Marko Vukolić, IBM Research - Zurich

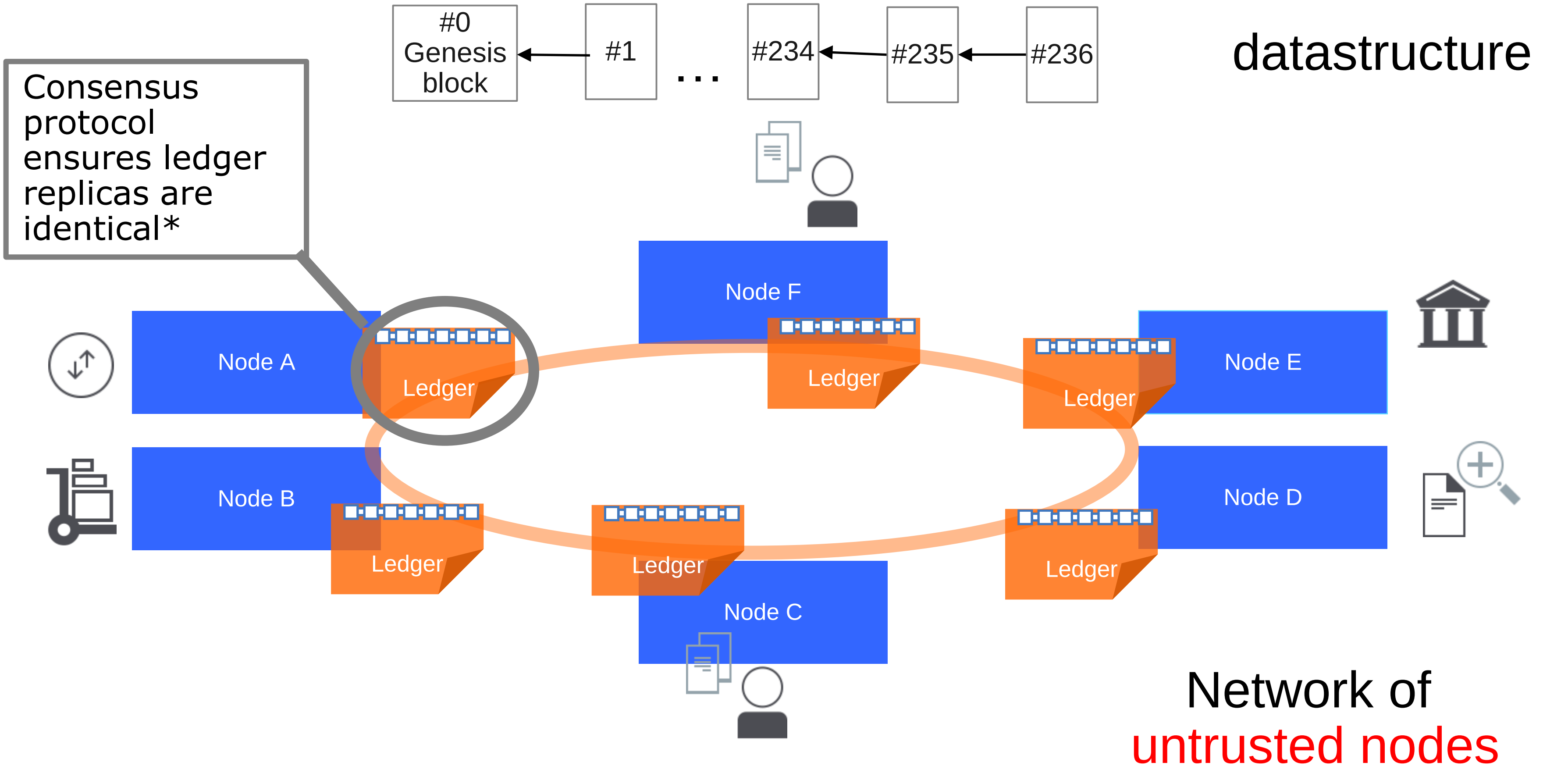
June 20, 2018

Blockchain Scalability

CDMDSW
Decision Making
& Data Science

What is a Blockchain?

- **A chain (sequence, typically a hash chain) of <u>blocks</u> of transactions**
  - Each block consists of a number of (ordered) transactions
  - Blockchain establishes total order of transactions



Blockchain evolution (2009-present)

2009
Bitcoin

- **A hard-coded cryptocurrency application w. limited stack-based scripting language**
- **Proof-of-work-consensus**
- **Native cryptocurrency (BTC)**
- **Permissionless blockchain system**

Blockchain 1.0

2014
Ethereum

- **Distributed applications (smart contracts) in a domain-specific language (Solidity)**
- Proof-of-work-consensus (transition to Proof of Stake?)
- Native cryptocurrency (ETH)
- Permissionless blockchain system

Blockchain 2.0

2017
Hyperledger
Fabric

- **Distributed applications (chaincodes) in different general-purpose languages (e.g., golang, Java, Node)**
- **Modular/pluggable consensus**
- **No native cryptocurrency**
- **Multiple instances/deployments**
- Permissioned blockchain system

Blockchain 3.0

Main scalability bottlenecks

- **Consensus/transaction ordering performance**
- **Sequential smart contract execution**
- In the rest of the talk we will see challenges and how to address both

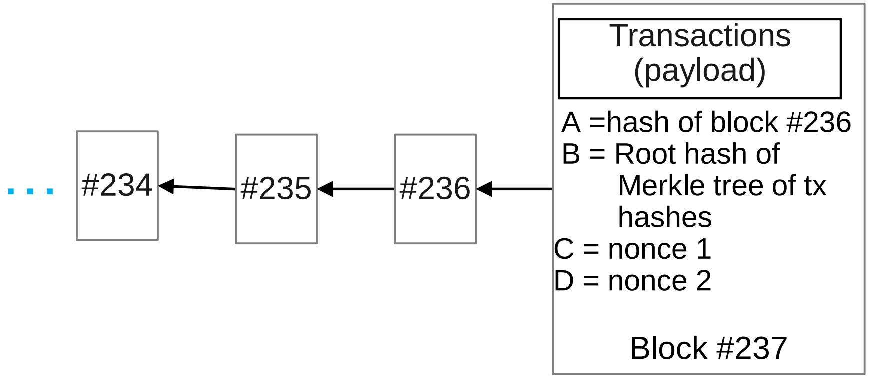
Growing the chain

- **How does the chain grow?**

Most popular blockchain technique (used also in Bitcoin):

Proof-of-Work (PoW)

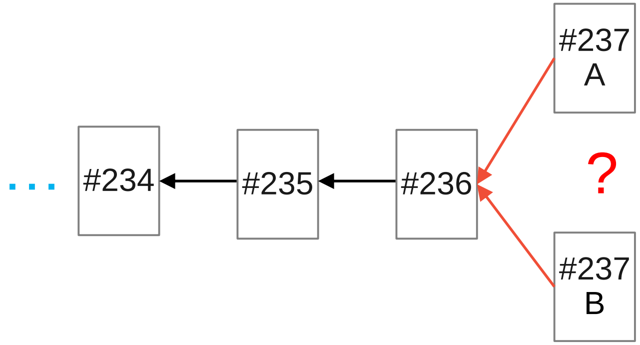
Growing Proof-of-Work (PoW)-based Blockchain



$$h = \text{hash of Block \#237} = \text{SHA256}(A||B||C||D)$$

- Block “mining”:
  - Every participant (“miner”) tries to find nonces
  - such that the hash of the block *h* **<u>is lower than a 256-bit *target*</u>**
- Bitcoin
  - Target dynamically adjusted every 2016 blocks
  - 1 block generated roughly every 10 minutes
  - This currently requires roughly 2^{80} expected hashes per block

Forks

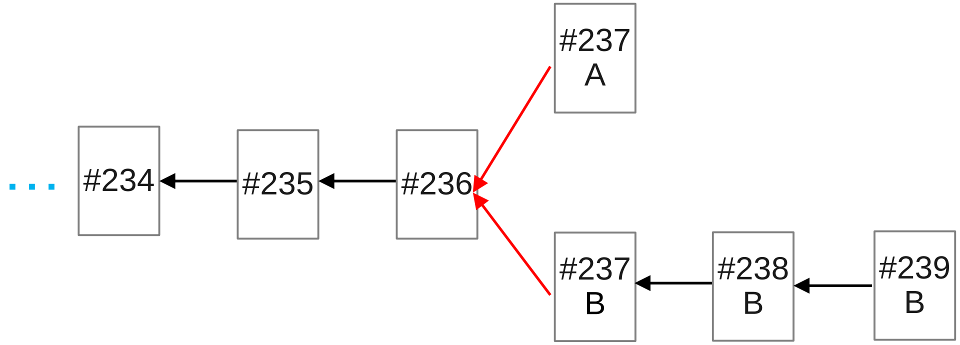


- If multiple miners mine the next block, consensus (on the next block) might be broken

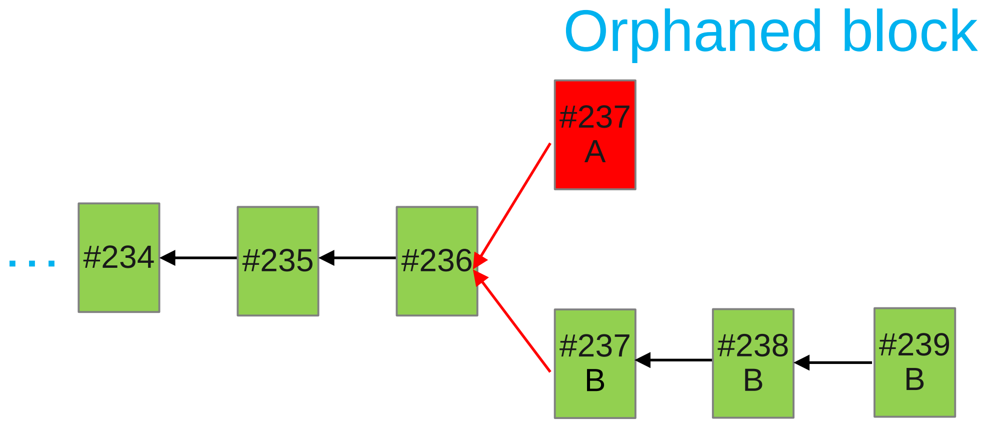
  PoW acts as an unreliable concurrency control mechanism – it may fail in this

- Hence, Bitcoin miners adopt a **conflict resolution policy**
  - They will temporarily store both 237A and 237B
  - A fork being extended longer (in fact with more work) eventually prevails

Example (longest/most difficult chain wins)



Example (longest/most difficult chain wins)



Implications and the performance issues

PoW way of extending the ledger heavily and negatively impacts system scalability and overall throughput

- Bitcoin: With 1 block every 10 minutes and fixed block size of 1 MB
  - Peak throughput: only 6-7 tx/sec
  - Latency (of 6 block confirmations): about 1h
  - Enormous energy consumption!

- https://digiconomist.net/bitcoin-energy-consumption
  - 71 TWh/year → 8GW of power
  - More than Switzerland, 0.32% of world electricity consumption
  - 987 kWh per transaction!
  - Average US household in 2016 → 897 kWh per month

Better performance by tuning PoW parameters?

- **<u>Limited benefits, potentially weaker security</u>**
  - shorter block generation times (increasing block frequency)?
  - larger blocks?
  - Different conflict resolution rules?

- From Gervais et al. CCS'16 paper https://eprint.iacr.org/2016/555

	Bitcoin	Litecoin	Dogecoin	Ethereum
Block interval	10 min	2.5 min	1 min	10-20 seconds
Public nodes	6000	800	600	4000 [11]
Mining pools	16	12	12	13
t_{MBP}	8.7 s [8]	1.02 s	0.85 s	0.5 - 0.75 s [12]
r_s	0.41%	0.273%	0.619%	6.8%
s_B	534.8KB	6.11KB	8KB	1.5KB

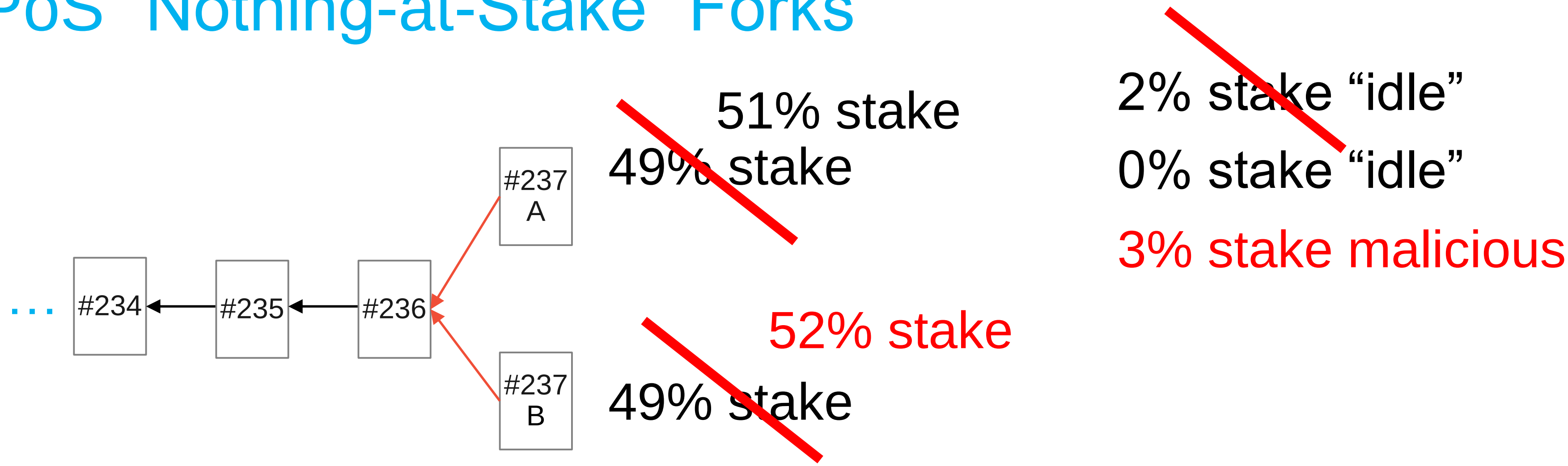
- Bitcoin 6 blocks (1hour) ~ Ethereum 37 blocks (9-10 minutes)

- PoW blockchains can attain up to 60 tps with Bitcoin-like probability of stale blocks

Boosting consensus: Enter Proof of Stake (PoS)

- PoS usually sits on top of PoW tree datastructure
- Allows nodes with more stake/weight to form blocks more often effectively lowering the difficulty
- “Nothing at stake” problem?
- Centralization?

PoS "Nothing-at-Stake" Forks



- PoS breaks ties selecting forks (branches) with more stake on them
- Very susceptible to "double-spend" attacks in absence of penalties
- Example with 3% of stake double spending

Casper – Friendly Finality Gadget

- Buterin/Griffith
  - https://arxiv.org/abs/1710.09437

- Leverages BFT techniques to limit the effects of forks and to address nothing at stake problem
  - Byzantine Fault Tolerant (BFT) agreement to settle on a single block and penalize equivocating nodes

- Relies on node synchrony as well

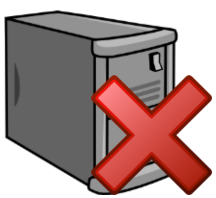
- Announced as early as in 2015, still being figured out…

Wait but what is this BFT?

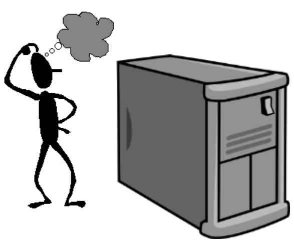
Enter State machine replication (SMR)

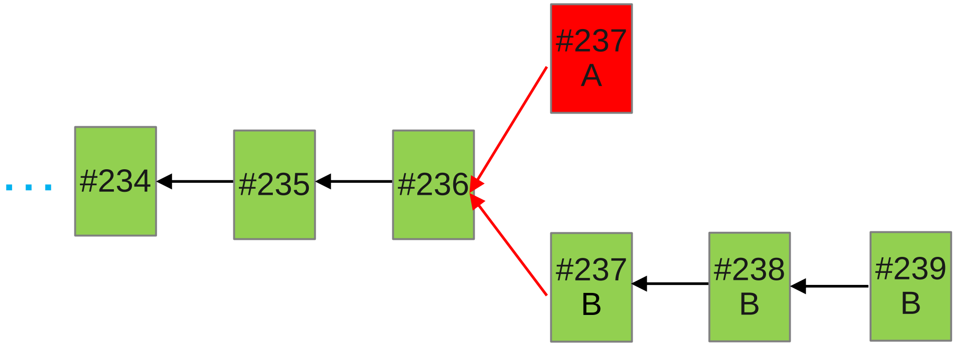
- [Lamport 78], countless follow-up papers
- Classical Distributed Computing problem
  - An illusion of a centralized system that never fails
  - Despite **machine** faults and (temporary) **network** partitions

<u>What machine faults?</u>



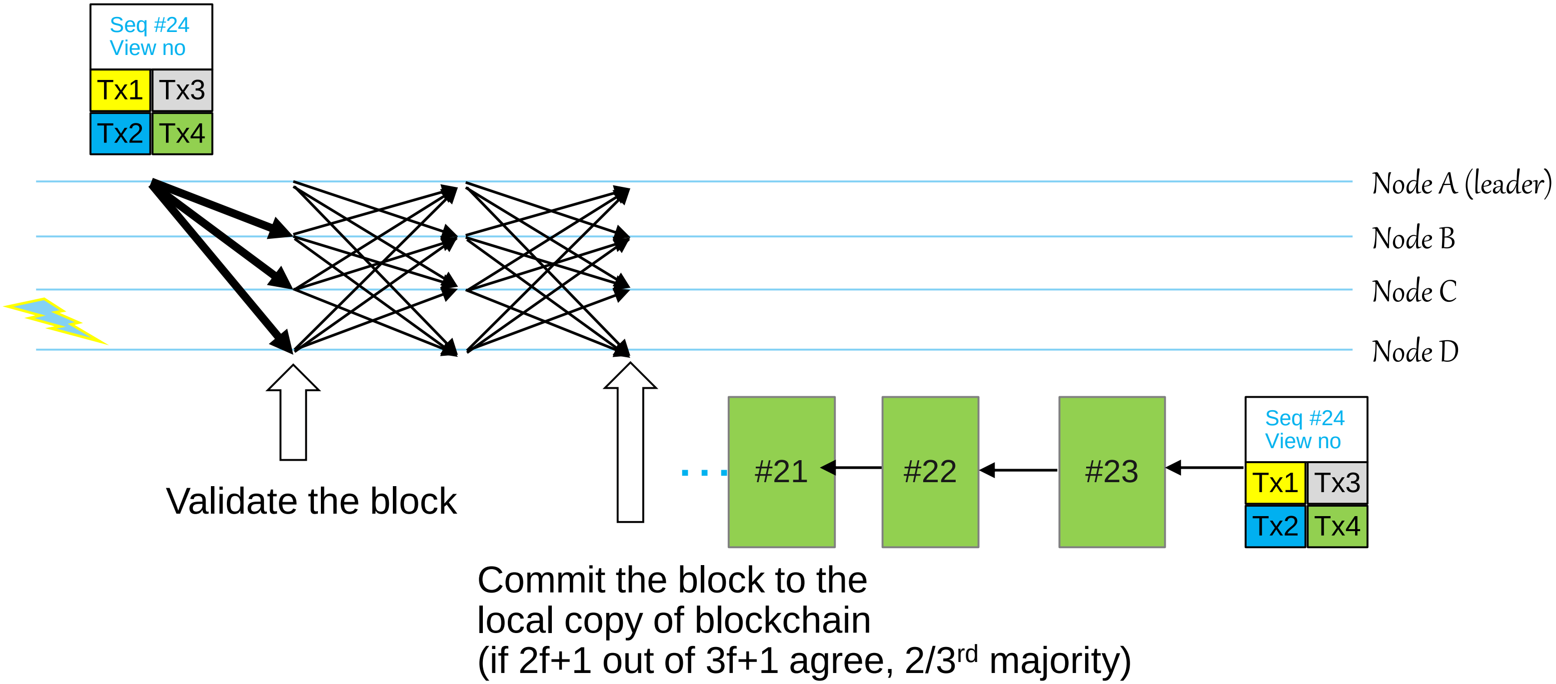
- Crash faults (CFT): A machine simply stops execution and halts
  - Paxos, RAFT, Zookeeper AB,…
- Non-crash (a.k.a. Byzantine) faults (BFT)
  - A model that blockchains adopt





No forks!

BFT Consensus (example of PBFT [TOCS2002], implemented in Hyperledger Fabric v0.6)



Many other things burden the implementation (it is not simple as it might look)

- Leader election
- State transfer (new, slow Party)
- Reconfiguration

In this example, all nodes have equal weights, the protocol can simply be adapted to weights/stakes

BFT in Blockchains

- BFT is known as a technology that matters for **permissioned** blockchains

- But with PoS BFT importance extends to **permissionless** blockchains as well

- Much better performance (throughput/latency) compared to PoW
  - Drawback: more intensive on network communication

PoW vs. BFT for Blockchain (simplified overview)

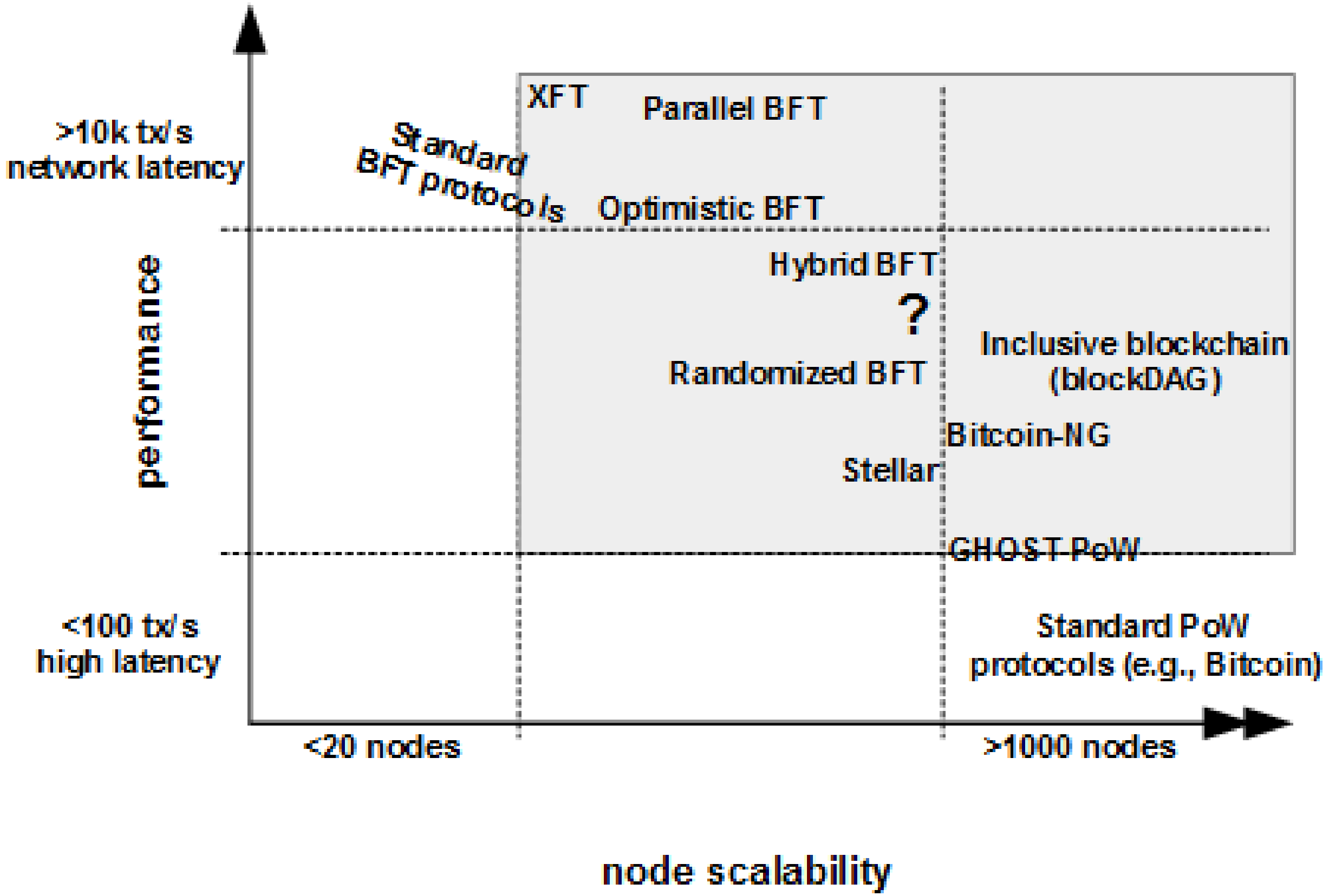
	Proof of Work (Bitcoin, Ethereum,...)	BFT state machine replication (Ripple, Stellar, Fabric,…)
Membership type	Permisionless	Permissioned
User IDs (Sybil attack)	Decentralized, Anonymous (Decentralized protection by PoW compute/hash power)	Centralized, all Nodes know all other Nodes (Centralized identity management or stake protect against Sybil attacks)
Scalability (no. of Nodes)	Excellent, >100k Nodes	~~Verified up to few tens (or so) Nodes~~ Can scale to 100 nodes with certain performance

<u>Open research problem:</u>

Given the use case, network, no. of nodes

What is the most suitable and scalable Blockchain technology/protocol?

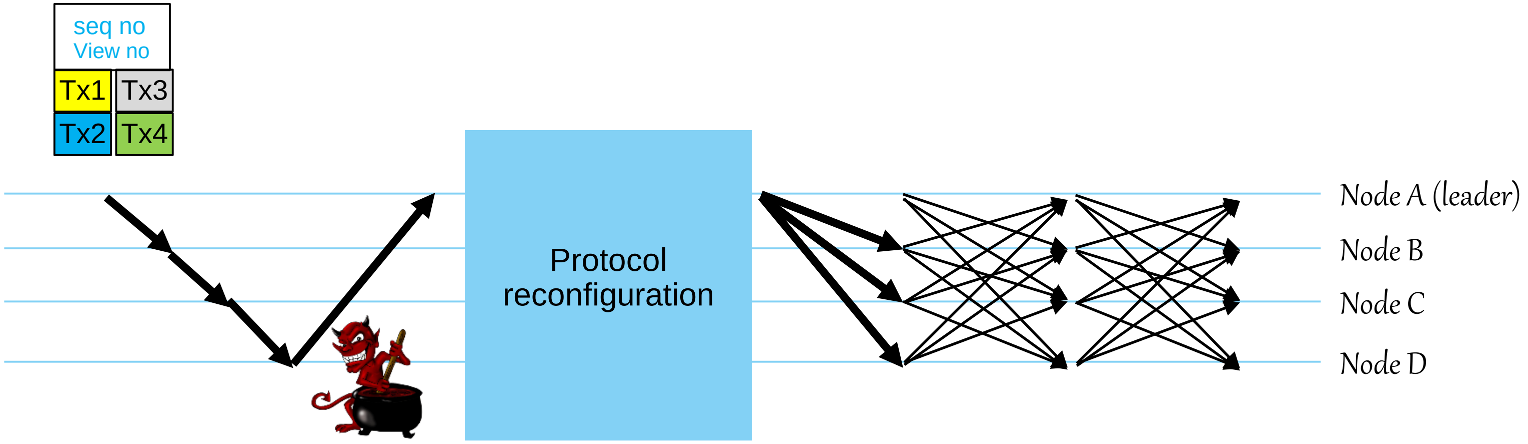
Peak Throughput	from 7 tx/sec (Bitcoin)	>10k tx/sec with existing implem. in software [<20…100 nodes]
Power efficiency	>8 GW (Bitcoin)	Good (commodity hardware)
Temporary forks in blockchain	Possible (leads to double-spending attacks)	Not possible



Marko Vukolić. *The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication*
Proceedings of the 2015 International workshop on open problems in network security (iNetSec 2015).

Optimistic protocols (a bit of our own work)

- Abortable state machine replication [Aublin et al, TOCS 2015]
  - Can run O(n) BFT protocol of a basically arbitrary communication pattern including (a very load-balanced one) in the optimistic case
  - Backed by any BFT protocol (e.g., PBFT) to cover the worst case without redesigning the entire protocol/system



Revisiting the assumptions (still our own work)

- XFT [Liu et al., OSDI 2016]   http://arxiv.org/abs/1502.05831
- BFT assumes powerful adversary
  - controlling the network among correct nodes
  - and f Byzantine nodes out of 3f+1 nodes
  - Simultaneous control over network and Byzantine nodes may be difficult to pull out in the blockchain setting
- XFT: at most f of partitioned nodes and Byzantine nodes at any time
  - Have the cost of CFT consensus without trusted hardware

SMR model	CFT	XFT NEW!	BFT
Number of Nodes	2f+1	2f+1	3f+1
Tolerating Byzantine Nodes	no	yes	yes
Performance	Good	Practically as good as CFT	Poor (compared to CFT)

- XPaxos as the primer for such XFT protocols

Putting SMR consensus in modern hardware

- Has been shown to dramatically increase performance of consensus
  - But so far with CFT only

- RDMA-based protocols
  - FaRM [Dragojevic et al, NSDI'14],
  - DARE [Poke/Hoeffler, HPDC'15]

- FPGAs [Istvan et al., NSDI'16]
  - 3 node Zookeper atomic broadcast (ZAB) up to 2.5 million tps*

  *CFT, excluding crypto overhead and application execution

Node scalability is still an unknown and BFT still to be implemented

Main scalability bottlenecks

- Consensus/transaction ordering performance
- **Sequential smart contract execution**

Introducing smart contracts/chaincode

Modern crypto ledgers (e.g., Ethereum, Hyperledger)

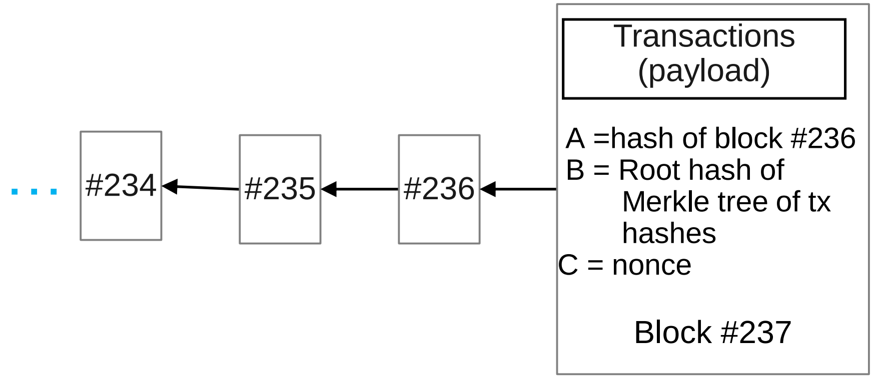
aim at supporting “smart contracts” or “chaincodes”

A smart contract is an event driven program, with state, which runs on a replicated, shared ledger and which can take custody over assets on that ledger. [Swanson2015]

“Smart contract” → (replicated) state machine

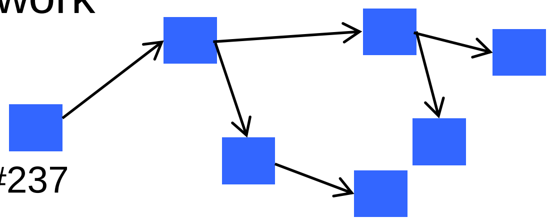
PoW with smart-contracts

- **PoW Consensus**
  - Block “mining”



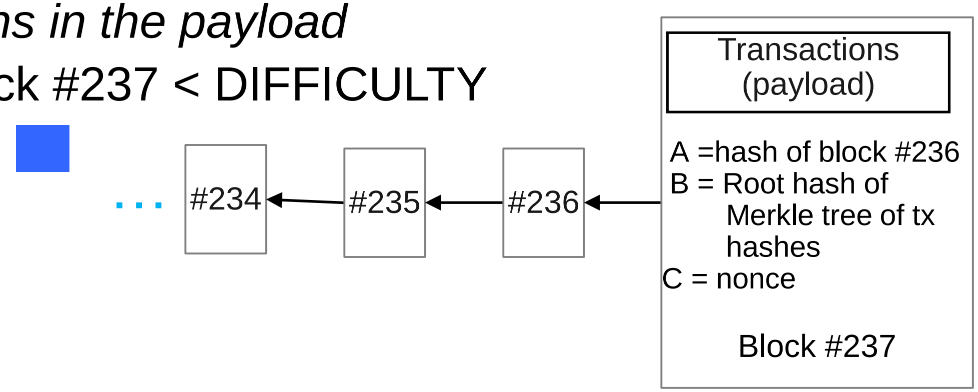
- *Pre-executing transactions in the payload*
- *Finding nonces such that*
  h = hash of Block #237 = SHA256(A||B||C) < DIFFICULTY

- Block #237 propagation to the network (gossip)

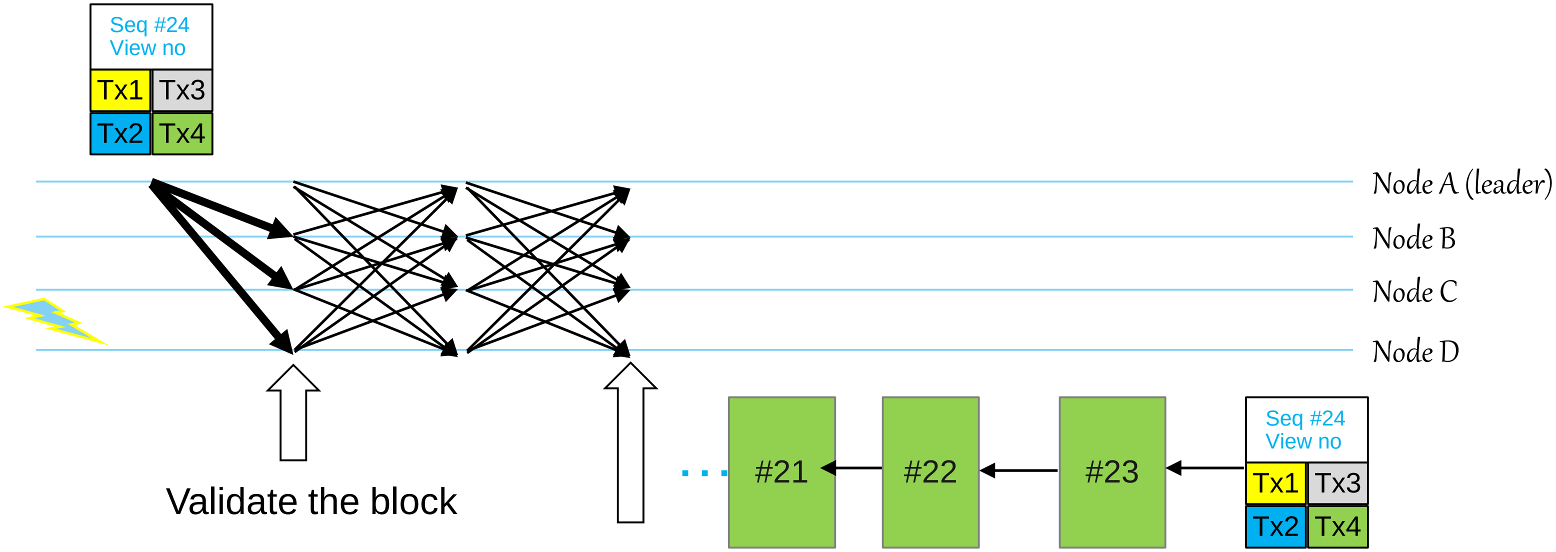


Miner of block #237

- **Block Validation / Smart Contract Execution (every miner)**
  - *Executing transactions in the payload*
  - *Verifying* hash of Block #237 < DIFFICULTY



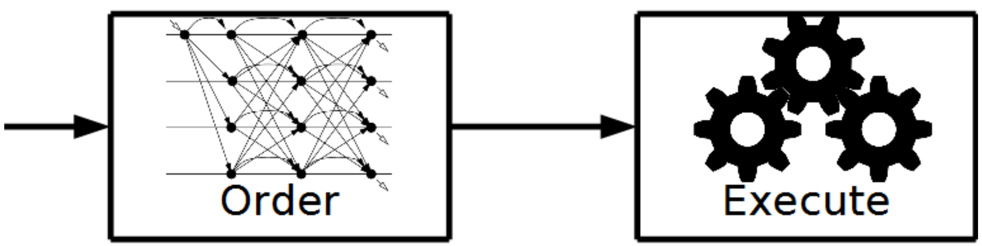
BFT Consensus (example of PBFT [TOCS2002], implemented in Hyperledger Fabric v0.6)



Commit the block to the local copy of blockchain (if 2f+1 out of 3f+1 agree, 2/3rd majority)

Execute transactions sequentially

Almost all blockchains follow order-execute architecture



- **Order** transactions using (PoW/BFT) consensus
- **Execute** transactions sequentially at each node (miner)

This approach works only when state-machine and transactions are deterministic

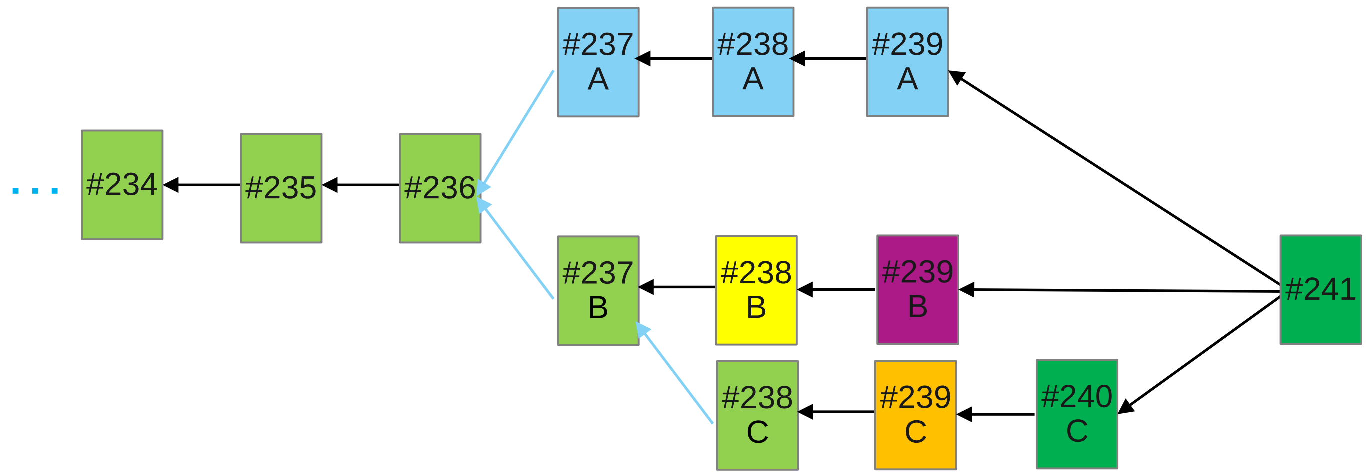
Order-Execute main design challenges

- **Enforcing determinism**
  - What happens if you code smart-contracts in general purpose programming language (Go, Java)?
    - Potential non-determinism!!!
  - Ethereum
    - Solidity domain specific language
    - Compiled to Ethereum VM stack-based bytecode

- **Infinite loop "application"? Long executing application?**
  - **Sequential execution**
  - **Gas, paying for every step of execution (computation)**
  - Systemic dependency on native cryptocurrency (Ether)

Scaling blockchain execution through parallelization

- BlockDAG [Lewenberger et al., FC2015]



- Parallel execution in BFT consensus [Kapritsos et al.,OSDI2012]
  – A variant of which is implemented in Hyperledger Fabric v1 (as we will see)
- And classical database sharding/partitioning
  – Implemented in Hyperledger Fabric through concept of channels
  – Targeted by Ethereum and other blockchains

Hyperledger Fabric v1

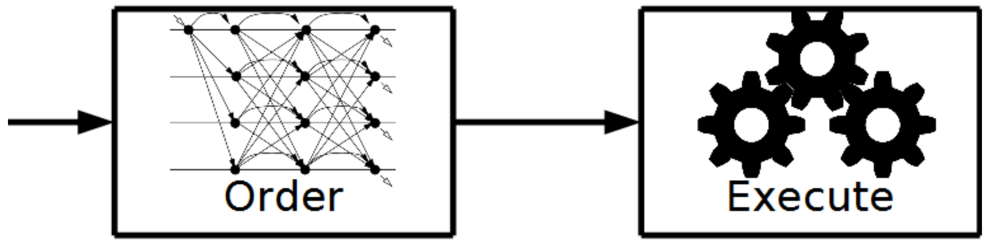
- Androulaki et al. **Hyperledger Fabric: a distributed operating system for permissioned blockchains**, Eurosys 2018
  - https://dl.acm.org/citation.cfm?id=3190538
  - Open source project (Apache 2.0), with strong push from IBM

- **Main features**
  - Parallel execution
  - Can code smart-contracts in general-purpose languages
  - Modular consensus (decoupled from execution, plug in the best available)
  - No native cryptocurrency (can have one as yet another application)

- **Current performance with Fabric v1.1 Bitcoin-like workload (Fabcoin)**
  - 3500 tps with 100 nodes on WANs with commodity cloud hardware
  - Simple optimizations pending to get to 10000 tps

Hyperledger Fabric v1 architecture in one slide

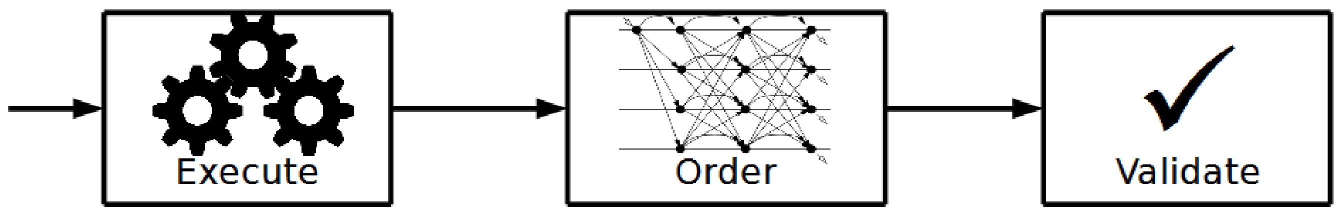
Existing blockchains' architecture



input tx

tx against smart contracts

Hyperledger Fabric v1 architecture



Tx against smart-contracts
Create versioned state updates
Collect endorsements

Versioned state updates/endorsements
Stateless ordering

Versioned updates&endorsements
Flag invalid and conflicting tx
Persist valid tx

Application consists of two components:

1) Chaincode (execution code)
2) Endorsement policy (validation code)

Hyperledger Fabric v1 Transaction flow

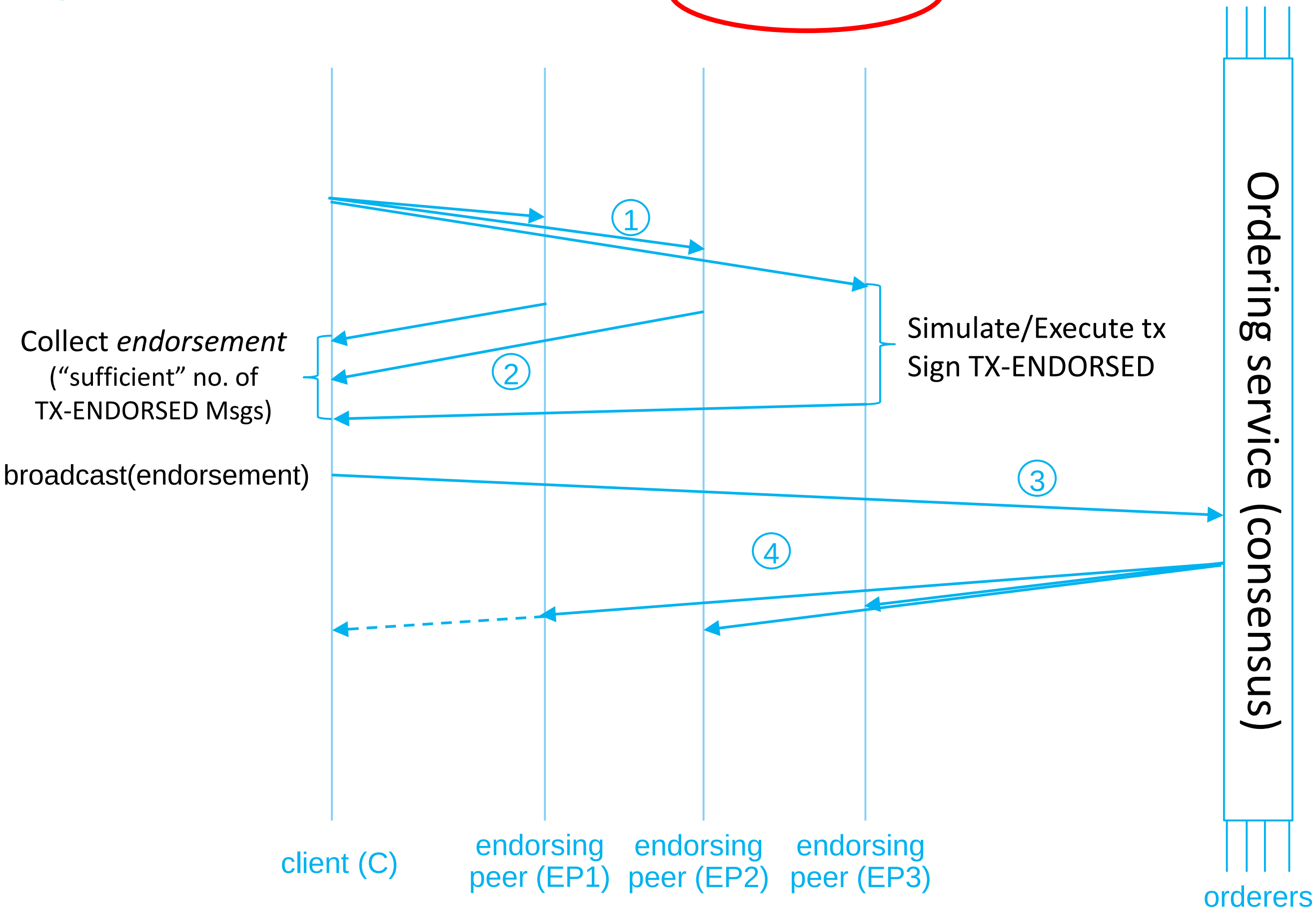
① <PROPOSE, clientID, chaincodeID, txPayload, timestamp, clientSig>

② <TX-ENDORSED, peerID, txID, chaincodeID, readset, writeset>

Total order semantics (ordering service)

③ BROADCAST(blob)

④ DELIVER(seqno,prevhash,block)



Hyperledger Fabric v1 Transaction flow

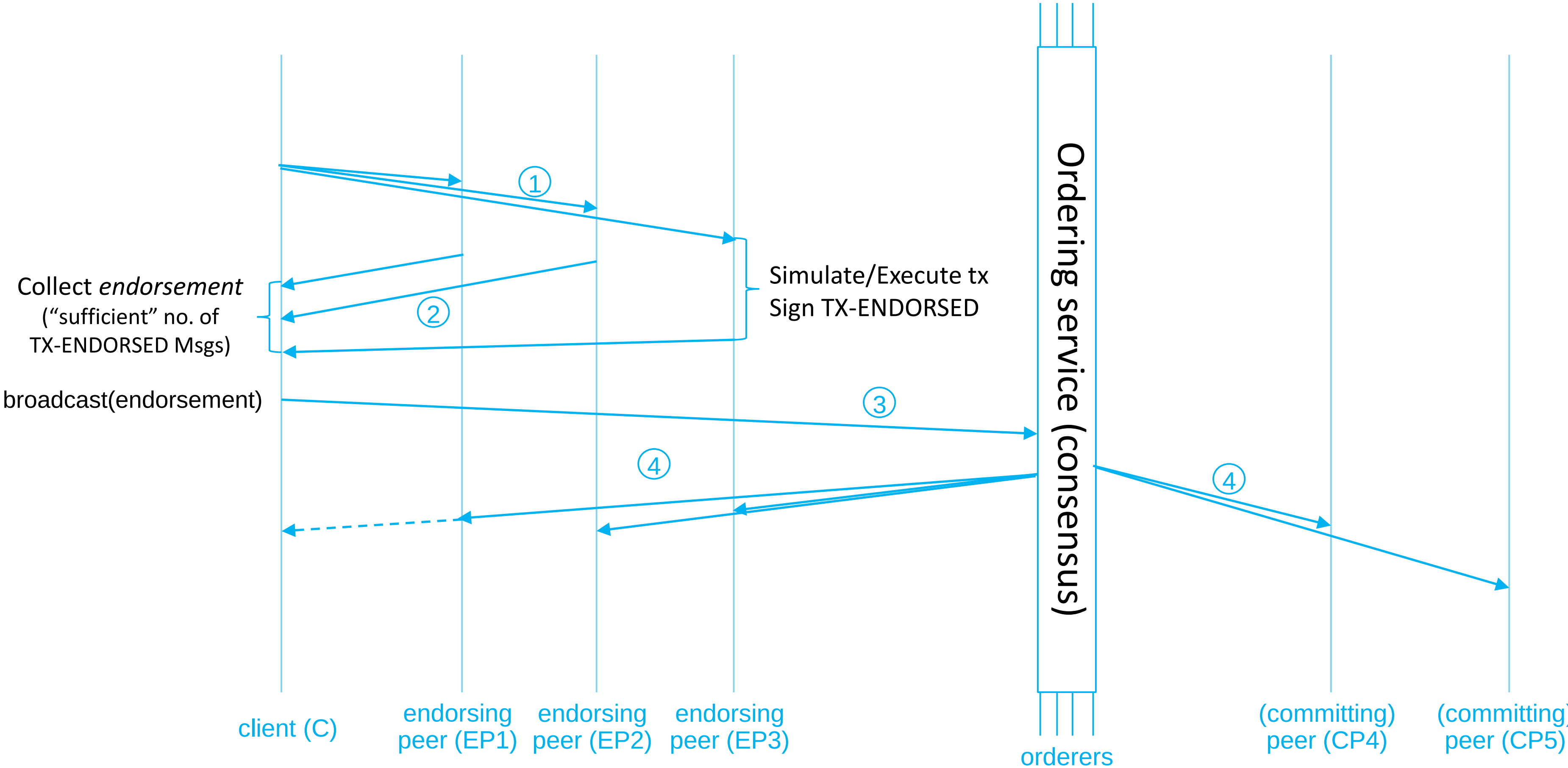
1 <PROPOSE, clientID, chaincodeID, txPayload, timestamp, clientSig>

2 <TX-ENDORSED, peerID, txID, chaincodeID, readset, writeset>

Total order semantics (ordering service)

3 BROADCAST(blob)

4 DELIVER(seqno,prevhash,block)

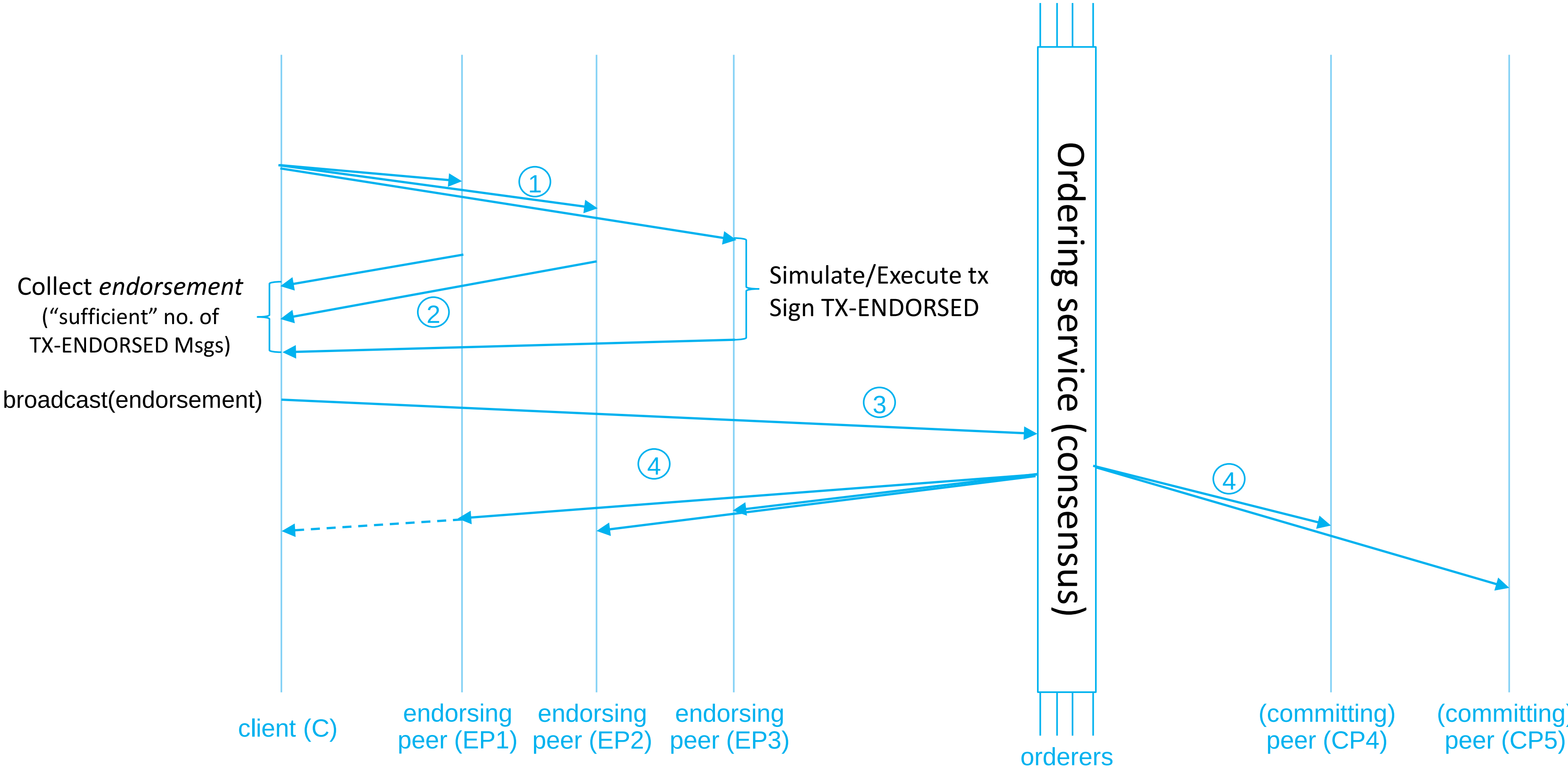


Hyperledger Fabric v1 Transaction flow

1. <PROPOSE, clientID, chaincodeID, txPayload, timestamp, clientSig>
2. <TX-ENDORSED, peerID, txID, chaincodeID, readset, writeset>

Total order semantics (ordering service)

3. BROADCAST(blob)
4. DELIVER(seqno,prevhash,block)



Hyperledger Fabric v1 Transaction flow

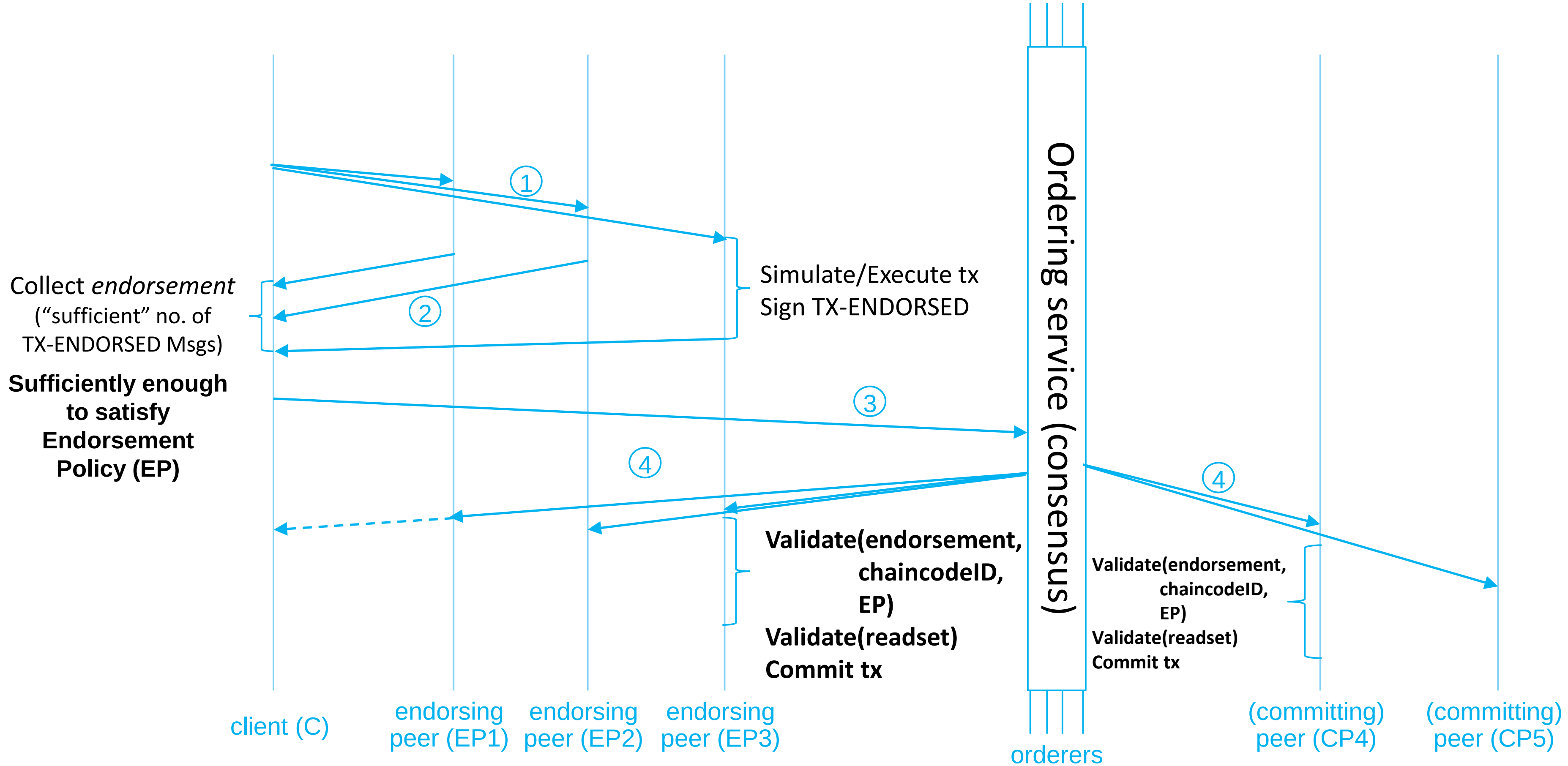
① <PROPOSE, clientID, chaincodeID, txPayload, timestamp, clientSig>

② <TX-ENDORSED, peerID, txID, chaincodeID, readset, writeset>

Total order semantics (ordering service)

③ BROADCAST(blob)

④ DELIVER(seqno,prevhash,block)



Can validate (and execute) transactions in embarasingly parallel manner

Near-term “Holy Grail” of blockchain scalability

Can we have a blockchain protocol

Scaling to hundreds of nodes

Sustaining VISA-like performance numbers?

(seconds latency, about 5k tps on average, few 10k tps peak throughput)

An even “Holier Grail”:

can we do this with some notion of

transaction confidentiality?

Ultimate “Holy Grail” of blockchain scalability

Can we have a blockchain protocol

Scaling to hundred(s) of nodes on WANs

With network bounded throughput

And network/speed of light bounded latency?

An even “More ultimate Holier Grail”:

can we do this with some notion of

transaction confidentiality?

Thank You!

Use of trusted hardware

- Known to, basically, reduce BFT consensus to CFT communication patterns [Chun et al. SOSP’07, Kapitza et al, Eurosys 2012]
  - Still does not address all the scalability issues

- Started impacting PoW consensus (see Intel POET on SGX)

Improving the performance of PoW blockchains

- GHOST (Greedy Heaviest-Observed Sub-Tree) rule [Sompolinsky2015]
  - resolves conflicts in by weighing the subtrees
  - More freedom in increasing the block frequency and the block size than the longest chain rule
  - A variant was due to be implemented in the Ethereum blockchain

- Bitcoin-NG by Eyal et al. [NSDI2016]
  - uses standard PoW for leader election
  - leader can append microblocks to the chain, which are not subject to PoW

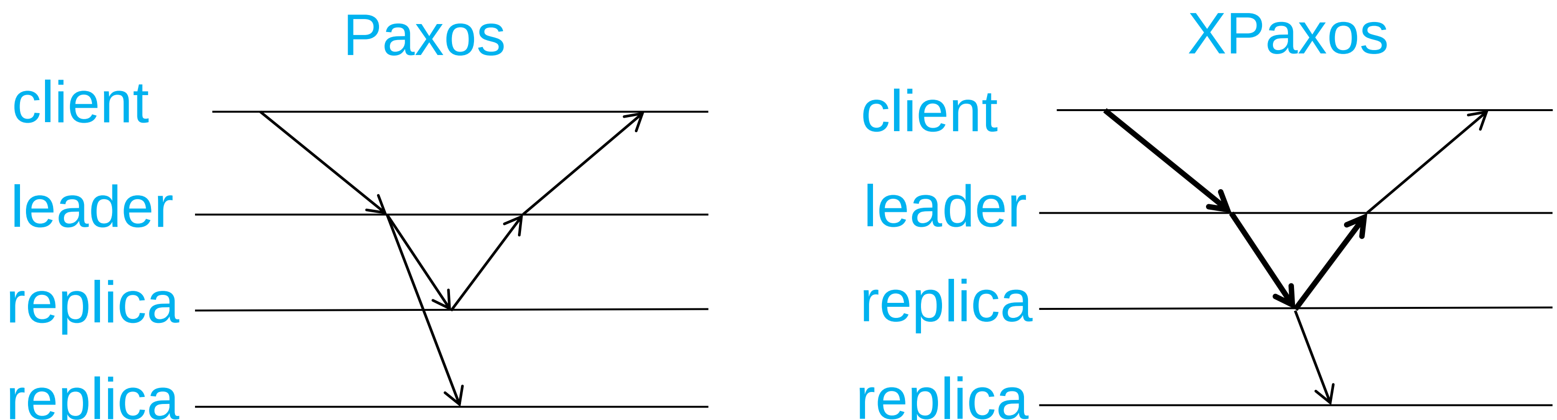
- Forks are still possible and consensus finality not ensured

Hierarchical BFT

- Establish BFT agreement in smaller cliques
- Disseminate the result to other nodes following a hierarchy

- Stellar [Mazieres, 2016]
- SCP [Luu et al, 2016]
  – Also a hybrid PoW/BFT protocol, using PoW for identity management and (parallel and hierarchical) BFT consensus for agreement

XPaxos message pattern (common case)



→ Digitally signed messages

Reconfiguration (leader election)
is more complex but its cost is amortized over time