Hyperledger Fabric

a Distributed Operating System for Permissioned Blockchains

TU Delft
Blockchain Lab Launch
What is a Blockchain?

- A chain (sequence, typically a hash chain) of blocks of transactions
  - Each block consists of a number of (ordered) transactions
  - Blockchain establishes total order of transactions

Consensus protocol ensures ledger replicas are identical*

Network of untrusted nodes
Blockchain transactions and distributed applications

• Bitcoin transactions
  - simple virtual cryptocurrency transfers

• Transactions do not have to be simple nor related to cryptocurrency
  - Distributed applications
  - smart contracts (Ethereum) or chaincodes (Hyperledger Fabric)

A smart contract is an event driven program, with state,
which runs on a replicated, shared ledger [Swanson2015]

“Smart contract” $\rightarrow$ (replicated) state machine
So we can just apply 40 years of research on RSM?

RSM = Replicated State Machines [Lamport 78, countless follow-up papers]

Well, not really…

Among other differences

<table>
<thead>
<tr>
<th>RSM approach</th>
<th>Blockchain smart-contracts</th>
</tr>
</thead>
<tbody>
<tr>
<td>single trusted application</td>
<td>Multiple applications</td>
</tr>
<tr>
<td>Not (necessarily) trusted!</td>
<td>Developed by third party application developers</td>
</tr>
</tbody>
</table>
Blockchain evolution (2009-present)

2009

Bitcoin

- A hard-coded cryptocurrency application w. limited stack-based scripting language
- Proof-of-work-consensus
- Native cryptocurrency (BTC)
- Permissionless blockchain system

Blockchain 1.0

2014

Ethereum

- Distributed applications (smart contracts) in a domain-specific language (Solidity)
- Proof-of-work-consensus
- Native cryptocurrency (ETH)
- Permissionless blockchain system

Blockchain 2.0

2017

Hyperledger Fabric

- Distributed applications (chaincodes) in different general-purpose languages (e.g., golang, Java, Node)
- Modular/pluggable consensus
- No native cryptocurrency
- Multiple instances/deployments
- Permissioned blockchain system

Blockchain 3.0
Hyperledger Fabric – key requirements

- No native cryptocurrency
- Ability to code distributed apps in general-purpose languages
- Modular/pluggable consensus

Satisfying these requirements required a complete overhaul of the (permissioned) blockchain design!

end result
Hyperledger Fabric v1
http://github.com/hyperledger/fabric
Open source, Apache 2.0 license
Blockchain Architecture 101
Permissionless Blockchains

- **PoW Consensus**
  - Block “mining”

- Block Validation / Smart Contract Execution (every miner)
  - **Validating (executing) transactions in the payload**
  - **Verifying** hash of Block #237 < DIFFICULTY

ORDER → EXECUTE architecture

Nodes execute smart-contracts after consensus (PoW)

- Finding nonces such that

\[ h = \text{hash of Block } #237 = \text{SHA256}(A || B || C) < \text{DIFFICULTY} \]
Permissioned Blockchain 2.0 architecture

ORDER → EXECUTE architecture
Active state machine replication [Schneider90]

- Inputs to the state machine (smart contract txs) are totally ordered
- Executed in sequence, after consensus (ordering)
- ALL permissioned blockchains are architected like this until Fabric v1
What are the issues with ORDER \(\rightarrow\) EXECUTE architecture?
ORDER → EXECUTE architecture issues (Blockchain 2.0)

- **Sequential execution of smart contracts**
  - long execution latency blocks other smart contracts, hampers performance
  - DoS smart contracts (e.g., infinite loops)
  - How Blockchain 2.0 copes with it:
    - Gas (paying for every step of computation)
    - Tied to a cryptocurrency

- **Non-determinism**
  - Smart-contracts must be deterministic (otherwise – state forks)
  - How Blockchain 2.0 copes with it:
    - Enforcing determinism: Solidity DSL, Ethereum VM
    - Cannot code smart-contracts in developers favorite general-purpose language (Java, golang, etc)

- **Confidentiality of execution:** all nodes execute all smart contracts

- **Inflexible consensus:** Consensus protocols are hard-coded
Hyperledger Fabric v1 Architecture

http://github.com/hyperledger/fabric
Application consists of two components:
1) Chaincode (execution code)
2) Endorsement policy (validation code)
Hyperledger Fabric v1 Transaction flow

1. \(<\text{PROPOSE}, \text{clientID}, \text{chaincodeID}, \text{txPayload}, \text{timestamp}, \text{clientSig}>\)
2. \(<\text{TX-ENDORSED}, \text{peerID}, \text{txID}, \text{chaincodeID}, \text{readset}, \text{writeset}>\)

Collect endorsement
("sufficient" no. of TX-ENDORSEDMsgs)

broadcast(endorsement)

Ordering service (consensus)

Total order semantics (ordering service)

3. BROADCAST(blob)
4. DELIVER(seqno,prevhash,block)

Simulate/Execute tx
Sign TX-ENDORSED

client (C)

endorse peer (EP1)

endorse peer (EP2)

endorse peer (EP3)

orders
Hyperledger Fabric v1 Transaction flow

1. <PROPOSE, clientID, chaincodeID, txPayload, timestamp, clientSig>
2. <TX-ENDORSED, peerID, txID, chaincodeID, readset, writeset>
3. BROADCAST(blob)
4. DELIVER(seqno, prevhash, block)

Collect endorsement
("sufficient" no. of TX-ENDORSED_msgs)

Broadcast(endorsement)

Ordering service (consensus)

(committing)

Ordering service (consensus)
Hyperledger Fabric v1 Transaction flow

1. <PROPOSE, clientID, chaincodeID, txPayload, timestamp, clientSig>
2. <TX-ENDORSED, peerID, txID, chaincodeID, readset, writeset>

Collect endorsement ("sufficient" no. of TX-ENDORSED Msgs)
broadcast(endorsement)

Simulate/Execute tx
Sign TX-ENDORSED

Total order semantics (ordering service)
3. BROADCAST(blob)
4. DELIVER(seqno,prevhash,block)

Ordering service (consensus)

client (C)
endorsing peer (EP1)
endorsing peer (EP2)
endorsing peer (EP3)
 endorsers
orderers
(committing) peer (CP4)
(committing) peer (CP5)
Hyperledger Fabric v1 Transaction flow

1. `<PROPOSE, clientID, chaincodeID, txPayload, timestamp, clientSig>
2. `<TX-ENDORSED, peerID, txID, chaincodeID, readset, writeset>
3. BROADCAST(blob)
4. DELIVER(seqno, prevhash, block)

Collect endorsement ("sufficient" no. of TX-ENDORSED Msgs)
Sufficiently enough to satisfy Endorsement Policy (EP)

Total order semantics (ordering service)

Simulate/Execute tx
Sign TX-ENDORSED

Validate(endorsement, chaincodeID, EP)
Validate(readset)
Commit tx

Validate(endorsement, chaincodeID, EP)
Validate(readset)
Commit tx

Validate(endorsement, chaincodeID, EP)
Validate(readset)
Commit tx

Validate(endorsement, chaincodeID, EP)
Validate(readset)
Commit tx

Ordering service (consensus)

Orderers

(client (C)
endorsing peer (EP1) endorsements peer (EP2) endorsing peer (EP3)

(ordering service)
(committing) peer (CP4) (committing) peer (CP5)
Challenge #1: Non-Determinism

- **Goals**
  - Enabling chaincodes in golang, Java, … (can be non-deterministic)
  - While preventing state-forks due to non-determinism

- **Hyperledger Fabric v1 approach**
  - Execute chaincode **before** consensus
  - Non-deterministic chaincode execution is tolerated
  - Use consensus to agree on propagation of versioned state-updates

**EXECUTE** ➔ **ORDER** ➔ **VALIDATE**:  
non-deterministic tx are not guaranteed to be live  
(e.g., cannot collect endorsement due to non-determinism)

**ORDER** ➔ **EXECUTE**
non-deterministic tx are not guaranteed to be safe (forks can occur)
Challenge #2: Sequential execution of smart-contracts

- **Goals**
  - Prevent slow smart-contracts from delaying the system
  - Address DoS without native cryptocurrency

- **Hyperledger Fabric v1 approach**
  - Partition execution of smart-contracts
  - Only a subset of peers are endorsers for a given smart-contract (chaincode)

- **DoS, resource exhaustion?**
  - HLF v1 transaction flow is resilient to non-determinism
  - Endorsers can apply local policies (non-deterministically) to decide when to abandon the execution of a smart-contract
  - No need for gas/cryptocurrency!
Challenge #3: Confidentiality of execution

- **Goal**
  - Not all nodes should execute all smart contracts

- **Hyperledger Fabric v1 approach**
  - Partition execution of smart-contracts
  - Only a subset of peers are endorsers for a given smart-contract (chaincode)

- **Confidentiality of data (versioned updates) is not yet addressed**
  - Support for ZKP-based data confidentiality in progress
Challenge #4: Consensus modularity/pluggability

Open research problem:
Given the use case, network, no. of nodes
What is the most suitable Blockchain consensus?

node scalability

M. Vukolić. *The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication*
Challenge #4: Consensus modularity/pluggability

- **Goal**
  - No-one-size-fits-all consensus → Consensus protocol must be modular and pluggable

- **Hyperledger Fabric v1 approach**
  - Fully pluggable consensus (was present in order-execute v0.6 design as well)

- **HLF v1 consensus (ordering service) implementations, Nov 2017**
  - Centralized! (SOLO, mostly for development and testing)
  - Crash FT (KAFKA, thin wrapper around Kafka/Zookeeper)
  - Proof of concept BFT
    - BFT-SMaRt library (University of Lisbon, September 2017)
    - Code: [https://github.com/jcs47/hyperledger-bftsmart](https://github.com/jcs47/hyperledger-bftsmart)
  - «Native» BFT implementation targeting about 100 orderers – in progress

- **Many more to come (TBC)**
  - Hybster SGX Consensus (TU Braunschweig, Eurosys 2017), Honeybadger BFT (UIUC, CCS’16, Stanford), XFT (IBM, OSDI’16)

Perhaps also your new, great blockchain consensus?
Fabric Validation – Endorsement Policies

- Deterministic (!) programs used for validation
- Executed by all peers post-consensus
- Cannot be deployed/installed by untrusted application developers
  - Unlike chaincode
  - Can only be parametrized by application developers

- Examples
  - K out of N chaincode endorsers need to endorse a tx
  - Alice OR (Bob AND Charlie) need to endorse a tx
  - Fabcoin – Bitcoin-inspired UTXO authority-minted cryptocurrency for Fabric
Fabric Hybrid Execution Model

- Endorsement Policy can, in principle, implement arbitrary logic
  - There are few things to have in mind

EXECUTE $\rightarrow$ ORDER $\rightarrow$ VALIDATE approach of Fabric v1
splits execution in two parts

EXECUTE (chaincode) $\rightarrow$ executed pre-consensus $\rightarrow$ can be non-deterministic

VALIDATE(endorsement policy) $\rightarrow$ evaluated post-consensus $\rightarrow$ must be deterministic

HLF v1 mixes passive and active replication into hybrid replication
We skipped many details

Membership Service Provider (and CAs), Chaincode details, Gossip, Ledger design, Channels

- Further reading
  - https://arxiv.org/abs/1801.10228
    Fabric systems paper incl. Fabcoin, out today!
Hyperledger Project and Fabric Use Cases (very briefly)
Hyperledger members

Source: https://www.hyperledger.org/about/members
Updated 21 August 2017
Over 400 prototypes, engagements and multiple active networks

<table>
<thead>
<tr>
<th>Trade Finance</th>
<th>Pre and Post Trade</th>
<th>Complex Risk Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital Trade Chain</td>
<td>DTCC</td>
<td>AIG</td>
</tr>
<tr>
<td>NATIXIS</td>
<td>CLS</td>
<td>Standard Chartered</td>
</tr>
<tr>
<td>TRAFIGURA</td>
<td>Bolsa Comercio Santiago</td>
<td></td>
</tr>
<tr>
<td>Mizuho</td>
<td>JPX</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Identity/ Know your customer (KYC)</th>
<th>Unlisted Securities/ Private Equity Funds</th>
<th>Loyalty Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>SECURE KEY</td>
<td>Crédit Mutuel ARKEA</td>
<td>SBI証券</td>
</tr>
<tr>
<td>DIA</td>
<td>NORTHERN TRUST</td>
<td>UnionPay</td>
</tr>
<tr>
<td>DIACC</td>
<td>BORSA ITALIANA</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Medicated Health Data Exchange</th>
<th>Fraud/ Compliance Registry</th>
<th>Distributed Energy/ Carbon Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDA</td>
<td>SMART DUBAI</td>
<td>TENNET</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supply Chain</th>
<th>Food Safety</th>
<th>Provenance/ Traceability</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAERSK</td>
<td>Walmart</td>
<td>everledger</td>
</tr>
<tr>
<td>PSA</td>
<td>Dole</td>
<td></td>
</tr>
<tr>
<td>Nestle</td>
<td>Driscoll’s</td>
<td></td>
</tr>
<tr>
<td>Unilever</td>
<td>Tyson</td>
<td></td>
</tr>
<tr>
<td>MCLANE</td>
<td>gsf</td>
<td></td>
</tr>
</tbody>
</table>
Further reading from IBM Research - Zurich

Fabric v1
https://arxiv.org/abs/1801.10228

On blockchain consensus out there...

Why we re-architected Fabric v1? (short-version)

On non-determinism

New consensus protocols (and fault models)

Pushing (crash tolerant) consensus over 3 million tps

PoW vs BFT consensus
Thank You!