Hyperledger Fabric

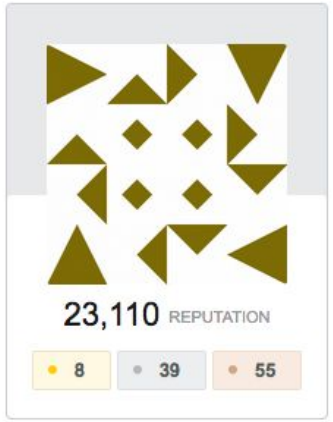
Hands on session – the guide on how to write your own chaincode

Artem Barger

(bartem@il.ibm.com)

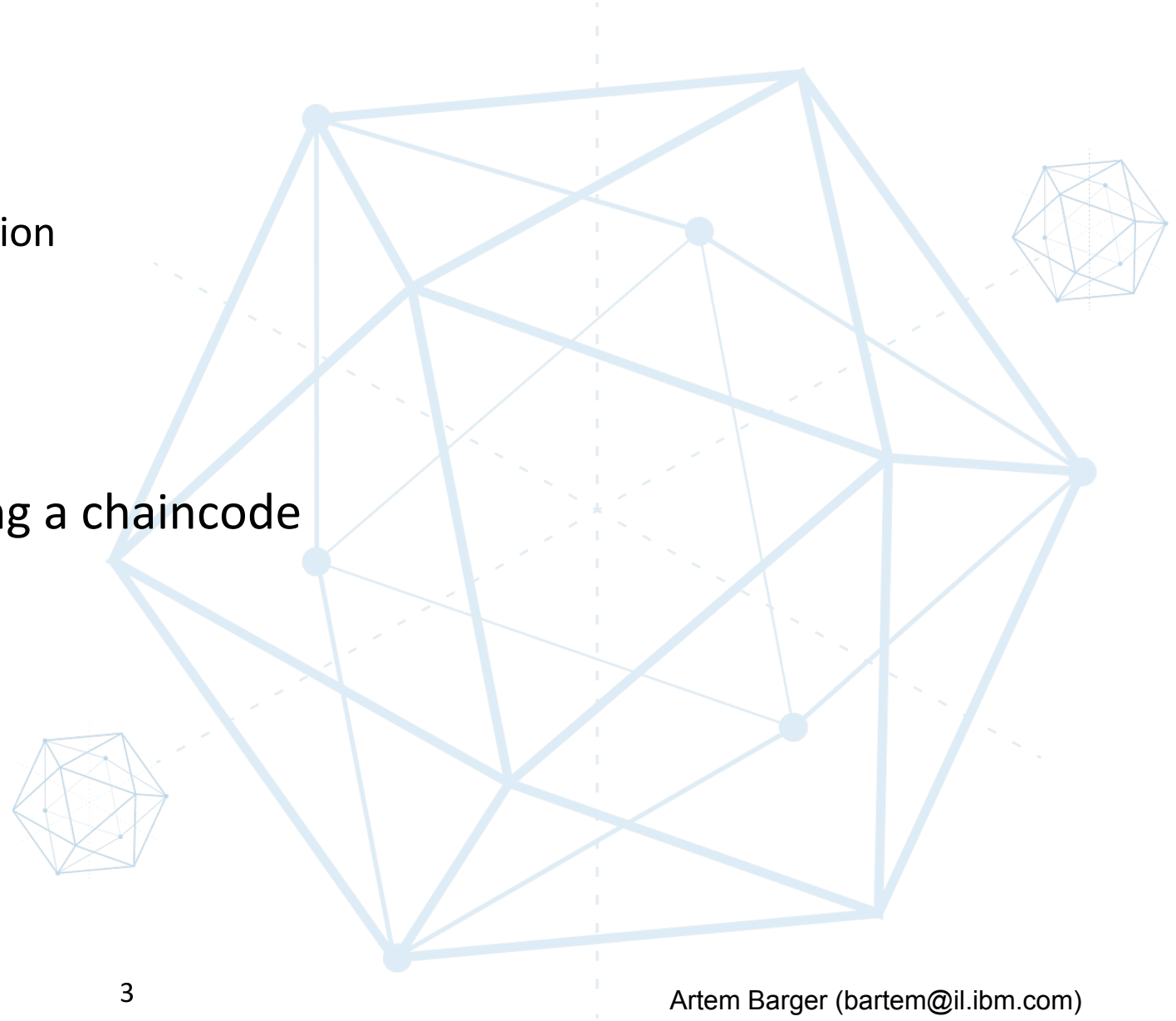
About me…

- IBM Haifa Lab Cloud Foundation Research
- 10+ years of experience in design and development of distributed system
- Maintainer of Linux Foundation Hyperledger Project
- Decent background in Java server side devlopment
- ASF Committer (Apache Commons)



Outline

- Hyperledger Fabric
  - ✓ Basic concepts
    - ❑ Endorsement, ordering, validation
    - ❑ Channels
- Setting Up a Fabric Network
  - ✓ Create/Join a channel
  - ✓ Writing/Installing/Instantiating a chaincode
    - ❑ Endorsement policies
  - ✓ Invoke transactions
- Demo
- QA



Artem Barger (bartem@il.ibm.com)

What is the HYPERLEDGER PROJECT?

Open source collaborative effort to advance cross-industry blockchain technologies.

Hosted by The Linux Foundation

Global collaboration including leaders in finance, banking, IoT, supply chain, manufacturing and technology

Artem Barger (bartem@il.ibm.com)

Hyperledger Project Members

Premier

accenture
AIRBUS
CME Group
Deutsche Börse Group
Digital Asset
DTCC
FUJITSU
HITACHI
Inspire the Next
IBM
intel
J.P.Morgan
R
万达·飞凡科技

General

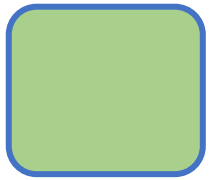
Associate

Artem Barger (bartem@il.ibm.com)

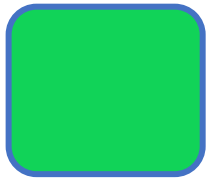
Endorsement, Ordering and Validation

Nodes and roles

Committing Peer: Maintains ledger and state. Commits transactions. May hold smart contract (chaincode).

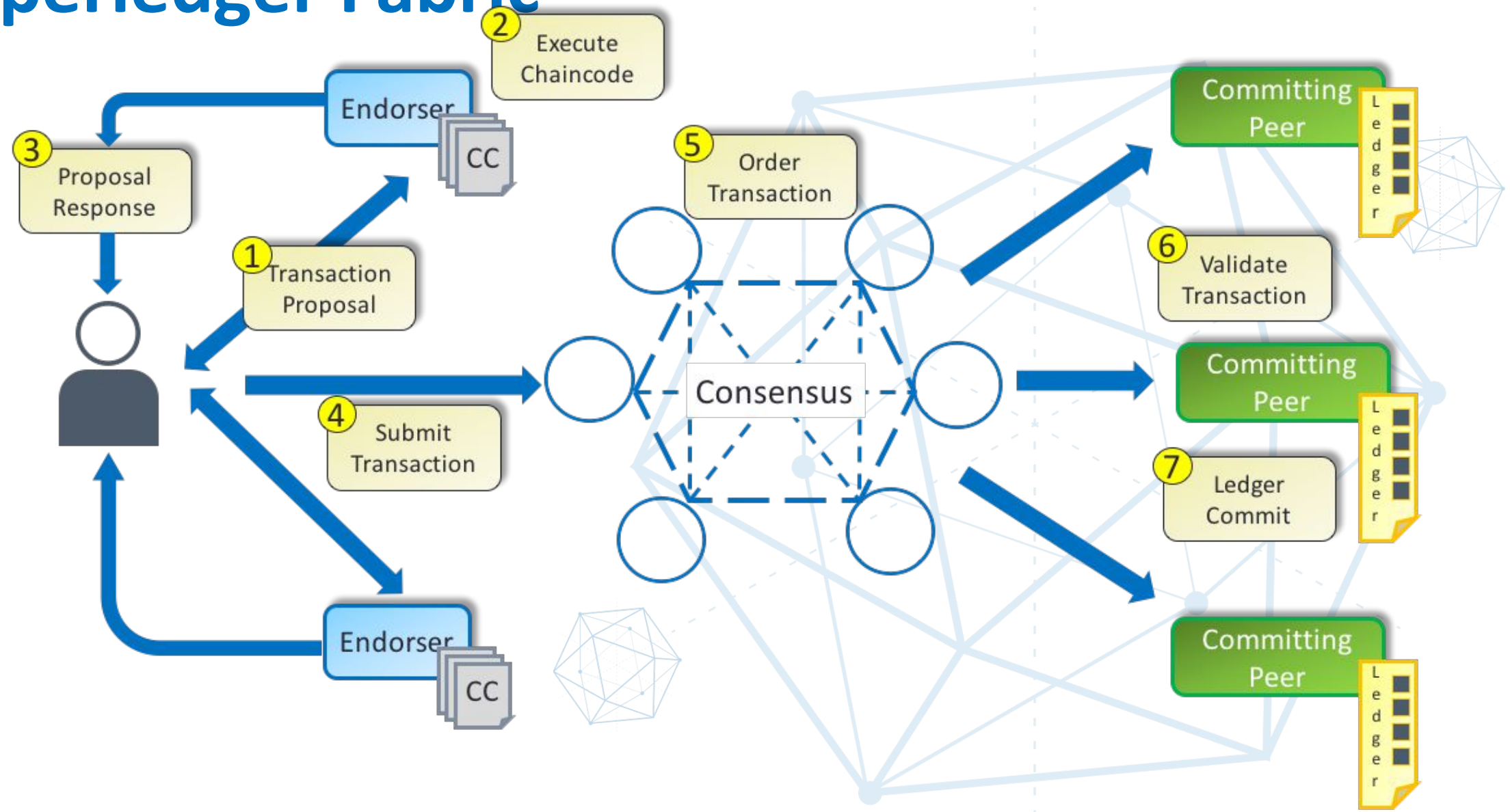


Endorsing Peer: Specialized committing peer that receives a transaction proposal for endorsement, responds granting or denying endorsement. Must hold smart contract



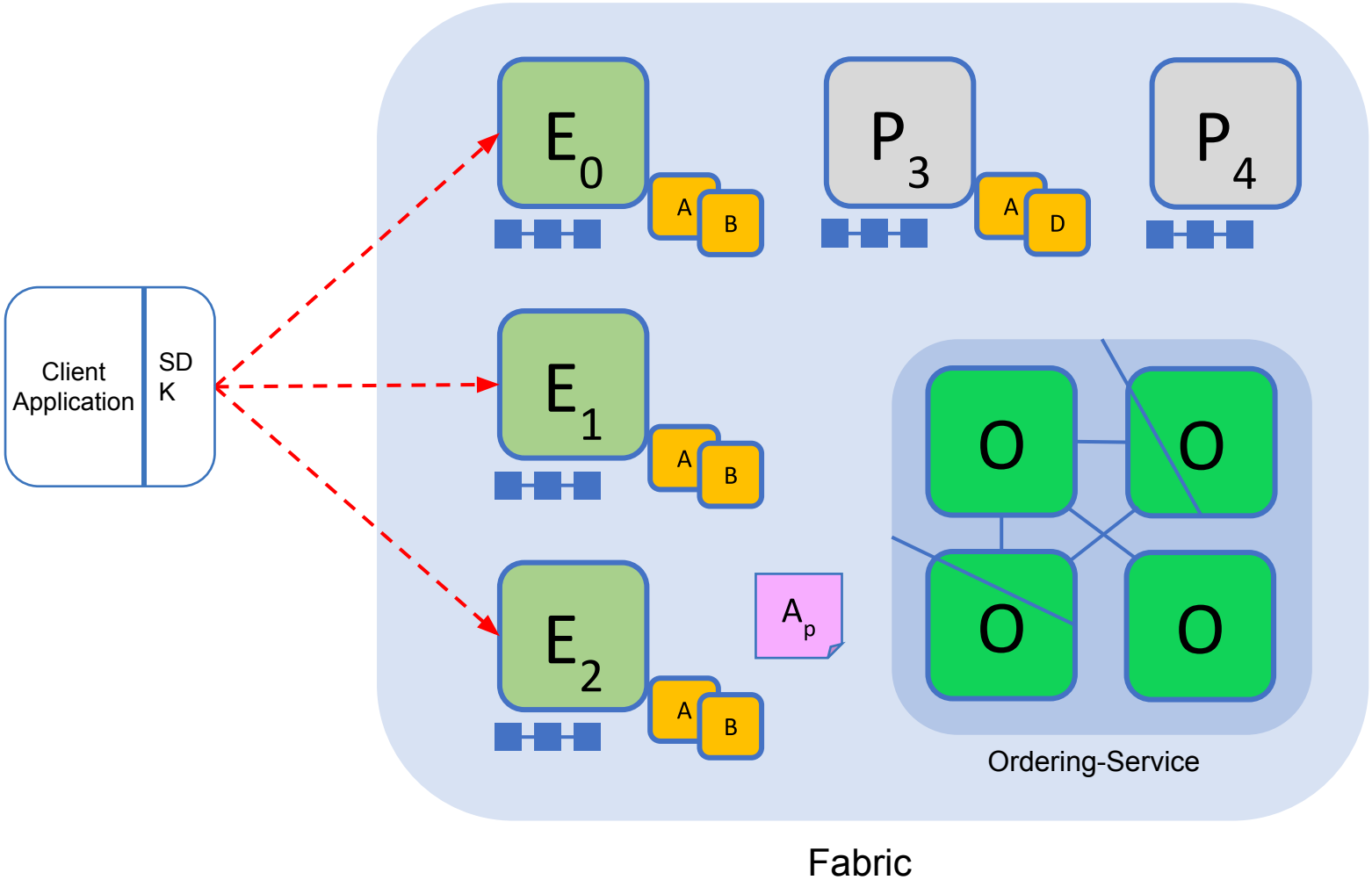
Ordering Nodes (service): Approves the inclusion of transaction blocks into the ledger and communicates with committing and endorsing peer nodes. Does not hold smart contract. Does not hold ledger.

Hyperledger Fabric



Artem Barger (bartem@il.ibm.com)

Sample transaction: Step 1/7 – Propose transaction

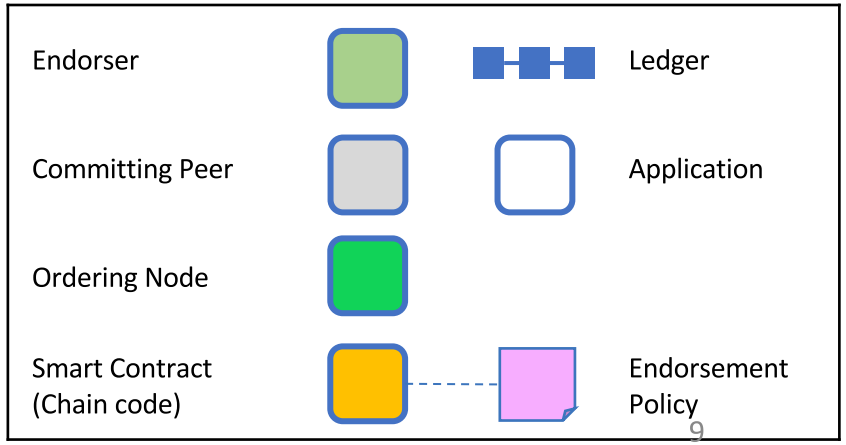


Application proposes transaction

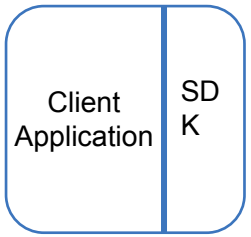
Endorsement policy:
- "E_0, E_1 and E_2 must sign"
- (P_3, P_4 are not part of the policy)

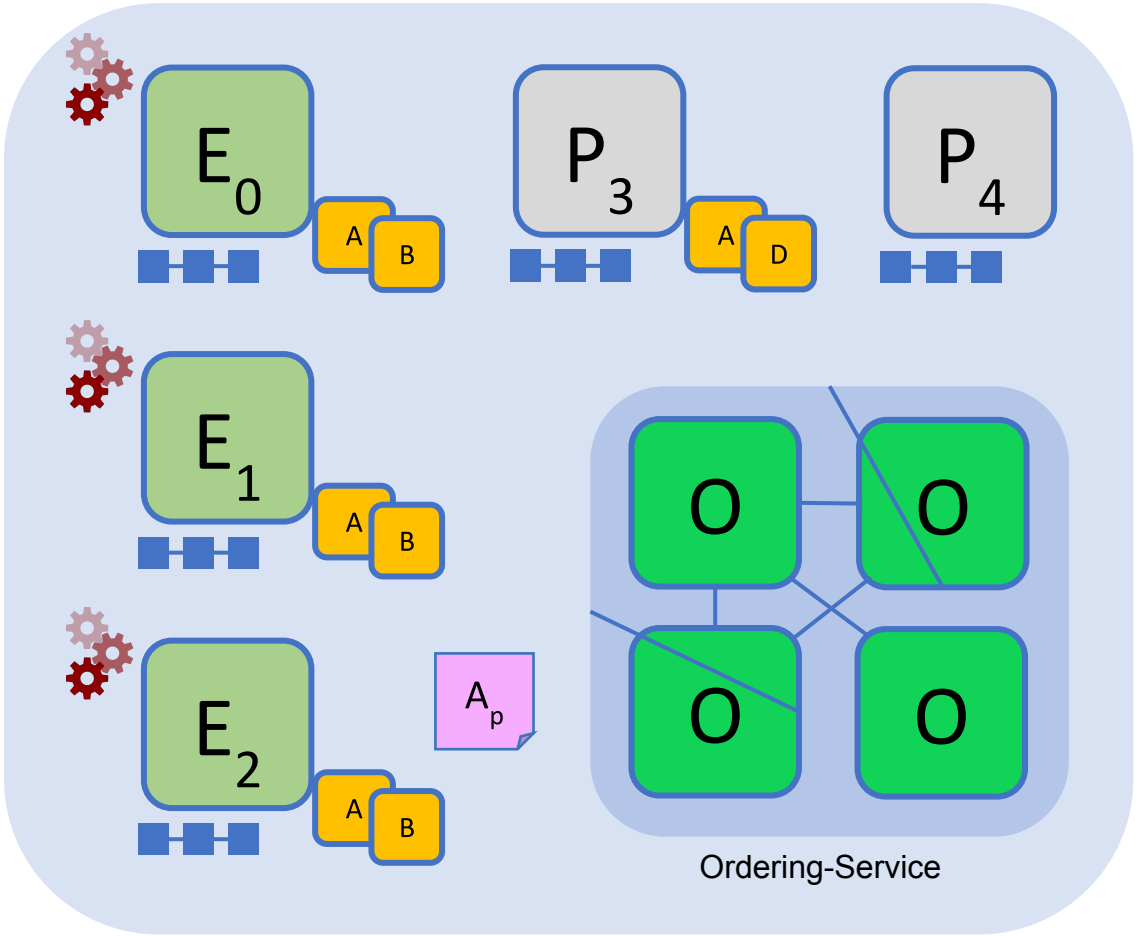
Client application submits a transaction proposal for Smart Contract A. It must target the required peers {E_0, E_1, E_2}

Key:



Sample transaction: Step 2/7 – Execute proposal





Fabric

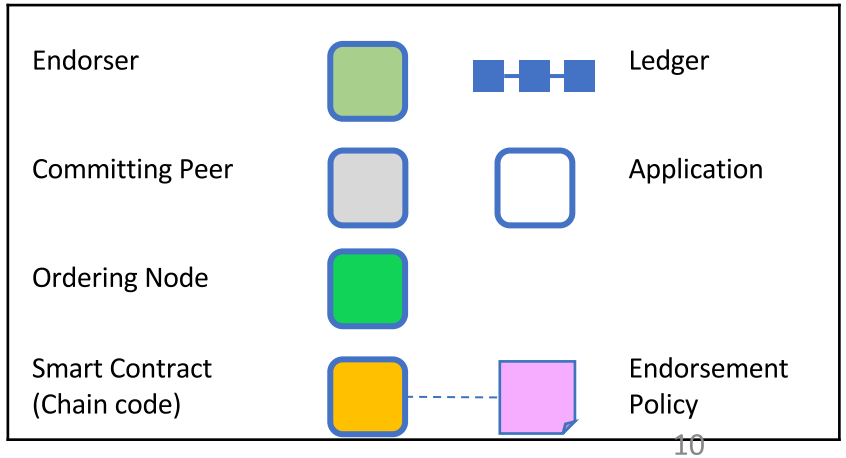
Endorsers Execute Proposals

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

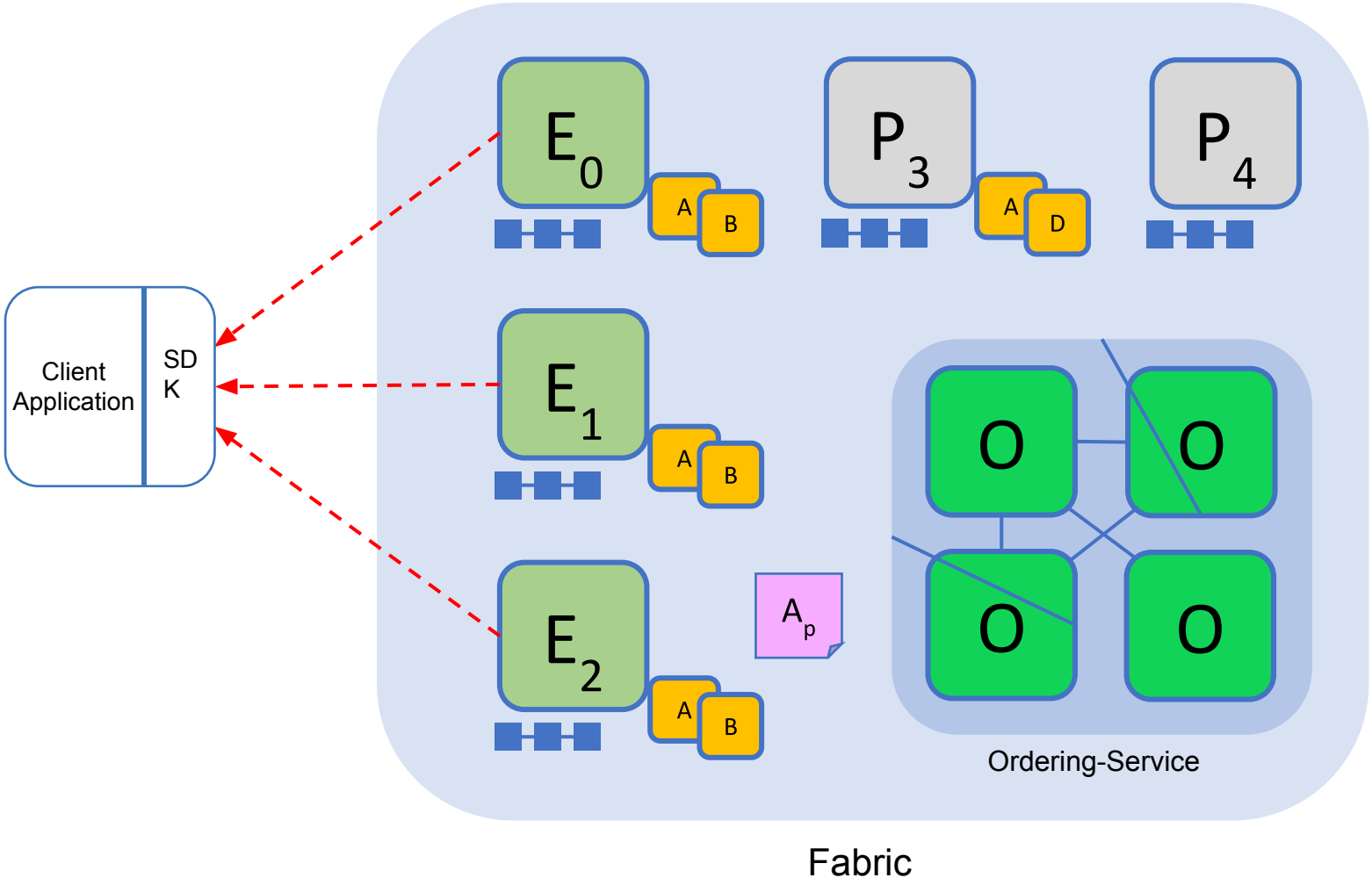
Each execution will capture the set of Read and Written data, called RW sets, which will now flow in the fabric.

Transactions can be signed & encrypted

Key:



Sample transaction: Step 3/7 – Proposal Response



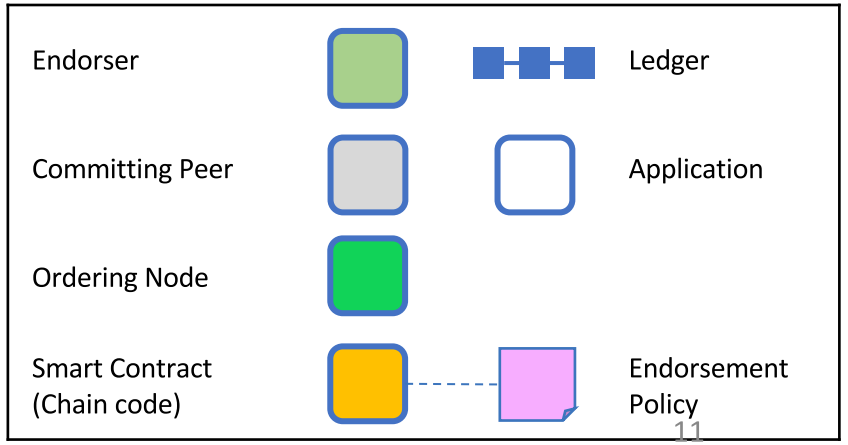
Application receives responses

RW sets are asynchronously returned to application

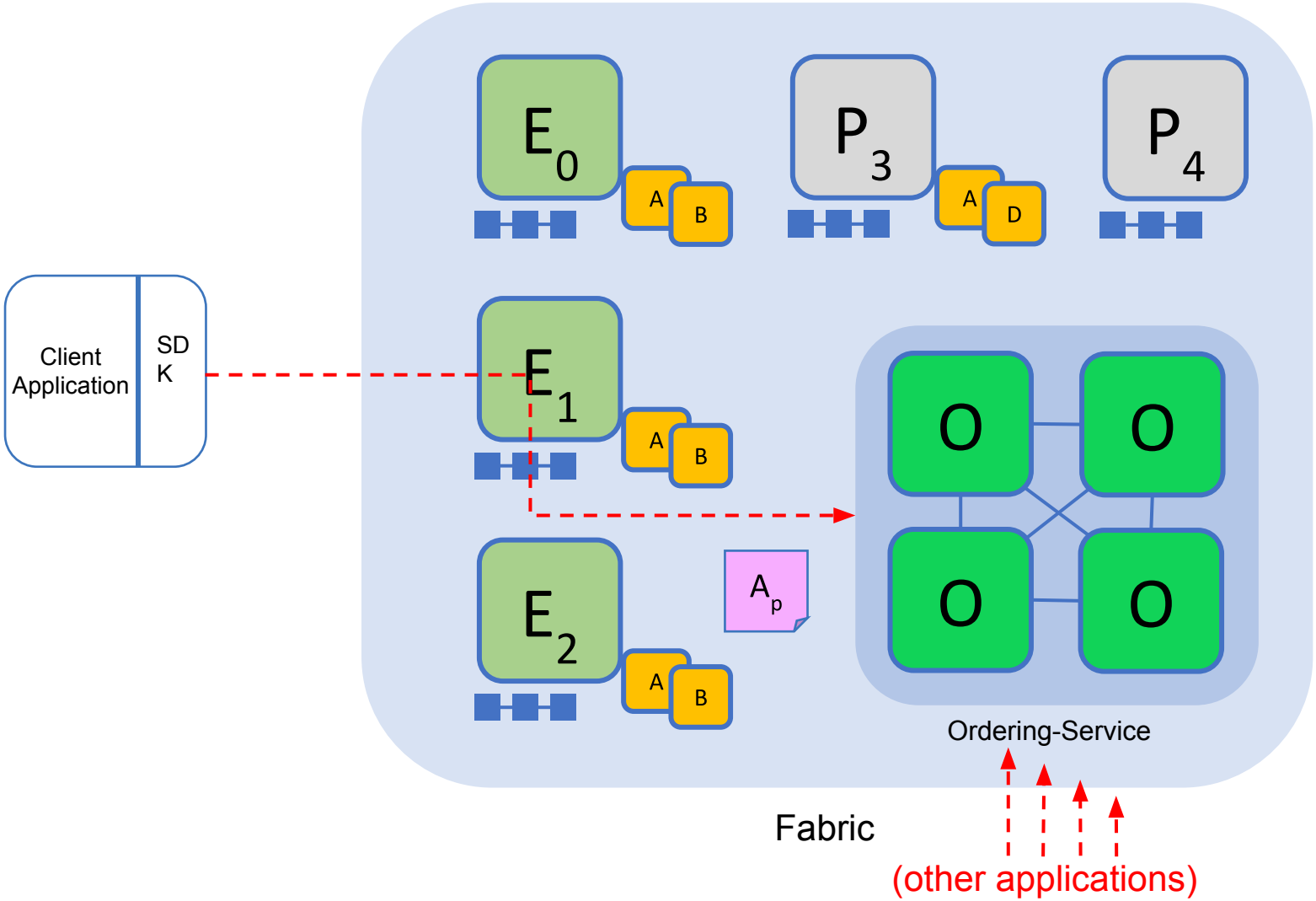
The RW sets are signed by each endorser, and also includes each record version number

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

Key:



Sample transaction: Step 4/7 – Order Transaction

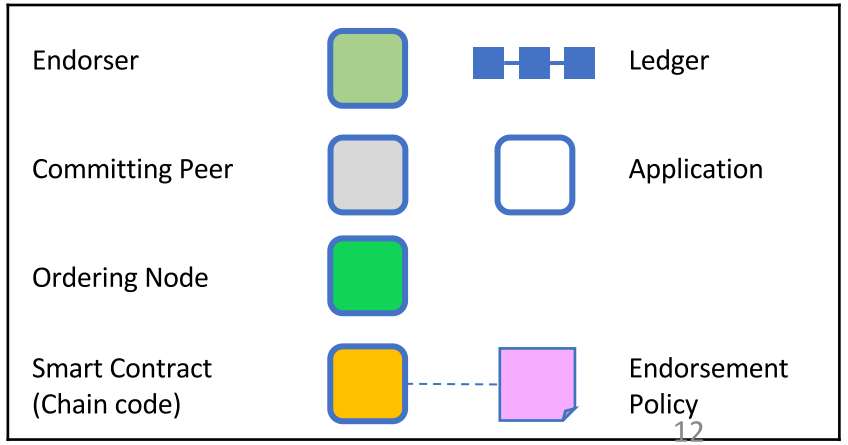


Application submits responses for ordering

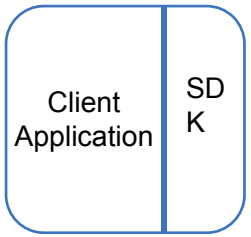
Application submits responses as a transaction to be ordered.

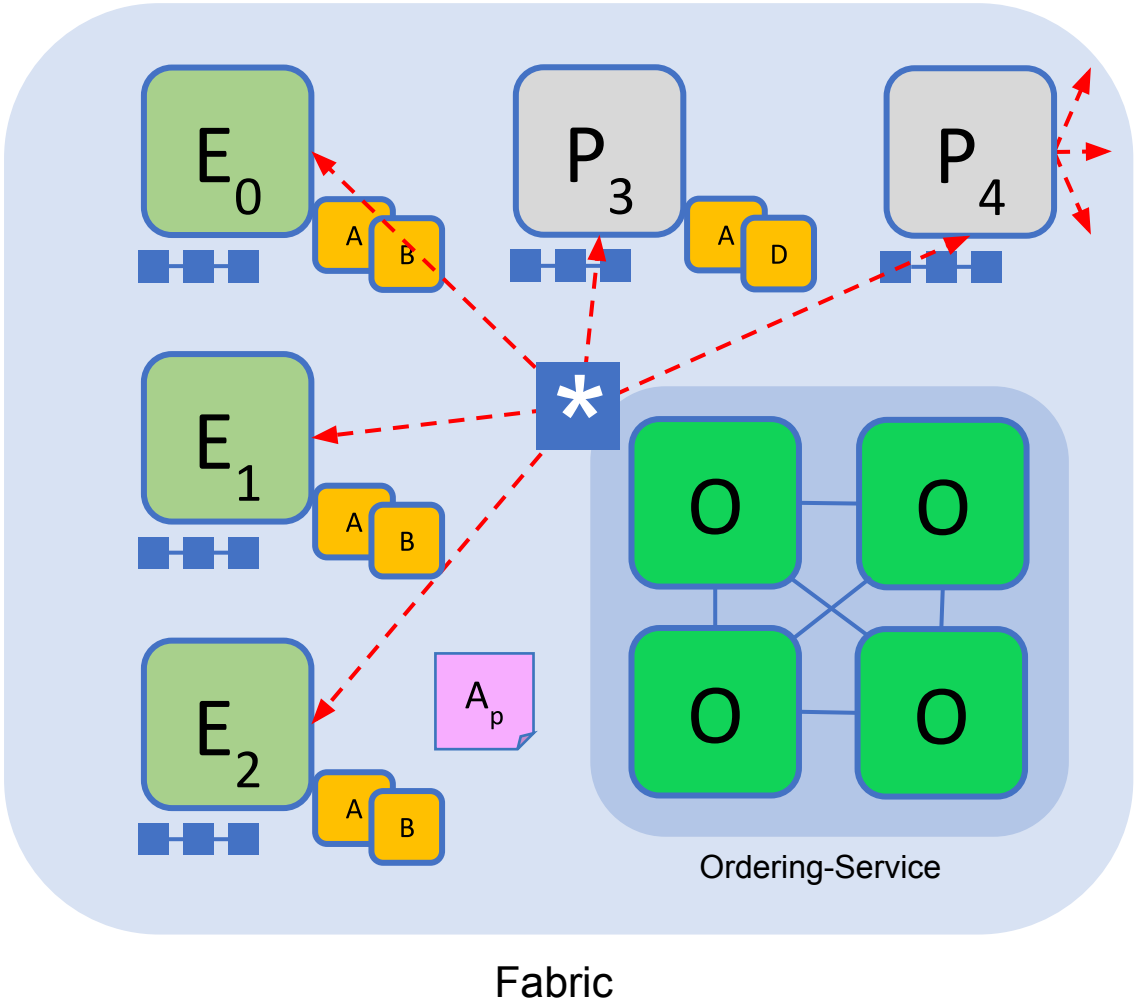
Ordering happens across the fabric in parallel with transactions submitted by other applications

Key:



Sample transaction: Step 5/7 – Deliver Transaction





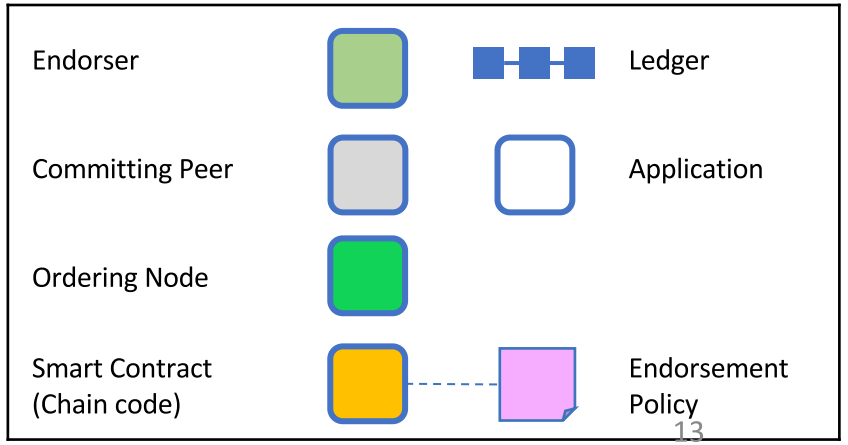
Orderer delivers to all committing peers

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

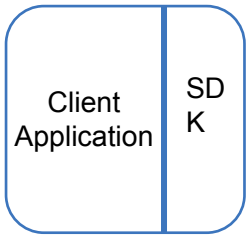
Different ordering algorithms available:

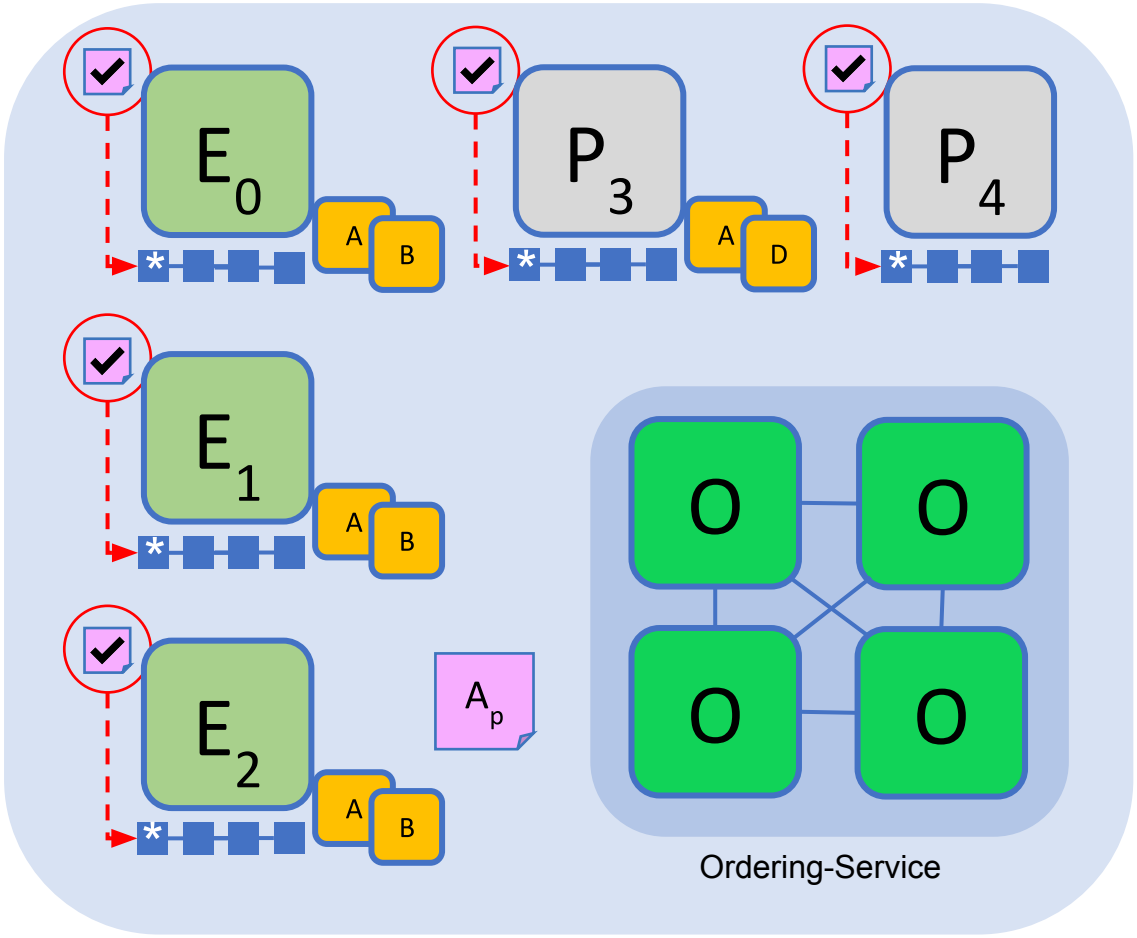
- SOLO (Single node, development)
- Kafka (Crash fault tolerance)
- SBFT (Byzantine fault tolerance)

Key:



Sample transaction: Step 6/7 – Validate Transaction





Fabric

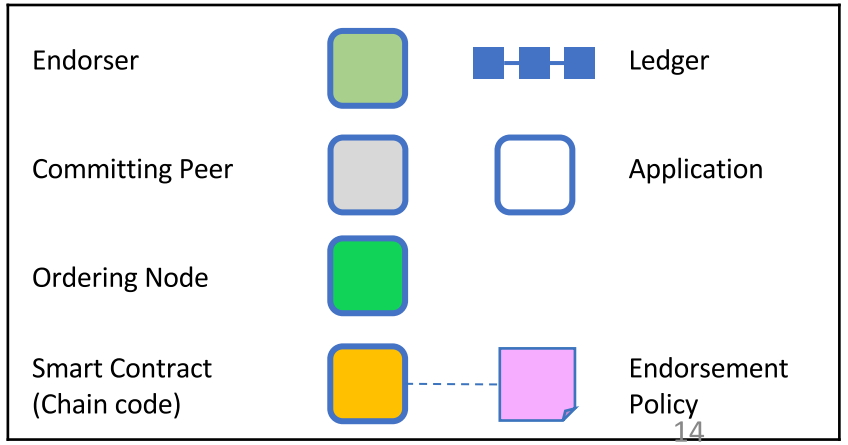
Committing peers validate transactions

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

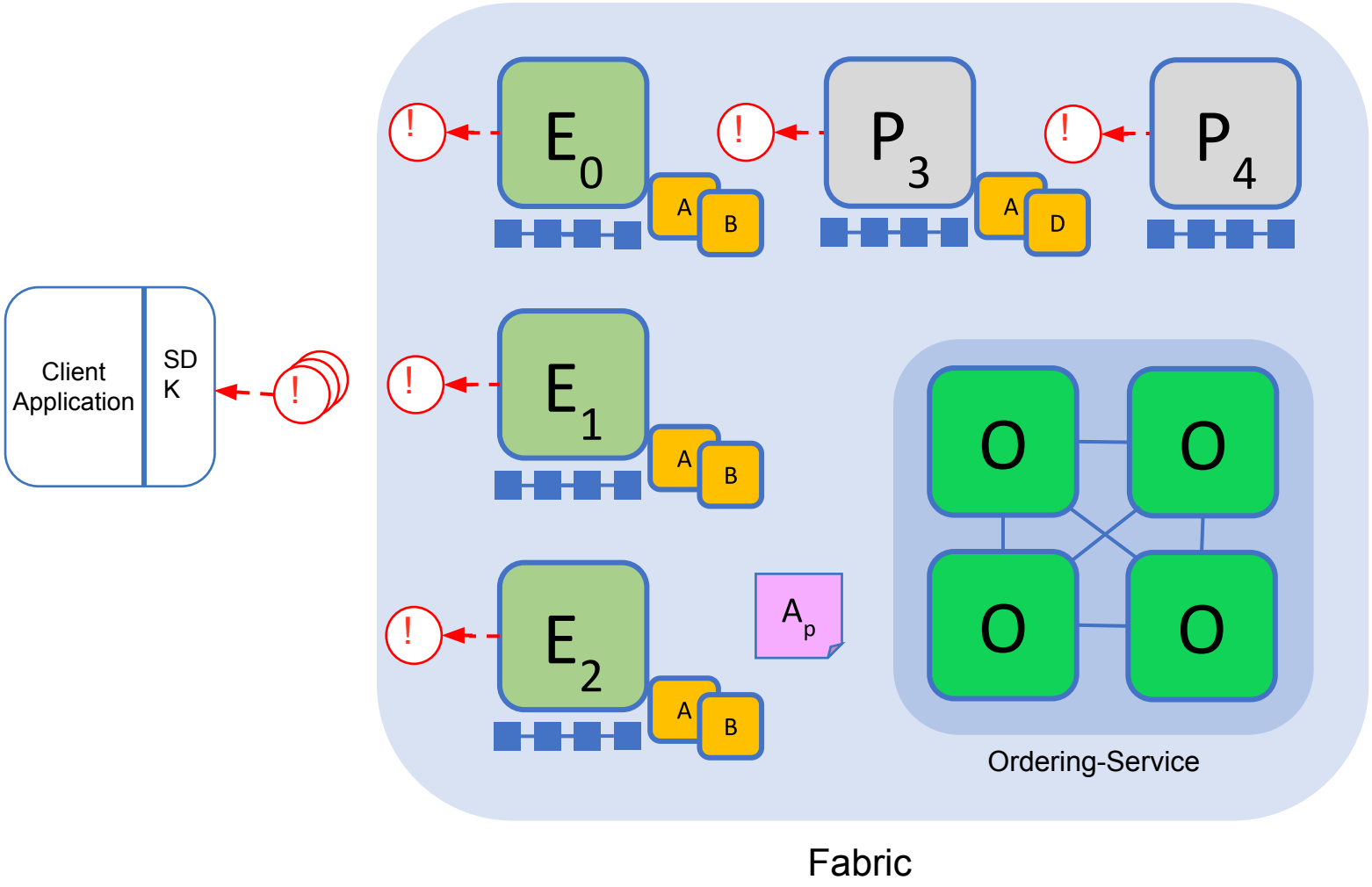
Validated transactions are applied to the world state and retained on the ledger

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

Key:



Sample transaction: Step 7/7 – Notify Transaction

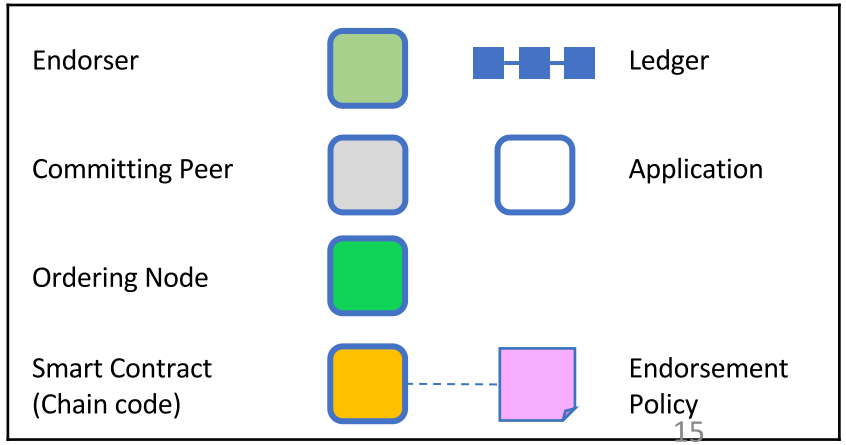


Committing peers notify applications

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

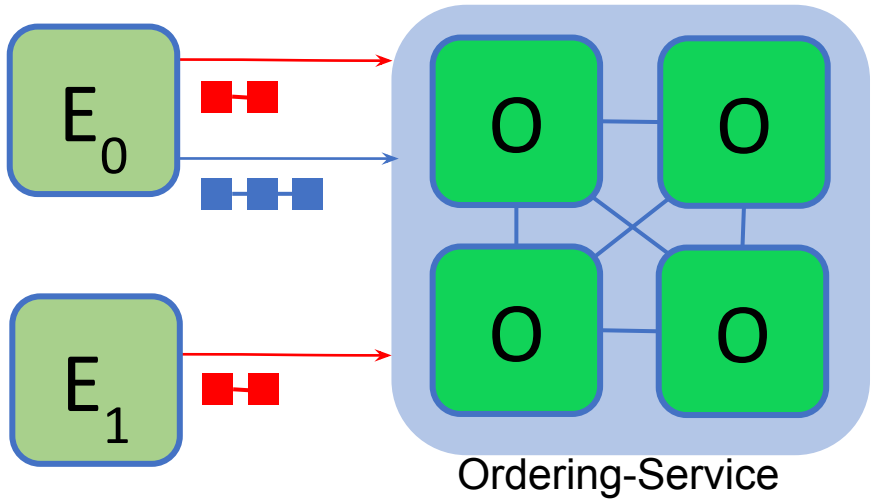
Key:



Channels

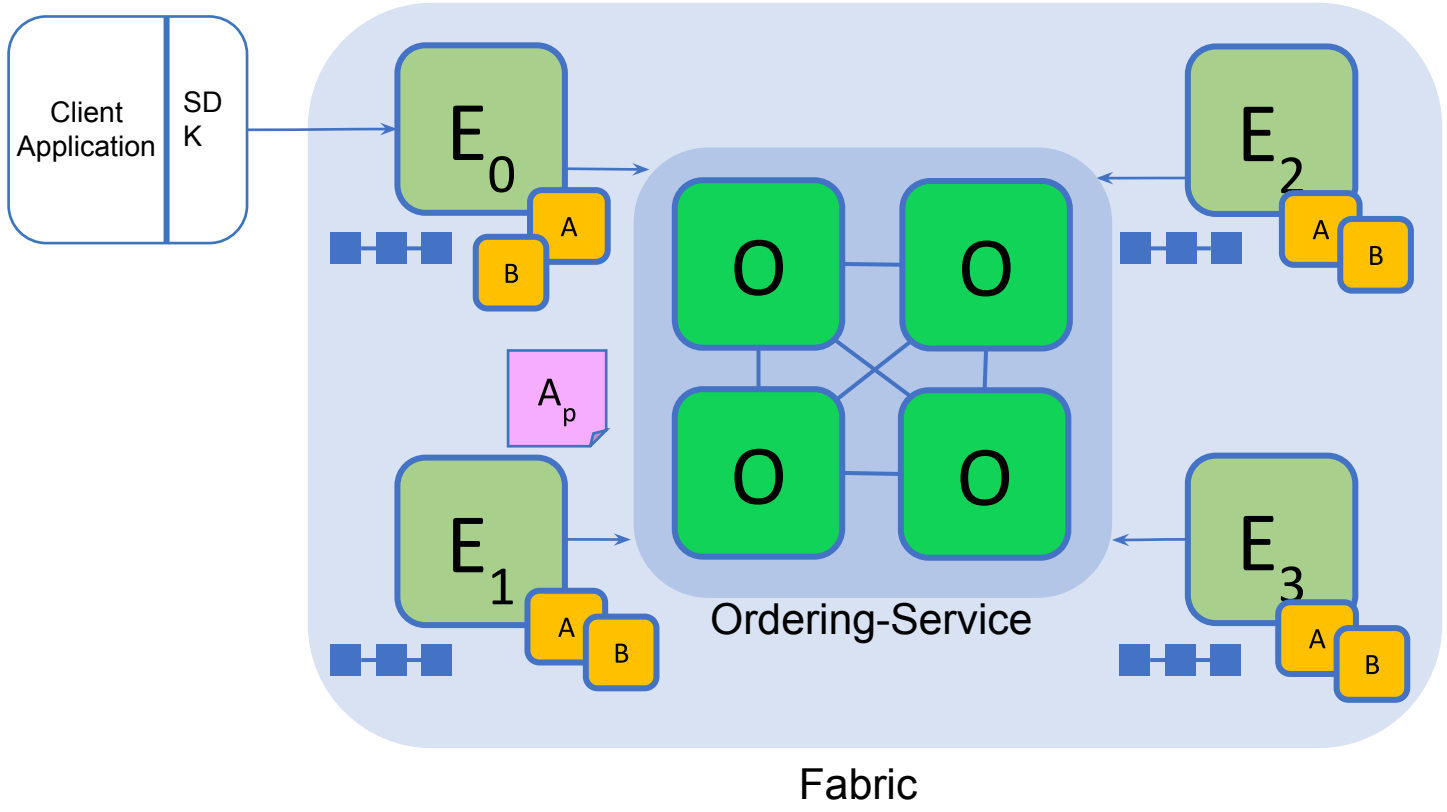
Channels

Separate channels isolate transactions on different ledgers

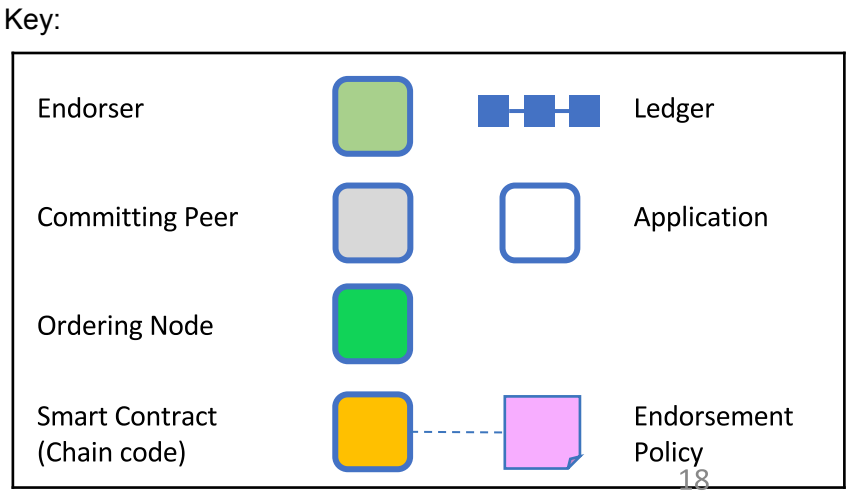


- Chaincode is installed on peers that need to execute business logic and participate in endorsement process
- Chaincode is instantiated on specific channels for specific peers
- Ledgers exist in the scope of a channel
  - Ledgers can be shared across an entire network of peers
  - Ledgers can be included only on a specific set of participants
- Peers can participate in multiple channels
- Concurrent execution for performance and scalability

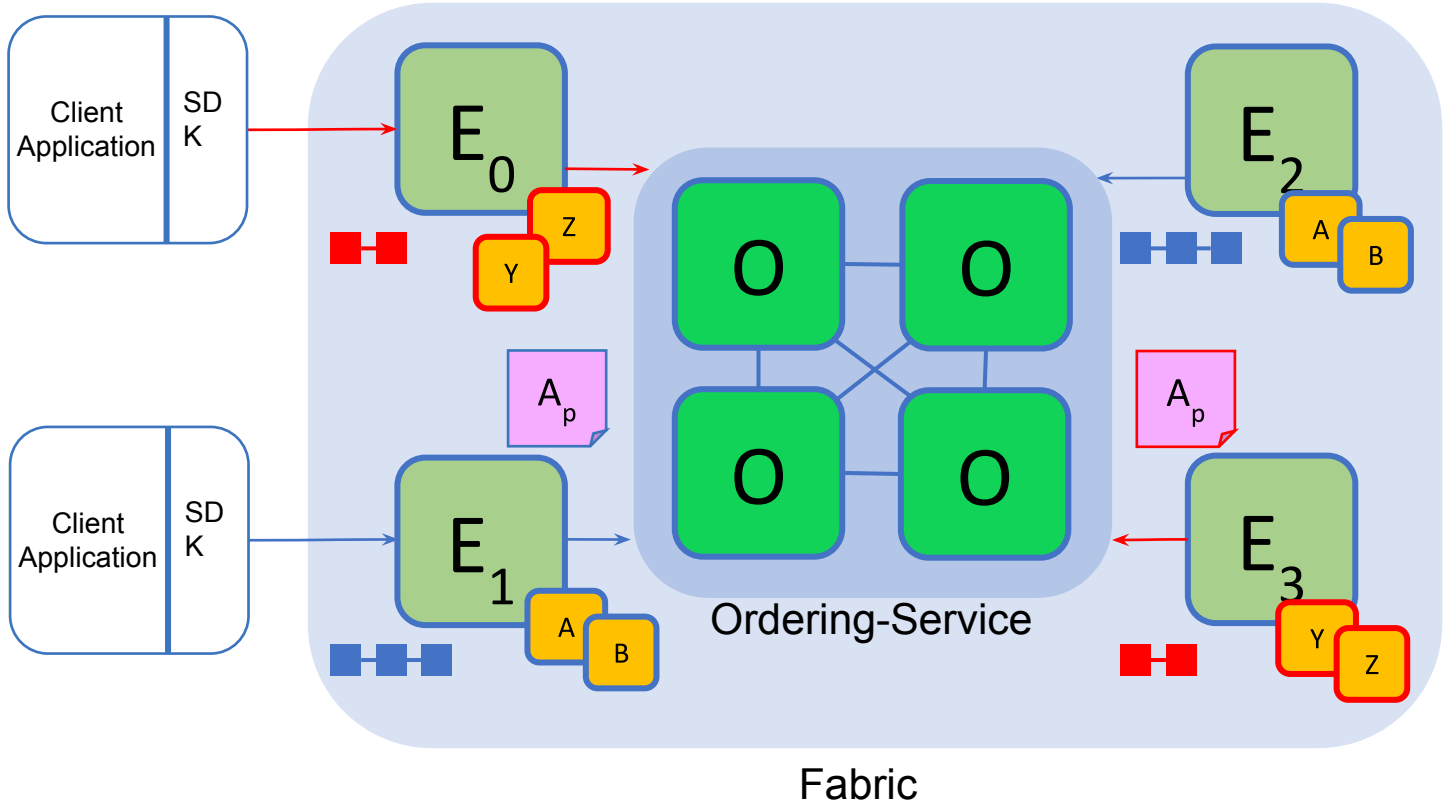
Single Channel Endorsement



- All peers connect to the same channel (blue).
- All peers consider the same chaincodes for execution and maintain the same ledger
- Endorsement by peers E_0, E_1, E_2 and E_3

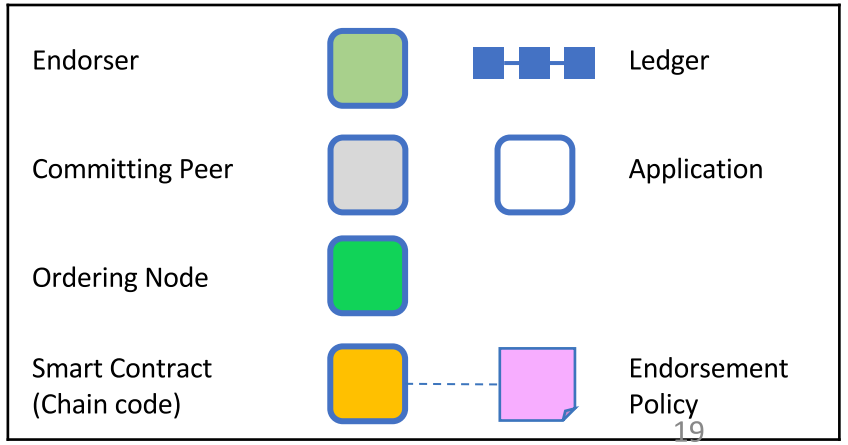


Multi Channel & Chaincode Endorsement



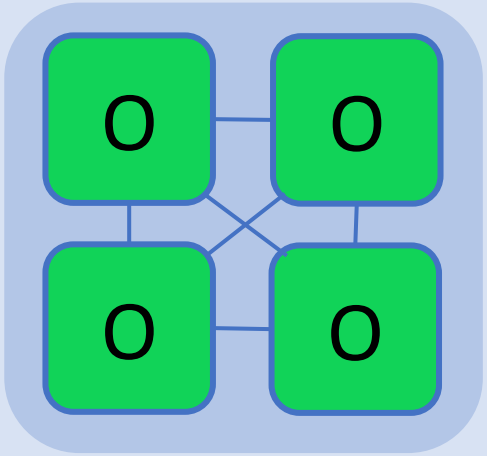
- Peers E_0 and E_3 connect to the red channel for chaincodes Y and Z
- Peers E_1 and E_2 connect to the blue channel for chaincodes A and B

Key:



Setting Up a Fabric Network

Bootstrapping the Network (1/6) – Configure & start Ordering Service

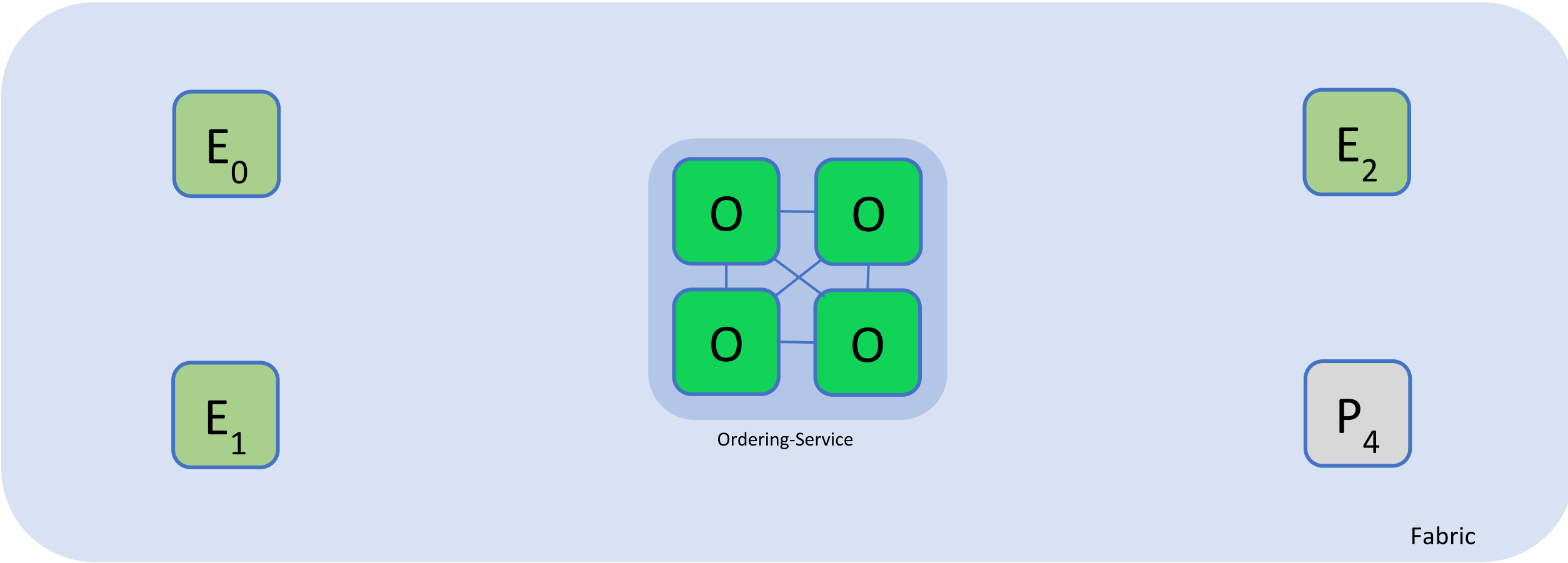


Ordering-Service

Fabric

- An Ordering Service is **configured** and started for other network peers to use
  `$ docker-compose [-f orderer.yml] ...`

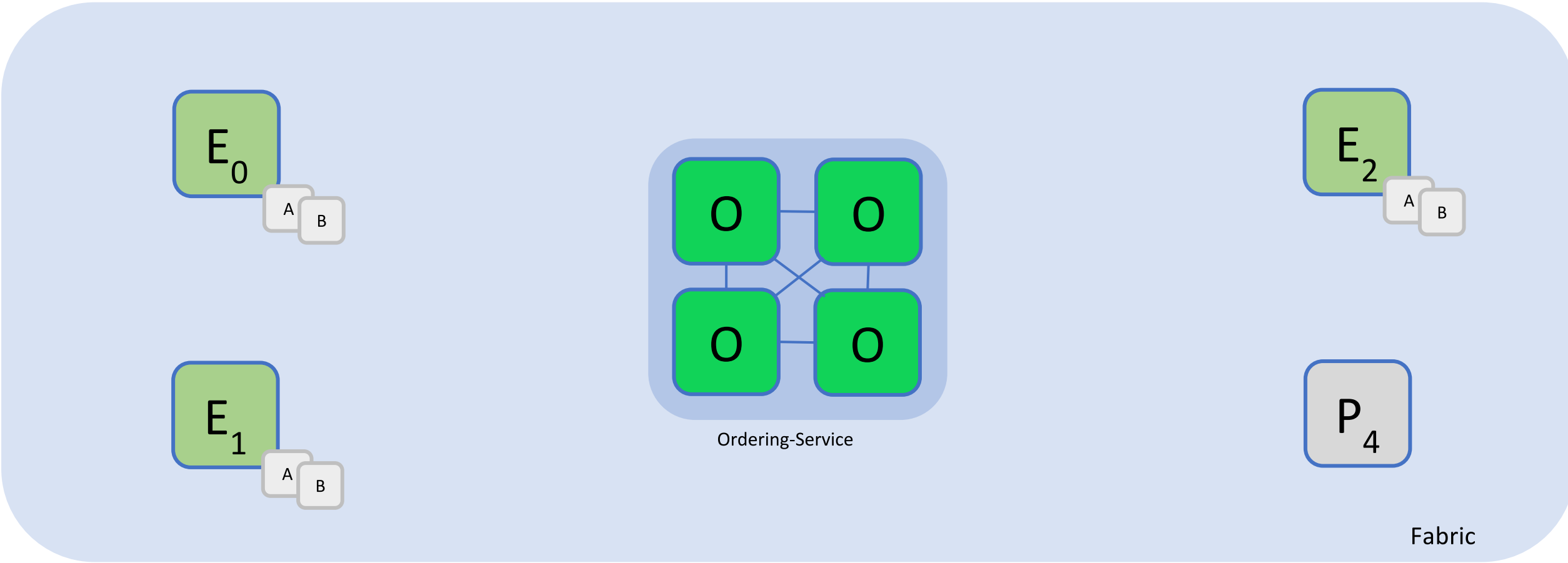
Bootstrapping the Network (2/6) – Configure and Start Peer Nodes



- A peer is configured and **started** for each Endorser or Committer in the network

```
$ peer node start ...
```

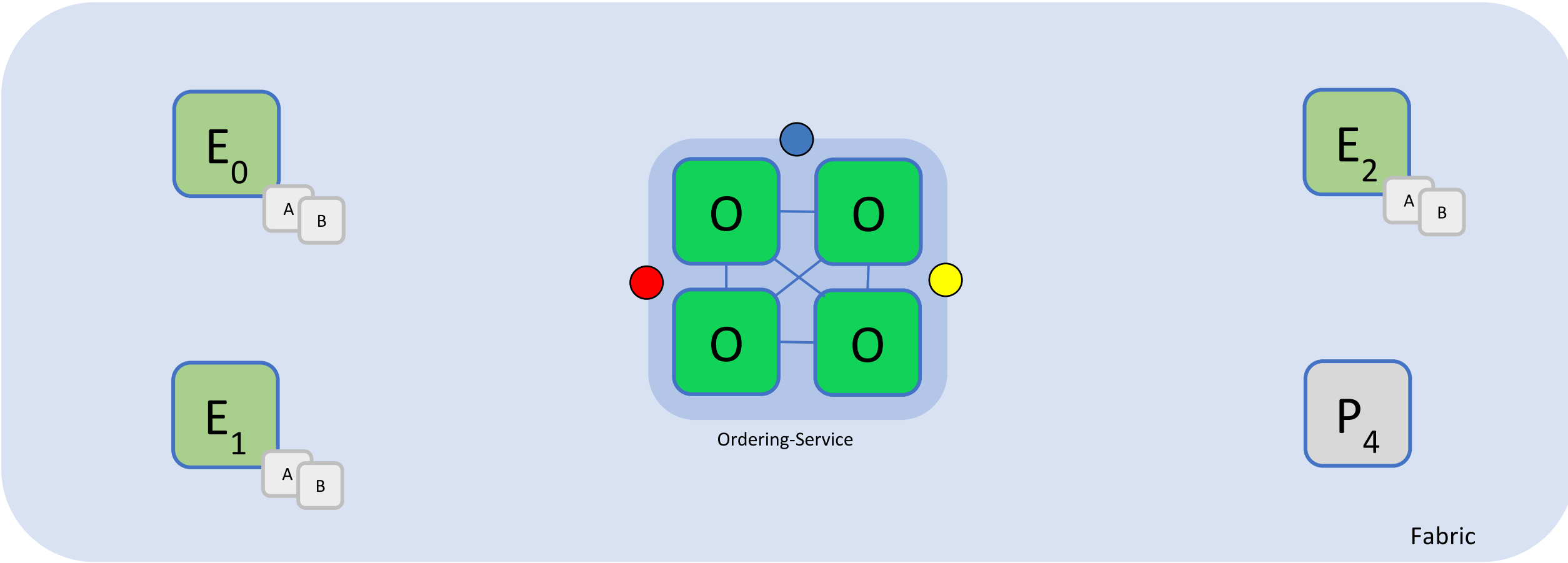
Bootstrapping the Network (3/4) – Install Chaincode



- Chaincode is **installed** onto each Endorsing Peer that needs to execute it

```
$ peer chaincode install ...
```

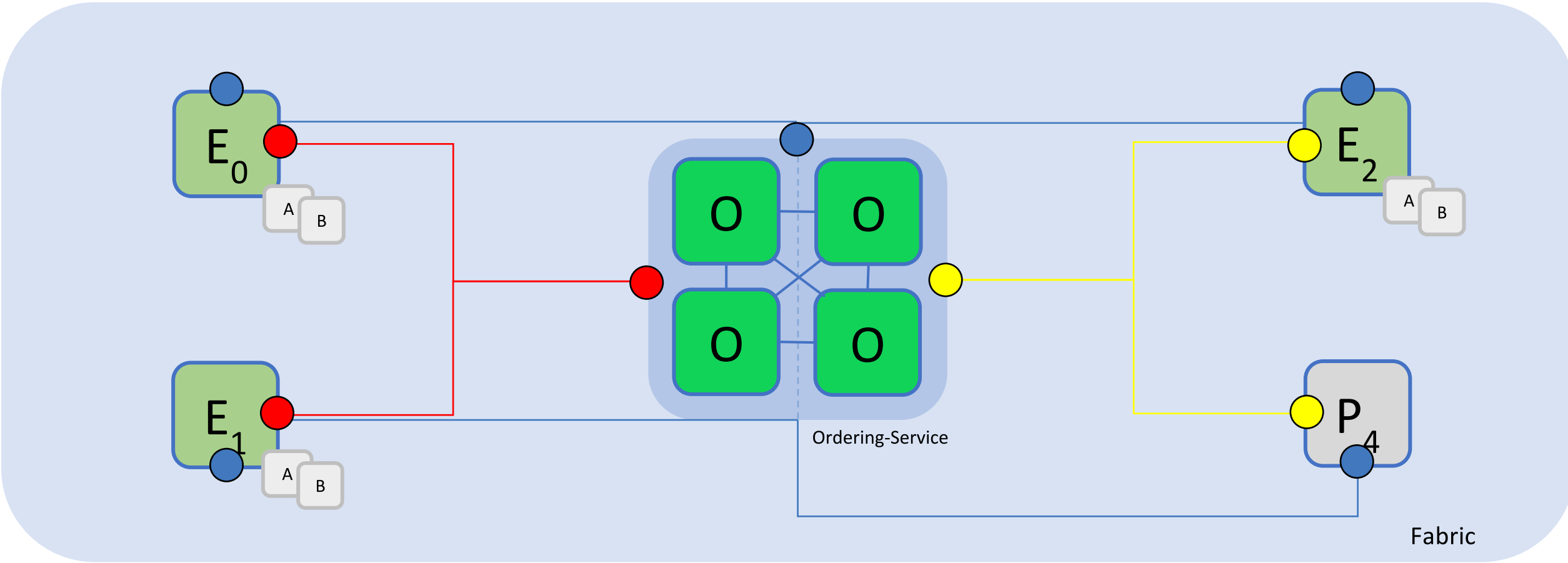
Bootstrapping the Network (4/6) – Create Channels



- Channels are **created** on the ordering service

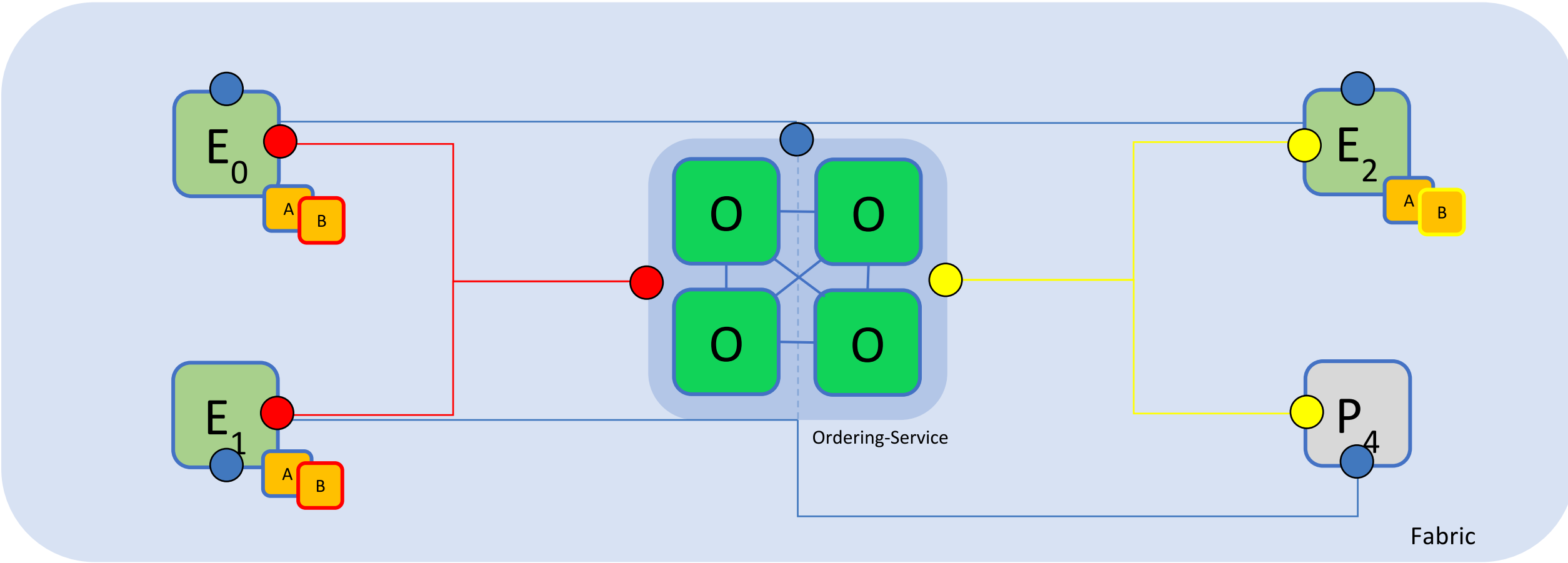
```
$ peer channel create -o [orderer] ...
```

Bootstrapping the Network (5/6) – Join Channels



- Peers that are permissioned can then **join** the channels they want to transact on
  `$ peer channel join ...`

Bootstrapping the Network (6/6) – Instantiate Chaincode



- Peers finally **instantiate** the Chaincode on the channels they want to transact on
  `$ peer channel instantiate ... -P 'policy'`
- Once instantiated a Chaincode is live and can process transaction requests
- Endorsement Policy is specified at instantiation time

Endorsement Policy Example

```
peer chaincode instantiate -C testchainid -n mycc \
       -p github.com/hyperledger/fabric/examples/chaincode/go/chaincode_example02 \
       -c '{"Args":["init","a","100","b","200"]}' \
       -P "AND('Org1.member', 'Org2.member')"
```

This command deploys chaincode mycc on chain testchainid with the policy AND('Org1.member', 'Org2.member').

- **AND('Org1.member', 'Org2.member', 'Org3.member') -** requests 1 signature from each of the three principals
- **OR('Org1.member', 'Org2.member') -** requests 1 signature from either one of the two principals
- **OR('Org1.member', AND('Org2.member', 'Org3.member')) -** requests either one signature from a member of the Org1 MSP or 1 signature from a member of the Org2 MSP and 1 signature from a member of the Org3 MSP.

DEMO

C0rWin / pgdays

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Settings | Insights

No description, website, or topics provided. Edit

Add topics

3 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

C0rWin Adding artifacts: demo crypto material, create channel config ... Latest commit 2eed399 2 days ago

app	Init PGDays'17 chaincode examples	3 days ago
artifacts	Adding artifacts: demo crypto material, create channel config	2 days ago
chaincode	Remove system temp files	2 days ago
.gitignore	Init PGDays'17 chaincode examples	3 days ago
README.md	Init PGDays'17 chaincode examples	3 days ago

README.md

PGDays'17 Hyperledger Fabric Demo

In this repository implemented demo chaincodes with primary goal to demonstrate capabilities of Hyperledger Fabric chaincode development process to be presented at PGDays'17 conference.

Use case

QUESTIONS?

Artem Barger (bartem@il.ibm.com)

Thank you!