Hyperledger Fabric:

A **Distributed Operating System** for Permissioned Blockchains

8 June 2023

Orestis Konstantinidis
@ Advanced Cryptography class

Order-execute Architecture

Blueprint of SMR
(active replication)

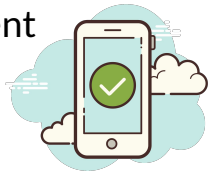
A protocol for **consensus** or **atomic broadcast**

1. orders the transactions and propagates them to all peers
2. each peer executes the transactions sequentially

- Consensus is **hard-coded**
- **Trust model** of transaction validation
  - Determined by the consensus
  - Cannot be adapted
- **Fixed, non-standard** or **domain-specific** language
  - Hinders adoption, coding errors
- **Sequential execution** of all transactions by all peers limits performance
  - Complex measures to prevent DoS attacks (e.g. “gas” in Ethereum)
- Transactions **must** be deterministic
- **Confidentiality** ??

Execute-order-validate Architecture

A Fabric Network

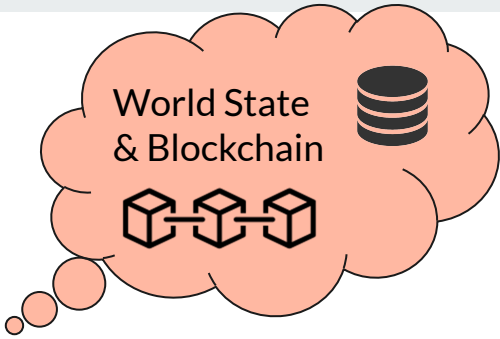


Client

Transaction proposal (1)

Proposal response (2)

Transaction (3)



Endorsers

= Peers | Endorsement Policy

Gossip (5)

Block dissemination (4)

Orderers

Non-Endorsing Peers

= !(Peers | Endorsement Policy)

Execution Phase

- Transaction proposal
- Proposal response
- Endorsement policy

Transaction proposal

Client Identity:

- According to the **Membership Service Provider (MSP)**

Transaction Payload:

- Operation to execute
- Parameters
- **Chaincode** id
- Nonce
- Transaction id <- **getTxId**(client id, nonce)

Proposal response

Proposal Simulation:

- Chaincode runs in a Docker container, **isolated** from the main endorser process
- Given the **appropriate permission**, a chaincode may invoke another chaincode

Endorsement:

- A message containing the result of the simulation **(readset, writeset)** and **metadata** (transaction id, endorser id)

Proposal Response:

- A **cryptographically signed** endorsement by the endorser

Endorsement Policy

Static library with strict rules of modification:

- Typically, a **monotone logical expression** on sets of peers (e.g. “3 out of 5”, “(A ∧ B) ∨ C”)
- **Custom** endorsement policies may implement **arbitrary logic**

Endorsement System Chaincode (ESCC):

- Endorsement is a message containing the result of the simulation **(readset, writeset)** and **metadata (transaction id, endorser id)**
- **[results, metadata]** <- **ESCC(***transaction proposal, transaction proposal simulation results***)**

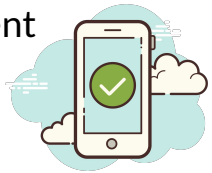
Validation System Chaincode (VSCC):

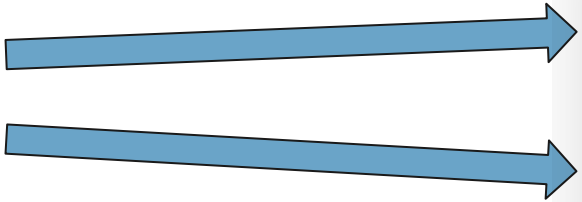
- Endorsement (output of ESCC) is validated as part of the input.

Ordering Phase

Ordering service interface

Client





```
class OrderingServiceAPI {

  function broadcast(Transaction tx){
    // ...
  }

  function deliver(Integer s){
    // ...
    return (Block) B;
  }
}
```

Built-in gossip service (optional):

- **Few nodes** are expected to implement the ordering service
- Implementation of the gossip service is **scalable** and **agnostic** to the implementation details of the ordering service.
- As a result, it works with both **CFT** and **BFT** ordering services **(modularity)**

Safety properties

Agreement: For any blocks **B, B'** delivered with sequence numbers **s, s':** $s = s' \Rightarrow B = B'$

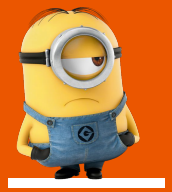
Hash chain integrity: For any blocks **B, B'** delivered with sequence numbers **s, s+1,** it holds that $h = Hash(B)$ is included in **B'**

No skipping: Delivering a block **B** with sequence number *s*, means that the orderer/peer has also delivered all blocks with sequence numbers in $S = \{0, 1, \ldots, s-1\}$

No creation: For any transaction **T** in a block, a client has broadcasted the transaction **T**

Validity: If a client broadcasts transaction **T**, then every correct peer eventually delivers a block **B** that includes **T**

Every individual ordering implementation
is allowed to come
with its own liveness and fairness guarantees
with respect to client requests



Other services

Access control (optional):

- Ordering service acts as a **trusted entity**

Reconfiguration of a channel:

- Its members modify the **channel** by broadcasting a *configuration update transaction*

Validation Phase

Three sequential steps

Endorsement policy evaluation (VSCC):

- Occurs in **parallel** for all transactions within the block

Read-Write conflict check:

- Compare the versions of the keys in the **readset** field to those in the **current state** of the ledger, as stored locally by the peer, and ensures they are still the same

Ledger update phase:

- The results of the above checks are persisted, in the form of a **bit mask** denoting the transactions that are valid within the block

The VSCC evaluation verifies that the set of peers, satisfy the expression of the endorsement policy

Configuration blocks

- Contain only the **full channel configuration**
- **Genesis** block is a configuration block
  (bootstraping the channel)

1. Definitions of the **MSPs (Membership Service provider)**
2. Network addresses of the **OSNs (Ordering Service Nodes)**
3. **Consensus** or ordering service configuration
   (batch size, timeouts)
4. Rules governing **access** to the ordering service operations
   (broadcast, deliver)
5. Rules governing how each part of the channel configuration may be **modified**

Thank you for your attention!