Distributing Trust with Blockchains

Christian Cachin

IBM Research – Zurich

October 2018

Blockchain – new opportunities

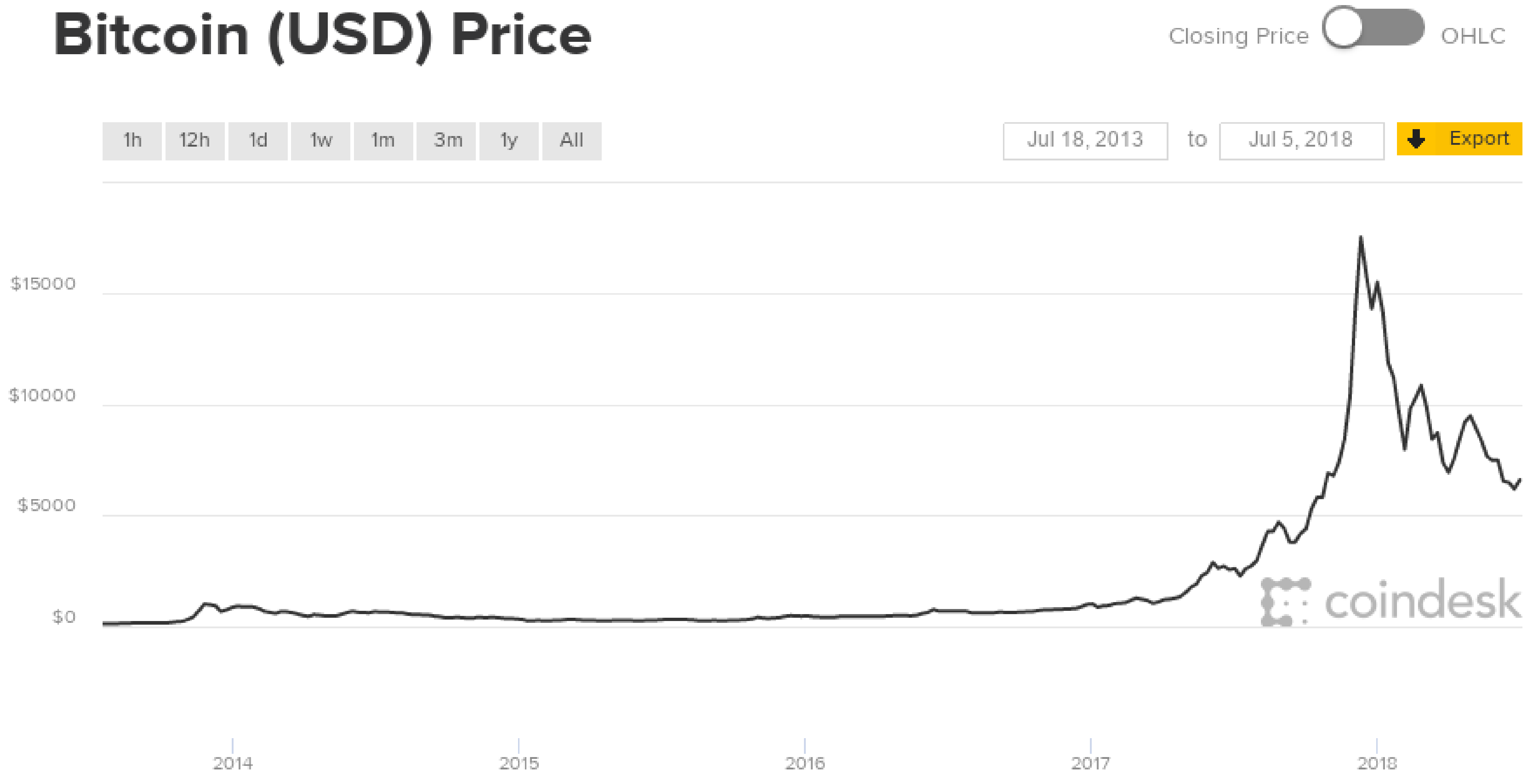
- Automates trust
- Replaces authorities by technology
- Eliminates intermediaries

- Adds transparency
- Reduces risk
- Stores significant value

Bitcoin

- First cryptocurrency
- Introduced blockchain
- Decentralized, trustless, anonymous
- Resists censorship
- Nobody owns it – Satoshi Nakamoto?
- Roots in "cypherpunks" movement of ~1990-1995

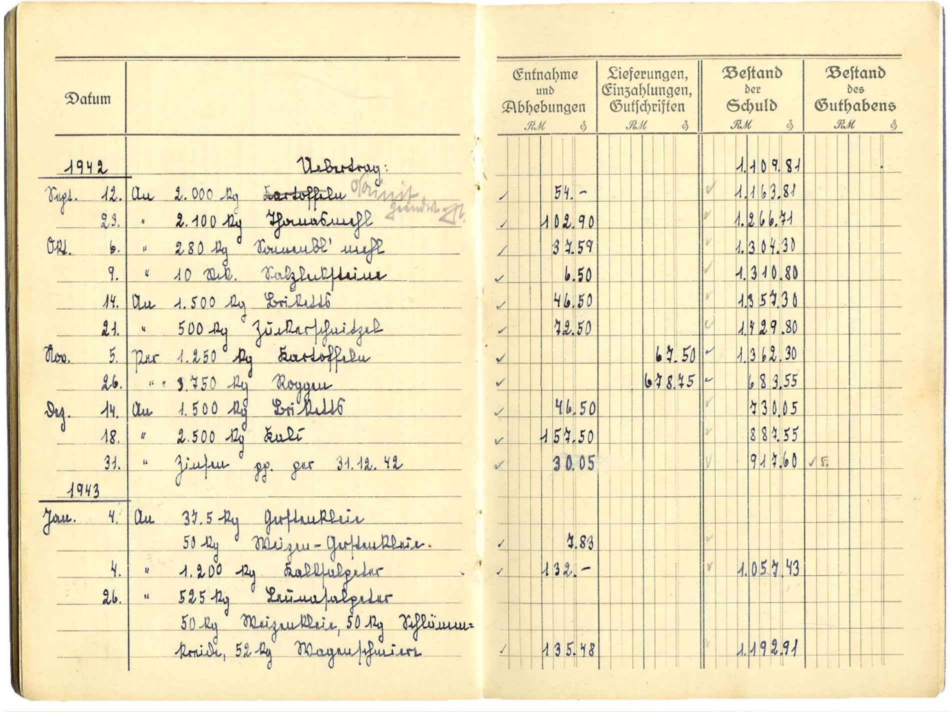
Bitcoin (USD) Price



In cryptography we trust (?)

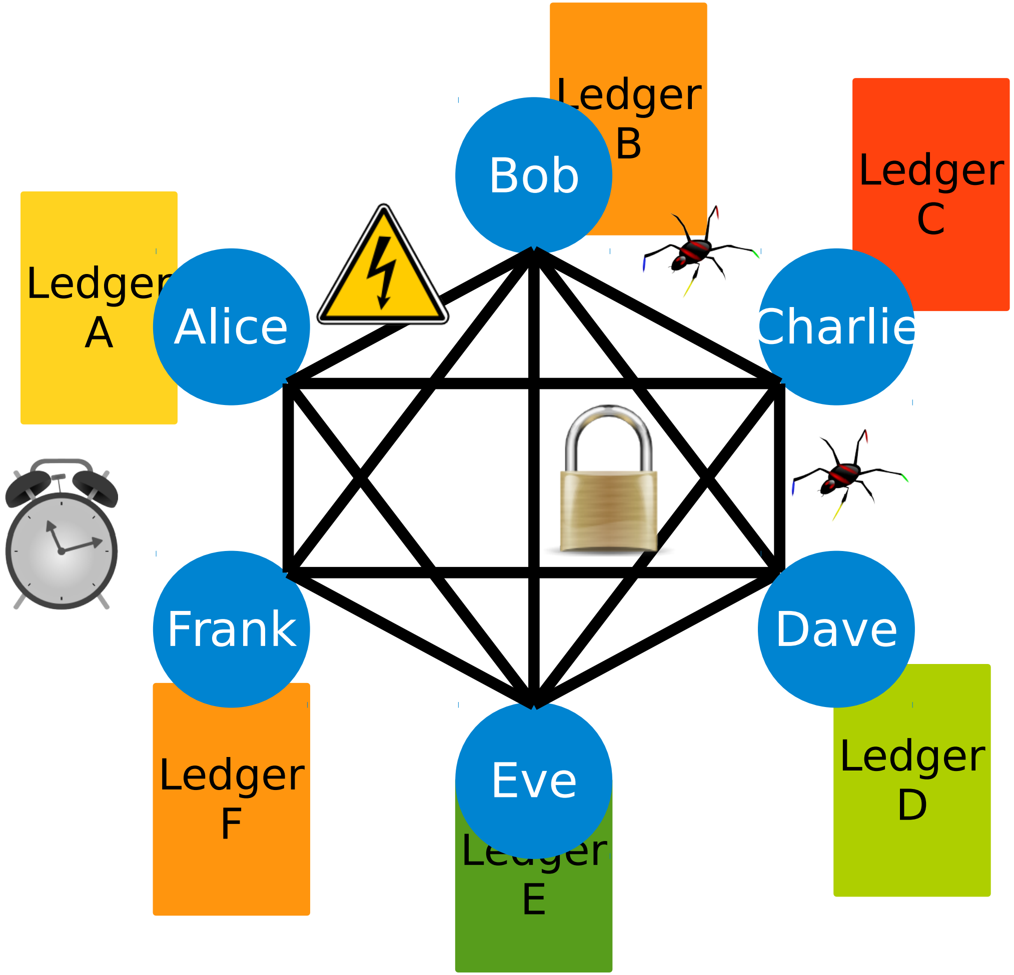
What is a blockchain?

Ledger



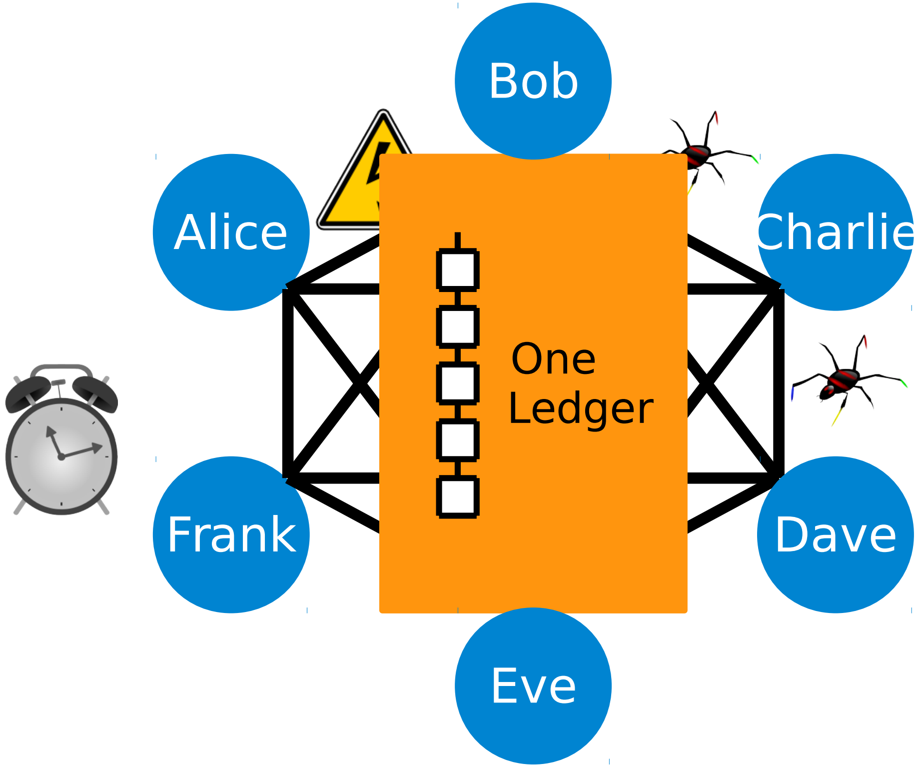
- Ledger records all business activity as transactions
  - Database
- Every market and network defines a ledger
- Ledger records asset transfers between participants
- Problem — (Too) many ledgers
  - Every market has its ledger
  - Every organization has its own ledger

Multiple ledgers



- Every party keeps its own ledger and state
- Problems, incidents, faults
- Ledgers diverge
- Reconciliation is expensive

Blockchain provides one virtual ledger



- One covirtual trusted ledger
- Today often implemented by a centralized intermediary
- Blockchain holds the world state for all parties
- Replicated and produced collaboratively
- Trust in ledger from
  - Cryptographic protection
  - Distributed validation

Four elements characterize Blockchain

Replicated ledger

- History of all transactions
- Append-only with immutable past
- Distributed and replicated

Cryptography

- Integrity of ledger
- Authenticity of transactions
- Privacy of transactions
- Identity of participants

Consensus

- Decentralized protocol
- Shared control tolerating disruption
- Transactions validated

Business logic

- Logic embedded in the ledger
- Executed together with transactions
- From simple "coins" to self-enforcing "smart contracts"

Blockchain simplifies complex transactions

Financial assets

Faster settlement times

Increased credit availability

Transparency & verifiability

No reconciliation cost

Property records

Digital but unforgeable

Fewer disputes

Transparency & verifiability

Lower transfer fees

Logistics

Real-time visibility

Improved efficiency

Transparency & verifiability

Reduced cost

Blockchain scenario features

- A given task or problem, but no (central) trusted party available

- Protocol among multiple nodes, solving a distributed task
  - The writing nodes decide and reach consensus collectively

- Key aspects of the distributed task
  - Stores data
  - Multiple nodes write
  - Not all writing nodes are trusted
  - Operations are (somewhat) verifiable

- If all writing nodes are known → permissioned or consortium blockchain
- Otherwise, when writing nodes are not known → permissionless or public blockchain

Why blockchain now?

- Cryptography has been a key technology in the financial world for decades
  – Payment networks, ATM security, smart cards, online banking ...

- Trust model of (financial) business has not changed
  – Trusted intermediary needed for exchange among non-trusting partners
  – Today cryptography mostly secures point-to-point interactions

- Bitcoin started in 2009
  – Embodies only cryptography of 1990s and earlier
  – First prominent use of cryptography for a new trust model (= trust no entity)

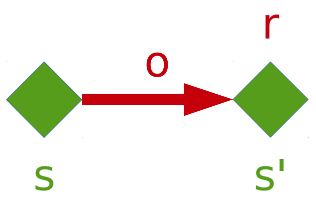
- The promise of Blockchain – Reduce trust and replace it by technology
  – Exploit advanced cryptographic techniques

Again – What is a blockchain?

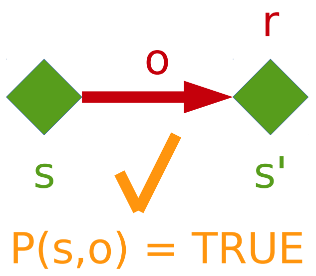
A state machine

- Functionality F
  - Operation o transforms a state s to new state s' and may generate a response r

$(s', r) \leftarrow F(s, o)$

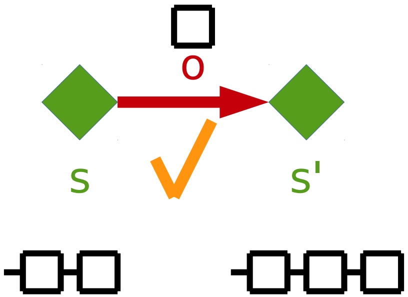


- Validation condition
  - Operation needs to be valid, in current state, according to a predicate P()



Blockchain state machine

- Append-only log
  - Every operation o appends a "block" of valid transactions (tx) to the log



- Log content is verifiable from the most recent element

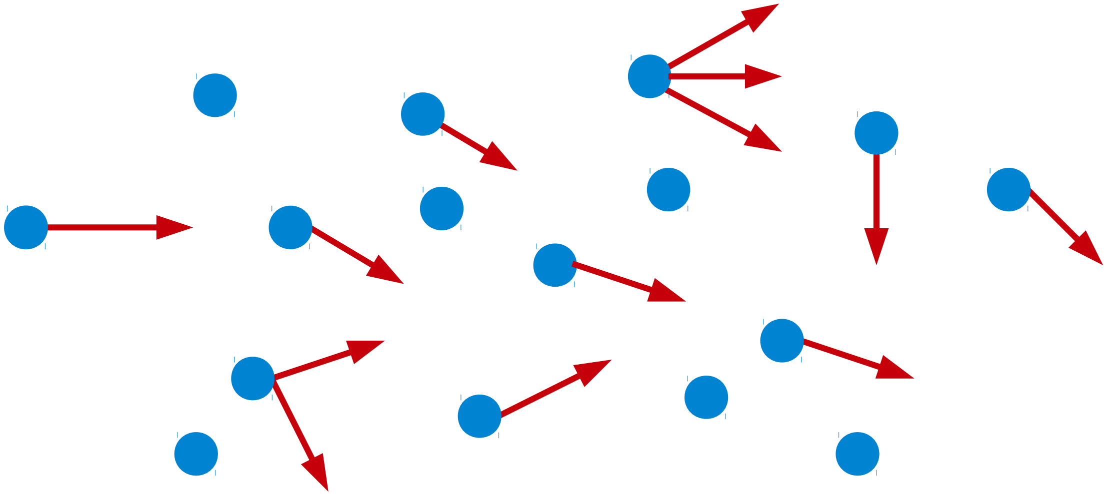
- Log entries form a hash chain

  $h_t \leftarrow \text{Hash}(\ [tx_1, tx_2, \ldots\ ]\ ||\ h_{t-1}\ ||\ t)$.

Example – The Bitcoin state machine

- Bitcoins are unforgeable bitstrings
  - "Mined" by the protocol itself (see later)
- Digital signature keys (ECDSA) own and transfer bitcoins
  - Owners are pseudonymous, e.g., 3JDs4hAZeKE7vER2YvmH4yTMDEfoA1trnC
- Every transaction transfers a bitcoin (fraction) from current to next owner
  - "This bitcoin now belongs to 3JDs..." signed by the key of current owner
  - The coin flow is linkable by design, not anonymous when connected to the real world
- Validation is based on the global history of past transactions
  - Signer has received the bitcoin before
  - Signer has not yet spent the bitcoin

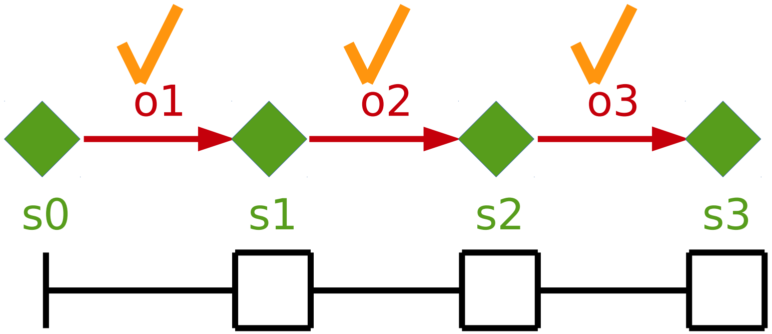
A consensus protocol creates the blockchain



Nodes

produce

transactions (tx)



Protocol orders transactions and constructs the ledger

Blockchain protocol features

- Only "valid" operations (transactions) are "executed"

- Transactions can be simple
  - Bitcoin tx are statement of ownership for coins, digitally signed
    "This bitcoin now belongs to K2" signed by K1

- Transactions can be arbitrary code (smart contracts)
  - Embody logic that responds to events (on blockchain) and may transfer assets in response
  - Auctions, elections, investment decisions, blackmail ...

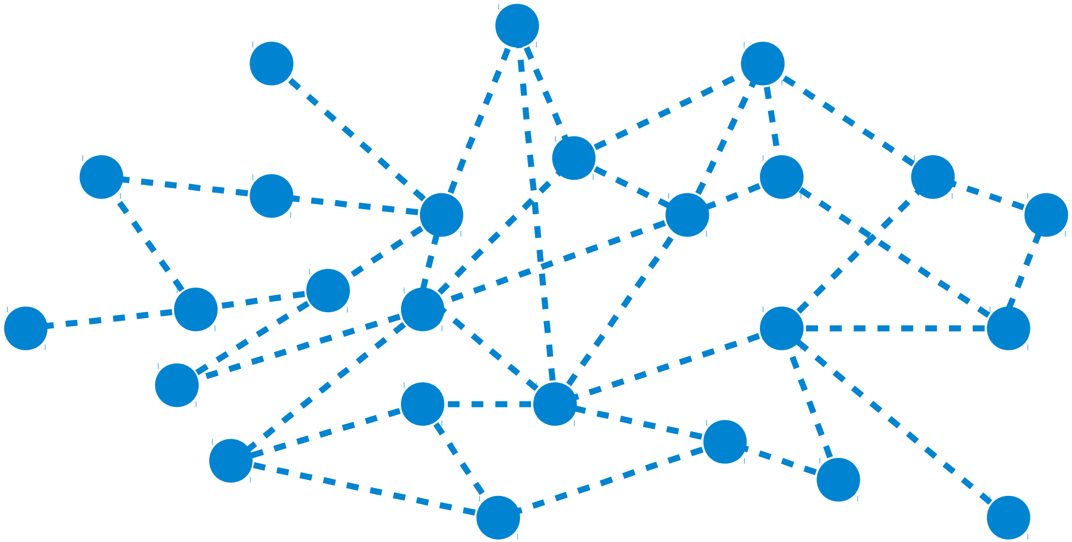
Consensus

Types of blockchain consensus

- Decentralized / permissionless / Nakamoto consensus
  - Bitcoin, Ethereum, ...
- Consortium / permissioned / BFT consensus
  - BFT (Byzantine fault tolerance) consensus, quorums
  - Flexible quorums: Ripple and Stellar
- Weighted by stake / rational agreement / proof-of-stake consensus
  - Peercoin, Cardano-Ouroboros, Algorand, Ethereum-Casper ...
- DAG protocols
  - SPECTRE, Hashgraph, IOTA Tangle, Snowflake-Avalanche, Conflux ...

Decentralized / permissionless / Nakamoto consensus

Decentralized – Permissionless



- Anyone can join
- Sybil attacks
- No traditional votes

- Bitcoin's idea: One CPU = One vote
- "Vote" by investing and proving work

Nakamoto consensus in Bitcoin, Ethereum …

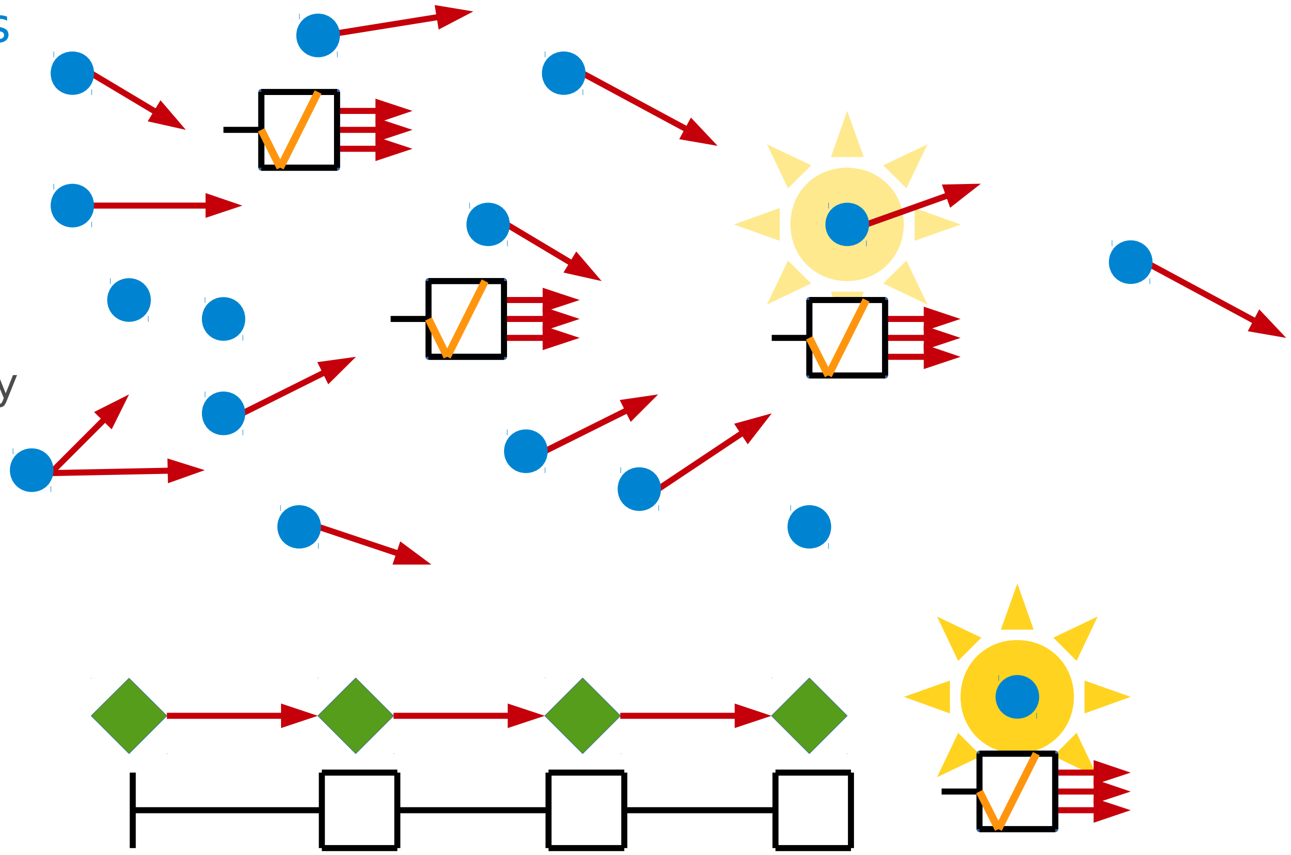
- Voting not possible
  - Anyone can join, a malicious party may claim many pseudonyms (Sybil attack)
- For consistency and ordering transactions, use a leader
- Idea
  - Probabilistically determine a leader (once every ~10 mins in Bitcoin)
  - Provide an incentive to be a good, correct leader → receives a newly "mined" coin
  - To be elected, a candidate grows the ledger and orders transactions
- Approach
  - Determine leader by lottery
  - The first candidate to solve a useless cryptographic puzzle wins

Decentralized – Nakamoto consensus/Bitcoin

- All nodes prepare blocks
  - List of transactions (tx)
  - All tx valid

- Lottery race
  - Solves a hard puzzle
  - Selects a winner randomly
  - Winner's block of tx are executed and
  - Winner "mines" a coin

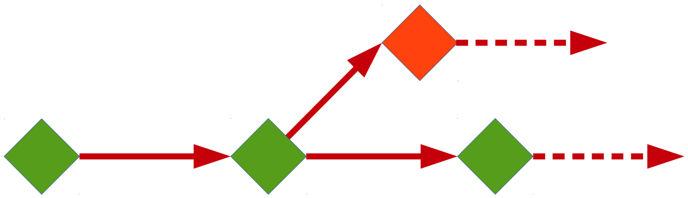
- All nodes verify and validate new block
  - "Longest" chain wins



How does proof-of-work ensure consistency?

- Miners solve puzzle to create blocks
  - Concurrent, include conflicting tx
  - Disseminate block, fast
  - Mining reward

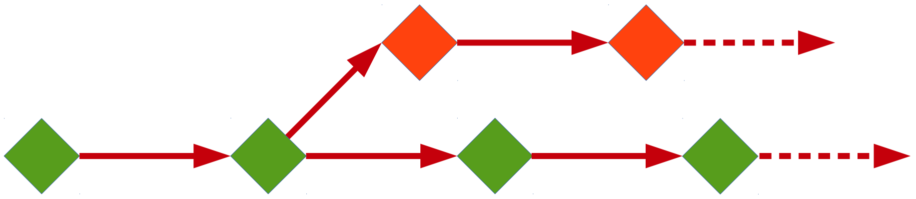
- "Longest" chain wins



- Forks occur regularly

How does proof-of-work ensure consistency?

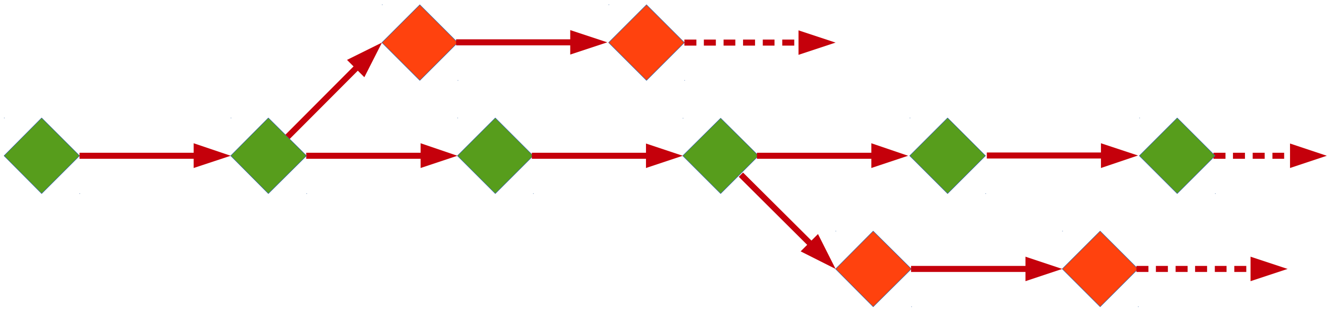
- Miners solve puzzle to create blocks
  - Concurrent, include conflicting tx
  - Disseminate block, fast
  - Mining reward
- "Longest" chain wins



- Forks occur regularly
  - With probability independent of past
- Forks do not last forever, with high probability

How does proof-of-work ensure consistency?

- Miners solve puzzle to create blocks
  - Concurrent, include conflicting tx
  - Disseminate block, fast
  - Mining reward
- "Longest" chain wins



- Forks occur regularly
  - With probability independent of past
- Forks do not last forever, with high probability
  - Bitcoin tx confirmed if 6 blocks deep
  - Probability of k-blocks long fork exponentially small in k
- Alternative rules exist to select winning chain (GHOST ...)

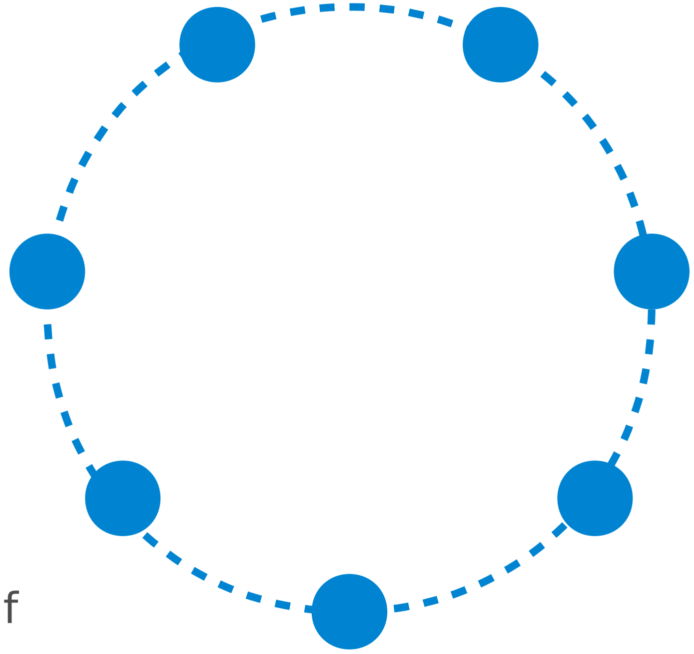
Features of decentralized consensus

- Survives censorship and suppression **(+ / —)**
  - No identities, no counting of nodes
  - Give incentive to participate with mining reward
- Scales to 1000s of nodes **(+)**
- High latency (minutes or more), and decisions are never final **(—)**
- Requires proof-of-work (PoW) **(—)**
  - Majority of hashing power controls the network
- Waste-of-work: Bitcoin's PoW consensus consumes huge amounts of power
  - Bitcoin consumes 20% more electricity than Switzerland
    (bitcoinenergyconsumption.com // Bundesamt für Energie (BFE), Stromverbrauch 2017)

Consortium / permissioned / BFT consensus

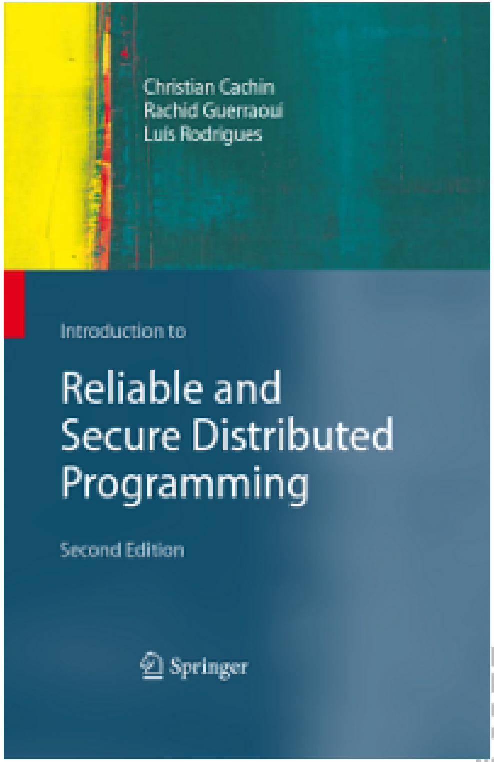
Consortium – Permissioned – BFT

- Traditional consensus based on voting
- Defined group of validator nodes
- Has been studied for decades
  - Byzantine Fault Tolerance (BFT)
  - Elaborate mathematical theory (quorums)
  - Clear assumptions and top-down design
- Many variations possible
  - Change group membership through protocol itself
  - Votes weighted by stake
- Implementations available, some open source



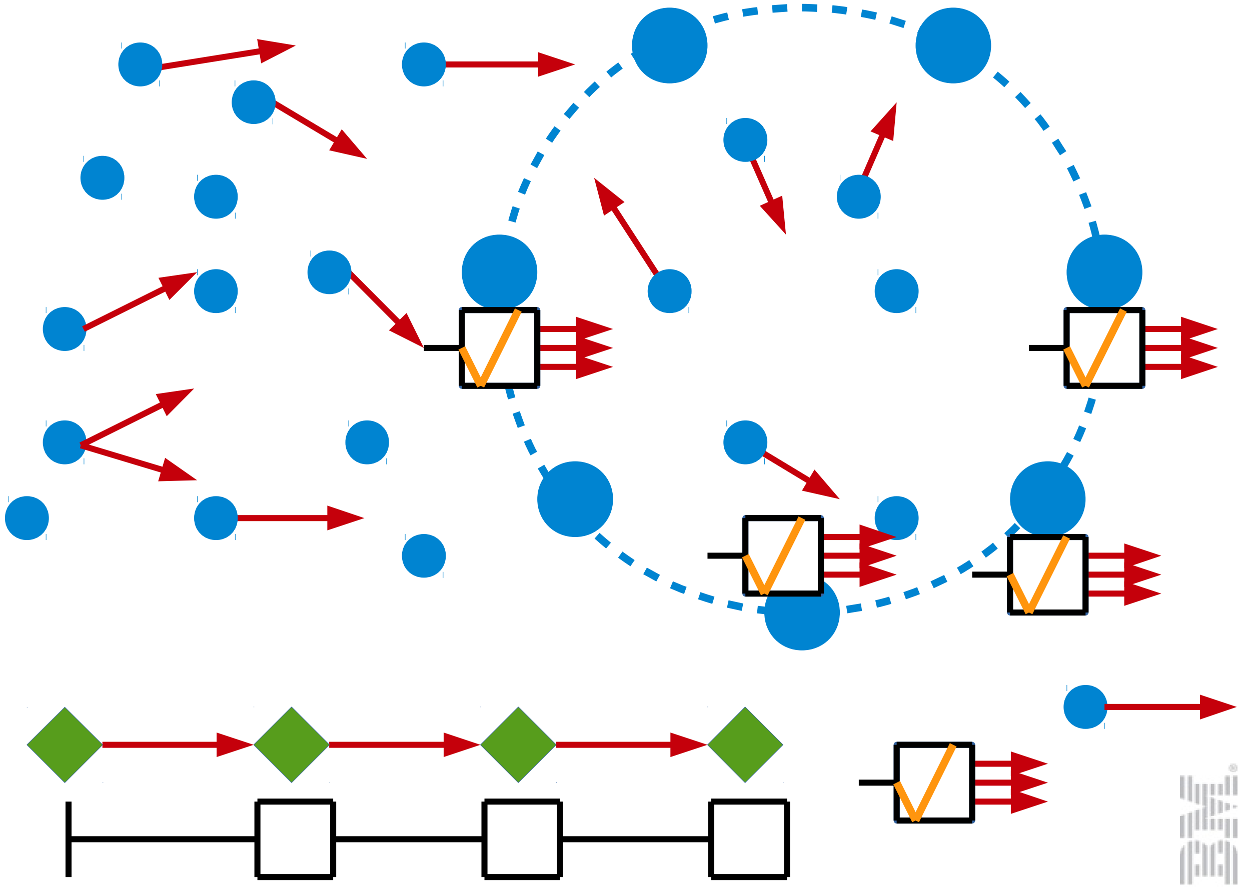
History of BFT consensus research

- Helped develop the field of distributed computing
  - The mathematical consensus abstraction plays a key role
  - Rich body of literature, textbooks ...
- Computer-science theory research
  - Very active topic ca. 1985–2000
  - Many theorems, no systems (cf. Paxos, VSR ...)
- Computer systems research
  - Very active topic ca. 1999–2010
  - Many systems, no deployment (cf. ZooKeeper, Raft/etcd ...)
- Blockchain research and development
  - Revived interest, starting ca. 2015
  - Deployment in practice



Consortium consensus (quorums & BFT)

- Designated set of N validator nodes for consensus
- BFT consensus
  - Tolerates F-out-of-N faulty/ adversarial nodes
  - Generalized quorums
- Send tx to validator nodes
- BFT consensus validates tx, decides, and disseminates



Protocols for BFT consensus

- PBFT = Practical Byzantine Fault Tolerance [CL02]
  - Assumes eventual synchrony (live only in synchronous networks)
  - Extends consensus tolerating crashes (Paxos, Viewstamped Replication, ZooKeeper, Raft/etcd) to Byzantine nodes [C09]
  - BFT-SMaRt toolkit, Hyperledger Fabric, Tendermint and many more
- Practical randomized Byzantine consensus from cryptography [CKS05, CKPS01]
  - No synchrony assumption, fully asynchronous
  - Uses distributed (threshold) cryptography to produce unpredictable randomness
  - SINTRA prototype [C01, CP02], HoneyBadger BFT [MXC+16] and more

Permissioned consensus overview

Which faults are tolerated by a protocol?	Special-node crash	Any $t < n/2$ nodes crash	Special-node subverted	Any $f < n/3$ nodes subverted
Hyperledger Fabric/Kafka	.	✓	.	–
Hyperledger Fabric/PBFT	.	✓	.	✓
Tendermint	.	✓	.	✓
Symbiont/BFT-SMaRt	.	✓	.	✓
R3 Corda/Raft	.	✓	.	–
R3 Corda/BFT-SMaRt	.	✓	.	✓
Iroha/Sumeragi (BChain)	.	✓	.	✓
Kadena/ScalableBFT	?	?	?	?
Chain/Federated Consensus	–	(✓)	–	–
Quorum/QuorumChain	–	(✓)	–	–
Quorum/Raft	.	✓	.	–
MultiChain +	.	✓	.	–
Sawtooth Lake/PoET	⊕	✓	⊕	–
Ripple	⊗	(✓)	⊗	–
Stellar/SCP	?	?	?	?
IOTA Tangle	?	?	?	?

Table 1: Summary of consensus resilience properties, some of which use statically configured nodes with a *special* role. Symbols and notes: '✓' means that the protocol is resilient against the fault and '–' that it is not; '.' states that no such *special node* exists in the protocol; '?' denotes that the properties cannot be assessed due to lack of information; (✓) denotes the crash of *other* nodes, different from the special node; + MultiChain has non-final decisions; ⊕ PoET assumes trusted hardware available from only one vendor; ⊗ Ripple tolerates *one* of the five default Ripple-operated validators (special nodes) to be subverted.

[CV17] C. Cachin, M. Vukolic: Blockchain consensus protocols in the wild, DISC 2017.

https://arxiv.org/abs/1707.01873

Features of BFT consensus

- Well-understood **(+)**
  - Many protocols, research papers (700 protocols … [AGK+15]), textbooks
  - Security proofs and open-source implementations
- Fast **(+)**
  - 1000s or 10'000s of tx/s
  - Latency of seconds
- Decisions are final **(+)**
- Requires all-to-all, $\Omega(N^2)$, communication **(—)**
  - Does not scale to 1000s of nodes
- Relies on identities of nodes **(+ / —)**

Meta-consensus

Two kinds of consensus

- Protocol-level consensus on transactions
  - Automatic and purely mechanical
  - No debates among humans
- Meta-level consensus on protocol
  - Which consensus protocol to run?
  - Social and economic process
  - Much more like diplomacy ... and difficult to automate

Why is a bitcoin worth anything?

Bitcoin

- Anonymous creator
  - Only an informal group of developers and code maintainers
  - Protocol execution controlled by miners
- Debate about block-size limit
  - Bitcoin's block size of 1MB limits throughput to 7tx/s
  - Starting 2015, intensive debates among developers and others to increase block size
  - No meta-consensus ... many developers left Bitcoin

Bitcoin meta-consensus issues

- No consensus either in 2017, but a new method to resolve: fork!
  - Forking always possible when a permissionless blockchain changes its protocol
  - Creates a new currency

- Bitcoin Cash (BCH) forked in July 2017, increasing block size to 8MB
  - Every bitcoin (BTC) also became a bitcoin-cash coin (BCH)
  - Today: 1 BTC = $6580; 1 BCH = $541

- Bitcoin Gold (BTG) forked in Nov. 2017, using a different hash function ("equihash", intended to be memory-hard, preventing mining with ASICs)
  - Today: 1 BTC = $6580; 1 BTG = $25

Ethereum

- Consortium and foundation with a legal status
  - Vitalik Buterin as main public figure
  - Development mostly controlled by the creators with close links to consortium
  - Protocol execution controlled by miners, as in bitcoin

- The DAO and the DAO attack
  - DAO was supposed to be the first decentralized autonomous organization
    - A kind of investment fund controlled only by the blockchain
    - DAO tokens controlled by smart contract on Ethereum
  - Shortly after its start in 2016, an attacker removed ~1/3 of the fund
    - Total worth about $160 M, about $55M at risk
    - DAO tokens were locked up for a period and could not immediately be taken out

Ethereum meta-consensus issues

- Before end of DAO token release period, the Ethereum blockchain forked
  - Buterin and creators decided for a protocol change (hard fork)
  - Buterin posted a blog and most miners followed this
  - Hard fork removed the DAO tokens owned by the attacker

- Ethereum Classic (ETC) forked, not executing the hard fork
  - Its supporters did not want to change the rules
  - ETC continued with the DAO alive and the funds available to attacker
  - Today: 1 ETH = $229; 1 ETC = $11.2

- Soon afterwards the DAO token disappeared completely

Meta-consensus in permissioned blockchains

- Consortium consensus always requires common goal

- A priori agreement on protocols, no issues with meta-consensus

- No public blockchain
  - Many deployments, one for every application
  - No native cryptocurrency (but it could be an application)

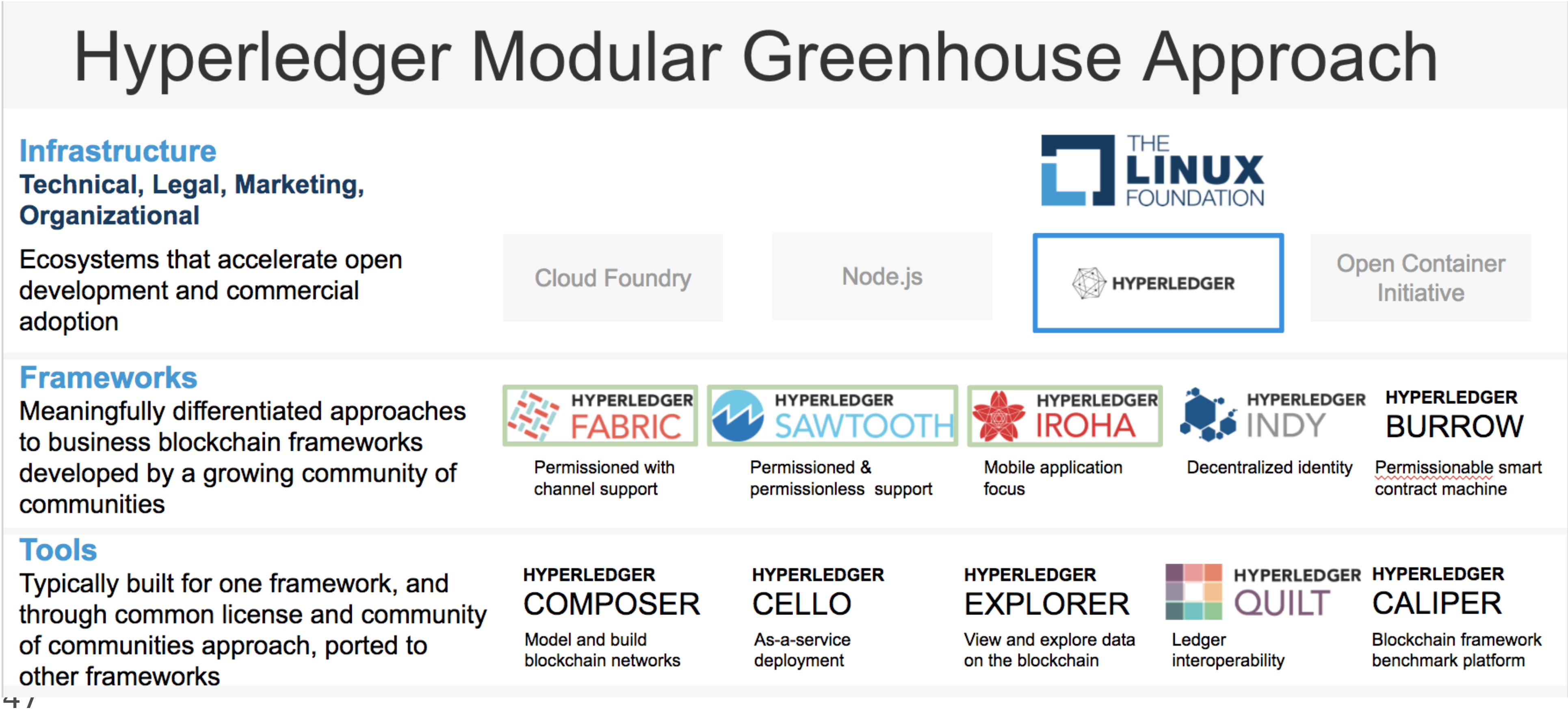
Hyperledger
&
Hyperledger Fabric

Hyperledger

- Hyperledger – www.hyperledger.org
- Global collaboration hosted by the Linux Foundation
  - Advances blockchain technologies for business, neutral, community-driven
  - Started in 2016: Hyperledger unites industry leaders to advance blockchain technology
  - ca. 230 members in May '18
- Develops and promotes blockchain technologies for business
- Today 5 frameworks and 5 tools, hundreds of contributors

- Hyperledger Fabric – github.com/hyperledger/fabric/
  - One blockchain framework of Hyperledger

Hyperledger overview



Hyperledger members

PREMIER MEMBERS

accenture High performance. Delivered.
AIRBUS
AMERICAN EXPRESS
Baidu 百度
CHANGE HEALTHCARE
CISCO
DAIMLER
Digital Asset
DTCC Securing Today. Shaping Tomorrow.
FUJITSU
HITACHI Inspire the Next
IBM
intel
J.P.Morgan
NEC
SAP
TRADESHIFT
万达·飞凡科技 WANDA FFAN TECHNOLOGY

GENERAL MEMBERS

ASSOCIATE MEMBERS

Hyperledger Fabric

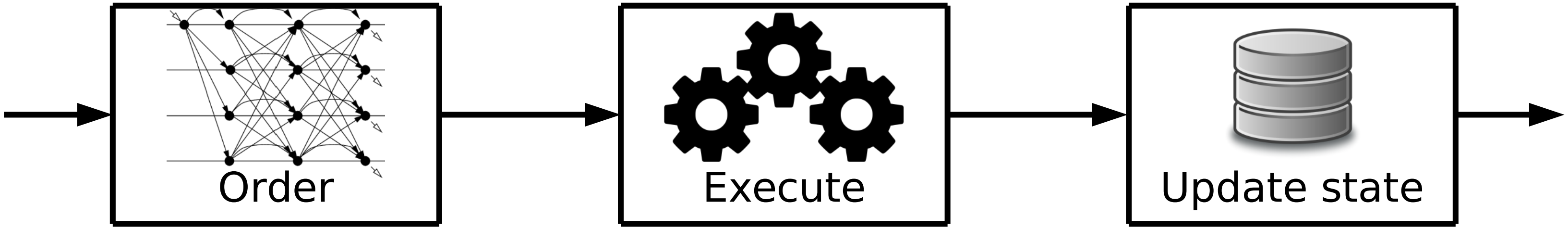
Hyperledger Fabric – An enterprise blockchain platform

- Fabric is a distributed ledger framework for consortium blockchains
  - One of multiple blockchain platforms in the Hyperledger Project (V0.6 in Oct. '16)
  - First active platform in Hyperledger project and production-ready (V1.0 in Jul. '17)

- Developed open-source
  - github.com/hyperledger/fabric
  - Initially developed as *openblockchain* and contributed by IBM
  - Driven IBM, State Street, Digital Asset Holdings, HACERA and others
    - IBM Research – Zurich (Rüschlikon) produced important designs and key components
  - Key technology for IBM's blockchain strategy

- Technical details [Androulaki et al., Eurosys 2018, doi.org/10.1145/3190508.3190538]
  - Modular architecture (e.g., pluggable consensus, cryptography, languages, trust model)
  - Programmable consortium blockchain, implemented in GO
  - Runs smart contracts called "chaincode" within Docker containers

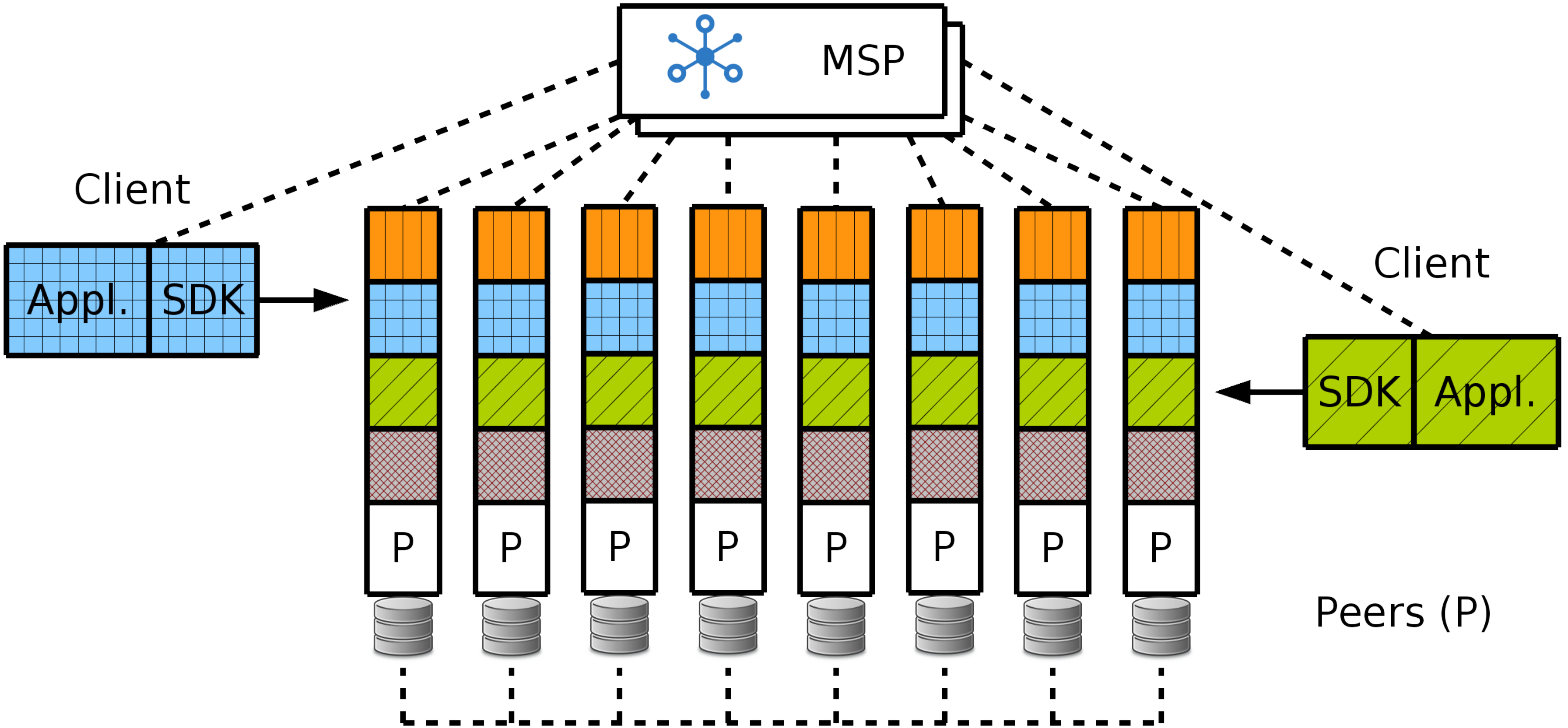
Traditional architecture – Replicated service



- Consensus or atomic broadcast
- Deterministic (!) tx execution
- Persist state on all peers

- All prior BFT systems operate as a replicated state machine [S90]
- All other (permissioned) blockchains operate like this
  - Including Hyperledger Fabric until V0.6

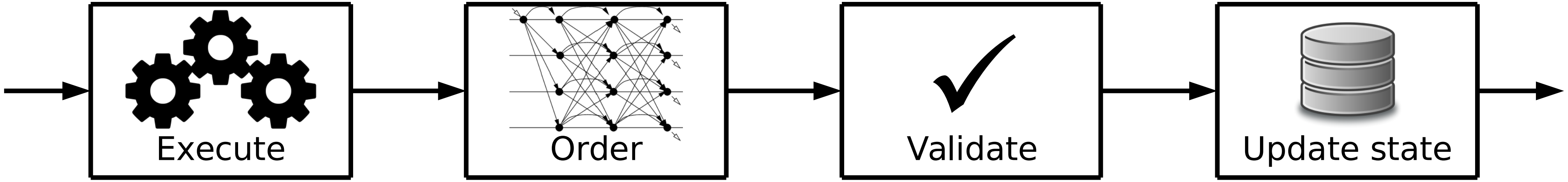
Traditional architecture (including Fabric 0.6)



Issues with the traditional replication design

- Sequential execution
  - Increased latency – or – complex schemes for parallelism
- Operations must be deterministic
  - Difficult to enforce with generic programming language (difficult per se!)
  - Modular filtering of non-deterministic operations is costly [CSV16]
- Trust model is fixed for all applications (smart contracts)
  - Typically some (F+1) validator nodes must agree to result (at least one correct)
  - Fixed to be the same as in consensus protocol
- Privacy is difficult, as data spreads to all nodes
  - All nodes execute all applications
- All these are lessons learned from Hyperledger Fabric, before V0.6

Fabric V1 architecture



- Simulate tx and endorse
- Create rw-set
- Collect endorse-ments

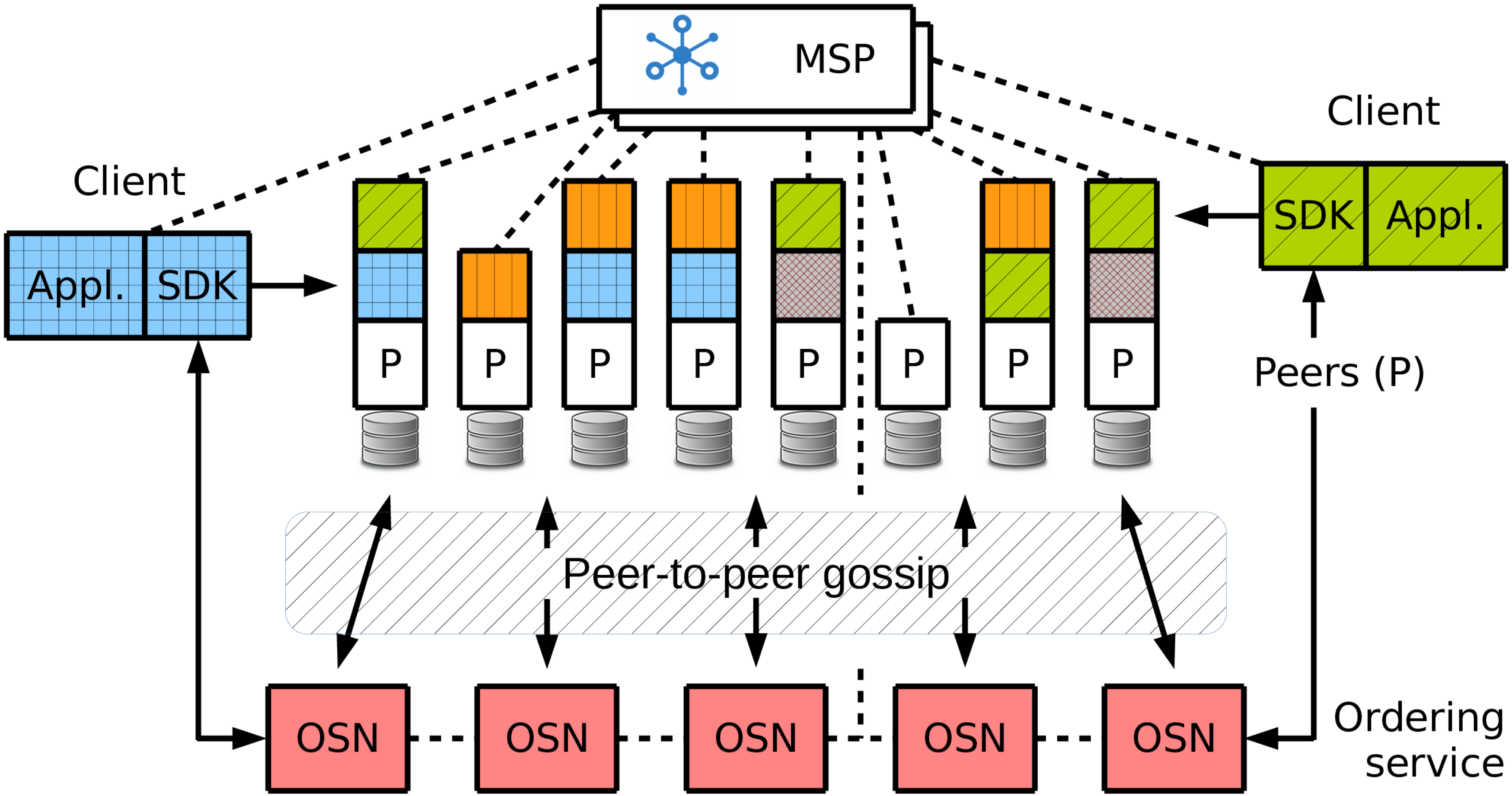
- Order rw-sets
- Atomic broadcast (consensus)
- Stateless ordering service

- Validate endorse-ments & rw-sets
- Eliminate invalid and conflicting tx

- Persist state on all peers

- Includes techniques from databases
- Extends a middleware-replicated database to BFT model

Fabric V1 – Separating endorsement and consensus



Transactions in Fabric V1

- Client (or submitter-peer)
  - Produces a tx (operation) for a chaincode (smart contract)
- Endorsing peer (one or more, according to policy)
  - Executes/simulates tx with chaincode
  - Does not change state, but records accessed state values → readset/writeset (resp., verifies an already given readset/writeset)
  - Endorses tx with a signature on readset/writeset
- Consensus (ordering) service
  - Receives endorsed tx, orders them, and outputs stream of "raw" tx (=atomic broadcast)
- All peers
  - Disseminate tx stream from consensus service with peer-to-peer communication (gossip)
  - Filter out the not properly endorsed tx, according to chaincode endorsement policy
  - Validate state changes from readset/writeset, filter out conflicting tx
  - Apply state changes

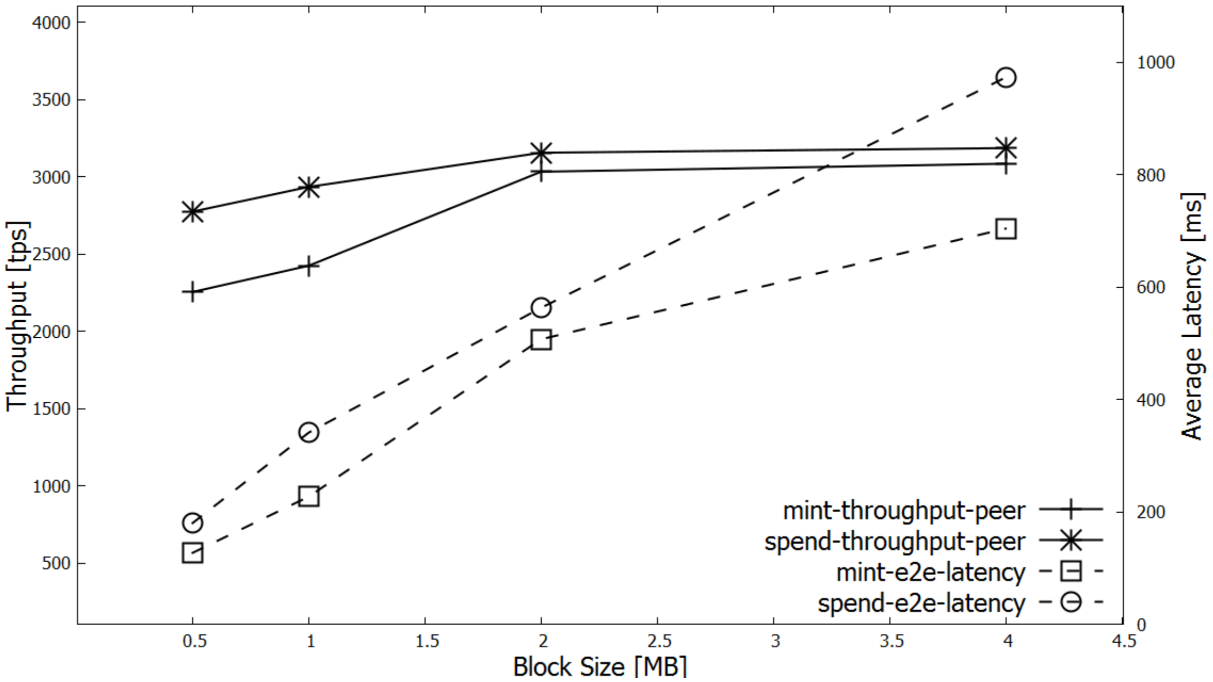
Fabric V1 – Benefits of the separation

- Possible parallel execution increases throughput
  - Off the critical path for consensus protocol
  - Intertwined with trust model
- Non-determinism is confined to chaincode
  - Diverging rw-sets do not properly endorse an operation
  - Turns safety problem (forked peers) for blockchain into a harmless liveness issue
- Flexible trust model
  - Designate different groups of peers responsible for each chaincode
- Private code execution on endorser nodes
  - May encrypt state with chaincode-specific key

Modular consensus in Fabric V1

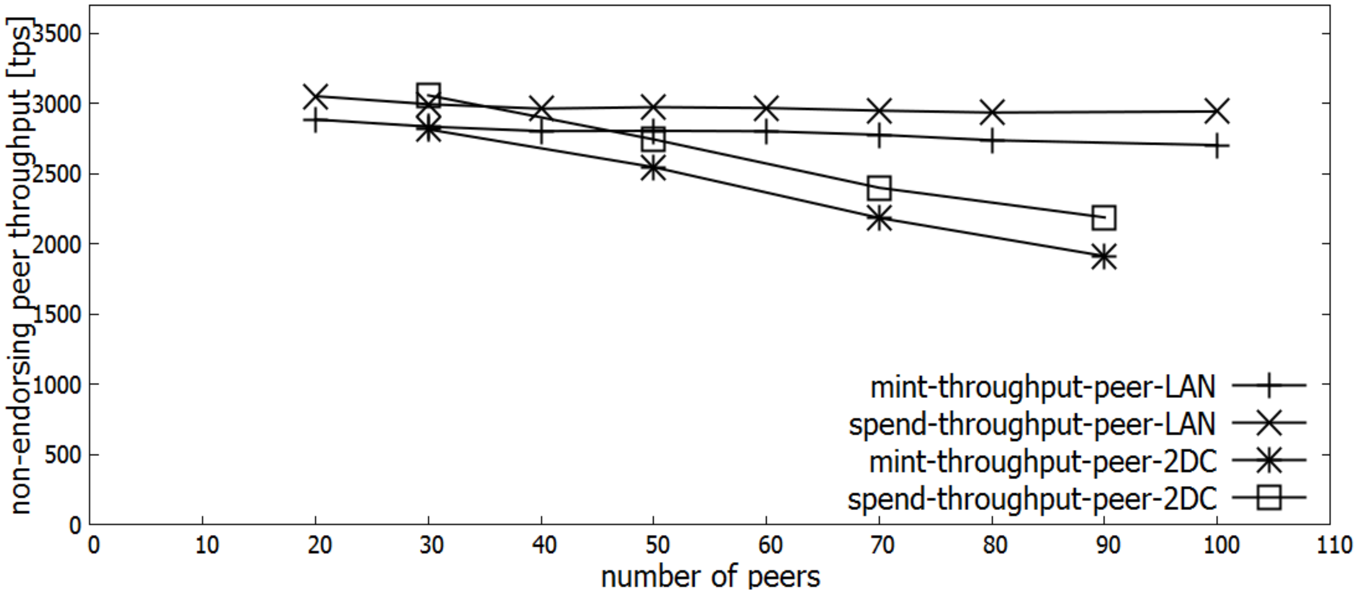
- "Solo orderer"
  - One host only, acting as specification during development (ideal functionality)
- Apache Kafka, a distributed pub/sub streaming platform
  - Tolerates crashes among member nodes, resilience from Apache Zookeeper inside
  - Focus on high throughput
- BFT-SMaRt – Research prototype
  - Tolerates $F < N/3$ Byzantine faulty nodes among N
  - Demonstration of functionality [SBV17]
- SBFT – Simple implementation of PBFT (currently under development)
  - Tolerates $F < N/3$ Byzantine faulty nodes among N
  - Focus on resilience

Fabric V1 – Performance of 'Fabric Coin'



- Impact of block size on throughput and latency
- Bitcoin-like transactions (UTXO): mint and spend
- Cloud deployment on a LAN [Androulaki et al., Eurosys 18]

Fabric V1 – Performance in LAN and WAN



- Impact of number of non-endorsing peers on throughput
- Cloud deployment on LAN
- Deployment on WAN in two data centers (2DC), ordering service in one
  [Androulaki et al., Eurosys 18]

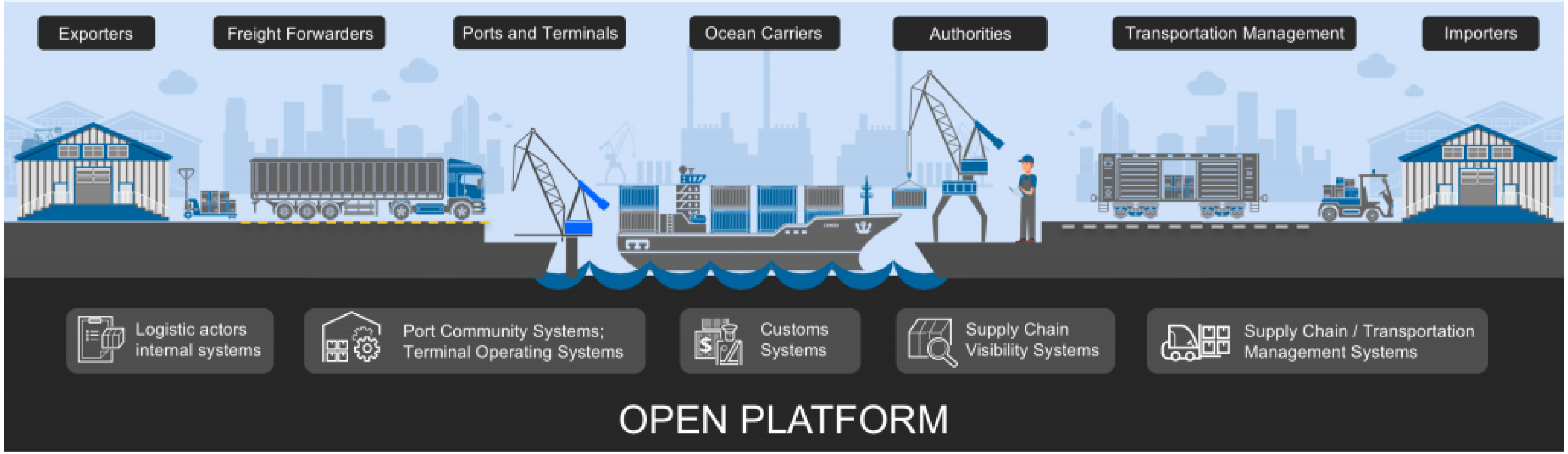
Hyperledger Fabric V1 - Skipped aspects

- Further important components
  – Organizations, Membership service providers (MSP), and Certification Authorities (CA)
  – Chaincode syntax (GO)
  – Gossip protocols for dissemination
  – Channels
  – Data format and ledger design (LevelDB)
- Most important
  – Industrial software engineering
  – Production releases, V1.0 in July '17, latest is V1.2 in July '18; current work is v1.3.0-rc1

Hyperledger Fabric deployment

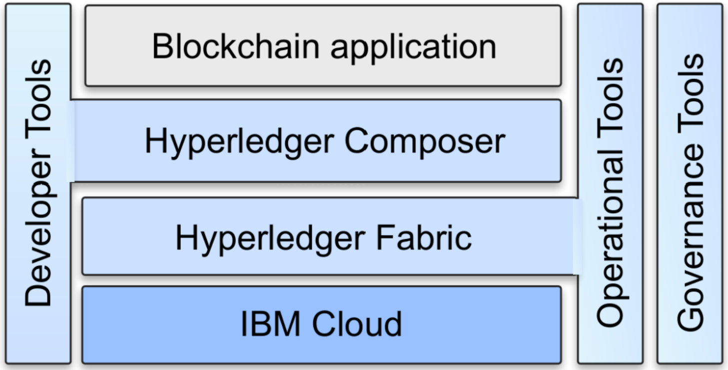
- Fabric is the most prominent and widely used blockchain platform for business
  - Cloud deployment (BaaS) by: IBM, Amazon, Azure, Oracle, Fujitsu, SAP ...
  - Hundreds of prototypes and in-production systems built by IBM alone

- At the core of many new businesses
  - Example: IBM-Maersk joint venture, building a blockchain platform for global trade



IBM Blockchain Platform

- Fully integrated blockchain service platform
  - Developer tools like Hyperledger Composer
  - Hyperledger Fabric distributed ledger technology
  - Governance tools
  - Deployed on IBM Cloud environment



- Provides enterprise-grade security
  - Keys managed by hardware security modules (HSM), certified by NIST at highest level
  - Secure service container (SSC) technology, protecting code and data from admins (such as available with IBM LinuxONE)

Current research directions

- Private transactions in Fabric
  - Privacy-preserving state-based endorsement (Side DB)
  - Share data selectively with channel-private data, ledger stores only hashes

- Zero-knowledge proofs (ZKP)
  - Anonymous authentication with IBM Identity Mixer, anonymity with attribute-based access control
  - Zero-Knowledge Asset Transfer (ZKAT), for privacy-preserving exchange of assets

- Secure smart-contract execution with Intel SGX technology
  - Hardware-based secure enclaves
  - Data and application logic protected from malicious peers
    [Brandenburger et al., arxiv.org/abs/1805.08541]

Conclusion

Conclusion

- Blockchain = Distributing trust over the Internet

- Blockchain enables new trust models
- Distributed computing + cryptography + economics
- We are only at the beginning

- Some links
  - cachin.com/cc
  - www.hyperledger.org
  - www.ibm.com/blockchain/
  - www.zurich.ibm.com/blockchain/
  - ibm.ent.box.com/v/BlockFiles

References

[ABBCCD+18] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro and others. Hyperledger Fabric: A distributed operating system for permissioned blockchains. Eurosys 2018.

[AGK+15] P.-L. Aublin, R. Guerraoui, N. Knezevic, V. Quéma, M. Vukolic: The Next 700 BFT Protocols. ACM TOCS, 32(4), 2015.

[BCG+14] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, M. Virza: Zerocash: Decentralized Anonymous Payments from Bitcoin. IEEE S&P 2014.

[C01] C. Cachin, Distributing trust on the Internet. DSN 2001.

[C09] C. Cachin, Yet another visit to Paxos. IBM Research Report RZ 3754, 2009.

[CGR11] C. Cachin, R. Guerraoui, L. Rodrigues: Introduction to Reliable and Secure Distributed Programming (2. ed.). Springer, 2011.

[CKS05] C. Cachin, K. Kursawe, V. Shoup: Random oracles in Constantinople: Practical asynchronous Byzantine agreement using cryptography. J. Cryptology, 20(3), 2005.

[CKPS01] C. Cachin, K. Kursawe, F. Petzold, V. Shoup: Secure and Efficient Asynchronous Broadcast Protocols. CRYPTO 2001.

References

[CP02] C. Cachin, J. Poritz. Secure intrusion-tolerant replication on the Internet. DSN 2002.

[CSV16] C. Cachin, S. Schubert, M. Vukolic: Non-determinism in Byzantine Fault-Tolerant Replication. OPODIS 2016.

[CV17] C. Cachin, M. Vukolic: Blockchain Consensus Protocols in the Wild, DISC 2017.

[CL02] M. Castro, B. Liskov: Practical Byzantine Fault Tolerance and Proactive Recovery. ACM TOCS, 20(4), 2002.

[DSW16] C. Decker, J. Seidel, R. Wattenhofer: Bitcoin Meets Strong Consistency. ICDCN 2016.

[EGS+16] I. Eyal, A. Gencer, E.G. Sirer, R. van Renesse: Bitcoin-NG: A Scalable Blockchain Protocol. NSDI 2016.

[KMS+16] A. Kosba, A. Miller, E. Shi, Z. Wen, C. Papamanthou: Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts. IEEE S&P 2016.

[KJP10] B. Kemme, R. Jiménez-Peris, M. Patiño-Martínez: Database Replication. Morgan & Claypool, 2010.

[LNB+15] L. Luu, V. Narayanan, K. Baweja, C. Zheng, S. Gilbert, P. Saxena: A Secure Sharding Protocol For Open Blockchains. ACM CCS 2016.

[LSP82] L. Lamport, R. Shostak, M. Pease: The {Byzantine} Generals Problem. ACM TOPLAS, 4(3), 1982.

References

[MR98] D. Malkhi, M. Reiter: Byzantine Quorum Systems. Distributed Computing, 1998.

[MXC+16] A. Miller, Y. Xia, K. Croman, E. Shi, D. Song: The Honey Badger of BFT Protocols. ACM CCS 2016.

[PS17] R. Pass, E. Shi: Feasibilities and Infeasibilities for Achieving Responsiveness in Permissionless Consensus (Hybrid Consensus), DISC 2017.

[S90] F. Schneider: Implementing Fault-Tolerant Services using the State Machine Approach: A Tutorial. ACM Comput. Surveys, 1990.

[SBV17] J. Sousa, A. Bessani, M. Vukolic. A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform. e-print, arXiv:1709.06921 [cs.CR], 2017, and DSN 2018.

[V16] M. Vukolic: The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication. LNCS 9591, Proc. iNetSeC 2015.

Hyperledger Fabric references

- www.hyperledger.org/projects/fabric
- Architecture of V1 – Eurosys 2018, https://doi.org/10.1145/3190508.3190538
- Designs – wiki.hyperledger.org/projects/fabric/design-docs
- Docs – hyperledger-fabric.readthedocs.io/en/latest/
- Code – github.com/hyperledger/fabric
- Chat – chat.hyperledger.org, all channels like #fabric-*