Lecture 22

Lecturer: Yu Shen

Today's Topics

- Proof-of-Stake Background
- Ouroboros
  - Protocol Execution, characteristic String and Forks
  - Security Analysis
  - Dynamic Stake
- Ouroboros Genesis
  - Bootstrapping from genesis

Bitcoin Challenges

Is this the best solution under the same assumptions? → Bitcoin is slow

What are other assumptions & hypothesis that may be used → Bitcoin has high energy consumption

~ 2000 tx/s

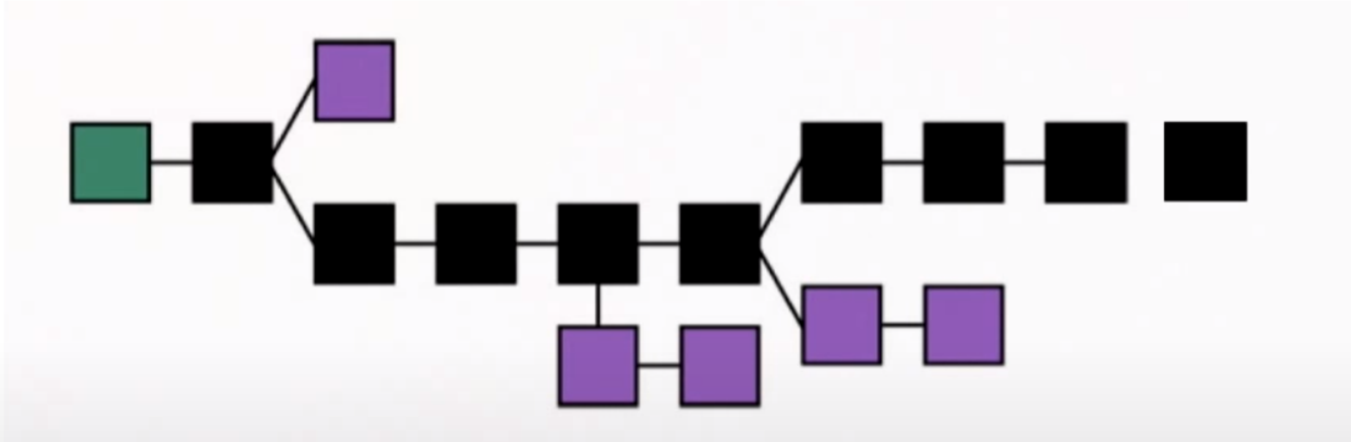
~ 100 tx/s

~ 15 tx/s

In 2020

Proof-of-Stake Background

- Generating the next block in Bitcoin is like an election.

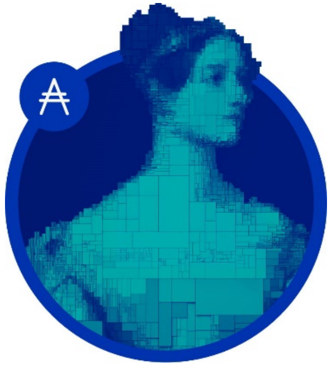


- A miner is elected with probability proportional to its hashing power.
- "Collisions" may occur but they can be solved by the longest chain rule or a similar concept.

Proof-of-Stake

- Use stake (a virtual resource) instead of hashing power (a physical resource).

CARDANO



Ada

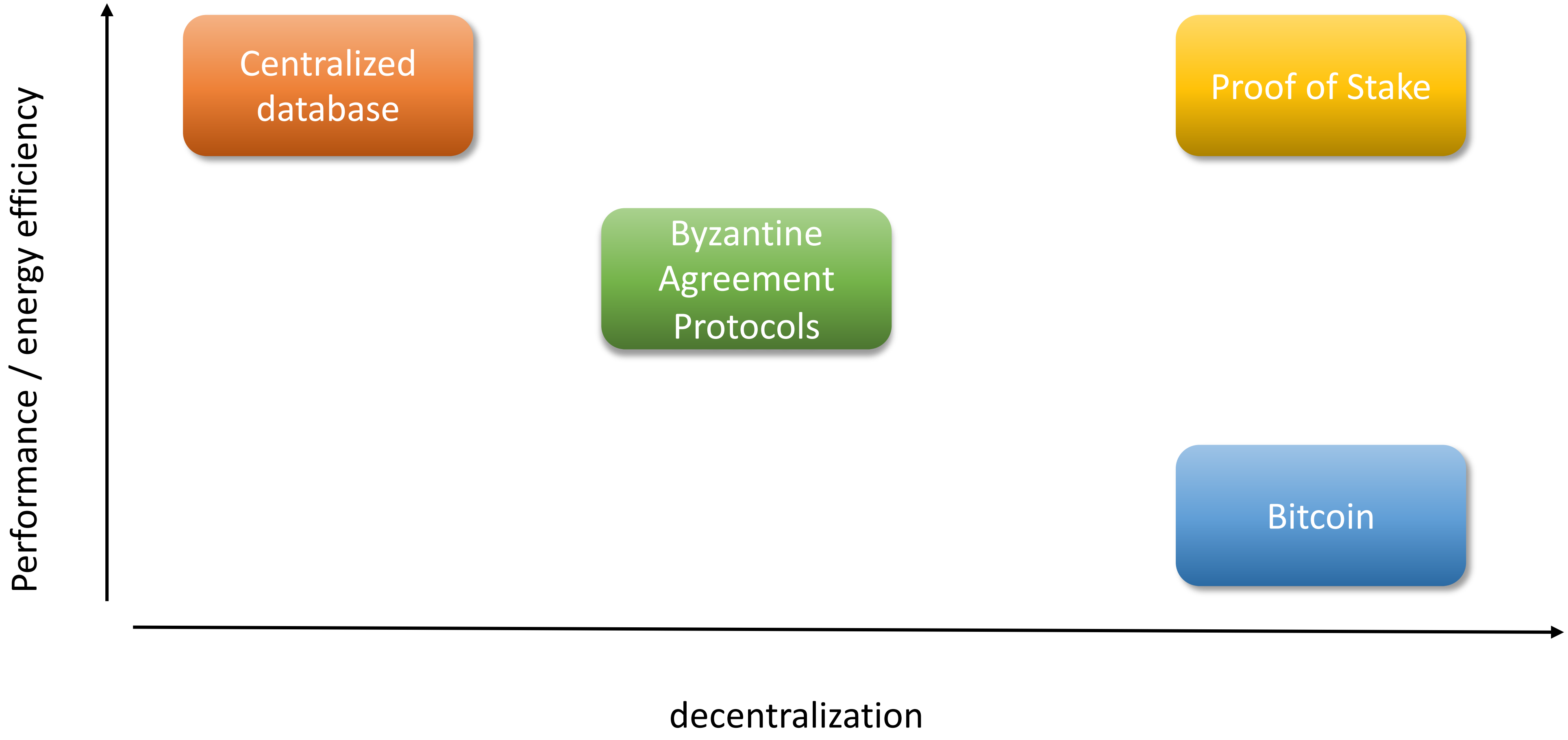
Proof-of-Stake

- Use stake (a virtual resource) instead of hashing power (a physical resource).

- Define a set of miners to be the set of all stakeholders, as reported in the ledger.

- Use a randomized process that takes the current stake into account to elect the next miner eligible to produce a block.

Performance vs. Decentralization



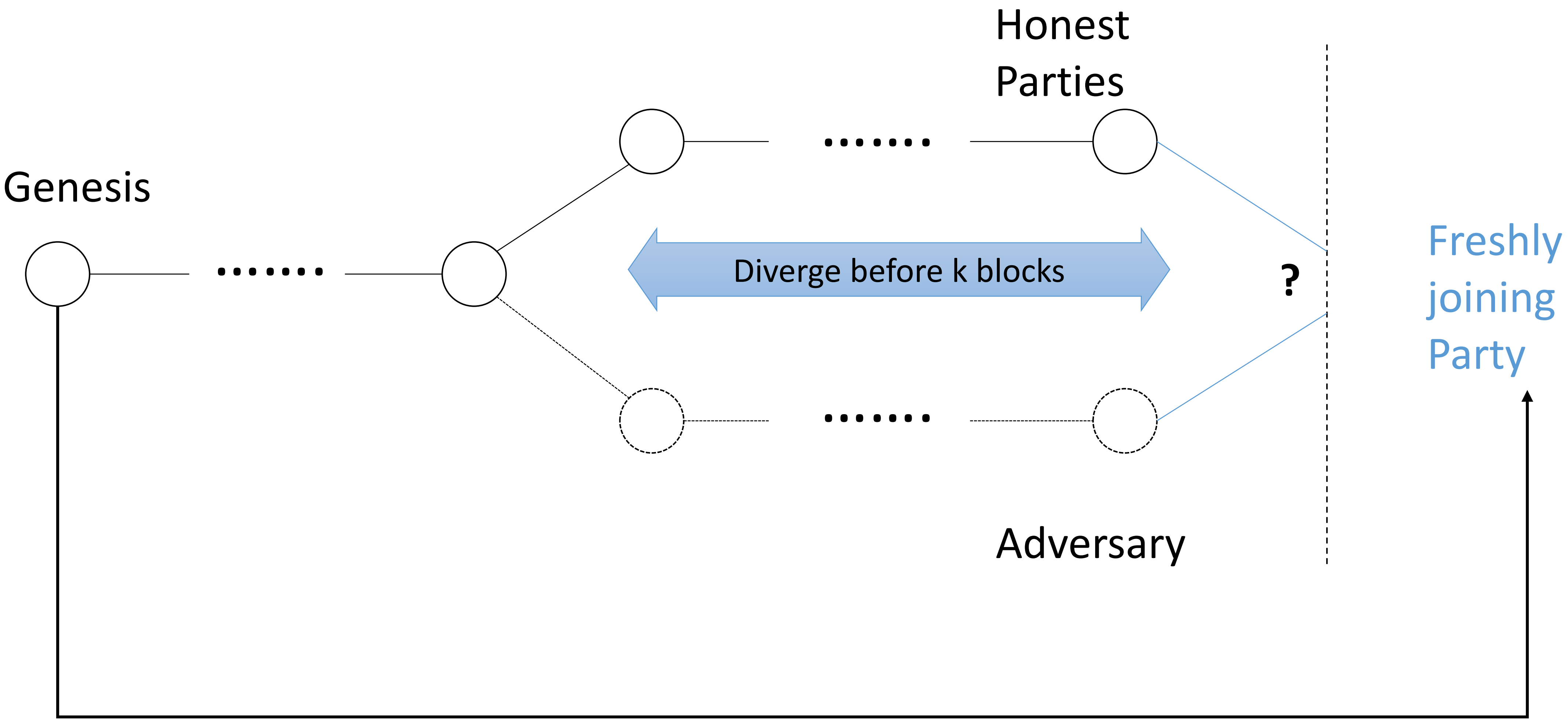
Proof-of-Stake Approaches

- PoS blockchains. Employ hash chains, digital signatures and some form of longest chain rule.
  - E.g., Ouroboros. Snow White. NXT.
- PoS BFT. Adapt classical Byzantine fault tolerant protocols to operate in the PoS setting.
  - E.g., Algorand.
- Both approaches are classified as PoS since protocol participation is based on proof of stake.

A folklore perspective

- PoS blockchains are impossible to work in the setting where Bitcoin operates.
- Reasons:
  - Costless simulation.
    - Given no physical resources are used in producing blocks, it is possible to build alternative transaction histories at essentially no cost.
    - nothing at stake
  - Long-range attacks.
    - In long-range attack the victim tries to distinguish between two alternative histories furnished by the network without any recent information.
    - The bootstrapping problem: how does a new (or long term desynchronized) node synchronize with the blockchain?

Long range attack



Today's Topics

- Proof-of-Stake Background
- Ouroboros
  - Protocol Execution, characteristic String and Forks
  - Security Analysis
  - Dynamic Stake
- Ouroboros Genesis
  - Bootstrapping from genesis

Ouroboros Papers

- *Aggelos Kiayias and Alexander Russell and Bernardo David and Roman Oliynykov.* **Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol.** https://eprint.iacr.org/2016/889
- *Bernardo David and Peter Gaži and Aggelos Kiayias and Alexander Russell.* **Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake protocol.** https://eprint.iacr.org/2017/573
- *Christian Badertscher and Peter Gazi and Aggelos Kiayias and Alexander Russell and Vassilis Zikas.* **Ouroboros Genesis: Composable Proof-of-Stake Blockchains with Dynamic Availability.** https://eprint.iacr.org/2018/378
- *Thomas Kerber and Markulf Kohlweiss and Aggelos Kiayias and Vassilis Zikas.* **Ouroboros Crypsinous: Privacy-Preserving Proof-of-Stake.** https://eprint.iacr.org/2018/1132
- *Christian Badertscher and Peter Gaži and Aggelos Kiayias and Alexander Russell and Vassilis Zikas.* **Ouroboros Chronos: Permissionless Clock Synchronization via Proof-of-Stake.** https://eprint.iacr.org/2019/838

Ouroboros PoS

- First provably secure proof of stake (Nakamoto-like) blockchain protocol.
- Introduced a basic design structure for building secure PoS blockchains.
- Introduced the forkable string combinatorial analysis toolset that can be used to analyaze longest chain protocols in PoS.

Early alchemical ouroboros illustration with the words ἓν τὸ πᾶν ("The All is One") from the work of Cleopatra the Alchemist in MS Marciana gr. Z. 299. (10th Century)

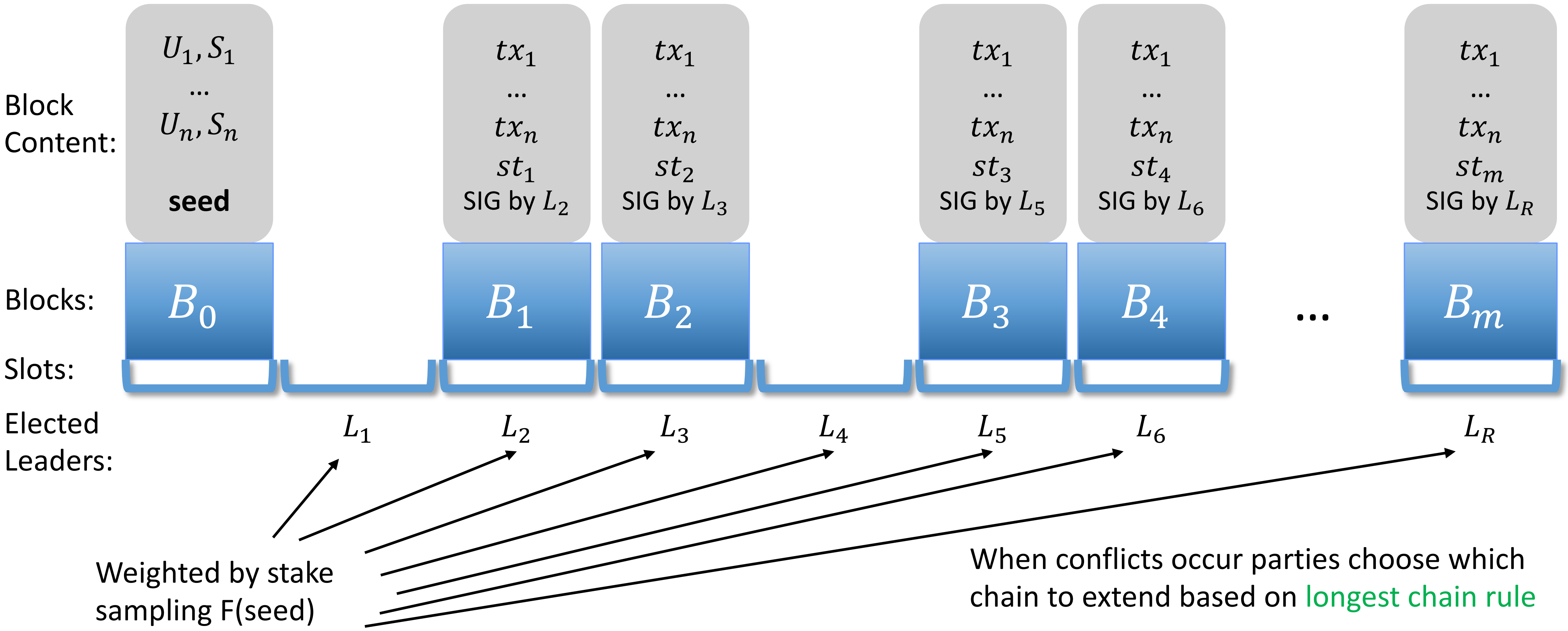
Ouroboros Design

- **Stage 1.** Static Stake. Assume that initial stakeholder distribution remains the root of trust of the system.
- **Stage 2.** Using a trusted beacon. Assume a randomness beacon emits a seed in regular intervals and show how this can be utilized to let the roof to trust stakeholder distribution evolve.
- **Stage 3.** Simulating a beacon cryptographically. Remove the trusted beacon by having the elected subset of stakeholders simulate it.

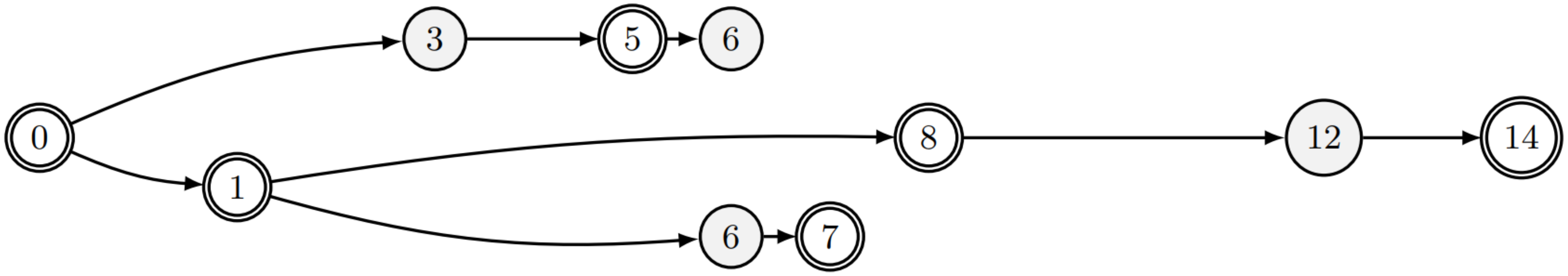
Synchronous Setting

- Time is divided in rounds (*slots*).
- Messages are sent through a “diffusion” mechanism.
- The adversary is rushing and may:
  - Spoof messages
  - Inject messages
  - Reorder messages
- Leader election is treated as an ideal functionality.
- The stakeholders are always online.

Ouroboros: Static Stake



Forks and Protocol Executions

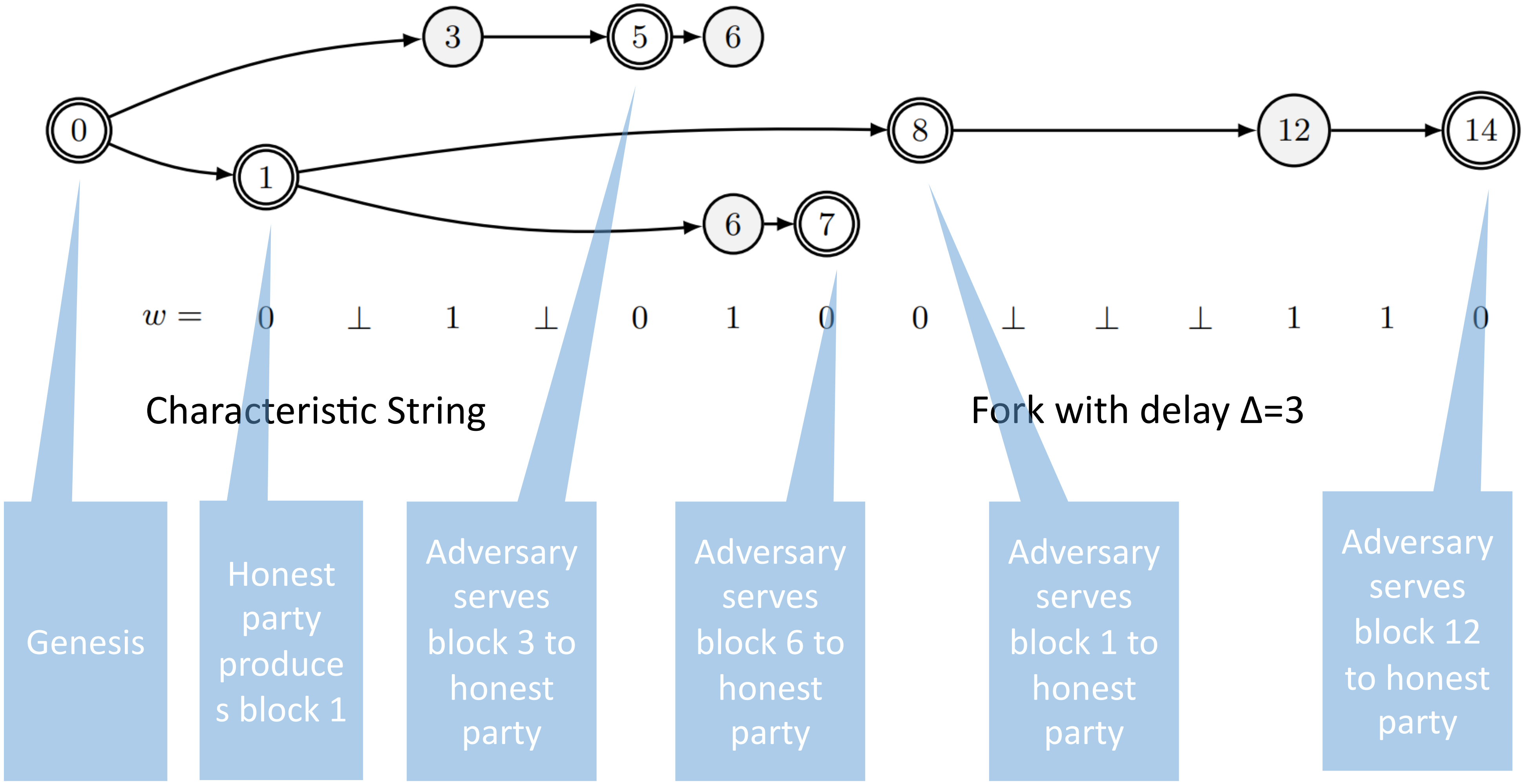


$$w = \quad 0 \quad \perp \quad 1 \quad \perp \quad 0 \quad 1 \quad 0 \quad 0 \quad \perp \quad \perp \quad \perp \quad 1 \quad 1 \quad 0$$

Characteristic String

$$w_i = \begin{cases} 0 & i\text{-th slot belongs to an honest party} \\ 1 & i\text{-th slot belongs to a malicious coalition} \\ \perp & i\text{-th cannot be claimed} \end{cases}$$

Forks and Protocol Executions

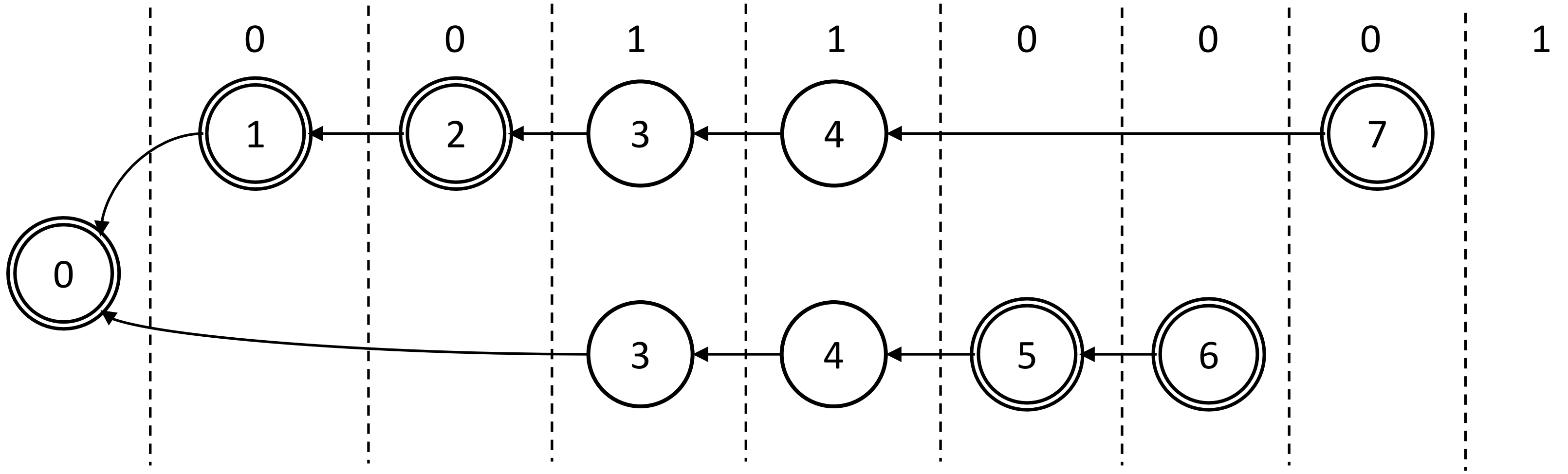


PoW vs. PoS Proof

- The adversary is at a much better position in this protocol execution compared to Bitcoin's PoW-based execution.
- It can see ahead of time how stakeholders are activated.
- It can generate multiple different blocks for the same slot at any time without cost.
- It can wait and act just before an honest party comes online.

Forkable Strings

- Strings that has a fork that it has two edge-disjoint paths of length equal to the height of the fork.
- The characteristic strings the adversary prefers!

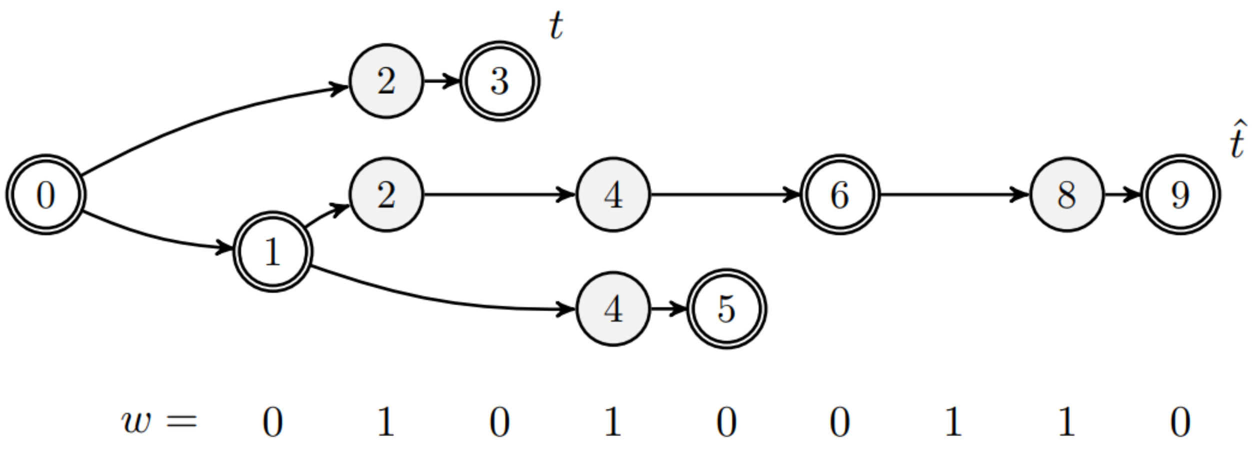


Today’s Topics

- Proof-of-Stake Background
- Ouroboros
  - Protocol Execution, characteristic String and Forks
  - Security Analysis
  - Dynamic Stake
- Ouroboros Genesis
  - Bootstrapping from genesis

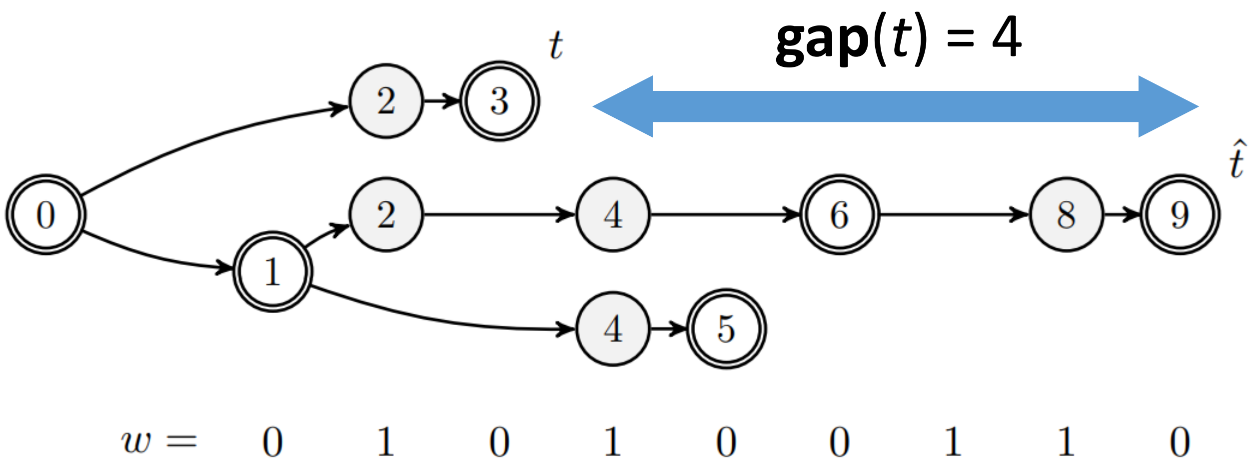
Forkable Strings (1)

- For a path *t*:
  - **gap**(*t*): length difference with deepest honest node.
  - **reserve**(*t*): number of adversarial slots after end of t.
  - **reach**(*t*): **reverse**(*t*) – **gap**(*t*)



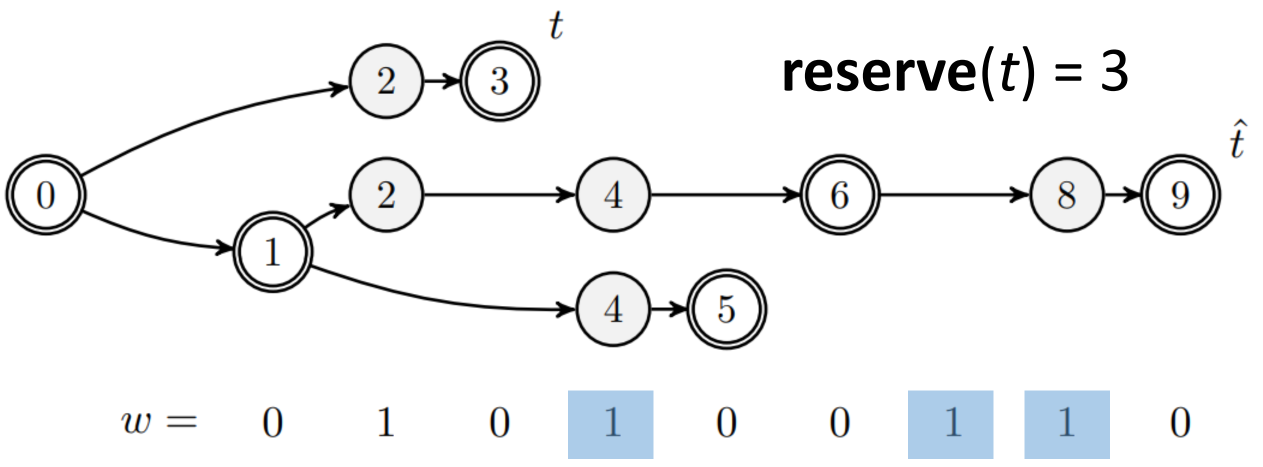
Forkable Strings (2)

- For a path *t*:
  - **gap**(*t*): length difference with deepest honest node.
  - **reserve**(*t*): number of adversarial slots after end of t.
  - **reach**(*t*): **reverse**(*t*) – **gap**(*t*)



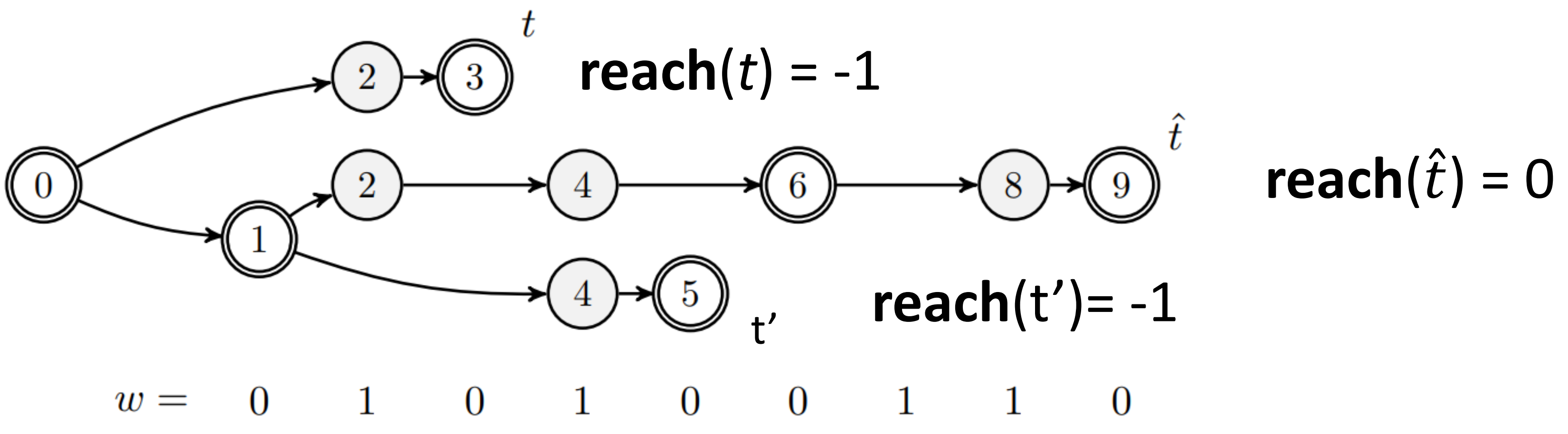
Forkable Strings (3)

- For a path *t*:
  - **gap**(*t*): length difference with deepest honest node.
  - **reserve**(*t*): number of adversarial slots after end of t.
  - **reach**(*t*): **reverse**(*t*) – **gap**(*t*)



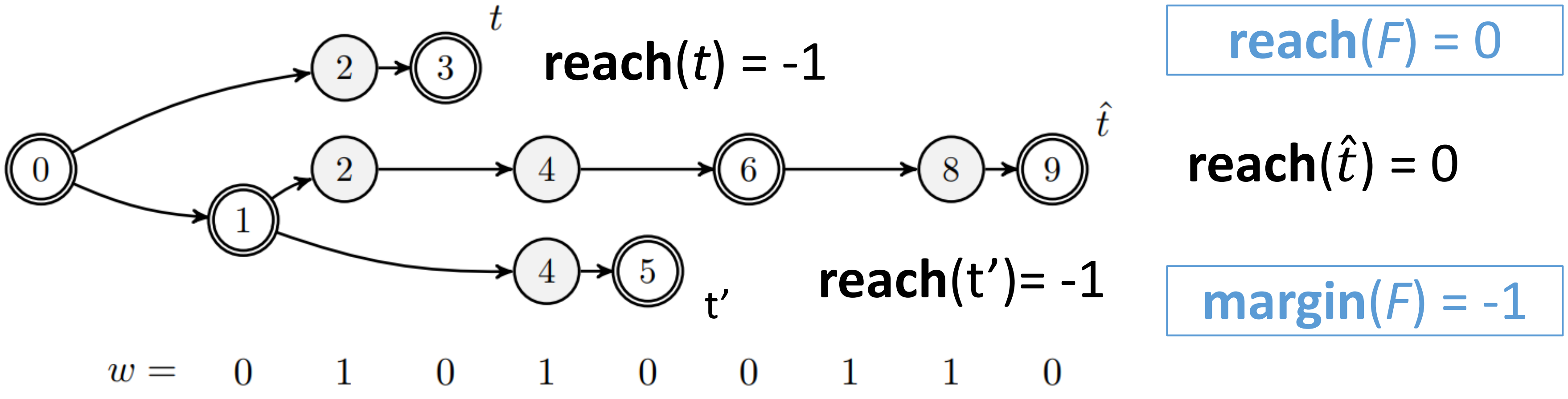
Forkable Strings (4)

- For a path t:
  - **gap**(t): length difference with deepest honest node.
  - **reserve**(t): number of adversarial slots after end of t.
  - **reach**(t): **reverse**(t) – **gap**(t)



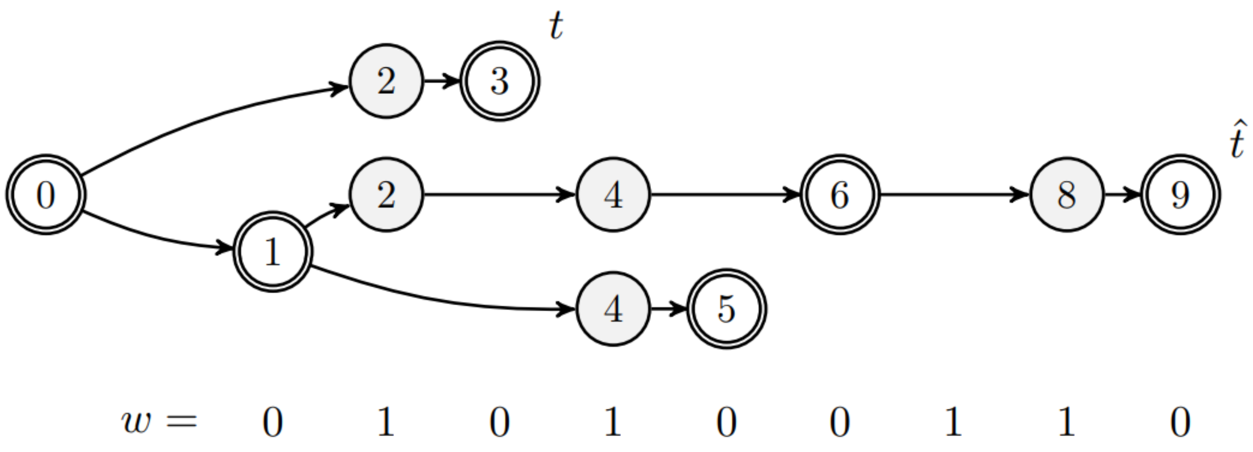
Forkable Strings (5)

- For a fork F:
  - **reach**(F) = max **reach**(t).
  - **margin**(F) = second best disjoint **reach**(t).



Forkable Strings (6)

- For a string w:
  - $\rho(w) = \max_F \mathbf{reach}(F)$
  - $\mu(w) = \max_F \mathbf{margin}(F)$



Forkable Strings (7)

- Theorem: a string *w* is forkable (adversary wins) iff. $\mu(w) \geq 0$.
  - ($\Longrightarrow$): consider a fork *F* with 2 path *t*, *t'* with the same length. Then, consider another fork F' which removes all the adversarial vertices after the last honest vertex on each path in F, the prefix of *t* and *t'* must have reverse no less then gap; thus **margin**(*F*)≥0.

  - ($\Longleftarrow$): consider a fork *F* with **margin**(*F*)≥0, there exist at least one path *t'* (ending with honest index) such that **reach**(*t'*) ≥0. So the adversary can append vertices with adversarial indices to make it the same length as the longest path *t*.

Recursive Formula for Reach & Margin

$$\mathbf{m}(w) = (\rho(w), \mu(w))\,.$$

$$\mathbf{m}(\epsilon) = (0,0) \textit{ and, for all nonempty strings } w \in \{0,1\}^*,$$

$$\mathbf{m}(w1) = (\rho(w)+1, \mu(w)+1)\,, \textit{ and}$$

$$\mathbf{m}(w0) = \begin{cases} (\rho(w)-1, 0) & \textit{if } \rho(w) > \mu(w) = 0, \\ (0, \mu(w)-1) & \textit{if } \rho(w) = 0, \\ (\rho(w)-1, \mu(w)-1) & \textit{otherwise.} \end{cases}$$

It is possible for the adversary to compensate for the margin, by sacrificing reach

Reach never drops below 0

Reach and margin decrement

Recursive Formula for Reach & Margin

$$\mathbf{m}(w) = (\rho(w), \mu(w)) \,.$$

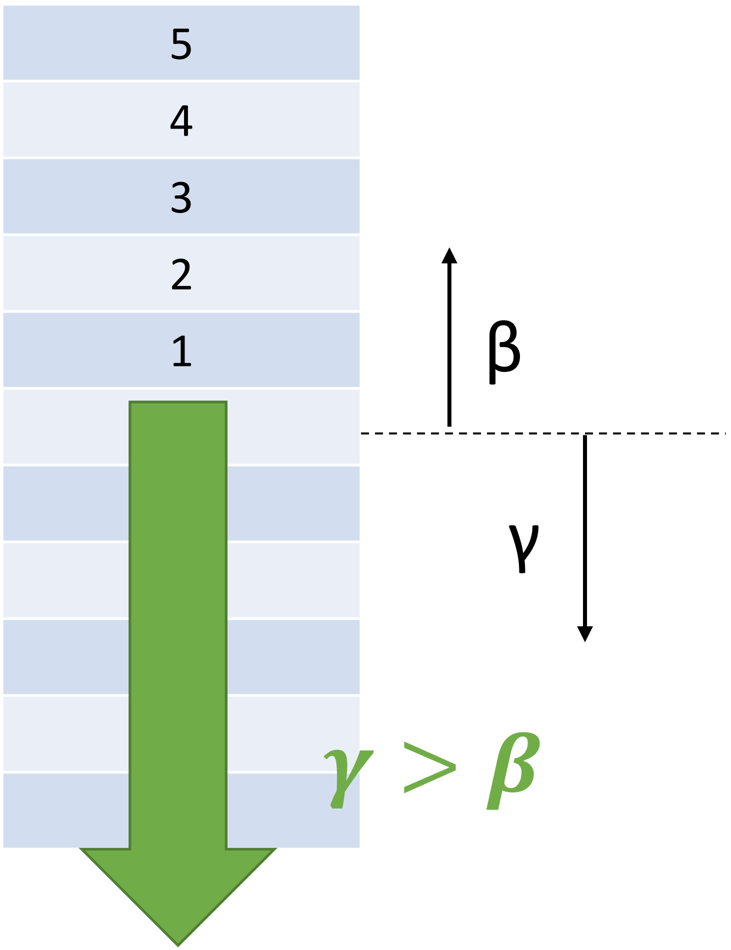
$\mathbf{m}(\epsilon) = (0,0)$ *and, for all nonempty strings* $w \in \{0,1\}^*$,

$$\mathbf{m}(w1) = (\rho(w)+1, \mu(w)+1) \text{, and}$$

$$\mathbf{m}(w0) = \begin{cases} (\rho(w)-1, 0) & \text{if } \rho(w) > \mu(w) = 0, \\ (0, \mu(w)-1) & \text{if } \rho(w) = 0, \\ (\rho(w)-1, \mu(w)-1) & \text{otherwise.} \end{cases}$$

- This forms a 2-D random walk.

Drawing from Bitcoin analysis



- At the core of the analysis lies a 1D random walk.
- α: probability an honest party finds a PoW.
  - $\gamma \approx \alpha - \alpha^2$
- β: probability the adversary finds a PoW.
- A favorable step is downwards.

From PoW to PoS

- Winning a slot for the honest parties (even uniquely) does not necessarily constitute a favorable step in the random walk.
- Reason: costless simulation / nothing-at-stake: the adversary may reuse an opportunity to issue a block in multiple paths of a fork.

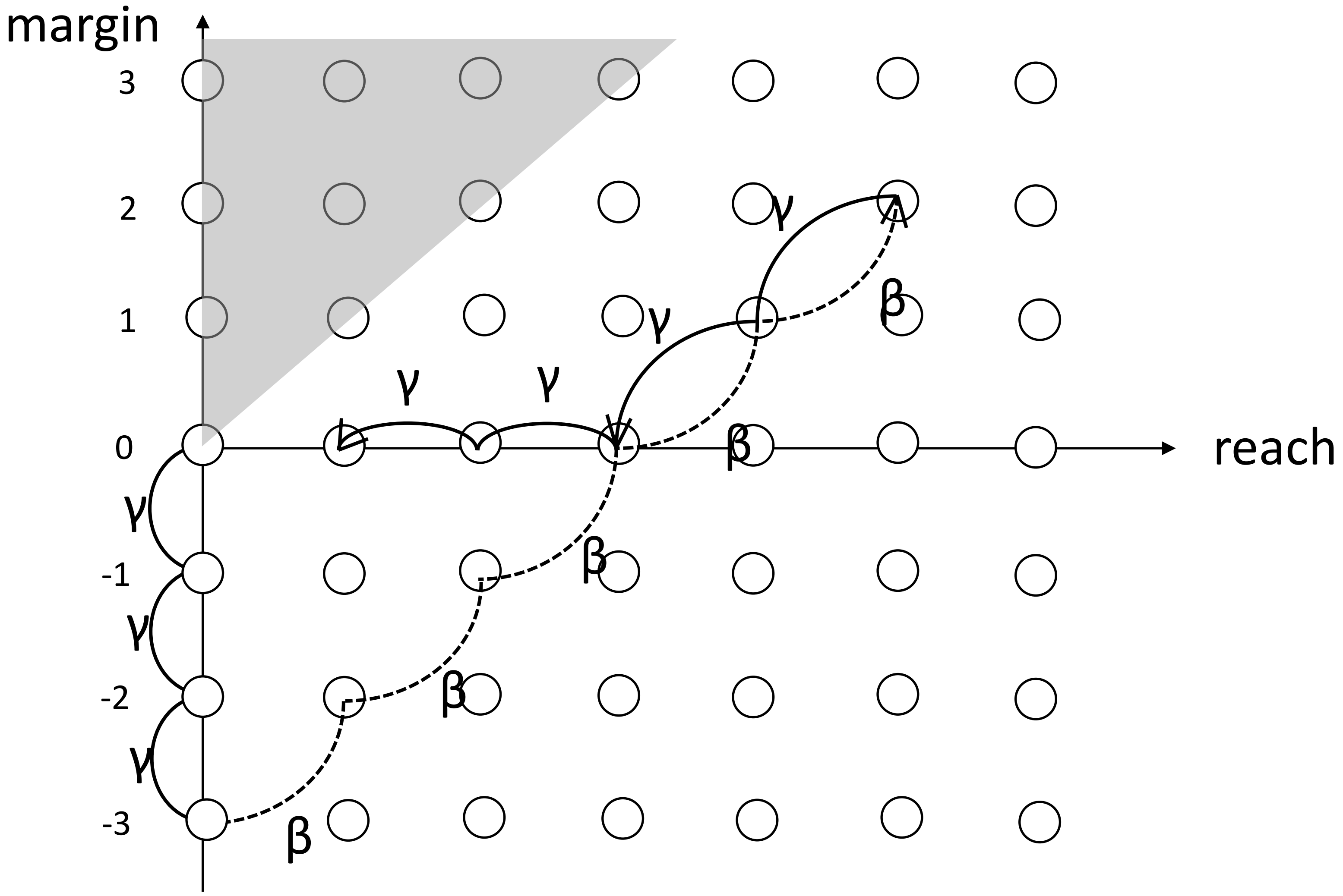
$$\mathbf{m}(w) = (\rho(w), \mu(w)) \,.$$

$\mathbf{m}(\epsilon) = (0,0)$ *and, for all nonempty strings* $w \in \{0,1\}^*$,

$$\mathbf{m}(w1) = (\rho(w)+1, \mu(w)+1)\,, \text{ and}$$

$$\mathbf{m}(w0) = \begin{cases} (\rho(w)-1, 0) & \text{if } \rho(w) > \mu(w) = 0, \\ (0, \mu(w)-1) & \text{if } \rho(w) = 0, \\ (\rho(w)-1, \mu(w)-1) & \text{otherwise.} \end{cases}$$

2-D Random Walk



- α: probability an honest party wins a slot.
  - $\gamma \approx \alpha - \alpha^2$
- β: probability the adversary wins a slot.
- **reach**(w) ≥ 0.
- **reach**(w) ≥ **margin**(w).
- A favorable step is [?]

Forkable Strings are rare

- Goal: $\Pr[w\ is\ forkable] = 2^{-\Omega(\sqrt{n})}$
- $w = 0101 \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots 1010$

w_1	w_2	$\dots \dots \dots \dots$	$w_{\sqrt{n}}$
$\sqrt{n}$	$\sqrt{n}$		$\sqrt{n}$

- $R_{(t)} = \rho(w_1 \dots w_t)$ and $M_{(t)} = \mu(w_1 \dots w_t)$.

- $\Pr[w\ is\ forkable] = \Pr[M_n \geq 0]$

RECALL

Recursive Formula for Reach & Margin

$$\mathbf{m}(w) = (\rho(w), \mu(w))\,.$$

$\mathbf{m}(\epsilon) = (0,0)$ *and, for all nonempty strings* $w \in \{0,1\}^*$,

$$\mathbf{m}(w1) = (\rho(w)+1, \mu(w)+1)\,,\ \textit{and}$$

$$\mathbf{m}(w0) = \begin{cases} (\rho(w)-1, 0) & \textit{if } \rho(w) > \mu(w) = 0, \\ (0, \mu(w)-1) & \textit{if } \rho(w) = 0, \\ (\rho(w)-1, \mu(w)-1) & \textit{otherwise.} \end{cases}$$

It is possible for the adversary to compensate for the margin, by sacrificing reach

Reach never drops below 0

Reach and margin decrement

Forkable Strings are rare

- Extract facts of random variables $R_{(t)}$ and $M_{(t)}$.
- Define 3 events:
  - **Hot**$_t$:
    $R_{(t)} \geq \delta\sqrt{n} \wedge M_{(t)} \geq -\delta\sqrt{n}$
  - **Volatile**$_t$: (initial)
    $-\delta\sqrt{n} \leq M_{(t)} \leq L_{(t)} \leq \delta\sqrt{n}$
  - **Cold**$_t$: $M_{(t)} \leq -\delta\sqrt{n}$

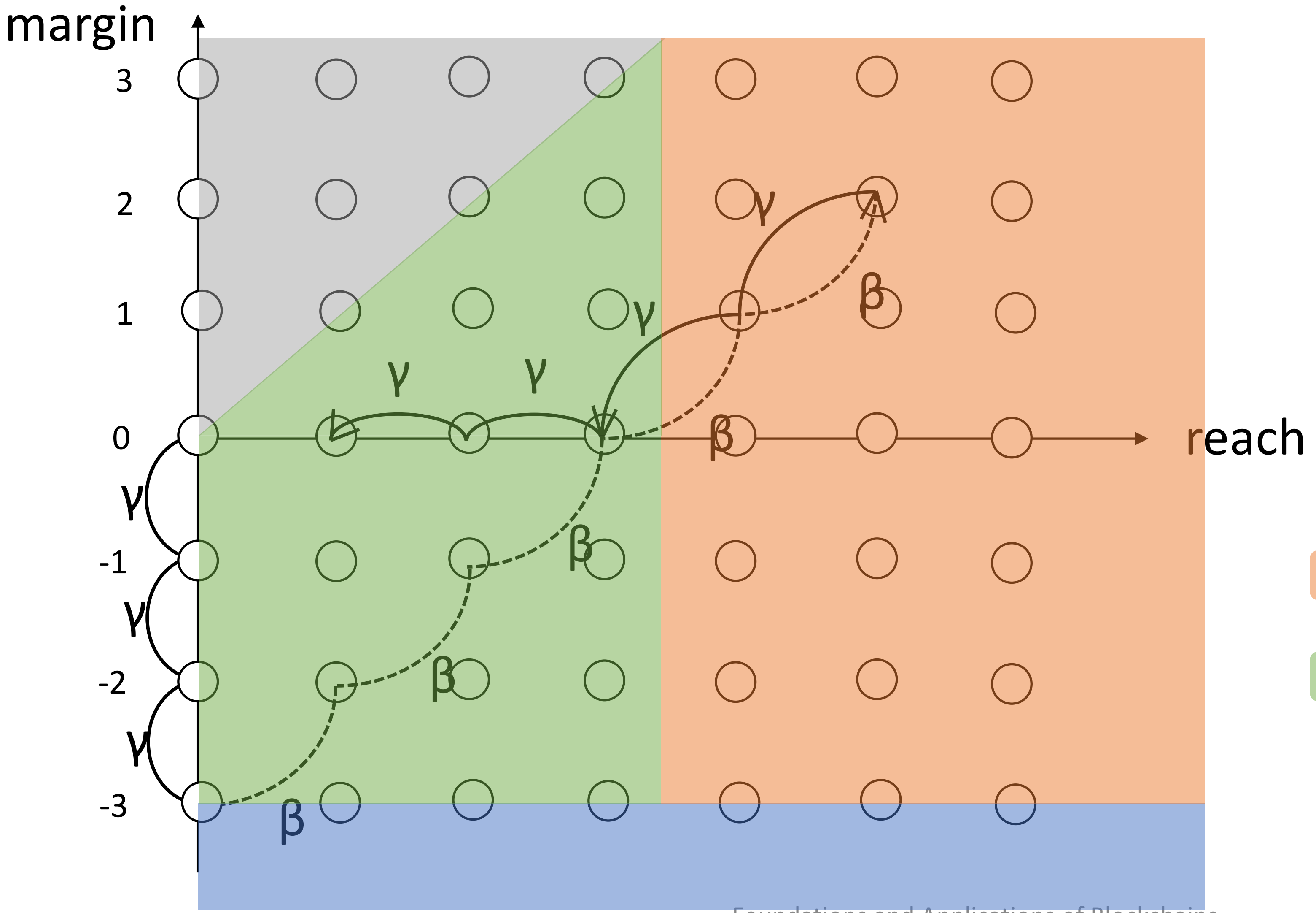
$$R_t > 0 \Longrightarrow \begin{cases} R_{t+1} = R_t + 1 & \text{if } w_{t+1} = 1, \\ R_{t+1} = R_t - 1 & \text{if } w_{t+1} = 0; \end{cases}$$

$$M_t < 0 \Longrightarrow \begin{cases} M_{t+1} = M_t + 1 & \text{if } w_{t+1} = 1, \\ M_{t+1} = M_t - 1 & \text{if } w_{t+1} = 0; \end{cases}$$

$$R_t = 0 \Longrightarrow \begin{cases} R_{t+1} = 1 & \text{if } w_{t+1} = 1, \\ R_{t+1} = 0 & \text{if } w_{t+1} = 0, \\ M_{t+1} < 0 & \text{if } w_t = 0. \end{cases}$$

- We want the execution stay in Cold.

2-D Random Walk



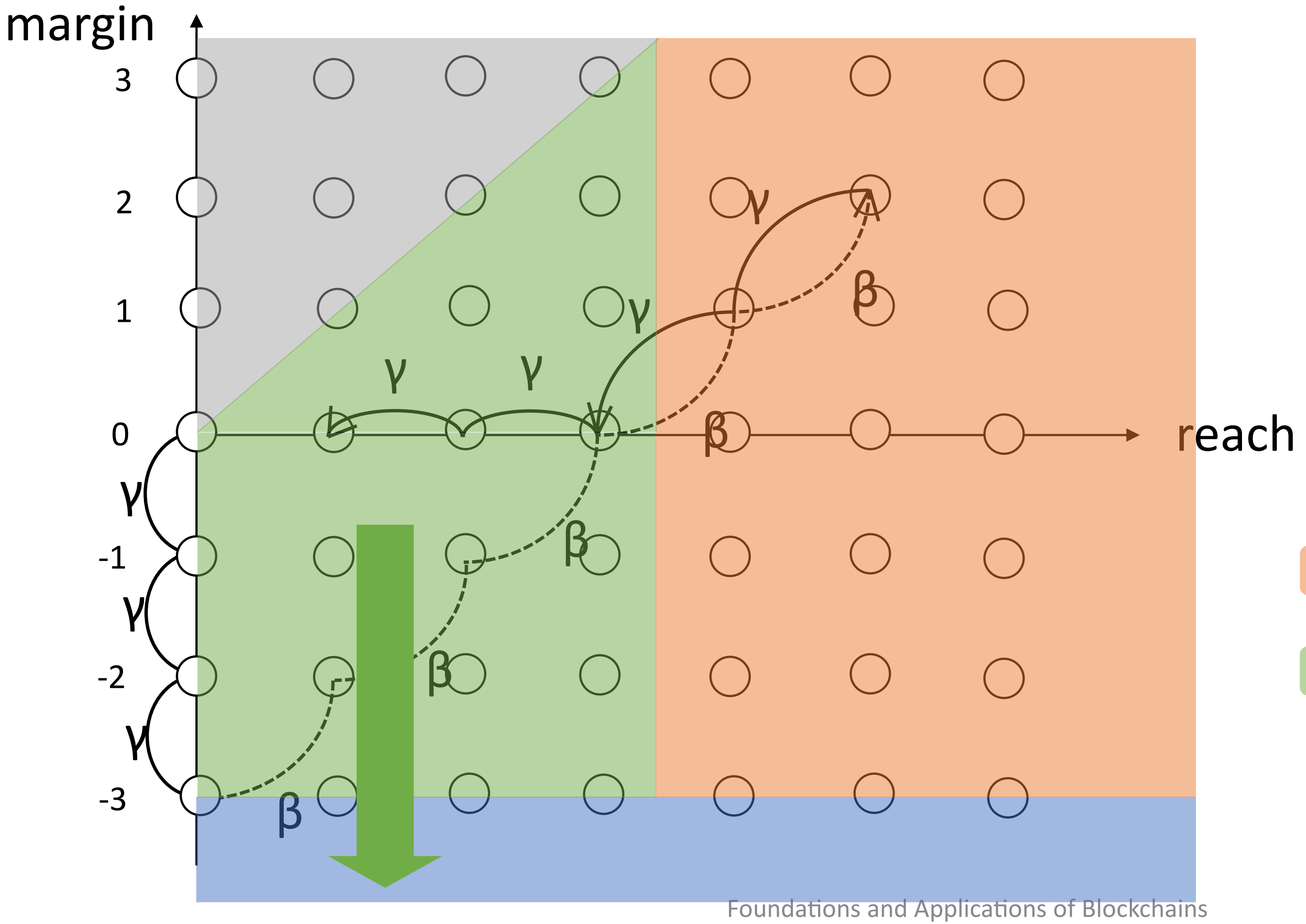
- α: probability an honest party wins a slot.
  - $\gamma \approx \alpha - \alpha^2$
- β: probability the adversary wins a slot.
- **reach**(w) ≥ 0.
- **reach**(w) ≥ **margin**(w).
- A favorable step is [?]

▪**Hot**$_t$: $R_{(t)} \geq \delta\sqrt{n} \wedge M_{(t)} \geq -\delta\sqrt{n}$

▪**Volatile**$_t$: $-\delta\sqrt{n} \leq M_{(t)} \leq L_{(t)} \leq \delta\sqrt{n}$

▪**Cold**$_t$: $M_{(t)} \leq -\delta\sqrt{n}$

2-D Random Walk



- α: probability an honest party wins a slot.
  - $\gamma \approx \alpha - \alpha^2$
- β: probability the adversary wins a slot.
- **reach**(w) ≥ 0.
- **reach**(w) ≥ **margin**(w).
- A favorable step is

▪Hot$_t$: $R_{(t)} \geq \delta\sqrt{n} \wedge M_{(t)} \geq -\delta\sqrt{n}$

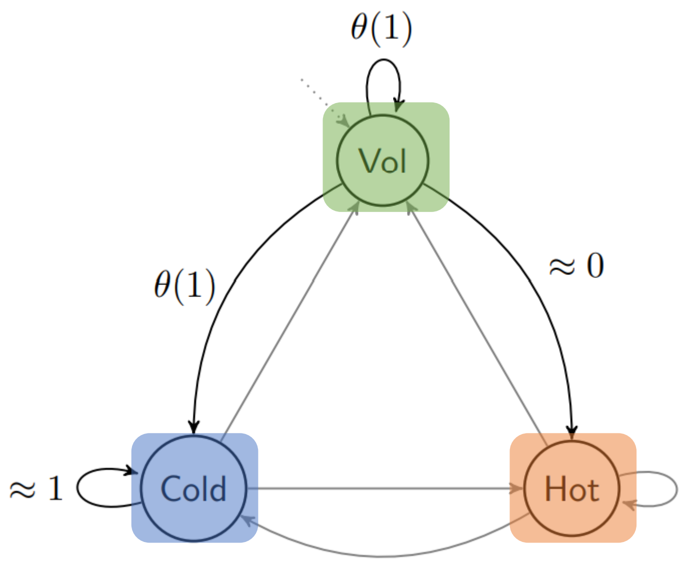
▪Volatile$_t$: $-\delta\sqrt{n} \leq M_{(t)} \leq L_{(t)} \leq \delta\sqrt{n}$

▪Cold$_t$: $M_{(t)} \leq -\delta\sqrt{n}$

2D Random Walk Analysis (1)

- Goal: $\Pr[w\ is\ forkable] = 2^{-\Omega(\sqrt{n})}$
- $R_{(t)} = \rho(w_1 \dots w_t)$ and $M_{(t)} = \mu(w_1 \dots w_t)$.
- **Hot$_t$:** $R_{(t)} \geq \delta\sqrt{n} \wedge M_{(t)} \geq -\delta\sqrt{n}$
- **Volatile$_t$:** $-\delta\sqrt{n} \leq M_{(t)} \leq L_{(t)} \leq \delta\sqrt{n}$
- **Cold$_t$:** $M_{(t)} \leq -\delta\sqrt{n}$

- $\Pr\left[\mathrm{Cold}_{(t+1)} \middle| \mathrm{Cold}_{(t)}\right] \geq 1 - 2^{-\Omega(\sqrt{n})} \Longrightarrow$ overwhelming
- $\Pr\left[\mathrm{Cold}_{(t+1)} \middle| \mathrm{Vol}_{(t)}\right] \geq \Omega(\epsilon) \Longrightarrow$ constant
- $\Pr\left[\mathrm{Hot}_{(t+1)} \middle| \mathrm{Vol}_{(t)}\right] \leq 2^{-\Omega(\sqrt{n})} \Longrightarrow$ negligible

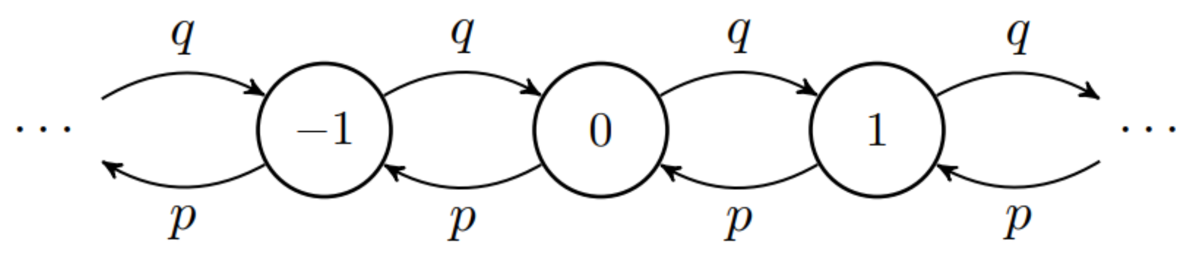


2D Random Walk Analysis (2)

- $\Pr\left[\mathrm{Cold}_{(t+1)} \middle| \mathrm{Cold}_{(t)}\right] \geq 1 - 2^{-\Omega(\sqrt{n})} \Longrightarrow$ overwhelming

$$M_t < 0 \Longrightarrow \begin{cases} M_{t+1} = M_t + 1 & \text{if } w_{t+1} = 1, \\ M_{t+1} = M_t - 1 & \text{if } w_{t+1} = 0; \end{cases}$$

- **Margin** performs a simple random walk when negative.
- The stake is honest majority, so $\Pr[\mathrm{w}_i = 0] > (1 + \epsilon) / 2$. The simple biased walk where $p = (1 + \epsilon)/2$ and $q = 1 - p$.



2D Random Walk Analysis (3)

- $\Pr\left[\mathrm{Cold}_{(t+1)} \middle| \mathrm{Cold}_{(t)}\right] \geq 1 - 2^{-\Omega(\sqrt{n})} \Longrightarrow$ overwhelming
- **Gambler's ruin**: a gambler playing a negative expected value game will eventually go broke, regardless of their betting system.
- Let denote $Z_i \in \{\pm 1\}$(for $\mathrm{i} = 1,2, \ldots$) a family of independent random variables for which $\Pr[Z_i = 1] = (1 - \epsilon)/2$. Then the biased walk given by the variables $Y_t = \sum_1^t Z_i$ has the following property:
  - **Constant escape probability**. With constant probability, depending only on ϵ, $Y_t \neq 1$, for all $t > 0$. In general, for each $\mathrm{k} > 0$, $\alpha = \frac{1-\epsilon}{1+\epsilon} < 1$,

$$\Pr[\exists t, Y_t = k] = \alpha^k.$$

2D Random Walk Analysis (4)

- $\Pr\left[\text{Cold}_{(t+1)} \middle| \text{Cold}_{(t)}\right] \geq 1 - 2^{-\Omega(\sqrt{n})} \Longrightarrow$ overwhelming
- Proof sketch:
  - Conditioned on $M_{(t)} \ = M_{a_t} < -\delta\sqrt{n}$, the probability that any future M_s ever climbs to value -1 is no more than $2^{-\Omega(\sqrt{n})}$.
  - There are at most $\sqrt{n}$ times this could happen, so

$$\Pr\left[\text{Cold}_{(t+1)} \middle| \text{Cold}_{(t)}\right] = (1 - 2^{-\Omega(\sqrt{n})})^{\sqrt{n}} \geq 1 - 2^{-\Omega(\sqrt{n})}.$$

2D Random Walk Analysis (5)

- $\Pr\left[\mathrm{Cold}_{(t+1)} \middle| \mathrm{Vol}_{(t)}\right] \geq \Omega(\epsilon) \Longrightarrow$ constant
- $\Pr\left[\mathrm{Hot}_{(t+1)} \middle| \mathrm{Vol}_{(t)}\right] \leq 2^{-\Omega(\sqrt{n})} \Longrightarrow$ negligible

$$\mathbf{m}(\epsilon) = (0,0) \textit{ and, for all nonempty strings } w \in \{0,1\}^*,$$

$$\mathbf{m}(w1) = (\rho(w)+1, \mu(w)+1) \text{ , } \textit{and}$$

$$\mathbf{m}(w0) = \begin{cases} (\rho(w)-1, 0) & \textit{if } \rho(w) > \mu(w) = 0, \\ (0, \mu(w)-1) & \textit{if } \rho(w) = 0, \\ (\rho(w)-1, \mu(w)-1) & \textit{otherwise.} \end{cases}$$

- **Reach** performs a simple random walk when positive.
- **Margin** performs a simple random walk when negative.
- **Margin** sticks to 0 when **reach** is positive.

2D Random Walk Analysis (6)

- $\Pr\left[\text{Cold}_{(t+1)} \middle| \text{Vol}_{(t)}\right] \geq \Omega(\epsilon) \Longrightarrow$ constant
- $\Pr\left[\text{Hot}_{(t+1)} \middle| \text{Vol}_{(t)}\right] \leq 2^{-\Omega(\sqrt{n})} \Longrightarrow$ negligible

- **Hot**$_t$: $R_{(t)} \geq \delta\sqrt{n} \wedge M_{(t)} \geq -\delta\sqrt{n}$
- **Volatile**$_t$: $-\delta\sqrt{n} \leq M_{(t)} \leq L_{(t)} \leq \delta\sqrt{n}$
- **Cold**$_t$: $M_{(t)} \leq -\delta\sqrt{n}$

$$R_t > 0 \Longrightarrow \begin{cases} R_{t+1} = R_t + 1 & \text{if } w_{t+1} = 1, \\ R_{t+1} = R_t - 1 & \text{if } w_{t+1} = 0; \end{cases}$$

$$M_t < 0 \Longrightarrow \begin{cases} M_{t+1} = M_t + 1 & \text{if } w_{t+1} = 1, \\ M_{t+1} = M_t - 1 & \text{if } w_{t+1} = 0; \end{cases}$$

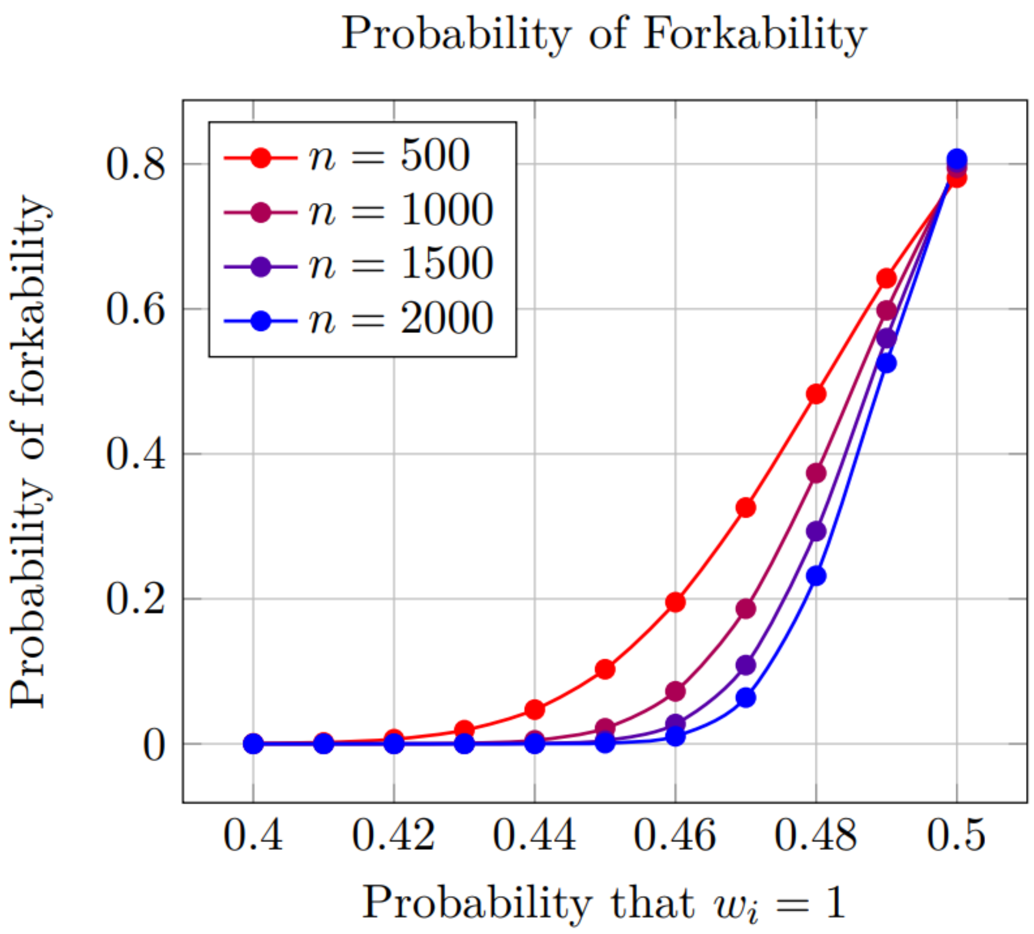
$$R_t = 0 \Longrightarrow \begin{cases} R_{t+1} = 1 & \text{if } w_{t+1} = 1, \\ R_{t+1} = 0 & \text{if } w_{t+1} = 0, \\ M_{t+1} < 0 & \text{if } w_t = 0. \end{cases}$$

- **Concentration (the Chernoff bound).** Consider T steps of the biased walk beginning at state 0; then the resulting value is tightly concentrated around $-\epsilon T$. Specifically, $\mathrm{E}[Y_T] = -\epsilon T$ and $\Pr\left[Y_T > -\frac{\epsilon T}{2}\right] = 2^{-\Omega(T)}$.

2D Random Walk Analysis (7)

- $\Pr[w\ is\ forkable] = \Pr[M_n \geq 0] = 2^{-\Omega(\sqrt{n})}$
- A more careful analysis using martingales can show a better error bound of $2^{-\Omega(n)}$
  - Blum, Erica & Kiayias, Aggelos & Moore, Cristopher & Quader, Saad & Russell, Alexander. (2019). Linear Consistency for Proof-of-Stake Blockchains. https://eprint.iacr.org/2017/241

Forkable Density



Blockchain properties

Common Prefix (CP); with parameter $k \in \mathbb{N}$. The chains $\mathcal{C}_1, \mathcal{C}_2$ adopted by two honest parties at the onset of the slots $sl_1 \leq sl_2$ are such that $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$, where $\mathcal{C}_1^{\lceil k}$ denotes the chain obtained by removing the last k blocks from $\mathcal{C}_1$, and $\preceq$ denotes the prefix relation.

Honest Chain Growth (HCG); with parameters $\tau \in (0, 1]$ and $s \in \mathbb{N}$. Consider the chain $\mathcal{C}$ adopted by an honest party. Let sl_2 be the slot associated with the last block of $\mathcal{C}$ and let sl_1 be a prior slot in which $\mathcal{C}$ has an honestly-generated block. If $sl_2 \geq sl_1 + s$, then the number of blocks appearing in $\mathcal{C}$ after sl_1 is at least τs. The parameter τ is called the speed coefficient.

Existential Chain Quality ($\exists$CQ); with parameter $s \in \mathbb{N}$. Consider the chain $\mathcal{C}$ adopted by an honest party at the onset of a slot and any portion of $\mathcal{C}$ spanning s prior slots; then at least one honestly-generated block appears in this portion.

Today's Topics

- Proof-of-Stake Background
- Ouroboros
  - Protocol Execution, characteristic String and Forks
  - Security Analysis
  - Dynamic Stake
- Ouroboros Genesis
  - Bootstrapping from genesis

Dynamic Stake with a Beacon

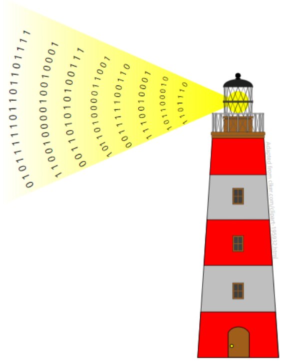
- Beacon: a randomness beacon is a functionality that emits random values at regular intervals.
- A cryptographic implementation of a beacon ahs the properties that:
  - (i) the beacon values cannot be predicted ahead of time by the adversary.
  - (ii) the beacon cannot be stifled.
- Why a beacon is useful in our setting?
  - As stake evolves over time, we need to be sure that fresh randomness "enters" the system and refreshes the test used for determining eligibility of participation.
  - If this is not available, an obvious attack can be mounted: perform rejection sampling using KeyGen(·) of the VRF until a suitable key vk is produced that wins the next round, then transfer funds to that account.

Dynamic Stake with a Beacon

- The Randomness Beacons project at NIST intends to promote the availability of trusted public randomness as a public utility. Such utility can be used for example to promote auditability and transparency of services that depend on randomized processes.
  - https://csrc.nist.gov/projects/interoperable-randomness-beacons

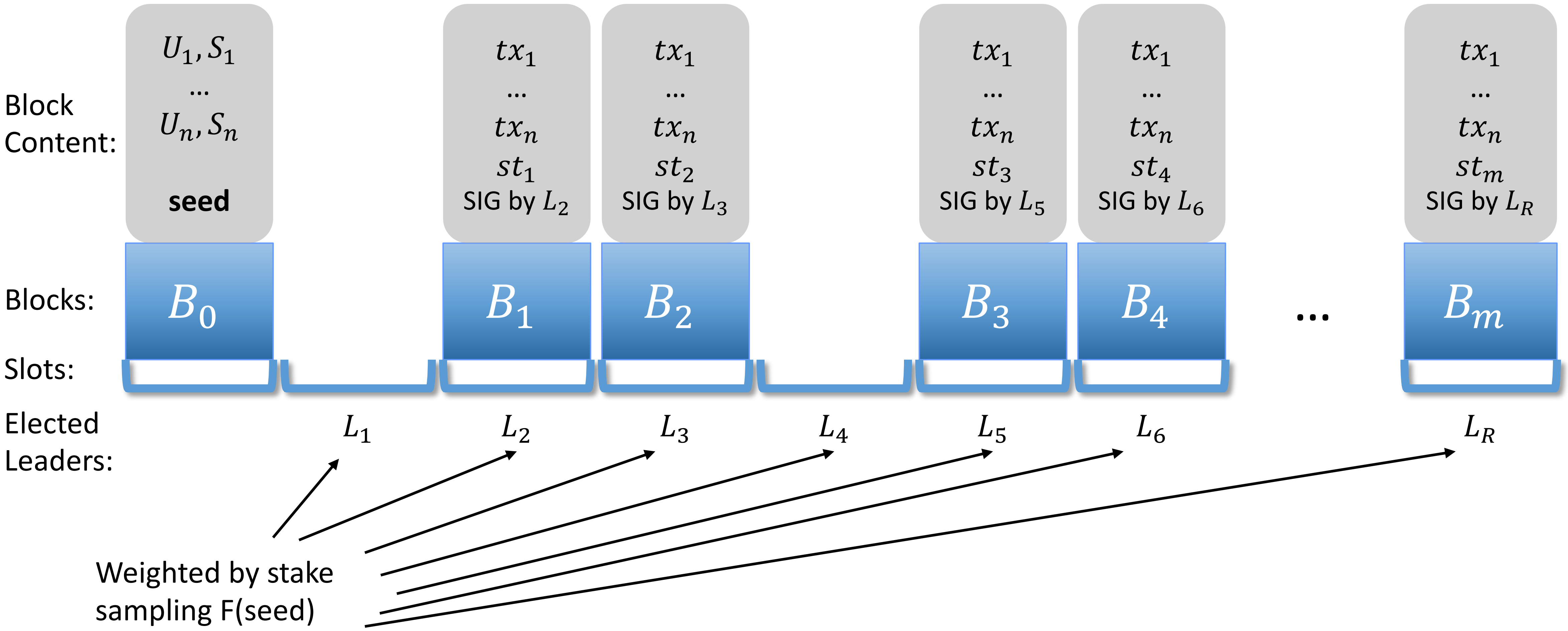
Some features of a beacon, as defined by the new reference:

- Periodically pulsates randomness (e.g., once a minute).
- Each pulse has a fresh 512-bit random string, cryptographically combining entropy from at least two separate random number generators (RNGs).
- Each pulse is indexed, time-stamped and signed.
- Any past pulse is publicly accessible.
- The sequence of pulses forms a hash chain.
- Far-apart pulses can be efficiently verified via a short chain (skiplist).
- A pre-commitment of local randomness enables securely combining randomness from multiple beacons.



Ouroboros: Static Stake

RECALL



Ouroboros: Dynamic Stake

U_1, S_1

…

U_n, S_n

R

B_0

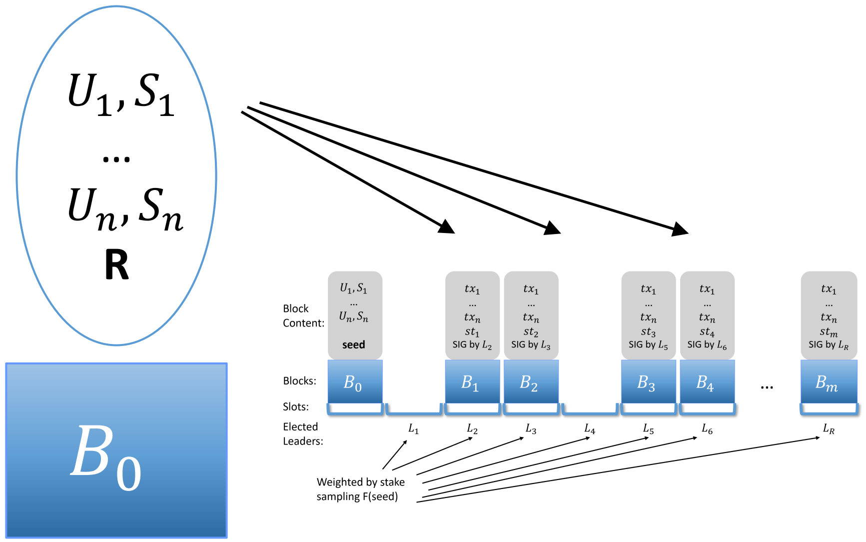
R

R’

R’’

Randomness beacon

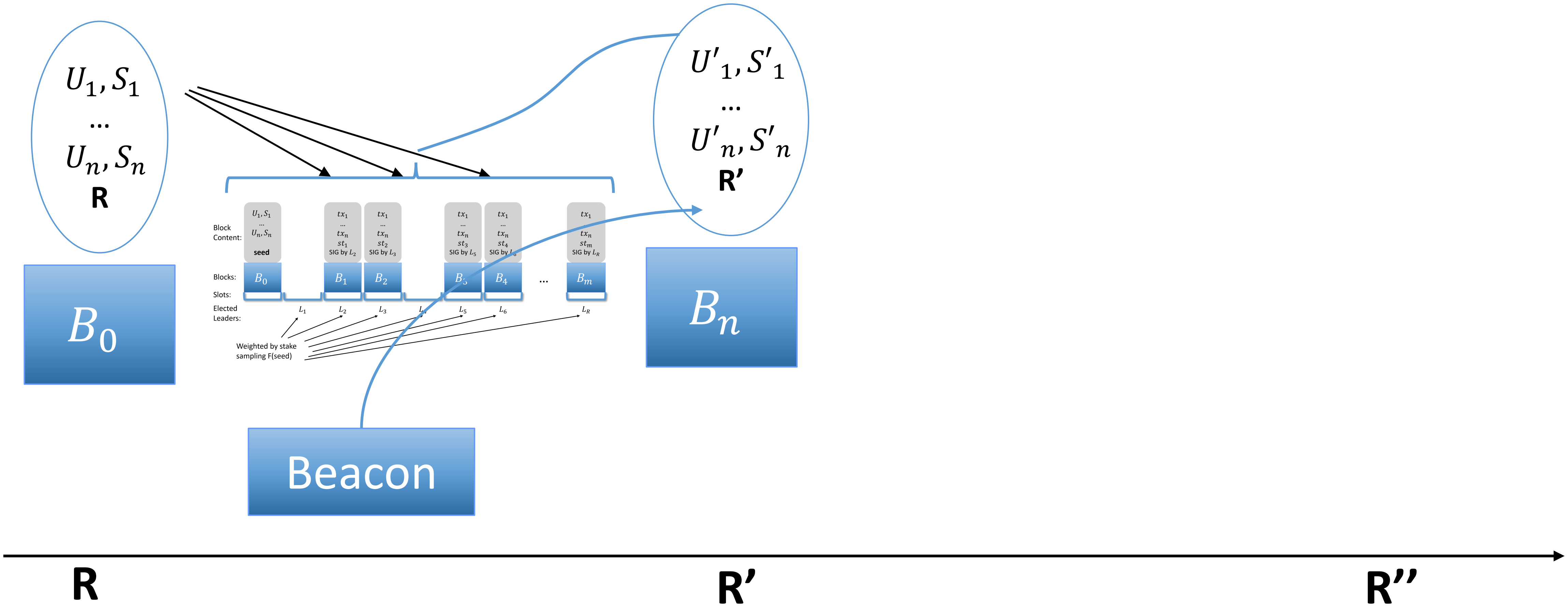
Ouroboros: Dynamic Stake



R **R’** **R’’**

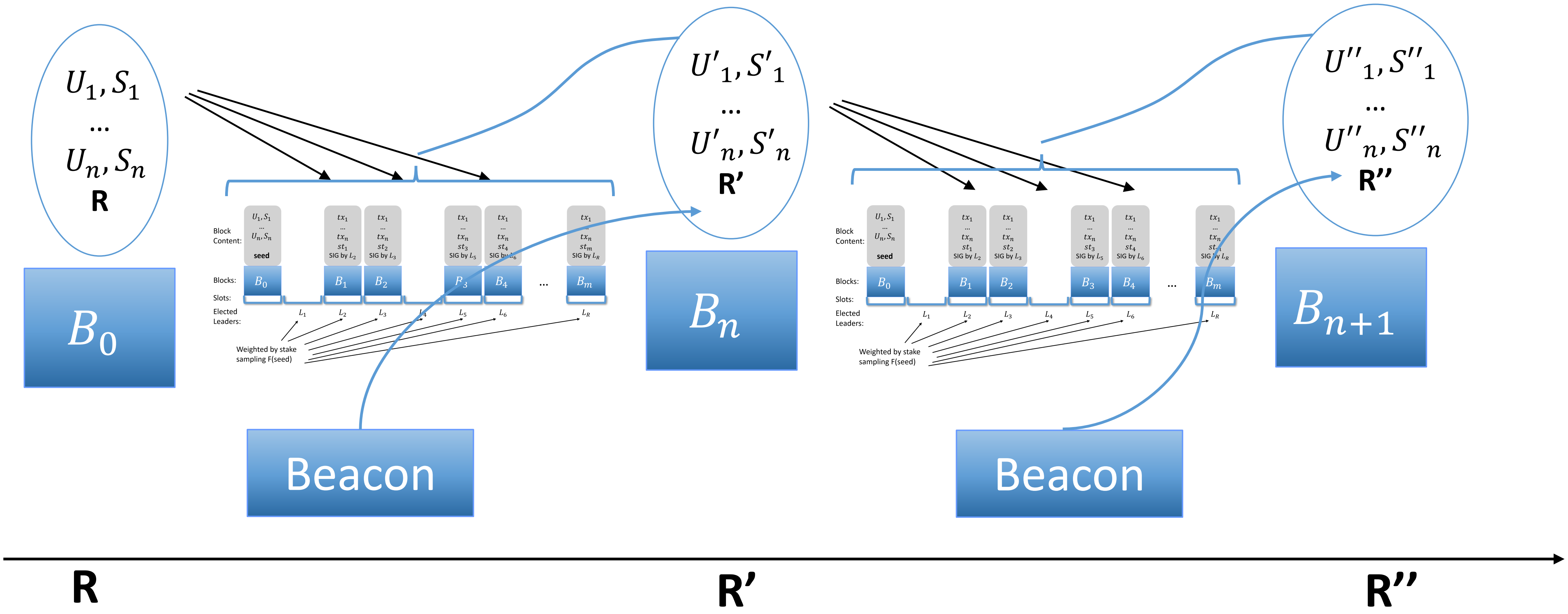
Randomness beacon

Ouroboros: Dynamic Stake



Randomness beacon

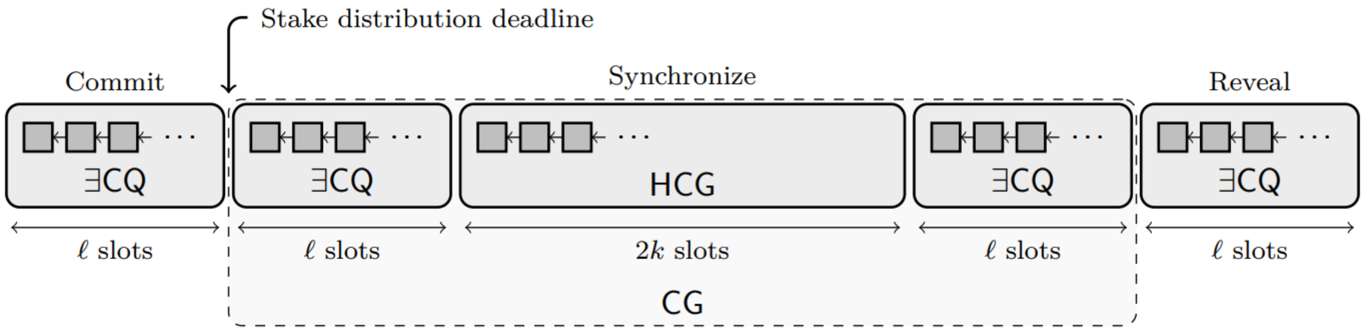
Ouroboros: Dynamic Stake



Randomness beacon

Simulating random beacon

- Coin Tossing Protocol
- For every stake holder, when each epoch will end:



- Use publicly verifiable secret sharing (PVSS) for distributed commitment openings.
- Discrete logarithm based.

Adaptive adversaries

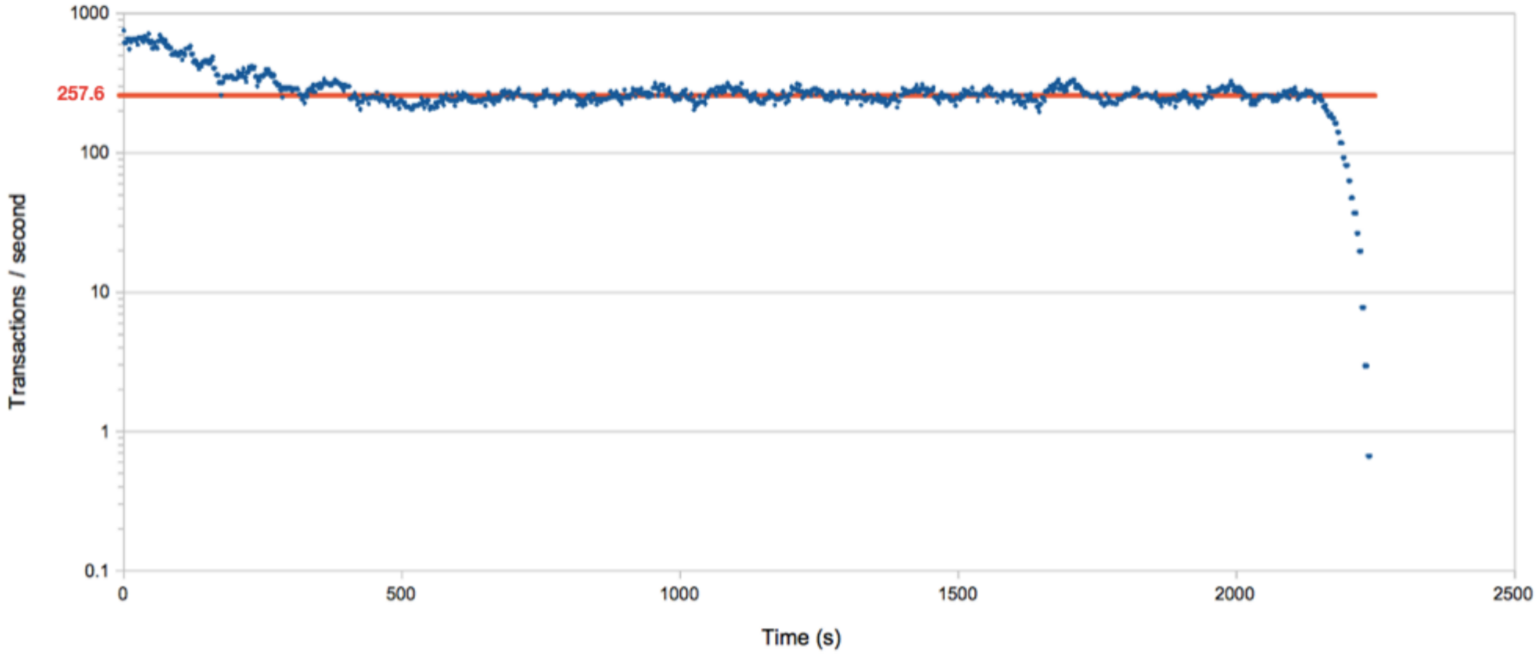
- In PoW, if a wallet is corrupted, it will never convert the previous blockchain.
- In PoS, if a wallet is corrupted, the adversary can re-construct the blockchain using those sign keys *sk* if they were once elected as the slot leader.
- KES is necessary to enable parties to erase key material so that when a wallet gets corrupted at some point in the protocol execution, past protocol steps depending on that wallet cannot be recreated.

Key evolving signatures (KES)

- A KES is comprised of four algorithms (KeyGen, Sign, Update, Verify)
  - KeyGen(1^n) $\Longrightarrow$ (vk, sk[0])
  - Sign(sk[i], m) $\Longrightarrow$ σ
  - Update(sk[i]) $\Longrightarrow$ sk[i+1]
  - Verify(vk, i, m, σ) $\Longrightarrow$ {0, 1}
- Intuitive properties:
  - Operates as a regular digital signature (unforgeability under existential chosen message attack)
  - Each index *i* can be viewed as a distinct epoch
  - Corruption of secret-key at epoch i does not jeopardize unforgeability at epochs < *i*
- Mihir Bellare, Sara K. Minery. A Forward-Secure Digital Signature Scheme. CRYPTO 1999.

Ouroboros Performance

- Measuring transactions per second in a 40 node, equal stake deployment with slot length of 5 seconds.



Today's Topics

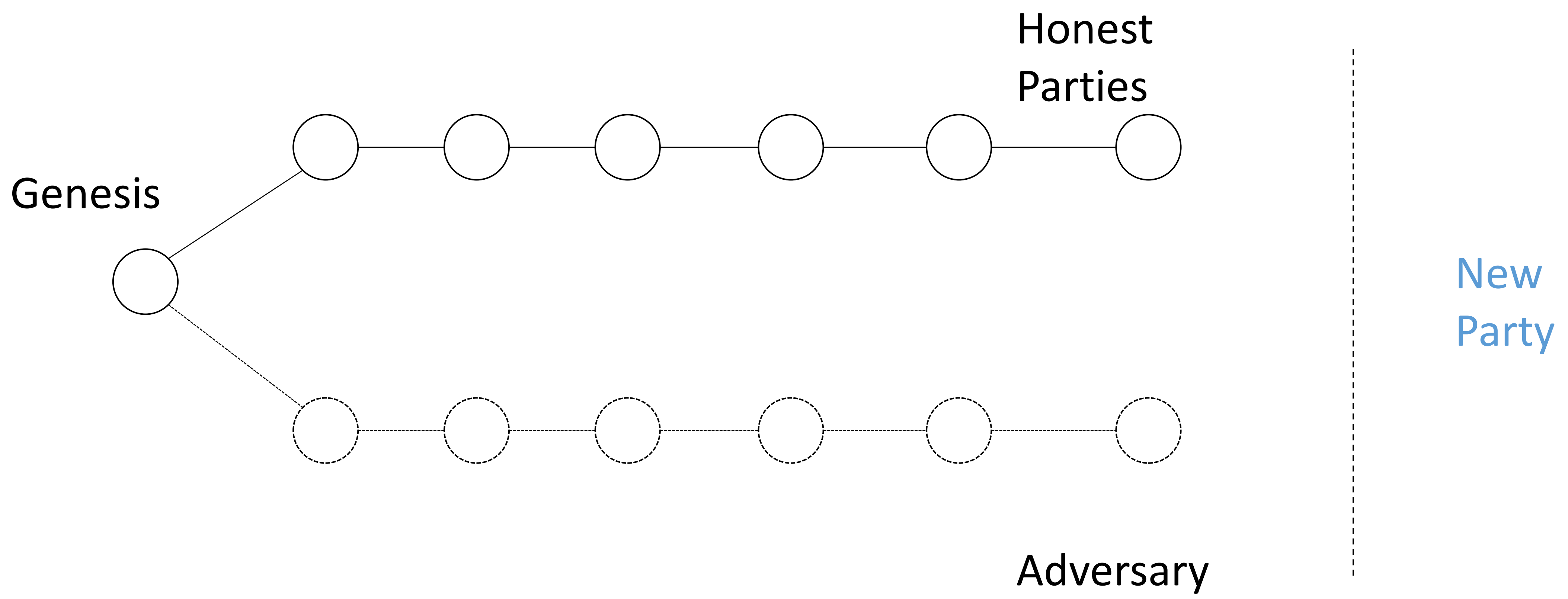
- Proof-of-Stake Background
- Ouroboros
  - Protocol Execution, characteristic String and Forks
  - Security Analysis
  - Dynamic Stake
- Ouroboros Genesis
  - Bootstrapping from genesis

A folklore perspective

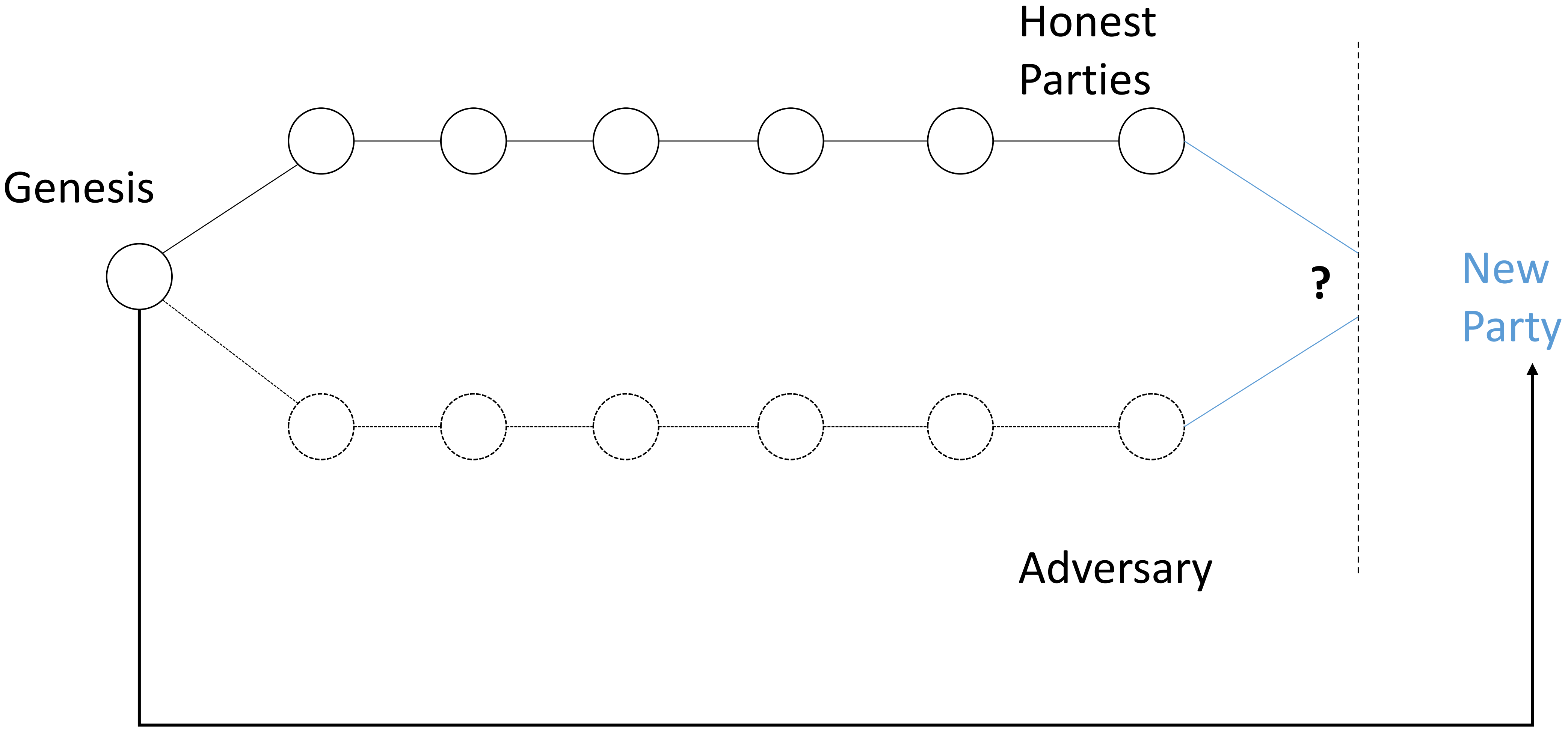
RECALL

- PoS blockchains are impossible to work in the setting where Bitcoin operates.
- Reasons:
  - Costless simulation.
    - Given no physical resources are used in producing blocks, it is possible to build alternative transaction histories at essentially no cost.
    - nothing at stake
  - Long-range attacks.
    - In long-range attack the victim tries to distinguish between two alternative histories furnished by the network without any recent information.
    - The bootstrapping problem: how does a new (or long term desynchronized) node synchronize with the blockchain?

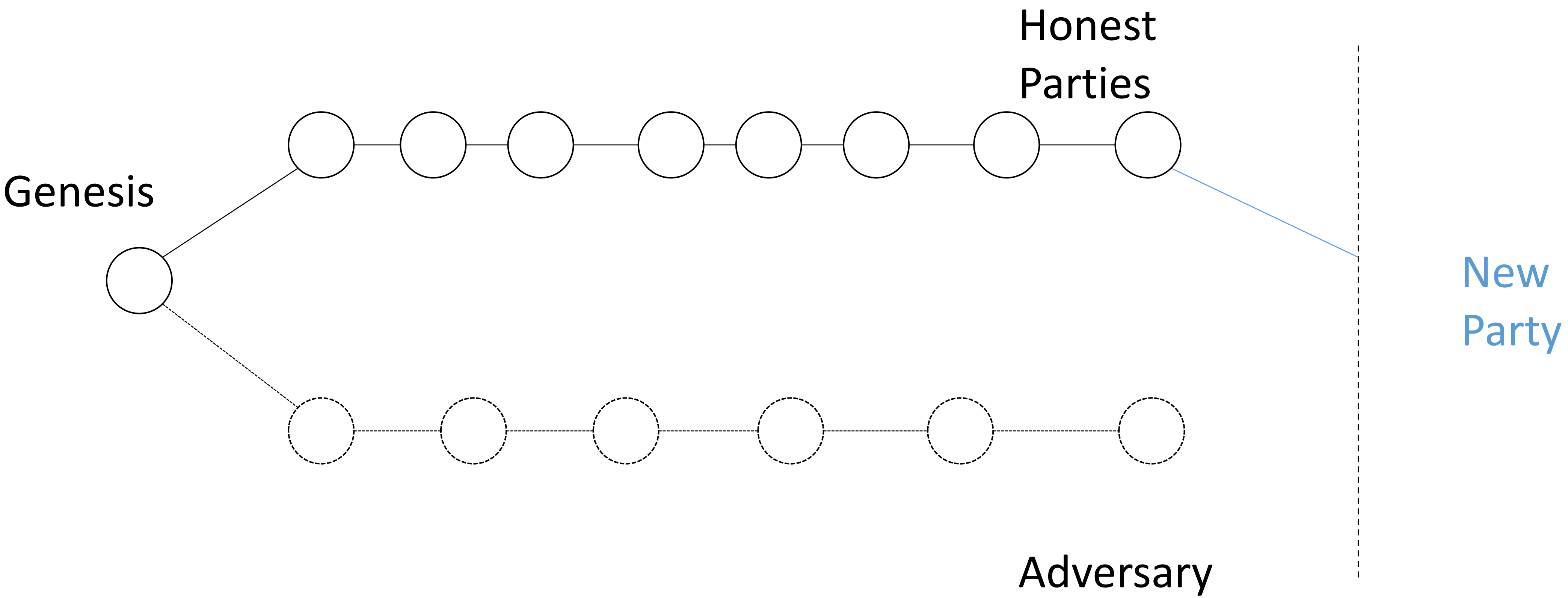
Bootstrapping from Genesis



Bootstrapping from Genesis



The PoW Approach



- Adversarial version will be substantially “shorter” (counting difficulty as length).

Maxvalid-mc

Protocol maxvalid-mc($\mathcal{C}_{\mathsf{loc}}, \mathcal{C}_1, \ldots, \mathcal{C}_\ell$)

```
1: Set C_max ← C_loc.
2: for i = 1 to ℓ do
3:     if IsValidChain(C_i) then
   // Compare C_max to C_i
4:         if (C_i forks from C_max at most k blocks) then
5:             if |C_i| > |C_max| then // Condition A
                   Set C_max ← C_i.
               end if
           end if
       end if
   end for
6: return C_max.
```

Dynamic Availability

- The permissionless environment where:
  - Parties join and leave at will.
  - Number of online/offline parties dynamically change over time, or lose clock synchronization, network connection.
  - Protocol does not have a-priori knowledge of the participation level.

	Basic types of *honest* parties	
Resource	**Resource unavailable**	**Resource available**
random oracle $\mathcal{G}_{\mathrm{RO}}$	*stalled*	*operational*
network $\mathcal{F}_{\text{N-MC}}$	*offline*	*online*
clock $\mathcal{G}_{\textsc{PerfLClock}}$	*time-unaware*	*time-aware*
synchronized state, local time	*desynchronized*	*synchronized*
KES capable of signing (w.r.t. local time)	*sign-capable*	*sign-uncapable*

Derived types:

$$\begin{aligned} \textit{alert} &:\Leftrightarrow \textit{operational} \wedge \textit{online} \wedge \textit{time-aware} \wedge \textit{synchronized} \wedge \textit{sign-capable} \\ \textit{active} &:\Leftrightarrow \textit{alert} \vee \textit{adversarial} \vee \textit{time-unaware} \end{aligned}$$

Note: *alert* parties are honest, *active* parties also contain all adversarial parties.

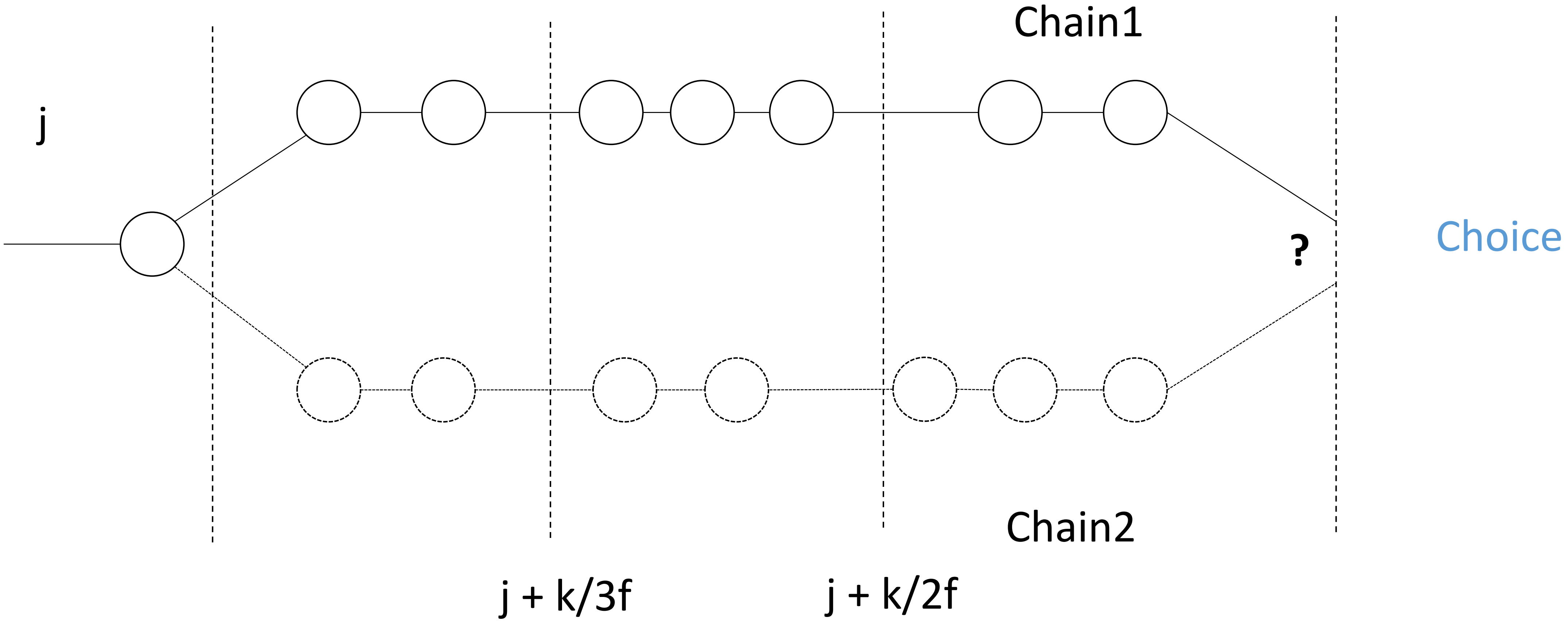
PoW vs. PoS (previously)

	Bitcoin	PoS Blockchain	PoS BFT
Setup Assumption	Common Random String	Public-key Directory	Public-key Directory
Long Range Attack	Longest(heaviest) chain rule	Longest chain rule + moving local checkpoint + Key Evolving Signature	Only one proper sequence of block certificates + Key Evolving Signature
Dynamic Availability	Possible using only the genesis block	(re)joining parties need a somewhat recent block	Parties need to know the participation level at all times in history

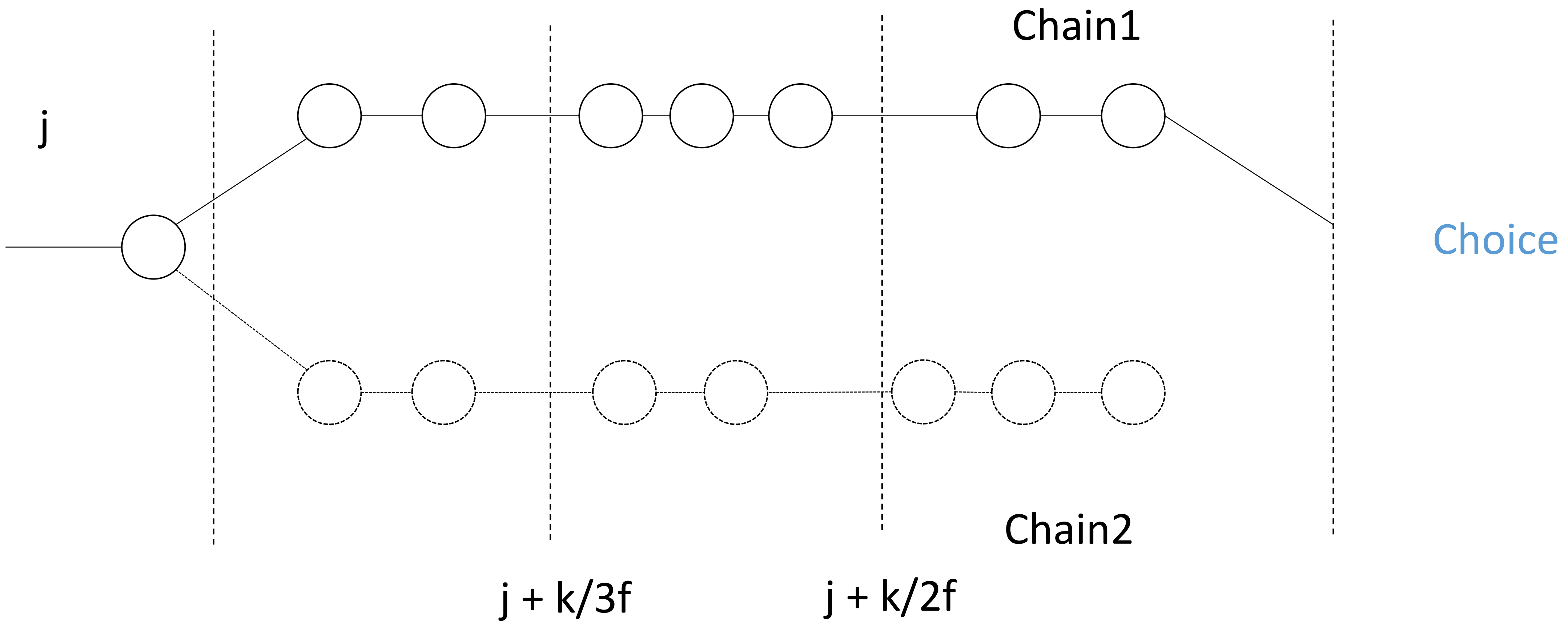
Ouroboros Genesis

- Main novel feature: new chain selection rule that enables parties to bootstrap from genesis.
  - Short range comparisons: (chains diverge up to k blocks) Nodes follow longest chain.
  - Long range comparisons: (chains diverge more than k blocks in the past) Nodes use a plenitude rule to pick the right chain.

Long Range Comparisons



Long Range Comparisons



Intuition for Plenitude Rule

- If majority of parties follow the protocol, then at any sufficiently long time segment, the corresponding chain will be more dense.

- Ouroboros Genesis proves: adversarial blockchains shortly after the divergence point will exhibit a less dense block distribution.

Maxvalid-bg

Algorithm maxvalid-bg$(\mathcal{C}_{\mathsf{loc}}, \mathcal{N} = \{\mathcal{C}_1, \ldots, \mathcal{C}_M\}, k, s, f)$

// Compare $\mathcal{C}_{\mathsf{max}}$ to each $\mathcal{C}_i \in \mathcal{N}$

Set $\mathcal{C}_{\mathsf{max}} \leftarrow \mathcal{C}_{\mathsf{loc}}$.

for $i = 1$ to M **do**

  if ($\mathcal{C}_i$ forks from $\mathcal{C}_{\mathsf{max}}$ at most k blocks) **then**

    if $|\mathcal{C}_i| > |\mathcal{C}_{\mathsf{max}}|$ **then** // Condition A

      Set $\mathcal{C}_{\mathsf{max}} \leftarrow \mathcal{C}_i$.

    end if

  else

    Let $j \leftarrow \max\{j' \geq 0 \mid \mathcal{C}_{\mathsf{max}}$ and $\mathcal{C}_i$ have the same block in $\mathtt{sl}_{j'}\}$

    if $\left|\mathcal{C}_i[0:j+s]\right| > \left|\mathcal{C}_{\mathsf{max}}[0:j+s]\right|$ **then** // Condition B

      Set $\mathcal{C}_{\mathsf{max}} \leftarrow \mathcal{C}_i$.

    end if

  end if

end for

return $\mathcal{C}_{\mathsf{max}}$.

PoW vs. PoS (now)

	Bitcoin	PoS Blockchain	PoS BFT
Setup Assumption	Common Random String	Public-key Directory	Public-key Directory
Long Range Attack	Longest(heaviest) chain rule	Longest chain rule + ***plenitude rule***+ Key Evolving Signature	Only one proper sequence of block certificates + Key Evolving Signature
Dynamic Availability	Possible using only the genesis block	***Ouroboros Genesis feasible using only the genesis block***	Parties need to know the participation level at all times in history

Ouroboros Chronos

- Ouroboros Genesis's chain selection rule has 2 assumptions:
  - Every party can get the genesis block
  - Every party synchronize with the global clock
- Freshly (re-)joining parties should have a common notion of a global clock.
- Strong synchrony assumption.
- Ouroboros solves the global synchronization problem by leveraging proof of stake.
  - Parties have local clocks advancing at approximately the same speed. After joing the protocol, they can quickly calibrate their local clocks so that they all show approximately the same time.
  - Parties can passively participate in the PoS protocol to calibrate their local clock.