EPFL

Consensus for Decentralized Ledgers

Prof. Bryan Ford
Decentralized and Distributed Systems (DEDIS)
School of Information and Communications (IC)
dedis@epfl.ch – dedis.epfl.ch

Dagstuhl seminar
“Rigorous Methods for Smart Contracts”
October 25, 2021

Talk Outline

Some blockchain consensus challenges and work

- Classic (permissioned) versus permissionless
- Latency and throughput scalability
- Practical asynchronous consensus
- Participation basis: investment or personhood?
- Smart contract execution, programming model

Distributed Ledgers or Blockchains

Problem: we don't want to trust any designated, centralized authority to maintain the ledger

Alice	5 BTC
Bob	2 BTC
Charlie	3 BTC
...	

Solution: “everyone” keeps a copy of the ledger!

- Everyone checks everyone else's changes to it

Alice's copy

Alice	5 BTC
Bob	2 BTC
Charlie	3 BTC
...	

Bob's copy

Alice	5 BTC
Bob	2 BTC
Charlie	3 BTC
...	

Charlie's copy

Alice	5 BTC
Bob	2 BTC
Charlie	3 BTC
...	

Applications of Distributed Ledgers

Can represent a distributed electronic record of:

- Who owns how much **currency**? (Bitcoin)
- Who owns a name or a digital work of art?
- What are the terms of a **contract**? (Ethereum)
- When was a **document** written? (notaries)

- What is the **provenance** of a part? (supply chain)
- Who **are** you? (self-sovereign identity)
- Who used **data** for what purpose? (access logs)
- …

Consensus for Ledgers

Key considerations and often-desired goals

- Security against adversarial network, nodes
- Commitment finality
- Commitment latency
- Scalability to high transaction load
- Scalability to many participants
- Bandwidth, computation, power efficiency
- Open “permissionless” participation
- Equitable, “fair” distribution of power/rewards

Talk Outline

Some blockchain consensus challenges and work

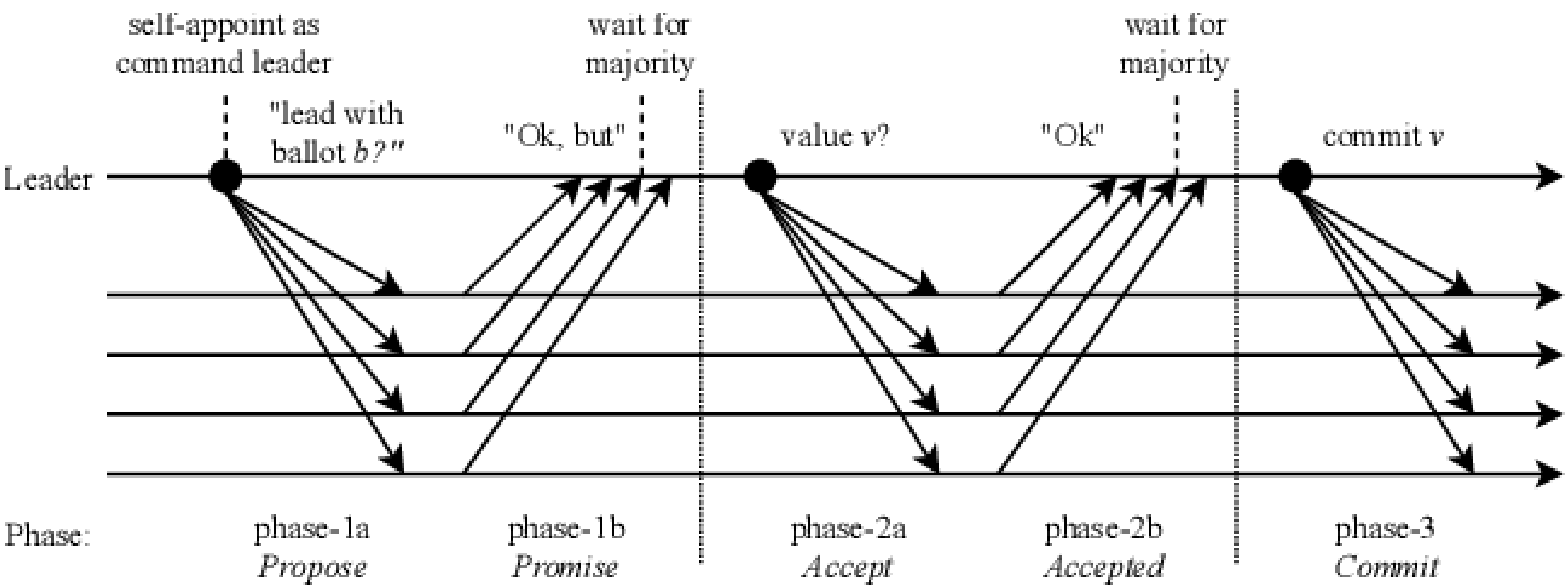
- **Classic (permissioned) versus permissionless**
- Latency and throughput scalability
- Practical asynchronous consensus
- Participation basis: investment or personhood?
- Smart contract execution, programming model


In the beginning...
there was Paxos

Paxos (Leslie Lamport)

Ubiquitous, practical for small consensus groups

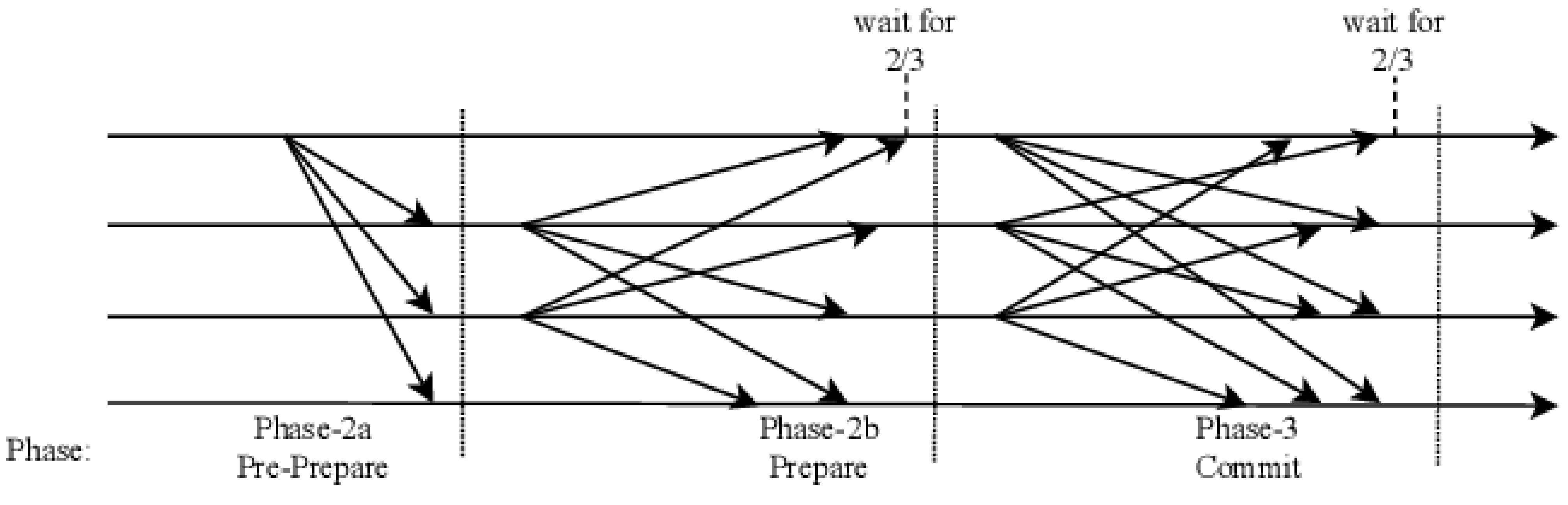
- Assumes well-defined group ("permissioned")
- Not robust to adversarial nodes *or* networks



Robustness to adversarial nodes

Practical Byzantine Fault Tolerance (PBFT)

- Tolerates <1/3 adversarial group members
- Reasonably practical for small groups
- Leader-driven: vulnerable to DoS attacks



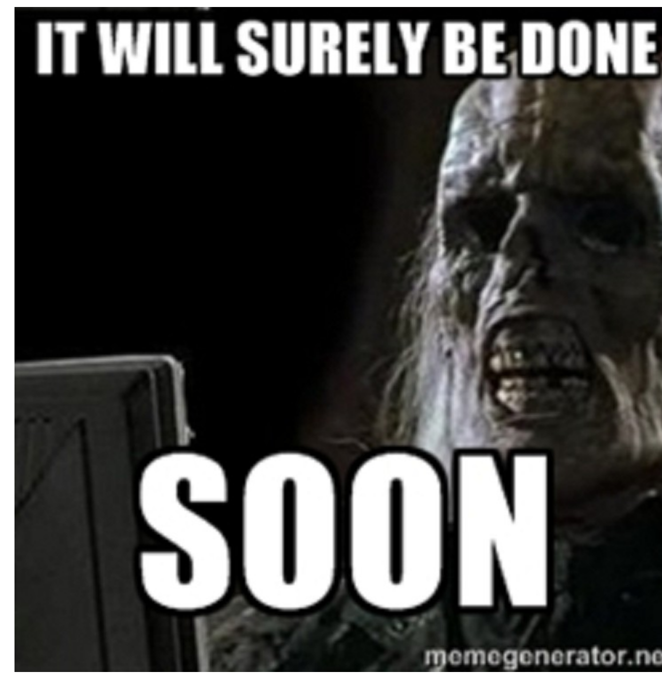
Open “Permissionless” Consensus

Bitcoin’s consensus - groundbreaking in 2 ways:

- Allow “anyone” to participate via proof of work
- Scalable to thousands of participants, not 3-10

Bitcoin's openness had many costs

- **Transaction delay**
  - Any transaction takes ~10 mins *minimum* in Bitcoin
- **Weak consistency/finality:**
  - You're not *really* certain your transaction is committed until you wait ~1 hour or more



- **Low throughput:**
  - Bitcoin: ~7 transactions/second
- **Proof-of-work mining:**
  - Enormous energy wasted in useless arms race

Talk Outline

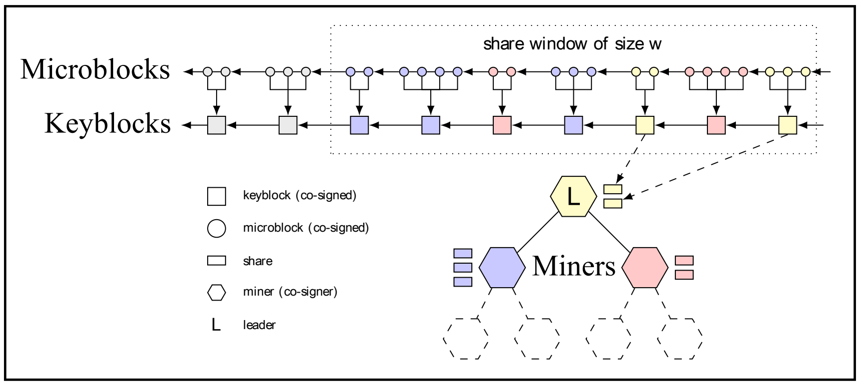
Some blockchain consensus challenges and work

- Classic (permissioned) versus permissionless
- **Latency and throughput scalability**
- Practical asynchronous consensus
- Participation basis: investment or personhood?
- Smart contract execution, programming model

Scaling Blockchains is Not Easy

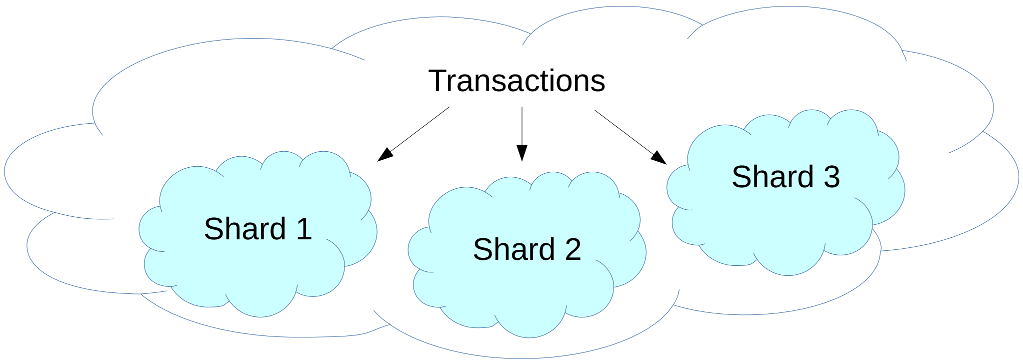
Many Approaches to Scaling

Scalable BFT

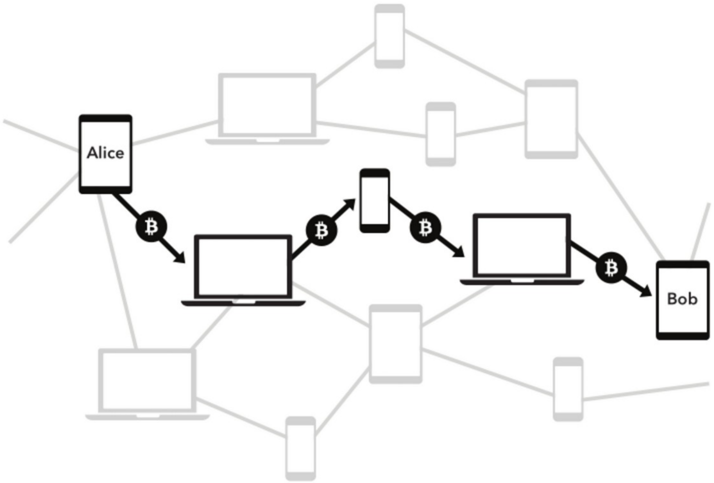


Sidechains

Horizontal Sharding



Payment Networks

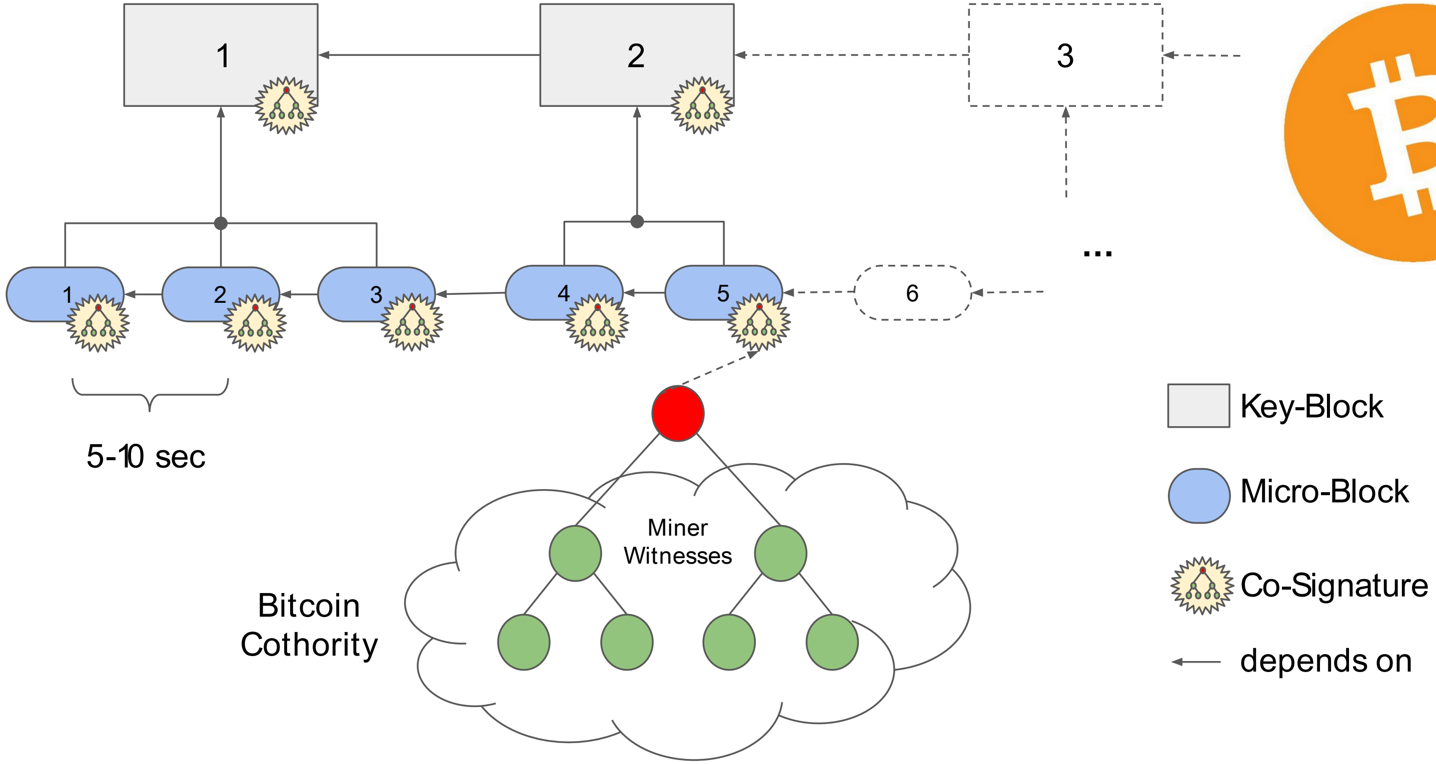


ByzCoin: scaling PBFT to open systems

Use PoW to pick rotating groups [USENIX Security ‘16]

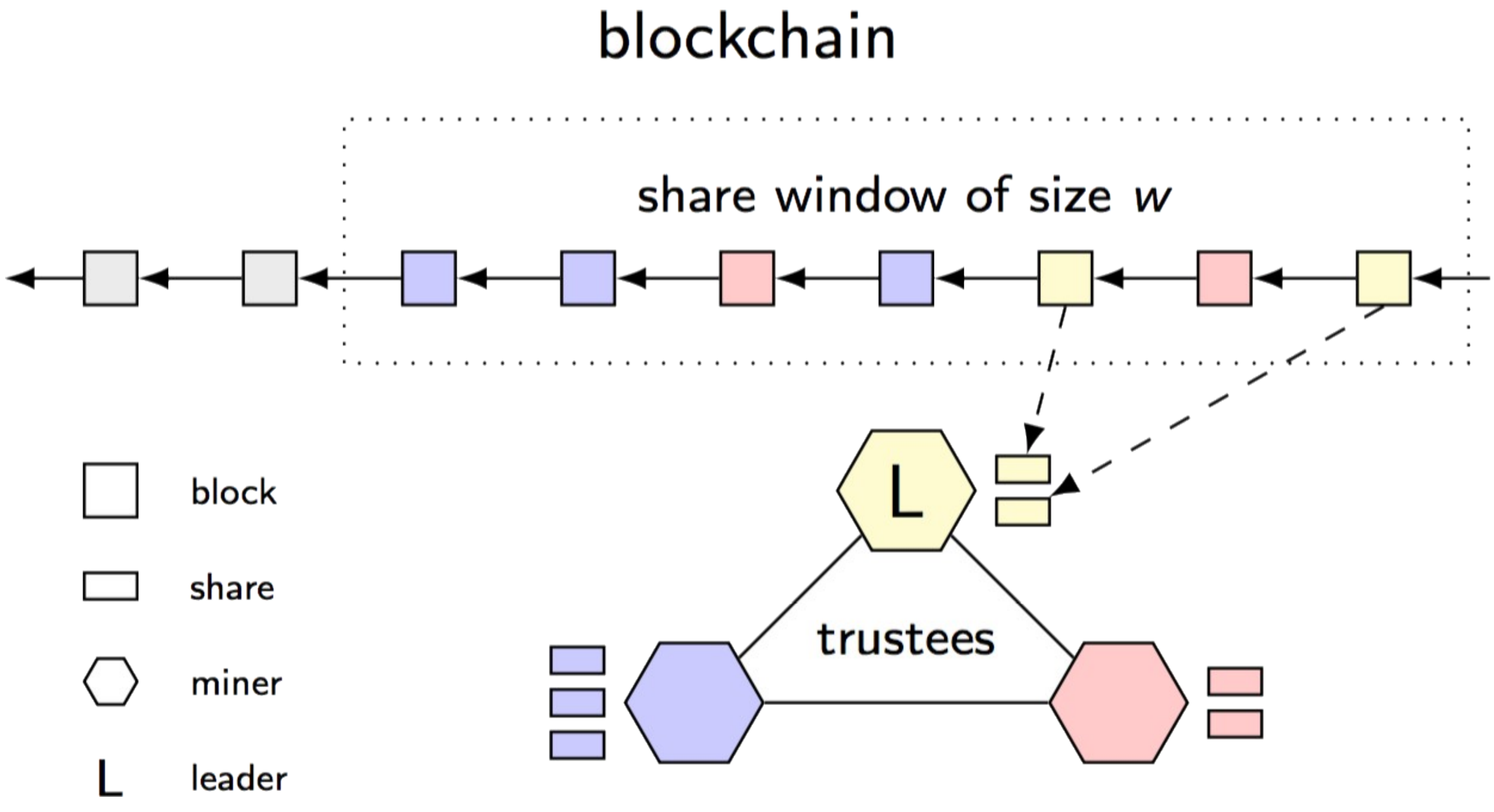
- Permanent transaction commitment in seconds
- 700+ TPS demonstrated (100x Bitcoin, ~PayPal)

Closely-related: Hybrid Consensus by Pass/Shi



ByzCoin Consensus Windows

Keeps Bitcoin's proof-of-work (PoW), but mining yields **temporary membership share** in a gradually-rotating consensus group

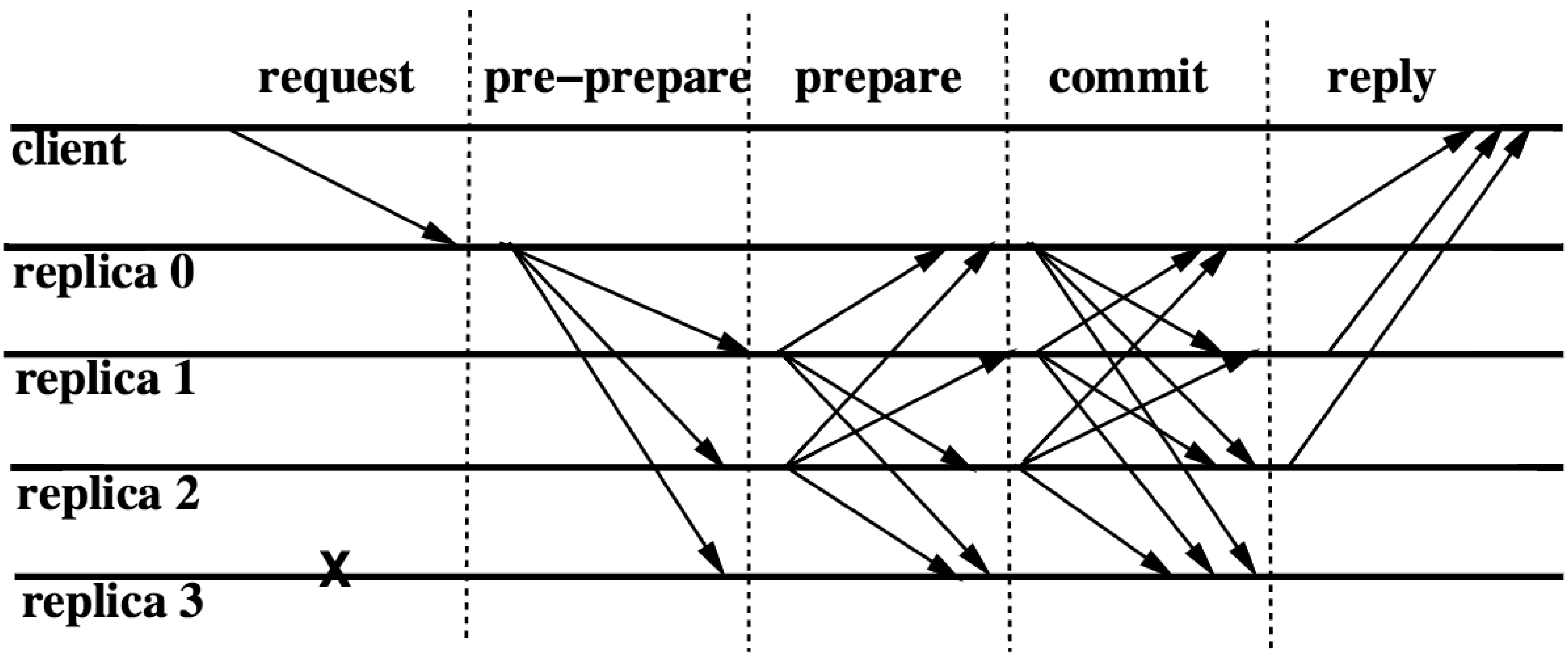


Why PBFT Doesn't Readily Scale

Three phase: pre-prepare, prepare, commit

In prepare & commit, leader must get at least two-thirds of all participants to "sign-off"

- Nodes sign-off via broadcast: $O(N^2)$



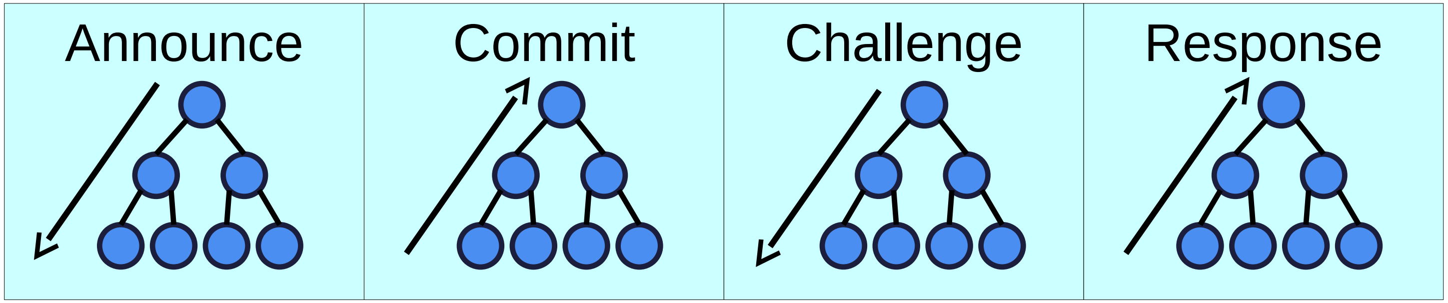
PBFT with Collective Signing (CoSi)

Builds on CoSi, presented in [IEEE S&P '16]

ByzCoin runs **collective signing** (CoSi) rounds to implement PBFT prepare, commit phases

- Efficient tree-structured communication
- Sign-offs compressed into 1 signature

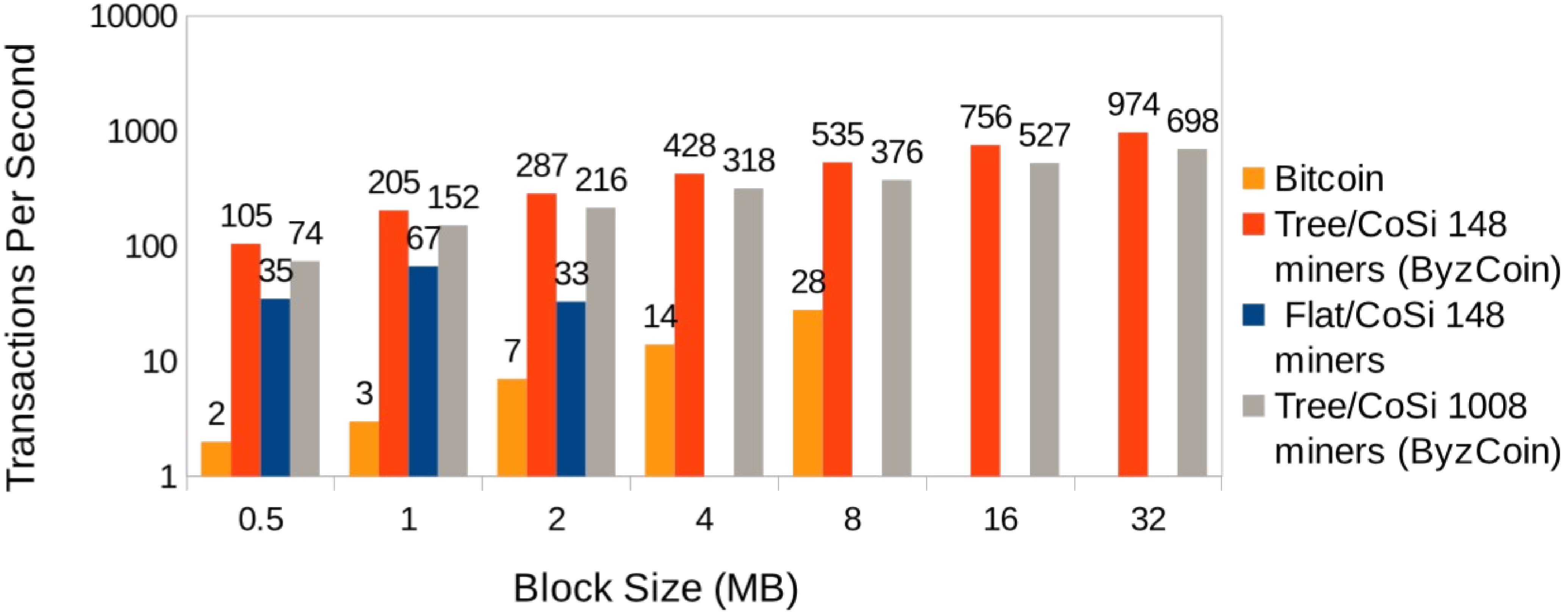
Reduce round cost from $O(N^2)$ to ~O(N)



ByzCoin transaction throughput

~100x improvement for similar block size

- higher throughput than PayPal
- scales to >1000 consensus peers



Excess redundancy in blockchains

Miners redundantly replicate *all* consensus effort in today's open blockchains like Bitcoin, Ethereum

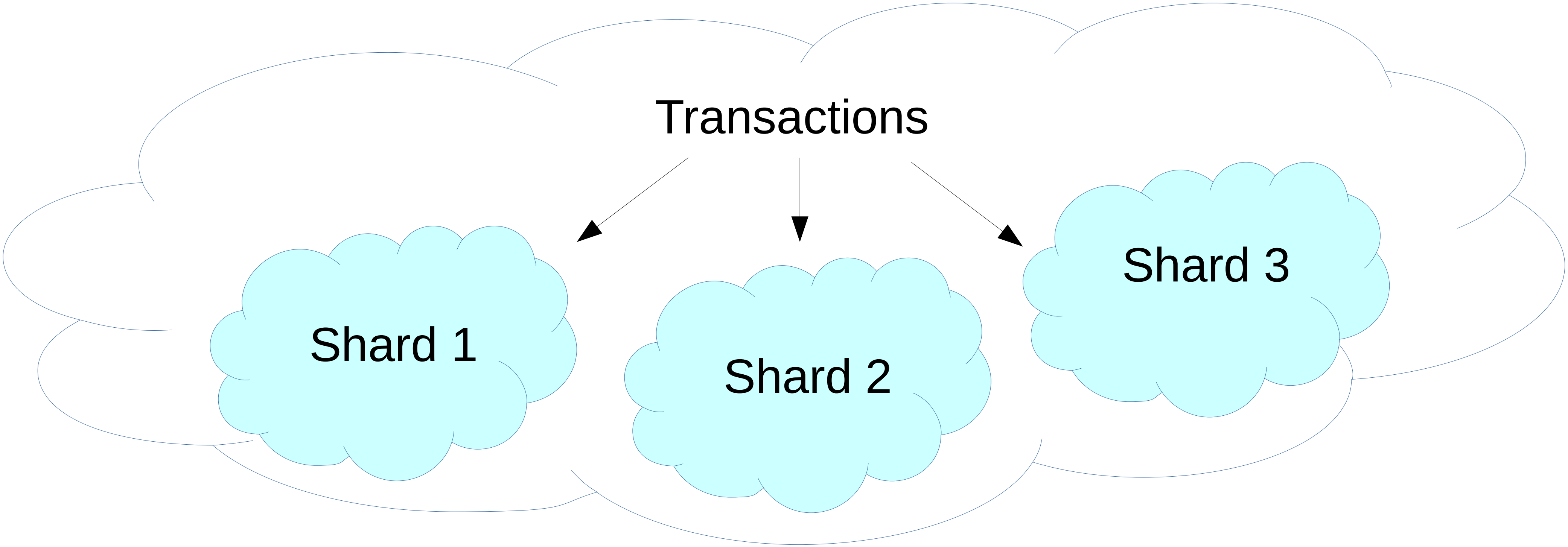
- **Storage:** each stores a *complete* copy forever
- **Processing:** each re-executes all contracts
- **Cost:** transaction fees pay for *everyone's* work
  - So Bitcoin/Ethereum transactions are expensive

Capacity doesn't "scale out" as participation grows

Horizontal Scaling via Sharding

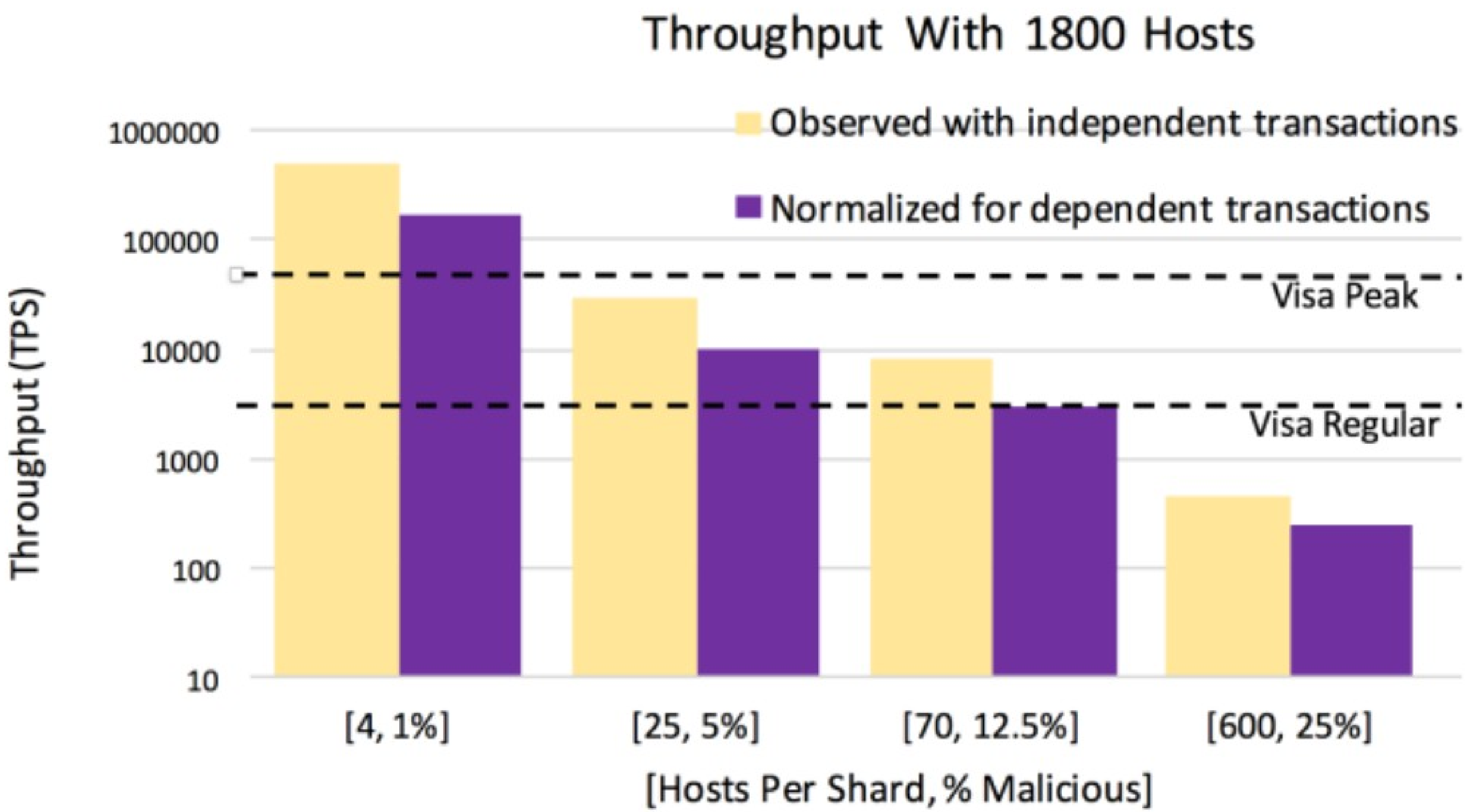
OmniLedger: A Secure Scale-Out Ledger [S&P 18]

- Break large collective into smaller random subgroups
- Builds on scalable bias-resistant randomness protocol (IEEE S&P 2017)
- Commit transactions cross-shard w/ 2-phase protocol



OmniLedger Throughput

Wide range of performance/security settings

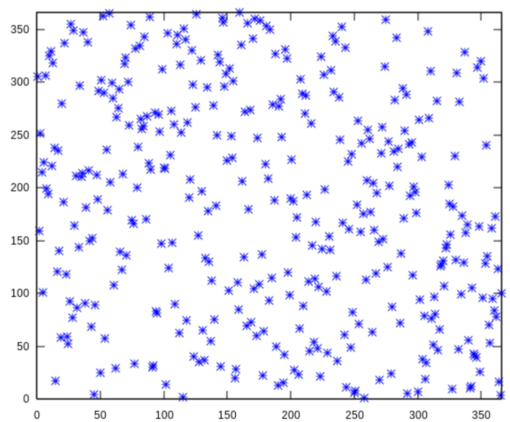


Two interesting sub-problems

- How to get **secure public randomness**?
  - For sharding or many consensus algorithms
- How to **follow a blockchain** efficiently?
  - Without requiring active gossip, even offline

Subproblem: public randomness

Vietnam War Lotteries (1969)

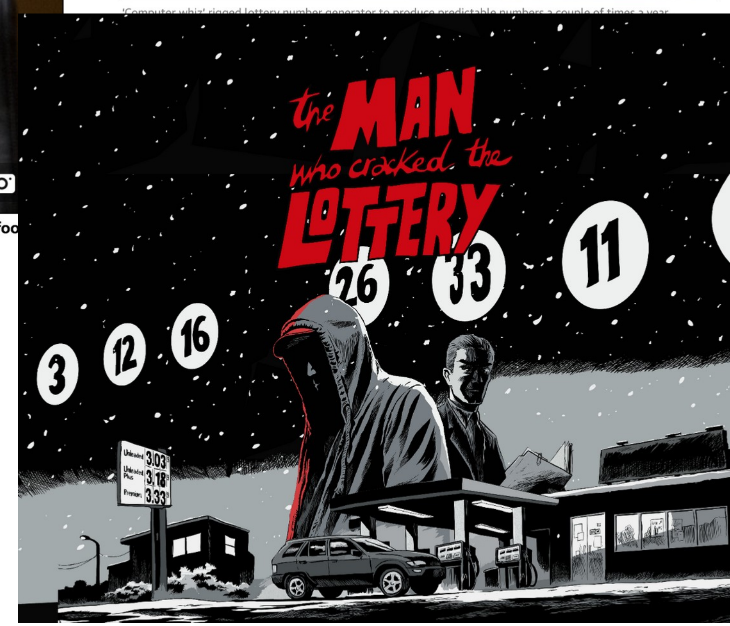


'European draws have been rigged': Ex-FIFA president Sepp Blatter claims to have seen hot and cold balls used to aid cheats

Former FIFA president Sepp Blatter said he had witnessed rigged draws for European foo competitions

Man hacked random-number generator to rig lotteries, investigators say

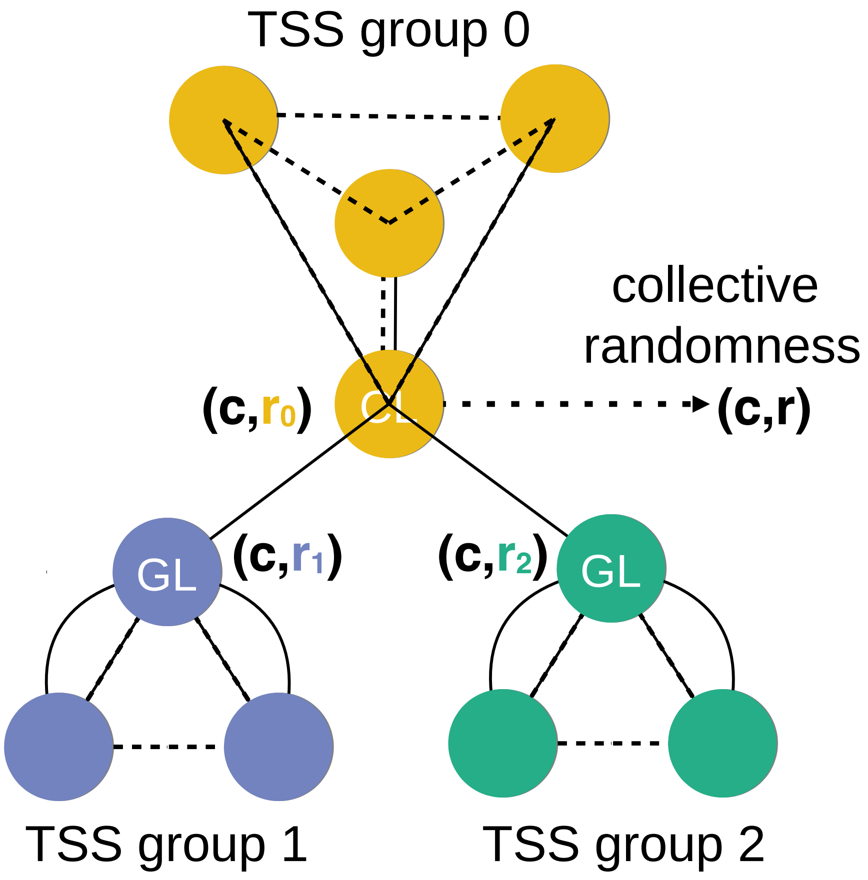
New evidence shows lottery machines were rigged to produce predictable jackpot numbers on specific days of the year netting millions in winnings



RandHound/RandHerd

“Scalable Bias-Resistant Distributed Randomness” [IEEE Security & Privacy ‘17]

- Standard t-of-n threshold model
- Efficient, scales to thousands of parties
- Compatible with ByzCoin, OmniLedger blockchains



The League of Entropy

Public randomness beacon based on RandHerd

- Launched by EFPL-DEDIS, Cloudflare, Kudelski, University of Chile, Protocol Labs
- Simplifications, BLS instead of Schnorr signing

Subproblem: following a ledger

How does a (lightweight) client securely know what has (or hasn’t) been committed to ledger?

- Contract/payment status, certificate validity, …
- PBFT: ask a 2/3 quorum of consensus nodes
- PoW: actively gossip at least block headers
- Bandwidth, latency, power, and safety costs

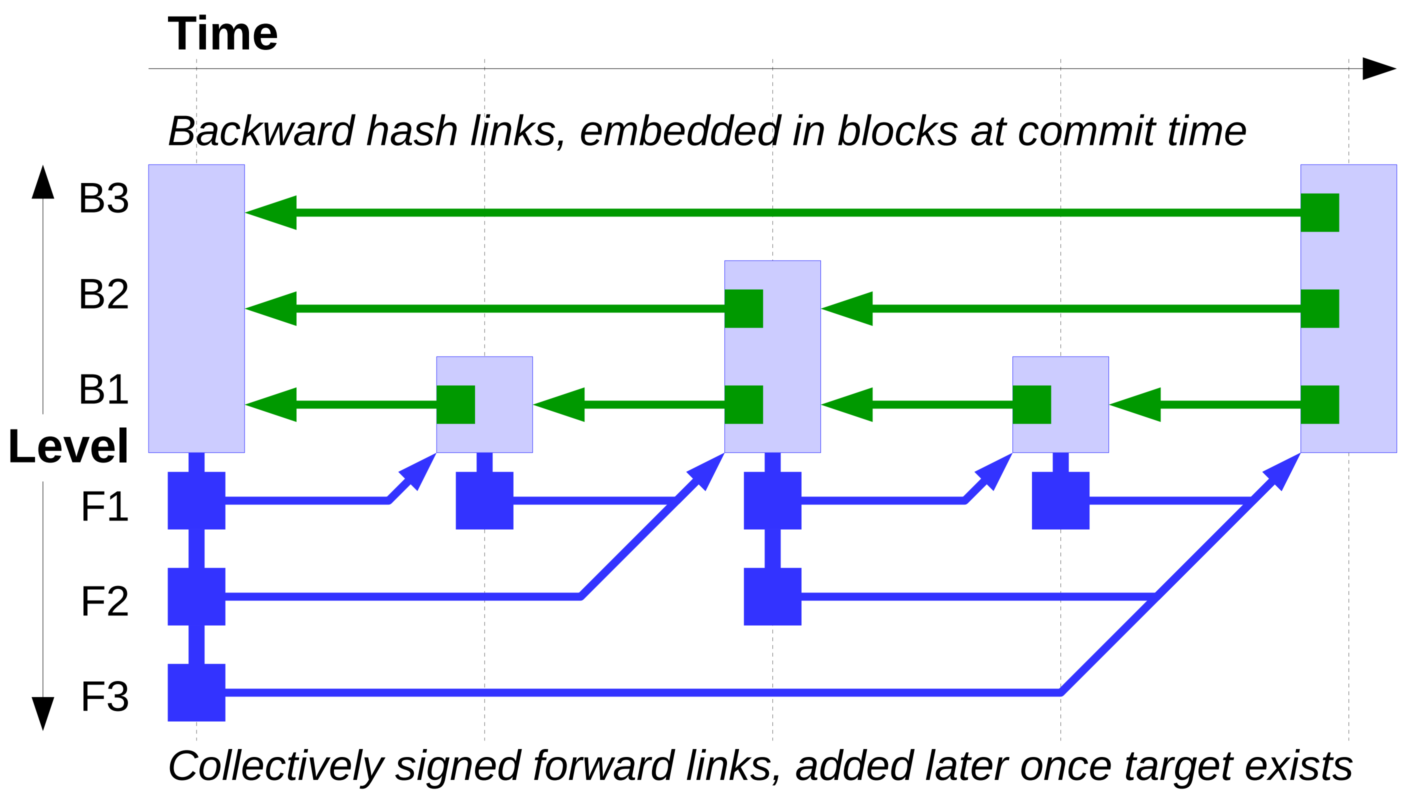
Can we follow a ledger *without* communication?

Secure offline blockchain verification

Collectively-signed SkipChains [CHAINIAC]

- Efficiently-verifiable cryptographic traversal both forwards and backwards in time

Disconnected verification of software updates, credentials, certificates, …



Talk Outline

Some blockchain consensus challenges and work

- Classic (permissioned) versus permissionless
- Latency and throughput scalability
- **Practical asynchronous consensus**
- Participation basis: investment or personhood?
- Smart contract execution, programming model

Resilience to Adversarial Networks

Most practical consensus today is *leader-based*

- Relies on synchrony assumptions and timeouts
- Paxos, Raft, PBFT, HotStuff, …

But a leader can be slow or a DoS attack target

- Slow everything to just below timeout threshold
- DoS attacks focused on current leader:

Resilience to performance attacks is hard!

Asynchronous Consensus

Wouldn’t it be nice if consensus

- Always proceeded as quickly as network conditions permit, however fast that is?
- Was (provably) immune to any slowdown of any (arbitrary) minority of participants?

That’s what asynchronous consensus achieves…

- *(in principle)*

Practical asynchronous consensus?

Most asynchronous consensus protocols are complex, slow, many layers, rarely implemented

- Often build multi-valued Byzantine consensus atop *n* instances of binary Byzantine consensus
- Examples: CKPS ‘01, HoneyBadgerBFT

Secure Causal Atomic Broadcast	
Atomic Broadcast	
Multi-valued Byzantine Agreement	
Broadcast Primitives	Byzantine Agreement

What is *time*, or a *clock*, anyway?

Tell the time

Wake you up

Clocks in distributed systems

Real-time systems define fixed event schedules based on real (wall-clock) time and deadlines.

Clocks in distributed systems

General-purpose distributed systems, however, we often prefer to be **self-timed**.

- Should progress *as quickly as conditions permit*
- Typically, *as network packet delivery permits*

We typically have some control over the **nodes** but not over the **network**.

Logical clocks in self-timed systems

Represent logical, not wall-clock, notions of time

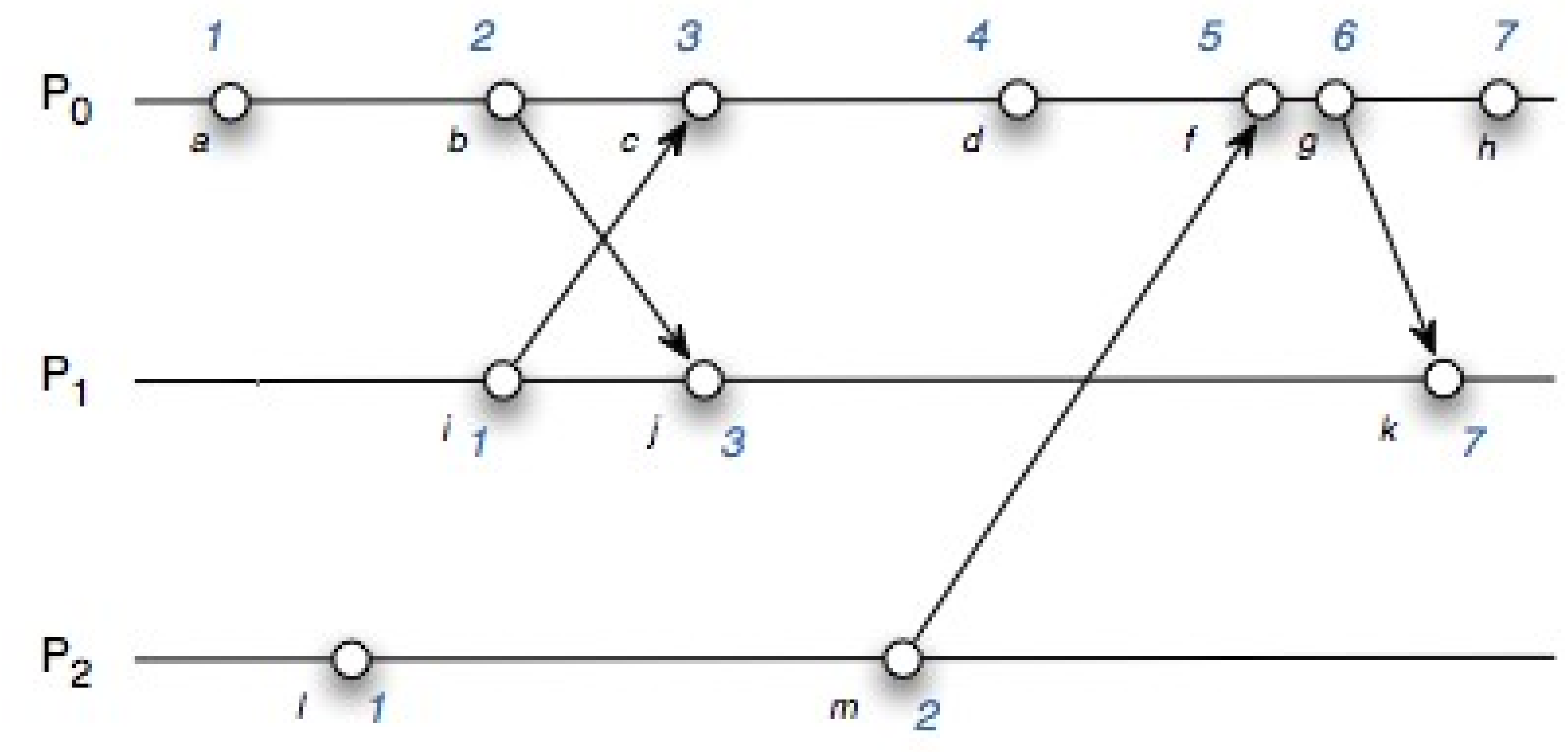
- How many logical **events** have passed?
- Under what **conditions** should the next start?

Examples:

- Lamport clocks, vector clocks, matrix clocks
- Van Jacobsen congestion control for TCP

Lamport clocks only “tell time”

Global event counters approximate causal history



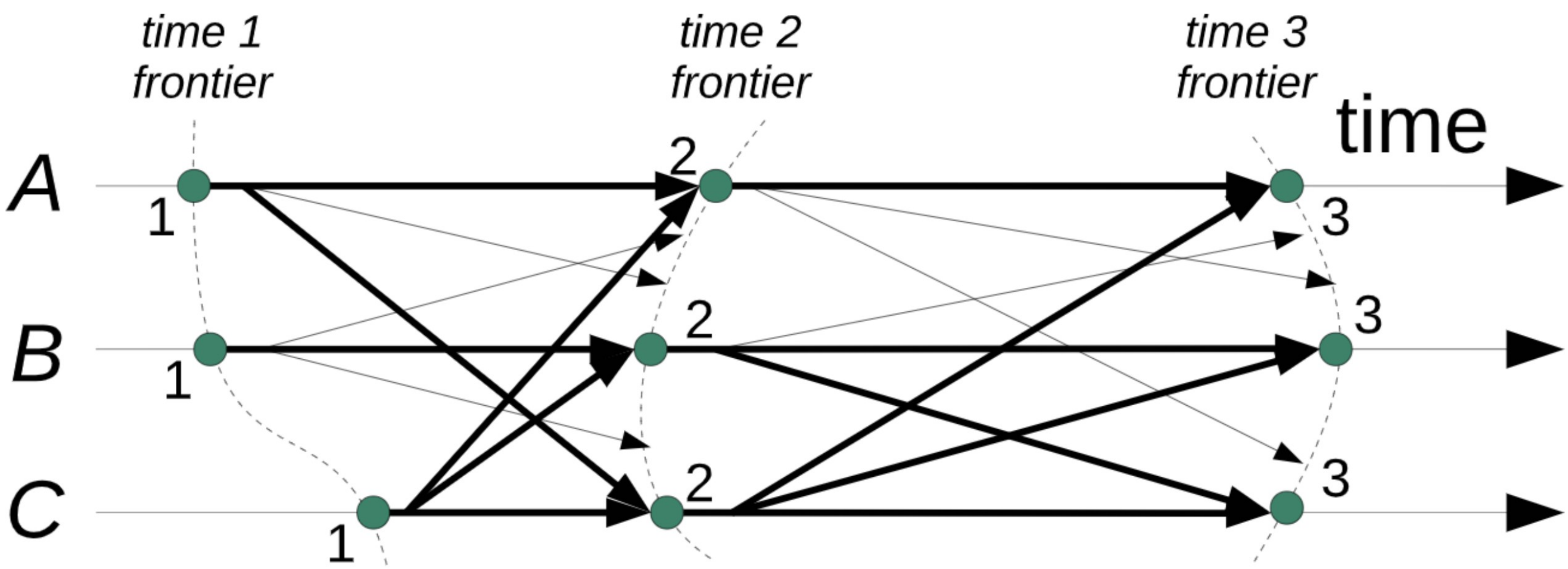
[credit: Paul Krzyzanowski, Rutgers]

Threshold Logical Clocks (TLC)

Like Lamport clocks, global integer metric of time

Unlike Lamport clocks, also offer *pacing* or “alarm”

Simulate *lock-step synchrony* atop async network



Que Sera Consensus (QSC)

Goal: make asynchronous consensus practical

- Not too complex, not too much overhead

Key idea: decompose *safety* & *liveness* problems

- **Consensus** layer: ensures safety (consistency), atop a simple *synchronous* network abstraction
- **Clocking** layer: ensures liveness (progress) through *threshold asynchronous coordination*

Consensus Layer (QSC)
Clocking Layer (TLC)

A Lock-Step Network Abstraction

QSC assumes a **syncast** network primitive:

(*received*, *delivered*) ← **syncast**(*message*)

Each **syncast** operation:

- Takes exactly one *logical time-step* (s=1, 2, 3…)
- Tries to send *message* to other group members
- *received:* some subset of messages sent in step
- *delivered:* some subset *all* members received

$delivered_i \subseteq received_j$, and $|delivered_i| >= threshold$

QSC Algorithm Summary

$H_0 \leftarrow$ genesis block

for time-step s = 1, 2, 3, … **do**:

- $P_s \leftarrow$ (proposed_block(), random_int(), hash(H_{s-1}))
- (E, D) $\leftarrow$ **syncast**(P_s)
- (C, U) $\leftarrow$ **syncast**(any best proposal in set D)
- $H_s \leftarrow$ any best proposal in set C
- If H_s is in U and is *uniquely* best in E, **commit**

That's it!

How QSC works, in brief

Each node has a tentative chain head (like BitCoin)

- Each time-step, add 1 block with random priority
- Call **syncast** twice to produce 3 proposal sets:
  - **Existent (*E*)**: proposed chains known to *exist*
  - **Common (*C*)**: chains that *all* nodes know to exist
  - **Universal (*U*)**: chains *all* nodes know are common
- Choose *any* highest-priority common (**C**) chain H_s to build on in next time-step s+1
- Commit when *all* nodes can *only* choose H_s

Byzantine QSC

To tolerate Byzantine nodes, must ensure:

- Hide honest nodes' priorities until end of round
  - Achievable with Shamir secret sharing
- All nodes must choose random priorities fairly
  - Enforceable via JVSS or VRFs

Result: at least 1/3 chance of commit each round

- Even with adversarial message scheduling

Consensus Layer (QSC)

Clocking Layer (TLC)

Implementing **syncast** abstraction

Simple *scatter/gather* approach with threshold *t*:

1. **Scatter**: distribute at least *t* nodes' messages to at least *t* nodes each
2. **Gather**: collect fully-scattered messages from at least *t* nodes

All fully-scattered messages reach *all* nodes if/when they successfully complete the time-step

- Provided *t* ensures quorum overlap property

Implementing **syncast** abstraction

procedure **syncast**(*m*):

1. Send [**echo**, *s*, *m*] by signed echo broadcast
   - Receivers sign & record *m* in their acked (*A*) set
2. *sigs* ← wait for threshold *t* of signatures on *m*
3. send [**done,** *s*, *m*, *sigs*] by normal broadcast
4. *D* ← wait for first *t* [**done**, *s*, *m*, *sigs*] messages
5. *R* ← union of *t* nodes' acked (*A*) sets
6. Return (*R*, *D*)

For more detailed information

Older (slightly different) formulations:

- Threshold Logical Clocks for Asynchronous Distributed Coordination and Consensus
  - https://arxiv.org/abs/1907.07010
- Que Sera Consensus: Simple Asynchronous Agreement with Private Coins and Threshold Logical Clocks
  - https://arxiv.org/abs/2003.02291

Talk Outline

Some blockchain consensus challenges and work

- Classic (permissioned) versus permissionless
- Latency and throughput scalability
- Practical asynchronous consensus
- **Participation basis: investment or personhood?**
- Smart contract execution, programming model

Membership, Stake, and Influence

Any human organization need a way to decide:

- Who holds a *stake* in decision-making
- How much *influence* each stakeholder wields
- How decisions are a actually agreed on: *consensus*

Without stake & consensus, organizations fail

Alternative Foundations for Stake

Permissioned: prove you’re in a meatspace club

Proof-of-Work: prove you’re wasting energy

Proof-of-Stake: prove you’re already rich

Proof-of-Storage: prove you have a big disk

Proof-of-*: prove you have a lot of *’s

Proof-of-Personhood: prove you’re a real person

[credit: me.me]

Membership in Blockchain Systems

Any organization must have a way to define:

- Who are the **members** involved in decisions?
- How much **power** does each member wield?

Example: how does Bitcoin define membership?

- Permissionless: open to anyone, *in principle…*
- But only if you constantly expend useless effort *just to prove you did it*.
  - Much like a **hazing ritual** for fraternity membership!

Equity in decentralized systems

Today's open blockchains are *investment-based*

- Proof-of-work: prove you wasted lots of energy
- Proof-of-storage: prove you bought big disks
- Proof-of-stake: prove you bought existing coin

None satisfy *democratic* fairness or inclusiveness

- More money buys more votes in consensus
- Most people can't compete with big investment

Environmental Costs

Proof-of-work = "scorched-earth" blockchains

- Bitcoin makes BTC scarce by making miners prove they **wasted energy**
- Serves **no purpose** except to prove they did it

Alternative: Proof-of-Stake (PoS)

- **Proof-of-Stake:** assigns consensus shares in proportion to prior capital investment
  - ☺ Could address energy waste problem
  - ☹ **Many nontrivial design challenges**
- Securing proof-of-stake is a nontrivial, interesting, but mostly-solved problem
  - e.g., Orobouros, Algorand
  - Also implementable with CoSi + SkipChains + OmniLedger + RandHound

Key Challenges with Proof-of-Stake

Implementing proof-of-stake securely requires:

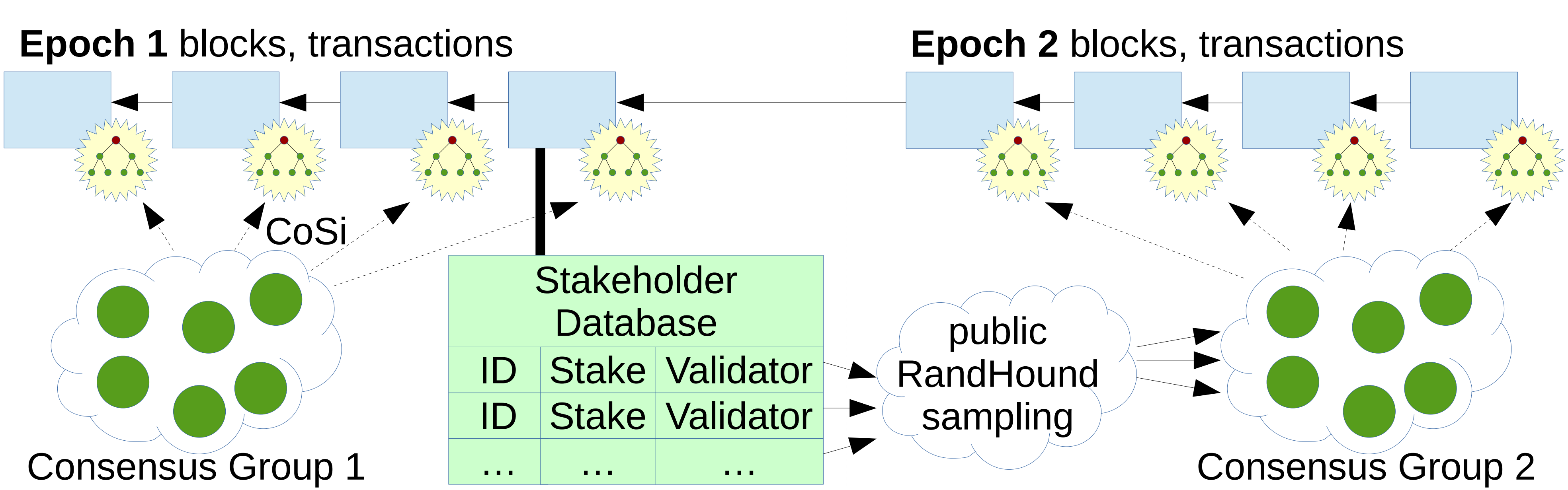
- **Agreement** on current set of stake-holders
  - e.g., list of public keys with number of “shares” each
- **Randomness** to sample future “minters” or consensus group members securely & fairly
- **Verifiability** of current state of the system
  - allow parties to distinguish the “one true blockchain” & avoid “nothing-at-stake” problem (chain mining)

All these tools are available as modules in ByzCoin, RandHerd, Chainiac, OmniLedger

Modular Proof-of-Stake

Assume we have a ByzCoin-like consensus group

- Use PBFT to agree on transactions and stake
  - List of stakeholders, # shares each, their validators
- After epoch, RandHound-sample next group
  - Old group collectively signs new, forms **SkipChain**



Is Proof-of-Stake What We Want?

A Proof-of-Stake cryptocurrency is essentially an automated analog of a **shareholder corporation.**

- May help hasten the takeover of automation, but won’t fix the world.

It’s all just “Proof-of-Investment”

Proof-of-Work, Proof-of-Stake, Proof-of-* are all **Proof-of-Investment**, aka investment capitalism.

- The more * you invest, the greater your reward.

All prone to re-centralization, aka **rich get richer**

- Larger stakeholders always in a better position to *exploit economies of scale* – or just *cheat* – to further increase their percentage of the pie.

Proof-of-stake *won’t keep systems decentralized!*

- At best they can *reduce rate of recentralization*

Long-Term Decentralization?

Can we build decentralized systems that will reliably *stay decentralized* over the long haul?

- **Inclusive:** allow “permissionless” participation by everyone *in practice*, not just in theory
  - Including developing world, homeless, refugees
- **Sustainable:** Ensure future generations will have the same opportunities that we do today
  - Regardless whether their grandparents were lucky
- **Empowering:** Provide opportunities for all while limiting vulnerability to abuse of power

Toward People-Centric Blockchains

Can we build decentralized technology that will

- Securely stay *open* and *widely decentralized*?
- Offer a fairness metric *meaningful to people*?
- Be accountable to *users* rather than *wealth?*

"We must act to ensure that technology is designed and developed to serve humankind, and not the other way around"
- Tim Cook, Oct 24, 2018

Person-Centric Decentralization

Proof-of-Personhood [IEEE S&B '17]

- Proof-of-Stake but *one stake unit per person*

Some Proof-of-Personhood Projects

Can we achieve “one person, one vote” online?

- Pseudonym Parties [Ford, 2008]
- Proof-of-Personhood [Borge et al, 2017]
- Encointer [Brenzikofer, 2018]
- BrightID [Sanders, 2018]
- Duniter [2018]
- Idena [2019]
- HumanityDAO [Rich, 2019]
- Pseudonym Pairs [Nygren, 2019]

Proof-of-Personhood: Approaches

- Legacy Identities (e.g., government-issued)
  - Require costly ID-checking, not that hard to fake
- Global Biometric Databases (India, UNHCR)
  - Huge privacy issues, false positives+negatives
- Trust Networks (PGP “Web of Trust” model)
  - Unusable in practice, doesn’t address Sybil attacks
- Pseudonym Parties [SocialNets ‘08]
  - Requires *in-person* participation, physical security
  - Low-cost: verifies *only* personhood, not ID or trust

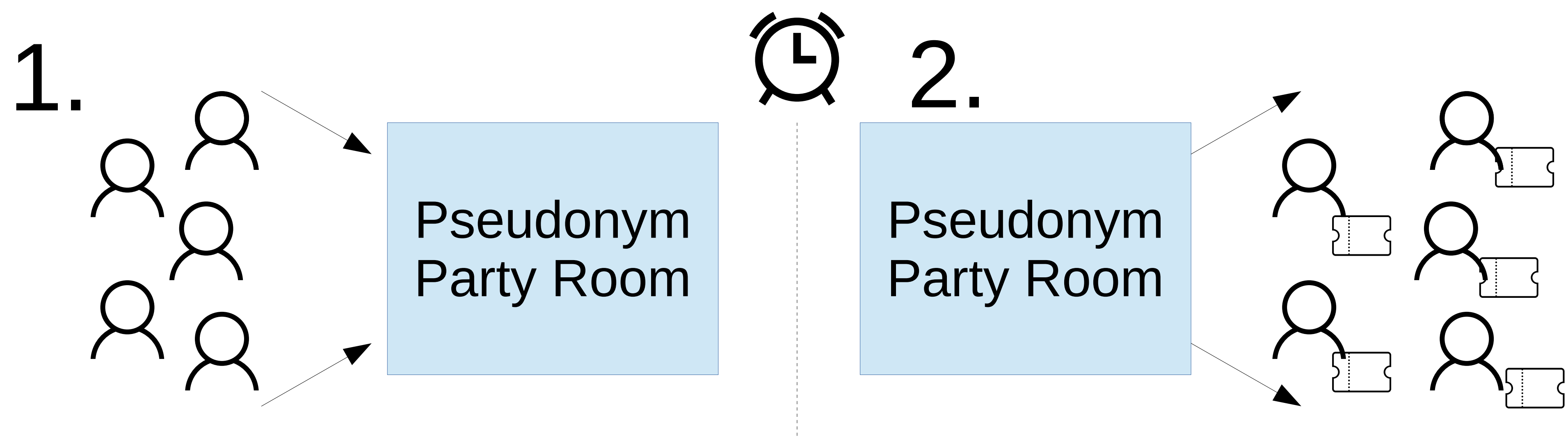
Pseudonym Parties: Summary

Locally-organized regular **physical meetings**

- Anyone can *enter* a space until a set deadline
- Then can only *exit*, each getting one credential

No need for IDs, biometrics, PGP key-signing, etc

- Just bodies: can be in only one place at a time



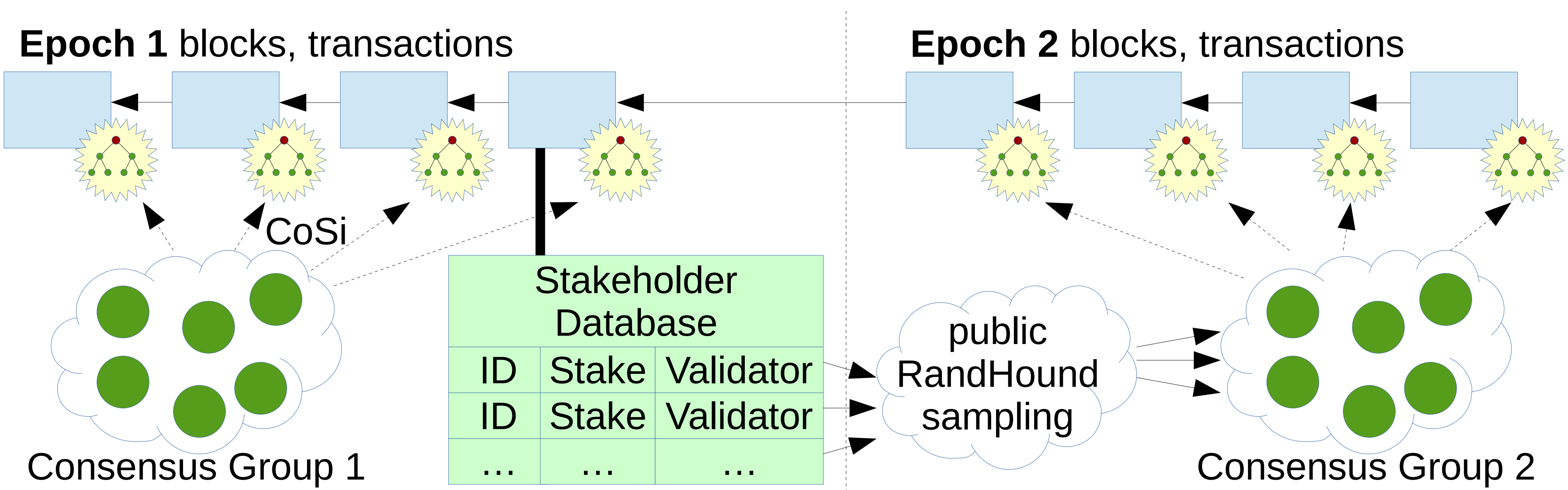
Proof-of-Personhood Consensus

Similar to Proof-of-Stake in technical challenges

- Many similar solutions apply in principle

Modular Proof-of-Personhood consensus

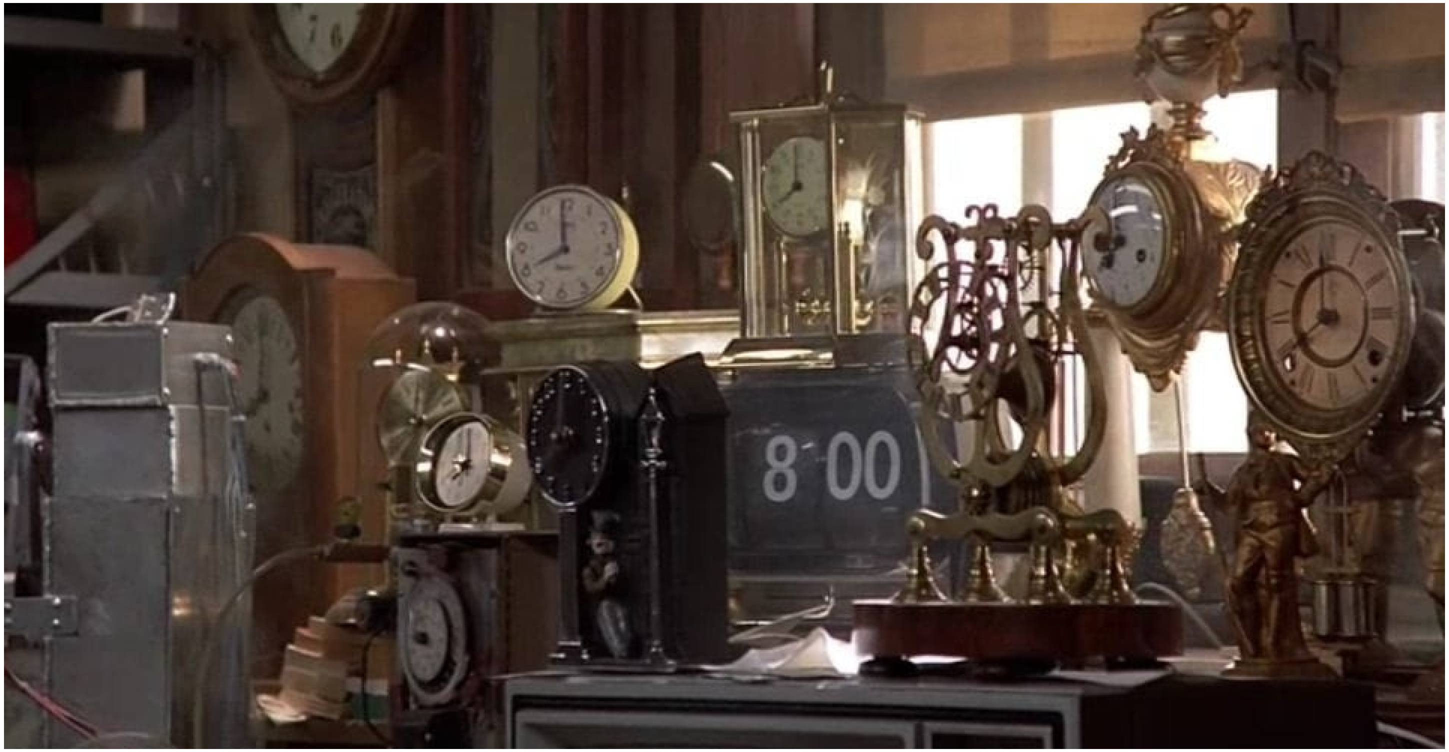
- PoP parties publish PoP token list each epoch
- Holders define servers for sampling committees



Regular Synchronized Events

Federation of PoP groups might hold *concurrent* events with *simultaneous* arrival deadlines

- No one can physically attend two at once



Proof-of-Personhood: Applications

A few promising applications:

- Democratic decentralized governance
- Cryptocurrency universal basic income (UBI)
- Replacement for CAPTCHAs
- Sockpuppet-resistant crowdsourcing
- Accountable anonymity & pseudonymity
- Decentralized single sign-on as “a person”

A Crypto Universal Basic Income?

Available on “opt-in” basis to *everyone*,
not just in particular jurisdictions

Talk Outline

Some blockchain consensus challenges and work

- Classic (permissioned) versus permissionless
- Latency and throughput scalability
- Practical asynchronous consensus
- Participation basis: investment or personhood?
- **Smart contract execution, programming model**

Consensus for Smart Contracts

Smart contract systems need consensus to agree on *what was computed* by an executed contract

- Execution typically must be deterministic
  - Disagreement in execution → consensus failures
- Deterministic VMs usually constrained, slow
  - Ethereum VM (EVM): complex user-defined computation, e.g., cryptography, mostly impractical
  - Bad solution: add special-purpose crypto opcodes to optimize common cases, one hard fork at a time

Can we have a *deterministic* VM that's also *fast*?

A few options

Exploration & development work in progress:

- High-level: deterministic language sandbox
  - e.g., early prototype restriction of Go language
- Mid-level: leverage a mature bytecode or IR
  - e.g., restriction of Java bytecode or LLVM IR
- Low-level: build on a “flat-model” architecture
  - e.g., x86, ARM, or WASM instruction set

Interesting tradeoffs & challenges in each: e.g., FP

Conclusion

Some blockchain consensus challenges and work

- Classic (permissioned) versus permissionless
- Latency and throughput scalability
- Practical asynchronous consensus
- Participation basis: investment or personhood?
- Smart contract execution, programming model