Hyperledger fabric: towards scalable blockchain for business

Trust in Digital Life
The Hague, Netherlands, June 17 2016
Blockchain – shared, replicated, ledger

Participants have possibly multiple shared ledgers

Consensus protocol ensures ledger replicas are identical*

Counter-party records

Party C’s Records

Party B Records

Ledger

Ledger

Ledger

Ledger

Ledger

Bank records

Auditor records

© 2015 IBM Corporation
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

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

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” \(\rightarrow\) (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])

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)

<table>
<thead>
<tr>
<th></th>
<th>Proof of Work (Bitcoin, Ethereum,...)</th>
<th>State machine replication (Ripple, Hyperledger, ...)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Membership type</td>
<td>Permisionless</td>
<td>Permissioned</td>
</tr>
<tr>
<td>User IDs (Sybil attack)</td>
<td>Decentralized, Anonymous (Decentralized protection by PoW compute/hash power)</td>
<td>Centralized, all Nodes know all other Nodes (Centralized identity management protects against Sybil attacks)</td>
</tr>
<tr>
<td>Scalability</td>
<td>Excellent &gt;100k Nodes</td>
<td>Verified up to few tens (or so) Nodes</td>
</tr>
<tr>
<td><strong>Open research problem:</strong></td>
<td><strong>Given the use case, network, no. of nodes</strong></td>
<td><strong>What is the most suitable and scalable Blockchain technology/protocol?</strong></td>
</tr>
<tr>
<td>Throughput</td>
<td>7 tx/sec upper bound (Bitcoin)</td>
<td>&gt;10k tx/sec with existing implementations in software</td>
</tr>
<tr>
<td>Power efficiency</td>
<td>&gt;1 GW (Bitcoin)</td>
<td>Good (commodity hardware)</td>
</tr>
<tr>
<td>Temporary forks in blockchain</td>
<td>Possible (leads to double-spending attacks)</td>
<td>Not possible</td>
</tr>
<tr>
<td>Consensus Finality</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
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).*
Existing blockchains unify many functionalities in one node

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

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


**Architecture-level approach to scalable and confidential blockchain**
## Blockchain fabric comparison

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

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)