CSE446: Blockchain & Cryptocurrencies

Lecture – 16: Hyperledger Fabric

BRAC UNIVERSITY

Inspiring Excellence

Agenda

- Hyperledger Fabric

Motivations for private blockchain systems

- As blockchain tech gaining maturity, interest in applying blockchain technology and distributed application platform to more innovative *enterprise* use cases started to grew
- Particularly they are intersected to disrupt current approach of many application domains
  - Disrupting the existing method brings innovation which leads to new service delivery models
  - CD/LP -> online music -> mp3 -> Napster (Torrents) -> Apple iPod/iTunes
- However, public blockchain systems are unsuitable for many enterprise use cases

Motivations for private blockchain systems

- For enterprise use, we need to consider the following requirements:
  - Participants must be identified/identifiable
    - E.g. financial transactions where Know-Your-Customer (KYC) and Anti-Money Laundering (AML) regulations must be followed
  - Networks need to be *permissioned*
  - High transaction throughput performance
  - Low latency of transaction confirmation
  - Privacy and confidentiality of transactions and data pertaining to business transactions

Hyperledger foundation

- Hyperledger Foundation is an open source collaborative effort created to advance cross-industry blockchain technologies
- It is a global collaboration, hosted by The Linux Foundation
- It includes leaders in finance, banking, Internet of Things, supply chains, manufacturing and Technology
- Joining forces to develop and promote private blockchain systems
- More information: https://www.hyperledger.org/

Hyperledger Fabric (HF)

- Hyperledger Fabric is an open source enterprise-grade permissioned distributed ledger technology (DLT) platform
- Designed for use in enterprise contexts
- To deliver some key differentiating capabilities over other popular distributed ledger or blockchain platforms
- More information: https://www.hyperledger.org/use/fabric
- Developer documentation: https://hyperledger-fabric.readthedocs.io/en/release-2.5/

Hyperledger Fabric (HF)

- Fabric is highly modular and configurable
- Fabric is the first distributed ledger platform to support smart contracts authored in general-purpose programming languages
  - such as Java, Go and JavaScript, rather than constrained domain-specific languages (DSL)
- Fabric supports pluggable consensus protocols that enable the platform to be more effectively customised to fit particular use cases and trust models
- Fabric does not require a native cryptocurrency

HF: Modularity

- A pluggable *ordering service* allows different types of consensus algorithms to be integrated with HF
- A pluggable *membership service provider* is responsible for associating entities in the network with cryptographic identities
- Smart contracts ("chaincode") run within a container environment (e.g. Docker) for isolation
  - They can be written in standard programming languages
- The ledger can be configured to support a variety of DBMSs
- A pluggable endorsement and validation policy enforcement that can be independently configured per application

HF: privacy & confidentiality

- Hyperledger Fabric, being a permissioned platform, enables confidentiality through its channel architecture
- Basically, participants on a Fabric network can establish a "channel" between the subset of participants that could grant visibility to a particular set of transactions
  - Think of this as a sub-network
- Thus, only those nodes that participate in a channel have access to the smart contract (chaincode) and data transacted, preserving the privacy and confidentiality of both

HF: performance and scalability

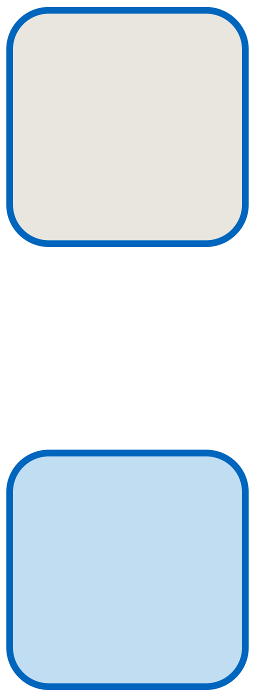
- Fabric is quite scalable because of its permissioned model of achieving consensus
- Performance of Fabric is reported to be quite satisfactory
  - Around 3500 tx/s (https://arxiv.org/pdf/1801.10228v1.pdf)
  - Compare this with only 3-5 tx/s for bitcoin and around 10 tx/s for Ethereum with PoW

HF: identity

- Fabric relies on a concrete identity management architecture
- HF provides a membership identity service that manages user IDs and authenticates all participants on the network
- Access control lists can be used to provide additional layers of permission through authorisation of specific network operations
  - For example, a specific user ID could be permitted to invoke a chaincode application, but be blocked from deploying new chaincode

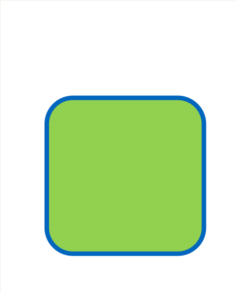
Nodes and roles

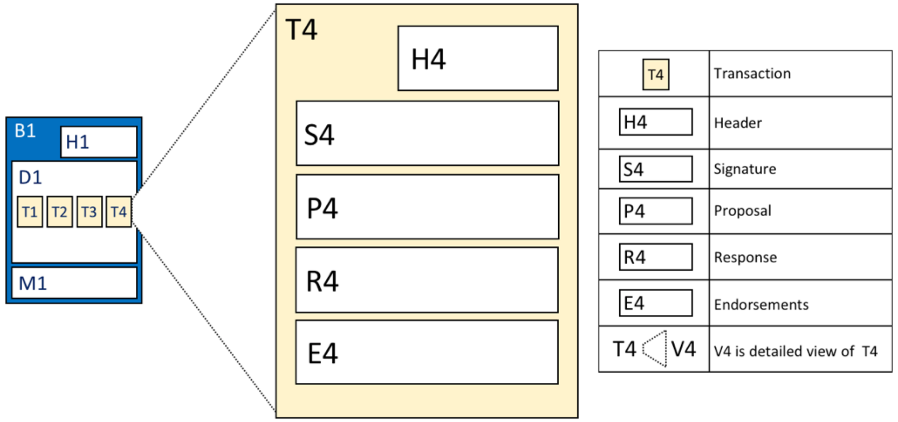
- Committing Peer
  - Maintains the state of the Blockchain and commits transactions
  - It verifies endorsements (rules and policies) and validate the results of a transaction
- Endorsing Peer
  - An endorsing peer is always also a committing peer
  - The difference is that an endorsing peer additionally takes transaction proposals (explained later) and executes them to create endorsements (explained later)



Nodes and roles

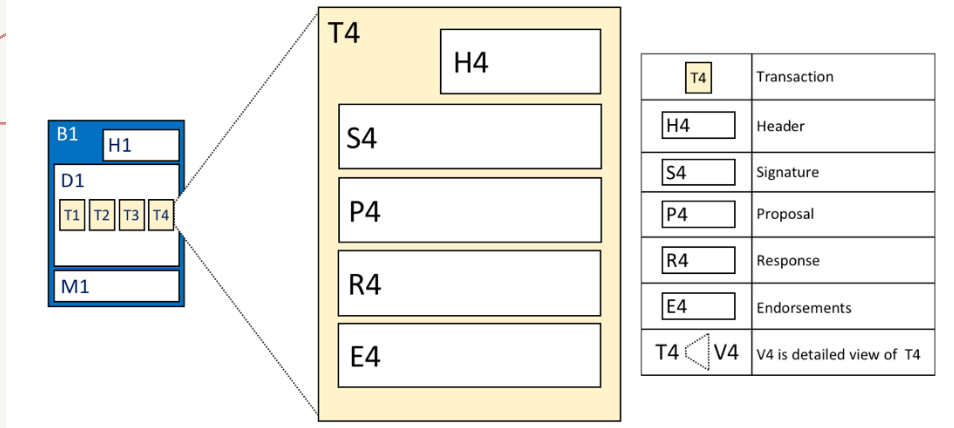
- Ordering service node
  - Applications must submit transactions to an ordering service node (Orderer)
  - The node then collects transactions and orders them sequentially
  - The transactions are then put into a new block and delivered to all peers of a specific channel
  - Does neither hold ledger nor chaincode





Fabric transactions

- Header: This section, illustrated by H4, captures some essential metadata about the transaction
  - for example, the name of the relevant chaincode, and its version
- Signature: This section, illustrated by S4, contains a cryptographic signature, created by the client application (user)
  - This field is used to check that the transaction details have not been tampered with, as it requires the application's private key to generate it
- Proposal: This field, illustrated by P4, encodes the input parameters supplied by an application to the smart contract which creates the proposed ledger update

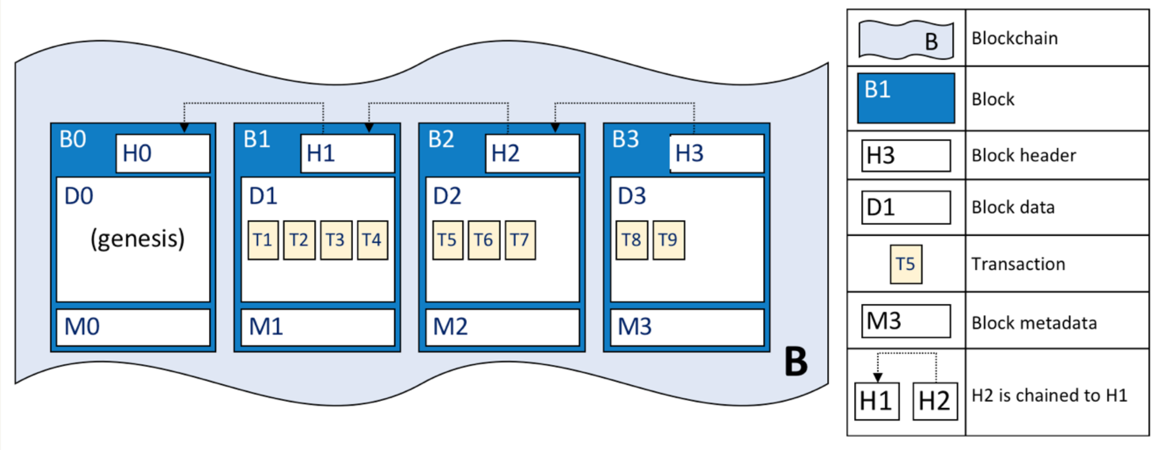


Fabric transactions

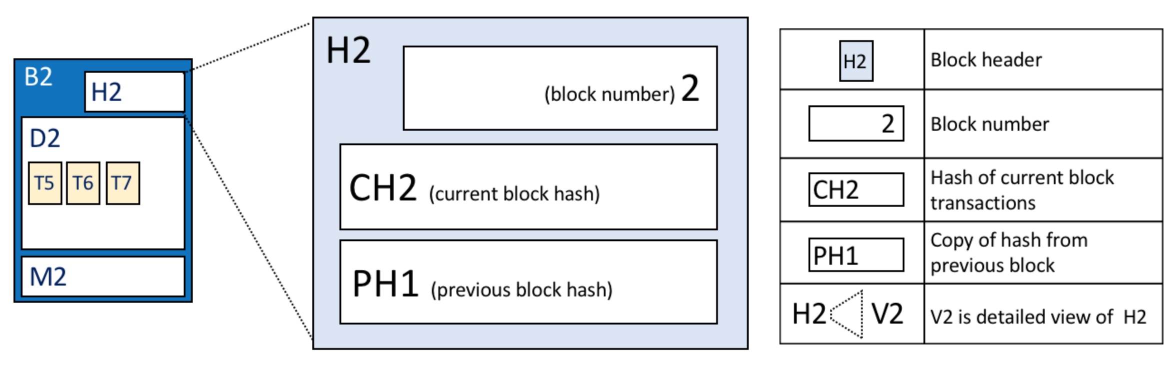
- Response: This section, illustrated by R4, captures the before and after values of the world state, as a Read Write set (RW-set)
  - It is the output of a smart contract
  - If the transaction is successfully validated, it will be applied to the ledger to update the world state
- Endorsements: As shown in E4, this is a list of signed transaction responses from each required organisation sufficient to satisfy the endorsement policy

Fabric blocks

- A blockchain B containing blocks B0 – B3
- B0 is the first block in the blockchain, the genesis block
- We can see that block B2 has a block data D2 which contains all its transactions: T5, T6, T7
- B2 has a block header H2
    - which contains a cryptographic hash of all the transactions in D2 as well as with the equivalent hash from the previous block B1
- This creates the chain among the blocks

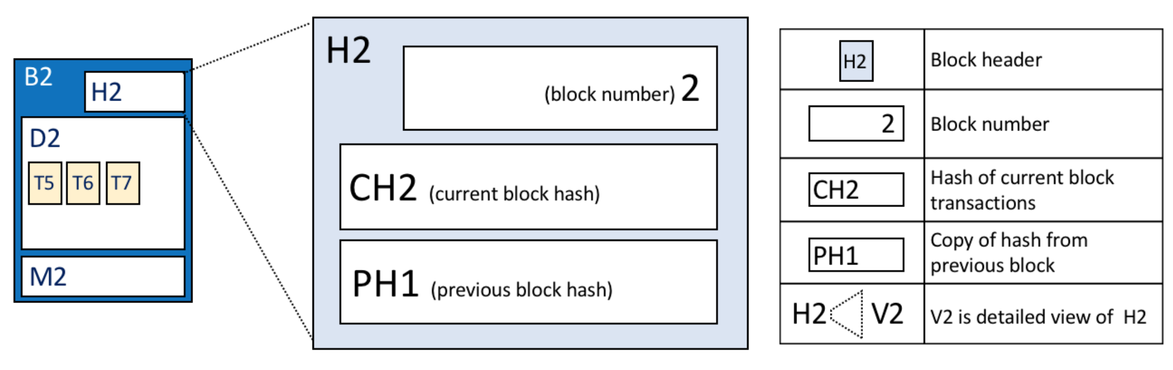


Fabric blocks



- Block Header: This section comprises three fields, written when a block is created
  - Block number: An integer starting at 0 (the genesis block), and increased by 1 for every new block appended to the blockchain
  - Current Block Hash: The hash of all the transactions contained in the current block
  - Previous Block Hash: A copy of the hash from the previous block in the blockchain

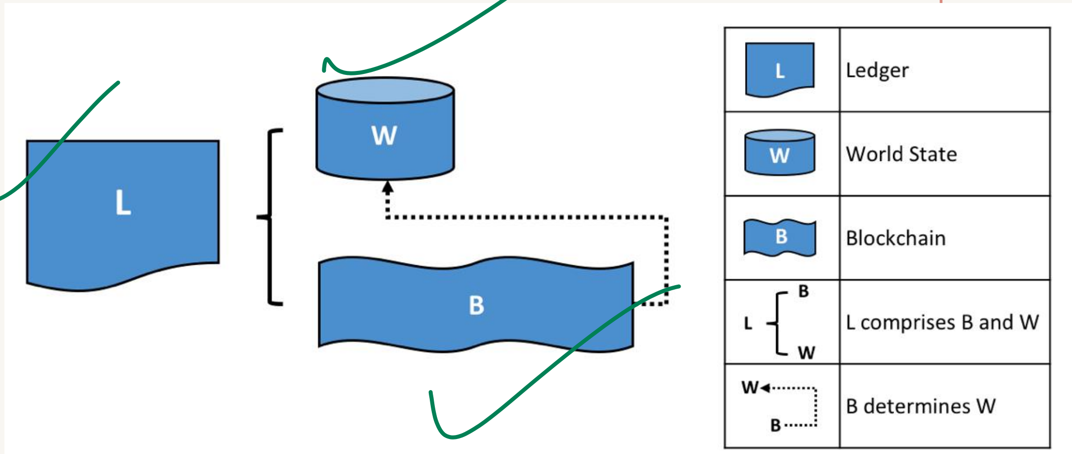
Fabric blocks



- Block Data
  - This section contains a list of transactions arranged in order
  - It is written when the block is created by the ordering service (block creation service)
- Block Metadata
  - This section contains the time when the block was written, as well as the certificate, public key and signature of the block writer
- Such metadata also contains a valid/invalid indicator for every transaction

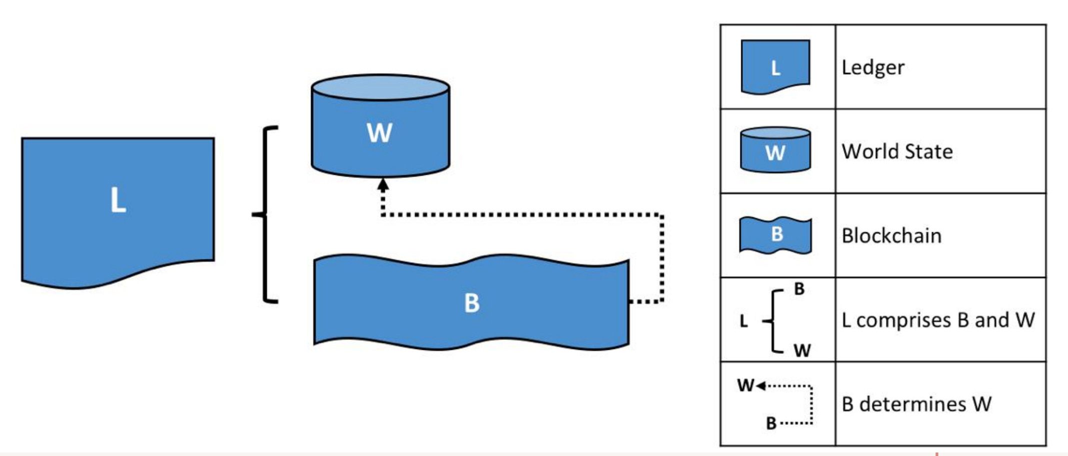
Fabric ledger

- In Hyperledger Fabric, a ledger consists of two distinct, though related, parts
  - a world state and a blockchain
- Each of these represents a set of facts about a set of business objects
- A **world state** is a database that holds a cache of the **current values** of a set of ledger states



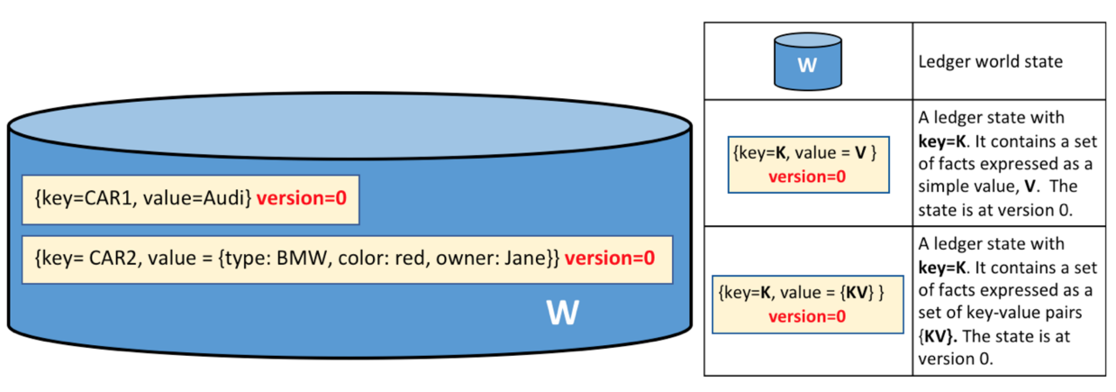
Fabric ledger

- The world state makes it easy for a program to directly access the current value of a state
- That is, you do not need to retrieve/calculate the state by traversing the entire transaction log as in Bitcoin
- Ledger states are, by default, expressed as **key-value** pairs



Fabric ledger

- A ledger world state might contain two types of states
- The first state is: key=CAR1 and value=Audi
- Here, the value is simple
- The second state has a more complex value:
- key=CAR2 and value={model:BMW, color=red, owner=Jane}
- Both states are at version 0
- The version number is incremented as the states are updated

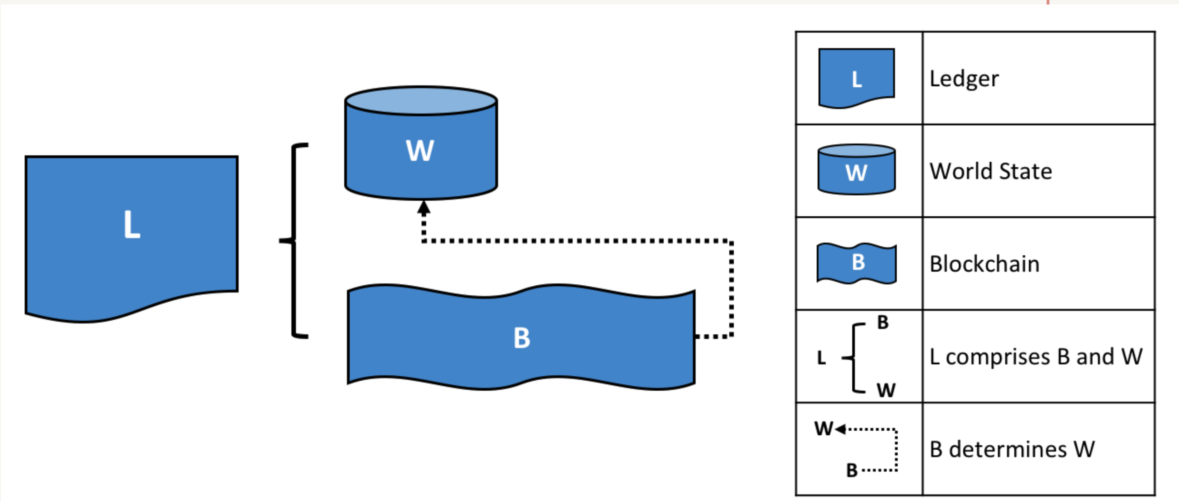


Fabric ledger

- At the initial stage, when a ledger is first created, the world state is empty
- When a transaction is created and it is then recorded in the blockchain
  - The world state is updated and recorded in the database

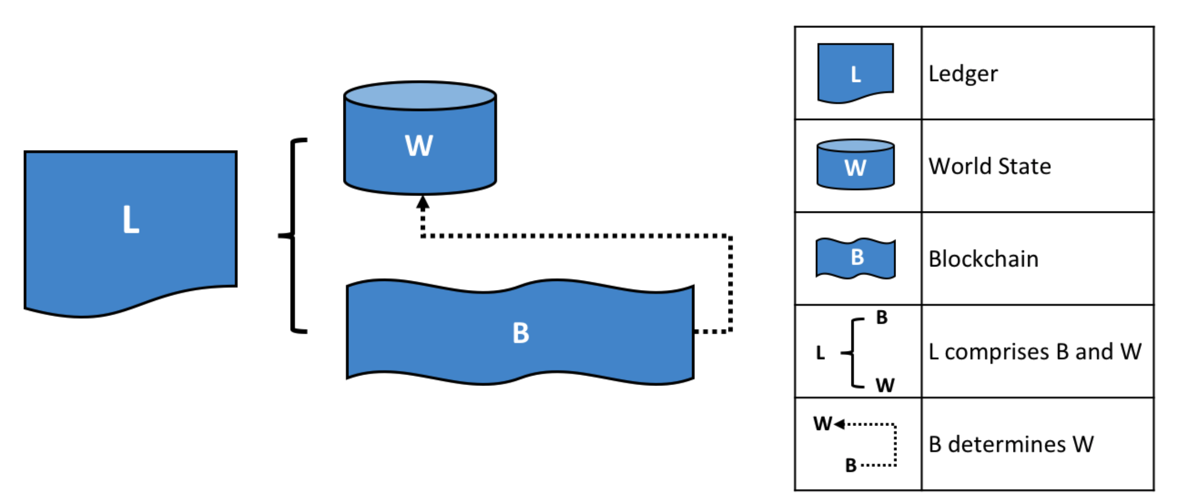
Fabric ledger

- The second part is the **blockchain**
- A blockchain is a transaction log that records all the changes that have resulted in the current the world state
- Transactions are collected inside blocks that are appended to the blockchain
- This enables you to understand the history of changes that have resulted in the current world state



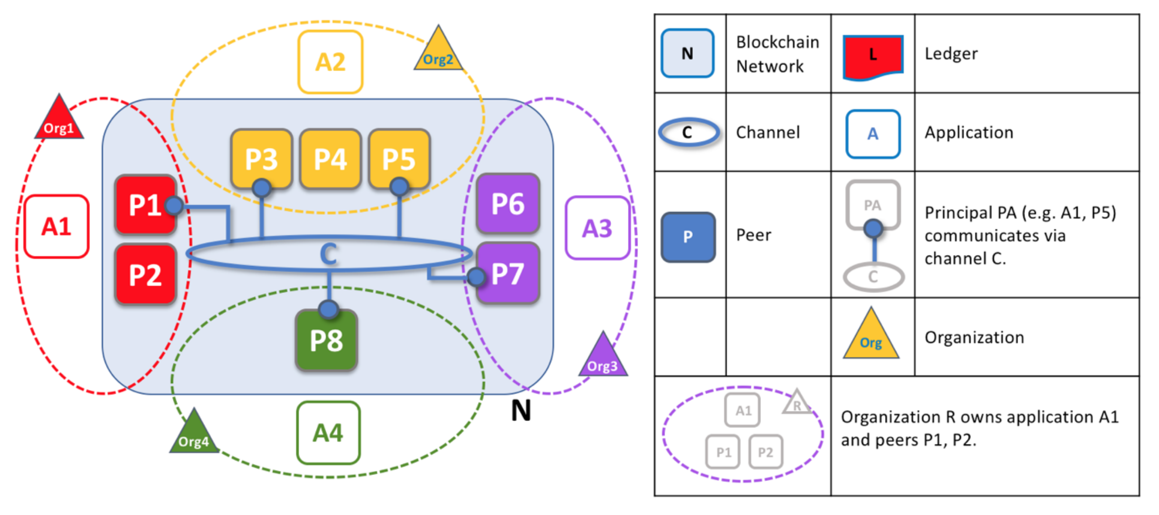
Fabric ledger

- The blockchain data structure is very different to the world state because once written, it cannot be modified
    - it is **immutable**
- We can also say that world state W is derived from blockchain B



Fabric network

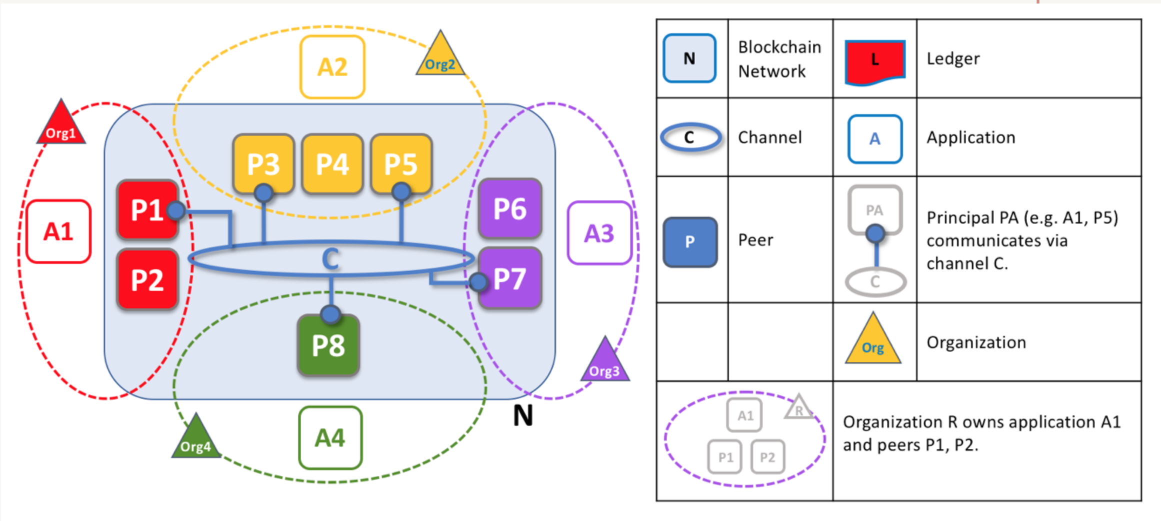
- In this example, we see four organisations contributing eight peers to form a network
- The channel C connects five of these peers in the network N – P1, P3, P5, P7 and P8
- The other peers owned by these organisations have not been joined to this channel, but are typically joined to at least one other channel



Peers in a blockchain network with multiple organisations

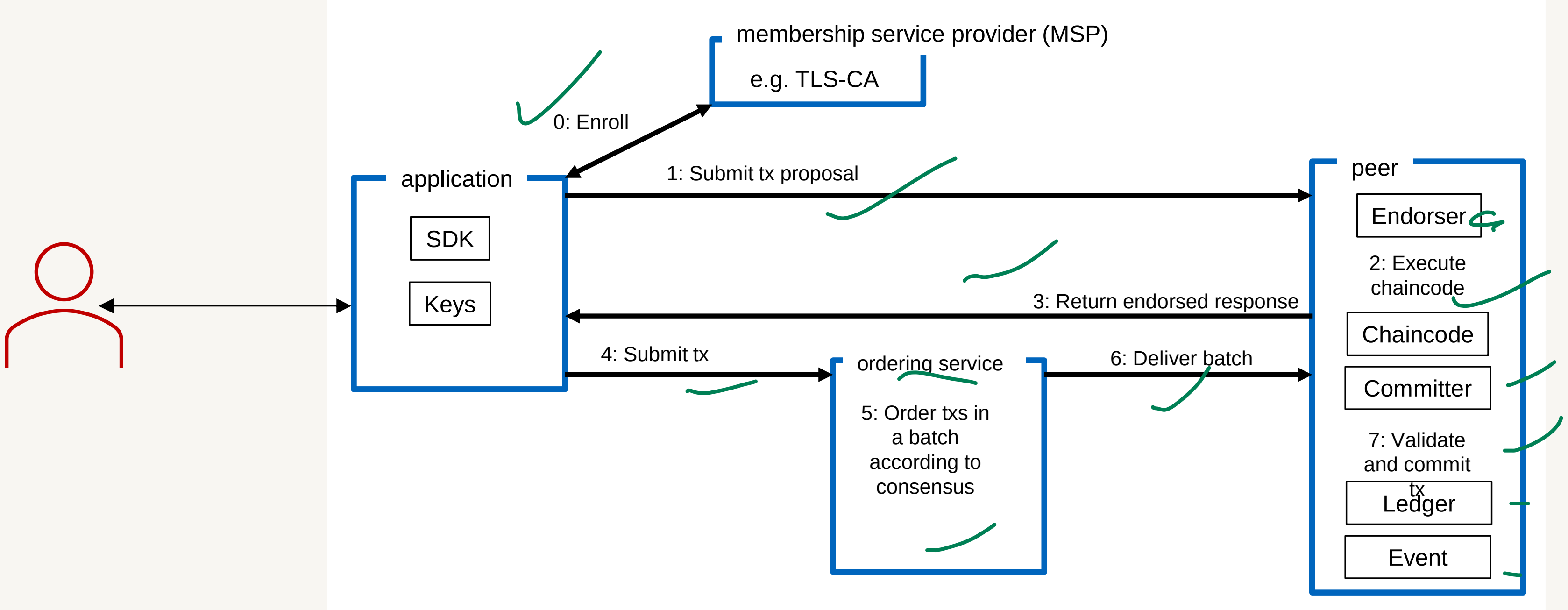
Fabric network

- Applications that have been developed by a particular organisation will connect to their own organisation's peers as well as those of different organisations
- For simplicity, an orderer node is not shown in this diagram



Peers in a blockchain network with multiple organisations

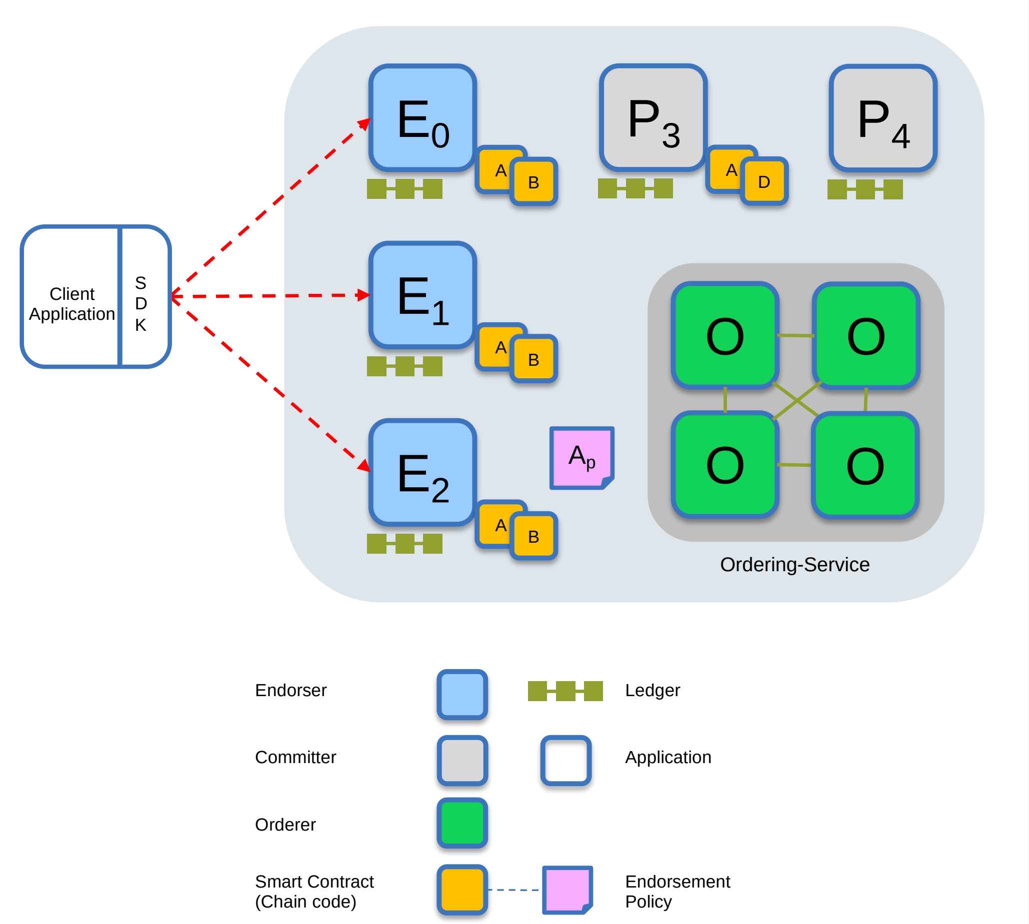
Transaction flow



In Fabric, a transaction initiates a seven step process from **simulation** of the executed chaincode (endorsement) to the **generation** of a read/write set which is then broadcasted to the ordering service and finally included in a new block (**committing**)

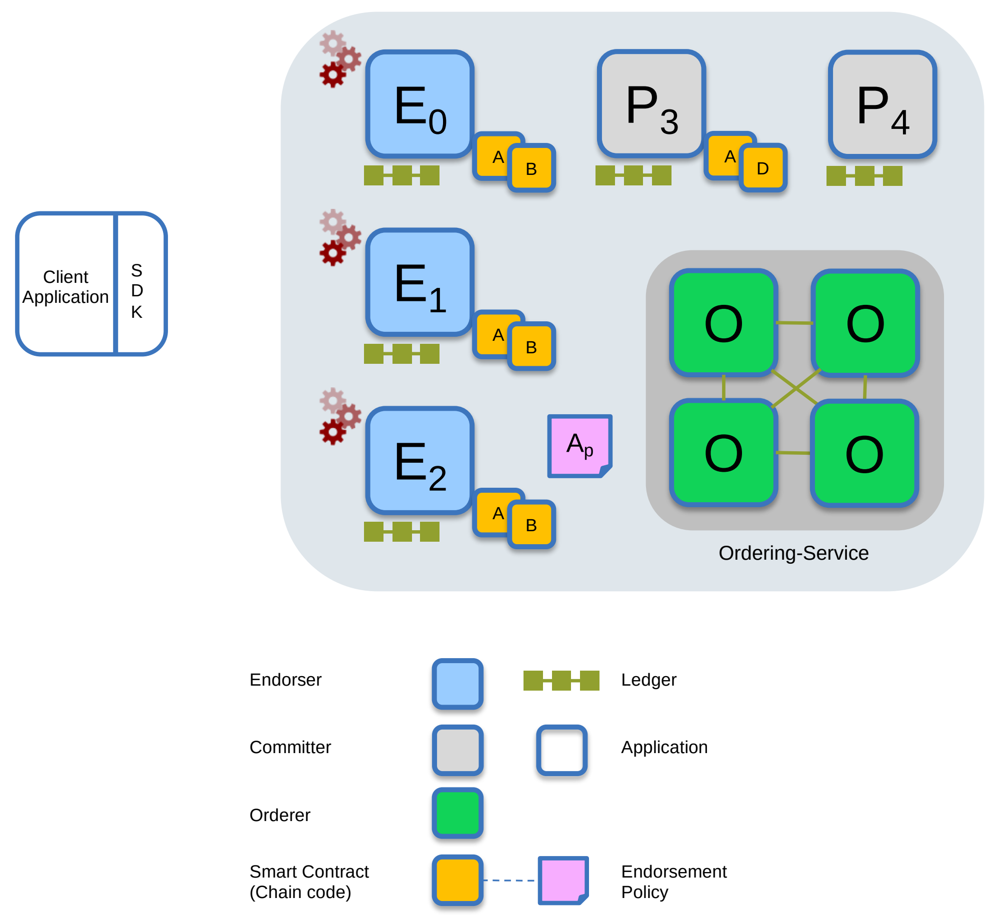
Transaction flow: steps 1/7

- Application creates a transaction proposal for **chaincode A** and sends it to all peers that are part of the endorsement policy **Ap**
- Endorsement Policy Ap
  - E_0, E_1 and E_2 must sign the transaction
  - P_3 and P_4 are not part of the policy
- Since only the peers $\mathbf{E_0}$, $\mathbf{E_1}$ and $\mathbf{E_2}$ are part of the endorsement policy **Ap,** it is not required to send the transaction proposal to $\mathbf{P_3}$ and $\mathbf{P_4}$



Transaction flow: steps 2/7

- E0, E1 and E2 will each simulate the execution of the *proposed* transaction from the application
- None of these executions will update the ledger
- The simulation will be used to capture the read and write operations on the ledger
- After the transaction is executed, each peer will have a generated read/write set (RW set)



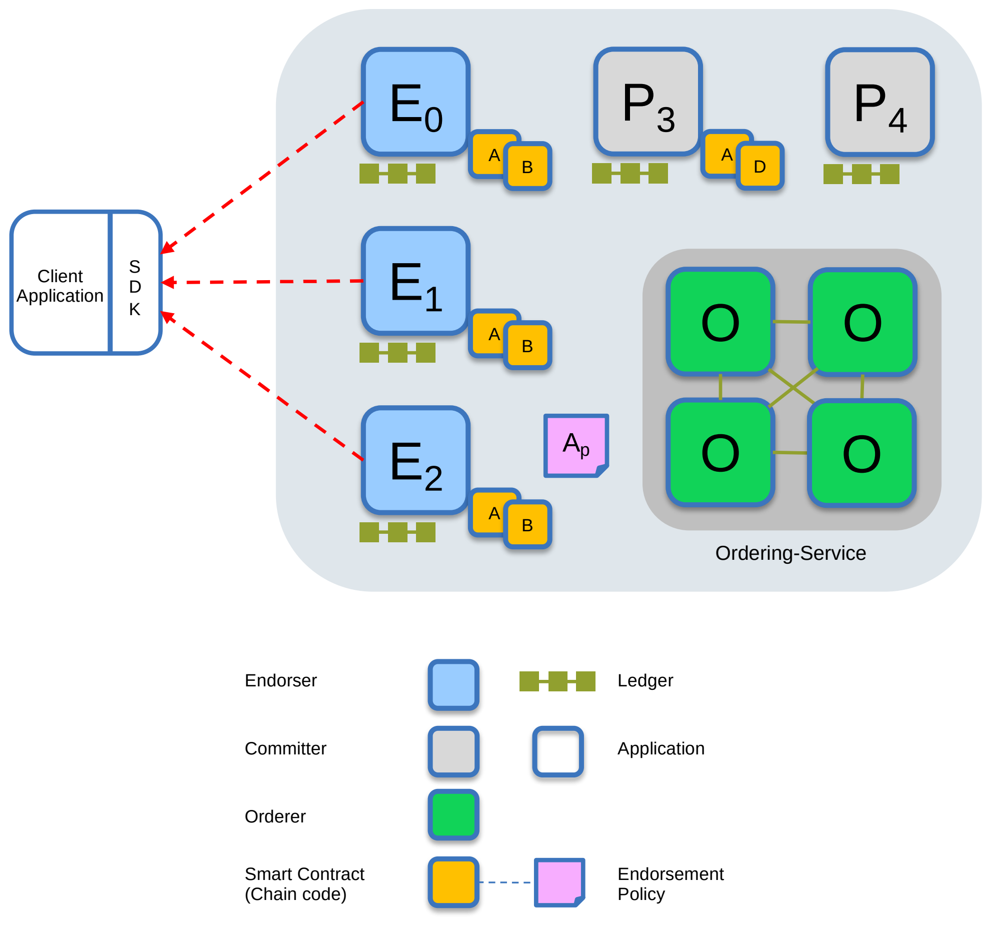

```xml
<TxReadWriteSet>
  <NsReadWriteSet name="chaincode1">
    <ReadSet>
      <read key="K1", version="1">
      <read key="K2", version="1">
    </ReadSet>
    <WriteSet>
      <write key="K1", value="V1">
      <write key="K3", value="V2">
      <write key="K4", isDelete="true">
    </WriteSet>
  </NsReadWriteSet>
<TxReadWriteSet>
```

Source: https://www

12 Hyperledger Fabric - Gallersdörfer, U., Holl, P., & Matthes, F. (2020). "Blockchain-based Systems Engineering". Lecture Slides. TU Munich.

Transaction flow: steps 3/7

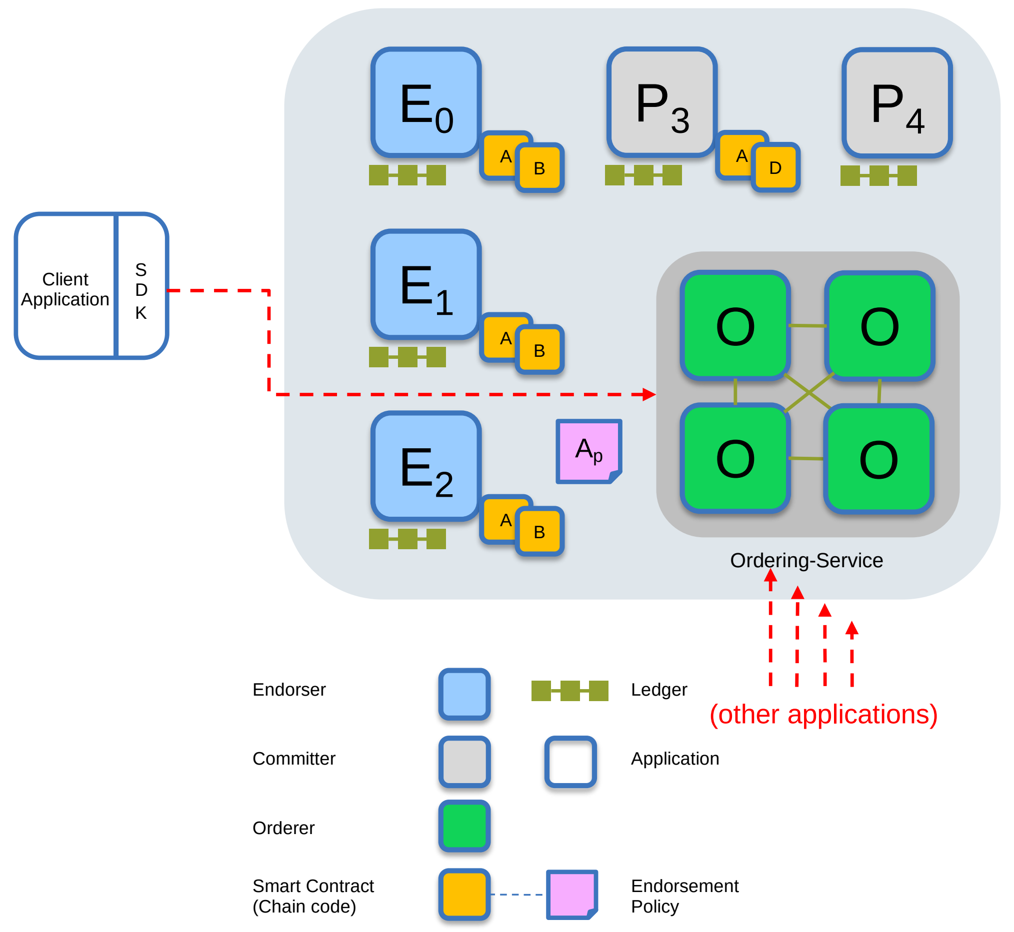
- **E_0**, **E_1** and **E_2** will each sign their generated read/write set and return it to the application that invoked the transaction
- This signed RW set is known as the endorsement



 Hyperledger Fabric - Gallersdörfer, U., Holl, P., & Matthes, F. (2020). "Blockchain-based Systems Engineering". Lecture Slides. TU Munich.

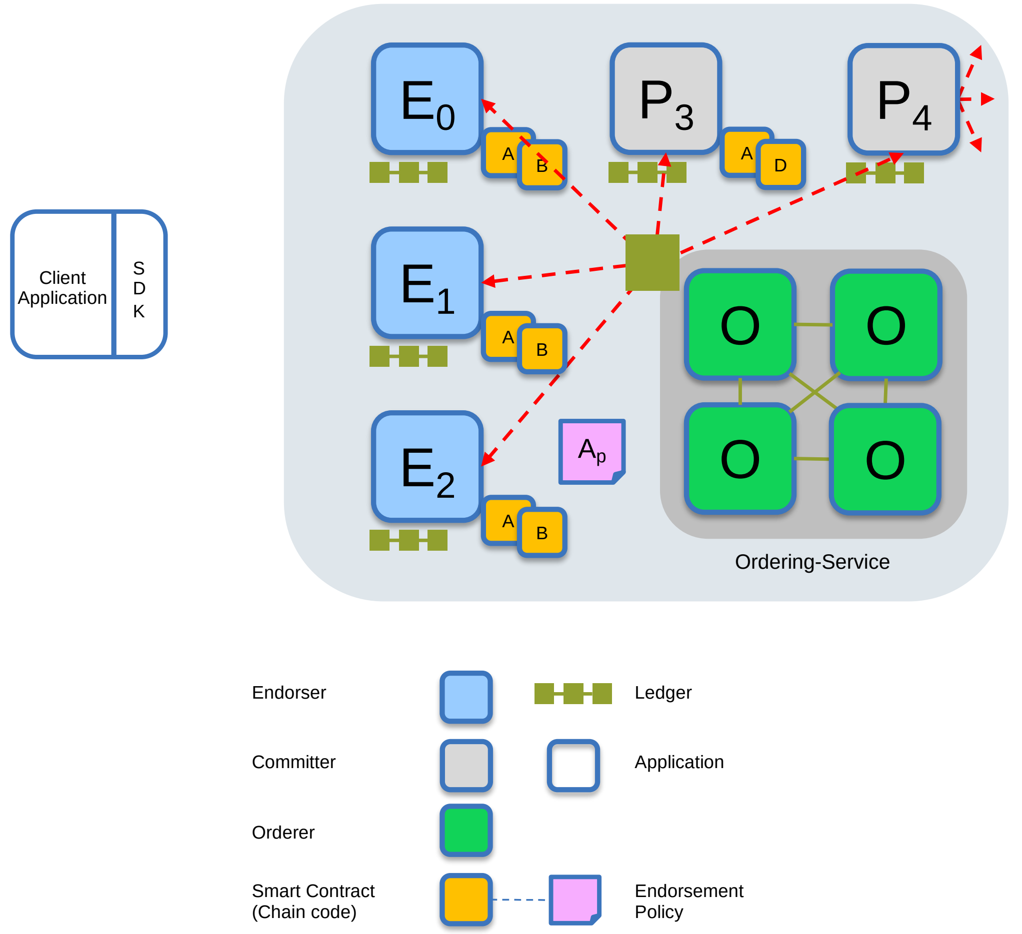
Transaction flow: steps 4/7

- The application submits the signed responses from **E0**, **E1** and **E2** to the ordering service
- The ordering service is responsible to order all transactions from all applications in the network
- The service tries to **serialize** the incoming transactions



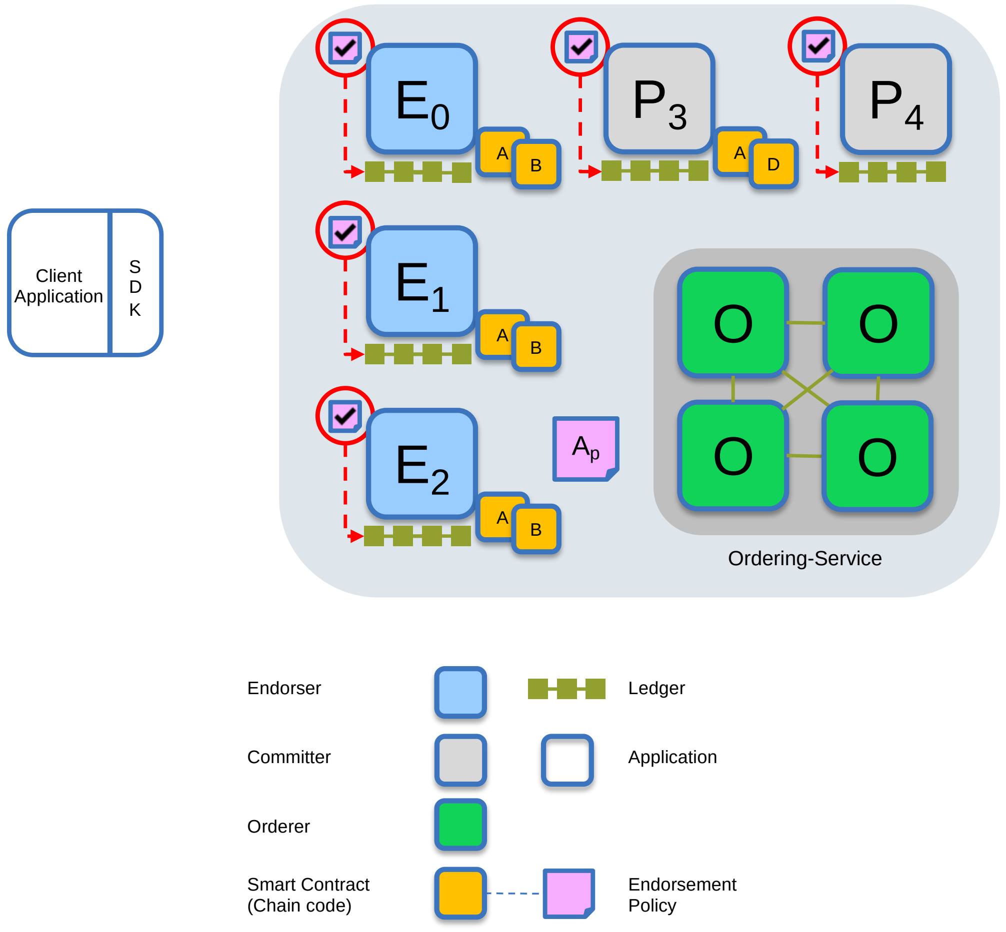
Transaction flow: steps 5/7

- The ordering service creates a new block based on the incoming transactions
- The block will then be broadcasted to all peers in the channel
- Currently, the ordering service supports three different ordering algorithms:
    - SOLO (single node, development)
    - Kafka (blocks map to topics)
    - Raft (crash fault tolerant (CFT), follows a "leader and follower" model)



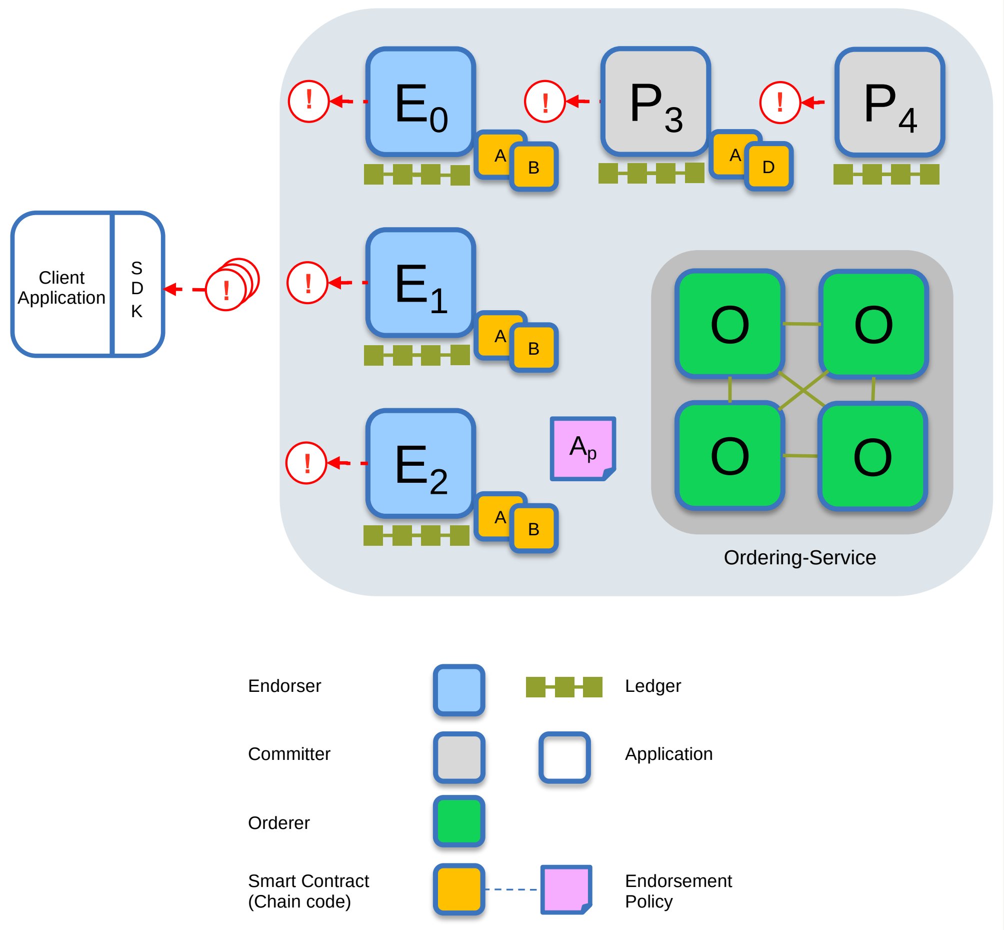
Transaction flow: steps 6/7

- All peers in the channel validate the transaction (read/write set) according to the endorsement policy of the chaincode A
- If the transaction is valid, the read and write set is written to the ledger and added as a new block to the blockchain
- The databases used for caching are updated with the new state information accordingly



Transaction flow: steps 7/7

- Applications can register to be notified by the peers once the transaction is done
- The peers will emit an event that indicates if the transaction succeeded or failed
- Applications can also subscribe to state changes of the ledger, i.e. a connected peer will then notify them if new blocks are added to the chain



 Hyperledger Fabric - Gallersdörfer, U., Holl, P., & Matthes, F. (2020). "Blockchain-based Systems Engineering". Lecture Slides. TU Munich.

Question?