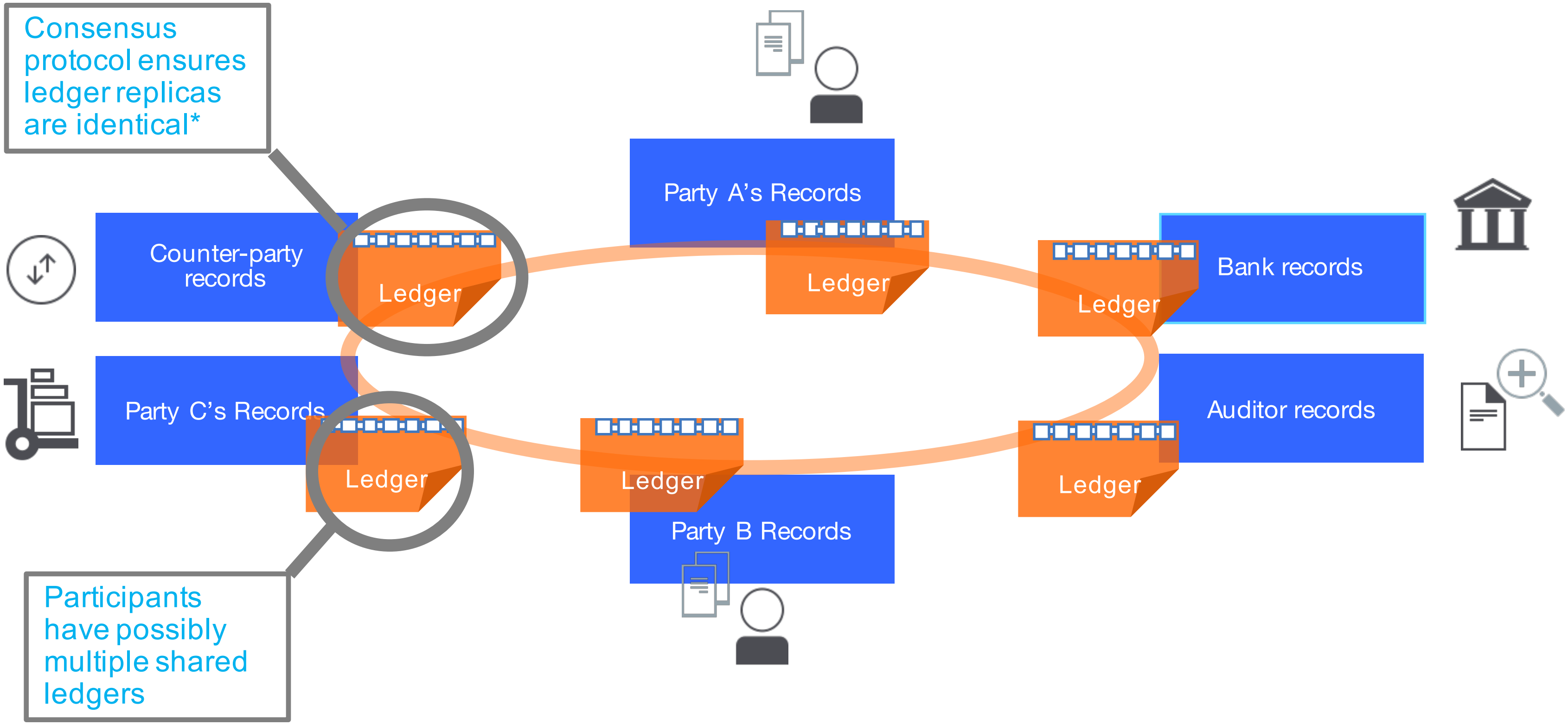
Marko Vukolić, IBM Research - Zurich

Hyperledger fabric: towards scalable blockchain for business

Trust in Digital Life

The Hague, Netherlands, June 17 2016

Blockchain – shared, replicated, ledger

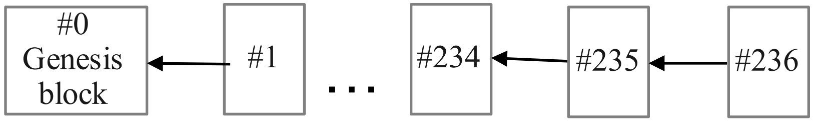


Consensus protocol ensures ledger replicas are identical*

Participants have possibly multiple shared ledgers

What is a Blockchain?

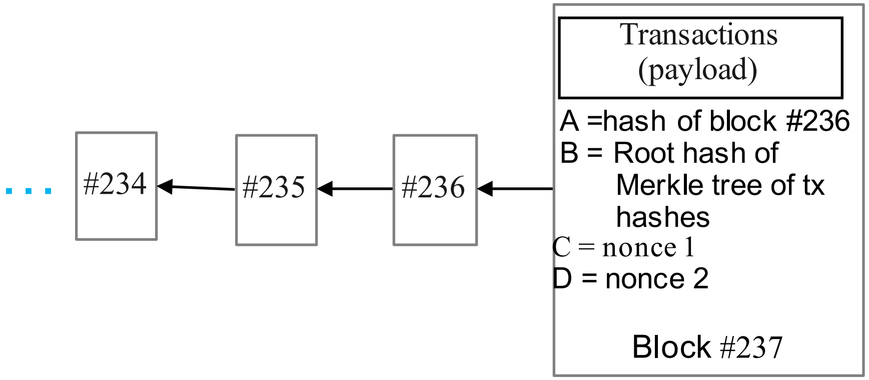
- **A chain (sequence) of blocks of transactions**
  - Each block consists of a number of transactions



- **Bitcoin transactions**
  - simple virtual cryptocurrency transfers
  - (address A, address B, amount)

- **Transactions do not have to be simple nor related to cryptocurrency**
  - E.g., smart contracts (Ethereum)
  - chaincode (Hyperledger)

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

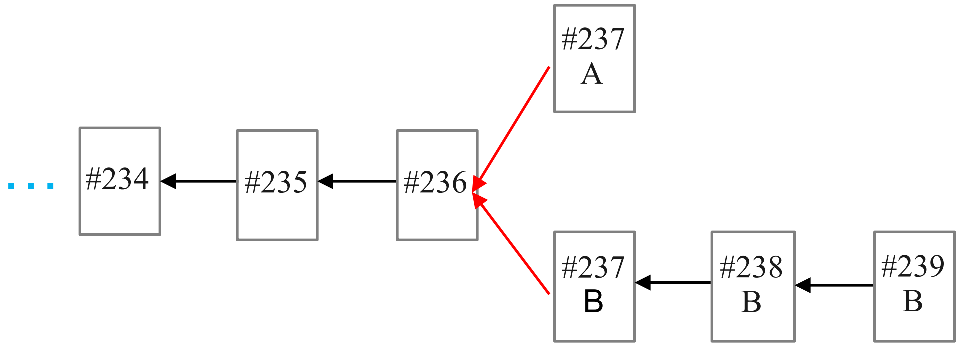


$$h = \text{hash of Block \#237} = \text{SHA256}(A||B||C||D)$$

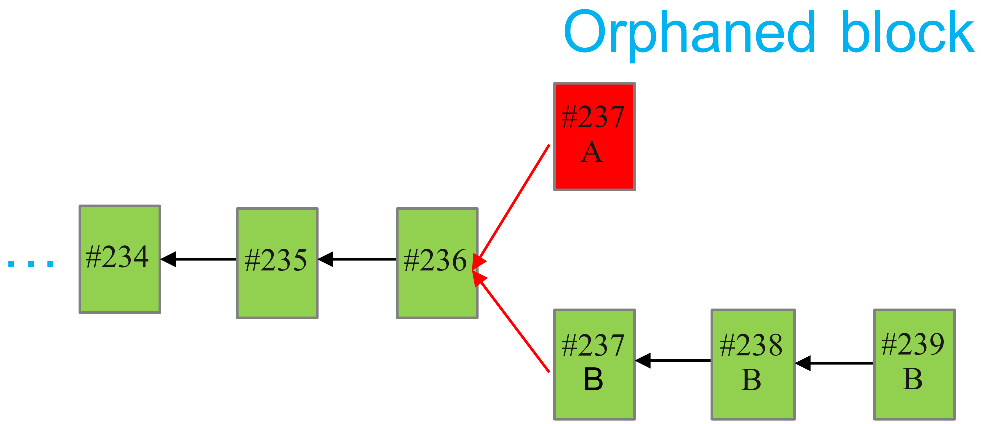
- Block “mining”:
  - Every participant (“miner”) tries to find nonces
  - such that the hash of the block *h* **<u>is lower than a 256-bit *target*</u>**

- Bitcoin
  - Target dynamically adjusted: 1 block generated roughly every 10 minutes
  - Already in 2014, this required more than 2^{80} expected hashes

Example (longest/most difficult chain wins)



Example (longest/most difficult chain wins)



Implications and the performance issue

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

- Better performance by tuning PoW parameters?
  - shorter block generation times (increasing block frequency)?
  - larger blocks?
  - Different conflict resolution rules?
  - **Limited benefits, potentially weaker security**

Introducing smart contracts/chaincode

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

aim at supporting “smart contracts” or “chaincodes”

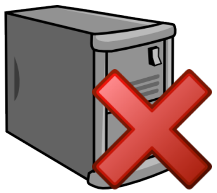
A smart contract is an event driven program, with state, which runs on a replicated, shared ledger and which can take custody over assets on that ledger. [Swanson2015]

“Smart contract” → (replicated) state machine

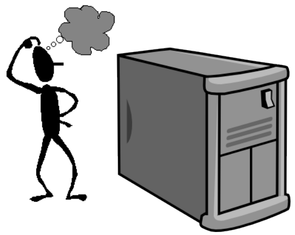
State machine replication (SMR)

- Classical Distributed Computing problem

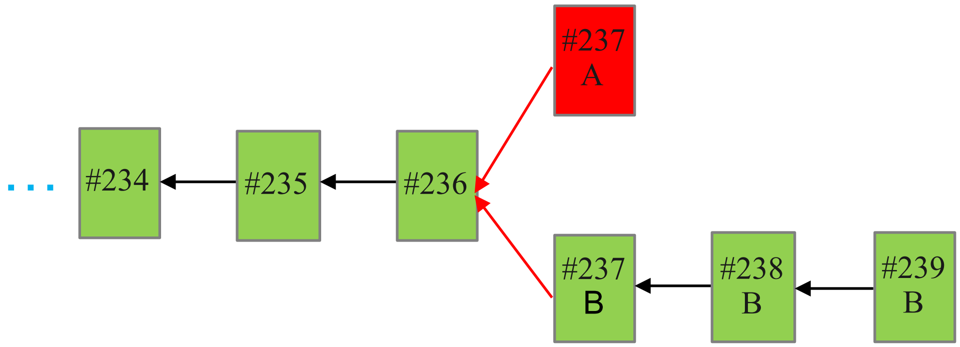
What machine faults?



- Crash faults (CFT): A machine simply stops execution and halts
  - Paxos, RAFT, Zookeeper AB,…

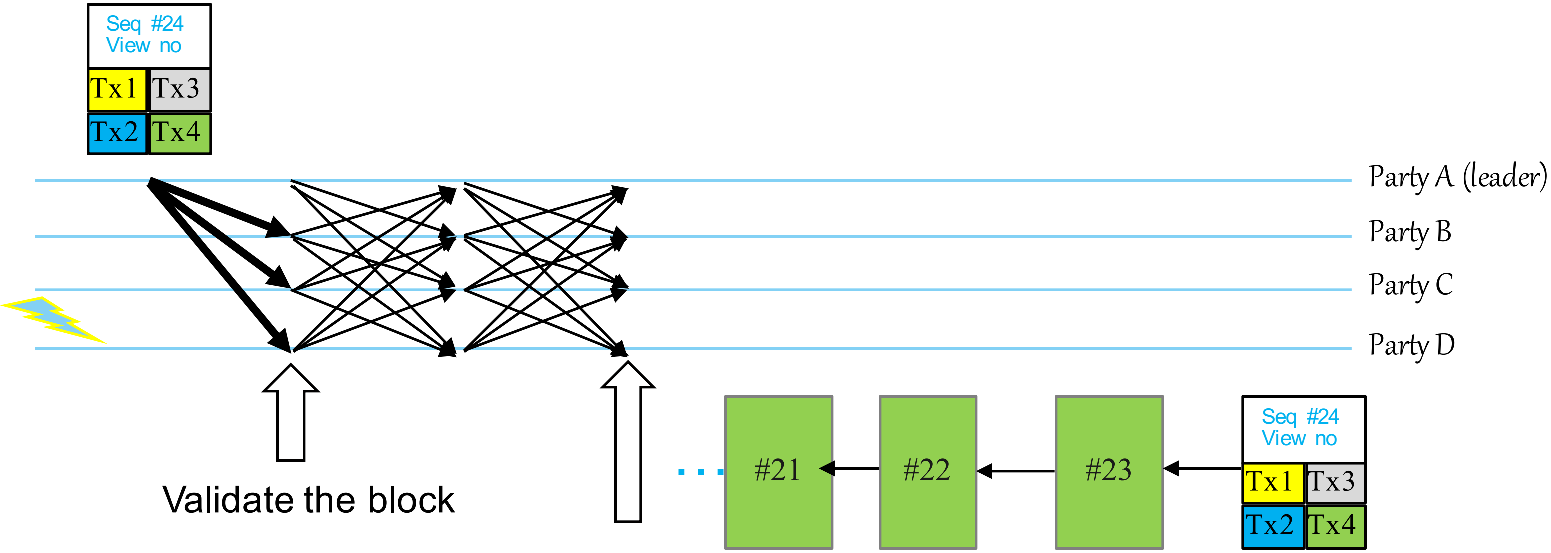


- Non-crash (a.k.a. Byzantine) faults (BFT)
  - A model that cryptocurrencies adopt



No forks!

BFT Consensus (example of PBFT [TOCS2002])



Validate the block

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

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

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

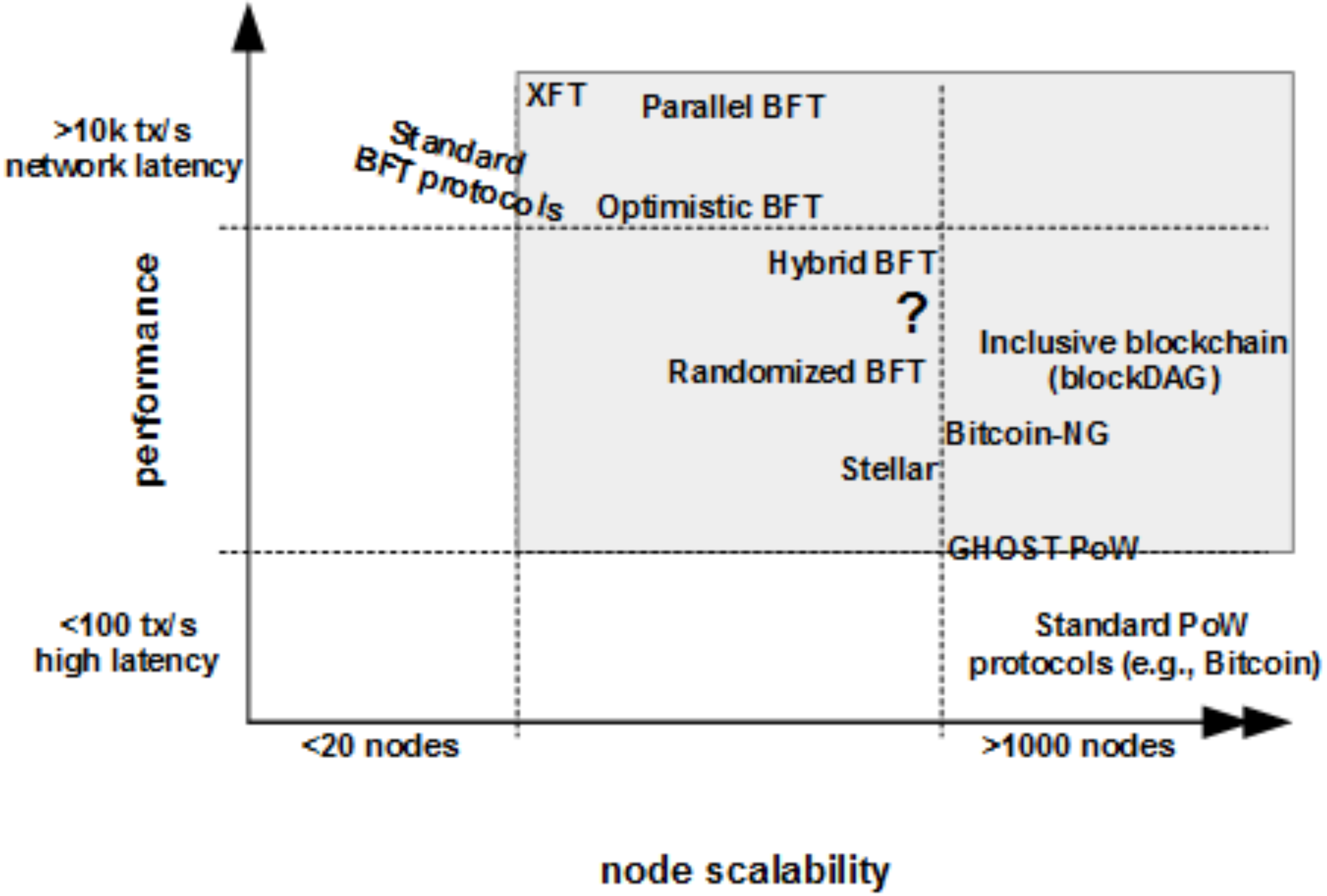
PoW vs. SMR for Blockchain (simplified overview)

	Proof of Work (Bitcoin, Ethereum,...)	State machine replication (Ripple, Hyperledger, …)
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 protects against Sybil attacks)
Scalability	Excellent >100k Nodes	Verified up to few tens (or so) Nodes
Throughput	7 tx/sec upper bound (Bitcoin)	>10k tx/sec with existing implementations in software
Power efficiency	>1 GW (Bitcoin)	Good (commodity hardware)
Temporary forks in blockchain	Possible (leads to double-spending attacks)	Not possible
Consensus Finality	No	Yes

Open research problem:

Given the use case, network, no. of nodes

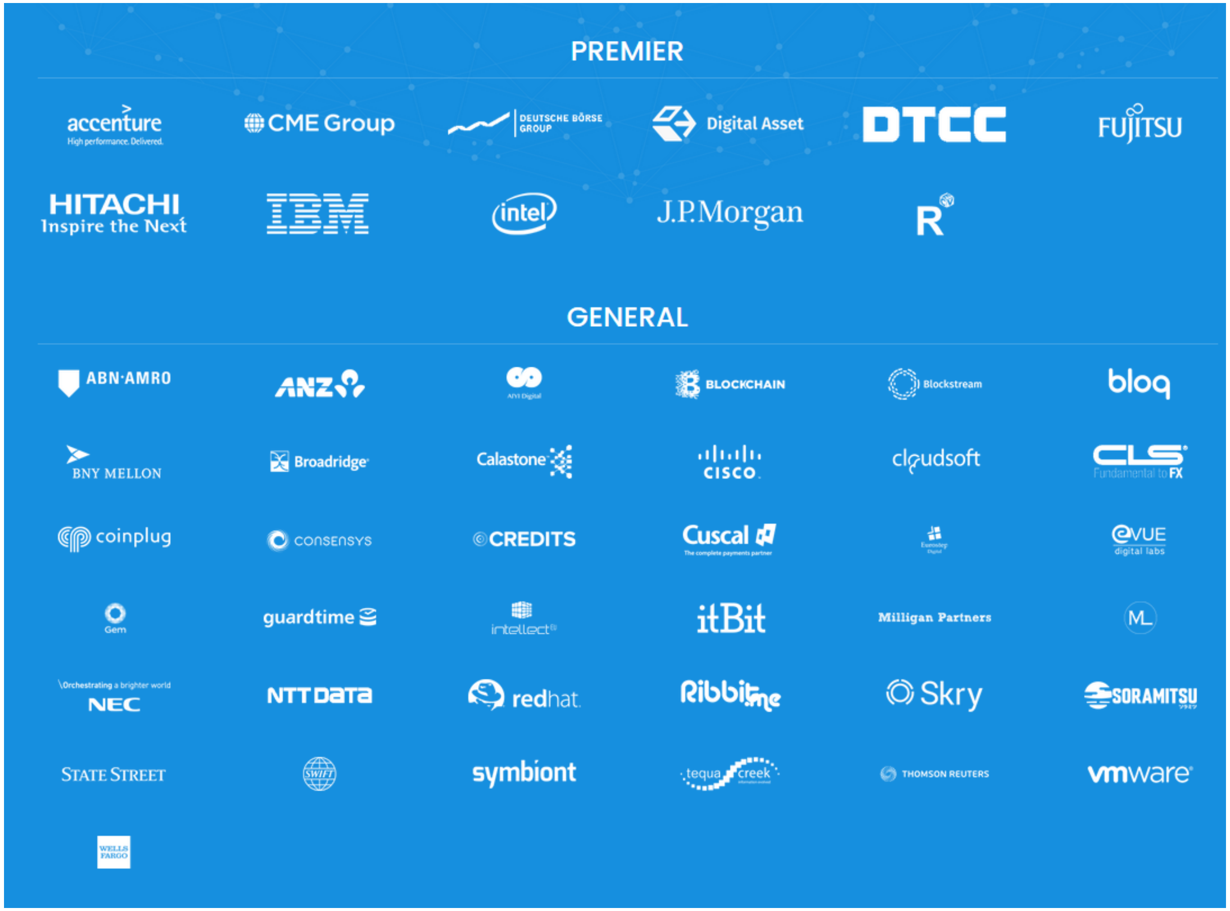
What is the most suitable and scalable Blockchain technology/protocol?



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).

THE LINUX FOUNDATION

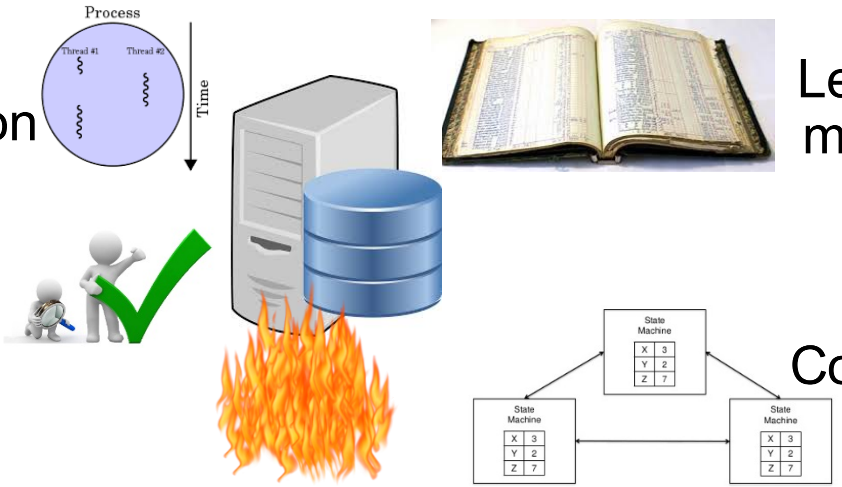
HYPERLEDGER PROJECT



https://github.com/hyperledger

https://www.hyperledger.org/

Existing blockchains unify many functionalities in one node



Smart contract execution

Ledger/state maintenance

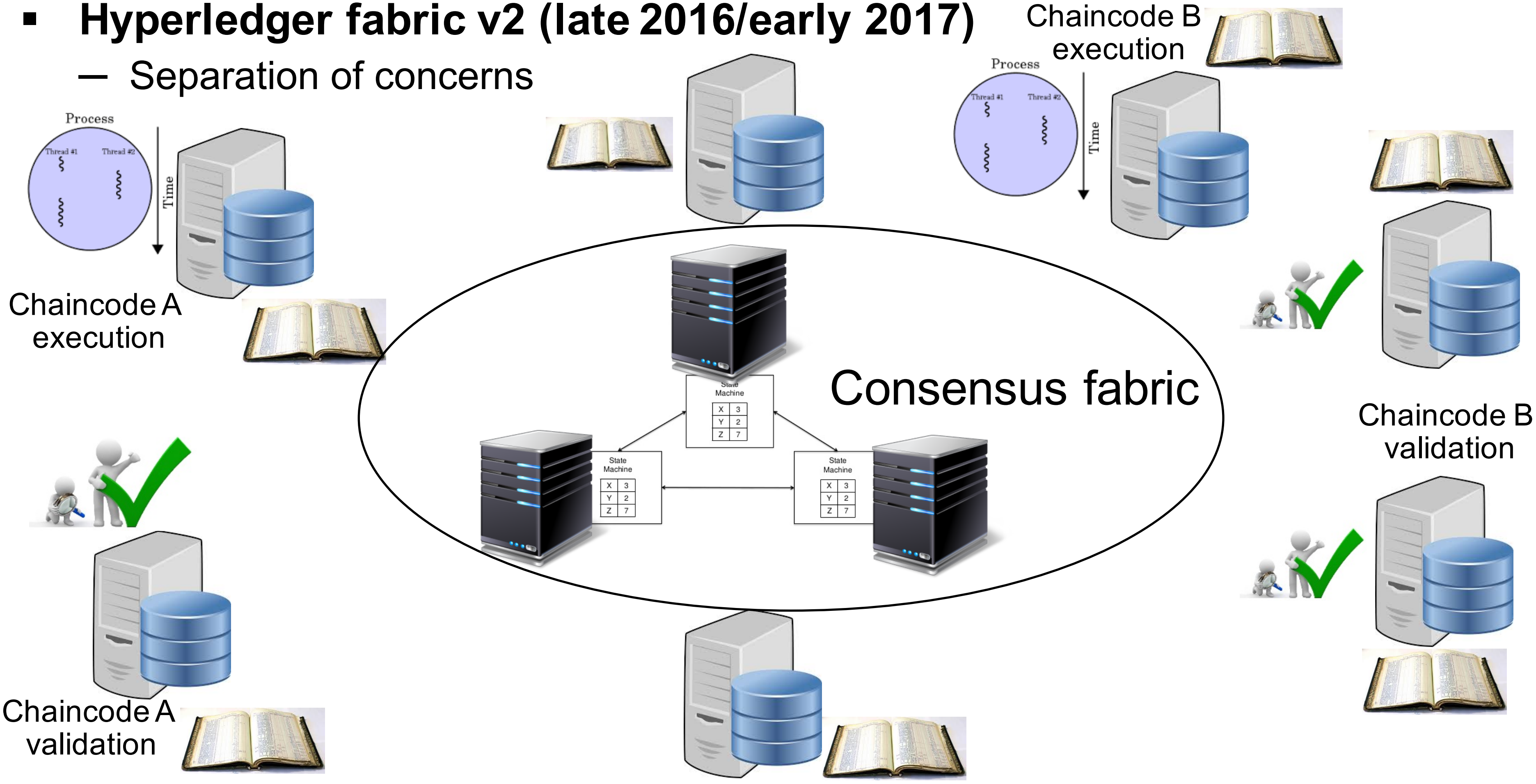
Transaction validation

Consensus logic

This limits achievable performance and harms scalability
At odds with confidentiality

Hyperledger fabric v2 – architecting a scalable blockchain

- **Hyperledger fabric v2 (late 2016/early 2017)**
  - Separation of concerns



Architecture-level approach to scalable and confidential blockchain

Goal: Towards hundreds of consenters/peers running many thousands tps

https://github.com/hyperledger/fabric/wiki/Next-Consensus-Architecture-Proposal

Blockchain fabric comparison

Feature	Attribute	Bitcoin (digital cash)	Ripple (inter-bank remittances)	Ethereum (distributed applications)	Hyperledger fabric (generic blockchain fabric)
Open Membership	*Permissioned vs. Permissionless*	Permissionless	Permissioned	Permissionless	Permissioned
No transaction, once verified, can be changed by any party	*Consensus algorithm*	Proof of work	(custom-made) Byzantine fault-tolerant (BFT) consensus	Proof of work, Proof of stake	Pluggable consensus framework (currently: proven practical BFT)
Prevention of asset double-spending					
Business logic can self-execute with assurance that the terms can not be altered by any party without agreement from stakeholders	*Smart contracts support*	Very limited (stack-based scripting language)	None (had Codius, but discontinued)	Solidity domain specific language (DSL) (Turing-complete)	Go (golang), Java (in progress) + Support for other languages and DSLs envisioned in future
Transaction execution evolves around a blockchain-specific digital currency	*Native cryptocurrency*	Yes (BTC)	Yes (XRP)	Yes (ETH)	No
Transaction confidentiality	*Encryption, key-distribution Cryptographic mechanisms*	No	No	Smart contract level confidentiality	Smart contract (chaincode) level + fabric-level confidentiality

https://github.com/hyperledger/fabric

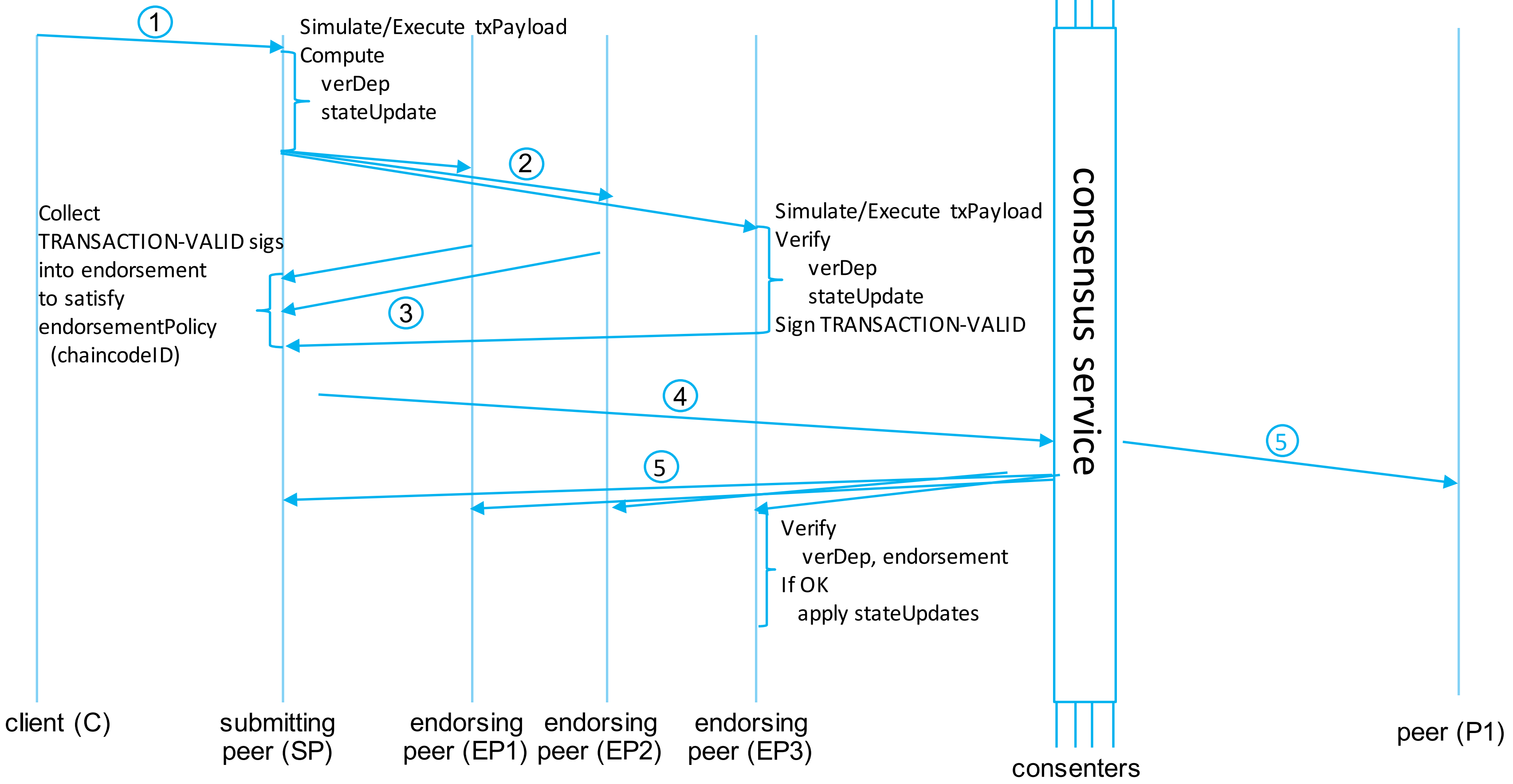
Thank You!

Hyperledger (v2) transaction flow

1 <SUBMIT,cID,chaincodeID,txPayload,sigC>

2 <PROPOSE,txPayload,tran-proposal,sigSP> (tran-proposal := (spID,clientID,chaincodeID,HASH(txPayload),stateUpdate,verDep))

3 <TRANSACTION-VALID, txID,sigEPi>



Consensus service API:

- Broadcast(blob) 4
- Deliver(seqno,prevHash,blob) 5

blob=(tran-proposal, endorsement)