Cryptography and Protocols in Hyperledger Fabric

Elli Androulaki, Christian Cachin, Angelo De Caro, Andreas Kind, Mike Osborne, Simon Schubert, Alessandro Sorniotti, Marko Vukolic and many more

IBM Research – Zurich

Real-World Cryptography Conference 2017

Hyperledger Fabric

§ Implementation of a blockchain platform [for the enterprise]

§ Uses familiar and proven technologies

§ Modular architecture

§ Container technology for smart contracts in any modern language

§ Developed open source & collaboratively in the Hyperledger Project

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"

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

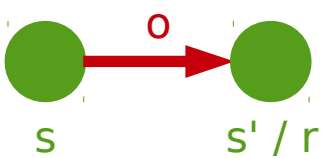
What is a blockchain?

A state machine

§ Functionality F

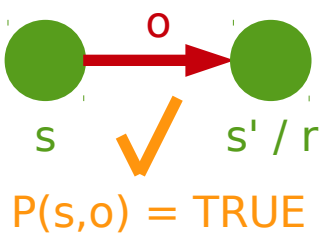
– Operation o transforms a state s to new state s' and may generate a response r

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



§ Validation condition

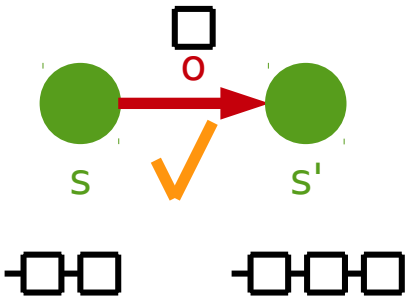
– Operation needs to be valid, in current state, according to a predicate P()



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

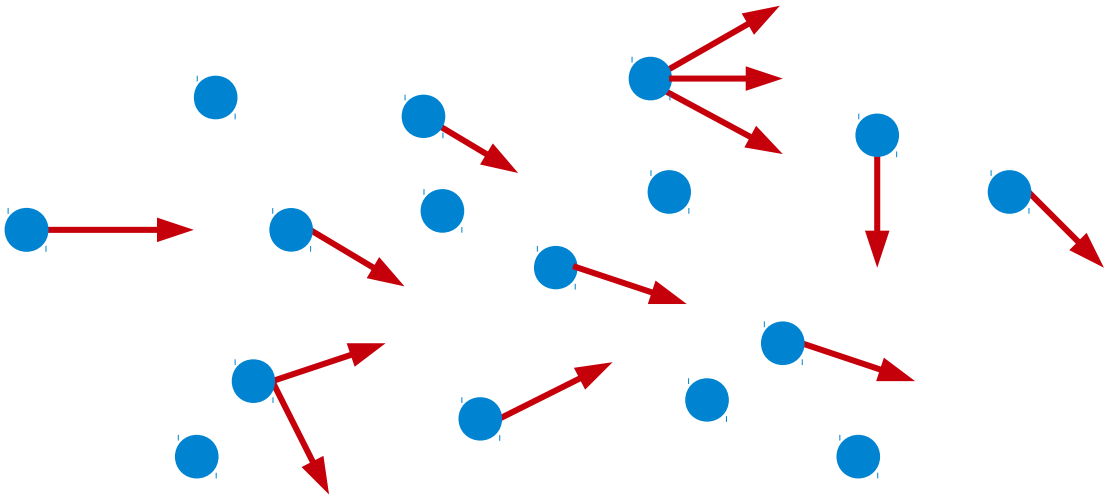
§ Bitcoins are unforgeable bitstrings
  – "Mined" by the protocol itself (see later)

§ Digital signature keys (ECDSA) own and transfer bitcoins
  – Owners are pseudonymous, e.g., 3JDs4hAZeKE7vER2YvmH4yTMDEfoA1trnC

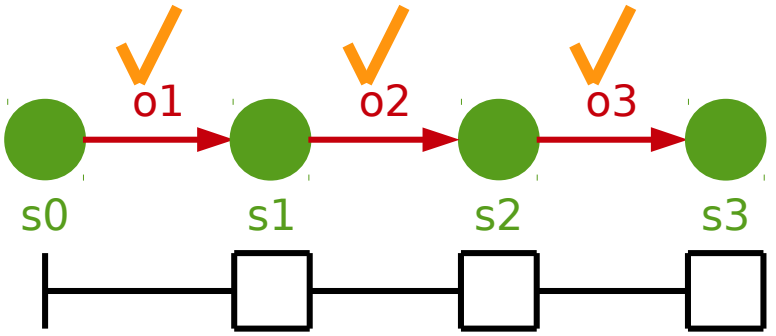
§ Every transaction transfers a bitcoin (fraction) from current to next owner
  – "This bitcoin now belongs to 3JDs..." signed by the key of current owner
  – (Flow linkable by protocol, and not anonymous when converted to real-world assets)

§ Validation is based on the global history of past transactions
  – Signer has received the bitcoin before
  – Signer has not yet spent the bitcoin

Distributed p2p protocol to create a ledger



Nodes
produce
transactions



Nodes run a protocol to construct 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

Three kinds of blockchain consensus

§ Decentralized / permissionless
  – Bitcoin

§ Somewhat decentralized – skipped here
  – Ripple, Stellar

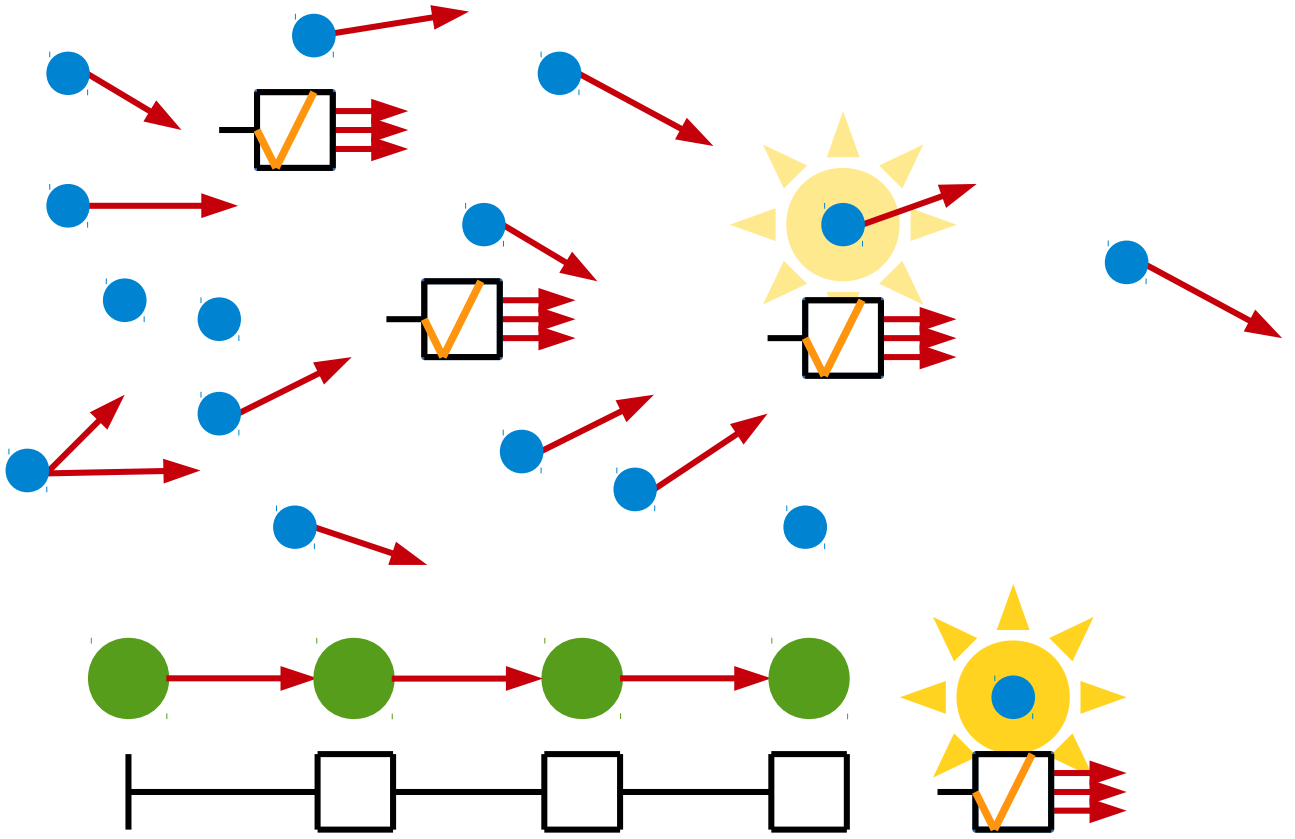
§ Consortium / permissioned
  – BFT (Byzantine fault tolerance) consensus

Decentralized – Nakamoto consensus/Bitcoin

§ Nodes prepare blocks
- List of transactions (tx)
- All tx valid

§ Lottery race
- Solves a hard puzzle
- Selects a random winner/leader
- Winner's operation/ block is executed and "mines" a coin

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



Decentralized = permissionless

§ Survives censorship and suppression
- No central entity

§ Nakamoto consensus requires proof-of-work (PoW)
- Original intent: one CPU, one vote
- Majority of hashing power controls network
- Gives economic incentive to participate (solution to PoW is a newly "mined" Bitcoin)

§ Today, total hashing work consumes a lot of electricity
- Estimates vary, 250-500MW, from a major city to a small country ...

§ Protocol features
- Stability is a tradeoff between dissemination of new block (10s-20s) and mining rate (new block on average every 10min)
- Decisions are not final ("wait until chain is 6 blocks longer before a tx is confirmed")

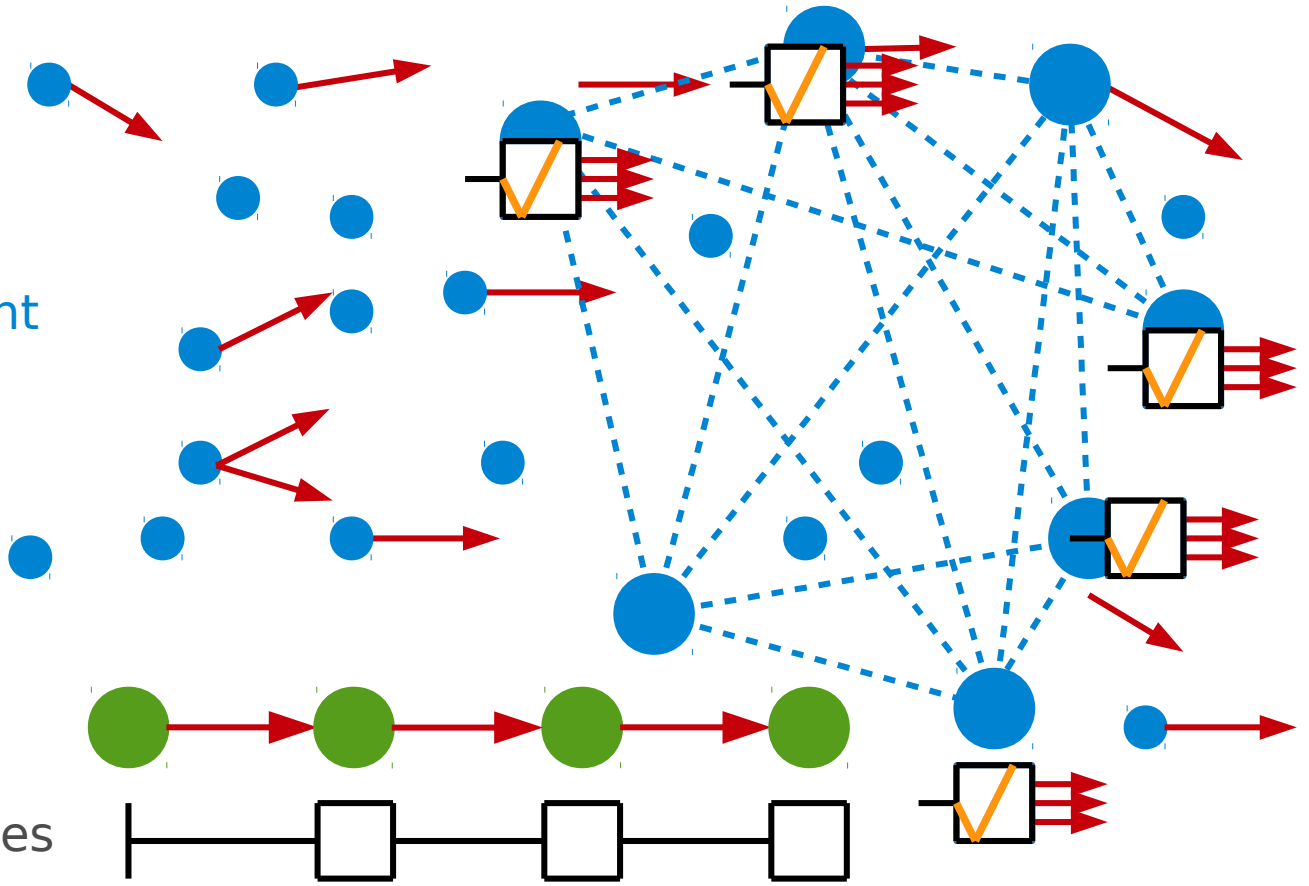
Consortium consensus (BFT, Hyperledger)

§ Designated set of homogeneous validator nodes

§ BFT/Byzantine agreement
- Tolerates f-out-of-n faulty/ adversarial nodes
- Generalized quorums

§ Tx sent to consensus nodes

§ Consensus validates tx, decides, and disseminates result



Consortium consensus = permissioned

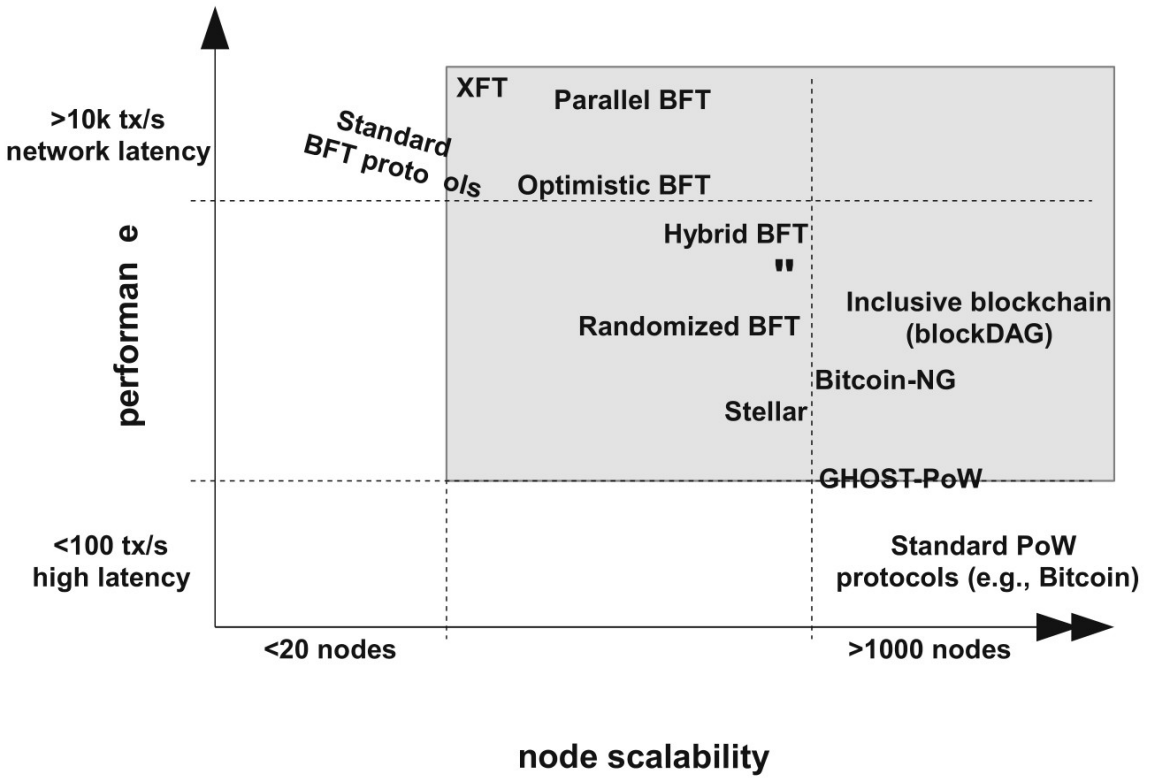
§ Used by Hyperledger Fabric and many other platforms

§ Central entity controls group membership (PKI)
- Membership may be decided inline dynamically

§ Features
- BFT and consensus are very-well understood problems
  - Clear assumptions and top-down design
  - 700 protocols and counting [AGK+15]
  - Textbooks [CGR11]
  - Open-source implementations (BFT-SMaRT)
- Many systems already provide crash tolerant consensus (Chubby, Zookeeper, etcd ...)
- Typically needs $\Omega(n^2)$ communication (OK for 10-100 nodes, not > 1000s)

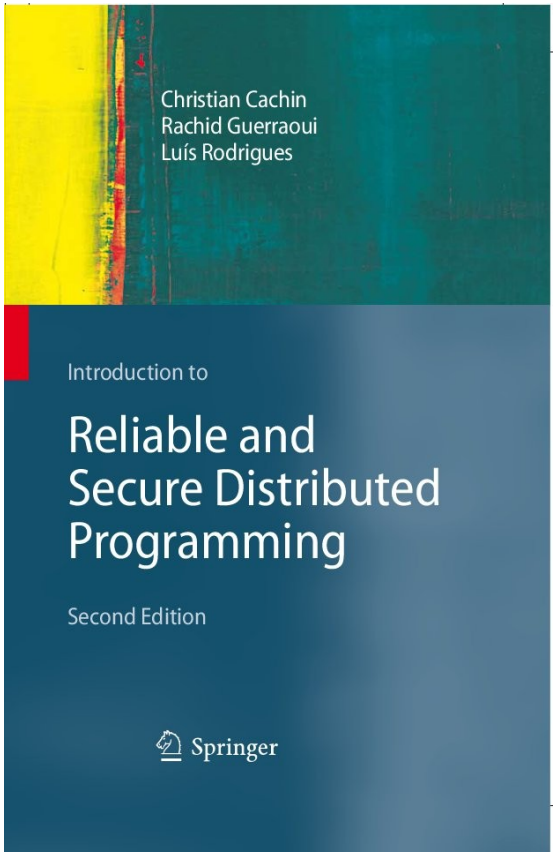
§ Revival of research in BFT consensus protocols

Scalability–performance tradeoff



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

More about consensus protocols



Introduction to Reliable and Secure Distributed Programming

C. Cachin, R. Guerraoui, L. Rodrigues

2nd ed., Springer, 2011

www.distributedprogramming.net

Validation

Validation of transactions – PoW protocols

§ Recall validation predicate P on state s and operation o: P(s, o) ✓

§ When constructing a block, the node
- Validates all contained tx
- Decides on an ordering within block

§ When a new block is propagated, all nodes must validate the block and its tx
- Simple for Bitcoin – verify digital signatures and that coins are unspent
- More complex and costly for Ethereum – re-run all the smart-contract code

§ Validation can be expensive
- Bitcoin blockchain contains the log of all tx – 97GB as of 1/2017 (https://blockchain.info/charts/blocks-size)

Validation of transactions – BFT protocols

§ Properties of ordinary Byzantine consensus

- Weak Validity: Suppose all nodes are correct: if all propose v, then a node may only decide v; if a node decides v, then v was proposed by some node.
- Agreement: No two correct nodes decide differently.
- Termination: Every correct node eventually decides.

§ Standard validity notions do not connect to the application!

§ Need validity anchored at external predicate [CKPS01]

- External validity: Given predicate P, known to every node, if a correct node decides v, then P(v); additionally, v was proposed by some node.

- Can be implemented with digital signatures on input tx

Public validation vs. private state

§ So far everything on blockchain is public – where is privacy?

§ Use cryptography – keep state "off-chain" and produce verifiable tx

- In Bitcoin, verification is a digital signature by key that owns coin
- In ZeroCash [BCG+14], blockchain holds committed coins and transfers uze zerk-knowledge proofs (zk-SNARKS) validated by P
- Hawk [KMS+16] uses verifiable computation (VC)
  - Computation using VC performed off-chain by involved parties
  - P checks correctness of proof for VC

§ Private computation requires additional assumption (MPC, trusted HW ...)

Security and privacy

§ Transactional privacy
- Anonymity or pseudonymity through cryptographic tools
- Some is feasible today (e.g., anonymous credentials in IBM Identity Mixer)

§ Contract privacy
- Distributed secure cryptographic computation on encrypted data

§ Accountability & non-repudiation
- Identity and cryptographic signatures

§ Auditability & transparency
- Cryptographic hash chain

§ Many of these need advanced cryptographic protocols

Hyperledger Fabric

Hyperledger project

§ Open-source collaboration under Linux Foundation
- www.hyperledger.org
- Hyperledger unites industry leaders to advance blockchain technology (Dec. '15)
- 100 members today

§ Develops enterprise-grade, open-source distributed ledger technology

§ Code contributions from several members

§ Fabric is the IBM-started contribution – github.com/hyperledger/fabric/
- Security architecture and consensus protocols from IBM Research - Zurich


PREMIER
accenture
High performance. Delivered.
CME Group
DEUTSCHE BÖRSE GROUP
Digital Asset
DTCC
FUJITSU
HITACHI
Inspire the Next
IBM
intel
J.P.Morgan
R
GENERAL
ABN·AMRO
ANZ
belink
bitse
BLOCKCHAIN
Blockstream
bloq
BNY MELLON
Broadridge
Calastone
CISCO
cloudsoft
CLS
Fundamental to FX
coinplug
CONSENSYS
CREDITS
Cuscal
eVUE
digital labs
Gem
guardtime
intellect
INVeSHARE
itBit
Milligan Partners
MonetaGo
ML
MOSCOW EXCHANGE
Orchestrating a brighter world
NEC
norbloc
NTT DATA
onchain
redhat.
Ribbit me
Skry
SORAMITSU
STATE STREET
SWIFT
symbiont
tequa creek
THOMSON REUTERS
vmware
WELLS FARGO

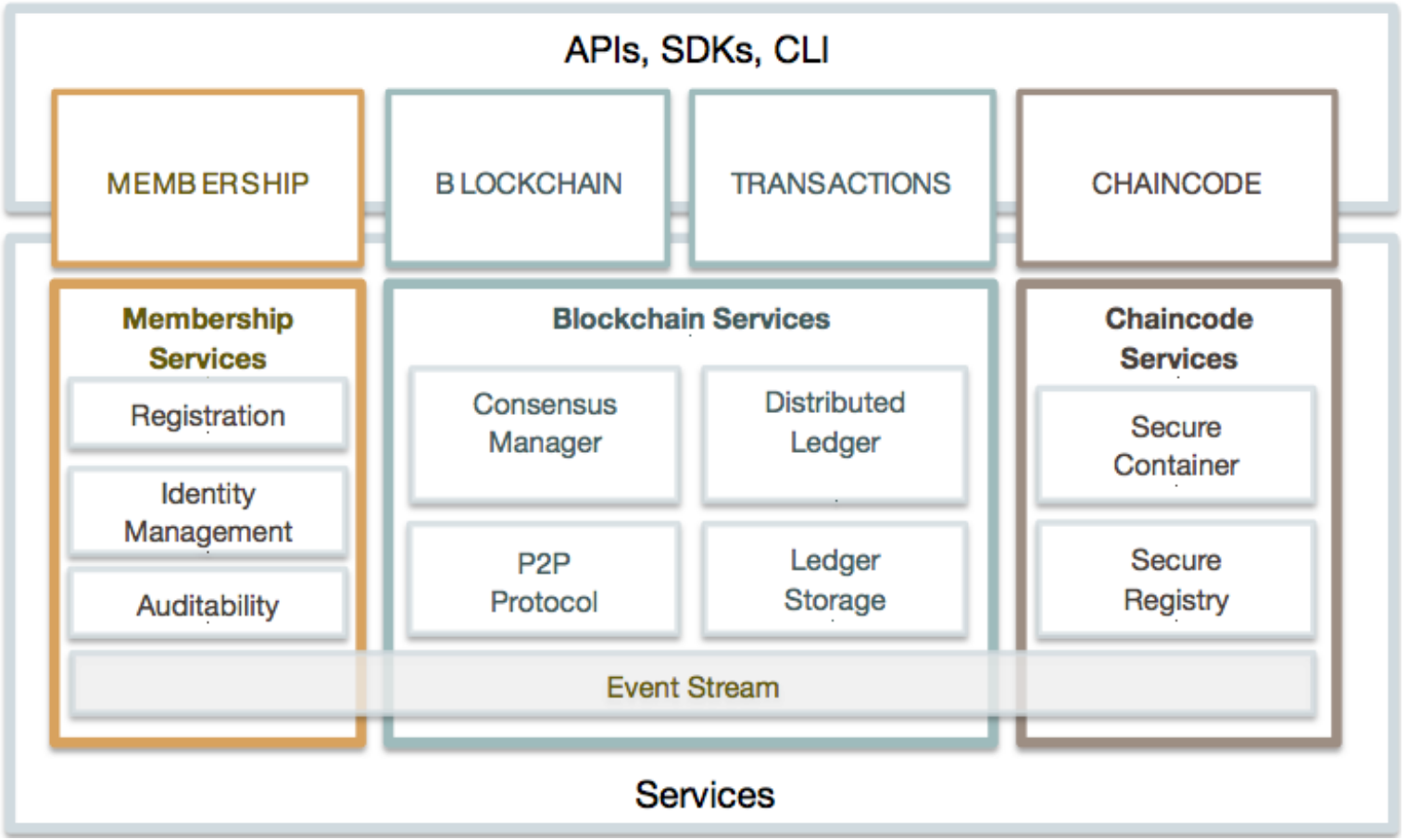
Hyperledger fabric

§ Enterprise-grade blockchain fabric and distributed ledger framework
  – A blockchain implementation in the Hyperledger Project

§ Developed open-source, by IBM and others (DAH, LSEG ...)
  – github.com/hyperledger/fabric
  – Initially called 'openblockchain' and donated by IBM to Hyperledger project
  – Actively developed, IBM and IBM Zurich play key roles

§ Technical details
  – Implemented in GO
  – Runs smart contracts ("chaincode") within Docker containers
  – Implements consortium blockchain using traditional consensus (BFT, Paxos)

Hyperledger fabric architecture



Hyperledger fabric details (v0.6-preview) / 1

§ Platform-agnostic
- GO, gRPC over HTTP/2

§ Peers
- Validating peers (all running consensus) and non-validating peers

§ Transactions
- Deploy new chaincode / Invoke an operation / Read state
- Chaincode is arbitrary GO program running in a Docker container

§ State is a key-value store (RocksDB)
- Put, get ... no other state must be held in chaincode
- Non-validating peers store state and execute transactions

Hyperledger fabric details / 2

§ Consensus in BFT model
- Modular architecture supports other consensus protocols
- Currently, Practical Byzantine Fault Tolerance (PBFT) [CL02]
- Non-determinism addressed by Sieve protocol [CSV16]
- Static membership in consensus group

§ Hash chain computed over state and transactions

Hyperledger fabric details / 3

§ Membership service issues certificates to peers

– Enrollment certificates (E-Cert, issued by E-CA)

• Assign identity to peer, gives permission to join and issue transactions

– Transaction certificates (T-Cert, issued by T-CA)

• Capability to issue one transaction (or more)
• Unlinkable to enrollment certificate, for anyone except for transaction CA

§ Pseudonymous transaction authorization

– Controlled by peer, how many Transaction-Signatures with same T-Cert

Non-determinism in BFT replication [CSV16]

§ Service-replication paradigm needs deterministic state machines
- Agree on order of operations, then every node executes

§ What if application is given as black-box? Deterministic? Undecidable!

§ Our approach – filter out inadvertent non-determinism
- Execute operation, compare results, and revert it if "too much" divergence is evident
- When "enough" nodes arrive at the same result, accept it

§ If application is randomized
- For algorithmic purpose (Monte Carlo): use master-slave approach
- For cryptography and security functions: cryptographic verifiable random functions (VRF)

Towards Hyperledger fabric V1

§ 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 [PWSKA00, 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
- Data seen only by endorsers assigned to run that chaincode

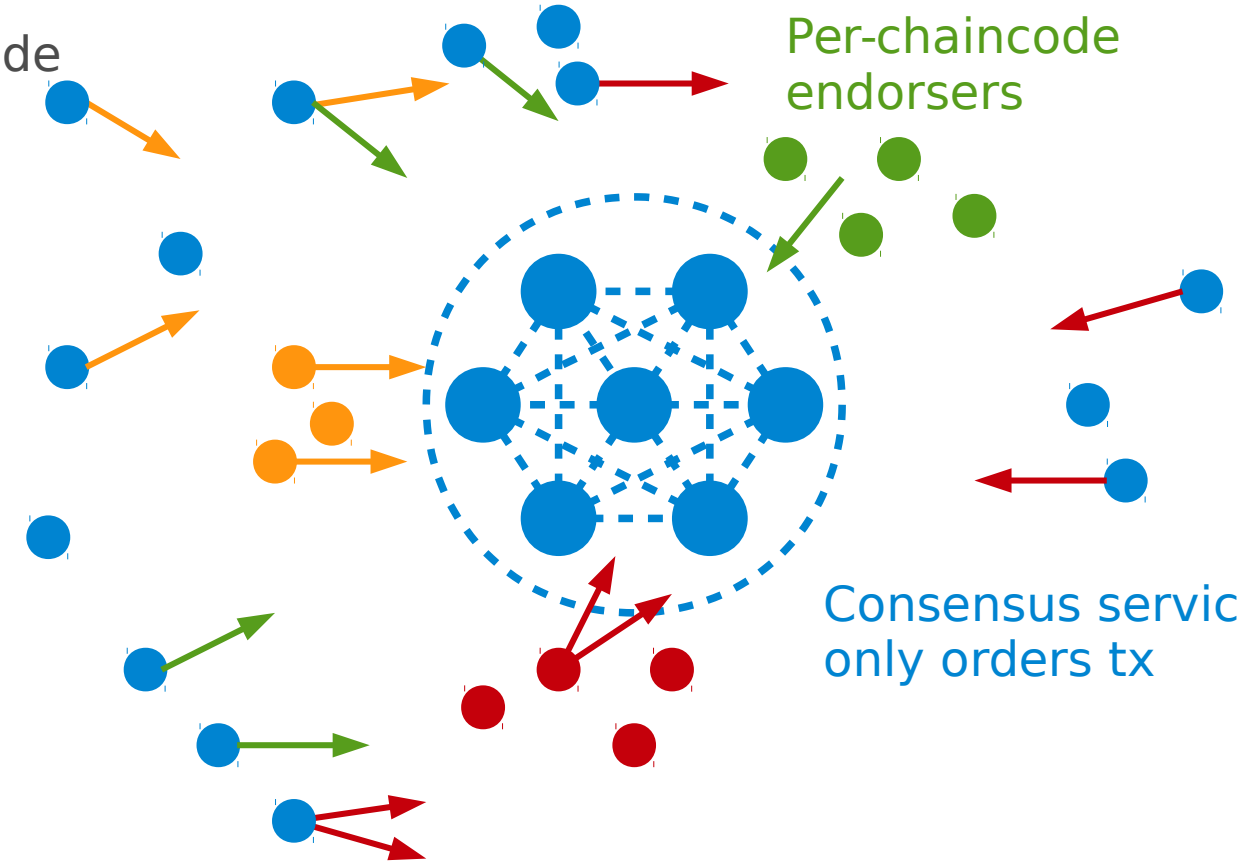
Separation of endorsement from consensus

§ Validation is by chaincode

§ Dedicated endorsers per chaincode

§ Consensus service
- Only communication
- Pub/sub messaging
- Ordering for endorsed tx

§ State and hash chain are common
- State may be encrypted



Transactions in fabric V1

§ Client
- Produces a tx (operation) for some chaincode (smart contract)

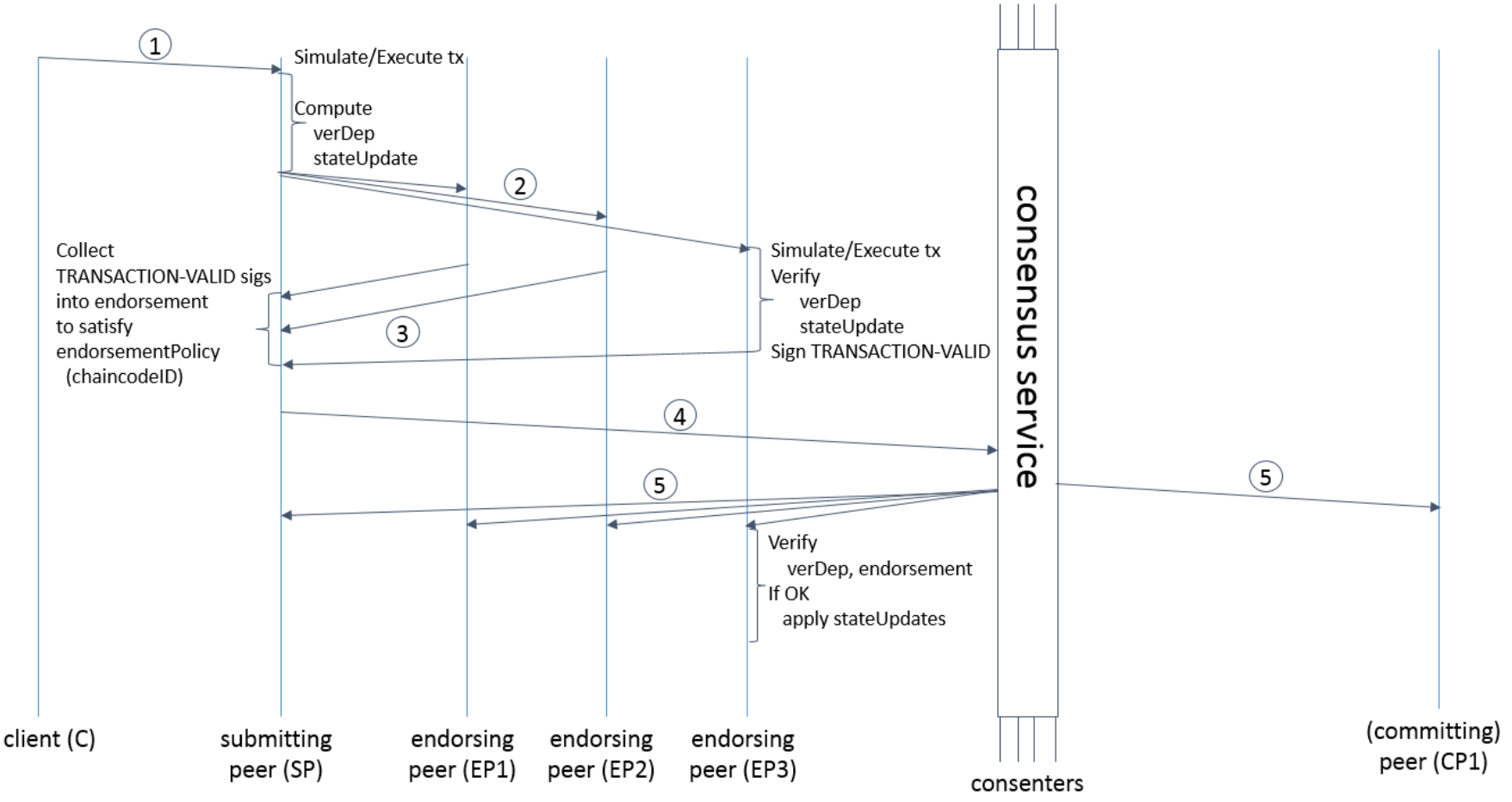
§ Submitter peer
- Execute/simulates tx with chaincode
- Records state values accessed, but does not change state → readset/writeset

§ Endorsing peer
- Re-executes tx with chaincode and verifies readset/writeset
- Endorses tx with a signature on readset/writeset

§ Consensus service
- Orders the endorsed tx, produces ordered stream of tx
- Filters out the not properly endorsed tx, according to chaincode endorsement policy

§ All peers
- Disseminate tx from consensus service with p2p communication (gossip)
- Execute state changes from readset/writeset of valid tx, in order

Transaction flow



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, has Apache Zookeeper
- Focus on high throughput

§ SBFT - A simple implementation of Practical Byzantine Fault Tolerance (PBFT)
- Tolerates f < n/3 Byzantine faulty nodes among n
- Focus on resilience

From the ideal world to the real world

Why does it take 15 years?

§ Agreement and consensus protocols have been researched for 30 years

§ Cryptographic e-cash (Chaum-style) from 1980s

§ 100s of protocols for anonymous communication/payment/credentials ...

§ Proof-of-work from 1990s

Blockchain and fintech today

§ Lots of activity and hype

§ 1000s (?) of startups

– Many with their "new" consensus protocols or crypto-magic (= rediscovered 90s results)

§ Is it all "déjà vu"? Are there any new ideas?

– DH 1975, RSA 1978 → → → PGP 1991, SSL 1995

– Ethernet Xerox PARC 1973 → → → IEEE 802.3 1983

Blockchain follows a typical technology cycle

§ Yes, many academic cryptography papers contain prototypes
  – but they remain academic prototypes
  – Almost never real-world systems

§ For the real world, it often takes another generation of people

§ Academics then re-visit practice and create new models for what is deployed

In theory, theory and practice are the same. In practice, they are not.

Summary

Blockchain – A golden opportunity for realizing cryptographic ideas

§ Blockchain enables new trust models

§ Many interesting technologies
- Distributed computing for consensus
- Cryptography for integrity, privacy, anonymity

§ We are only at the beginning

§ Blockchain = Distributing trust over the Internet

- www.hyperledger.org
- www.ibm.com/blockchain/
- www.research.ibm.com/blockchain/

Hyperledger Fabric references

§ www.hyperledger.org

§ Docs – hyperledger-fabric.readthedocs.io/en/latest/

§ Slack – hyperledgerproject.slack.com, all channels like #fabric-*

§ Designs – wiki.hyperledger.org/community/fabric-design-docs

§ Architecture of V1 – github.com/hyperledger/fabric/blob/master/proposals/r1/Next-Consensus-Architecture-Proposal.md

§ Code – github.com/hyperledger/fabric

References

[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.

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

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

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

[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.

References

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

[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.

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

[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.

[PWSKA00] F. Pedone, M. Wiesmann, A. Schiper, B. Kemme, G. Alonso: Understanding Replication in Databases and Distributed Systems. ICDCS 2000.

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