal notarized, and then finalized.

Liveness

Claim: if the network is good (after GST), an honest leader can always get its block proposal notarized, and then finalized.

Fact: if some honest player enters iteration h by time t, if $t > GST$, then every honest player enters iteration h by time $t + \delta$.

When an honest player enters an iteration h, it sends its notarized blockchain of length $h-1$ to everyone else.

Liveness

Claim: if the network is good (after GST), an honest leader can always get its block proposal notarized, and then finalized.

Liveness

Subclaim 1: every honest node will see a notarization for some block of height h by time $t + 2\delta$.

Liveness

Subclaim 1: every honest node will see a notarization for some block of height $\boldsymbol{h}$ by time $\boldsymbol{t + 2\delta}$.

Leader enters iteration $\boldsymbol{h}$ and proposes a new block $\mathbf{b_h}$ extending a notarized chain $\mathbf{b_1} \ldots \mathbf{b_{h-1}}$.

Every honest player enters iteration $\boldsymbol{h}$ and sees the proposal.

Either everyone sends "**vote** $\mathbf{b_h}$", or someone already entered iteration **h+1**.

Liveness

Subclaim 1: every honest node will see a notarization for some block of height ***h*** by time ***t* + 2*δ***.

time *t*

Leader enters iteration ***h*** and proposes a new block $\mathbf{b_h}$ extending a notarized chain $\mathbf{b_1 \ldots b_{h-1}}$.

time *t* + *δ*

Every honest player enters iteration ***h*** and sees the proposal.

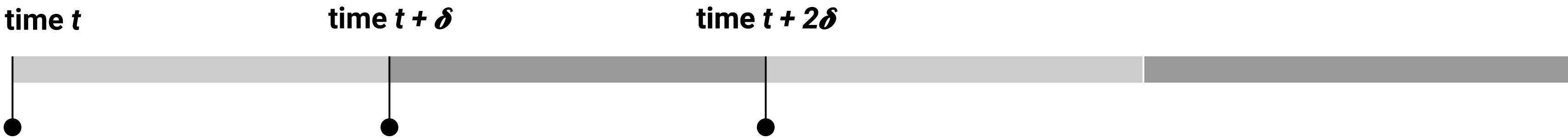
Either everyone sends "**vote** $\mathbf{b_h}$", or someone already entered iteration **h+1**.

time *t* + 2*δ*

Every honest player sees some notarized block of height ***h***.

Liveness

Subclaim 2: The dummy block of height ***h*** (denoted $\perp_h$) cannot be notarized in any honest view before time ***t + 2δ***.



Leader enters iteration ***h*** and proposes a new block $\mathbf{b_h}$ extending a notarized chain $\mathbf{b_1} \dots \mathbf{b_{h-1}}$.

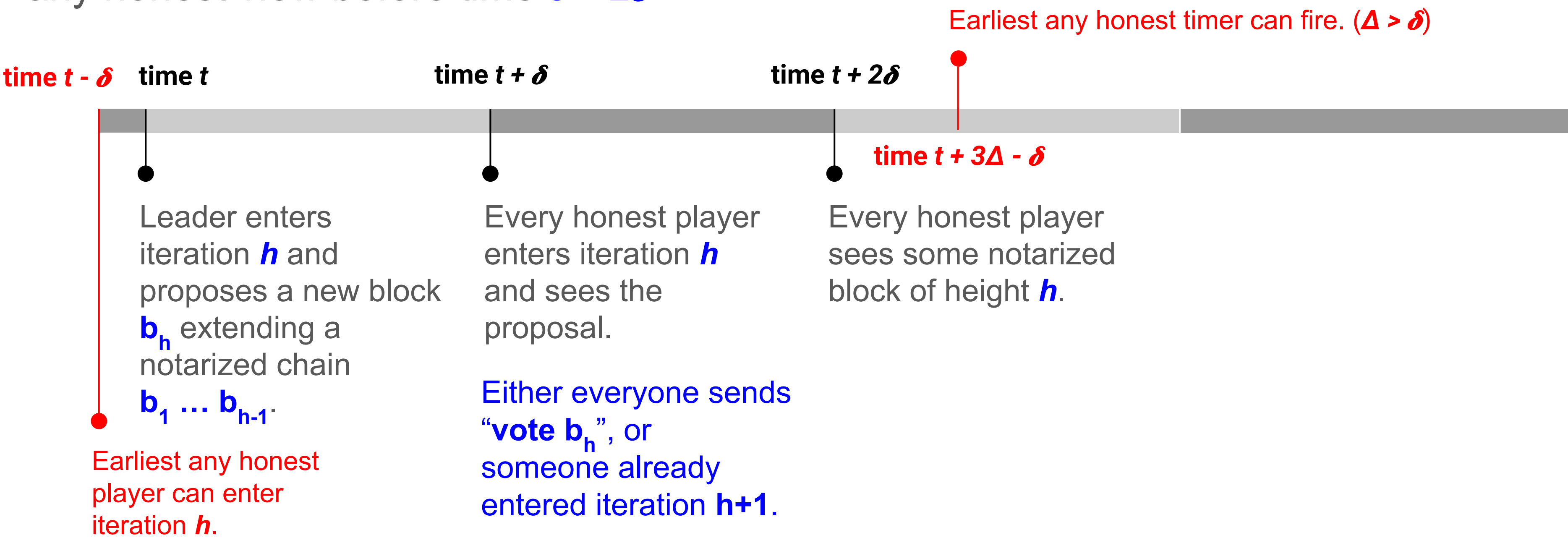
Every honest player enters iteration ***h*** and sees the proposal.

Either everyone sends "**vote** $\mathbf{b_h}$", or someone already entered iteration **h+1**.

Every honest player sees some notarized block of height ***h***.

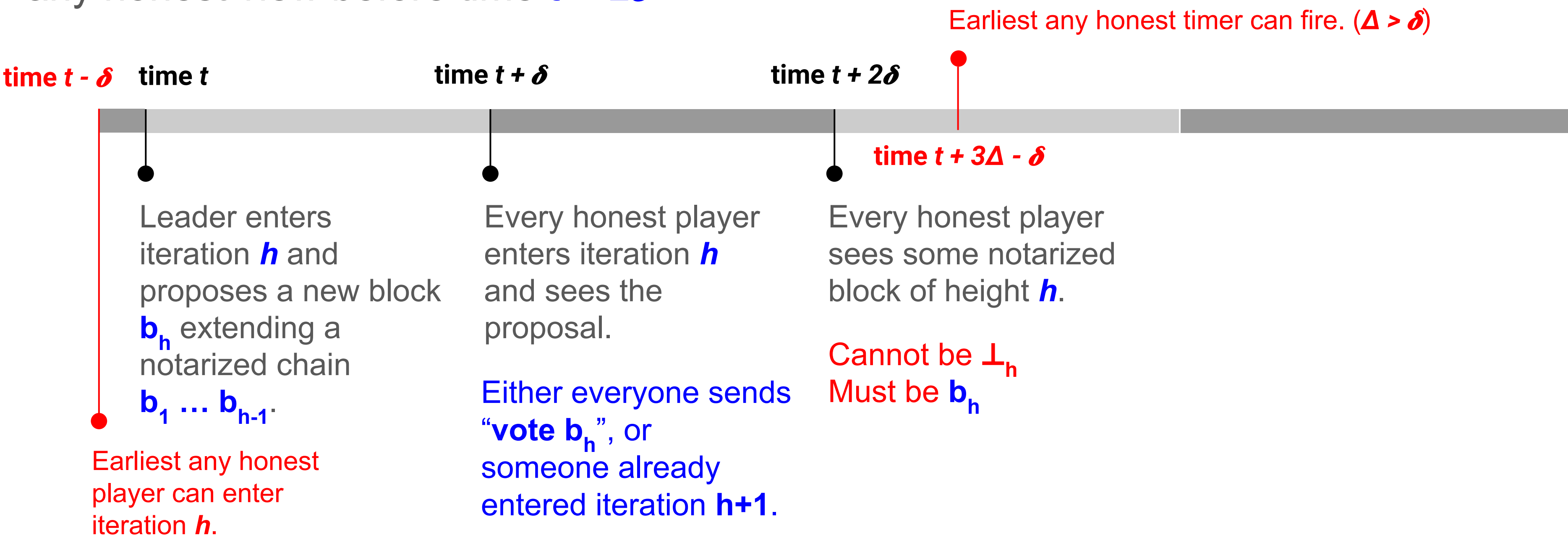
Liveness

Subclaim 2: The dummy block of height ***h*** (denoted $\perp_h$) cannot be notarized in any honest view before time $\boldsymbol{t + 2\delta}$.



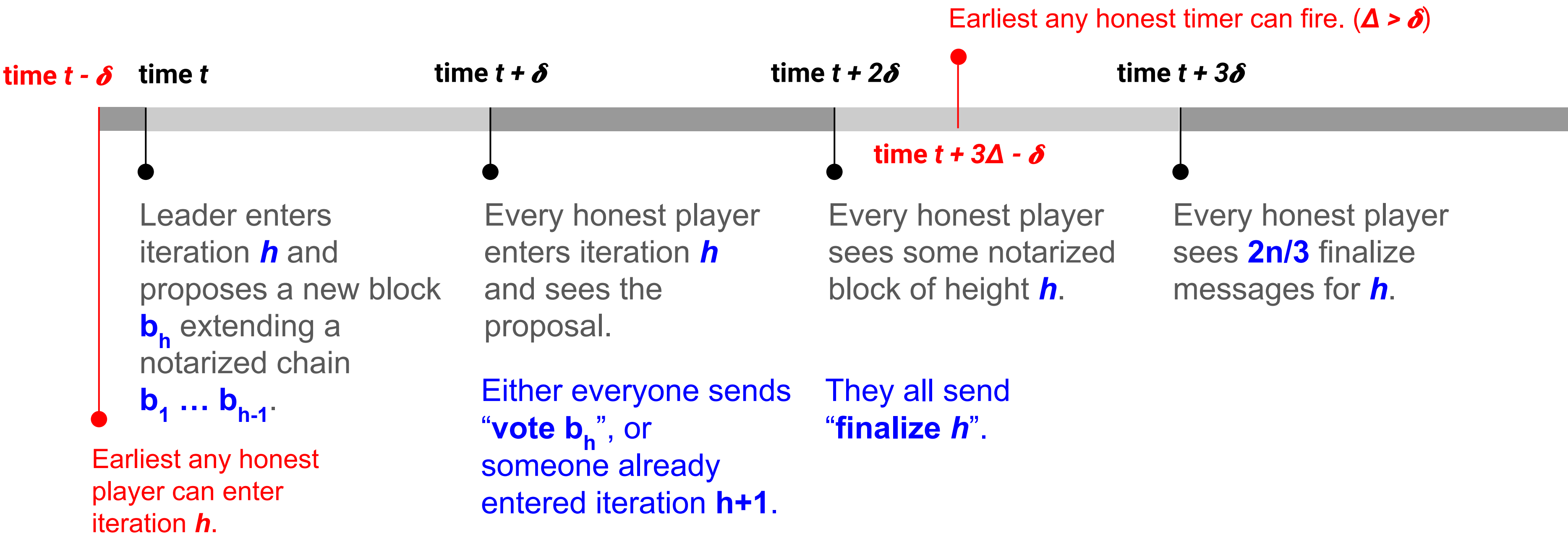
Liveness

Subclaim 2: The dummy block of height ***h*** (denoted $\bot_h$) cannot be notarized in any honest view before time $\boldsymbol{t + 2\delta}$.



Liveness

Thus, every honest player finalizes the leader's block proposal by time $t + 3\delta$.



Liveness for faulty leaders

Claim: if the network is good (after GST), **any** iteration will conclude after $\mathbf{3\Delta + \delta}$ time.

Liveness for faulty leaders

Claim: if the network is good (after GST), **any** iteration will conclude after **$3\Delta + \delta$** time.

Every honest player has entered iteration ***h***.

Either every honest timer for iteration ***h*** has fired, or some honest process entered iteration ***h+1*** already.

If timer fires, multicast "**vote $\bot_h$**".

Liveness for faulty leaders

Claim: if the network is good (after GST), **any** iteration will conclude after $3\Delta + \delta$ time.

time *t*

Every honest player has entered iteration ***h***.

time $t + 3\Delta$

Either every honest timer for iteration ***h*** has fired, or some honest process entered iteration ***h+1*** already.

If timer fires, multicast "**vote $\bot_h$**".

time $t + 3\Delta + \delta$

Every honest player enters iteration ***h+1***.

Expected Liveness

Claim: Suppose that every honest player sees TX before iteration ***h***. Suppose every honest player enters iteration **h** by time **t**. Then TX is in the output of every honest player by time $\mathbf{t + 3.5\delta + 1.5\Delta}$, in expectation.

Proof: In expectation, it takes 3/2 iterations to get an iteration with an honest leader. Thus, in expectation the number of iterations with faulty leaders is 1/2. Thus, the waiting time is at most

$$1/2 \cdot (3\Delta + \delta) + 3\delta$$
$$= 3.5\delta + 1.5\Delta$$

as desired.

In Conclusion

A new consensus protocol, called **Simplex Consensus**

- Partial synchrony, **$f < n/3$** byzantine faults
- In our eyes, easiest security proofs!
- Can get communication efficiency using “sortition” [Algorand]

Thm: Assuming a (Bare) PKI, CRH, there exists a partially synchronous consensus protocol in the “random-leader model” with:

- Proposal confirmation time of **3δ**
- Optimistic block time of **2δ**
- Expected pessimistic liveness of **$3.5\delta + 1.5\Delta$**
- Worst-case liveness of **$4\delta + \omega(\log \lambda) \cdot (3\Delta + \delta)$**

What Next?

Work on understandable, efficient permissioned consensus

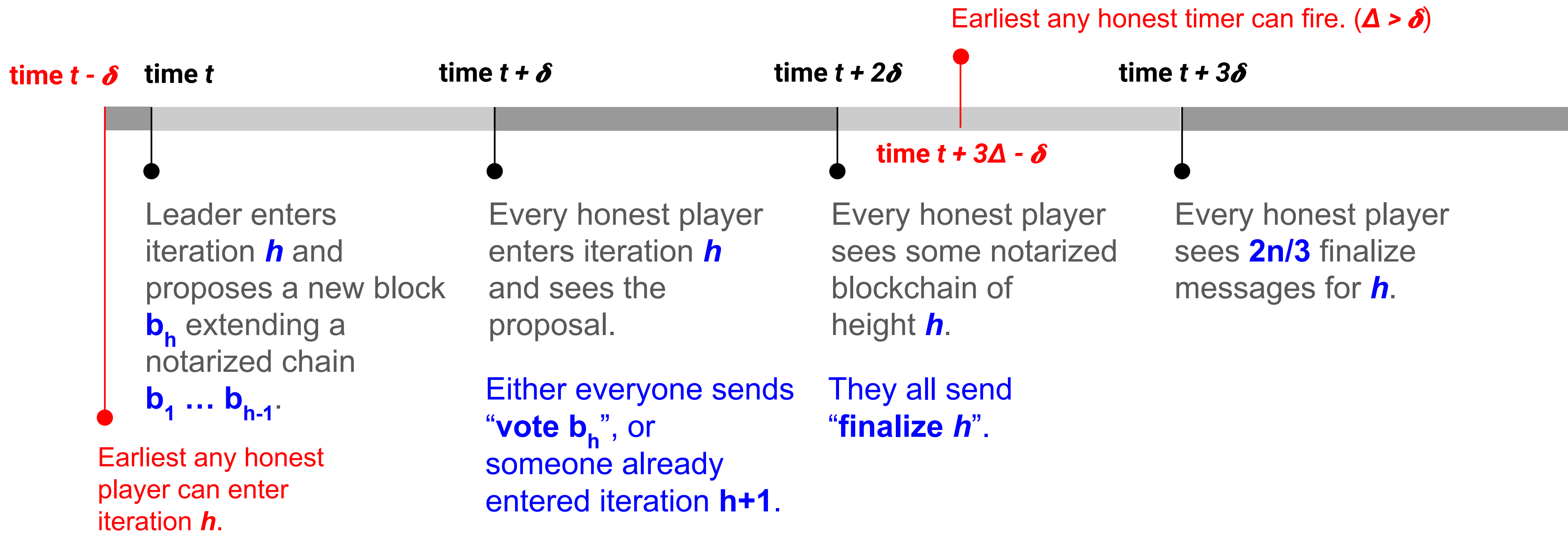
- Simplex [**C**P23], Streamlet [**C**S20]

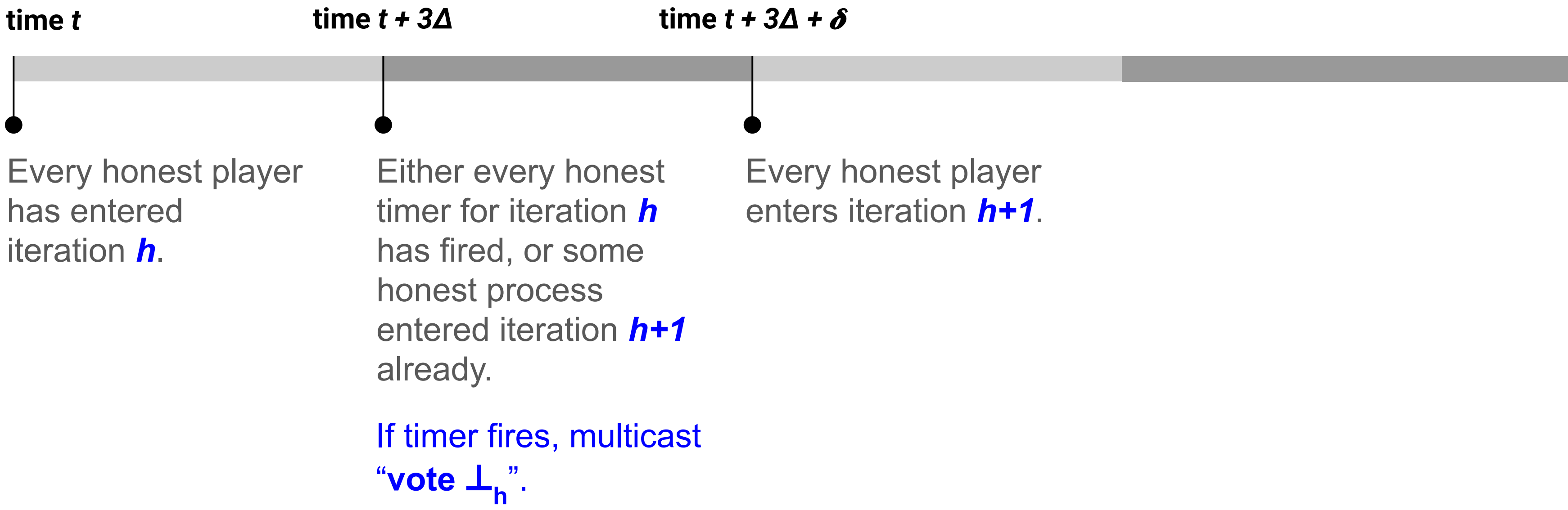
Work on formalizing execution environments of protocols in the presence of various adversaries:

- Universal Reductions [**C**FP22]
- Non-equivocation in Distributed Protocols [B**C**S22]

Next

- The permissionless setting, dynamic participation
- Decentralized exchanges


Earliest any honest timer can fire. ($\Delta > \delta$)
time $t - \delta$
time t
time $t + \delta$
time $t + 2\delta$
time $t + 3\delta$
time $t + 3\Delta - \delta$
Leader enters iteration h and proposes a new block b_h extending a notarized chain $b_1 \dots b_{h-1}$.
Earliest any honest player can enter iteration h.
Every honest player enters iteration h and sees the proposal.
Either everyone sends "vote b_h", or someone already entered iteration **h+1**.
Every honest player sees some notarized blockchain of height h.
They all send "**finalize** h".
Every honest player sees **2n/3** finalize messages for h.


time t
time $t + 3\Delta$
time $t + 3\Delta + \delta$
Every honest player has entered iteration ***h***.
Either every honest timer for iteration ***h*** has fired, or some honest process entered iteration ***h+1*** already.
If timer fires, multicast "**vote $\perp_h$**".
Every honest player enters iteration ***h+1***.