

Distributing Trust with Blockchains

Christian Cachin

IBM Research – Zurich

July 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

Closing Price OHLC

1h 12h 1d 1w 1m 3m 1y All

Jul 18, 2013

to

Jul 5, 2018

Export

\$15000

\$10000

\$5000

\$0

2014

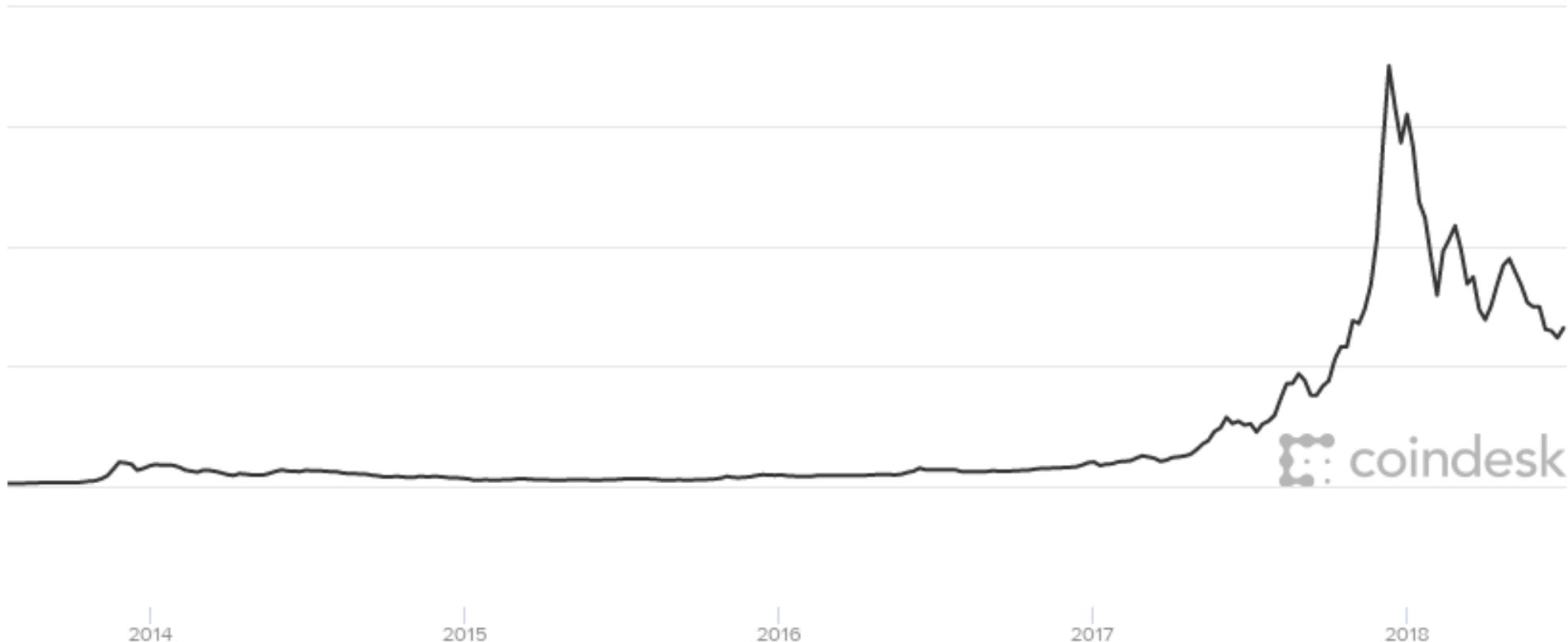
2015

2016

2017

2018

coindesk



In cryptography we trust (?)



What is a blockchain?

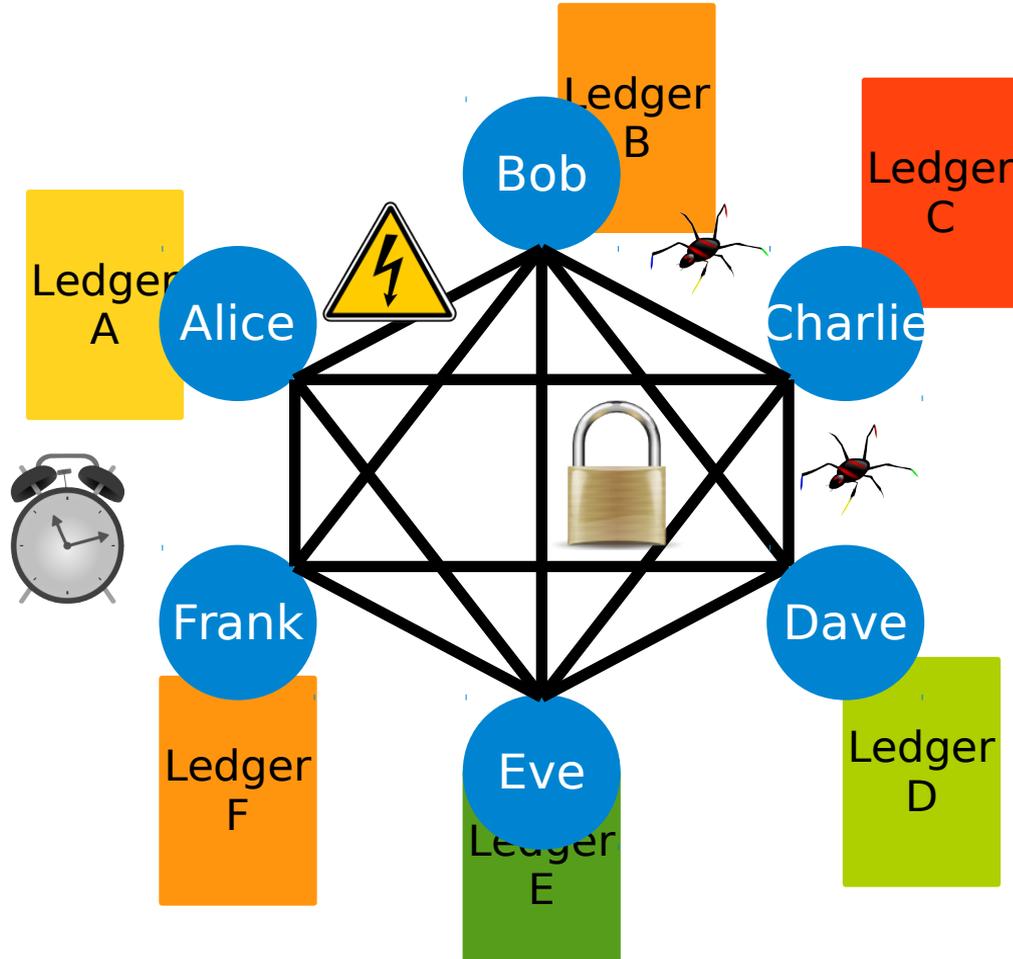
Ledger

Datum		Entnahme und Abhebungen <i>R.M.</i> <i>₪</i>	Lieferungen, Einzahlungen, Gutschriften <i>R.M.</i> <i>₪</i>	Bestand der Schuld <i>R.M.</i> <i>₪</i>	Bestand des Guthabens <i>R.M.</i> <i>₪</i>
1942	<i>Übertrag:</i>				
Aug. 12.	An 2.000 kg Kartoffeln	✓ 54.-		✓ 1.109,81	
23.	" 2.100 kg Grunderwehl	✓ 102,90		✓ 1.212,71	
Ok. 6.	" 280 kg Tomaten	✓ 37,59		✓ 1.350,30	
9.	" 10 Mel. Kalbblut	✓ 6,50		✓ 1.356,80	
14.	An 1.500 kg Erdbeeren	✓ 46,50		✓ 1.310,30	
21.	" 500 kg Zuckerrüben	✓ 72,50		✓ 1.237,80	
Nov. 5.	Per 1.250 kg Kartoffeln		✓ 67,50	✓ 1.305,30	
26.	" 3.750 kg Roggen		✓ 678,25	✓ 627,05	
Dez. 14.	An 1.500 kg Erdbeeren	✓ 46,50		✓ 580,55	
18.	" 2.500 kg Äpfel	✓ 154,50		✓ 735,05	
31.	Zinsen gg. per 31.12.42	✓ 30,05		✓ 765,10	✓ 52
1943					
Jan. 4.	An 37,5 kg Getreide				
	50 kg Meizen-Getreide	✓ 4,83			
4.	" 1.200 kg Erdbeeren	✓ 132,-		✓ 1.057,43	
26.	" 525 kg Erdbeeren				
	50 kg Meizen, 50 kg Weizen				
	50 kg Weizen, 50 kg Roggen	✓ 135,48		✓ 1.192,91	

- ▶ 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

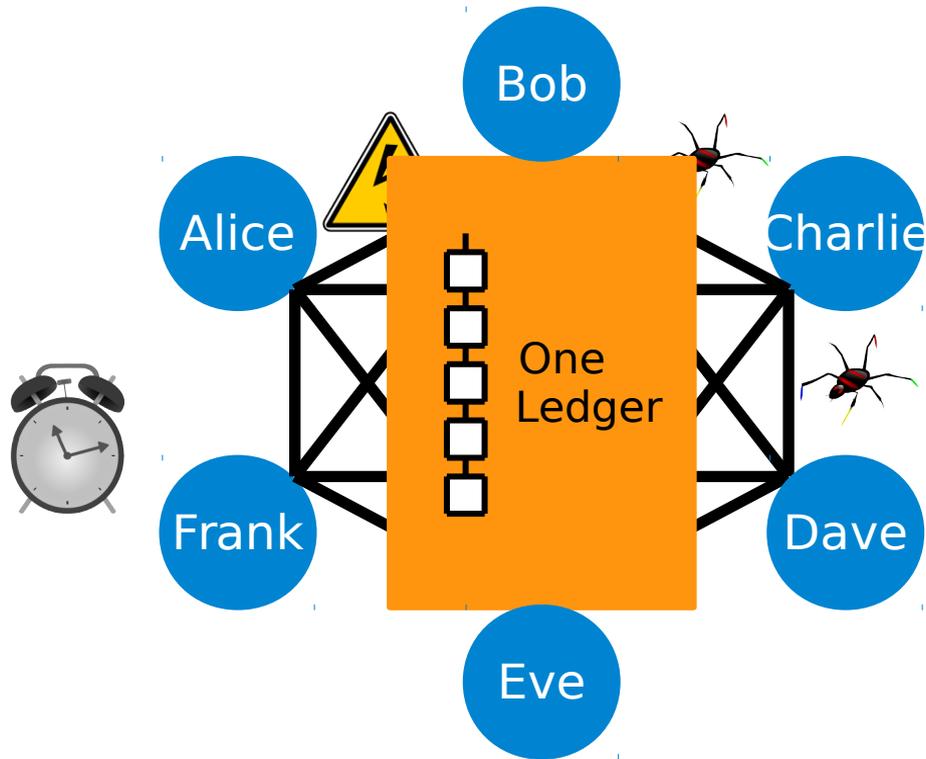


Multiple ledgers



- ▶ 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

Again – What is a blockchain?

Distributing Trust on the Internet

Christian Cachin

IBM Research
Zurich Research Laboratory
CH-8803 Rüschlikon, Switzerland
cca@zurich.ibm.com

March 8, 2001

Abstract

This paper describes an architecture for secure and fault-tolerant service replication in an asynchronous network such as the Internet, where a malicious adversary may corrupt some servers and control the network. It relies on recent protocols for randomized Byzantine agreement and for atomic broadcast, which exploit concepts from threshold cryptography. The model and its assumptions are discussed in detail and compared to related work from the last decade in the first part of this work, and an overview of the broadcast protocols in the architecture is provided. The standard approach in fault-tolerant distributed systems is to assume that at most a certain fraction of servers fails. In the second part, novel general failure patterns and corresponding protocols are introduced. They allow for realistic modeling of real-world trust assumptions, beyond (weighted) threshold models. Finally, the application of our architecture to trusted services is discussed.

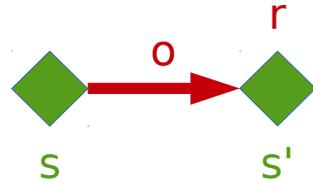


A state machine

► Functionality F

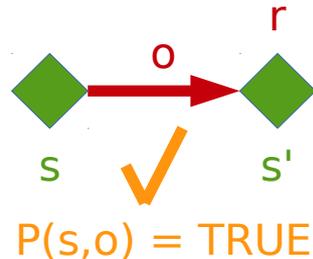
- Operation o transforms a state s to new state s' and may generate a response r

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



► Validation condition

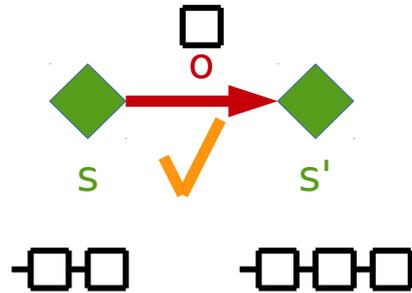
- Operation needs to be **valid**, in current state, according to a predicate $P()$



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**

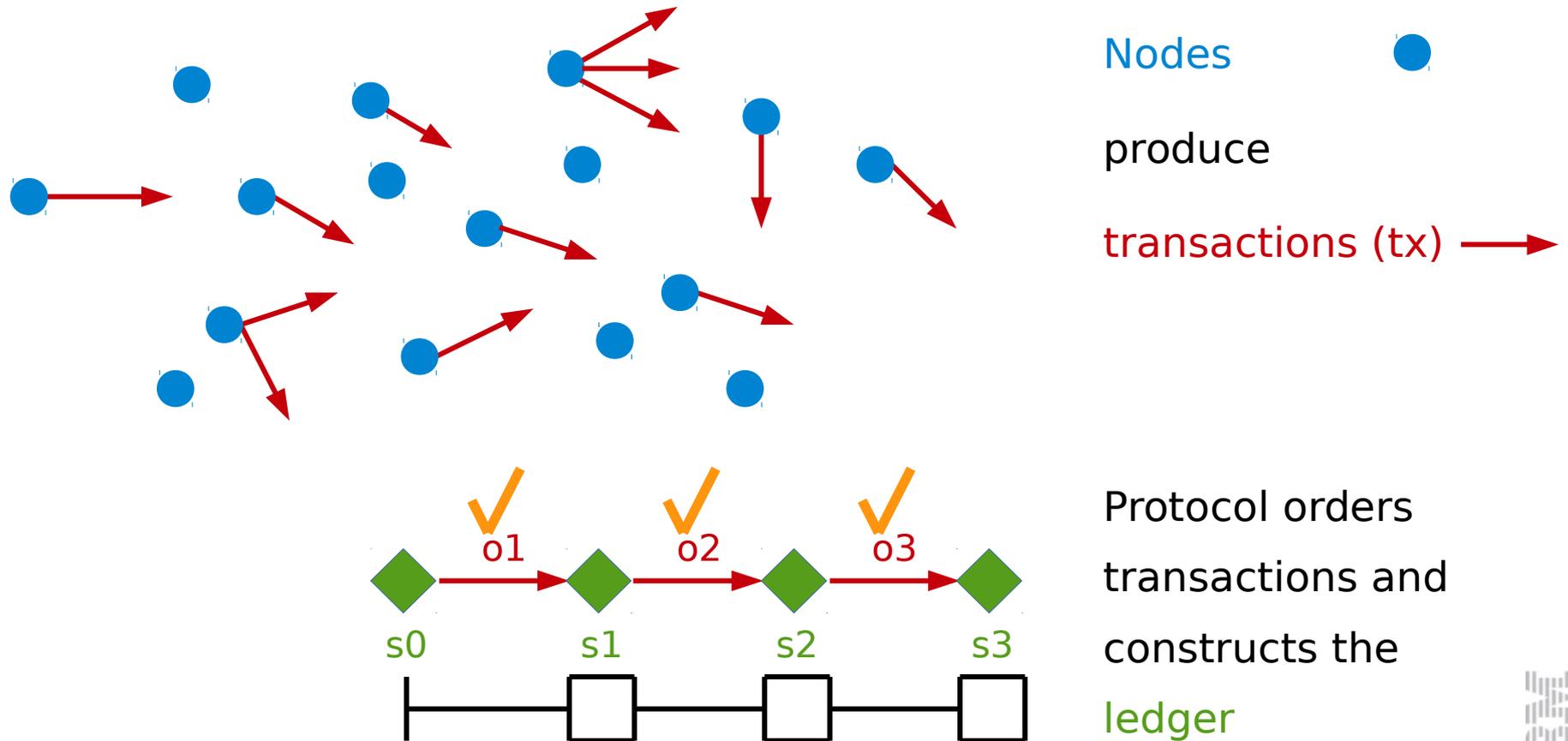
$$h_t \leftarrow \text{Hash}([tx_1, tx_2, \dots] \parallel h_{t-1} \parallel t) .$$

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



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

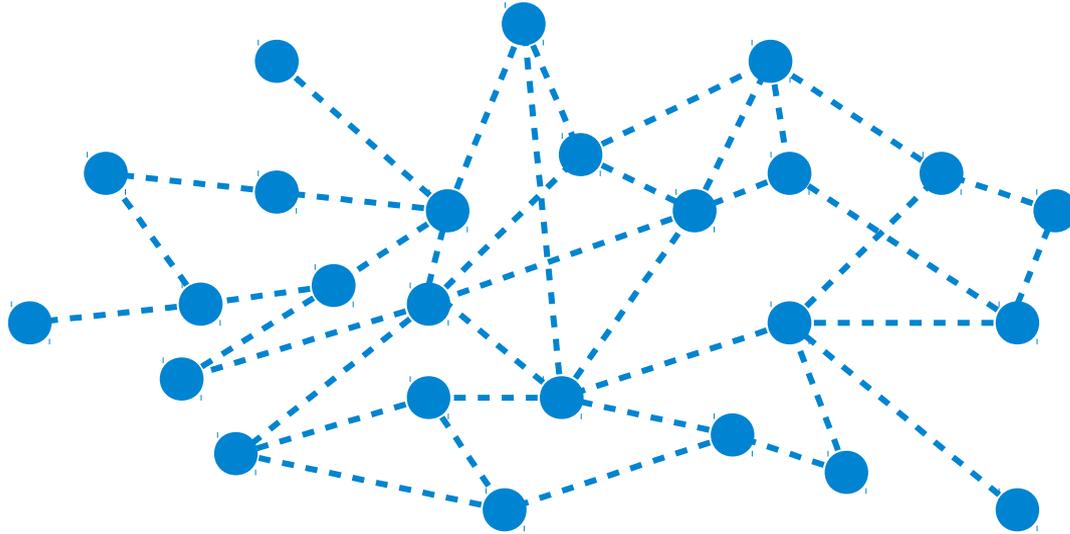
Types of blockchain consensus

- ▶ **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

▶ All nodes prepare blocks

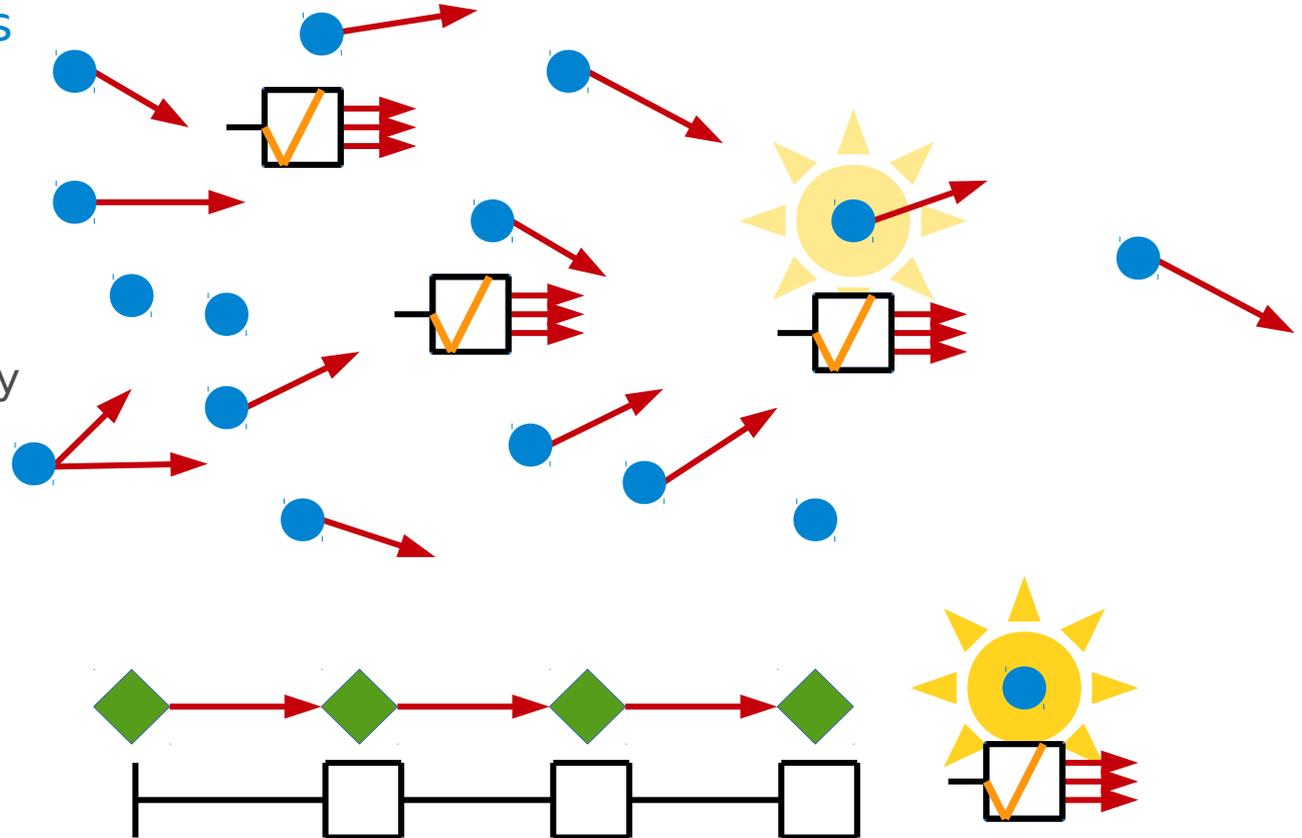
- List of transactions (tx)
- All tx valid

▶ Lottery race

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

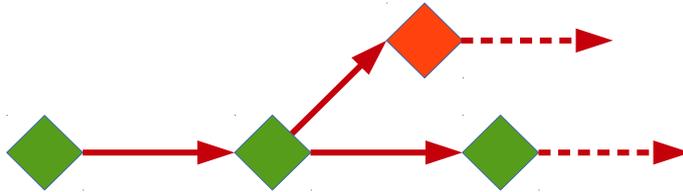
▶ All nodes verify and validate new block

- "Longest" chain wins



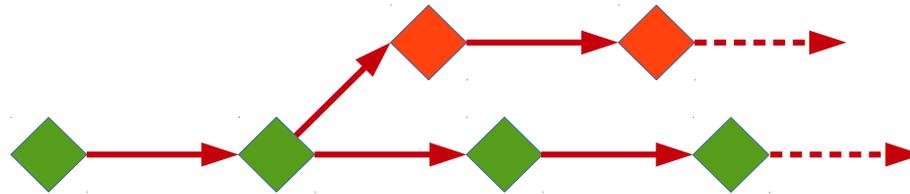
How does proof-of-work ensure consistency?

- ▶ 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



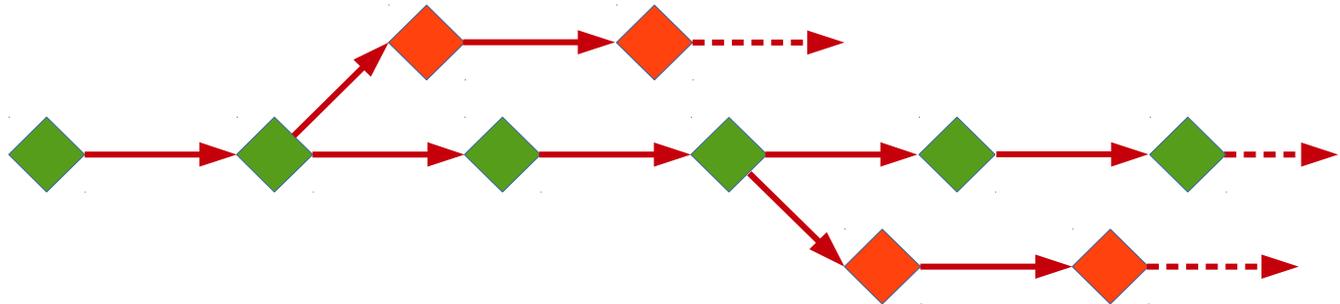
- ▶ "Longest" chain wins

- ▶ 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

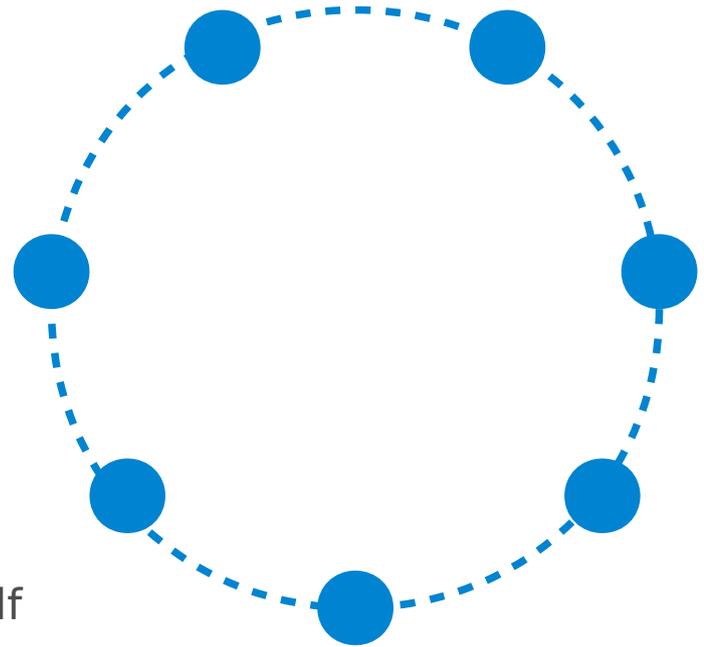
- ▶ Survives censorship and suppression (+ / -)
 - No identities, no counting of nodes
 - Give incentive to participate with mining reward
- ▶ Scales to 1000s of nodes (+)
- ▶ High latency (minutes or more), and decisions are never final (-)
- ▶ Requires proof-of-work (PoW) (-)
 - Majority of hashing 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

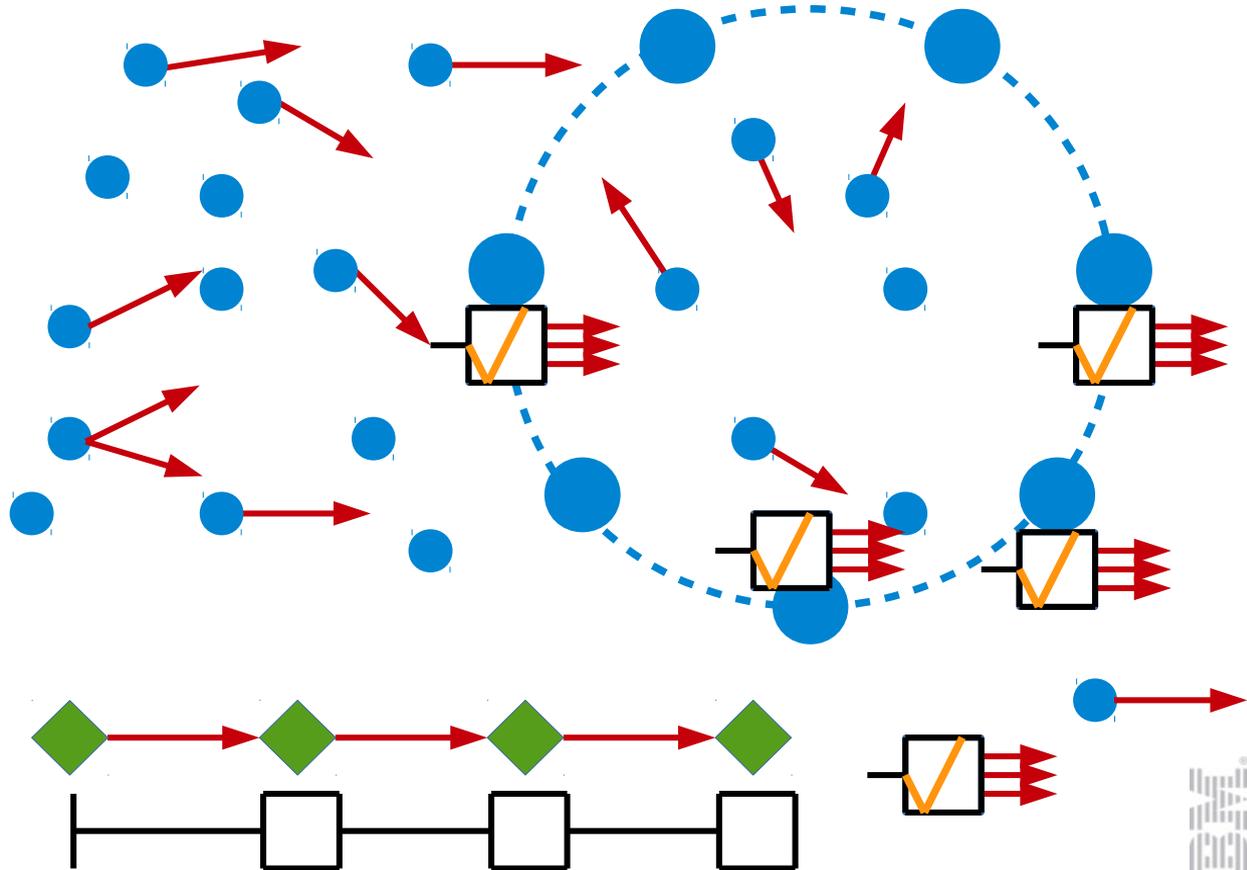
Consortium – Permissioned – BFT

- ▶ 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



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



Many protocols for permissioned consensus

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	?	?	?	?

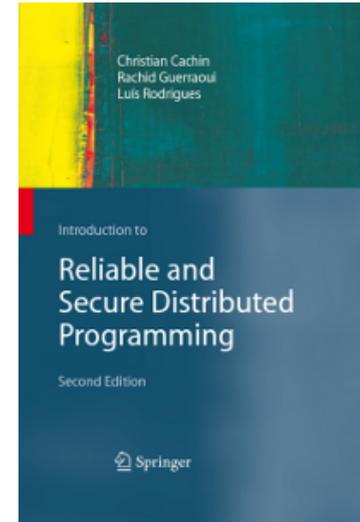
[CV17] C. Cachin, M. Vukolic: Blockchain consensus protocols in the wild, DISC 2017.

<https://arxiv.org/abs/1707.01873>

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.

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?



Dilbert.com DilbertCartoonist@gmail.com



7-22-13 ©2013 Scott Adams, Inc./Dist. by Universal Uclick



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 = \$6750; 1 BCH = \$764
- ▶ **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 = \$6750; 1 BTG = \$30



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 = \$482; 1 ETC = \$17.7
- ▶ 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

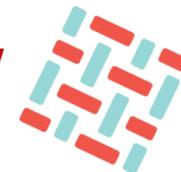
Hyperledger



HYPERLEDGER

- ▶ [Hyperledger](http://www.hyperledger.org) – 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](https://github.com/hyperledger/fabric/) – github.com/hyperledger/fabric/
 - One blockchain framework of Hyperledger



**HYPERLEDGER
FABRIC**



Hyperledger overview

Hyperledger Modular Greenhouse Approach

Infrastructure

Technical, Legal, Marketing, Organizational

Ecosystems that accelerate open development and commercial adoption

Cloud Foundry

Node.js



Open Container Initiative

Frameworks

Meaningfully differentiated approaches to business blockchain frameworks developed by a growing community of communities



Permissioned with channel support



Permissioned & permissionless support



Mobile application focus



Decentralized identity



Permissionable smart contract machine

Tools

Typically built for one framework, and through common license and community of communities approach, ported to other frameworks



Model and build blockchain networks



As-a-service deployment



View and explore data on the blockchain



Ledger interoperability



Blockchain framework benchmark platform



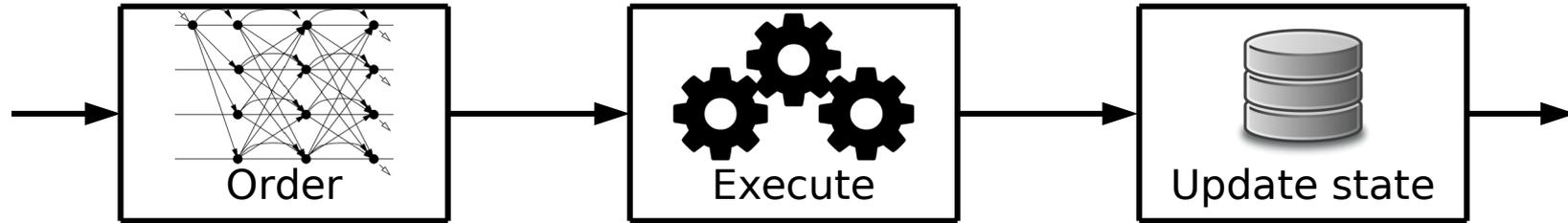
Hyperledger Fabric

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 [ABBCCD+18]**
 - 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



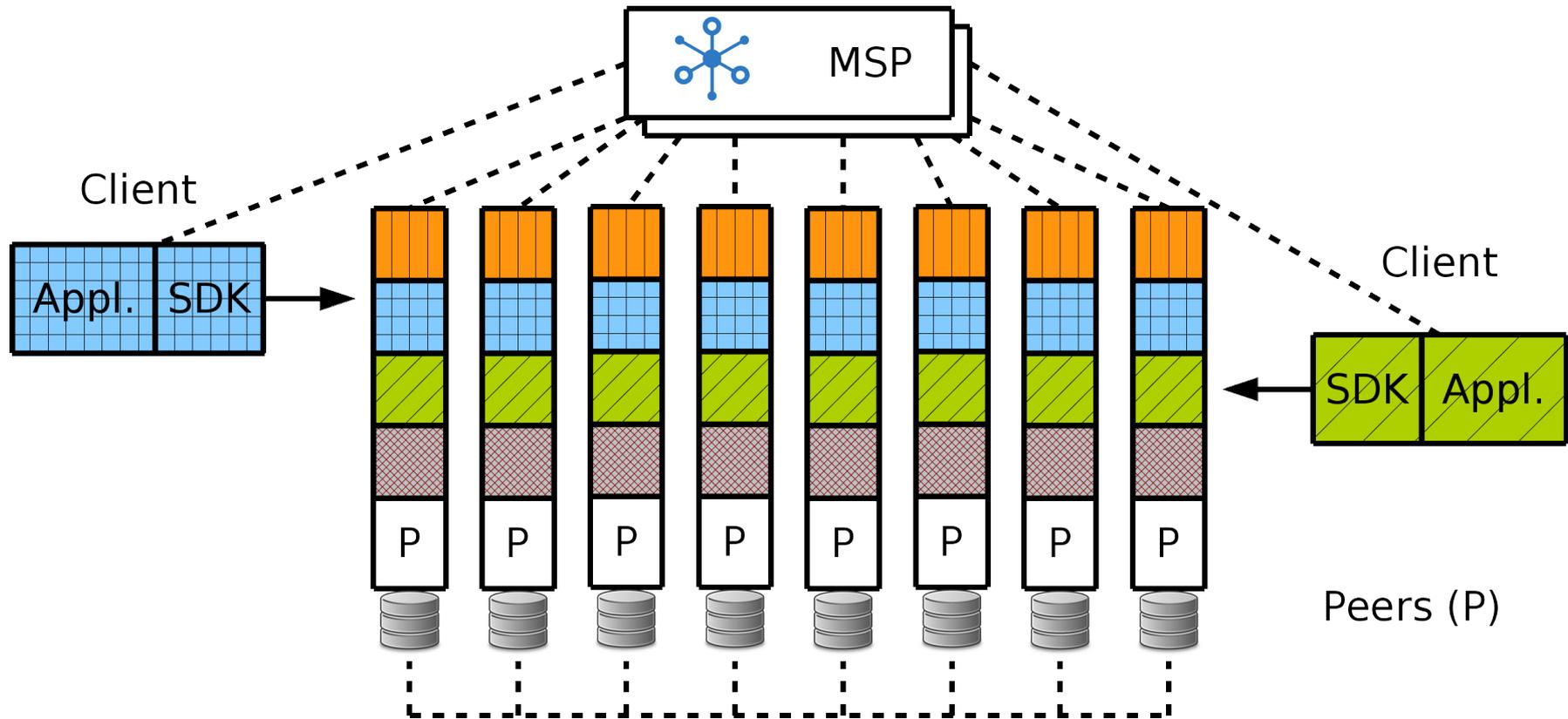
- Consensus or atomic broadcast

- 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)



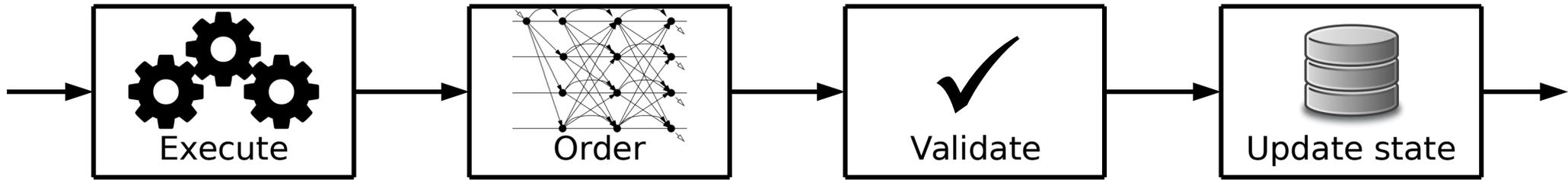
Issues with the traditional replication design

- ▶ **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

49 **All these are lessons learned from Hyperledger Fabric, before V0.6**



Fabric V1 architecture



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

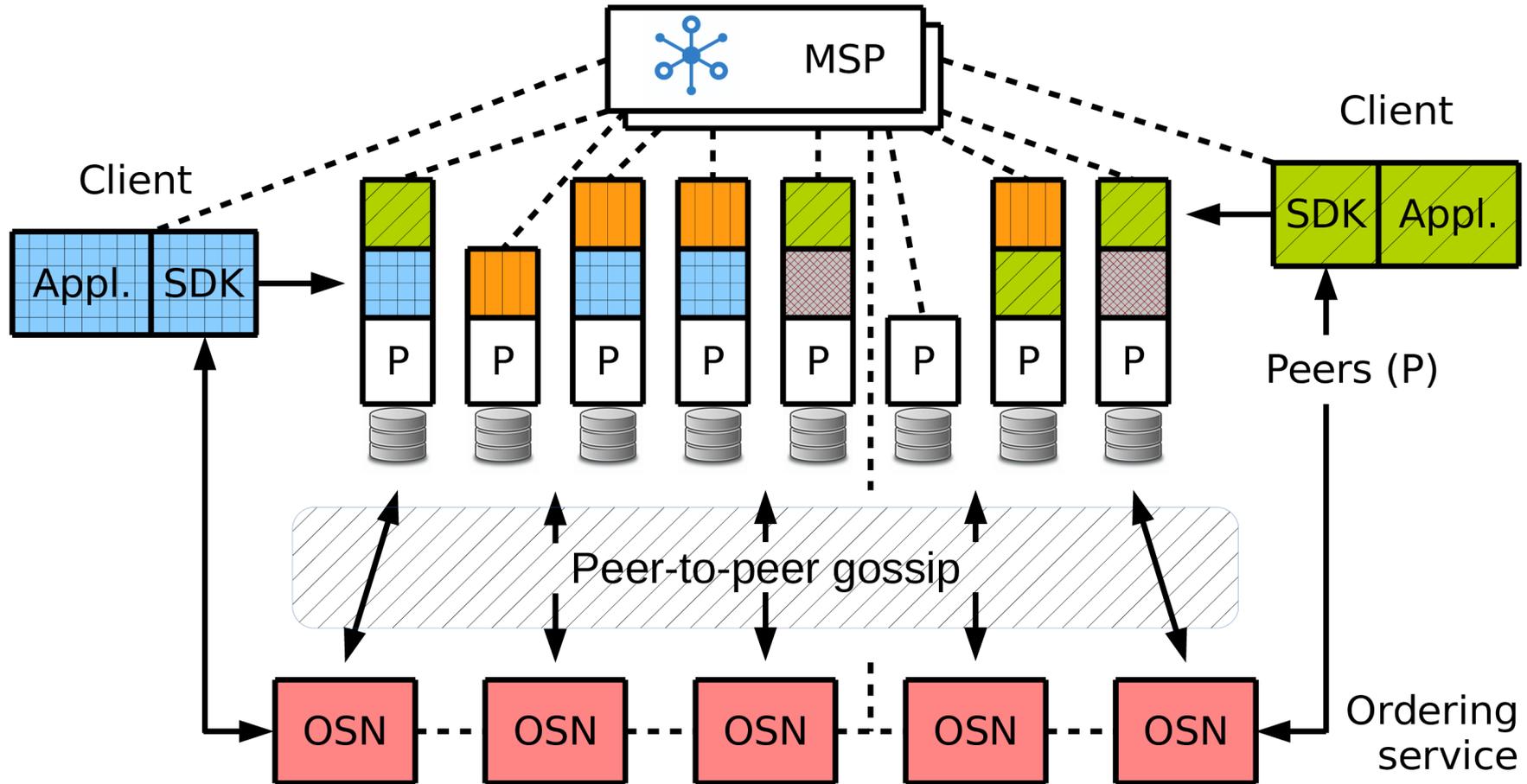
- Order rw-sets
- Atomic broadcast (consensus)
- Stateless ordering service

- Validate endorsements & 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



Fabric V1 details

- ▶ **Separate the functions of nodes into endorsers and consensus nodes**
 - Every chaincode may have different endorsers
 - Endorsers have state, run tx, and validate tx for their chaincode
 - Chaincode specifies endorsement policy
 - Consensus nodes order endorsed and already-validated tx
 - All peers apply all state changes in order, only for properly endorsed tx
- ▶ **Functions as replicated database maintained by peers [KJP10]**
 - Replication via (BFT) atomic broadcast in consensus
 - Endorsement protects against unauthorized updates
- ▶ Scales better – only few nodes execute, independent computations in parallel
- ▶ Permits **some confidential data** on blockchain via partitioning state
 - 52 Data seen only by endorsers assigned to run that chaincode



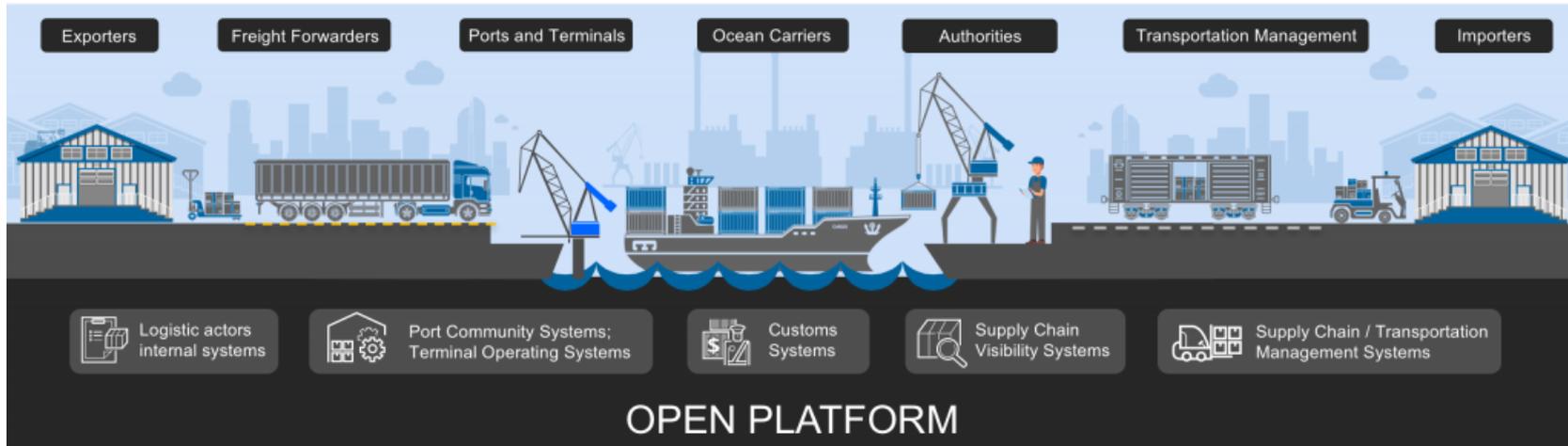
Modular consensus in Fabric V1

- ▶ "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



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



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-*