Consensus in Hyperledger Fabric

Mohan Dhawan
IBM Research, India
## Hyperledger Fabric v/s Ethereum

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Hyperledger Fabric</th>
<th>Ethereum</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description of platform</strong></td>
<td>Modular blockchain platform</td>
<td>Generic blockchain platform</td>
</tr>
<tr>
<td><strong>Governance</strong></td>
<td>Linux Foundation</td>
<td>Ethereum developers</td>
</tr>
<tr>
<td><strong>Usage</strong></td>
<td>Mission-critical managed enterprise applications</td>
<td>Anonymous “bitcoin-like” network optimized for “volume”, at the expense of managed governance</td>
</tr>
<tr>
<td><strong>Mode of operation</strong></td>
<td>Permissioned, private</td>
<td>Permissionless, public or private</td>
</tr>
<tr>
<td><strong>Consensus</strong></td>
<td>Broad understanding of consensus that allows multiple approaches</td>
<td>Mining based on Proof of Work (PoW) or Proof of Stake (PoS)</td>
</tr>
<tr>
<td><strong>Transaction level</strong></td>
<td></td>
<td>Ledger level</td>
</tr>
<tr>
<td><strong>Identity</strong></td>
<td>Participants declare who they are and their identities are confirmed</td>
<td>Anonymous</td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td>“OPEN” by providing transparency of participants</td>
<td>“OPEN” to everyone anonymously</td>
</tr>
<tr>
<td><strong>Trust</strong></td>
<td>Trust in the parties</td>
<td>Transactions can be trusted; parties cannot be trusted as they are anonymous</td>
</tr>
<tr>
<td><strong>Smart contracts</strong></td>
<td>Smart contract code (e.g., Go, Java, JavaScript)</td>
<td>Smart contract code (e.g., Solidity)</td>
</tr>
<tr>
<td><strong>Currency</strong></td>
<td>None</td>
<td>Ether</td>
</tr>
<tr>
<td></td>
<td>Currency and tokens via chaincode</td>
<td>Tokens via smart contract</td>
</tr>
</tbody>
</table>
Hyperledger Fabric V1 Architecture

- **Ledger** is a tamperproof record of transactions replicated across nodes in the network
  - A blockchain is a decentralized and distributed digital ledger
- **Chaincode** is a program that typically handles business logic agreed to by members of the network, so it may be considered as a “smart contract”
  - Initializes and manages ledger state through transactions submitted by applications
- **Types of nodes**
  - **Client**
    - Submits an actual transaction-invocation to the endorsers, and broadcasts transaction-proposals to the ordering service
  - **Peer**
    - Commits transactions and maintains the state and a copy of the ledger. Besides, peers can have a special endorser role
  - **Orderer**
    - Runs the communication service that implements a delivery guarantee, such as atomic or total order broadcast
Hyperledger Fabric V1 Architecture

- Client Application
- SDK (HFC)
- Membership Services Provider
- Ordering Service
- Peer
  - Endorser
  - Committer
    - Ledger
    - Chaincode
    - Events
Step 1/7 – Propose Transaction

Application proposes transaction

Endorsement policy:
• “E₀, E₁ and E₂ must sign”
• (P₃, P₄ are not part of the policy)

Client application submits a transaction proposal for Smart Contract A. It must target the required peers \{E₀, E₁, E₂\}

Key:

- Endorser
- Ledger
- Committing Peer
- Application
- Ordering Node
- Ordering-Service
- Smart Contract (Chaincode)
- Endorsement Policy
Endorsement Policy

• Describes the conditions by which a transaction can be endorsed
  – A transaction can only be considered valid if it has been endorsed according to its policy
  – Each chaincode is associated with an Endorsement Policy

• Examples of policies
  – AND('Org1.member', 'Org2.member', 'Org3.member')
  – OR('Org1.member', 'Org2.member')
  – OR('Org1.member', AND('Org2.member', 'Org3.member'))
Endorsers Execute Proposals

$E_0$, $E_1$ & $E_2$ will each execute the proposed transaction. None of these executions will update the ledger.

Each execution will capture the set of Read and Written data, called RW sets, which will now flow in the fabric.

Transactions can be signed & encrypted

Key:

- **Endorser**
- **Ledger**
- **Committing Peer**
- **Application**
- **Ordering Node**
- **Smart Contract (Chaincode)**
- **Endorsement Policy**
Step 3/7 – Proposal Response

Application receives responses

RW sets are asynchronously returned to application

The RW sets are signed by each endorser, and also includes each record version number

(This information will be checked much later in the consensus process)

Key:

- **Endorser**
- **Ledger**
- **Committing Peer**
- **Application**
- **Ordering Node**
- **Smart Contract (Chaincode)**
- **Endorsement Policy**
Application submits responses for ordering

Application submits responses as a transaction to be ordered.

Ordering happens across the fabric in parallel with transactions submitted by other applications.
Orderer delivers to all committing peers

Ordering service collects transactions into proposed blocks for distribution to committing peers. Peers can deliver to other peers in a hierarchy (not shown).

Different ordering algorithms available:
- Solo (Single node, development)
- Kafka (Crash fault tolerance)
Step 6/7 – Validate Transaction

Committing peers validate transactions

Every committing peer validates against the endorsement policy. Also check RW sets are valid for world state.

Validated transactions are applied to world state and retained on the ledger.

Invalid transactions are also retained on the ledger but do not update world state.

Key:

- **Endorser**
- **Committing Peer**
- **Ordering Node**
- **Smart Contract (Chaincode)**
- **Ordering-Service**
- **Hyperledger Fabric**
- **Client Application**
- **SDK**
- **Ledger**
- **Application**
- **Endorsement Policy**
Step 7/7 – Notify Transaction

Applications can register to be notified when transactions succeed or fail, and when blocks are added to the ledger. Applications will be notified by each peer to which they are connected.

Key:

- **Endorser**
- **Ledger**
- **Committing Peer**
- **Application**
- **Ordering Node**
- **Ordering-Service**
- **Smart Contract (Chaincode)**
- **Endorsement Policy**
Ordering Service

• Packages transactions into blocks to be delivered to peers
• Communication with the service is via channels
• Different configurations for the ordering service include
  – Solo: Single node for development
  – Kafka: Crash fault tolerant consensus
    • 3 nodes minimum, Odd number of nodes recommended
Channels

- Separate channels isolate transactions on different ledgers
  - Concurrent execution for performance and scalability
- Peers can participate in multiple channels
- Ledgers exist in the scope of a channel
  - Can be shared across an entire network of peers
Single Channel Network

- All peers connect to the same system channel (blue).
- All peers have the same chaincode and maintain the same ledger.
- Endorsement by peers $E_0$, $E_1$, $E_2$ and $E_3$.

Key:

- Endorser
- Committing Peer
- Ordering Node
- Smart Contract (Chaincode)
- Ledger
- Application
- Endorsement Policy
• Peers $E_0$ and $E_3$ connect to the red channel for chaincodes $Y$ and $Z$

• Peers $E_1$ and $E_2$ connect to the blue channel for chaincodes $A$ and $B$
Thank You

www.ibm.com/blockchain
developer.ibm.com/blockchain
www.hyperledger.org