Dynamic Ad Hoc Clock Synchronization

Eurocrypt 2021

Christian Badertscher
IOHK

Peter Gaži
IOHK

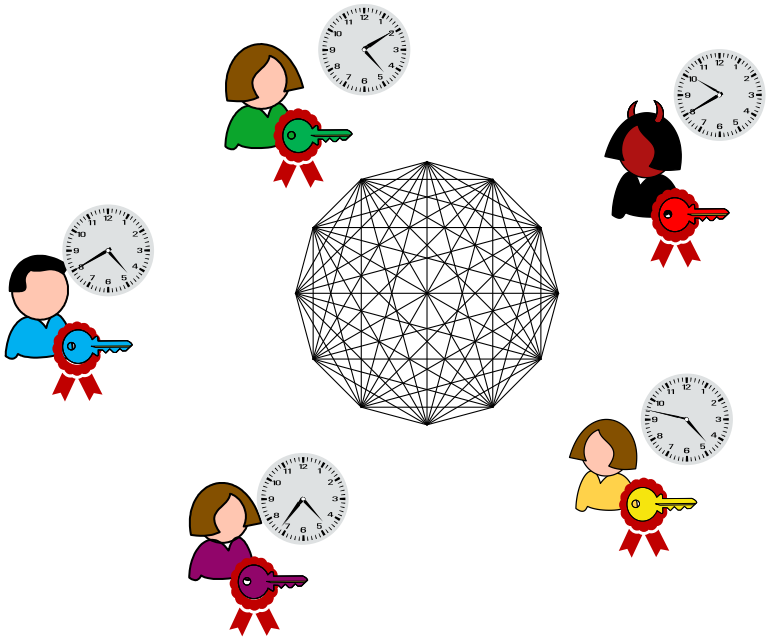
Aggelos Kiayias
IOHK & Univ. of Edinburgh

Alexander Russell
IOHK & Univ. of Connecticut

Vassilis Zikas
Purdue Univ.

https://ia.cr/2019/838

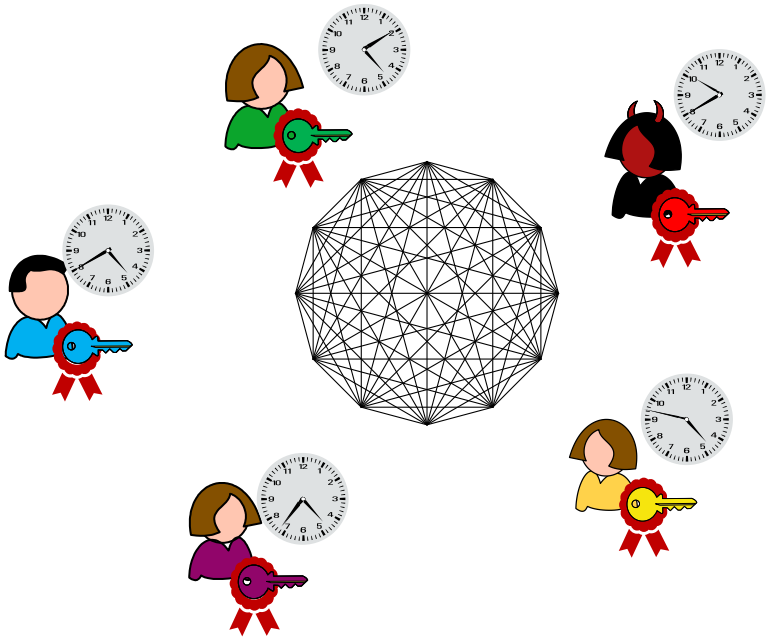
Clock Synchronization



Basic setup:

- **Local clocks** (duration timer) which run at approx. same speed
- PKI / CRS
- Bounded-delay network
- Majority of parties honest

Clock Synchronization

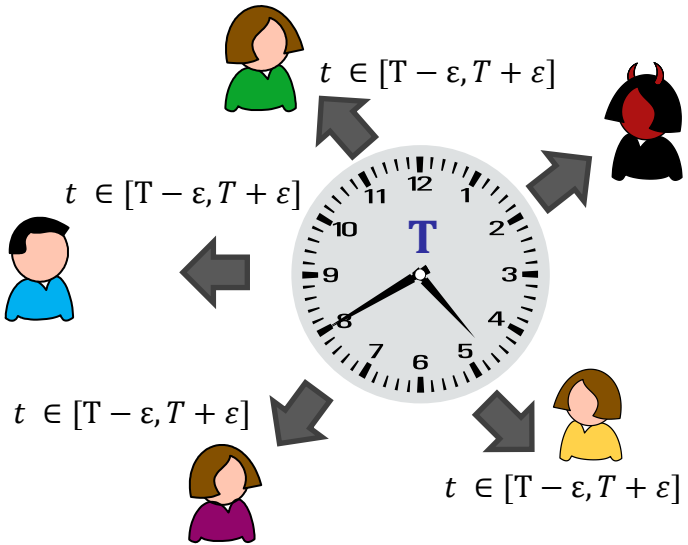


Goal:

Emulate “a global clock”, i.e., compute time values that satisfy:

- Approximate **synchrony**
- **Liveness**
- **Monotonicity** & limited “jumps”

Clock Synchronization



Realization of a **Global Clock Functionality** (with ε-synchrony)

Clock Synchronization – Models

Prior models:

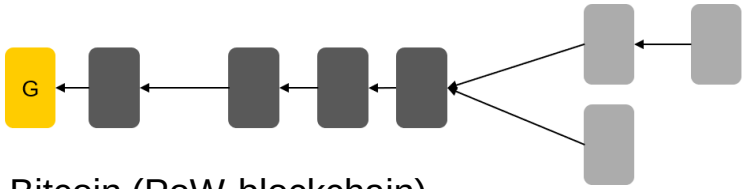
- **Fixed** set of parties in the protocol.
- **All parties active** except for byzantine nodes.
- Relaxation: **“ad hoc” model:** an unknown subset remains inactive
- **Security threshold** relative to size of **active party set.**

Dynamic ad hoc model:

- Number of online/offline parties changes over time.
- No a priori knowledge of participation levels.
- Unannounced disappearance.
- Newcomers need to be bootstrapped.
- **Security threshold** relative to **dynamic participation level.**

Ad hoc model:
Amos Beimel, Yuval Ishai, and Eyal Kushilevitz. Ad hoc PSM protocols: Secure computation without coordination. Eurocrypt 2017

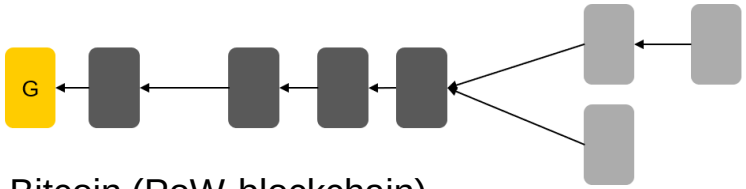
Ideas for Novel Clock Synchronization



Bitcoin (PoW-blockchain)

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

Ideas for Novel Clock Synchronization

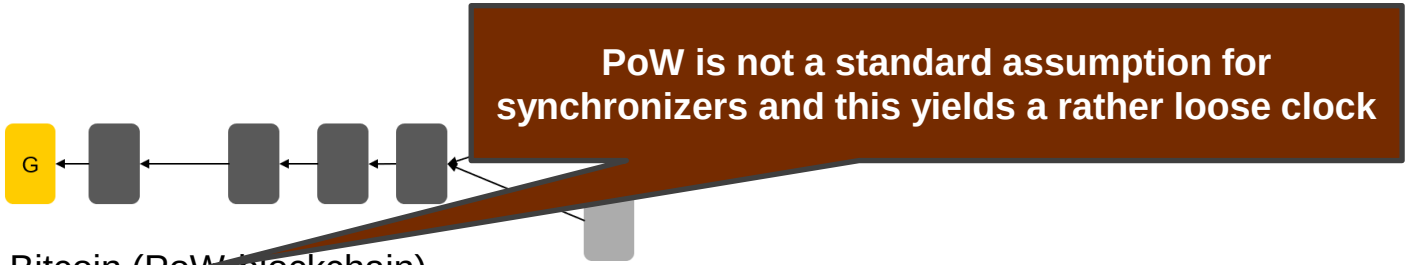


Bitcoin (PoW-blockchain)

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

→ **Block-depth as proxy for time**

Ideas for Novel Clock Synchronization

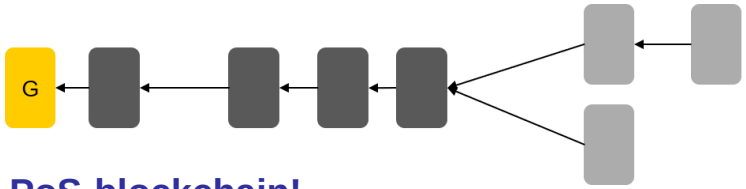


Bitcoin (PoW-blockchain)

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

→ **Block-Depth as Proxy for time**

Ideas for Novel Clock Synchronization

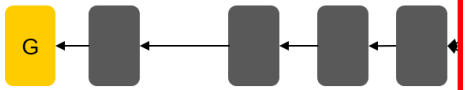


PoS-blockchain!

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

Source of trust: Genesis block

Ideas for Novel Clock Synchronization



PoS-b[illegible]chain!

- **Cons[illegible]ncy**
- **Liveness**

- Require a lower bound on or accurate estimate of participation…
- …or have to assume a reliable timing service!

But maybe we can still use some ideas from those protocols?

Source of trust: Genesis block

Clock Synchronization – Dynamic Ad Hoc Setting

Prior models:

- **Fixed** set of parties in the protocol.
- **All parties active** except for byzantine nodes. **Honest majority.**
- Relaxation: **Ad hoc:** active subset of parties fixed but not known.
- **Security threshold** relative to size of **active party set.**

Dynamic ad hoc:

- Number of online/offline parties changes over time.
- No a priori knowledge of participation levels.
- Unannounced disappearance.
- Newcomers need to be bootstrapped.
- **Security threshold** relative to **dynamic participation level.**

Is Clock Synchronization possible in this setting?

Clock Synchronization – Dynamic Ad Hoc Setting

Prior models:

- **Fixed** set of parties in the protocol.
- **All parties active** except for byzantine nodes. **Honest majority.**
- Relaxation: **Ad hoc:** active subset of parties fixed but not known.
- **Security threshold** relative to size of **active party set.**

Dynamic ad hoc:

- Number of online/offline parties changes over time.
- No a priori knowledge of participation levels.
- Unannounced disappearance.
- Newcomers need to be bootstrapped.
- **Security threshold** relative to **dynamic participation level.**

YES - and even more…

Main Result

We design **the first PoS-blockchain** protocol with the following features:

Main Result

We design **the first PoS-blockchain** protocol with the following features:

1) **It is secure in the dynamic ad hoc setting** (bounded delays, honest majority of stake) and does **not need a global clock** (instead: the weaker assumption of approx. same-speed clocks/timers).

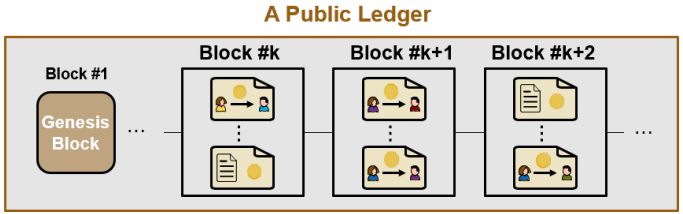
Main Result

We design **the first PoS-blockchain** protocol with the following features:

1) **It is secure in the dynamic ad hoc setting** (bounded delays, honest majority of stake) and does **not need a global clock** (instead: the weaker assumption of approx. same-speed clocks/timers).

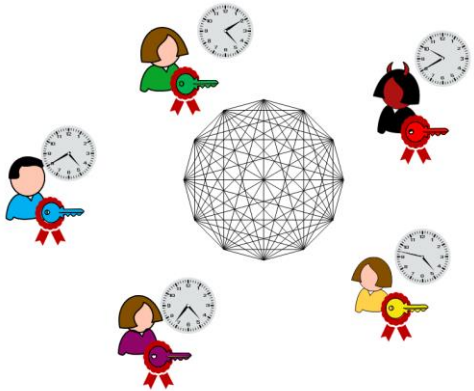
2) **It is a clock synchronizer** and allows parties (and in fact any external observer) to compute time values that are only a bounded distance apart.

Chronos =

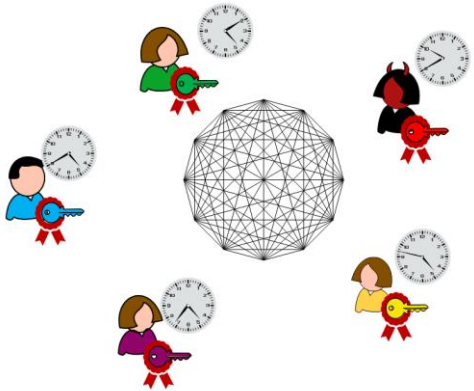


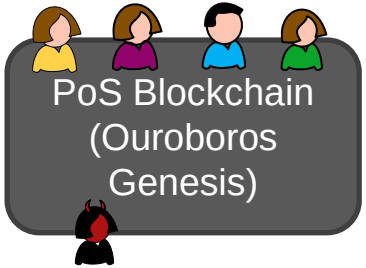
+

Informal Idea of The Solution

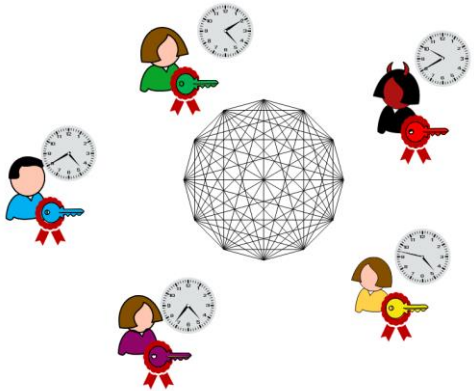


Informal Idea of The Solution

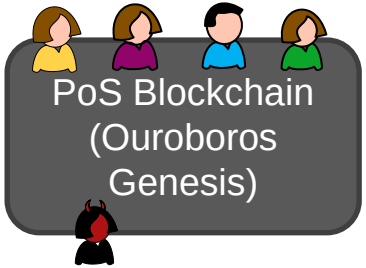




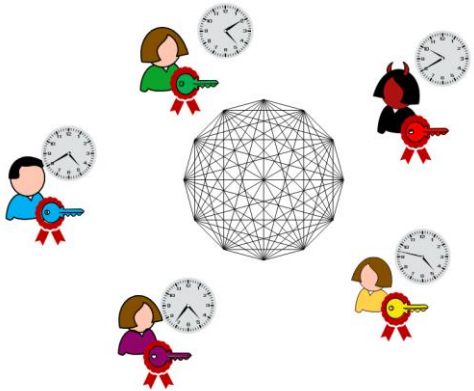
Informal Idea of The Solution



1 **Exchange** authenticated **sync-messages** containing local time stamps (restricted by the PoS lottery).

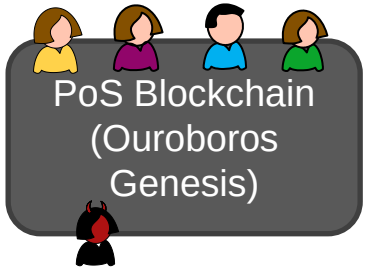


Informal Idea of The Solution

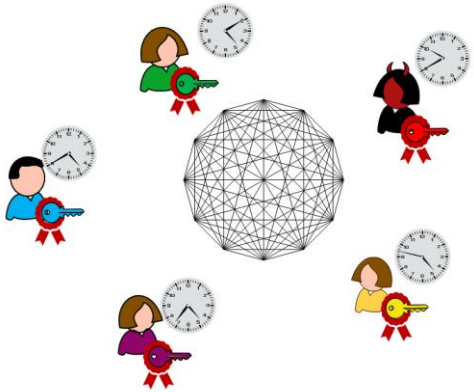


1 Exchange Authenticated Sync-messages containing local time stamps (restricted by the PoS lottery).

2 **Reach consensus** over the set of sync-messages using the blockchain.



Informal Idea of The Solution



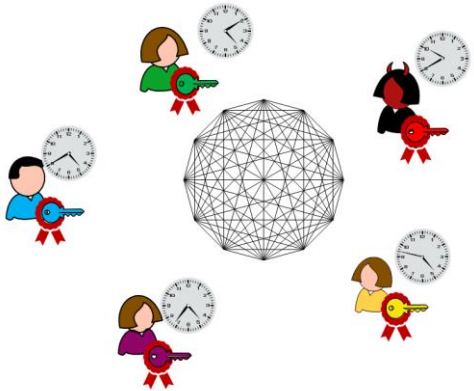
1 Exchange Authenticated Sync-messages containing local time stamps (restricted by the PoS lottery).

2 Reach consensus over the set of sync-messages using the blockchain.

PoS Blockchain (Ouroboros Genesis)

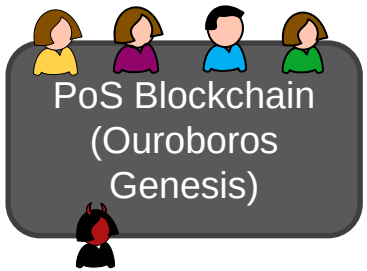
3 **Adjust time value** based on agreed set and observed “delays”.

Informal Idea of The Solution



1 Exchange Authenticated Sync-messages containing local time stamps (restricted by the PoS lottery).

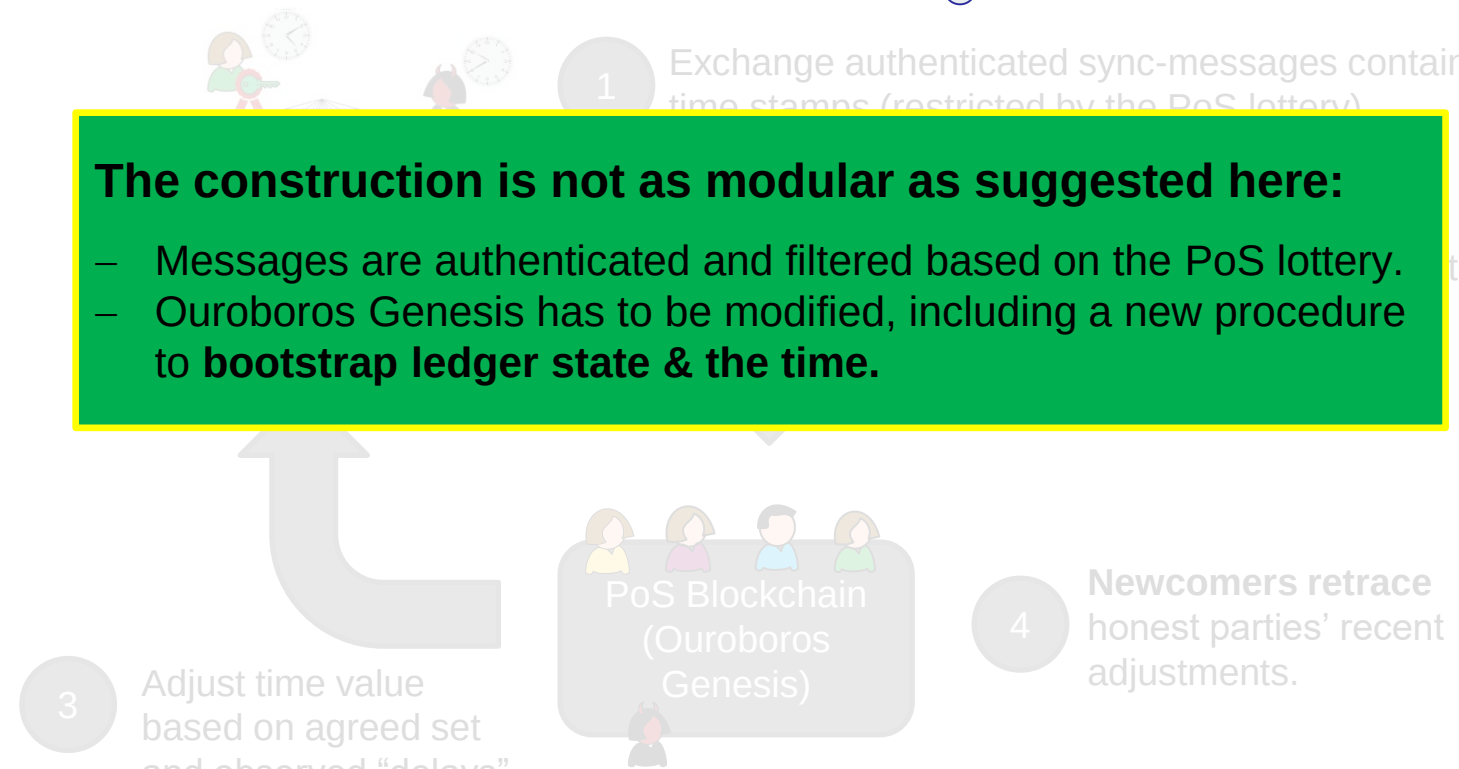
2 Reach consensus over the set of sync-messages using the blockchain.



3 Adjust time value based on agreed set and observed “delays”.

4 **Newcomers retrace** honest parties’ recent adjustments.

Informal Idea of The Solution



The construction is not as modular as suggested here:

- Messages are authenticated and filtered based on the PoS lottery.
- Ouroboros Genesis has to be modified, including a new procedure to **bootstrap ledger state & the time.**

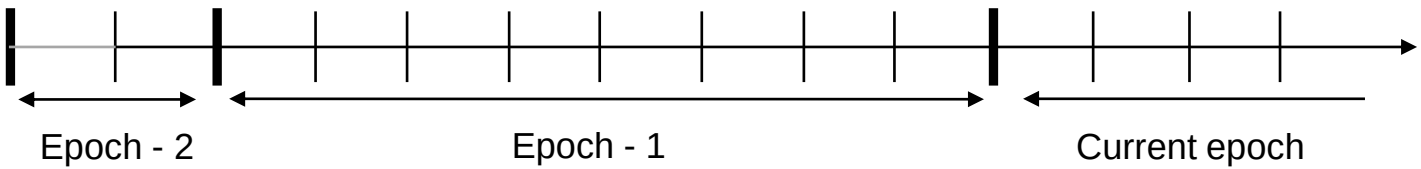
Outline of this Talk

1. Recap of Ouroboros Genesis

2. We see what happens to Genesis when we have local clocks/timers instead of a global clock

3. Presentation of Chronos

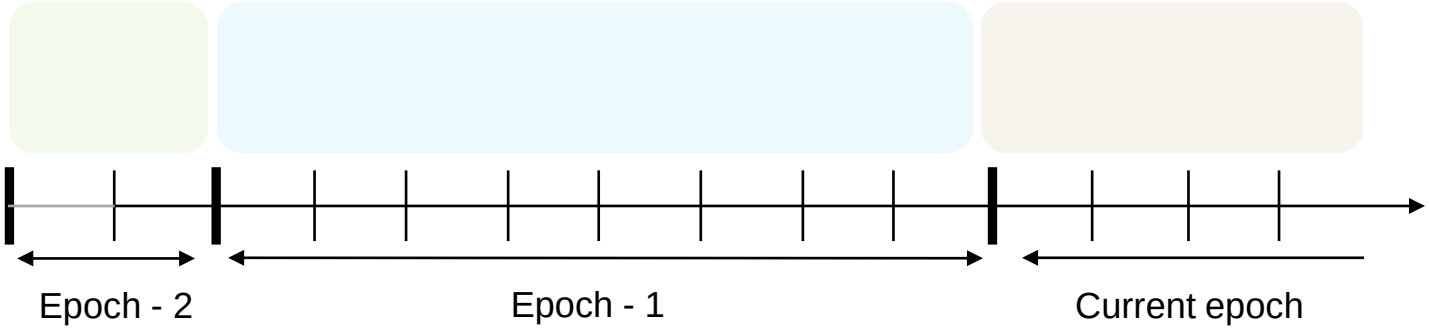
Ouroboros Genesis



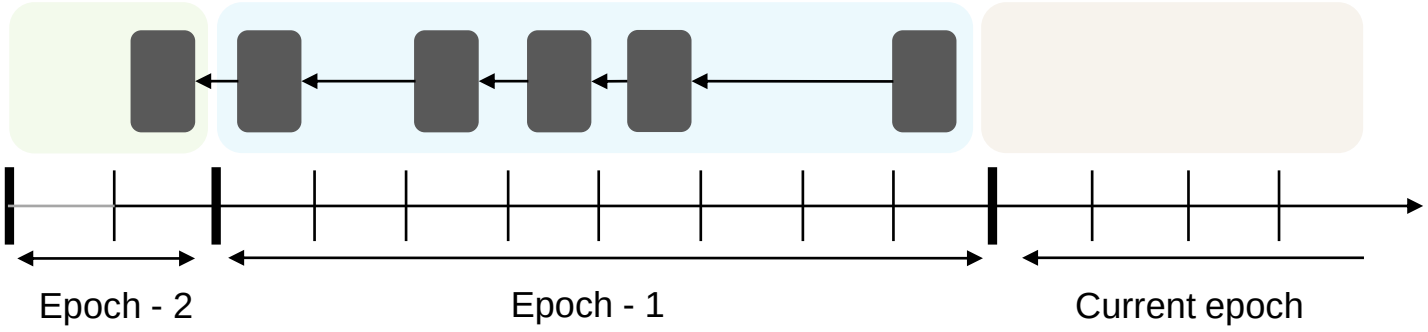
Ouroboros Genesis

- In each slot, each party evaluates slot-leadership.
  Private election, proportional to stake, including recent randomness from the chain

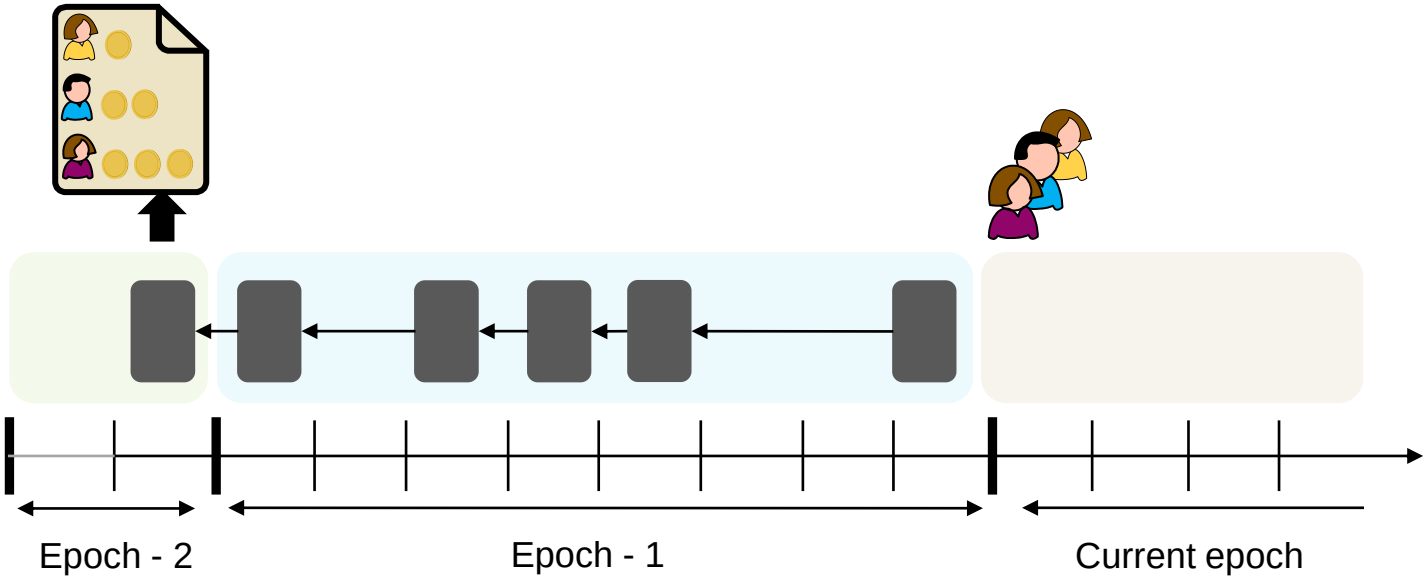
- A slot leader extends its most preferred chain by creating the block for this slot.



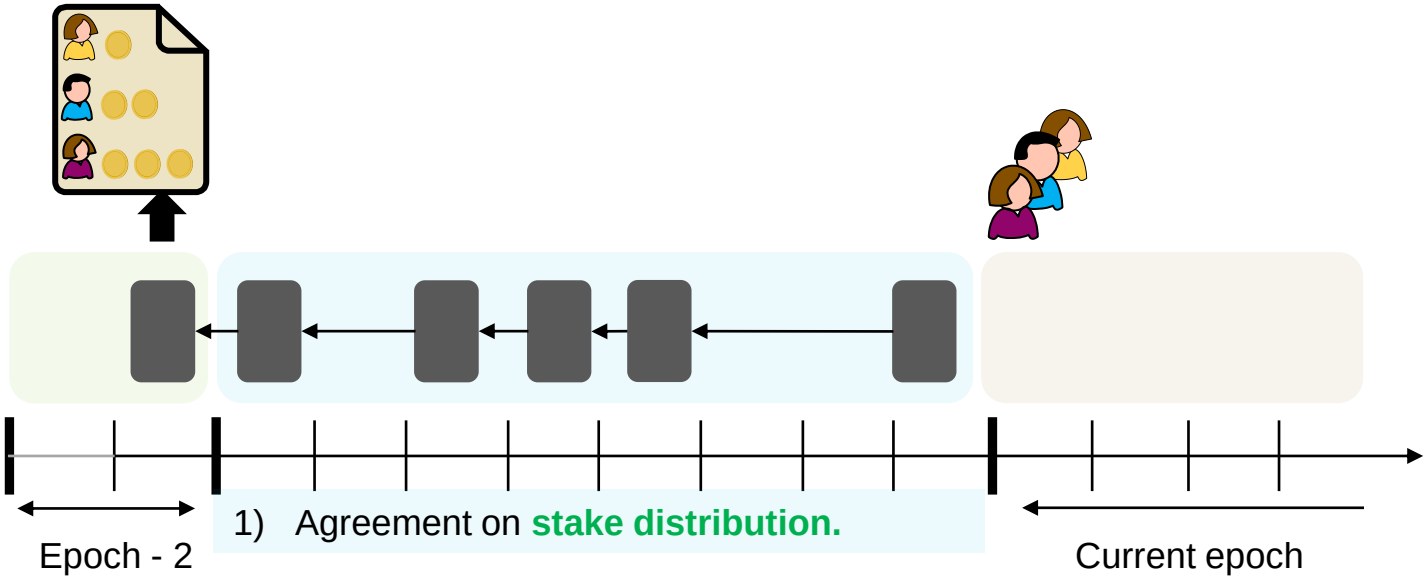
Ouroboros Genesis



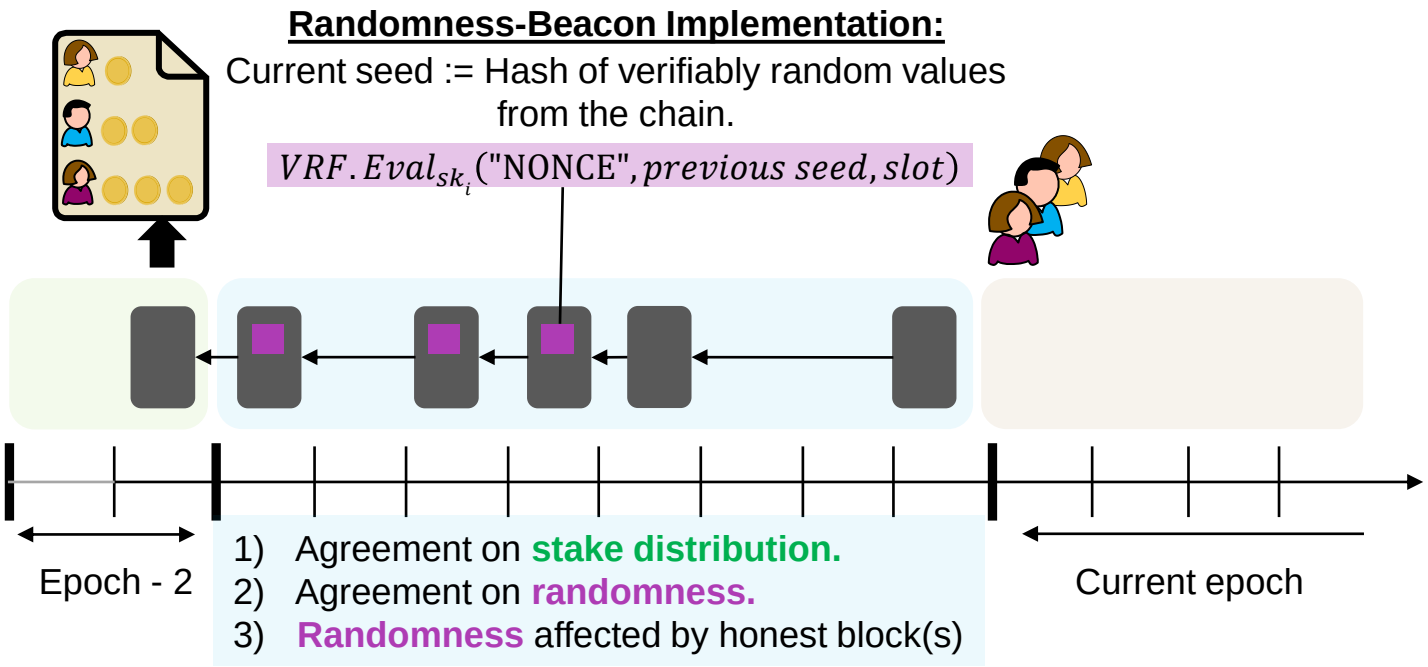
Ouroboros Genesis



Ouroboros Genesis



Ouroboros Genesis

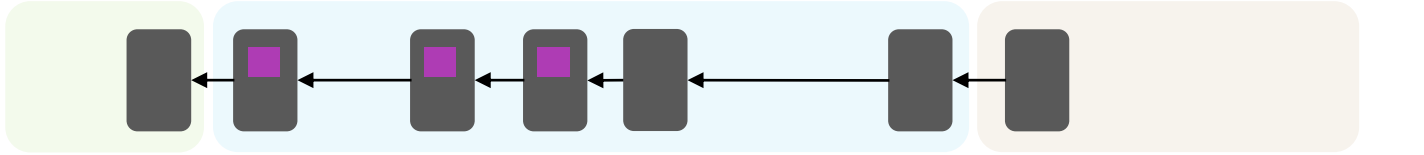


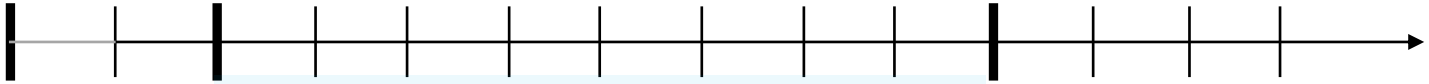
Ouroboros Genesis

Lottery in each slot:

A party i is leader if and only if $VRF.Eval_{sk_i}(\text{"TEST"}, \boldsymbol{seed}, slot) < T(\boldsymbol{stake_i})$

- Empty slots possible
- Multiple leaders possible
- Leadership proof from VRF.



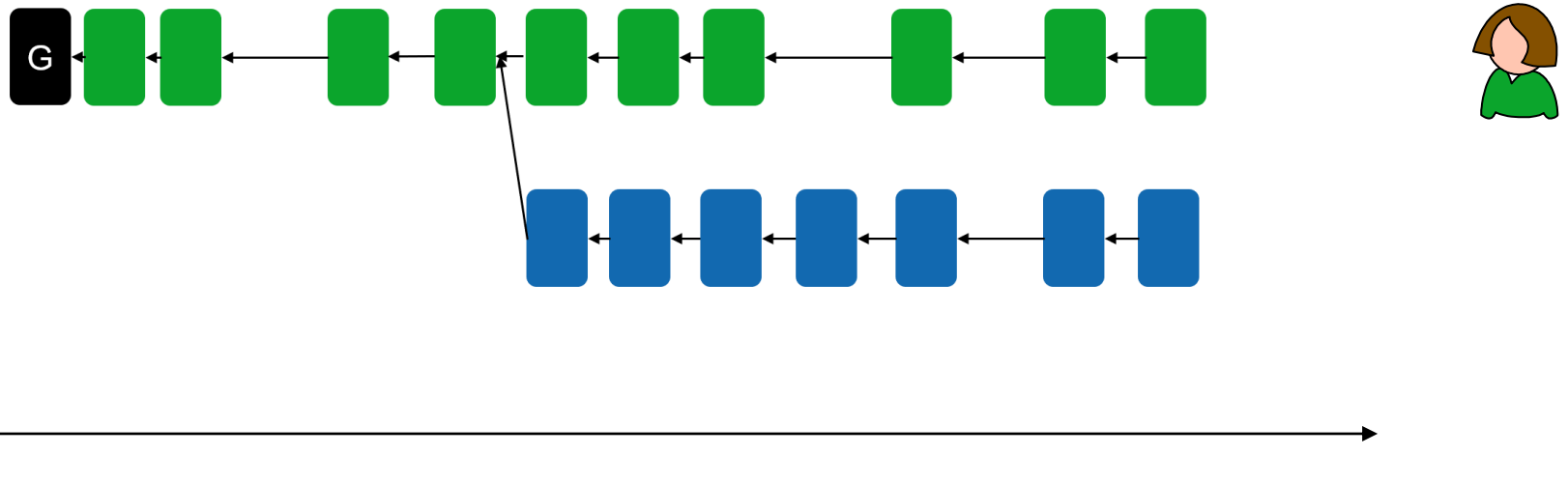


Epoch - 2

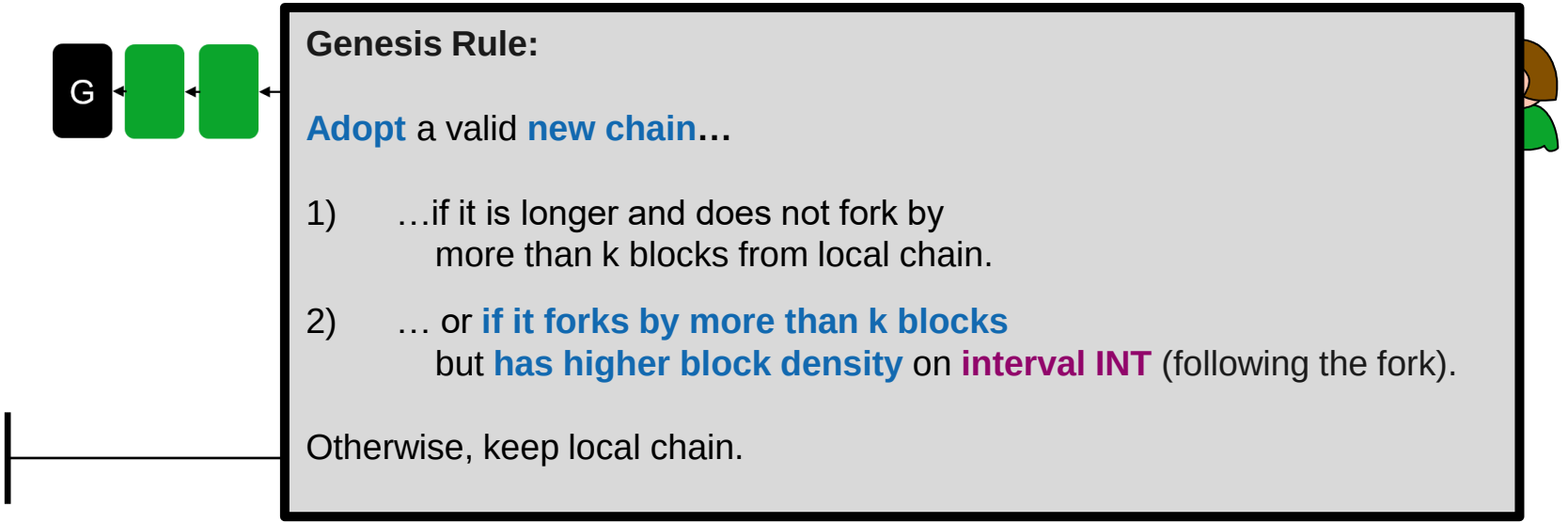
1) Agreement on **stake distribution.**
2) Agreement on **randomness.**
3) **Randomness** affected by honest block(s)

Current epoch

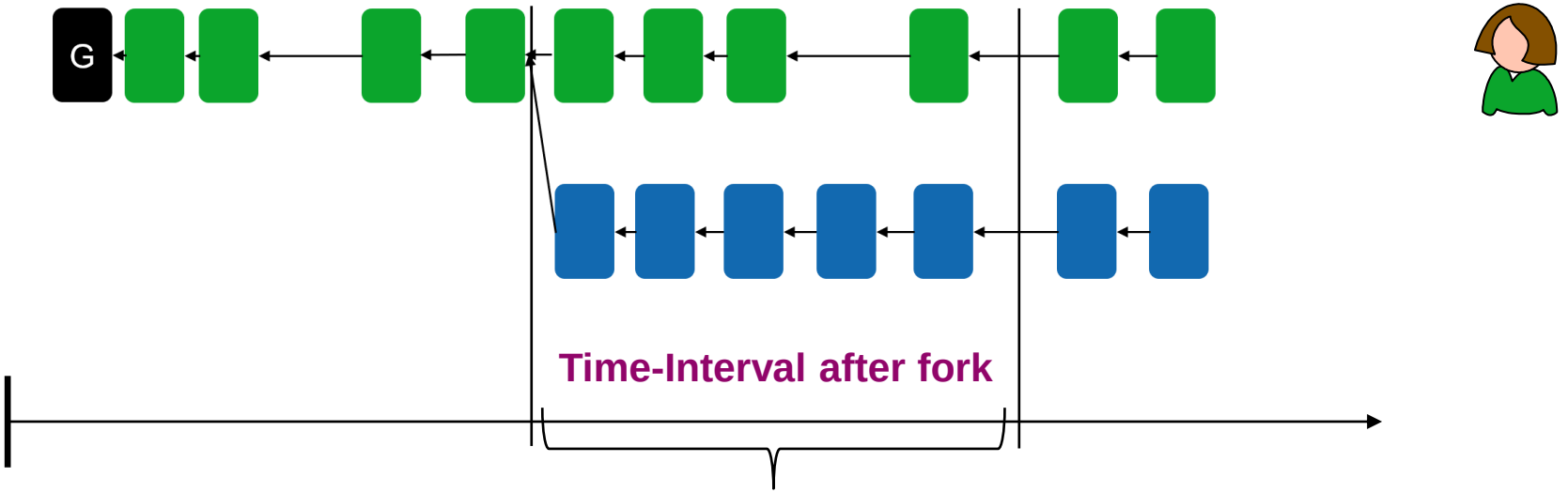
Ouroboros Genesis – Chain Selection



Ouroboros Genesis – Chain Selection

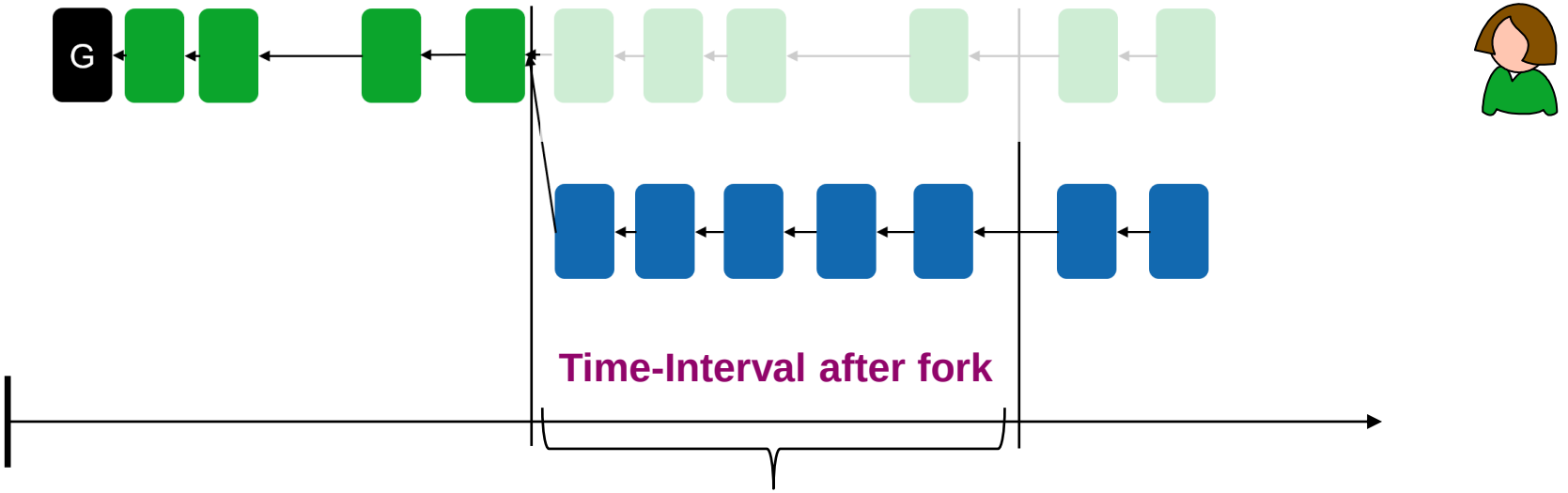


Ouroboros Genesis – Chain Selection



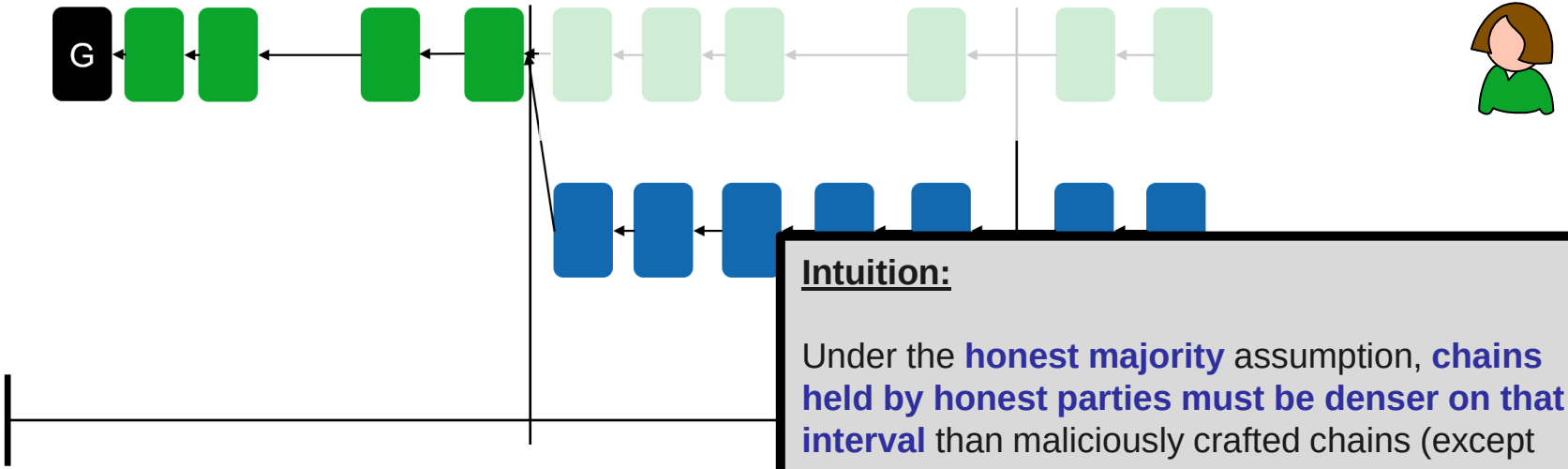
(Protocol parameter: **size** of INT)

Ouroboros Genesis – Chain Selection



(Protocol parameter: **size** of INT)

Ouroboros Genesis – Chain Selection



Intuition:

Under the **honest majority** assumption, **chains held by honest parties must be denser on that interval** than maliciously crafted chains (except with negligible probability).

Reference:
C. Badertscher, P. Gaži, A. Kiayias, A. Russell, V. Zikas.
Ouroboros Genesis: Composable Proof-of-Stake Blockchains with Dynamic Availability. CCS 18.

Ouroboros Genesis – Useful Property

Using the Genesis chain-selection rule, a newly **joining party** will **adopt a chain** with large common prefix w.r.t. honest parties. No other advice than **the genesis block** is needed.

Ouroboros Genesis – Useful Property

Using the Genesis chain-selection rule, a newly **joining party** will **adopt a chain** with large common prefix w.r.t. honest parties. No other advice than **the genesis block** is needed.

Reason:

- *The genesis rule establishes density as a proxy for honesty.*
- *Thus, all parties' chains contain the stable prefix held by alert parties, because that prefix wins any genesis (density) comparison.*

Ouroboros Genesis – Useful Property

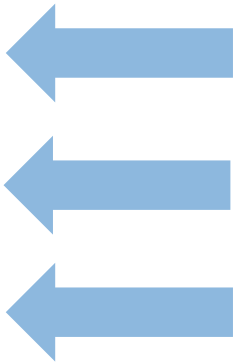
This argument does not assume reliable time information!

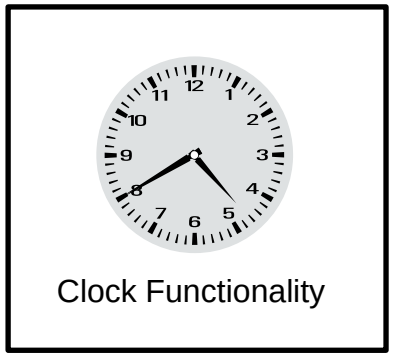
n rule, a newly **joining party** will
n prefix w.r.t. honest parties. No
ock is needed.

Reason:

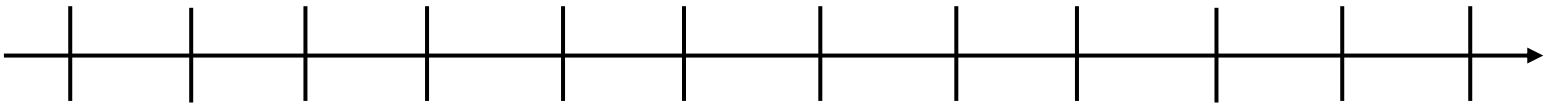
- *The genesis rule establishes density as a proxy for honesty.*
- *Thus, all parties' chains contain the stable prefix held by alert parties, because that prefix wins any genesis (density) comparison.*

So far: Availability of a Global Clock

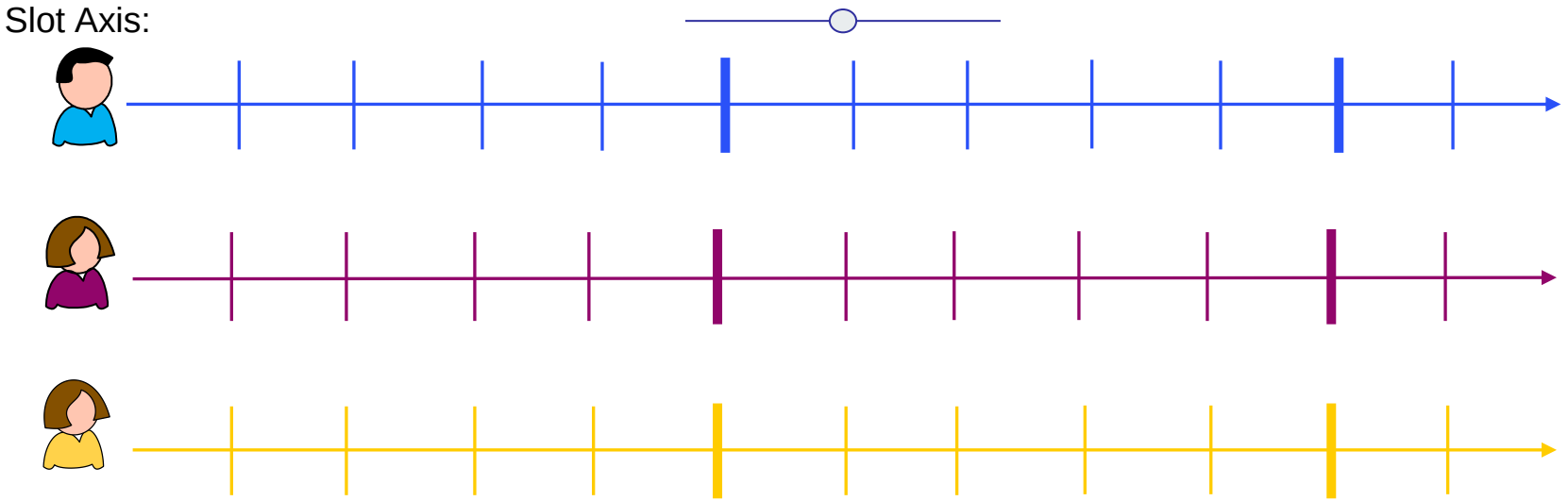


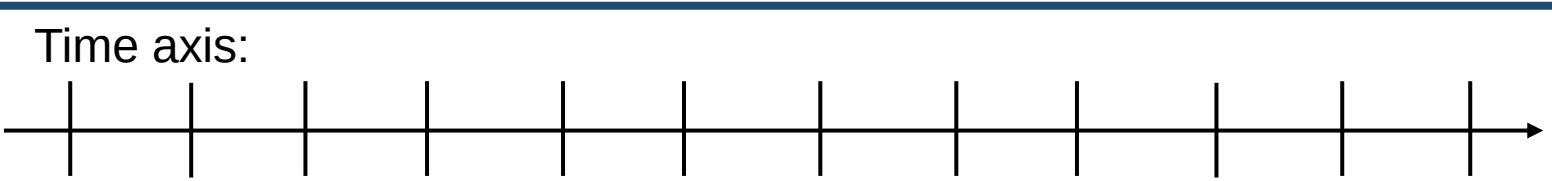


Time axis:

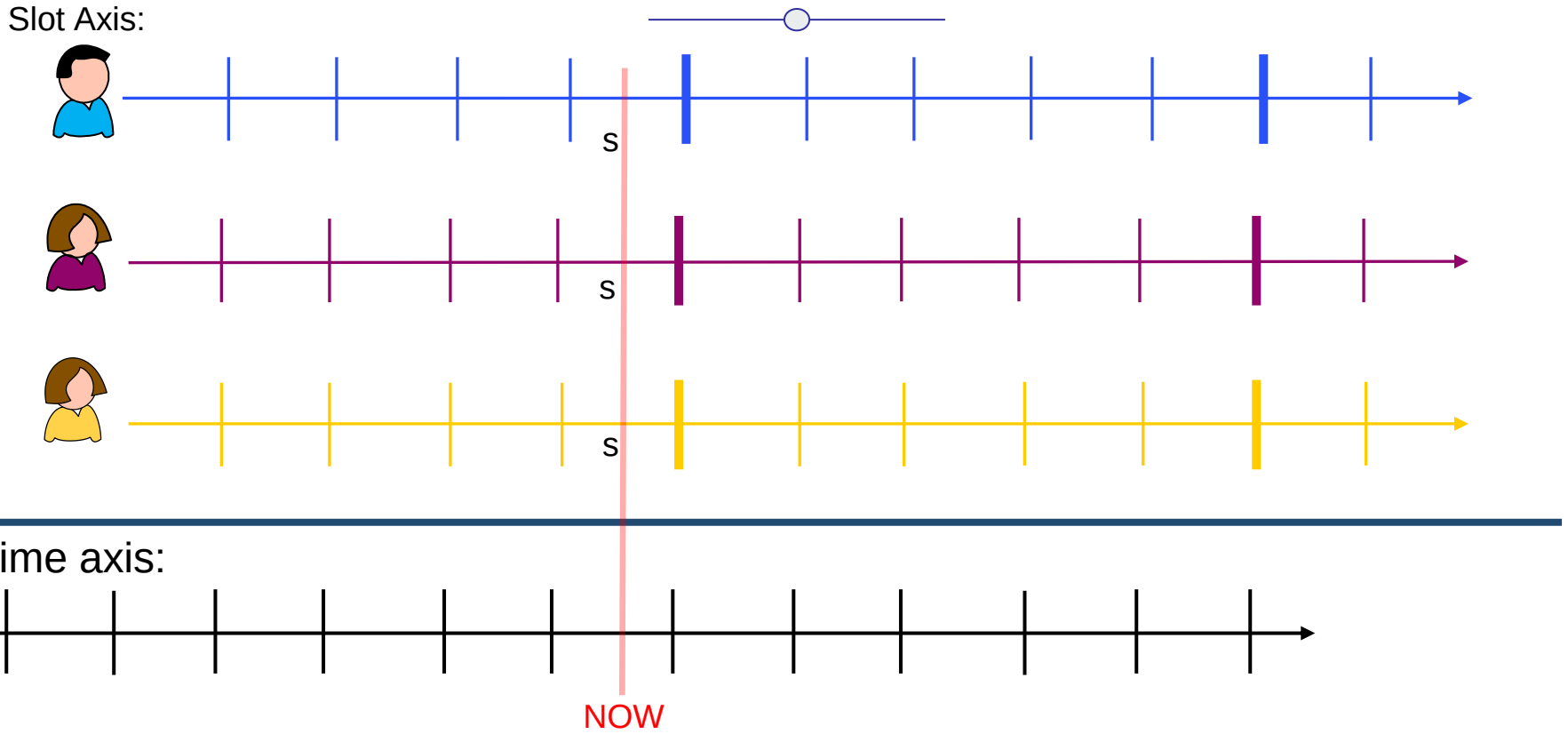


So far: Availability of a Global Clock



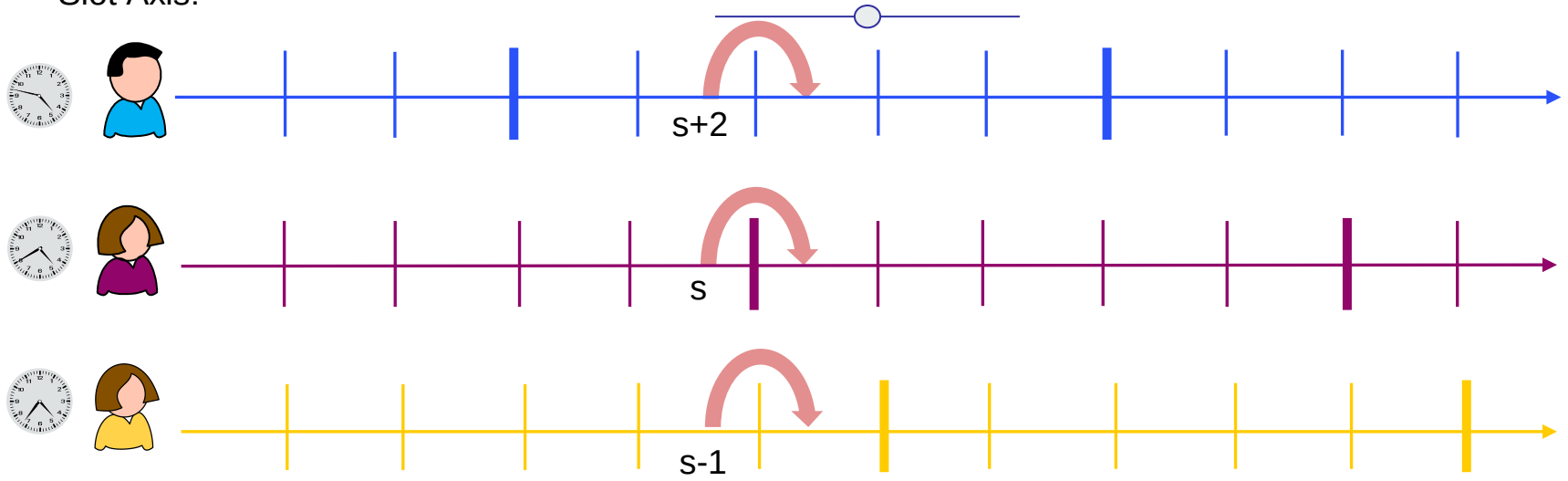


So far: Availability of a Global Clock

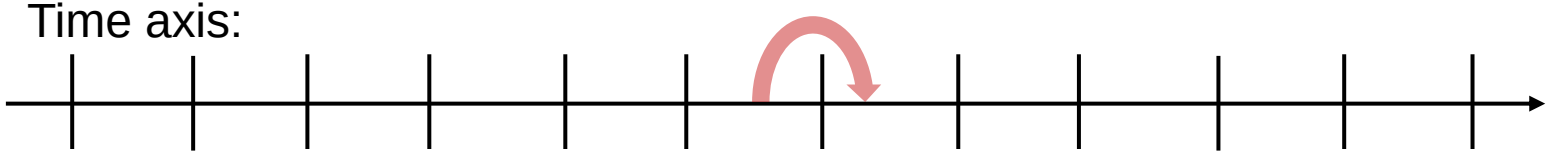


We want: Same-Speed Assumption

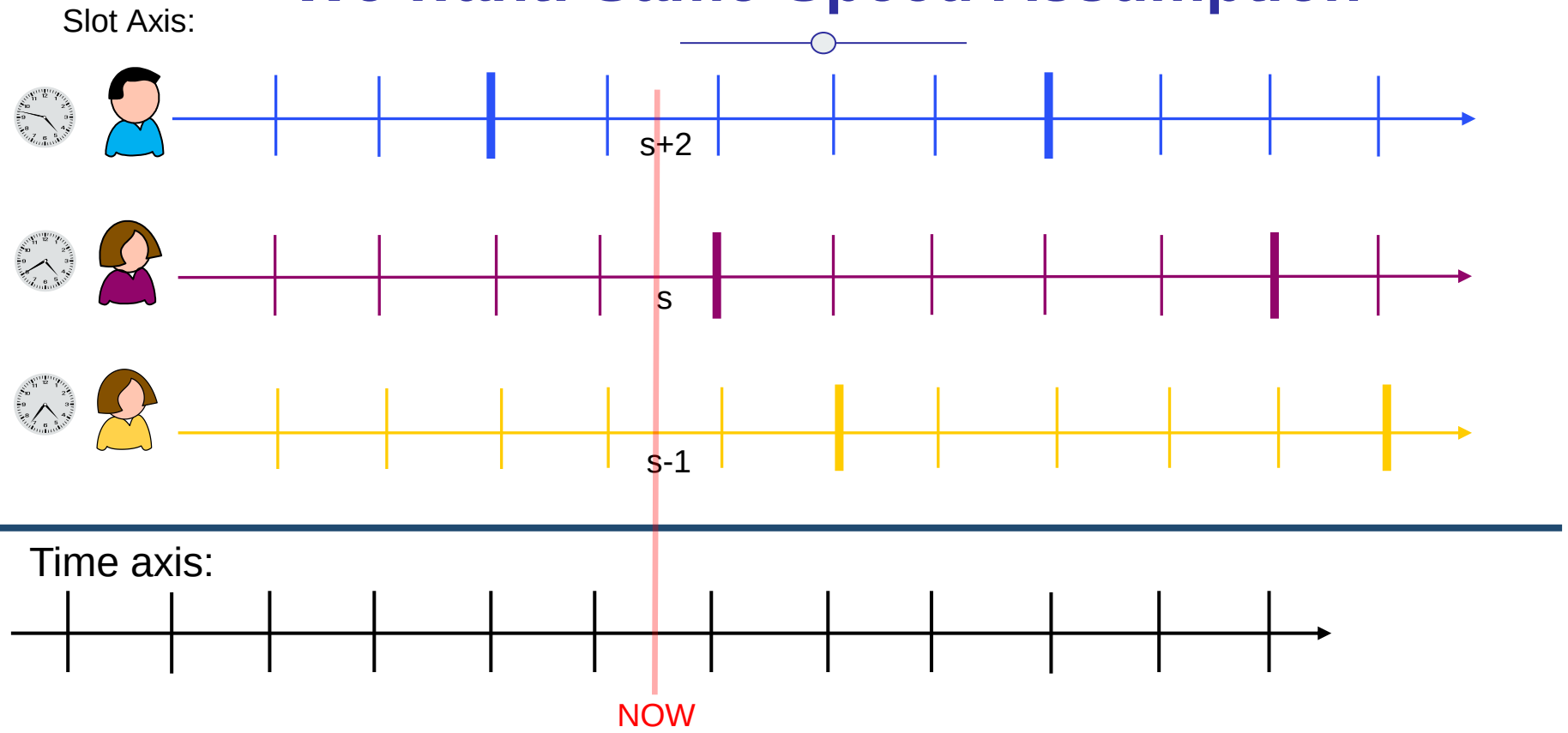
Slot Axis:



Time axis:



We want: Same-Speed Assumption



Genesis: Situation not that bad…

If we manage to keep **all honest parties somewhat close** we're kind of good.

Genesis: Situation not that bad…

If we manage to keep **all honest parties somewhat close** we're kind of good.

- **Close together:** Honest parties' timestamps are never more than Δ apart (order of network delay + local clock drift).
- **Small adjustments** needed to Ouroboros Genesis to deal with future chains

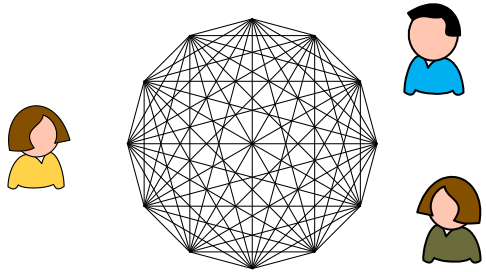
Genesis: Situation not that bad…

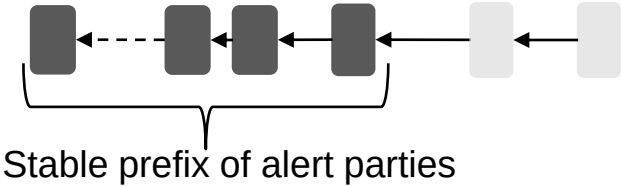
If we manage to keep **all honest parties somewhat close** we're kind of good.

- **Close together:** Honest parties' timestamps are never more than Δ apart (order of network delay + local clock drift).
- **Small adjustments** needed to Ouroboros Genesis to deal with future chains

→ (Approx.) Same-speed: Initial parties do stay close enough.
→ **Joining parties** have a harder life…

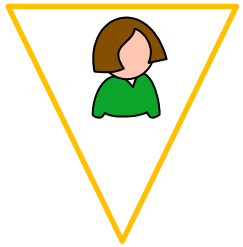
Genesis: Situation not that bad…

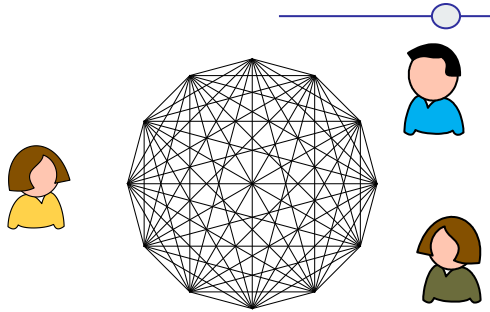


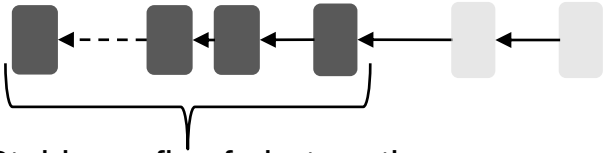


Genesis: Situation not that bad…

Joining party:

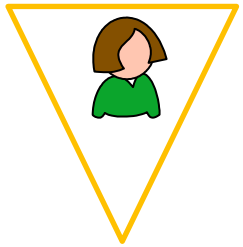


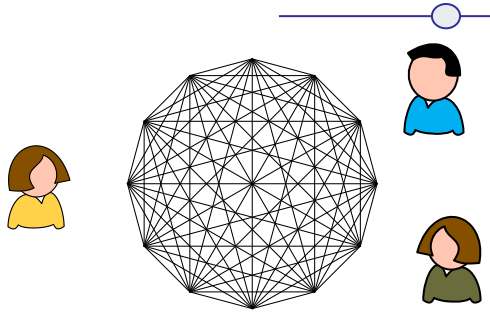


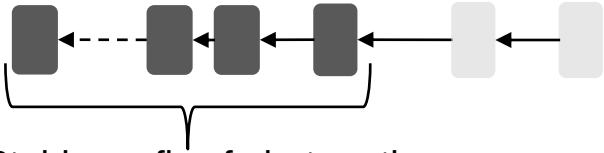


Genesis: Situation not that bad…

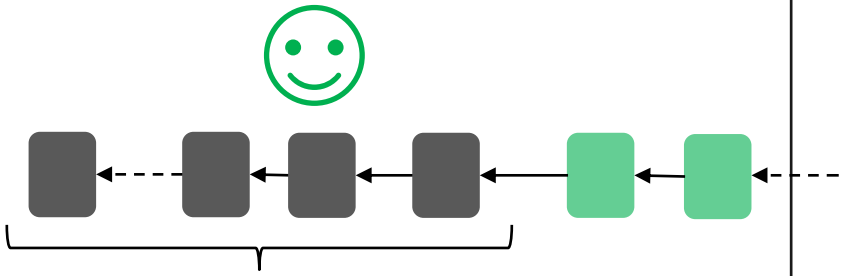
Joining party:







Stable prefix of alert parties

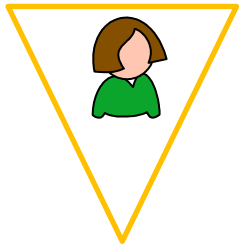


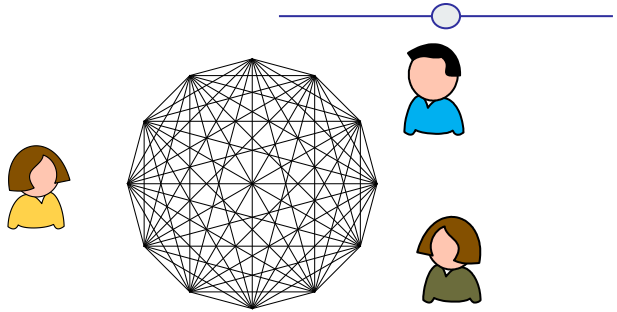
Genesis chain selection rule:

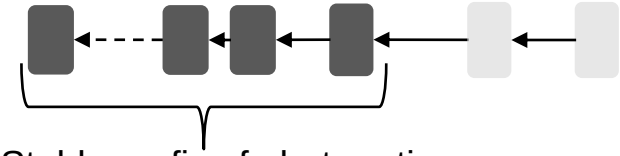
- **Good prefix** is the **densest prefix**
- **Genesis rule prefers** densest prefix

Genesis: Situation ~~not that~~ bad…

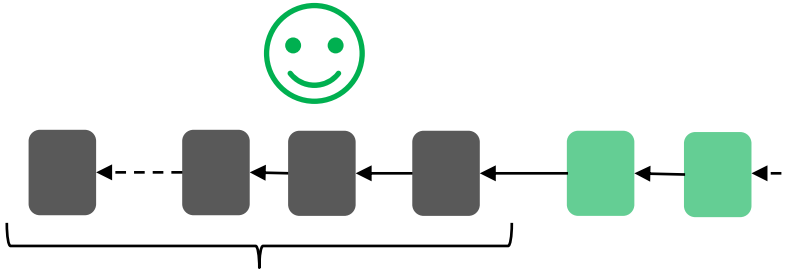
Joining party:





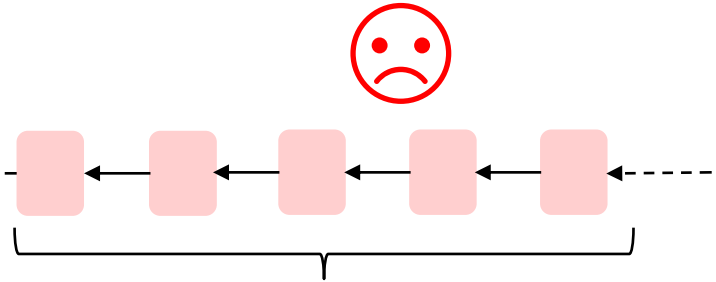


Stable prefix of alert parties



Genesis chain selection rule:

- **Good prefix** is the **densest prefix**
- **Genesis rule prefers** densest prefix



- **No cut-off** possible
- **No reliable** ledger state

…. without time-synchronization

Chronos - Overview

Chronos - Overview

- **Alert parties:** Execute Ouroboros Genesis, broadcast **sync-beacons** and **leave evidence** of beacons in the blockchain.

Chronos - Overview

- **Alert parties:** Execute Ouroboros Genesis, broadcast **sync-beacons** and **leave evidence** of beacons in the blockchain.

- They perform **local-clock adjustments** based on the evidence in the chain.
  - Small adjustments to local clocks at the end of an epoch
  - Based on the evidence left in the chain.

Chronos - Overview

- **Alert parties:** Execute Ouroboros Genesis, broadcast **sync-beacons** and **leave evidence** of beacons in the blockchain.

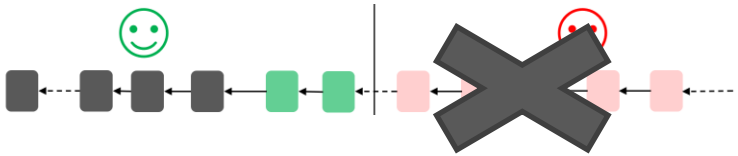
- They perform **local-clock adjustments** based on the evidence in the chain.
  - Small adjustments to local clocks at the end of an epoch
  - Based on the evidence left in the chain.

- **Joining parties:** Once hooked up on a prefix of the densest chain, record beacons and **retrace the evidence.**
  - Perform the very same clock adjustments to compute a good timestamp

Chronos - Overview

This **not only** yields **a good time-synchronizer** but also **a good blockchain**:

→ Once the time is synchronized with alert parties, the ledger state can be computed



- **Joining parties:** Once hooked up on a prefix of the densest chain, record beacons and **retrace the evidence.**
  - Perform the very same clock adjustments to compute a good timestamp

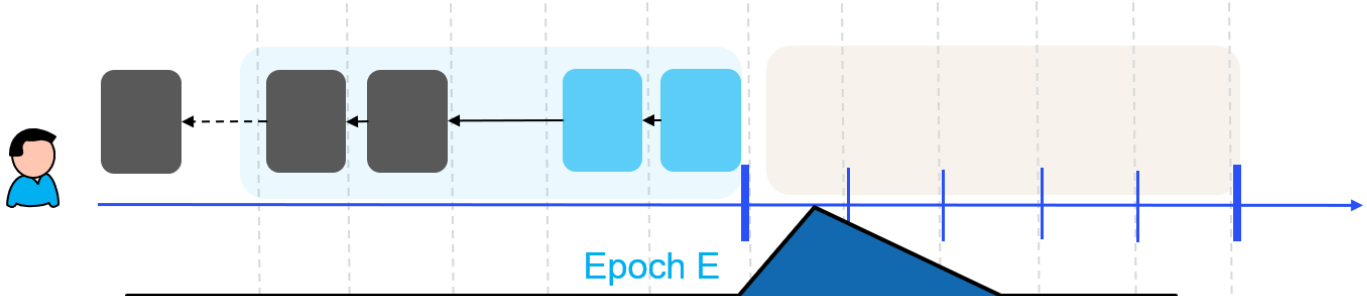
Chronos

- **Alert parties:** Execute Ouroboros Genesis, broadcast **sync-beacons** and **leave evidence** of beacons in the blockchain.

- They perform **local-clock adjustments** based on the evidence in the chain.
  - Small adjustments to local clocks at the end of an epoch
  - Based on the evidence left in the chain.

- **Joining parties:** Once hooked up on a prefix of the densest chain, record beacons and **retrace the evidence.**
  - Perform the very same clock adjustments to compute a good timestamp

Chronos – Sync-Beacons



Additional "Timing Lottery" in the first part of the epoch:

- IF $VRF_{sk_i}(\text{"SYNC"}, seed, slot) < T(stake_i)$ THEN
  - Broadcast Sync-Beacon:

- **Normal slot leaders** pack transactions **+ beacons**.

Chronos

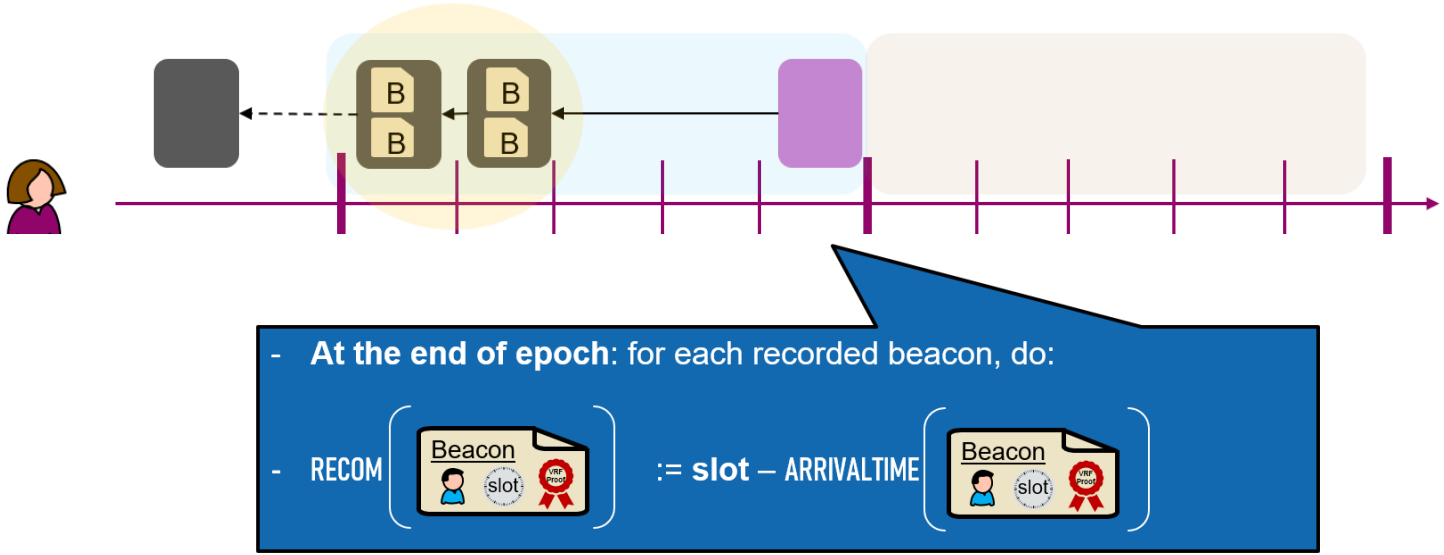
- **Alert parties:** Execute Ouroboros Genesis, broadcast **time-beacons** and **leave evidence** of beacons in the blockchain.

- They perform **local-clock adjustments** based on the evidence in the chain.
  - Small adjustments to local clocks at the end of an epoch
  - Based on the evidence left in the chain.

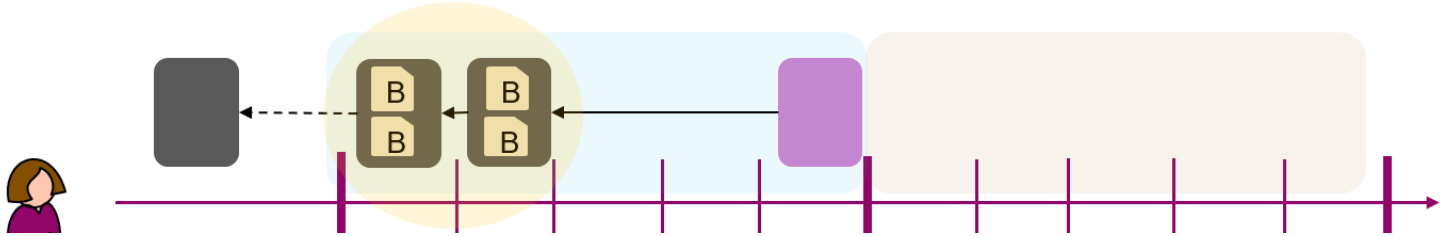
- **Joining parties:** Once hooked up on a prefix of the densest chain, record beacons and **retrace the evidence.**
  - Perform the very same clock adjustments to compute a good timestamp

Chronos: Synchronization Procedure

- **Throughout the epochs:** Record the arrival times of valid beacons (filter out duplicates, invalid ones etc.)
- **At the end of each epoch:** Compute local clock-adjustment.



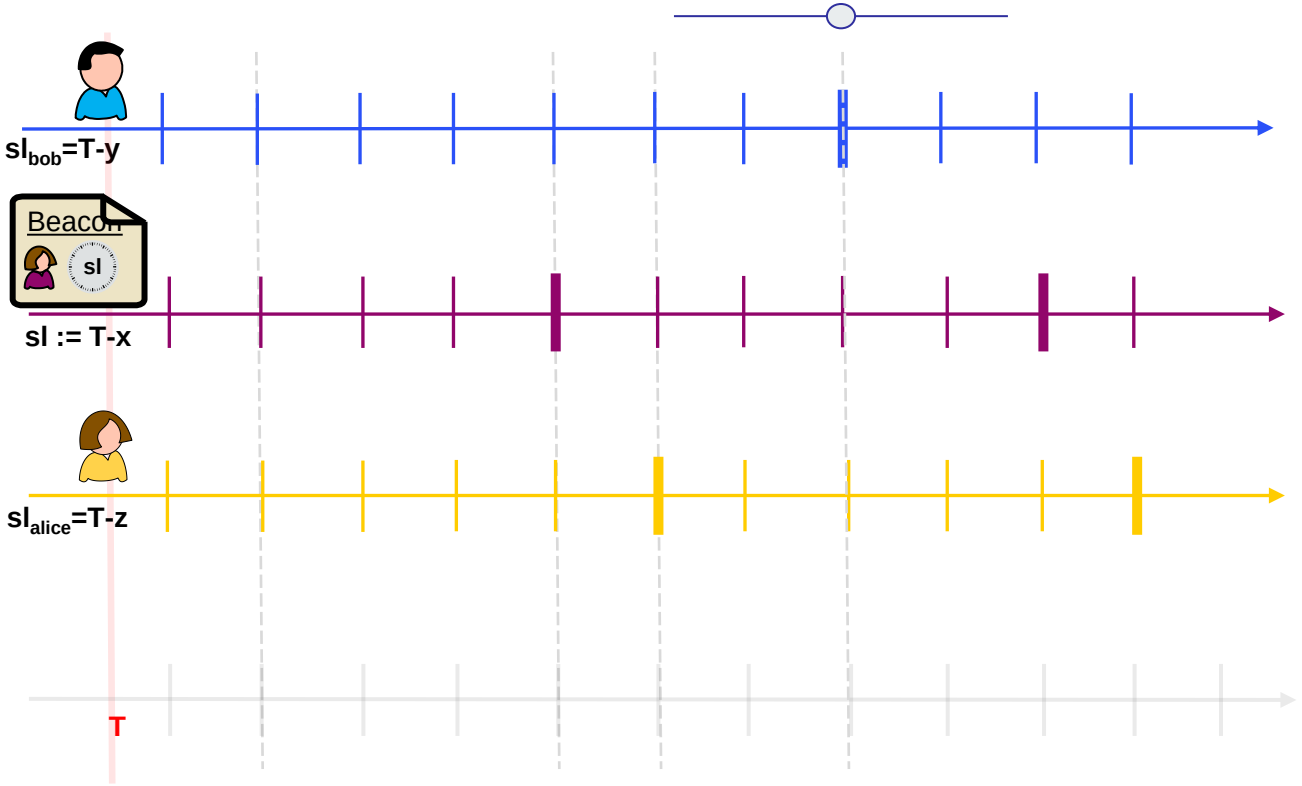
Chronos: Synchronization Procedure



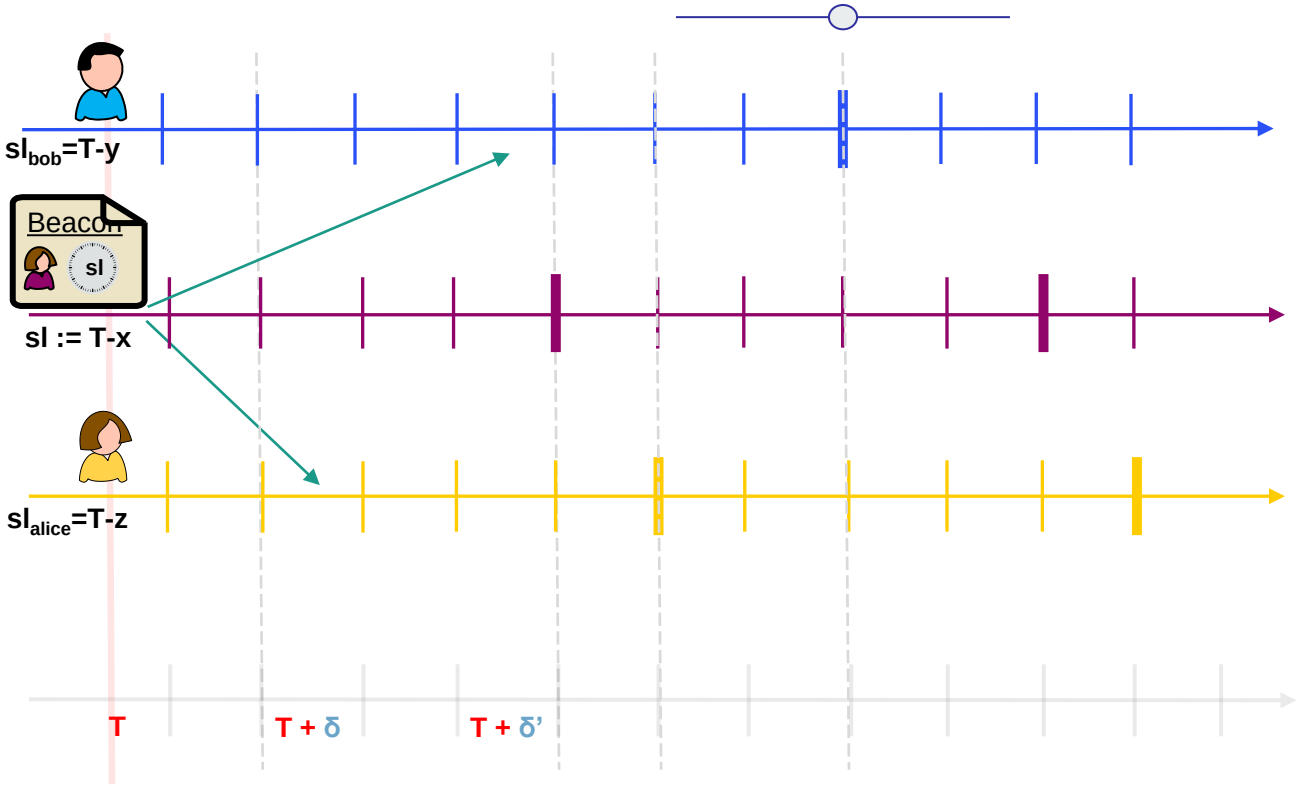
Adjustment rule:

- At the end of epoch: **add** the **median of recommendations** to local time:

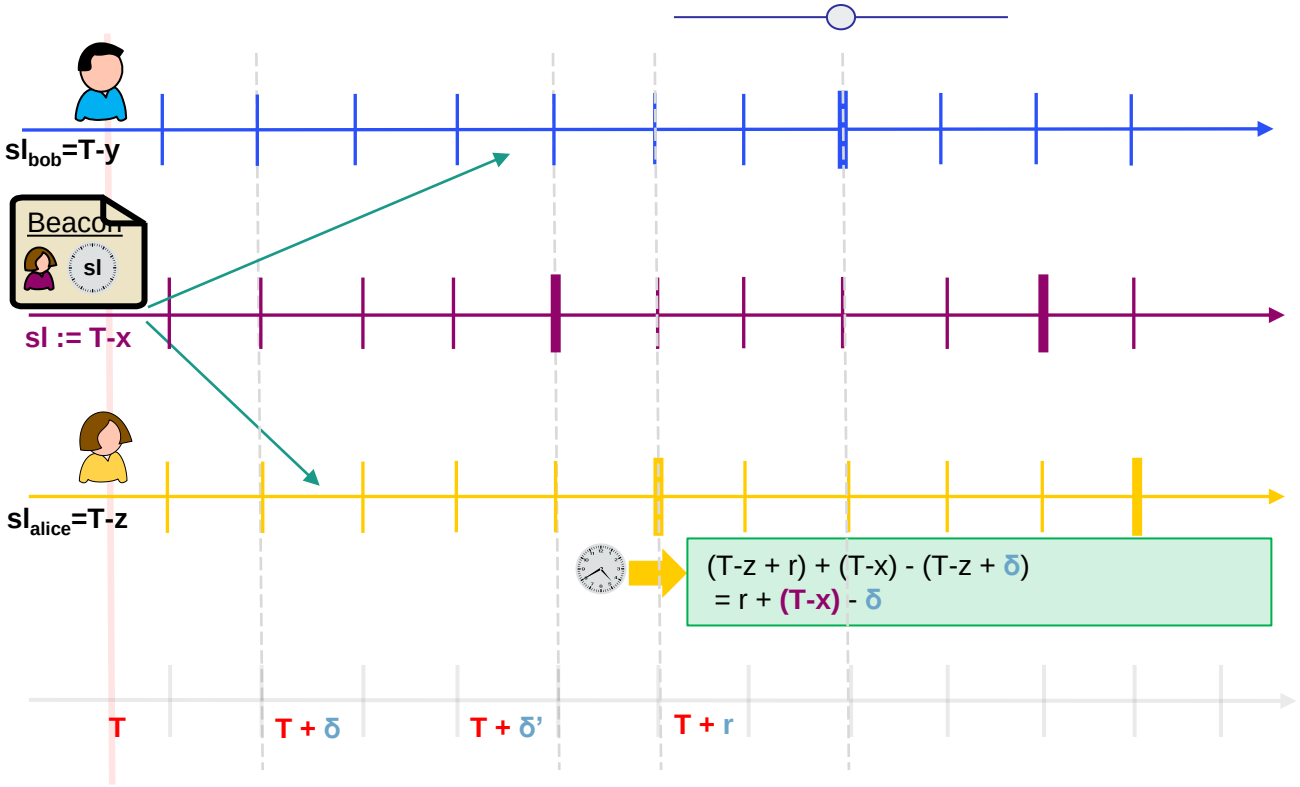
Synchronization Procedure - Example



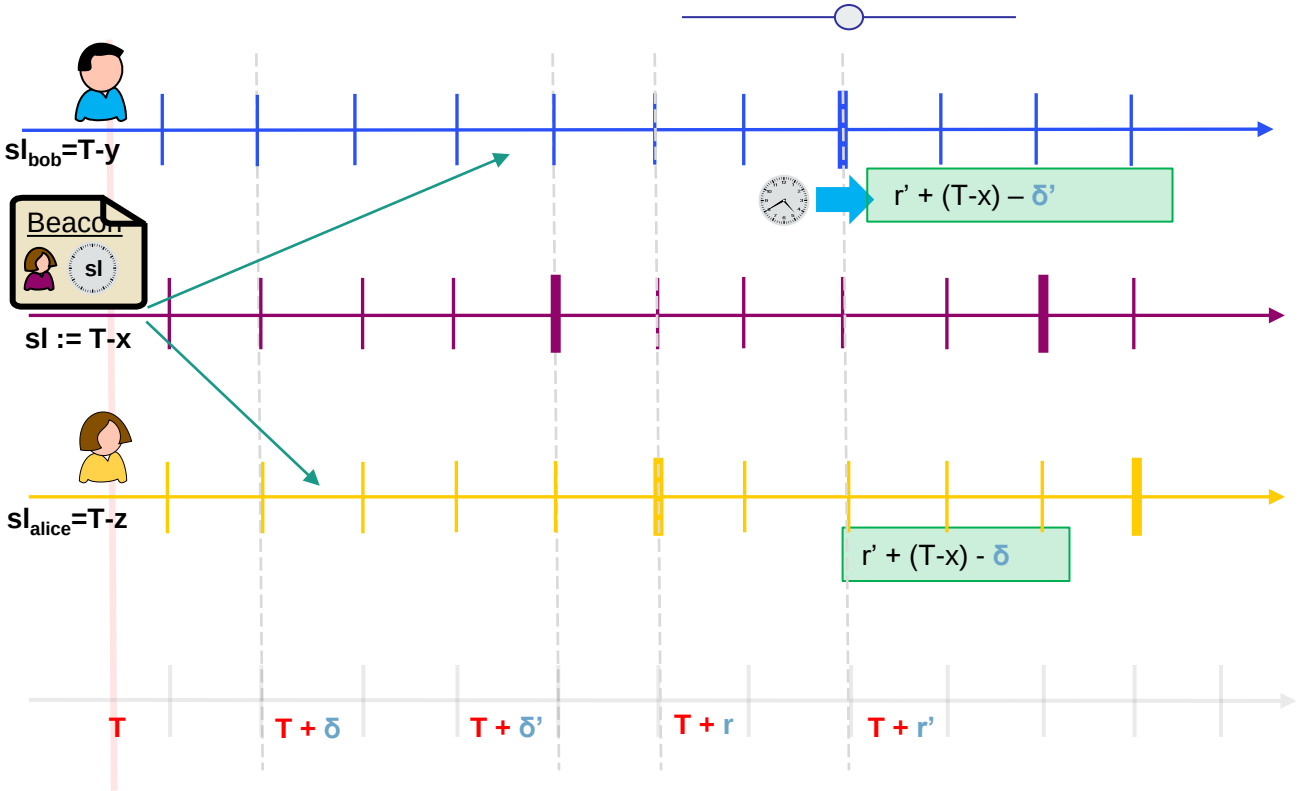
Synchronization Procedure - Example



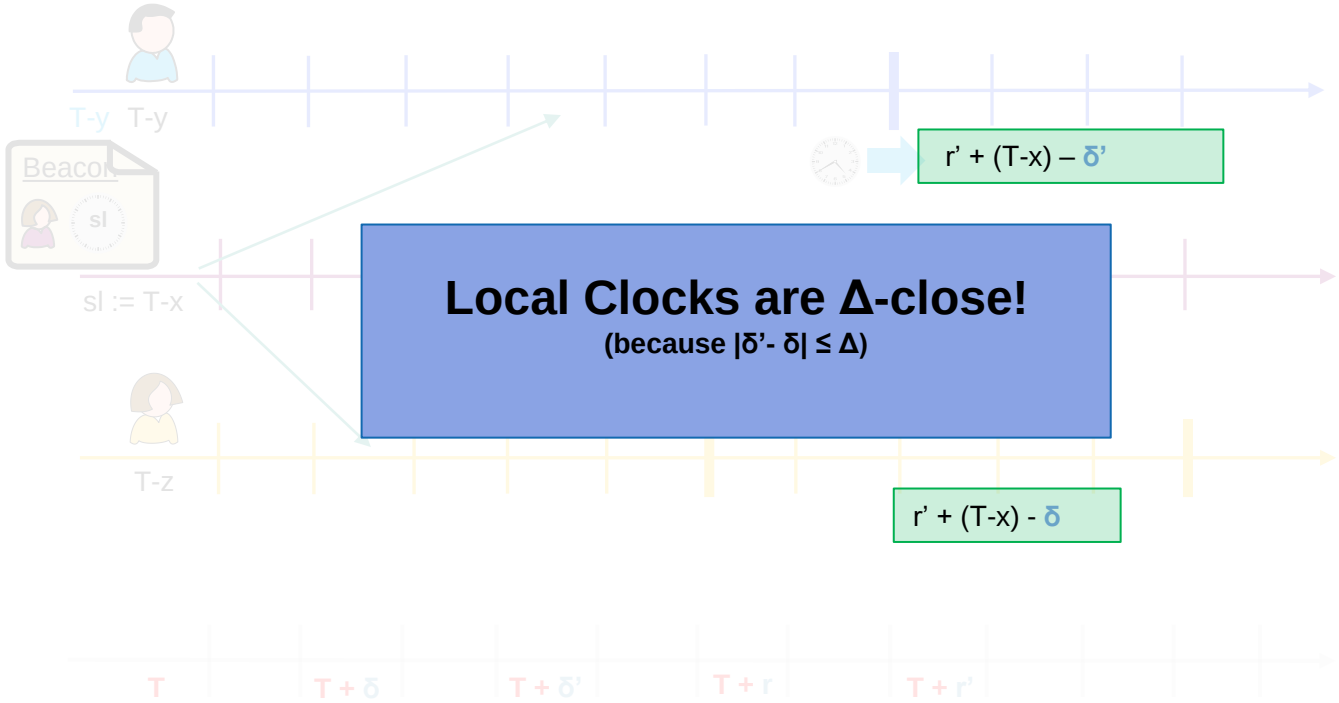
Synchronization Procedure - Example



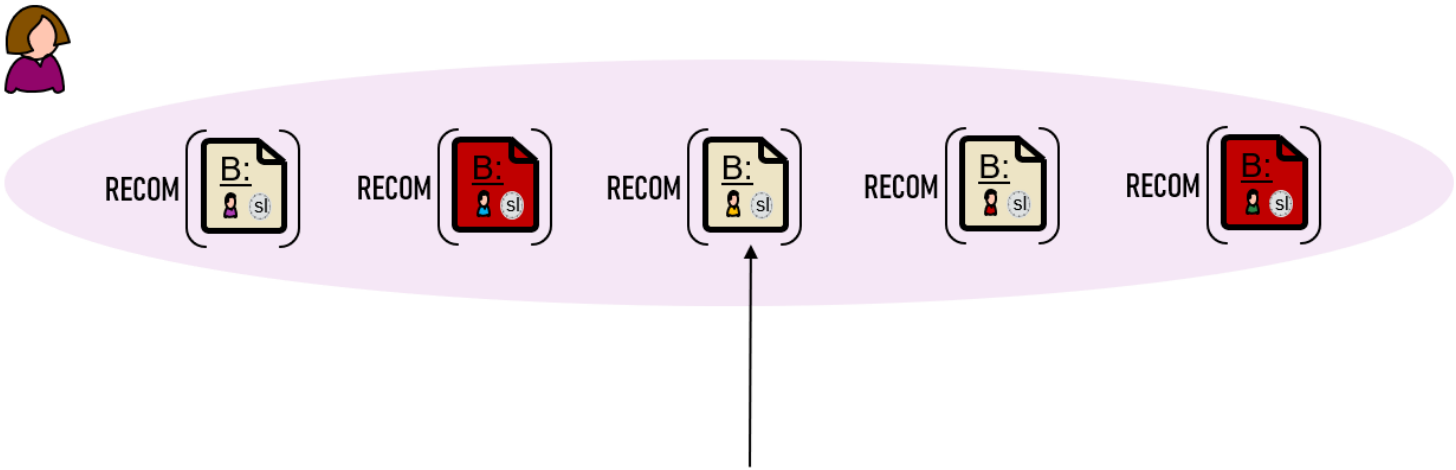
Synchronization Procedure - Example



Synchronization Procedure – Property 1



Synchronization Procedure – Property 2



Furthermore, by honest-majority assumption:
→ Median, i.e., **adjustment is bounded**.

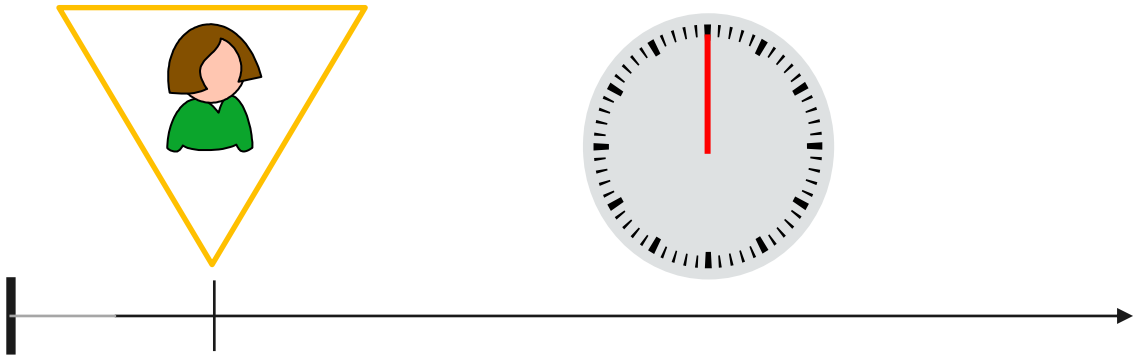
Chronos

- **Alert parties:** Execute Ouroboros Genesis, broadcast **time-beacons** and **leave evidence** of beacons in the blockchain.

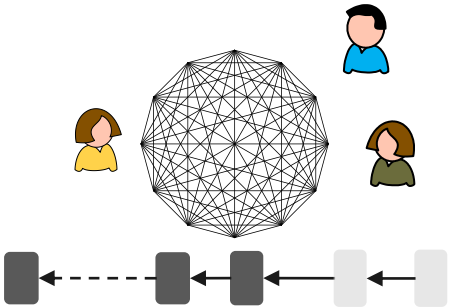
- They perform **local-clock adjustments** based on the evidence in the chain.
  - Small adjustments to local clocks at the end of an epoch
  - Based on the evidence left in the chain.

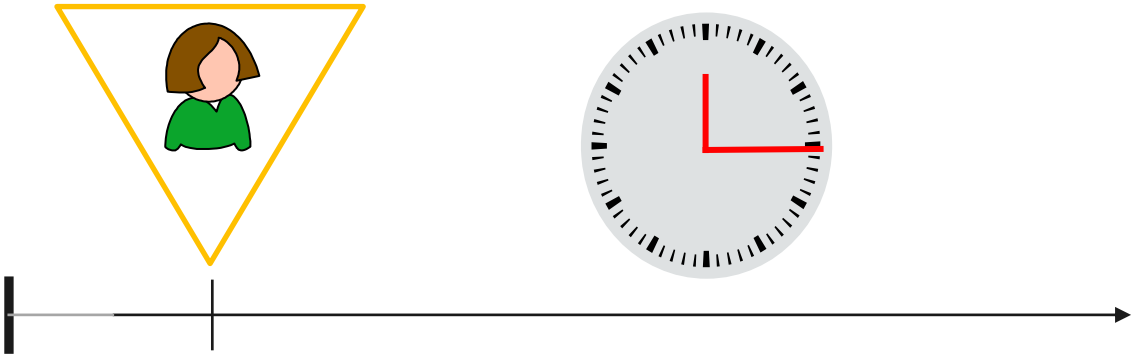
- **Joining parties:** Once hooked up on a prefix of the densest chain, record beacons and **retrace the evidence.**
  - Perform the very same clock adjustments to compute a good timestamp

Chronos: Joining Procedure

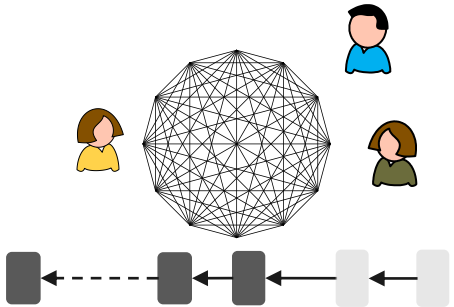


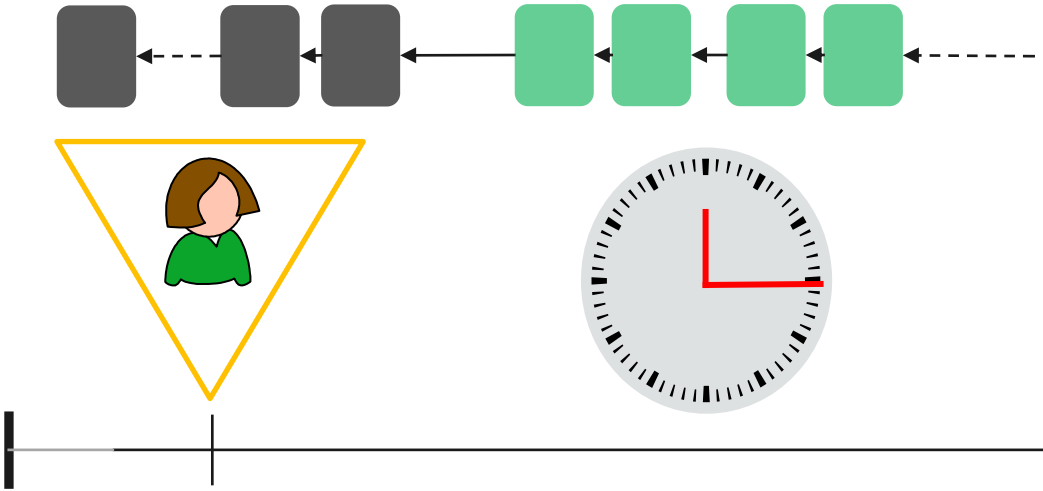
Chronos: Joining Procedure



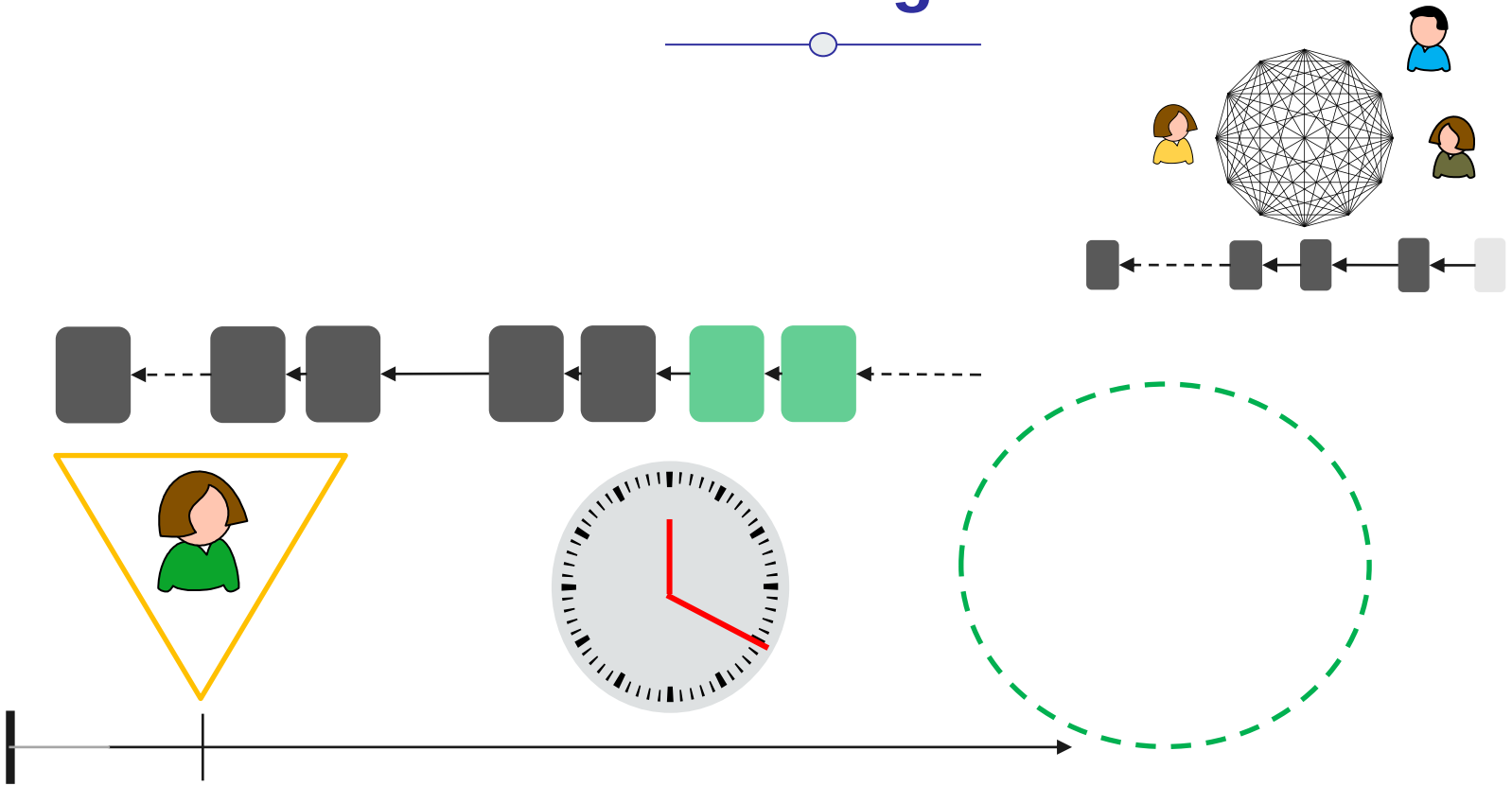


Chronos: Joining Procedure

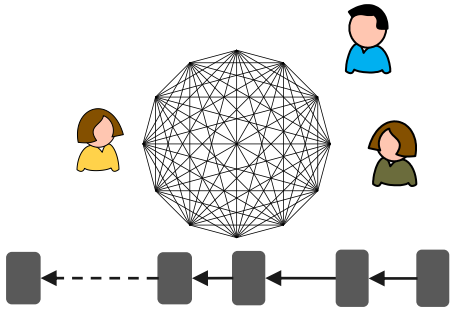


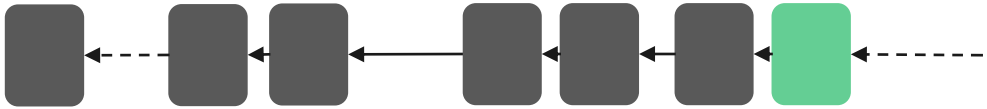


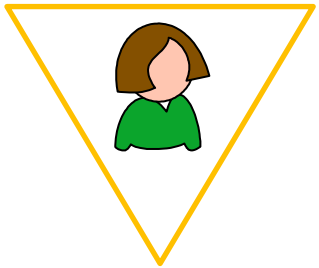
Chronos: Joining Procedure

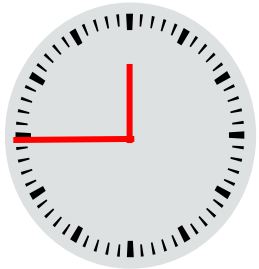


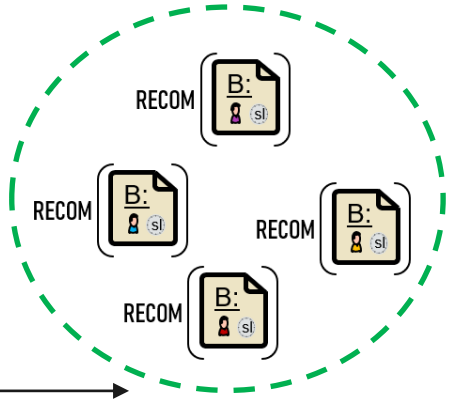
Chronos: Joining Procedure

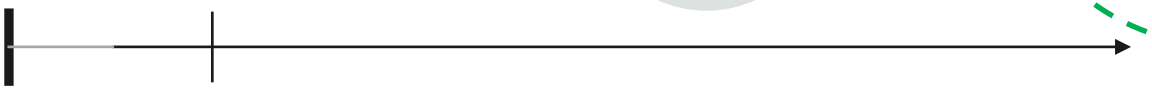




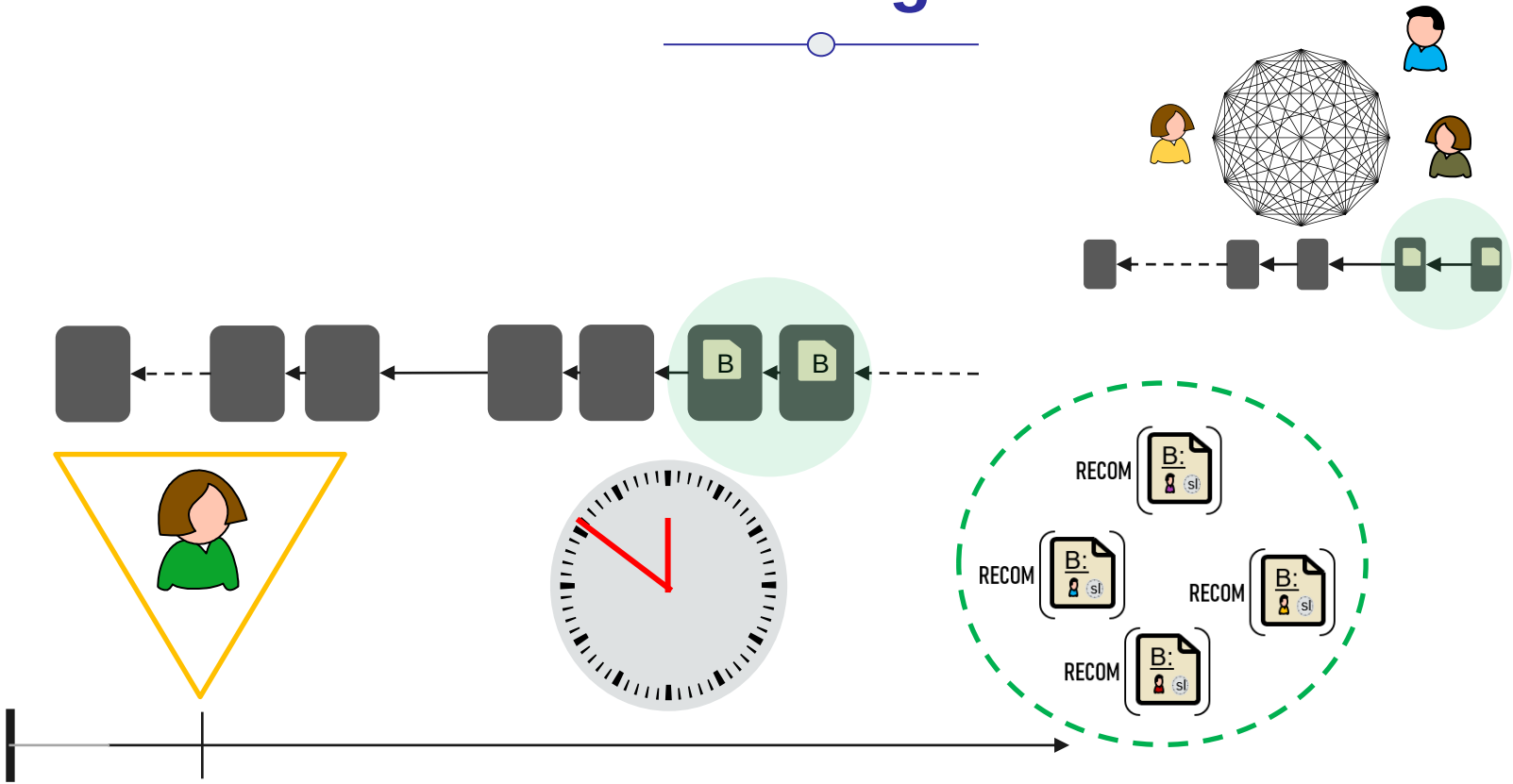




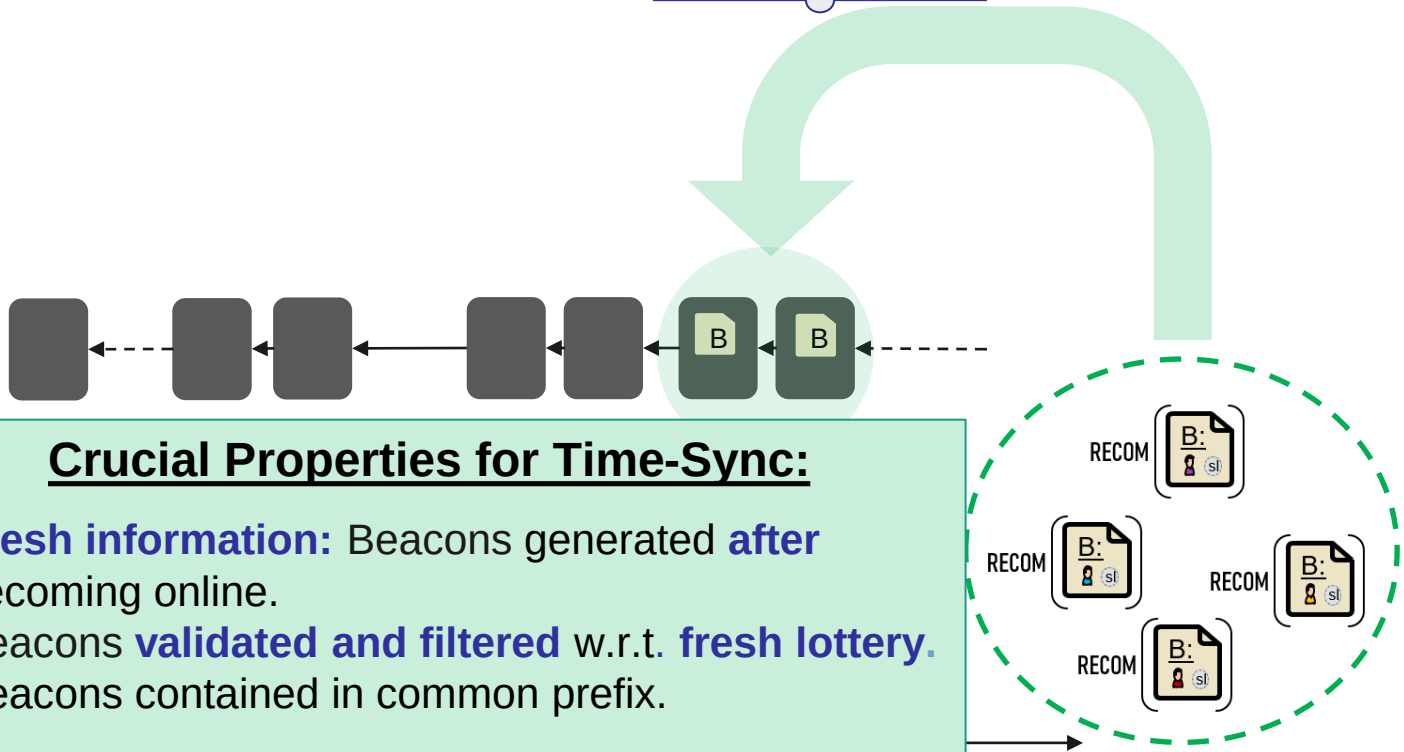




Chronos: Joining Procedure



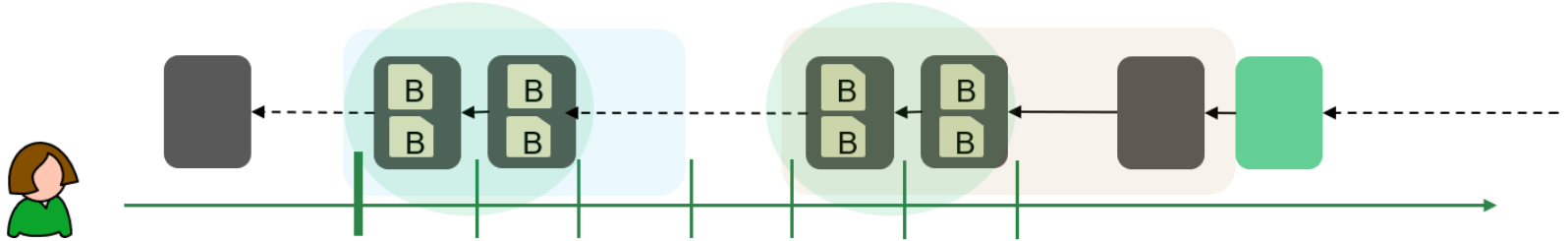
Chronos: Joining Procedure



Crucial Properties for Time-Sync:

- **Fresh information:** Beacons generated **after** becoming online.
- Beacons **validated and filtered** w.r.t. **fresh lottery**.
- Beacons contained in common prefix.

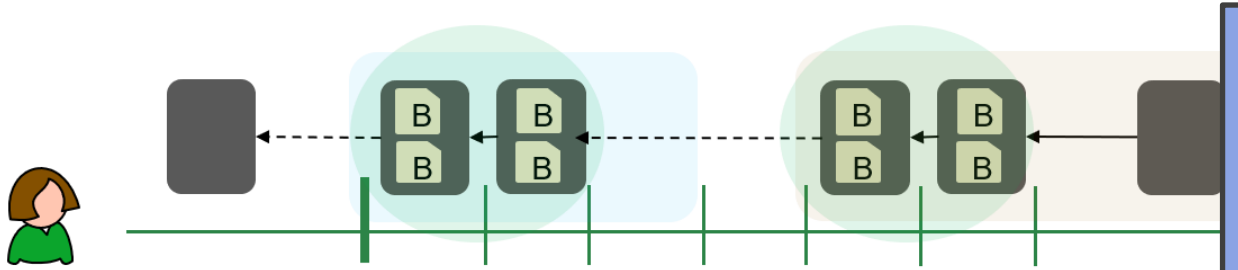
Chronos: Joining Procedure



- **Time-synchronization is possible:**
  - Agreement on evidence & freshness of beacons:
    → Reasoning like before to conclude Δ-closeness.

- **Clock adjustments of** alert parties **can be retraced** exactly!
  - Stop when computed timestamp is before the next sync-slot

Chronos: Joining Procedure



Note:

Any external observer can perform these operations.

- **Time-synchronization is possible:**
  - Agreement on evidence & freshness of beacons:
    → Reasoning like before to conclude Δ-closeness.

- **Clock adjustments of** alert parties **can be retraced** exactly!
  - Stop when computed timestamp is before the next sync-slot

Summary

Ouroboros Chronos is:

- A time synchronizer for the dynamic ad-hoc setting, …

  …where parties have access to a CRS (Genesis block);

  …where parties have access to a bounded-delay diffusion network and (approx.) same-speed clocks;

  …and where the honest-majority condition holds.

Summary

Ouroboros Chronos is:

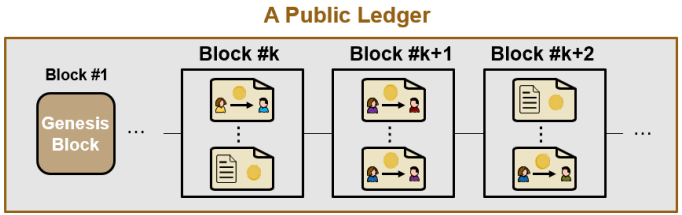
- A PoS blockchain protocol in the dynamic ad hoc setting (with a bounded-delay network), …

  … where parties can bootstrap the blockchain from the Genesis Block only.

  … where no external timing service is needed as new parties bootstrap the time (based on (approx.) same-speed local clocks).

  … whose security is based on the honest-stake majority assumption.

Ouroboros Chronos =



+

A Global Clock

Images: https://openclipart.org/, https://publicdomainvectors.org/

Thanks to Dominic Hicks for editing the video.