Simplex Consensus
A Fast and Simple Consensus Protocol

Benjamin Chan

Cornell Tech

Joint work with Rafael Pass

This talk:
A consensus protocol with the
easiest security proofs*
best latency
of all the protocols we've seen thus far.
(partial synchronous, $f < n/3$, PKI)

This talk:

A consensus protocol with the

easiest security proofs*

best latency

of all the protocols we’ve seen thus far.

(partial synchronous, $f < n/3$, PKI)

*subjective

Key primitive for:
Blockchains (Proof-of-Stake)
MPC (broadcast)

This talk:

A consensus protocol with the

easiest security proofs*

best latency

of all the protocols we've seen thus far.

(partial synchronous, $f < n/3$, PKI)

*subjective

Key primitive for:
Blockchains (Proof-of-Stake)
MPC (broadcast)

Very well-studied… but to this day, protocols have non-trivial liveness proofs/hard to intuit*

This talk:

A consensus protocol with the
easiest security proofs*
best latency
of all the protocols we've seen thus far.
(partial synchronous, $f < n/3$, PKI)

*subjective

Key primitive for:
Blockchains (Proof-of-Stake)
MPC (broadcast)

Very well-studied… but to this day, protocols have non-trivial liveness proofs/hard to intuit*

This talk:

A consensus protocol with the
easiest security proofs*
best latency
of all the protocols we've seen thus far.
(partial synchronous, $f < n/3$, PKI)

Faster is better

*subjective

Key primitive for:
Blockchains (Proof-of-Stake)
MPC (broadcast)

Very well-studied… but to this day, protocols have non-trivial liveness proofs/hard to intuit*

This talk:

A consensus protocol with the
easiest security proofs*
best latency
of all the protocols we've seen thus far.
(partial synchronous, $f < n/3$, PKI)

Faster is better

Good communication complexity

*subjective

Key primitive for:
Blockchains (Proof-of-Stake)
MPC (broadcast)

Very well-studied… but to this day, protocols have non-trivial liveness proofs/hard to intuit*

This talk:

A consensus protocol with the
easiest security proofs*
best latency
of all the protocols we've seen thus far.
(partial synchronous, $f < n/3$, PKI)

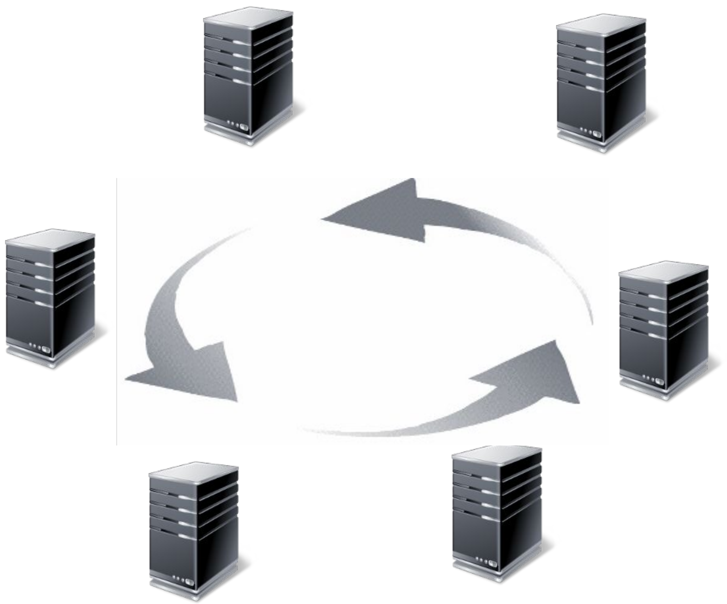
Faster is better

Good communication complexity

Goal: A practical protocol that is intuitive* enough to teach in class. (I will try to teach it now.)

*subjective

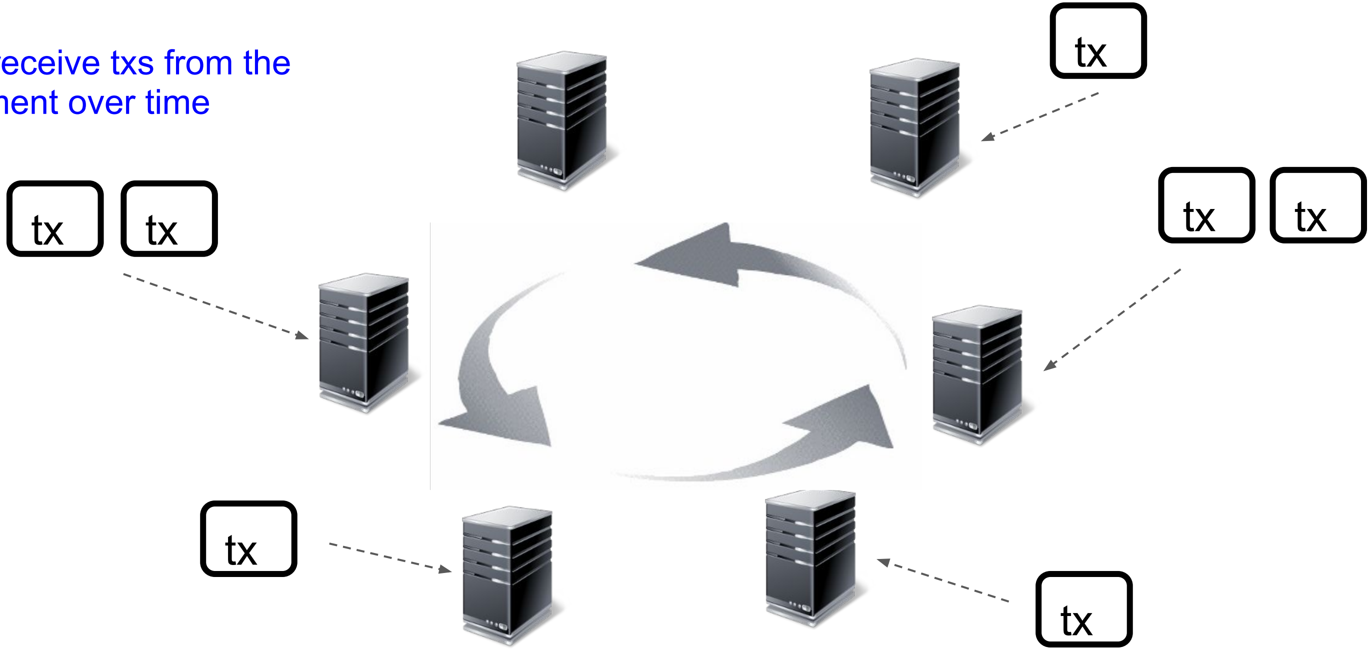
Our setting: the consensus problem



n players, **f < n/3** malicious faults, **partial synchronous** network.
know public keys ahead of time (bare PKI)

Our setting: the consensus problem

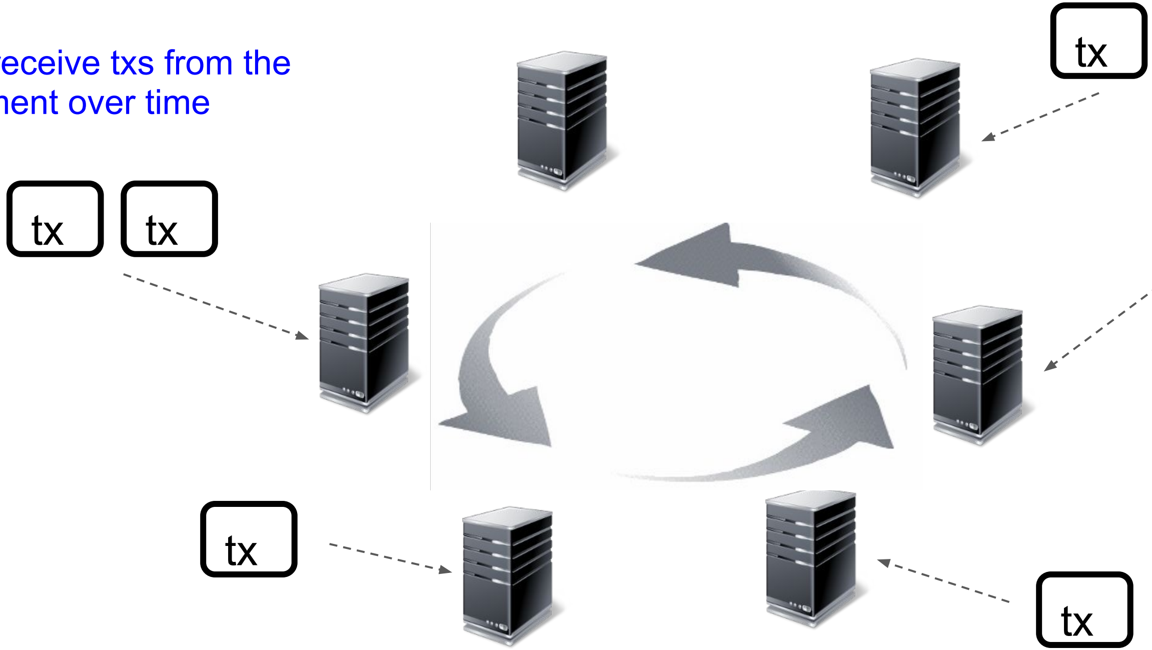
1. players receive txs from the environment over time



n players, **f < n/3** malicious faults, **partial synchronous** network.
know public keys ahead of time (bare PKI)

Our setting: the consensus problem

1. players receive txs from the environment over time



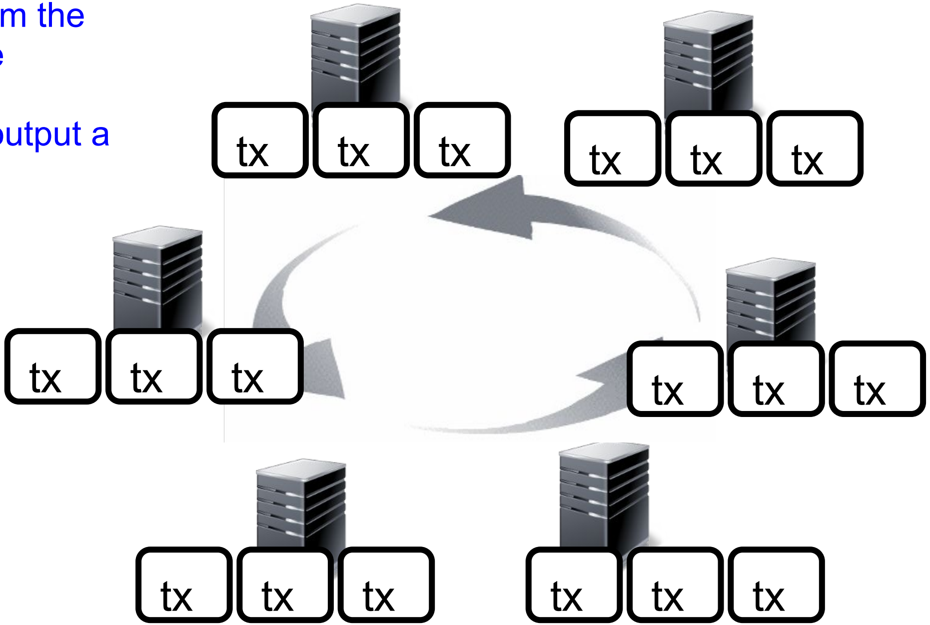
Different players may see different orders of txs!

n players, **f < n/3** malicious faults, **partial synchronous** network.
know public keys ahead of time (bare PKI)

Our setting: the consensus problem

1. players receive txs from the environment over time

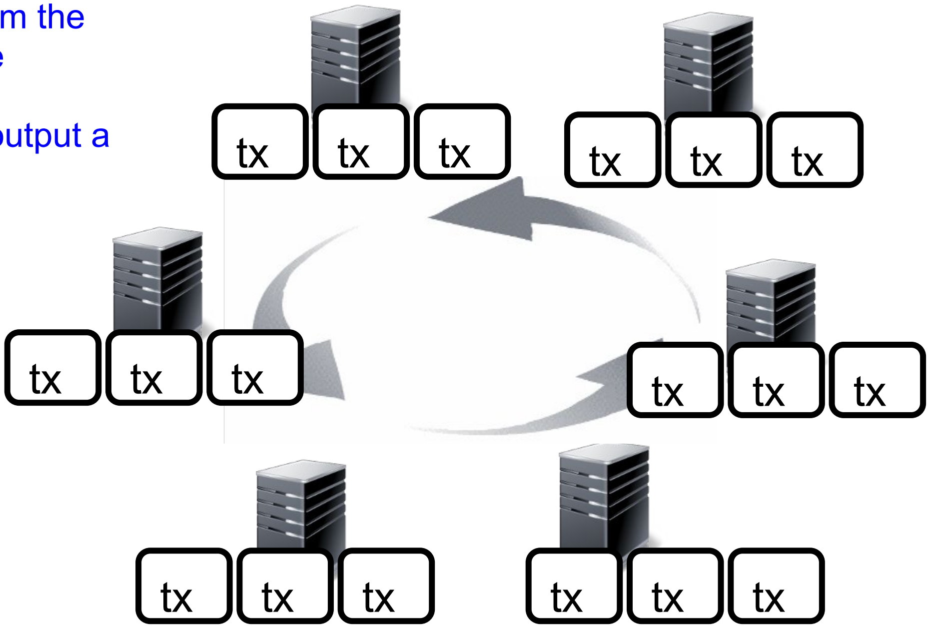
2. players continuously output a log of “finalized txs”



n players, **f < n/3** malicious faults, **partial synchronous** network.
know public keys ahead of time (bare PKI)

Our setting: the consensus problem

1. players receive txs from the environment over time
2. players continuously output a log of “finalized txs”



Consistency

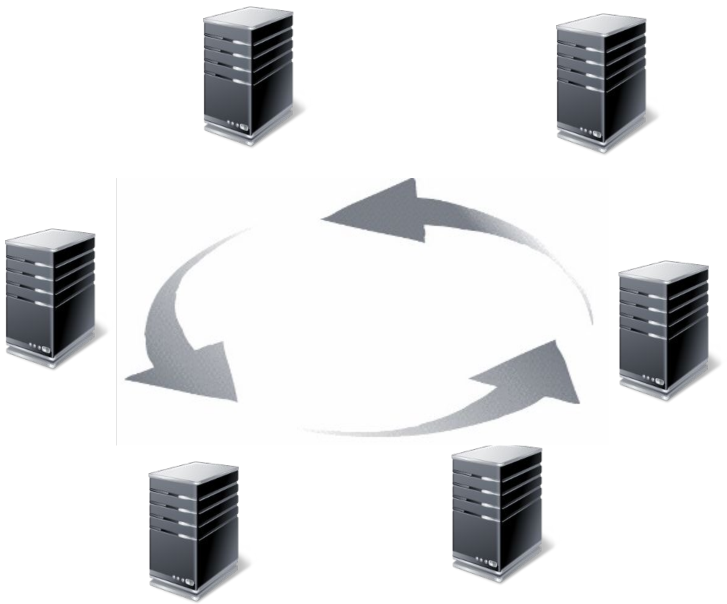
(all players output the same ordering of finalized txs)

Liveness

(transactions eventually get finalized)

n players, **f < n/3** malicious faults, **partial synchronous** network.
know public keys ahead of time (bare PKI)

Our setting: the consensus problem



∃ unknown time **GST**, an unknown **δ**, and a known time bound **Δ > δ** s.t.

- **After GST**, every message is delivered within **δ seconds.**
- **Before GST,** no guarantee.

Models unreliable network [DSL88]

n players, **f < n/3** malicious faults, **partial synchronous** network.
know public keys ahead of time (bare PKI)

Our Work

Thm: Assuming a (Bare) PKI, CRH, there exists a partially synchronous "random-leader" consensus protocol for $f < n/3$ static corruptions, and:

- Optimistic confirmation time of $\boldsymbol{3\delta}$ (excluding block time)
- Optimistic block time of $\boldsymbol{2\delta}$
- Expected pessimistic confirmation time* of $\boldsymbol{3.5\delta + 1.5\Delta}$
- Worst-case confirmation time of $\boldsymbol{4\delta + \omega(\log \lambda) \cdot (3\Delta + \delta)}$
- Easiest security proofs (in our eyes)

δ: unknown, true message delay during periods of synchrony
Δ: known, public upper bound on δ

Get efficient communication via "sortition" [CM18]

Comparisons

Theoretical latency of protocols that support random leaders

	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

Fun note: all protocols differ only slightly in protocol description

	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

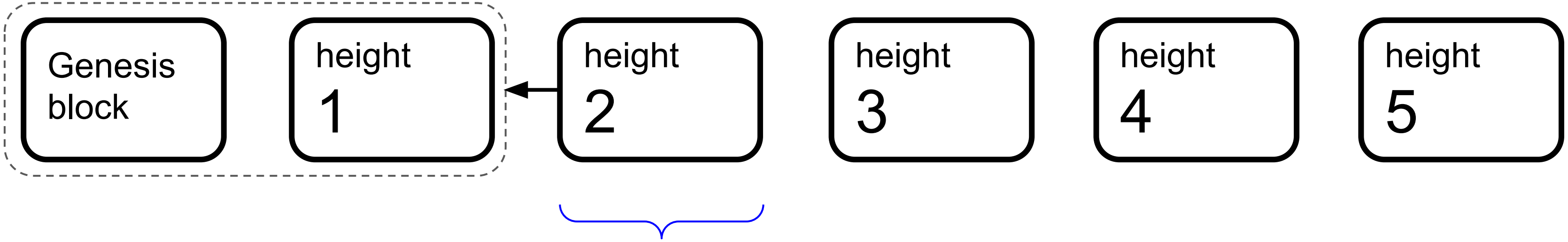
*Base protocol without sortition.

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

Protocol Description

Preliminaries

Key data structure: **blockchain**

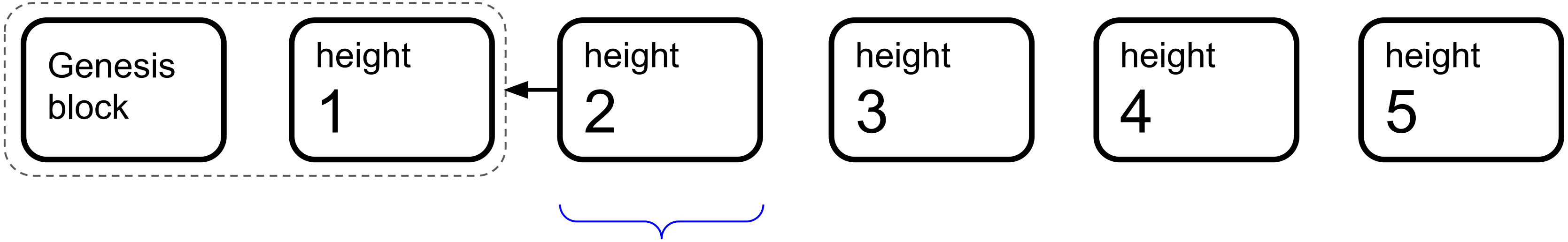


each block of height **h** is a tuple of the form

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

Preliminaries

Key data structure: **blockchain**

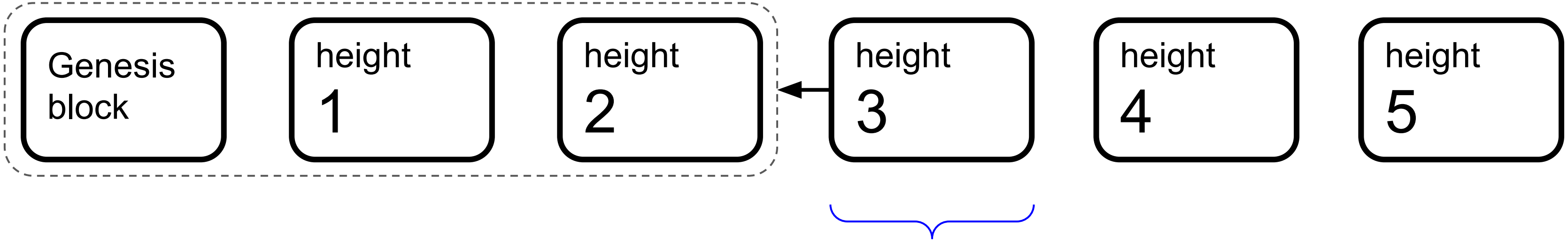


each block of height **h** is a tuple of the form

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

Preliminaries

Key data structure: **blockchain**

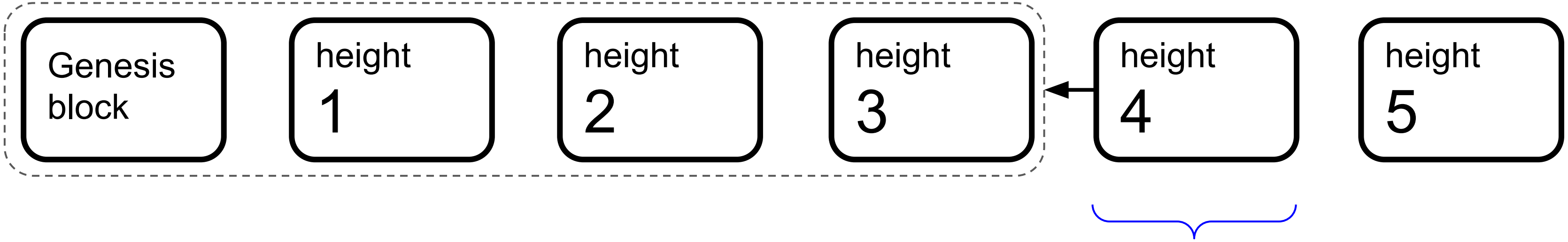


each block of height **h** is a tuple of the form

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

Preliminaries

Key data structure: **blockchain**

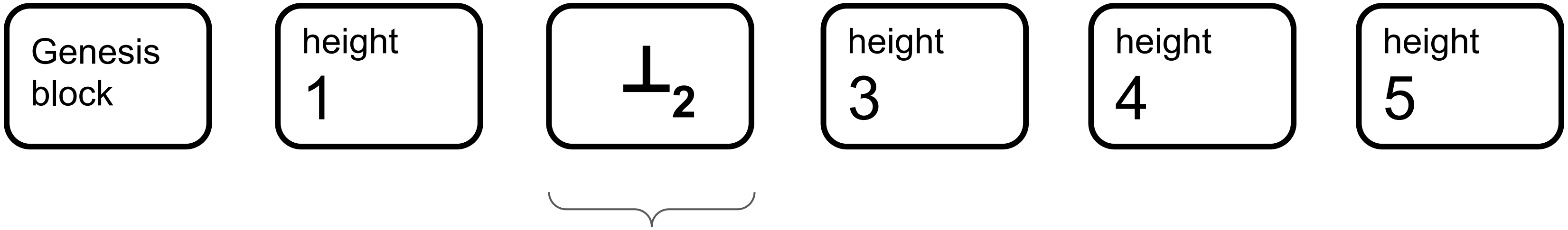


each block of height **h** is a tuple of the form

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

Preliminaries: dummy blocks

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

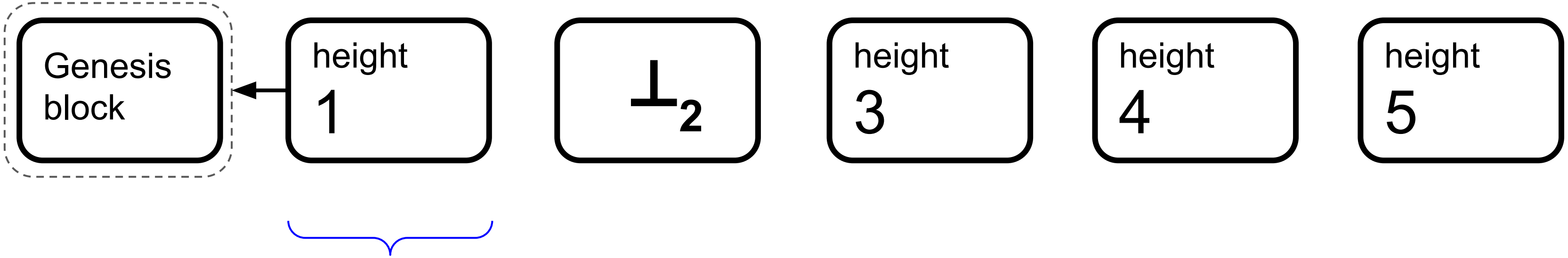


a dummy block of height **h** is the tuple

$$\bot_h = (h, \bot, \bot)$$

Preliminaries: dummy blocks

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

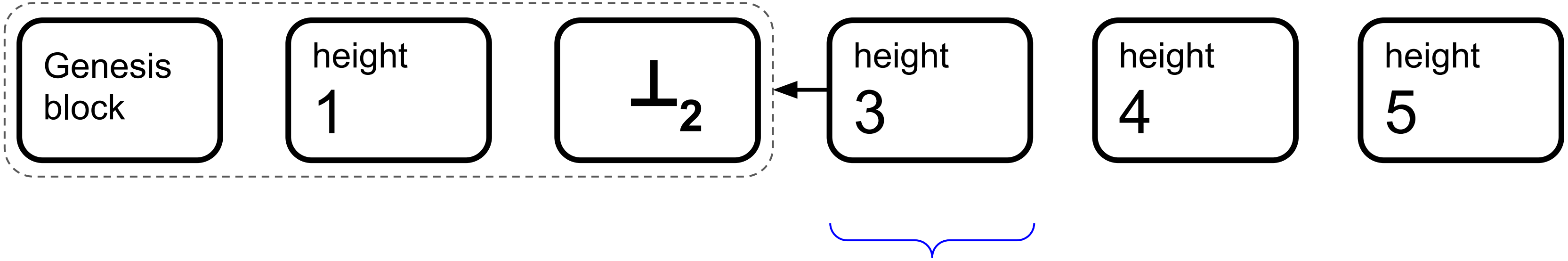


(again, each block that is not a dummy block must extend a parent chain)

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

Preliminaries: dummy blocks

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

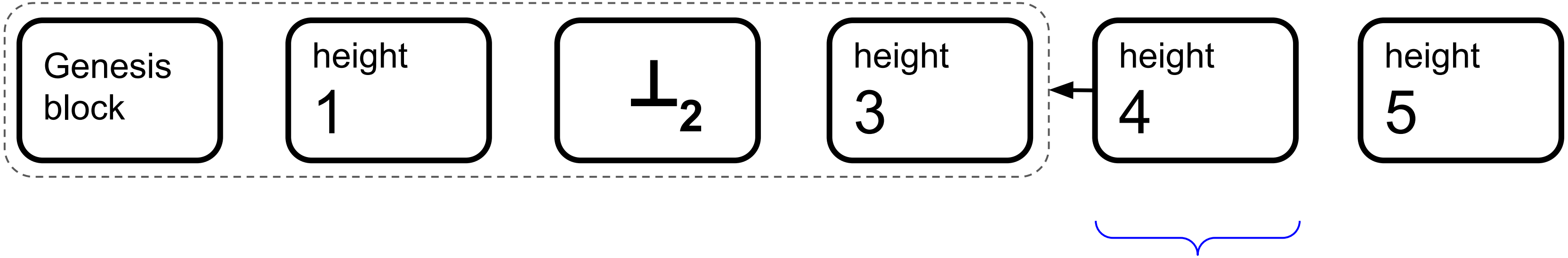


(again, each block that is not a dummy block must extend a parent chain)

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

Preliminaries: dummy blocks

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

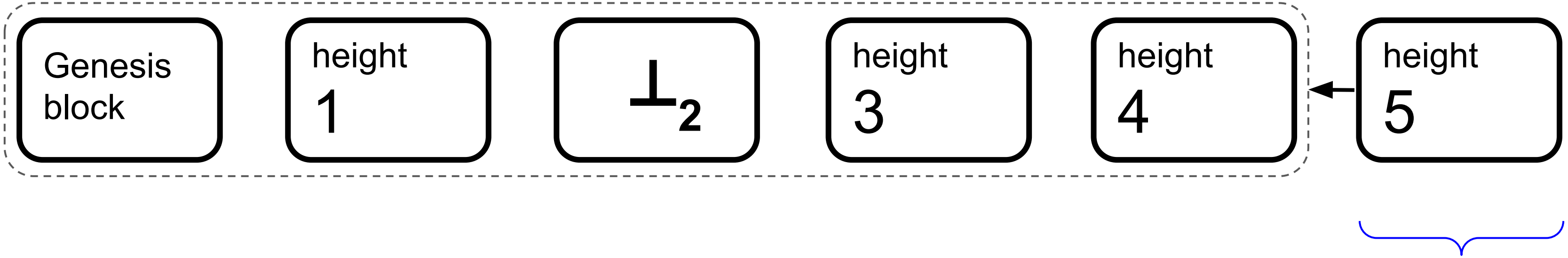


(again, each block that is not a dummy block must extend a parent chain)

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

Preliminaries: 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)

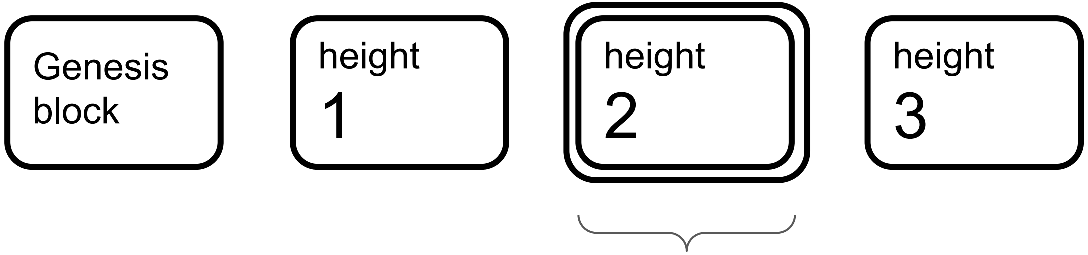
$$\mathbf{b_h = (h,}\ \mathit{Hash}\mathbf{(b_1 \ldots b_{h-1}), txs}$$

Preliminaries: voting for blocks

A player **i** votes for a block $\mathbf{b_h}$ by signing the message "**vote** $\mathbf{b_h}$"

Preliminaries: notarized blocks

Key data structure: **notarized blocks**



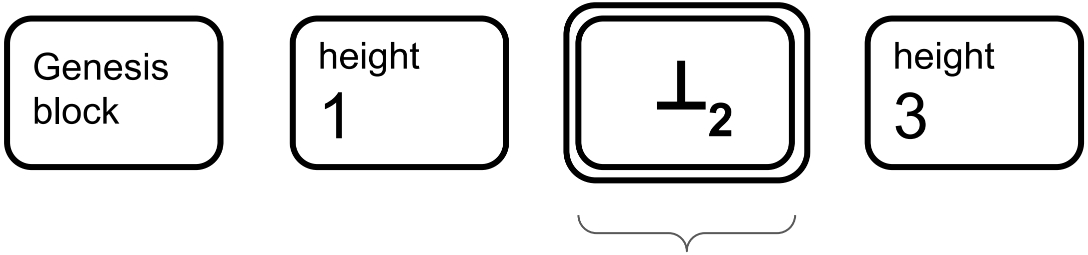
a block is notarized in my view if I've seen

> 2n/3 votes for it

(i.e. signatures from > 2n/3 different players)

Preliminaries: notarized blocks

Dummy blocks can also be **notarized.**



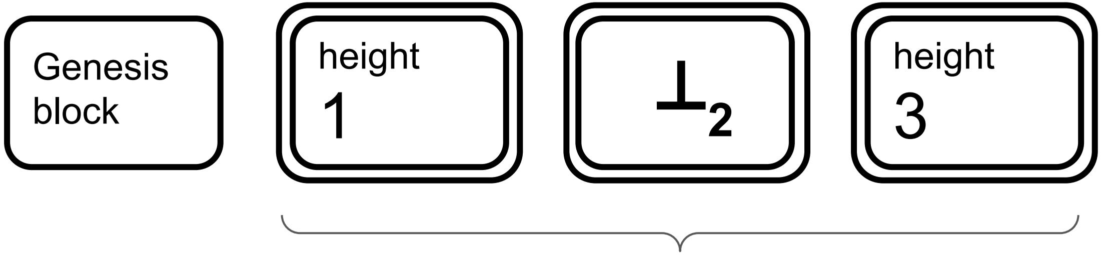
a block is notarized in my view if I've seen

> 2n/3 votes for it

(i.e. signatures from > 2n/3 different players)

Preliminaries: Notarized blockchains

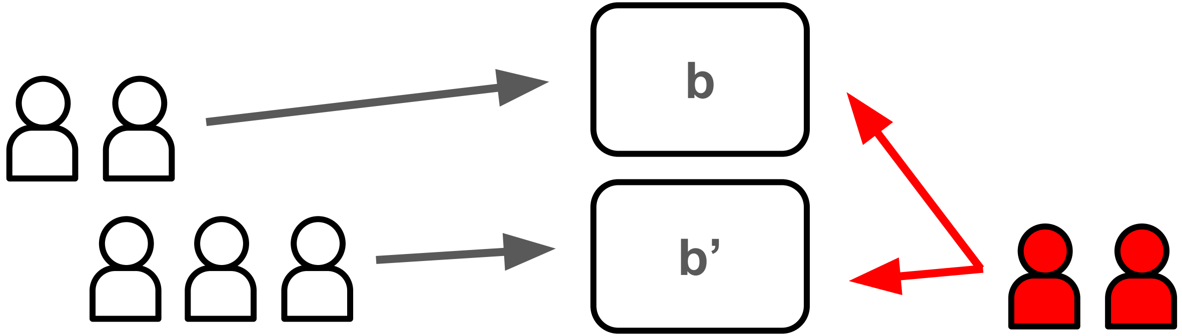
Key data structure: **notarized blockchain**



every block of the chain is notarized (except genesis)

Preliminaries: “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’**.

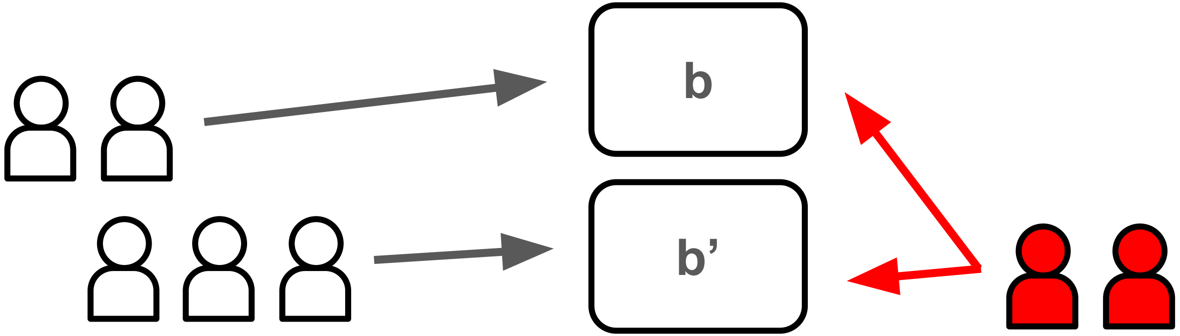


suppose each honest player only votes for one

corrupt players can always vote for both

Preliminaries: "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

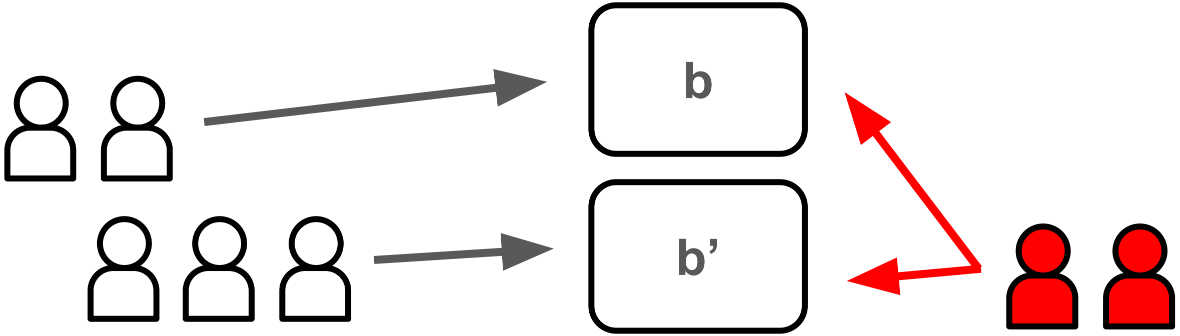
n – f votes

corrupt players can always vote for both

2f votes

Preliminaries: "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**

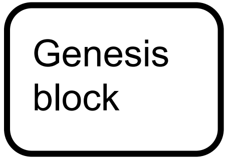
corrupt players can always vote for both

2f votes

The Simplex Consensus Protocol

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

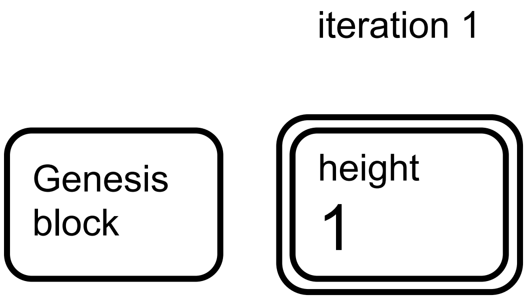
In each iteration **h**, collectively try to build a notarized block of height **h.**



The Simplex Consensus Protocol

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

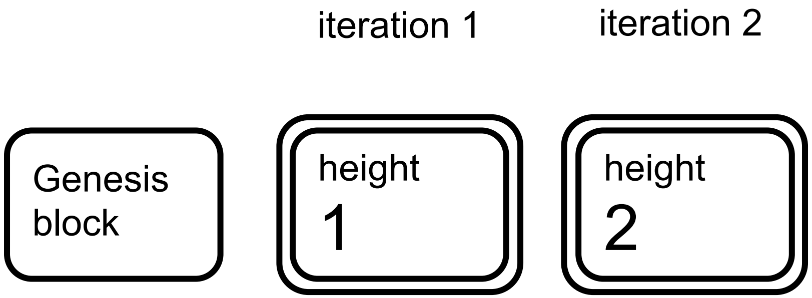
In each iteration **h**, collectively try to build a notarized block of height **h.**



The Simplex Consensus Protocol

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

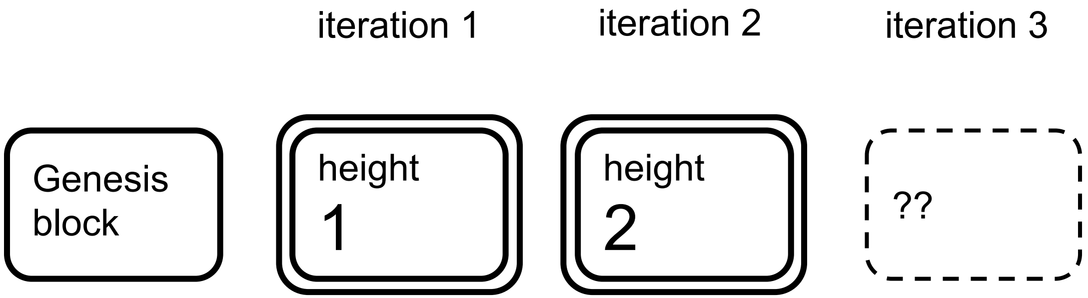
In each iteration **h**, collectively try to build a notarized block of height **h.**



The Simplex Consensus Protocol

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

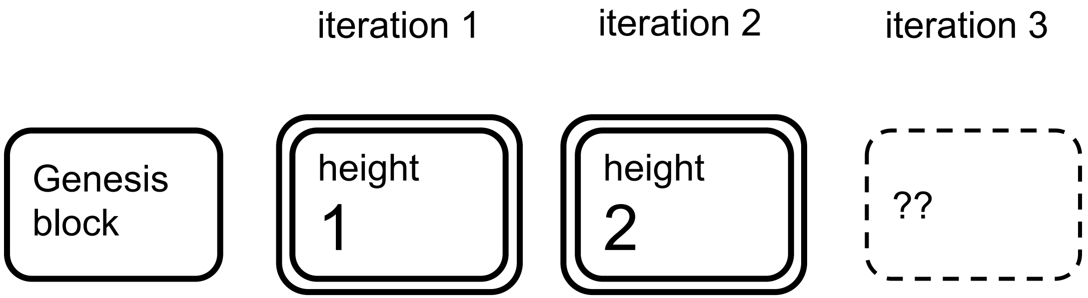
In each iteration **h**, collectively try to build a notarized block of height **h.**



The Simplex Consensus Protocol

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

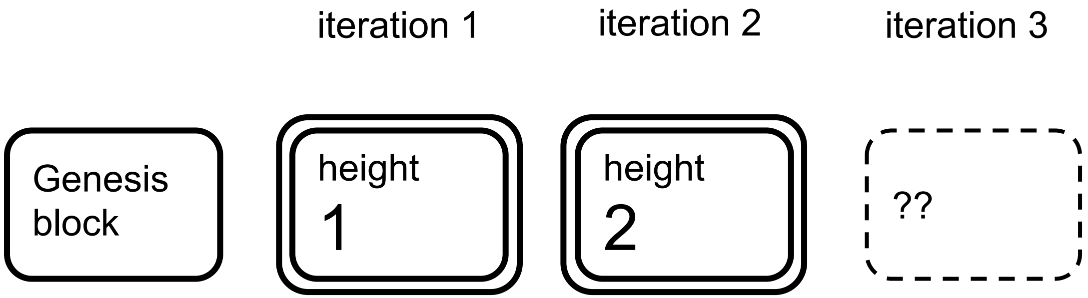
Move to the next iteration when I've seen a notarized blockchain of length **h**



The Simplex Consensus Protocol

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

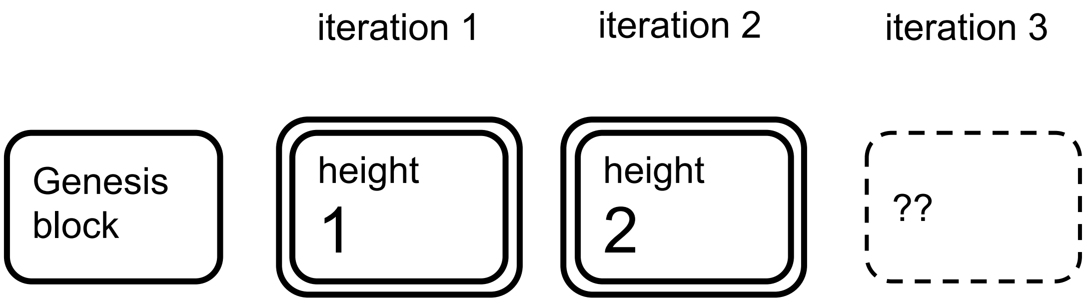
Move to the next iteration when I've seen a notarized blockchain of length **h** (and send this notarized blockchain to everyone else).



Constructing notarized blocks

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

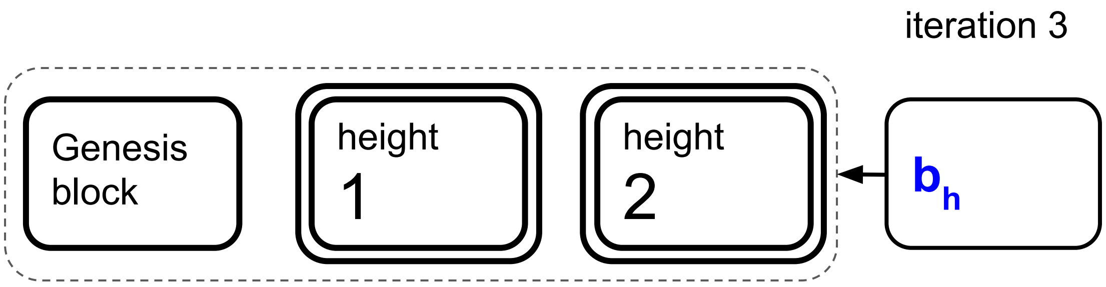
Specifically, the leader of iteration **h** = H^* **(h)** mod **n**, where H^* is a random oracle.



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 $\mathbf{b_h}$"**.

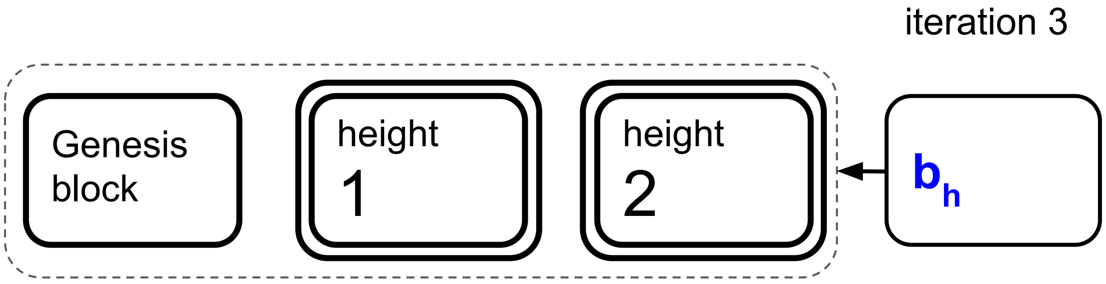


Constructing notarized blocks

Should include all pending transactions.

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

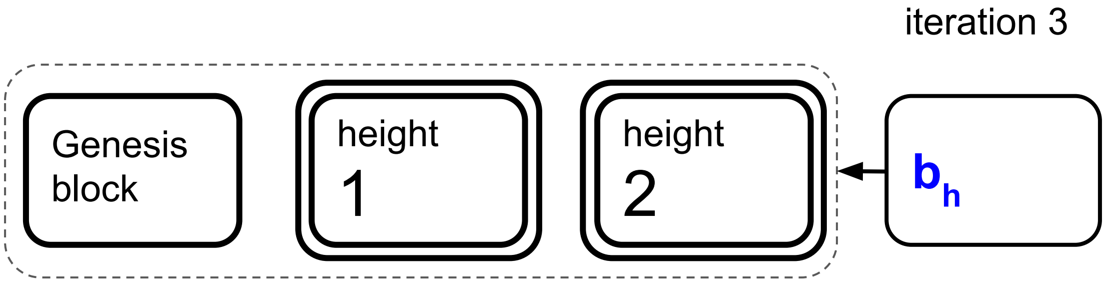
1. If **i** is the leader, **i** chooses notarized blockchain of length **h-1**, extends it with a new block b_h and sends everyone a signed message "**propose** b_h".



Constructing notarized blocks

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

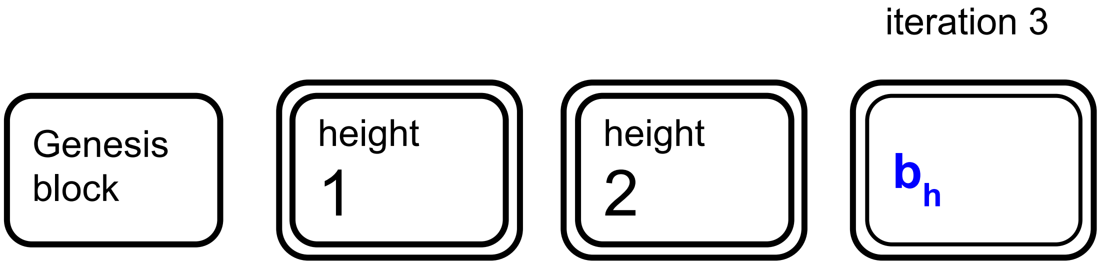
1. If **i** is the leader, **i** chooses notarized blockchain of length **h-1**, extends it with a new block b_h and sends everyone a signed message "**propose** b_h".
2. On seeing the ***first*** valid proposal from the leader, player **i** sends everyone a signed message "**vote** b_h".



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** $\mathbf{b_h}$".

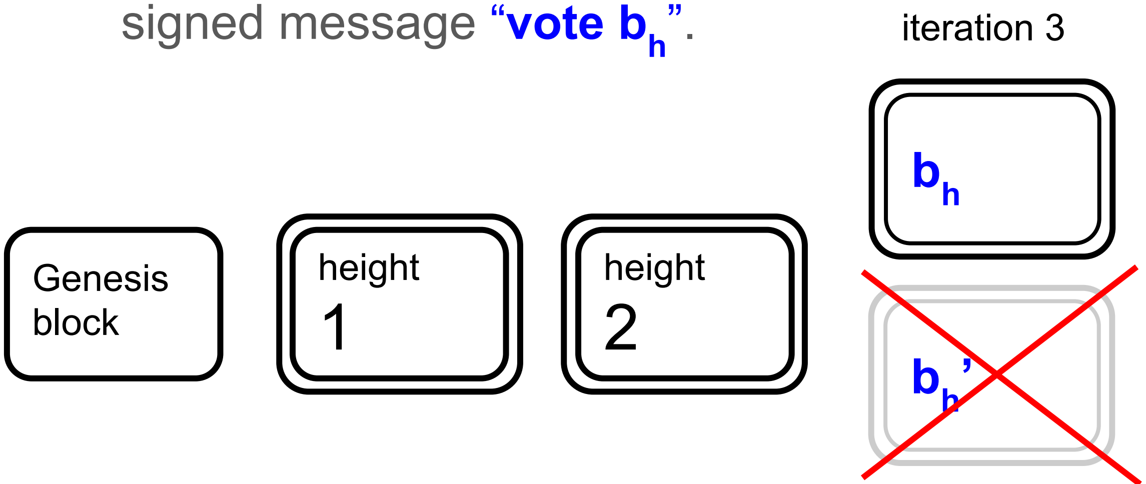


If the network is good and the leader is honest, the block proposal will get notarized!

Constructing notarized blocks

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

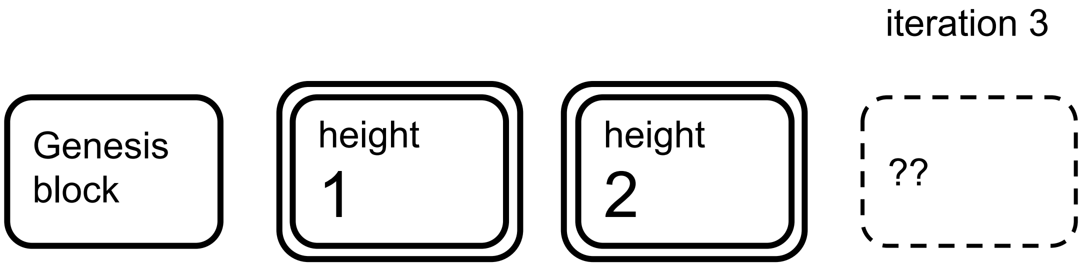
1. If **i** is the leader, **i** chooses notarized blockchain of length **h-1**, extends it with a new block b_h and sends everyone a signed message "**propose** b_h".
2. On seeing the ***first*** valid proposal from the leader, player **i** sends everyone a signed message "**vote** b_h".



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

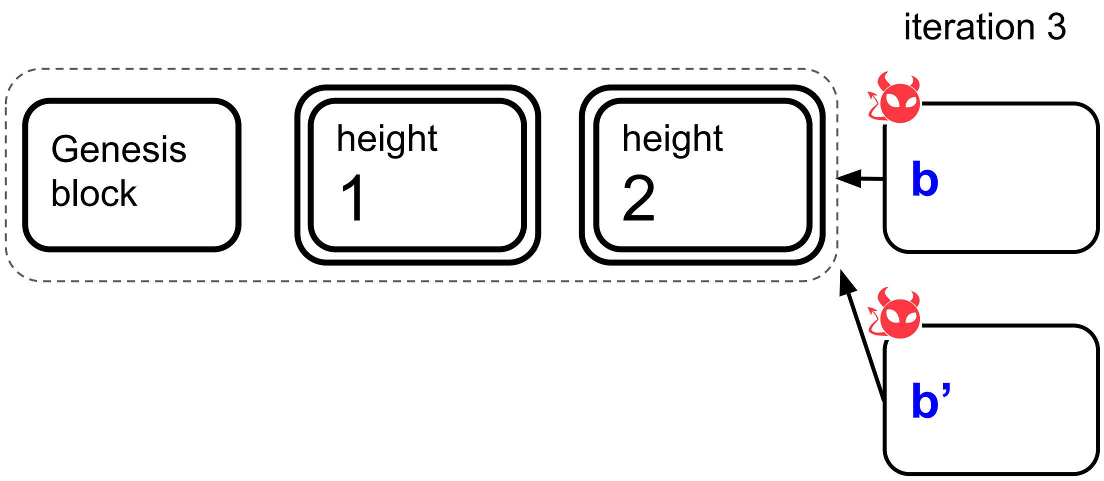
Handling faulty iterations

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



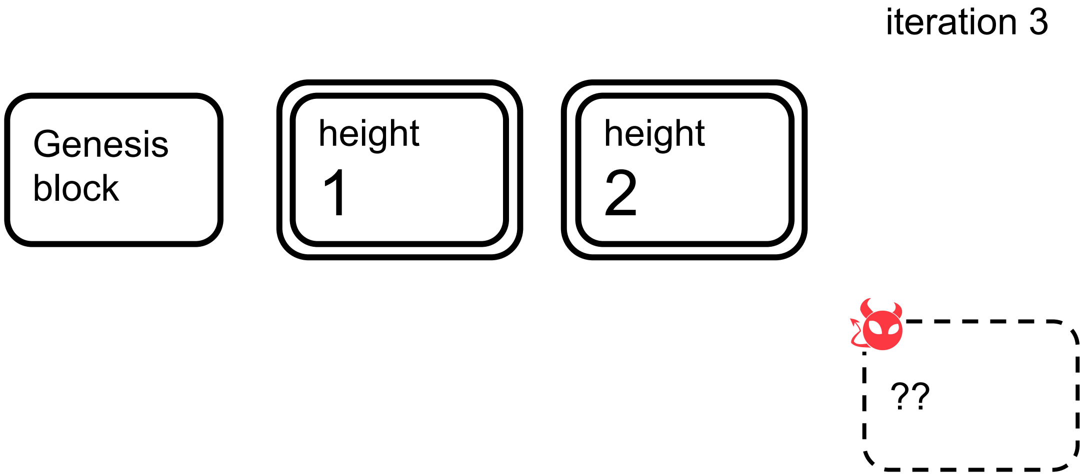
Handling faulty iterations

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.

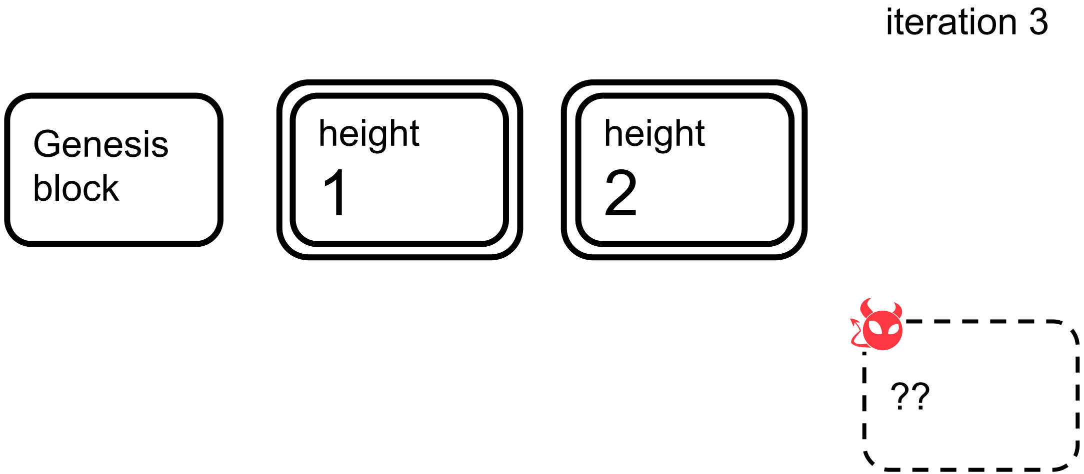
If **3Δ time** has passed since player **i** has entered iteration **h**, and if **i** still has not entered iteration **h+1**, player **i** sends to everyone a signed message “**vote $\bot_h$**”.



Solution: dummy blocks.

Recall: **Δ** is a public parameter that upper bounds message delay after GST.

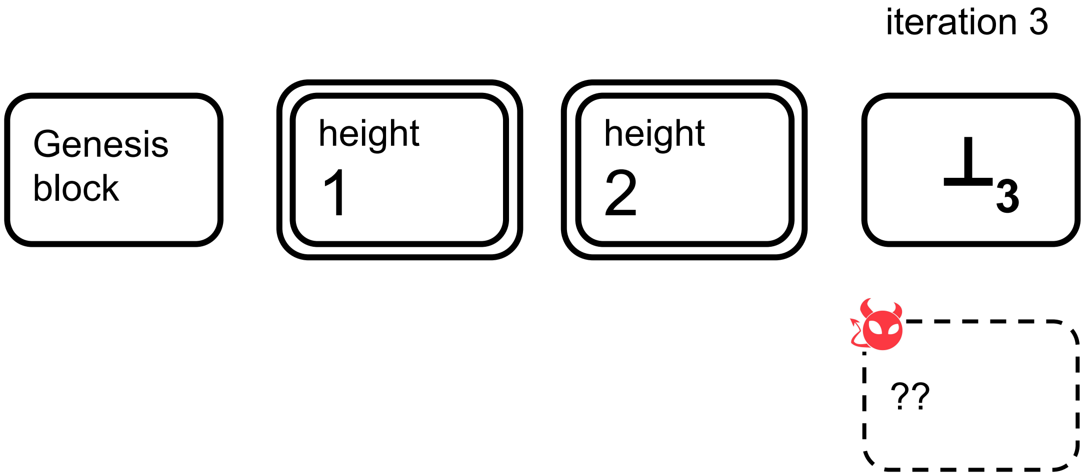
If **3Δ time** has passed since player **i** has entered iteration **h**, and if **i** still has not entered iteration **h+1**, player **i** sends to everyone a signed message **"vote $\perp_h$"**.



Solution: dummy blocks.

Recall: **Δ** is a public parameter that upper bounds message delay after GST.

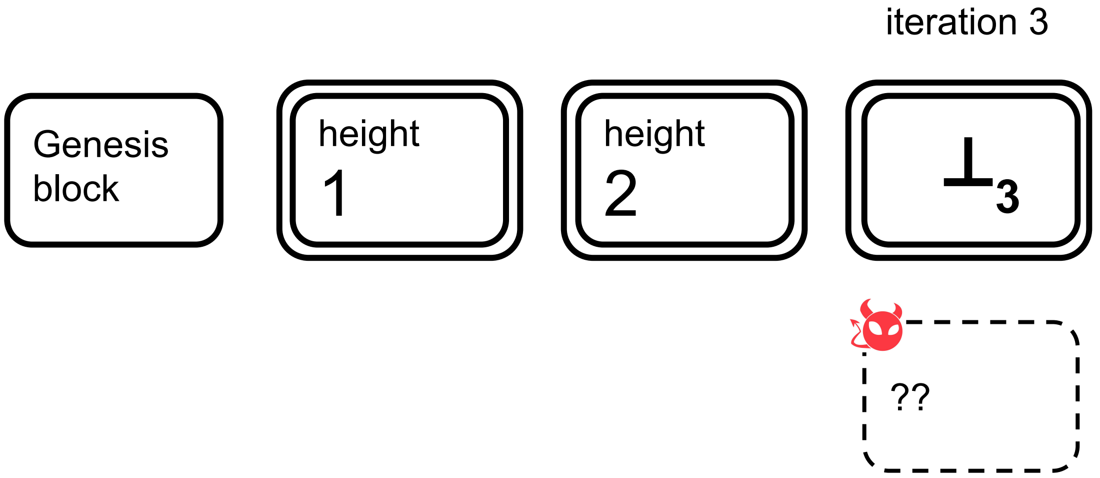
If **3Δ time** has passed since player **i** has entered iteration **h**, and if **i** still has not entered iteration **h+1**, player **i** sends to everyone a signed message **"vote $\perp_h$"**.



Solution: dummy blocks.

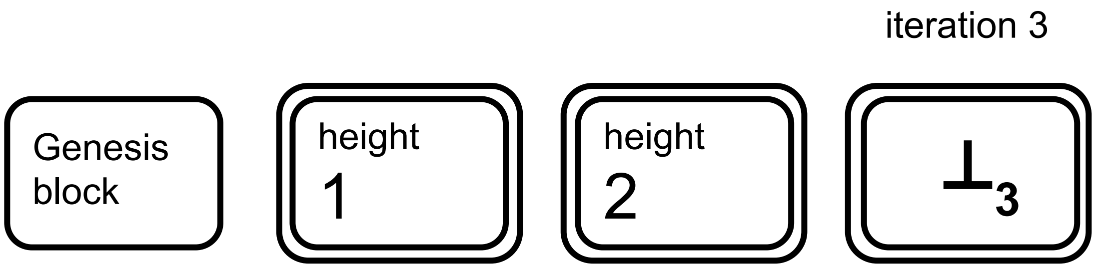
Recall: **Δ** is a public parameter that upper bounds message delay after GST.

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.

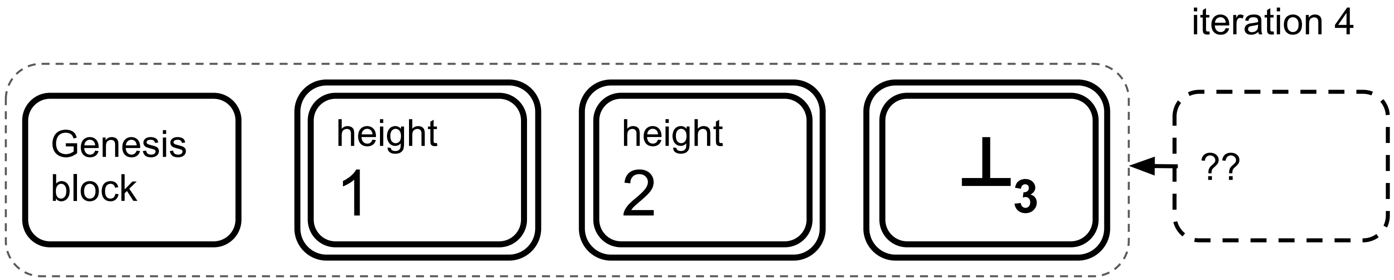
If **3Δ time** has passed since player **i** has entered iteration **h**, and if **i** still has not entered iteration **h+1**, player **i** sends to everyone a signed message "**vote** $\perp_h$".



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!

Interlude…

Due to faults, there may be ***both***

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

in the view of honest players.

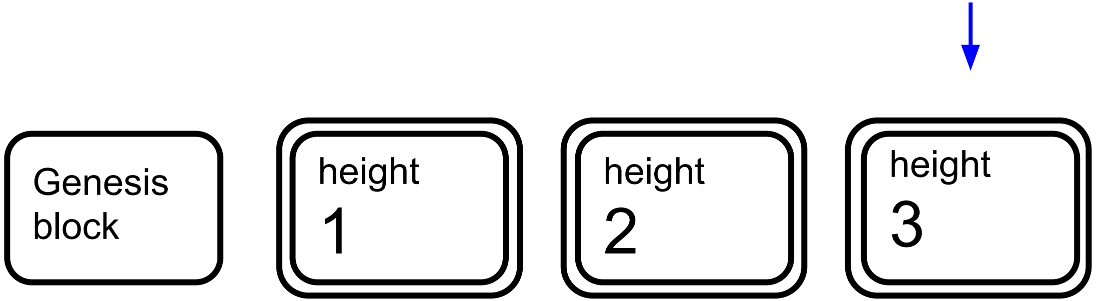
Interlude…

Due to faults, there may be ***both***

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

in the view of honest players.

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

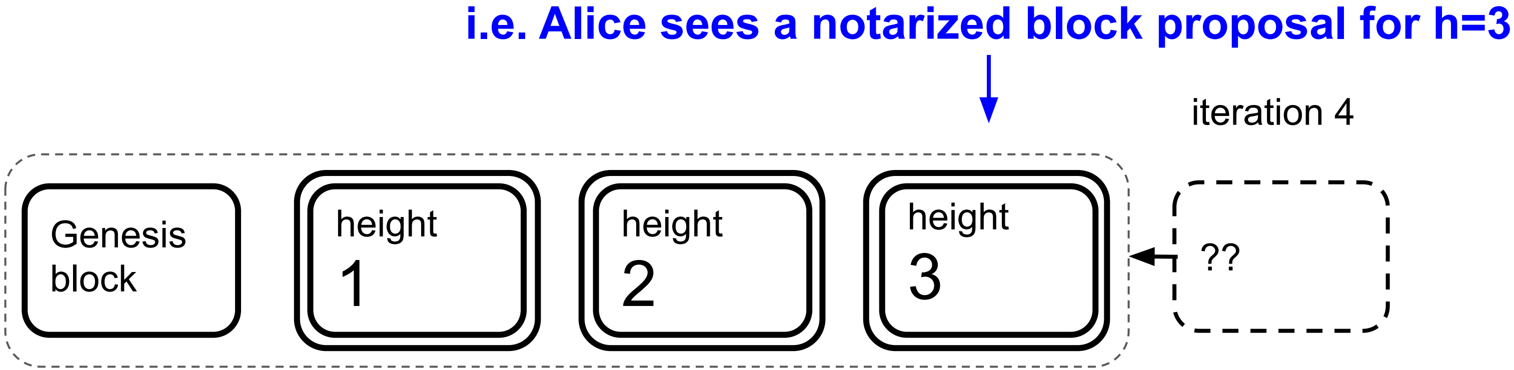


Interlude…

Due to faults, there may be ***both***

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

in the view of honest players.



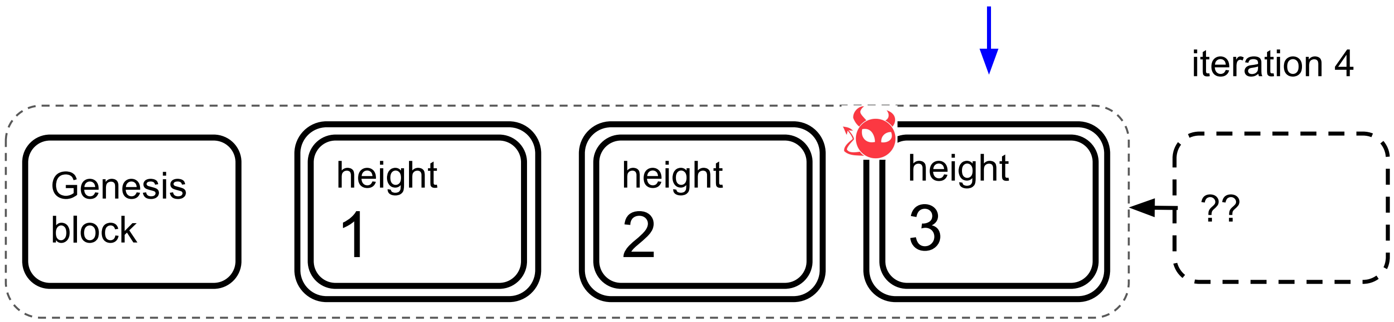
Interlude…

Due to faults, there may be ***both***

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

in the view of honest players.

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



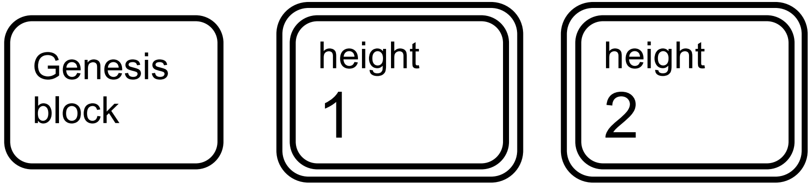
Interlude…

Due to faults, 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 due to asynchrony (and votes for $\perp_3$)



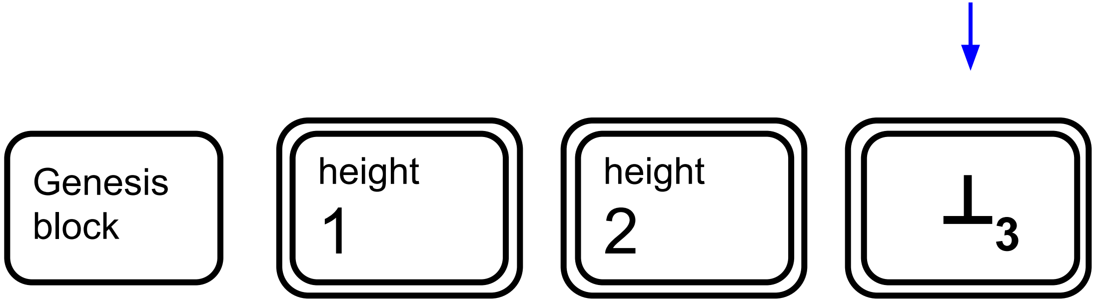
Interlude…

Due to faults, there may be ***both***

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

in the view of honest players.

so Bob sees a notarized dummy block $\perp_3$



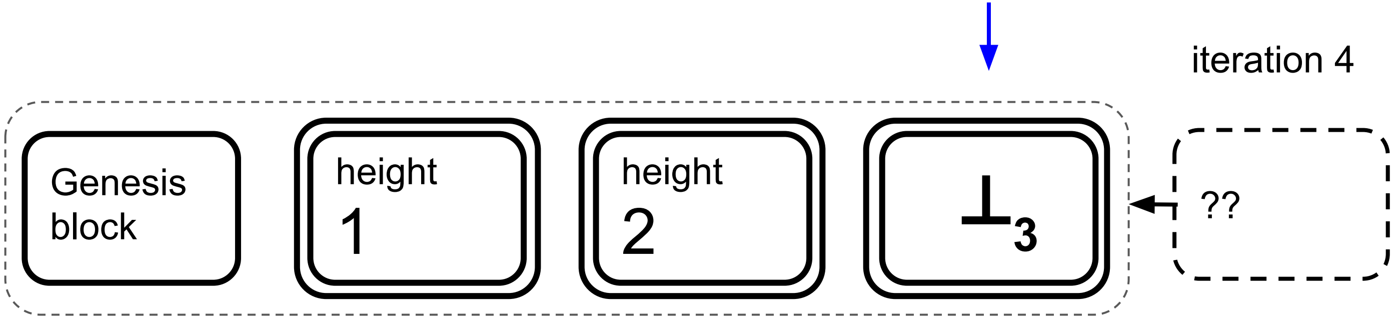
Interlude…

Due to faults, there may be ***both***

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

in the view of honest players.

so Bob sees a notarized dummy block $\perp_3$

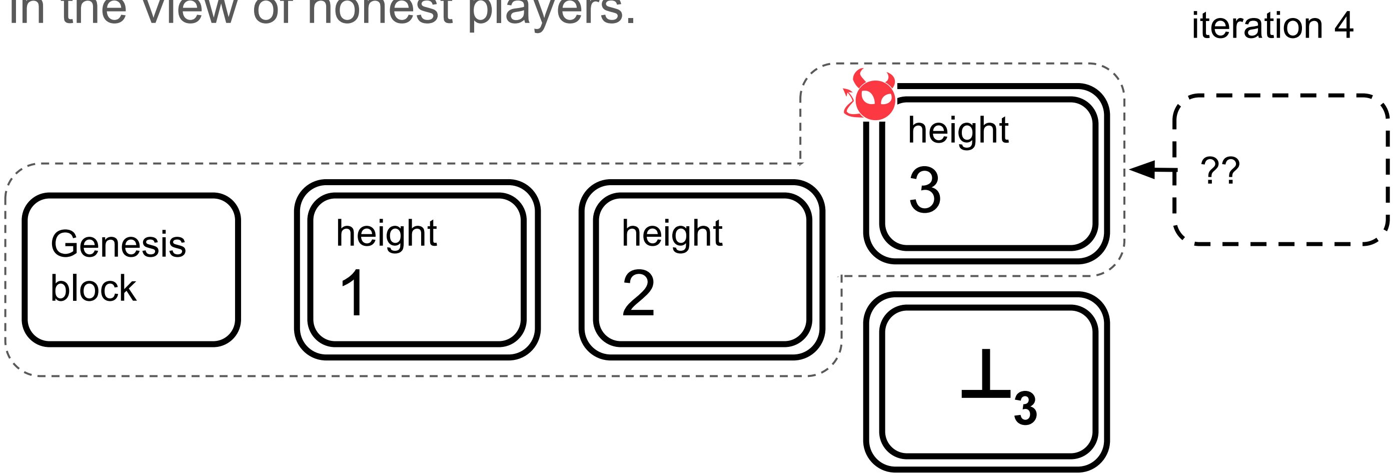


Interlude…

Due to faults, there may be ***both***

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

in the view of honest players.



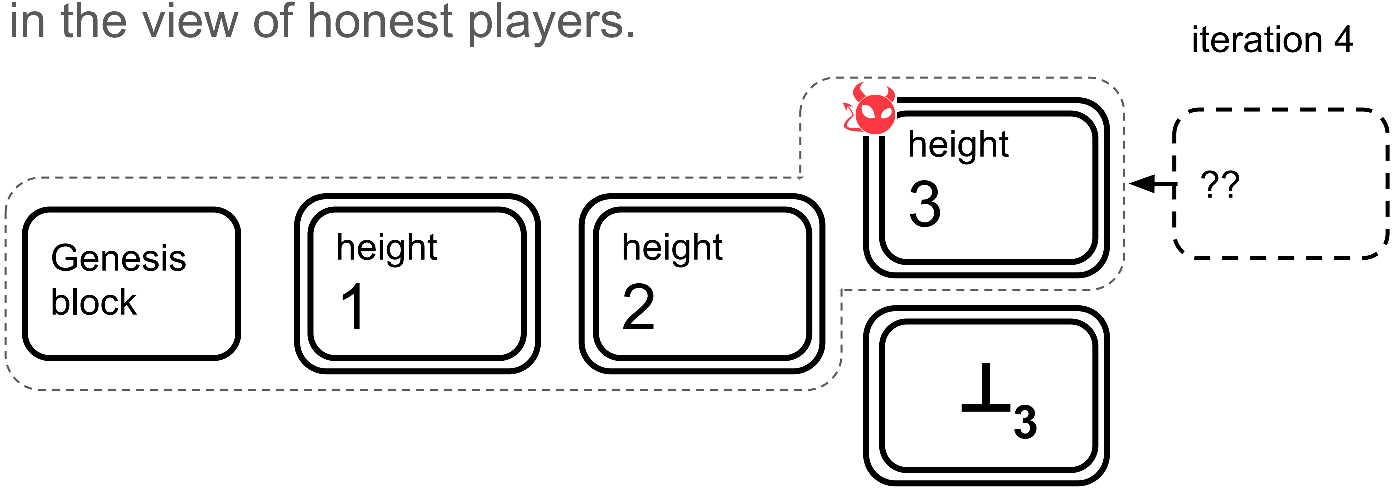
Interlude…

Due to faults, there may be ***both***

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

in the view of honest players.

The next leader can extend either notarized chain



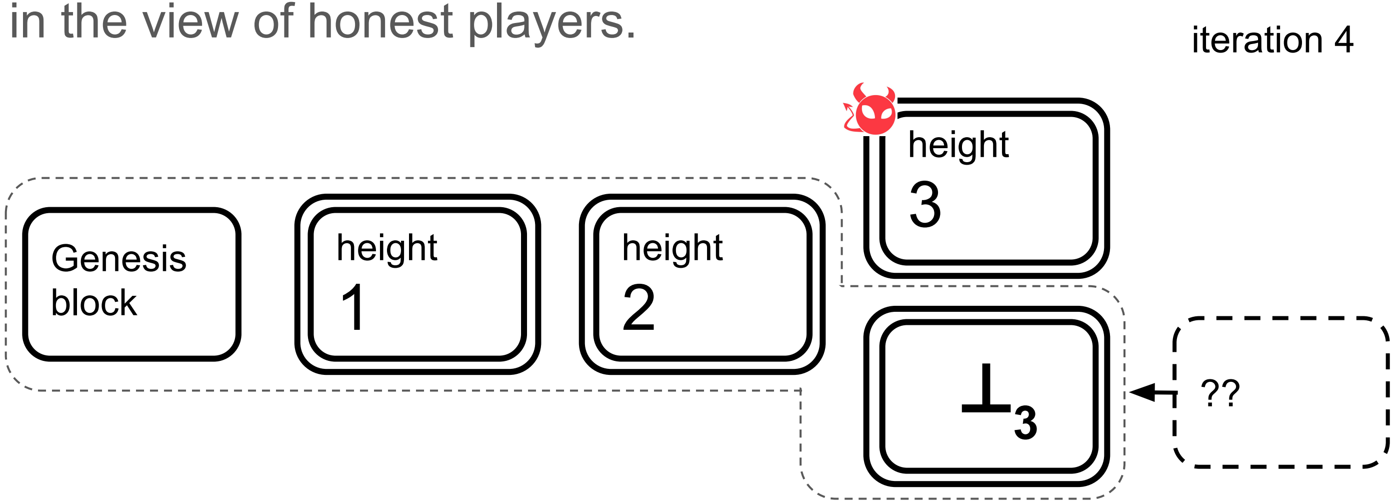
Interlude…

Due to faults, there may be ***both***

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

in the view of honest players.

The next leader can extend either notarized chain



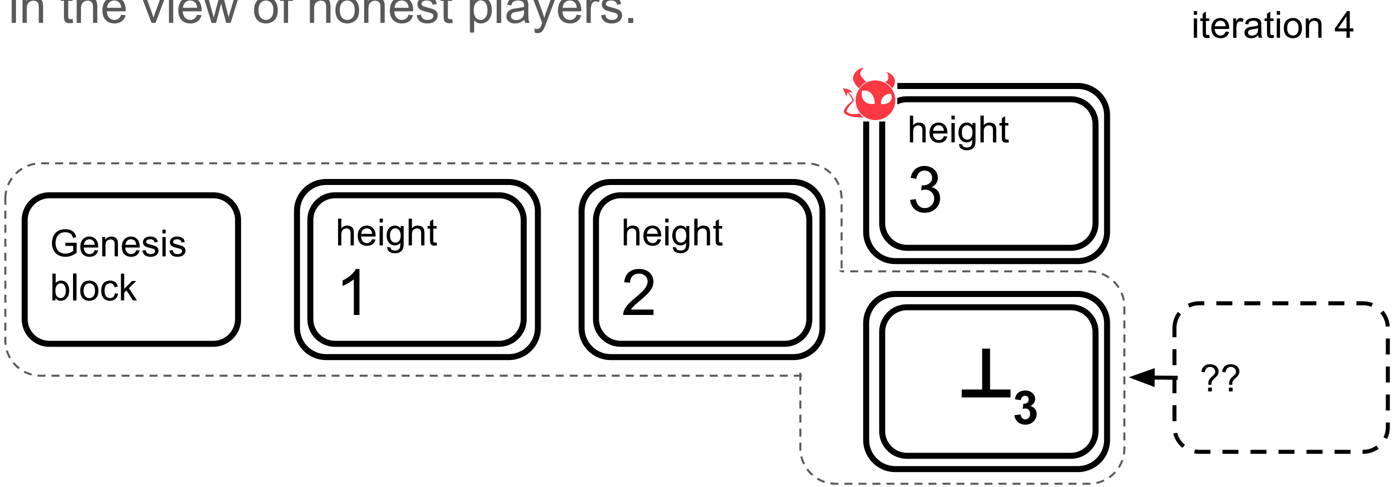
Interlude…

Due to faults, there may be ***both***

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

in the view of honest players.

For agreement, need to decide on a single block at each height **h**



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

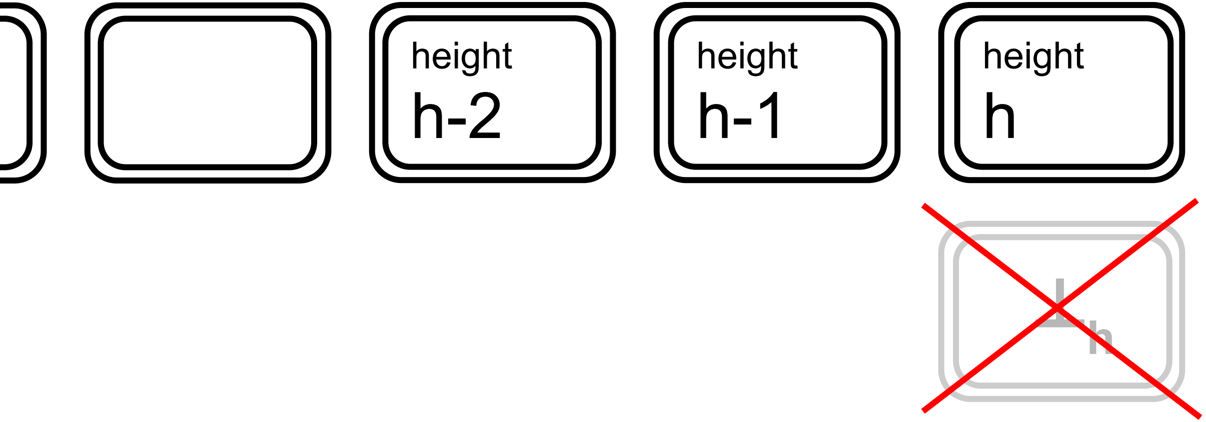
When player **i** enters iteration **h+1**, if **i** did not time out and vote for the dummy block for **h**, player **i** sends everyone a signed "**finalize h**" message.

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

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 (and the txs inside).

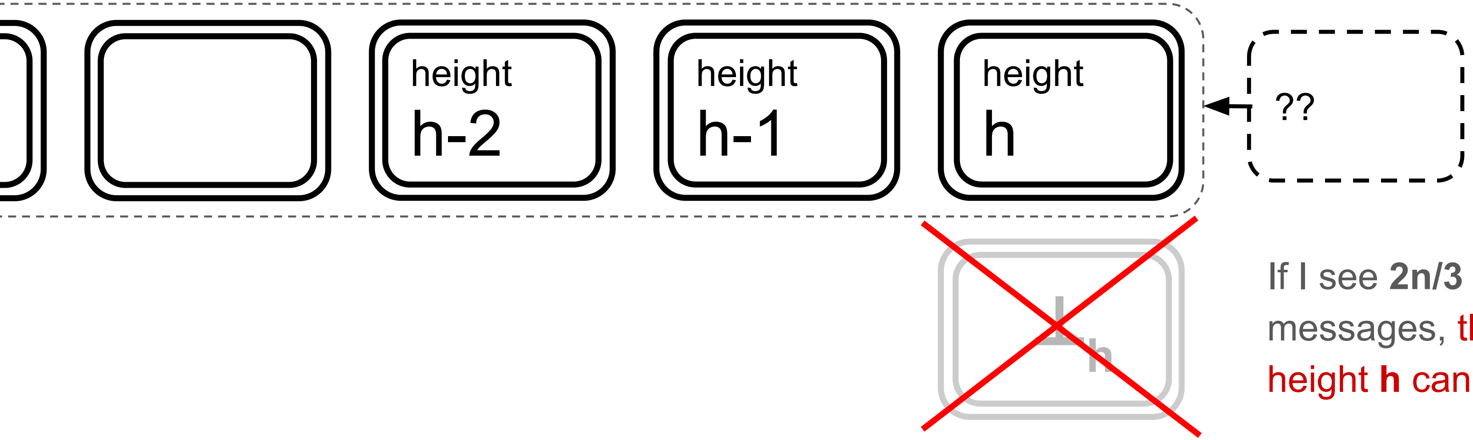


If I see **2n/3** "**finalize h**" messages, the dummy block of height **h** cannot be notarized!

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 (and the txs inside).



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

Security Proofs

Consistency

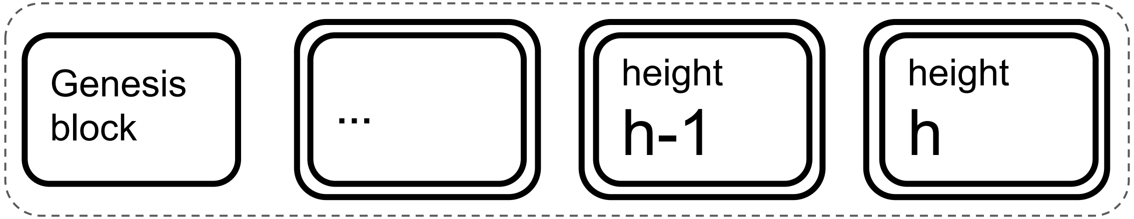
Thm: Let Alice and Bob be two honest players.

Suppose Alice outputs **LOG**, and Bob outputs **LOG'**, s.t **|LOG| ≤|LOG'|**.

Then, **LOG** is a prefix (or equal to) **LOG'**.

Consistency

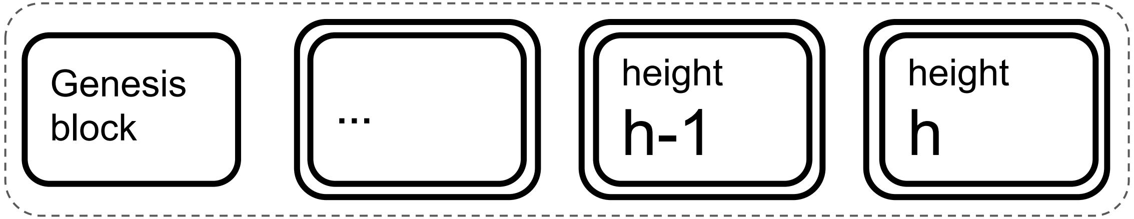
Proof: Consider Alice's chain **LOG**, which is the shorter one; let its length be **h**



Consistency

Since **LOG** is finalized by Alice, Alice sees **2n/3** "**finalize h**" messages.

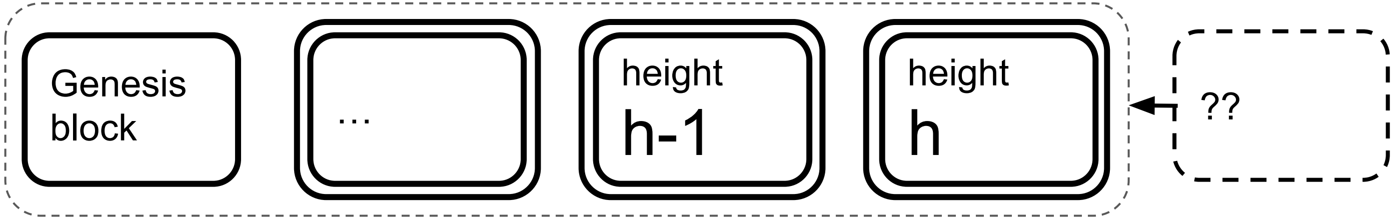
Claim: there can be only one notarized blockchain of length **h**, across all honest views



Consistency

Since **LOG** is finalized by Alice, Alice sees **2n/3** "**finalize h**" messages.

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

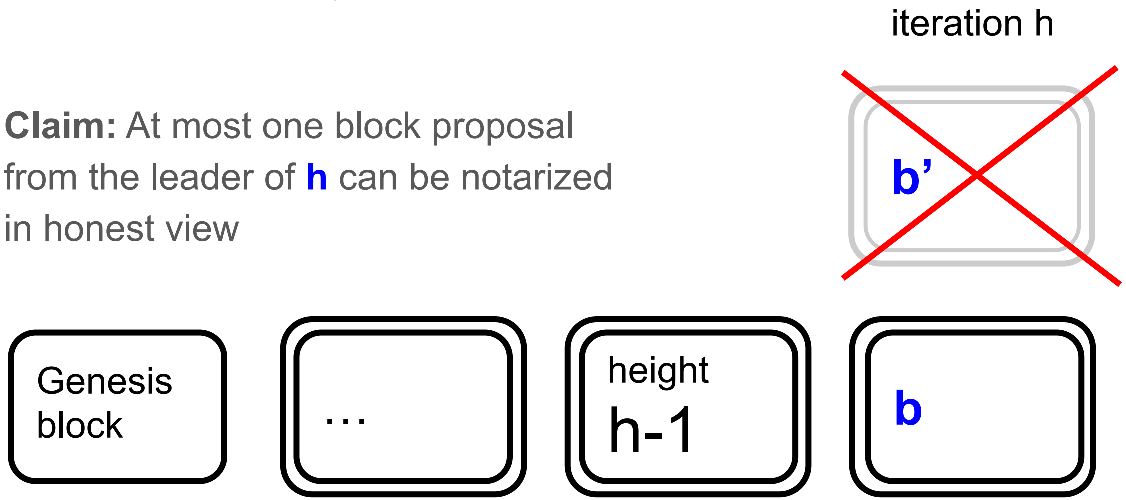


Thus Bob's chain must extend Alice's chain

Consistency

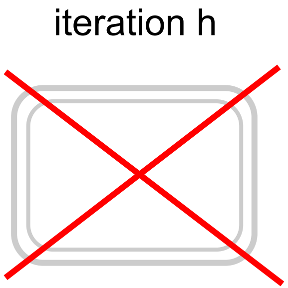
Claim: At most one block proposal from the leader of **h** can be notarized in honest view

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

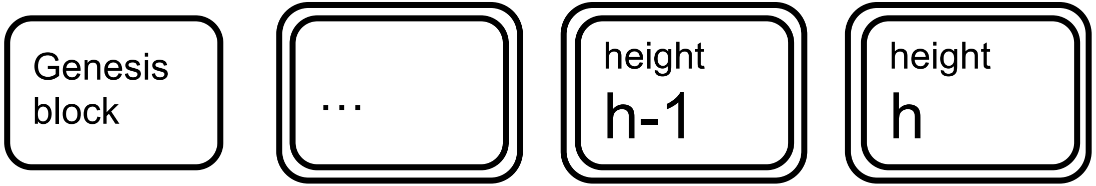


Consistency

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

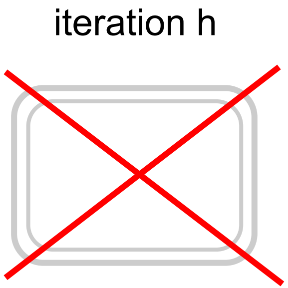


Proof: Each honest player votes for at most one proposal. Quorum intersection.

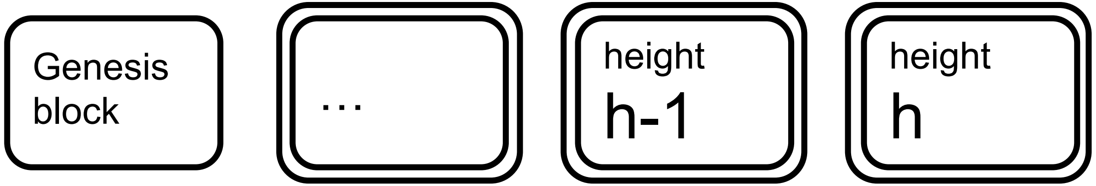


Consistency

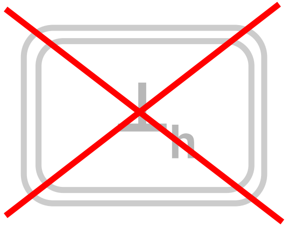
Claim: At most one block proposal from the leader of **h** can be notarized in honest view



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



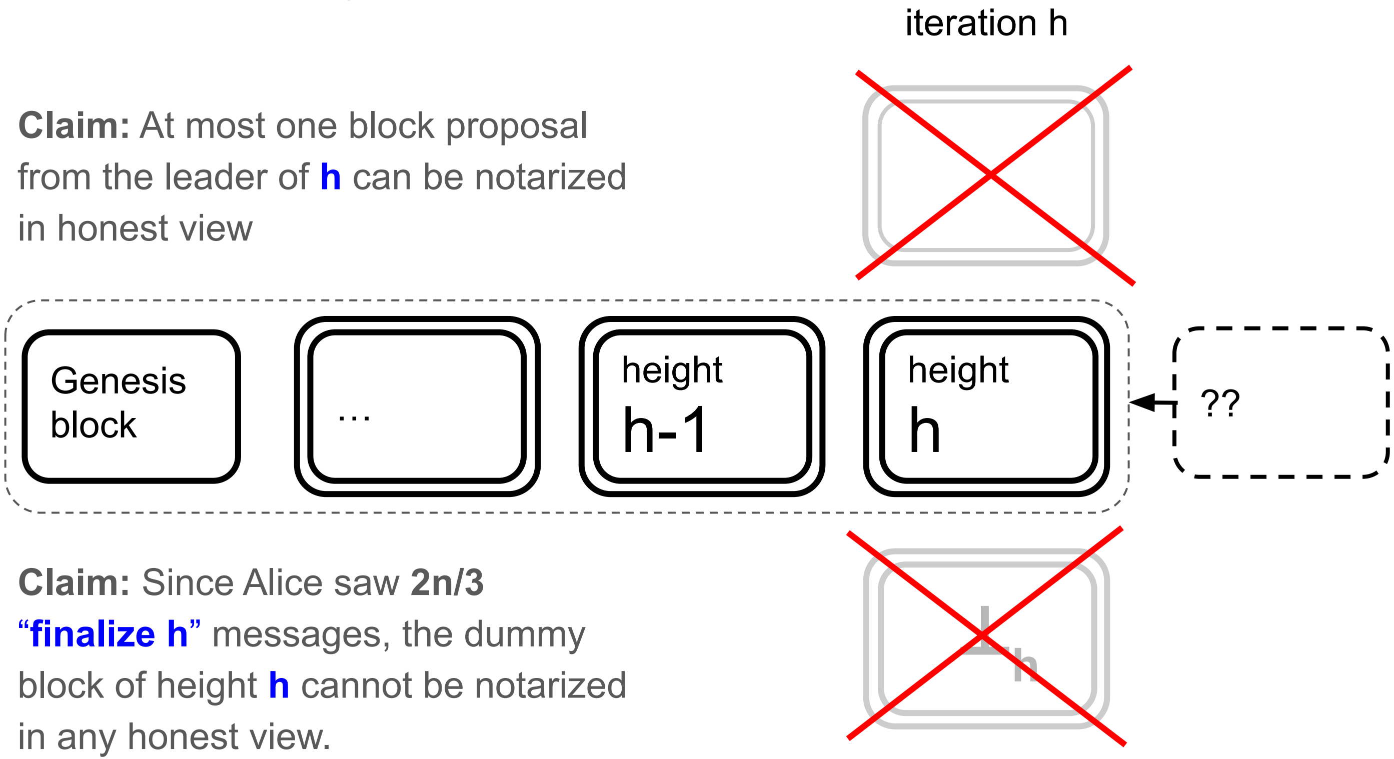
Claim: Since Alice saw **2n/3** **"finalize h"** messages, the dummy block of height **h** cannot be notarized in any honest view.



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

Consistency

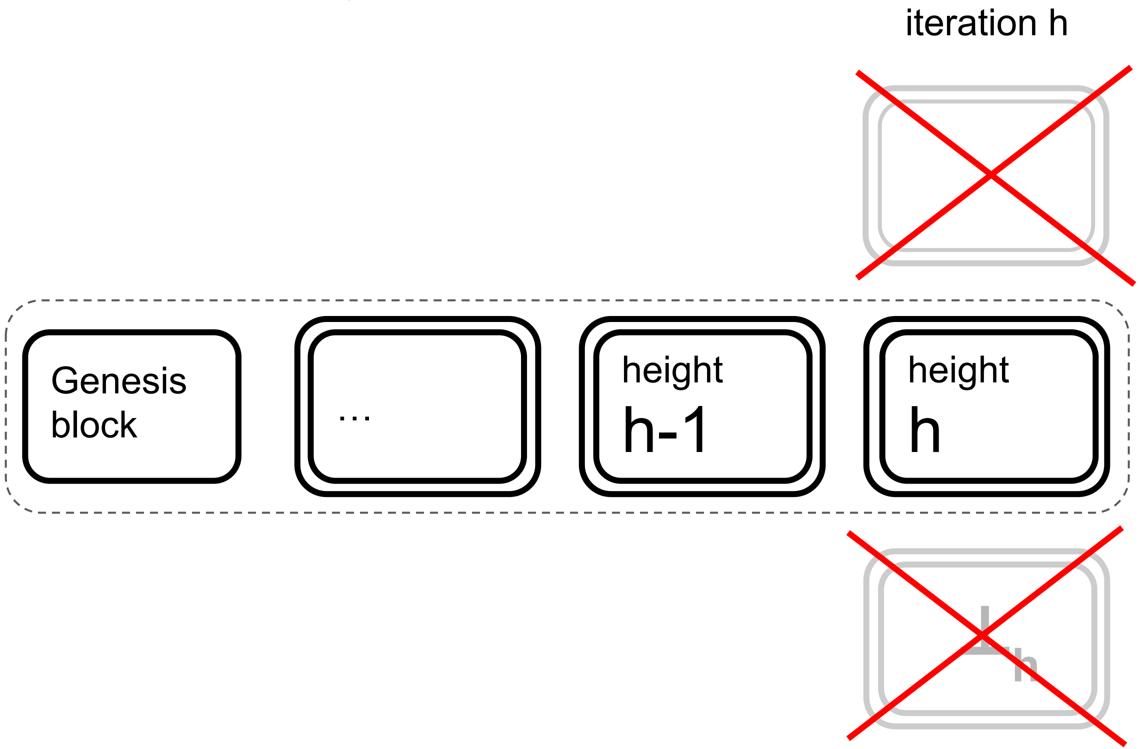
Claim: At most one block proposal from the leader of **h** can be notarized in honest view



Bob's chain (by virtue of being notarized) must extend Alice's chain.

Claim: Since Alice saw **2n/3** "**finalize h**" messages, the dummy block of height **h** cannot be notarized in any honest view.

Consistency



Safe to finalize the transactions in Alice's chain!

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.

time *t*

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

Liveness

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

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

Liveness

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

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

Every honest player enters iteration h and sees the proposal.

Either everyone sends "**vote** 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.

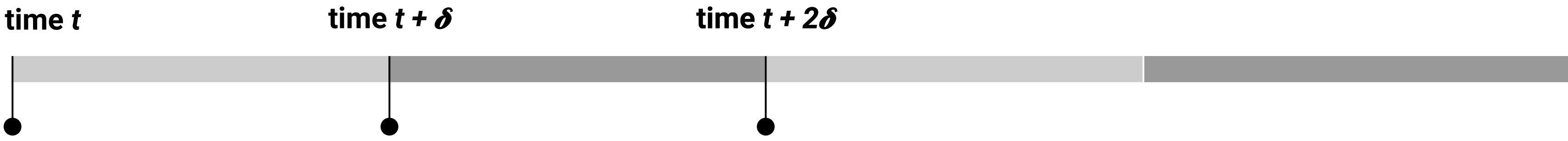
Either everyone sends "**vote** $\mathbf{b_h}$", or someone already entered iteration **h+1**.

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 $\boldsymbol{t + 2\delta}$.



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

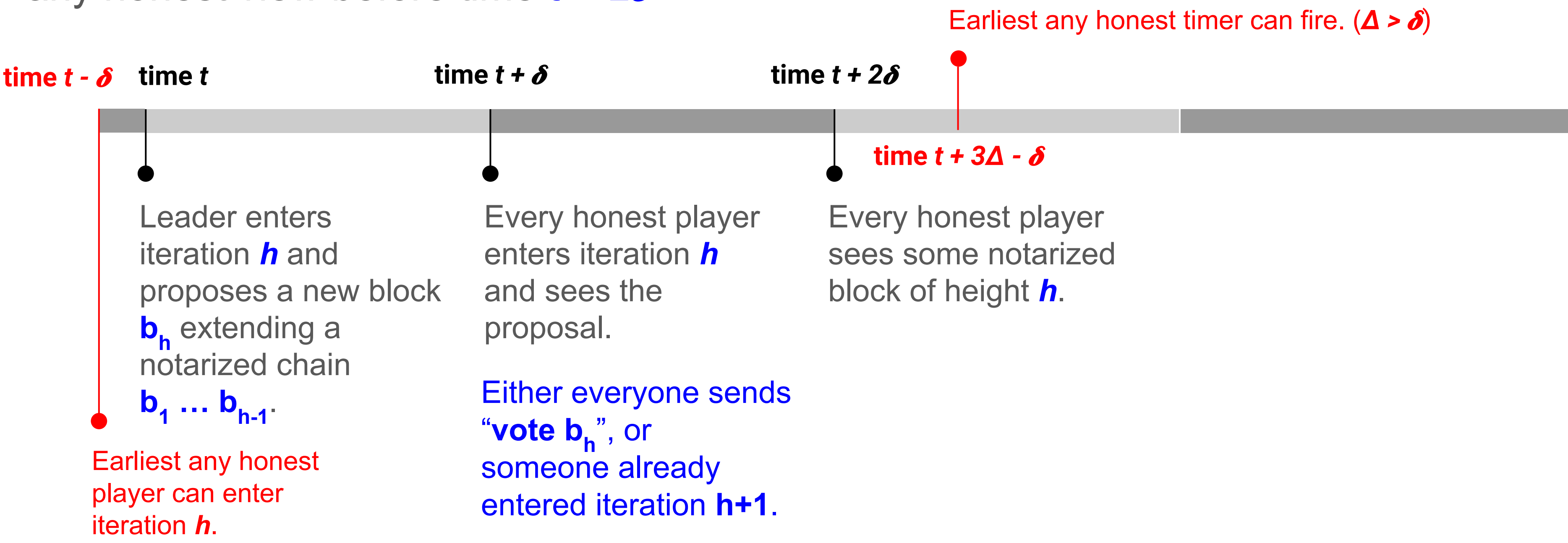
Every honest player enters iteration ***h*** and sees the proposal.

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

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

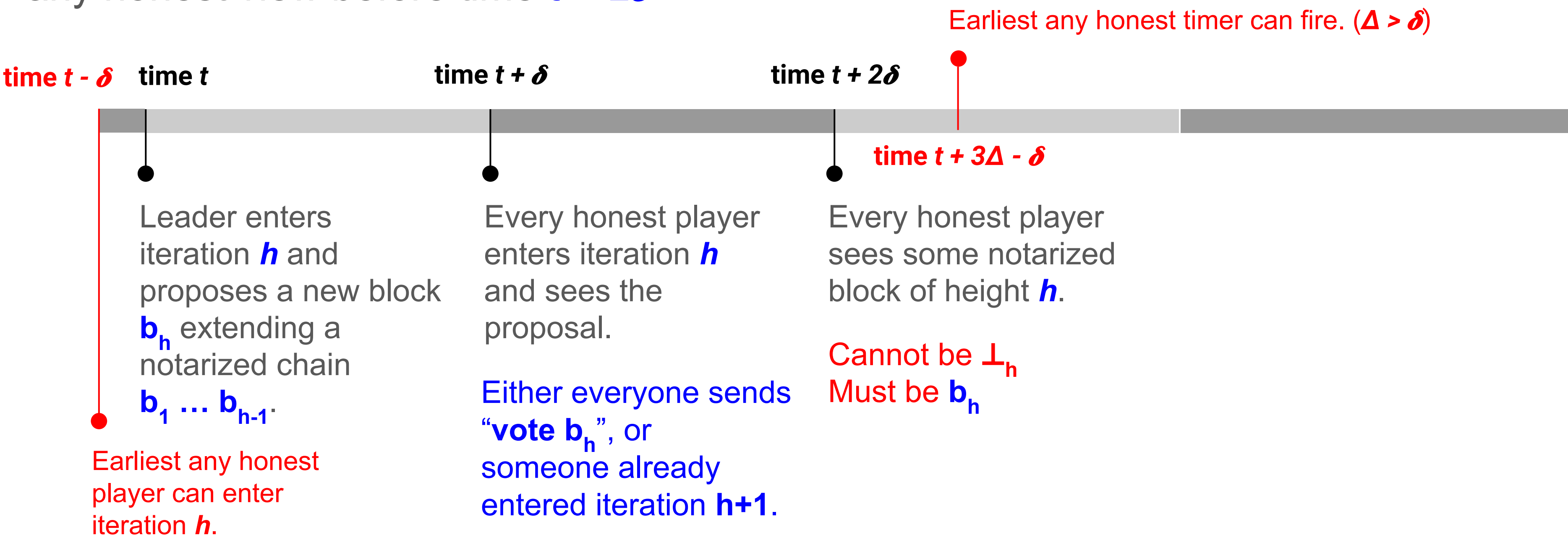
Liveness

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



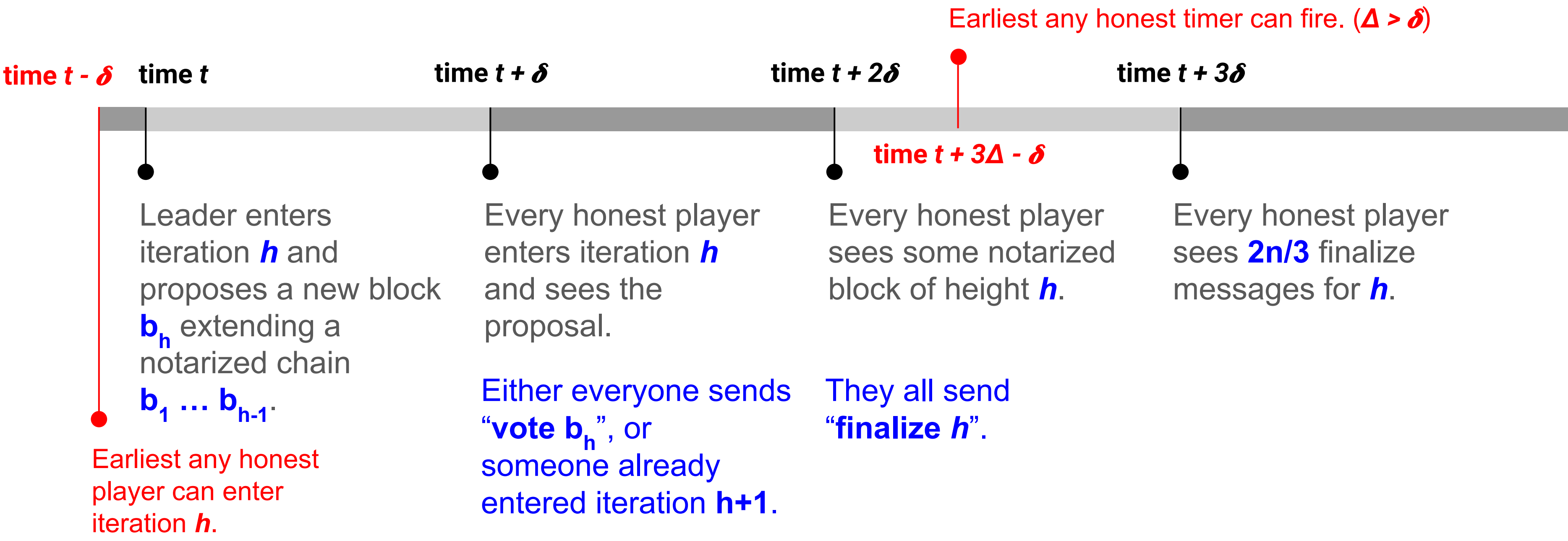
Liveness

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



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 $\mathbf{3\Delta + \delta}$ time.

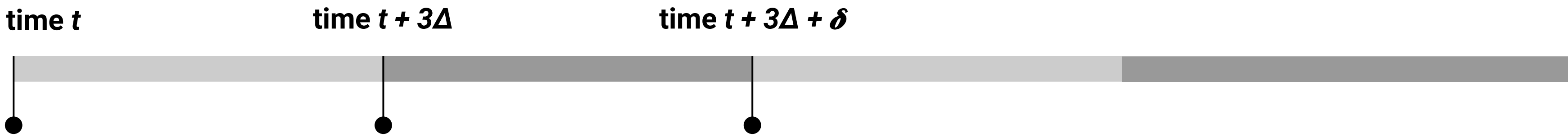
Every honest player has entered iteration ***h***.

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 $\mathbf{3\Delta + \delta}$ 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** $\mathbf{\bot_h}$".

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

In Conclusion

A new consensus protocol

- 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 **$4\delta + \omega(\log \lambda) \cdot (3\Delta + \delta)$**

In Conclusion

A new consensus protocol

- 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 **$4\delta + \omega(\log \lambda) \cdot (3\Delta + \delta)$**

Thank you!

Appendix

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.

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]	$\mathbf{3\delta}$	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

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 [CS20]

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)

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.)