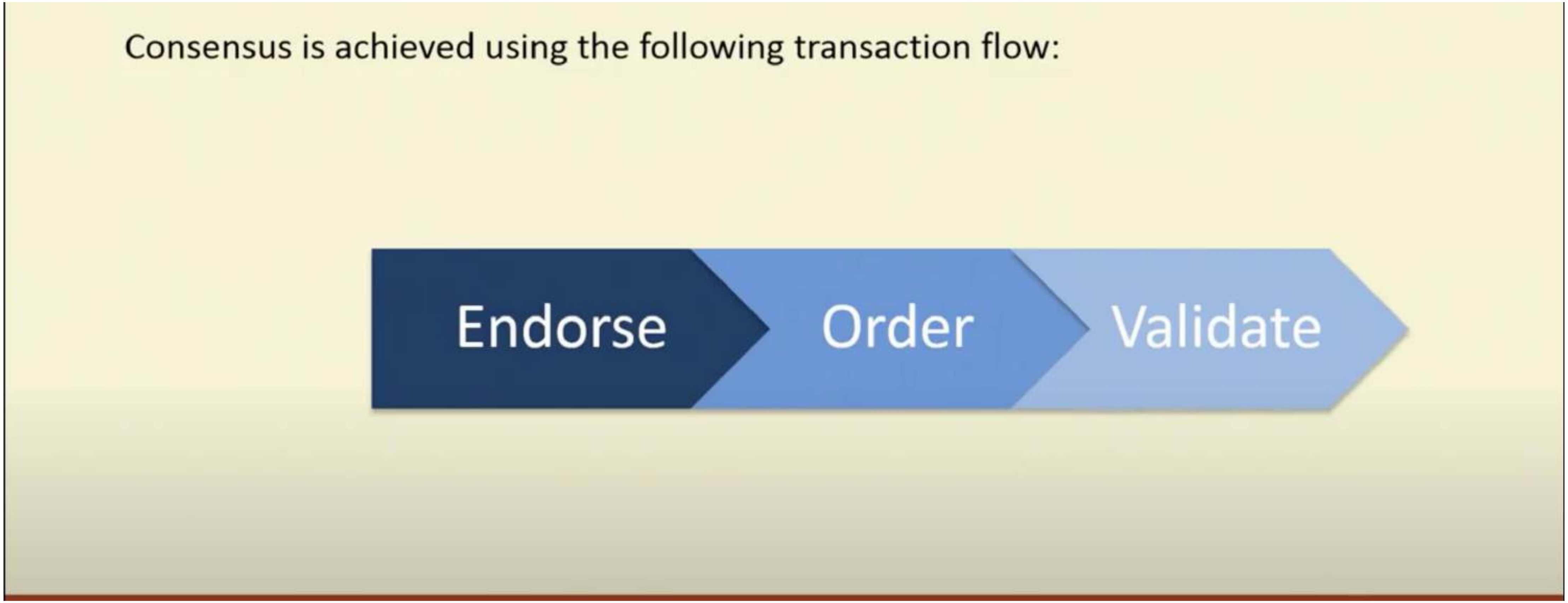
Transaction Flow

Transaction Flow



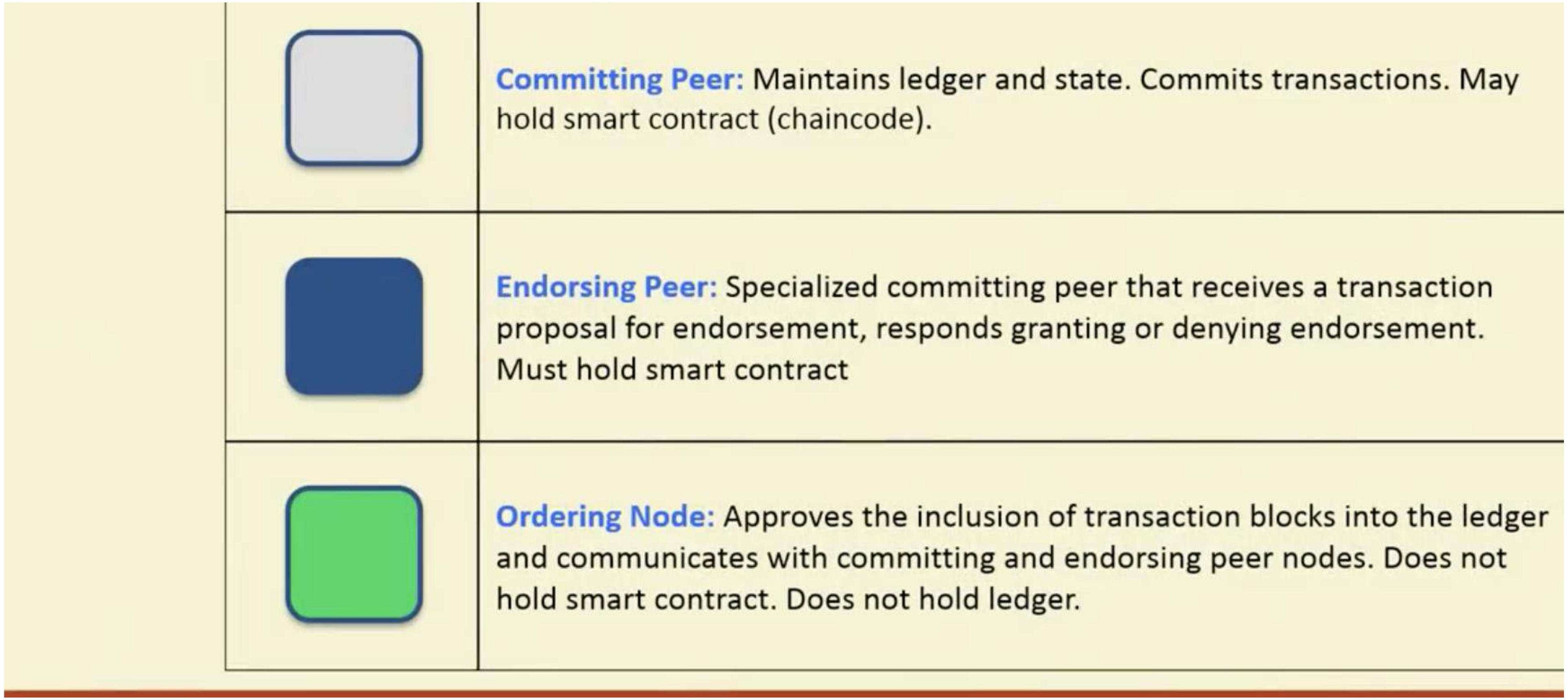
Consensus is achieved using the following transaction flow:

Endorse

Order

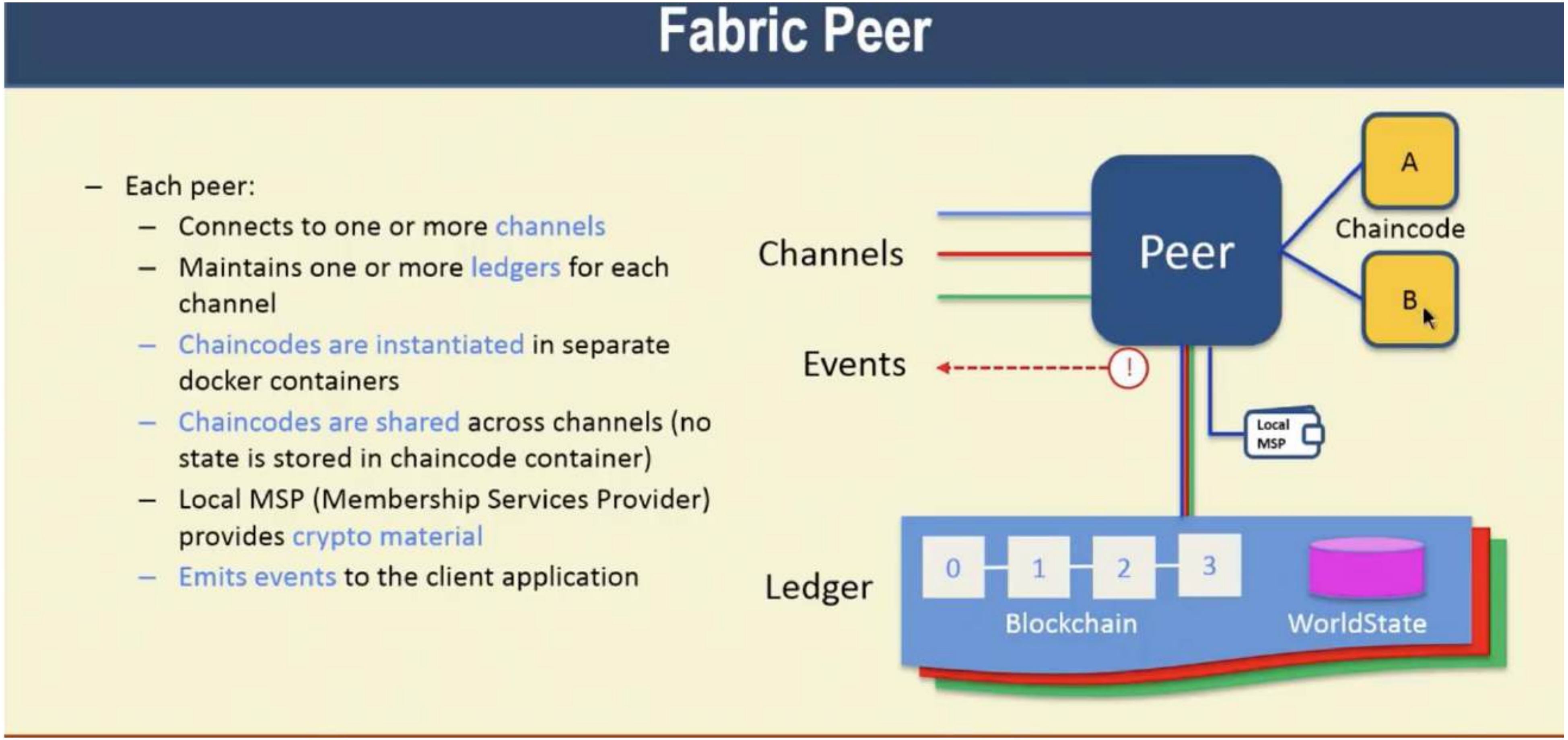
Validate

Nodes and Roles



⬜	**Committing Peer:** Maintains ledger and state. Commits transactions. May hold smart contract (chaincode).
🟦	**Endorsing Peer:** Specialized committing peer that receives a transaction proposal for endorsement, responds granting or denying endorsement. Must hold smart contract
🟩	**Ordering Node:** Approves the inclusion of transaction blocks into the ledger and communicates with committing and endorsing peer nodes. Does not hold smart contract. Does not hold ledger.

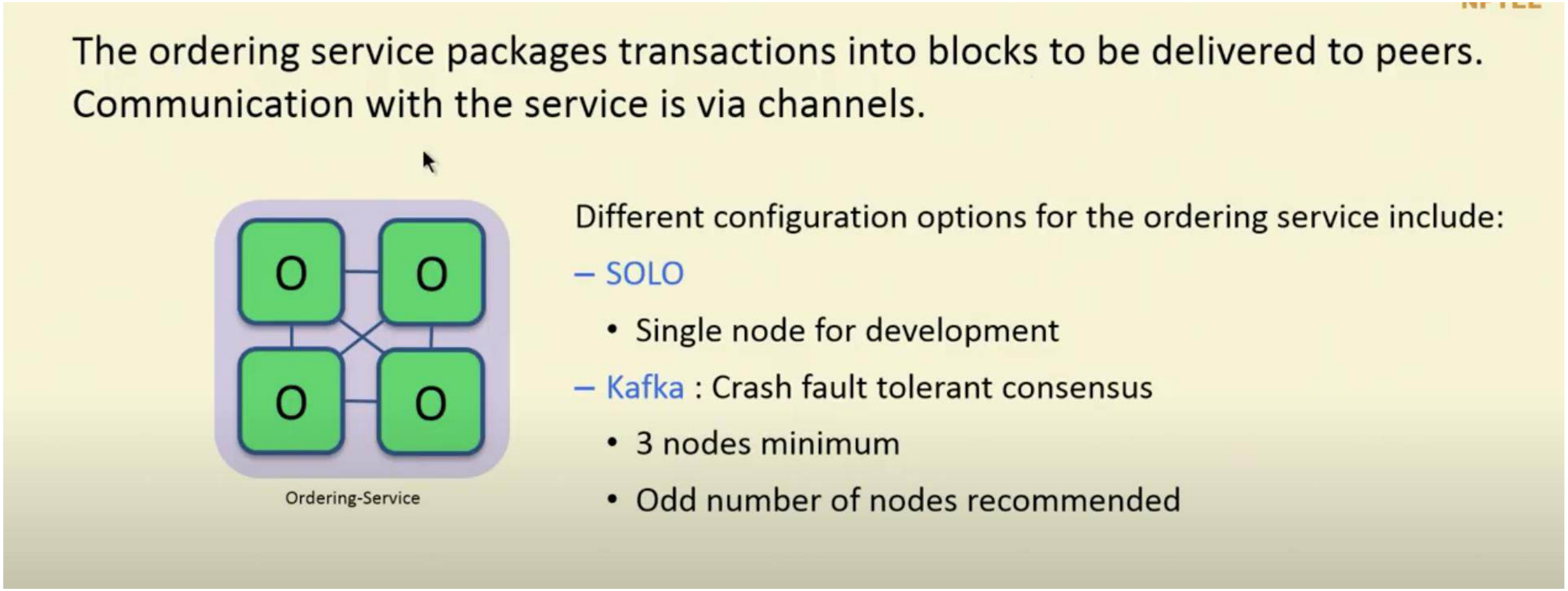
Fabric PEER



Fabric Peer

- Each peer:
  - Connects to one or more channels
  - Maintains one or more ledgers for each channel
  - Chaincodes are instantiated in separate docker containers
  - Chaincodes are shared across channels (no state is stored in chaincode container)
  - Local MSP (Membership Services Provider) provides crypto material
  - Emits events to the client application

Ordering Service



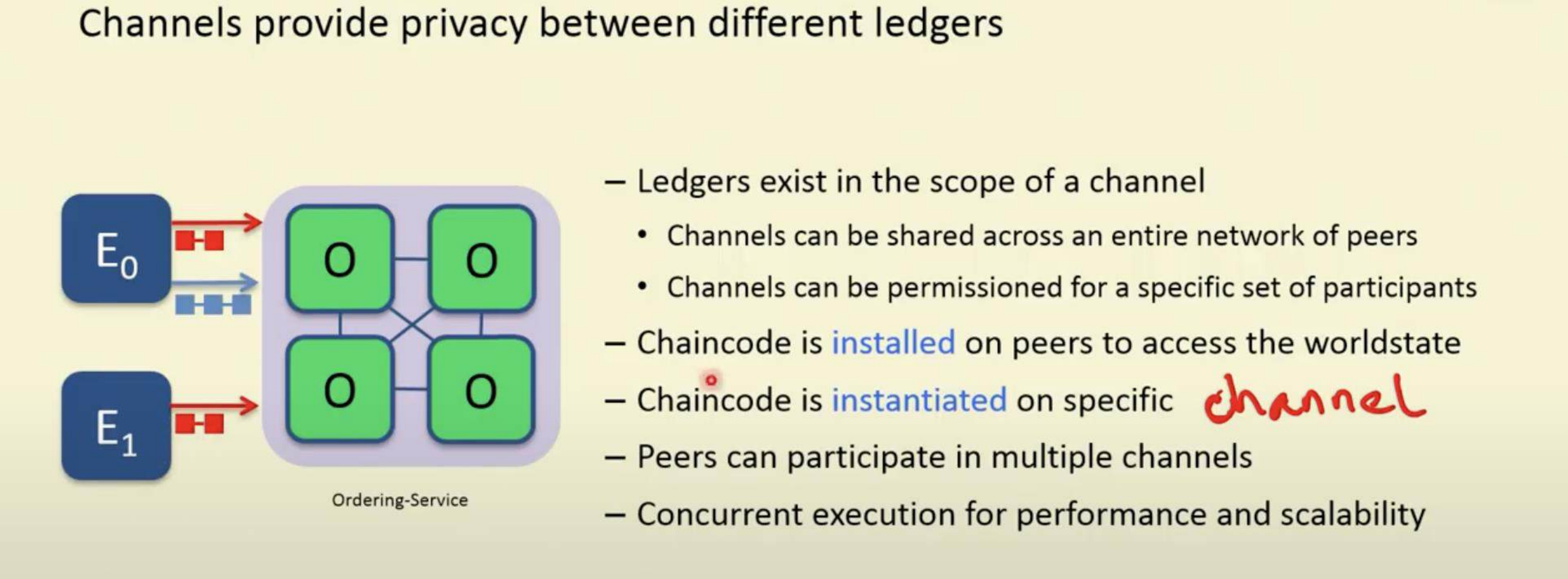
The ordering service packages transactions into blocks to be delivered to peers. Communication with the service is via channels.

Ordering-Service

Different configuration options for the ordering service include:

- SOLO
  - Single node for development
- Kafka : Crash fault tolerant consensus
  - 3 nodes minimum
  - Odd number of nodes recommended

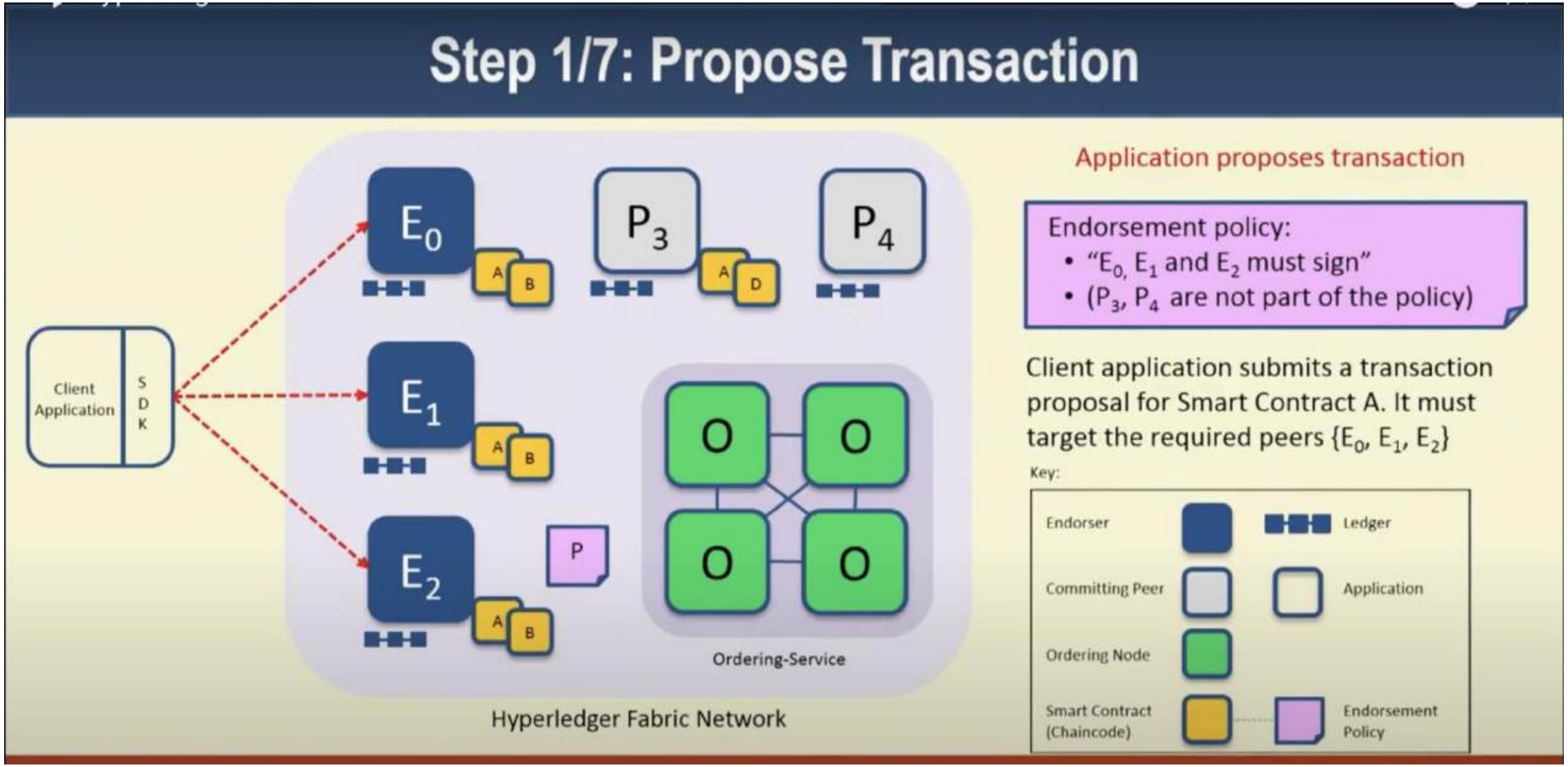
Channels



Channels provide privacy between different ledgers

- Ledgers exist in the scope of a channel
  - Channels can be shared across an entire network of peers
  - Channels can be permissioned for a specific set of participants
- Chaincode is installed on peers to access the worldstate
- Chaincode is instantiated on specific channel
- Peers can participate in multiple channels
- Concurrent execution for performance and scalability

Step 1/7: Propose Transaction



Step 1/7: Propose Transaction

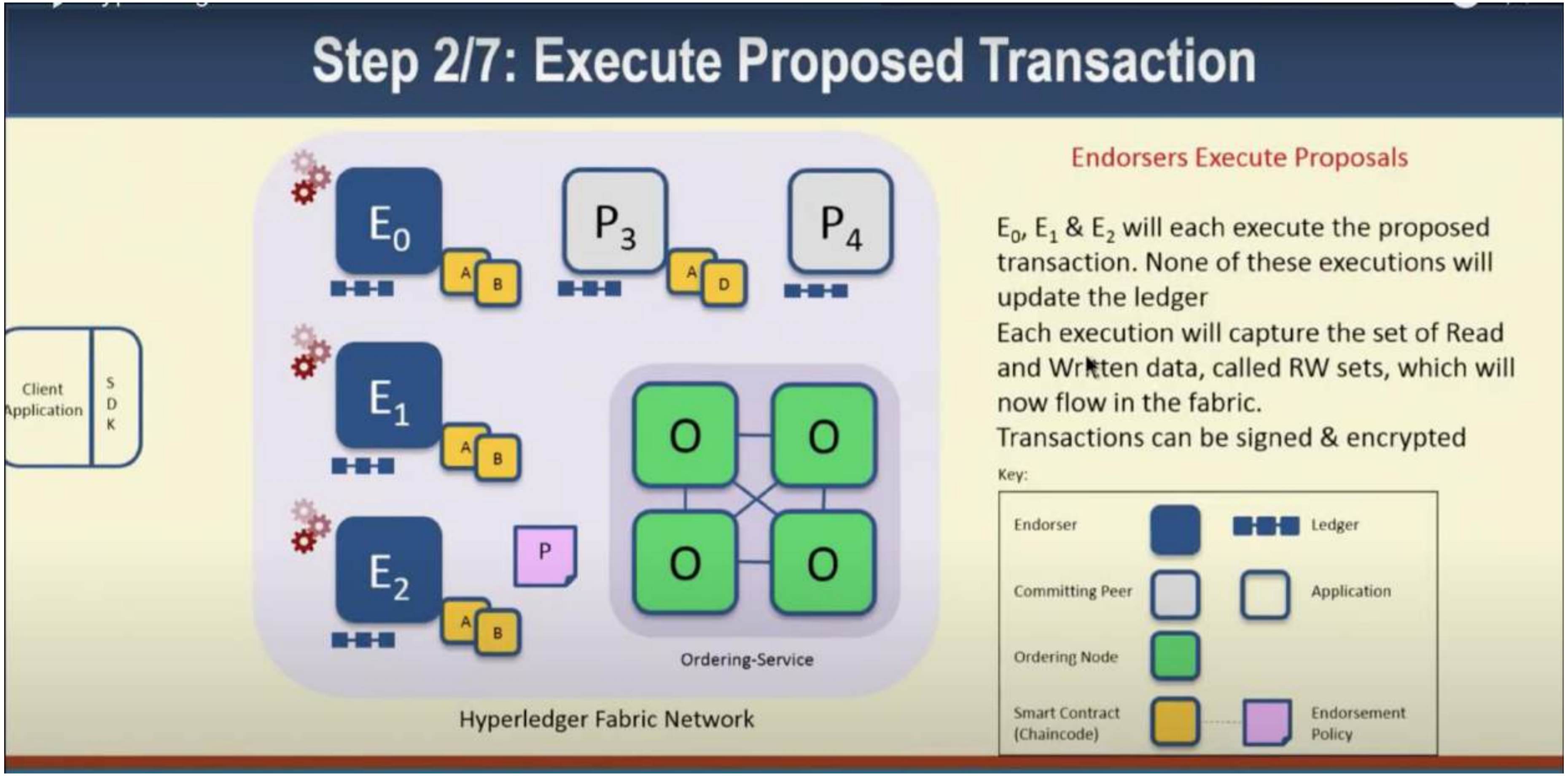
Application proposes transaction

Endorsement policy:

- "E_0, E_1 and E_2 must sign"
- (P_3, P_4 are not part of the policy)

Client application submits a transaction proposal for Smart Contract A. It must target the required peers {E_0, E_1, E_2}

Step 2/7 :Execute Proposed Transaction



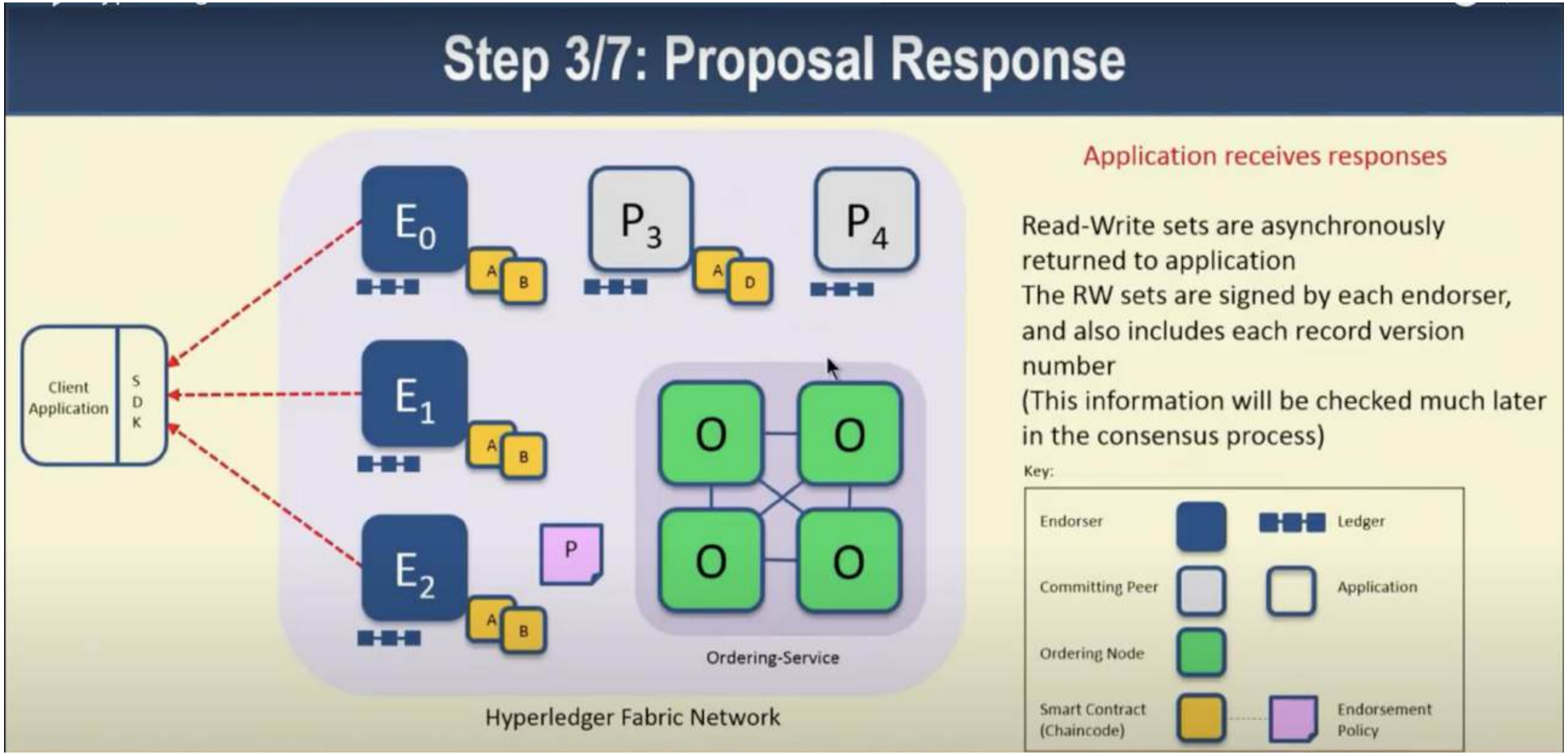
Step 2/7: Execute Proposed Transaction

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



Step 3/7: Proposal Response

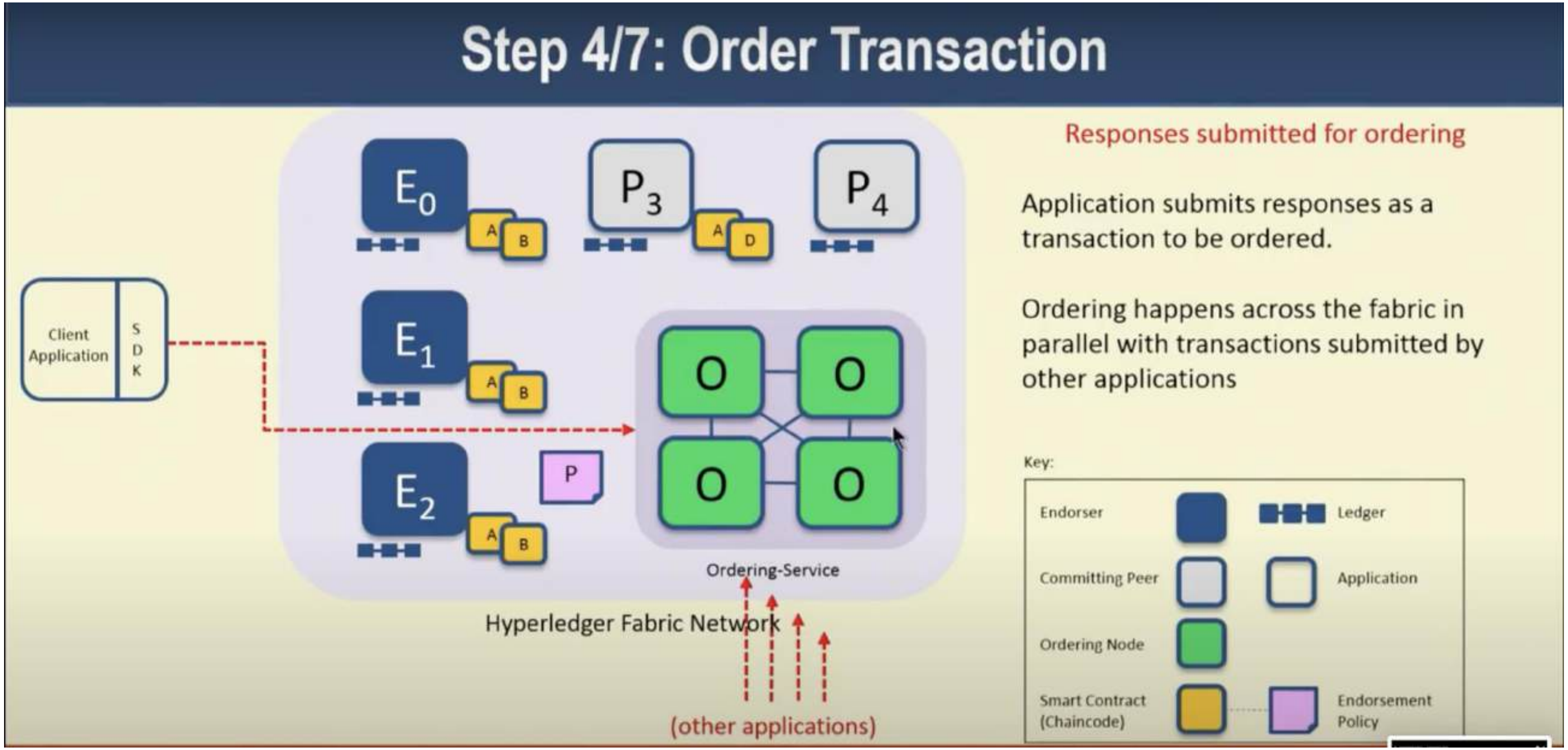
Application receives responses

Read-Write 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)

Step 4/7:Order Transaction



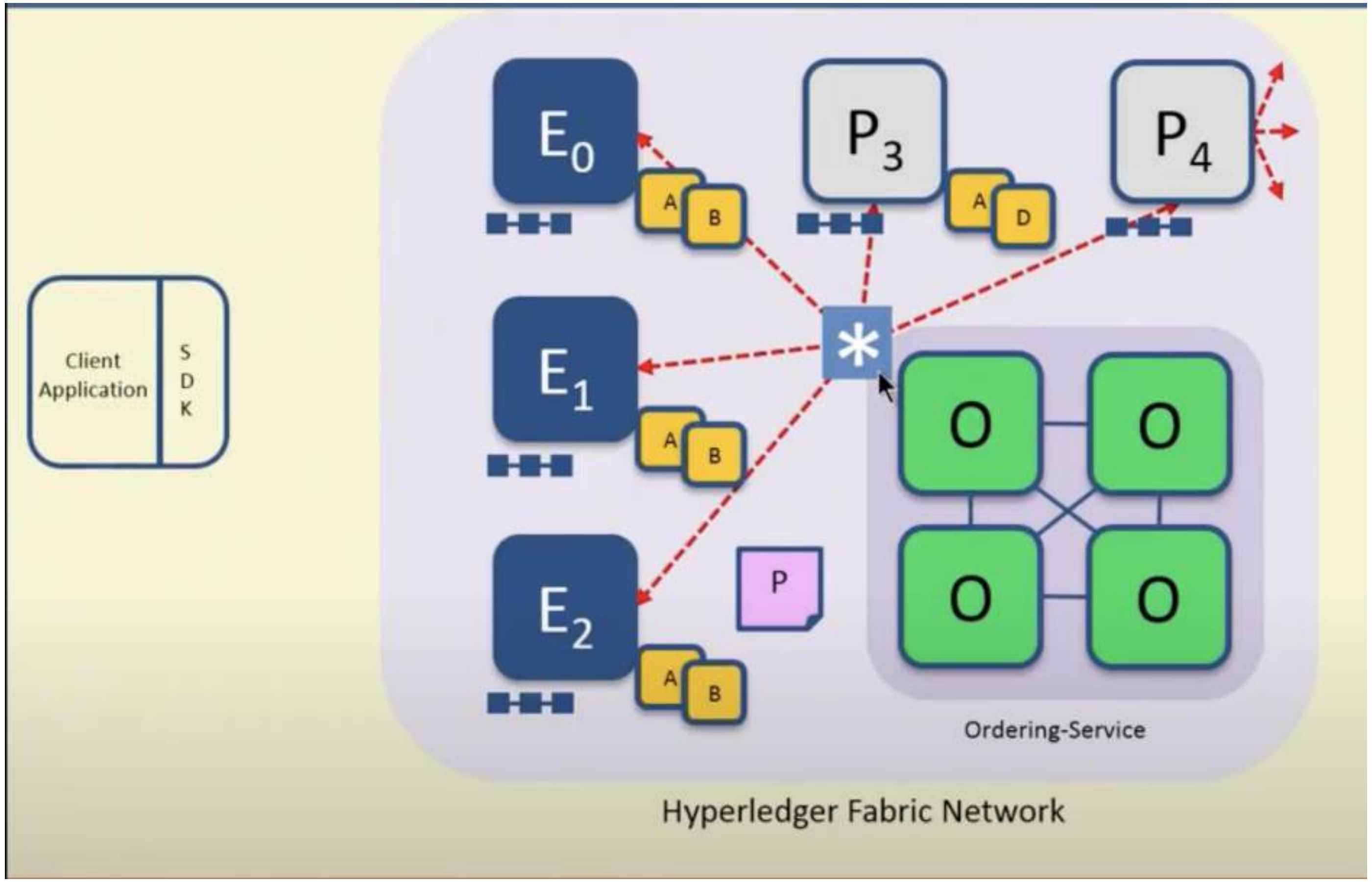
Step 4/7: Order Transaction

Responses submitted for ordering

Application submits responses as a transaction to be ordered.

Ordering happens across the fabric in parallel with transactions submitted by other applications

Step 5/7: Deliver Transaction

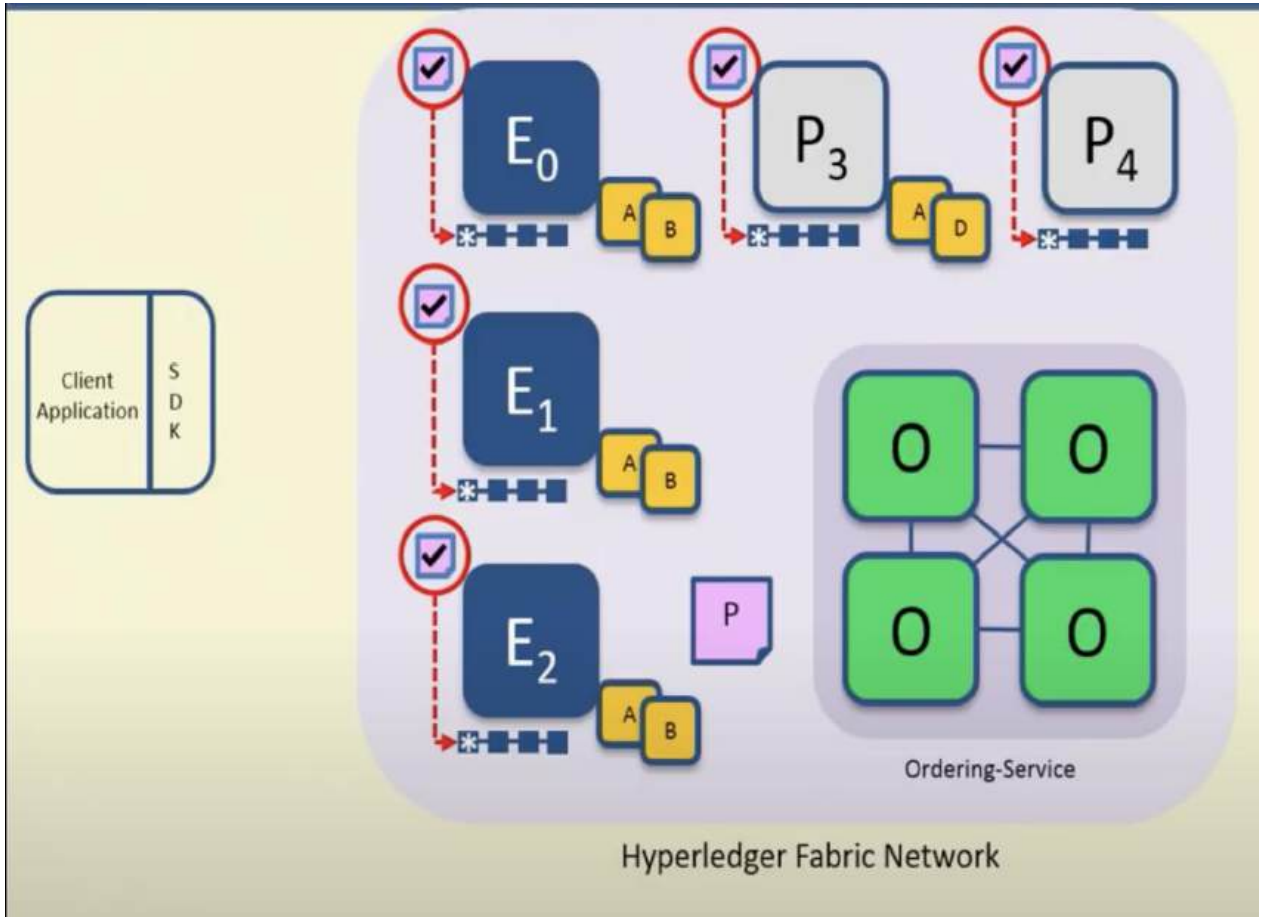


Orderer delivers to committing peers

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)

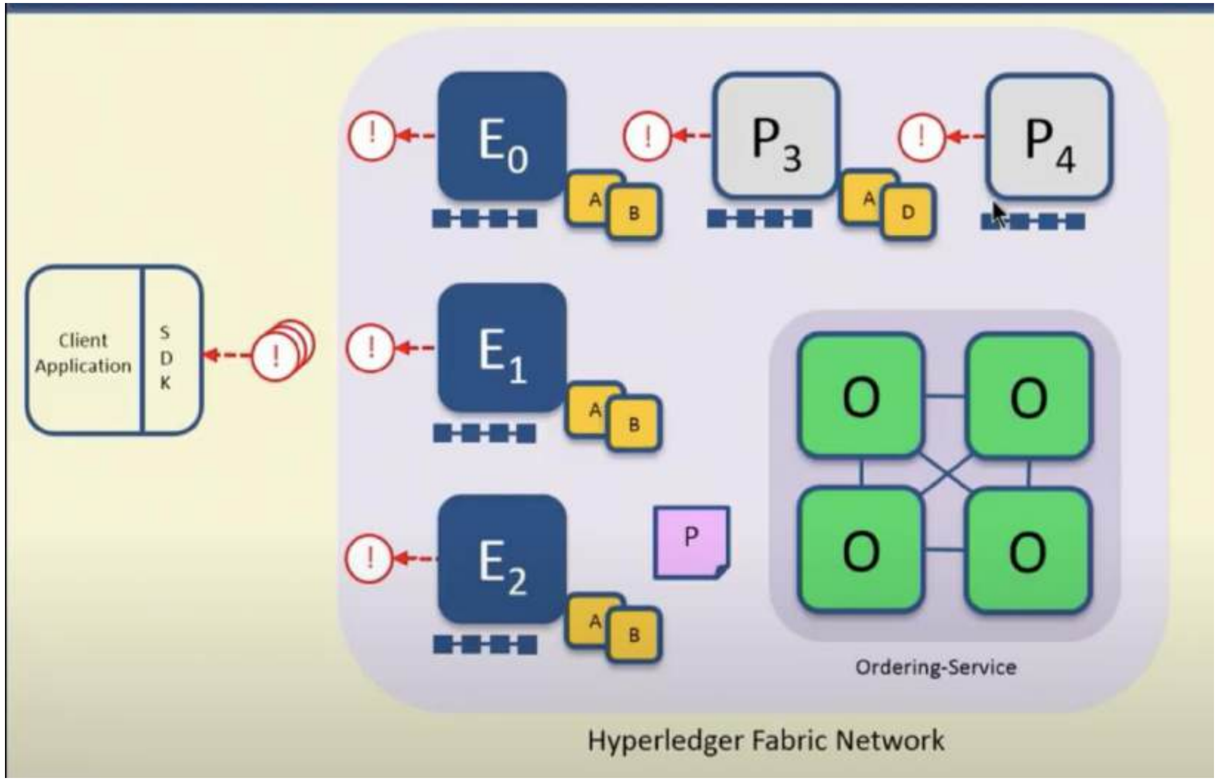
Step 6/7: Validate Transaction



Committing peers validate transactions

Every committing peer validates against the endorsement policy. Also check RW sets are still valid for current world state
Validated transactions are applied to the world state and retained on the ledger
Invalid transactions are also retained on the ledger but do not update world state

Step 7/7 : Notify Transaction



Committing peers notify applications

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 benefits of the Transaction flow

- Better reflect business processes by specifying who endorses transaction.
- Eliminate non deterministic transactions
- Scale the number of participants and transaction output.