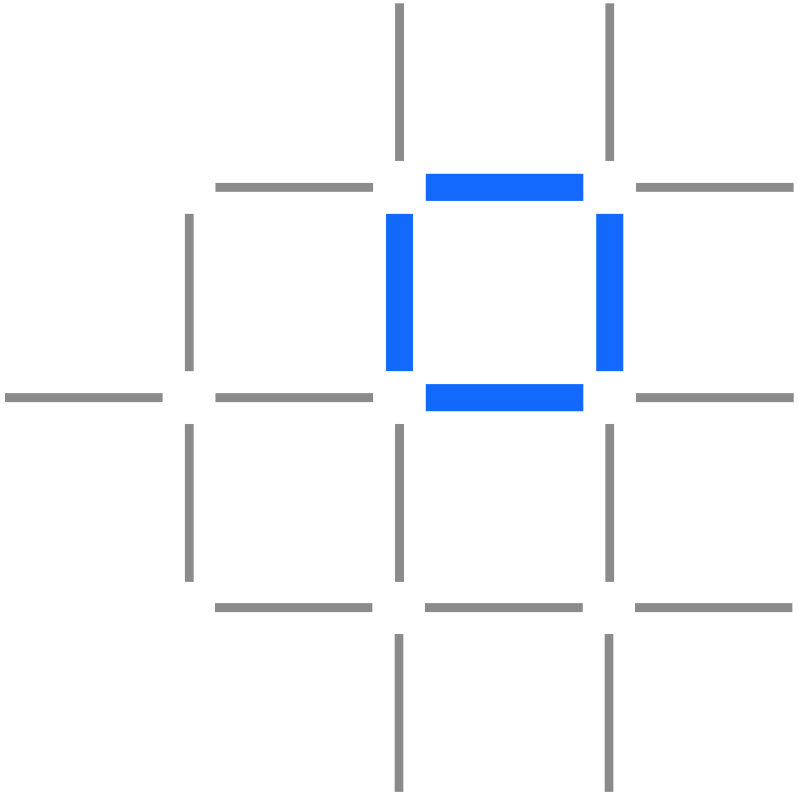
Lab 2

Bootstraping Blockchain Network

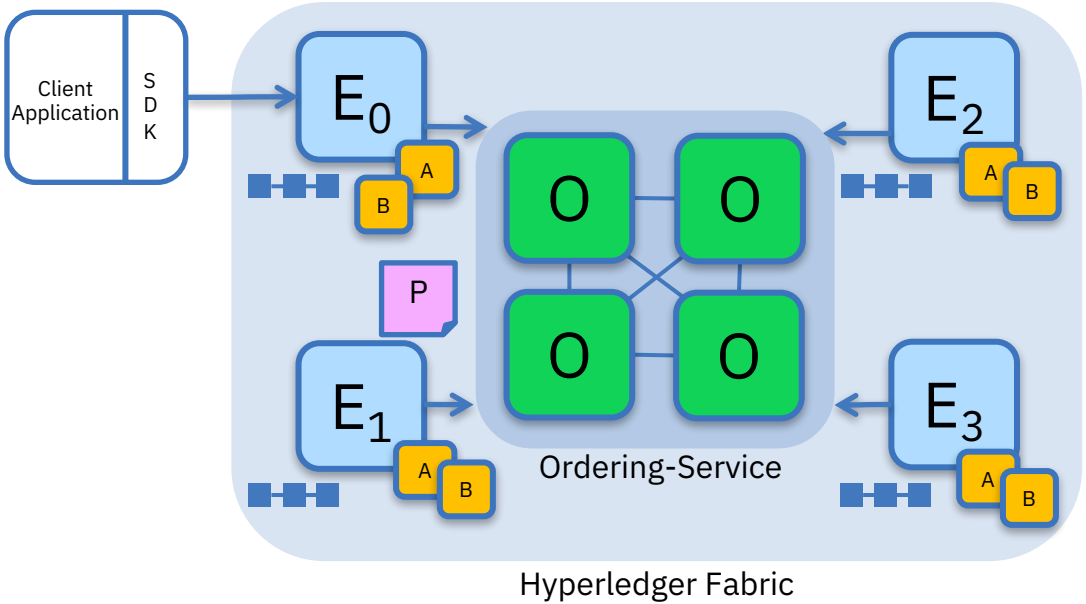
IBM **Blockchain**



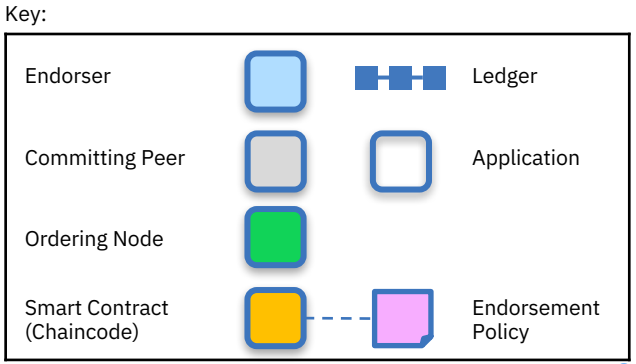
Blockchain education series

V4.0, 23 August 2017

Single Channel Network

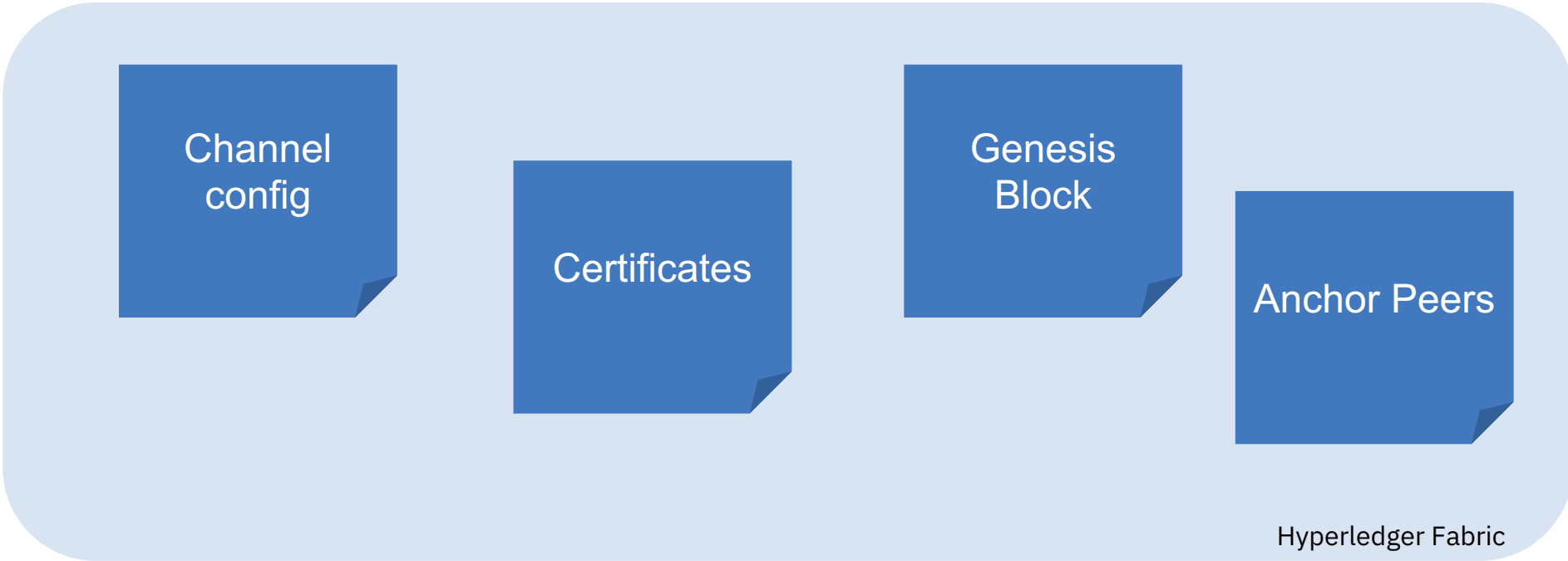


- Similar to v0.6 PBFT model
- All peers connect to the same system channel (blue).
- All peers have the same chaincode and maintain the same ledger
- Endorsement by peers E_0, E_1, E_2 and E_3



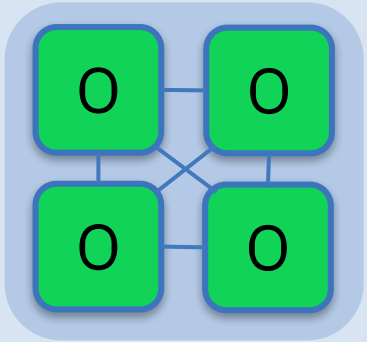
Network Setup

Defining network topology (1/7) – Generating crypto material



- Necessary crypto material is generated and ready for use
- `$ cryptogen…`
- `$ configtxgen…`

Bootstrapping the Network (2/7) – Configure & start Ordering Service

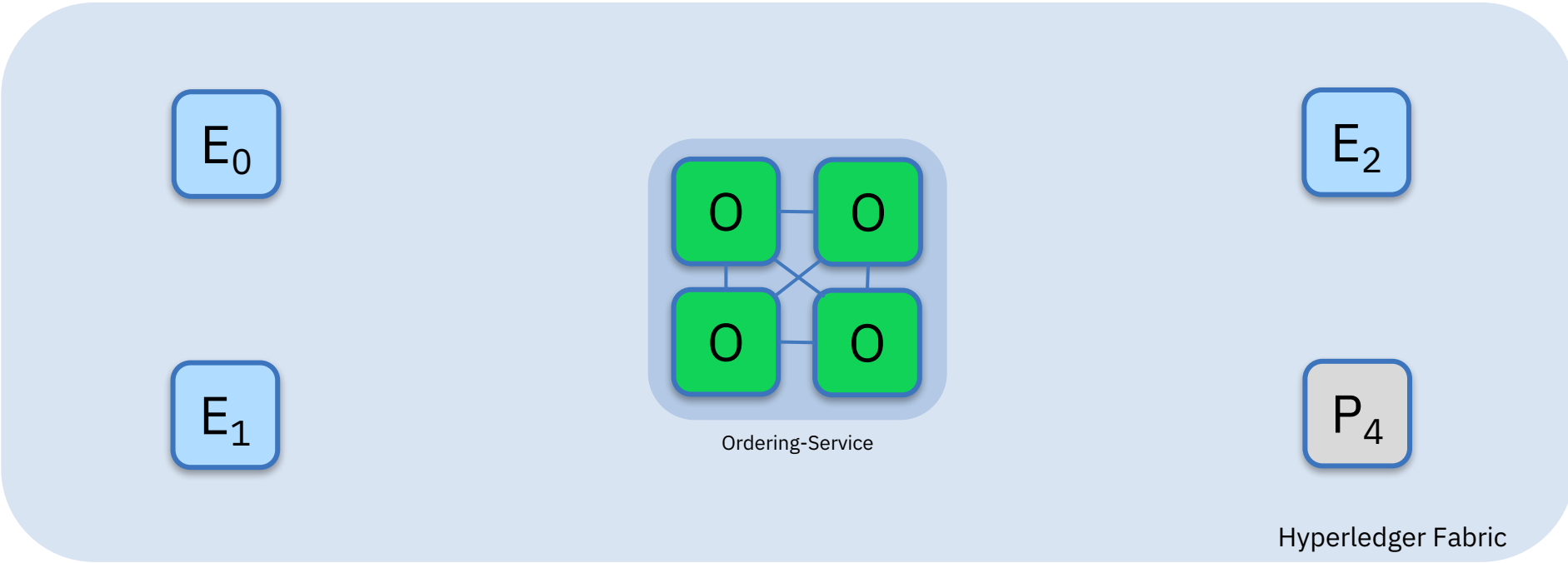


Ordering-Service

Hyperledger Fabric

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

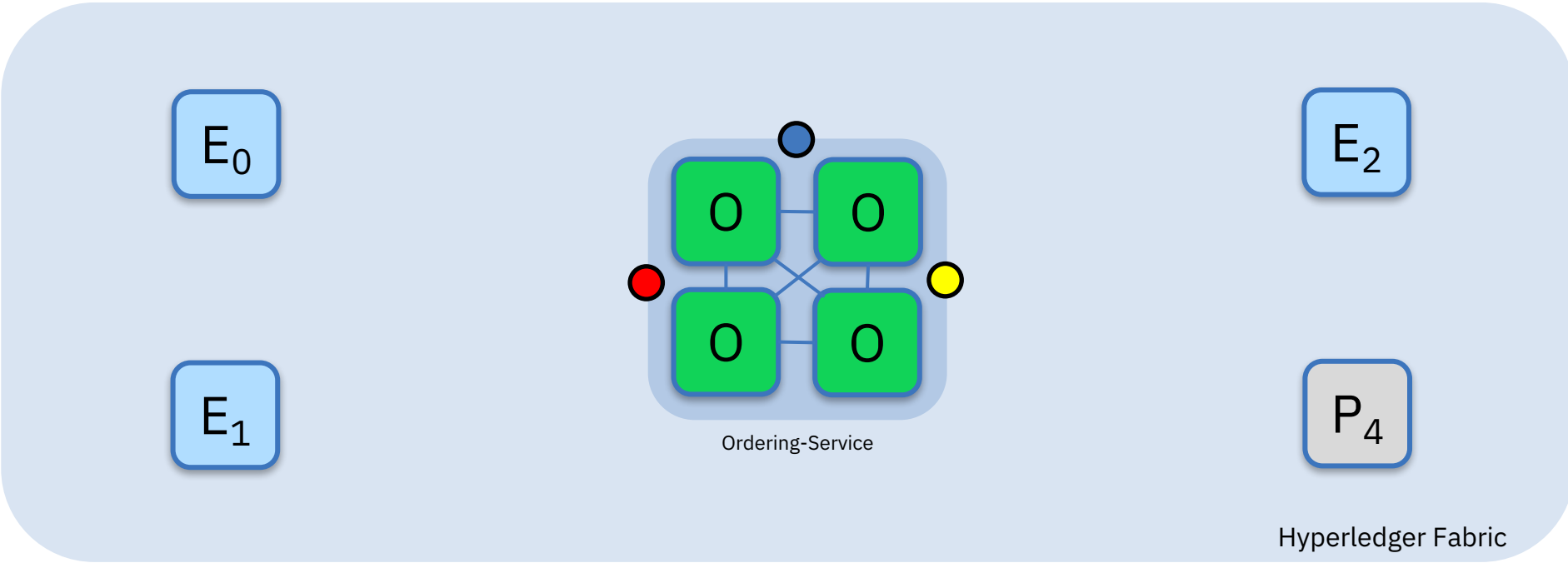
Bootstrapping the Network (3/7) – Configure and Start Peer Nodes



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

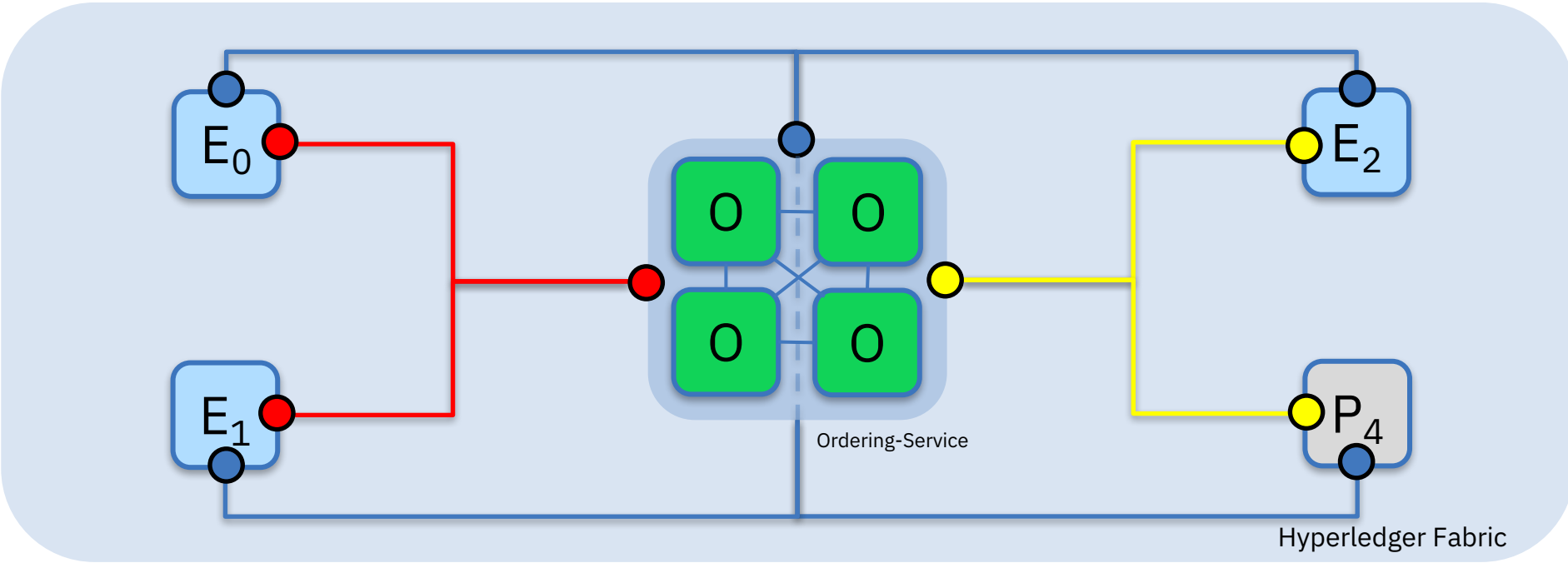
```
$ peer node start ...
```

Bootstrapping the Network (4/7) – Create Channels



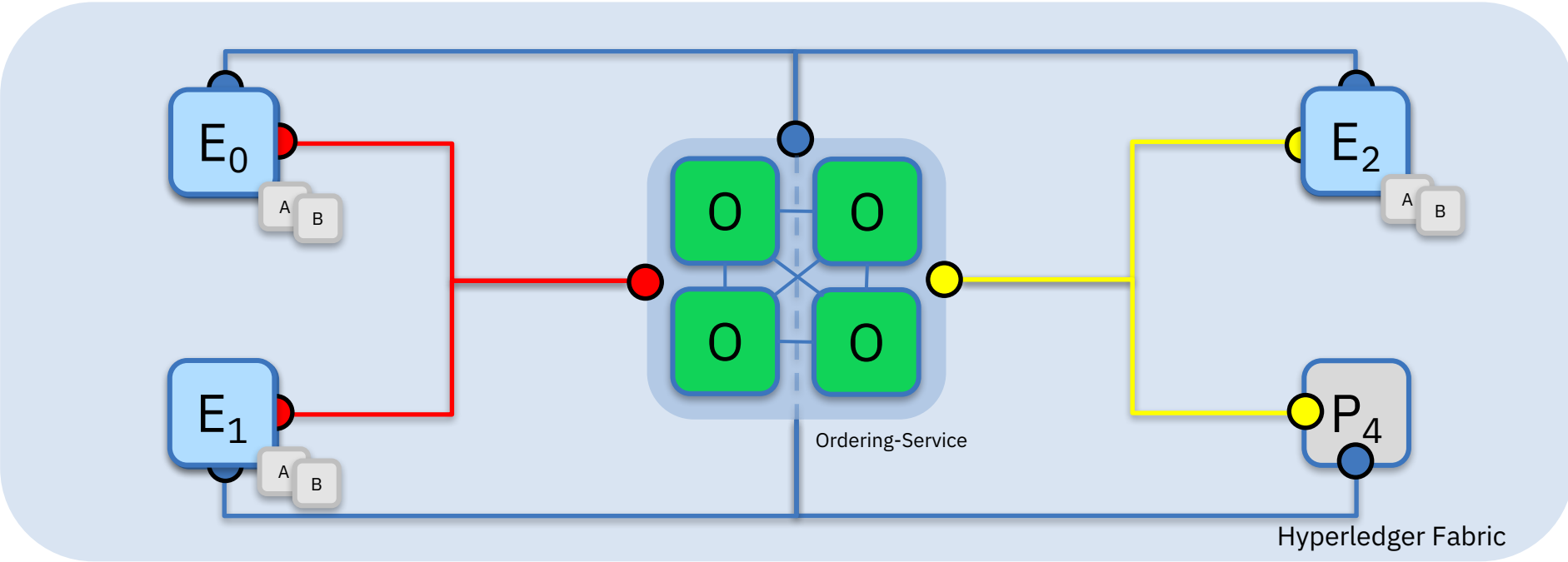
- Channels are created on the ordering service
  `$ peer channel create -o [orderer] ...`

Bootstrapping the Network (5/7) – Join Channels



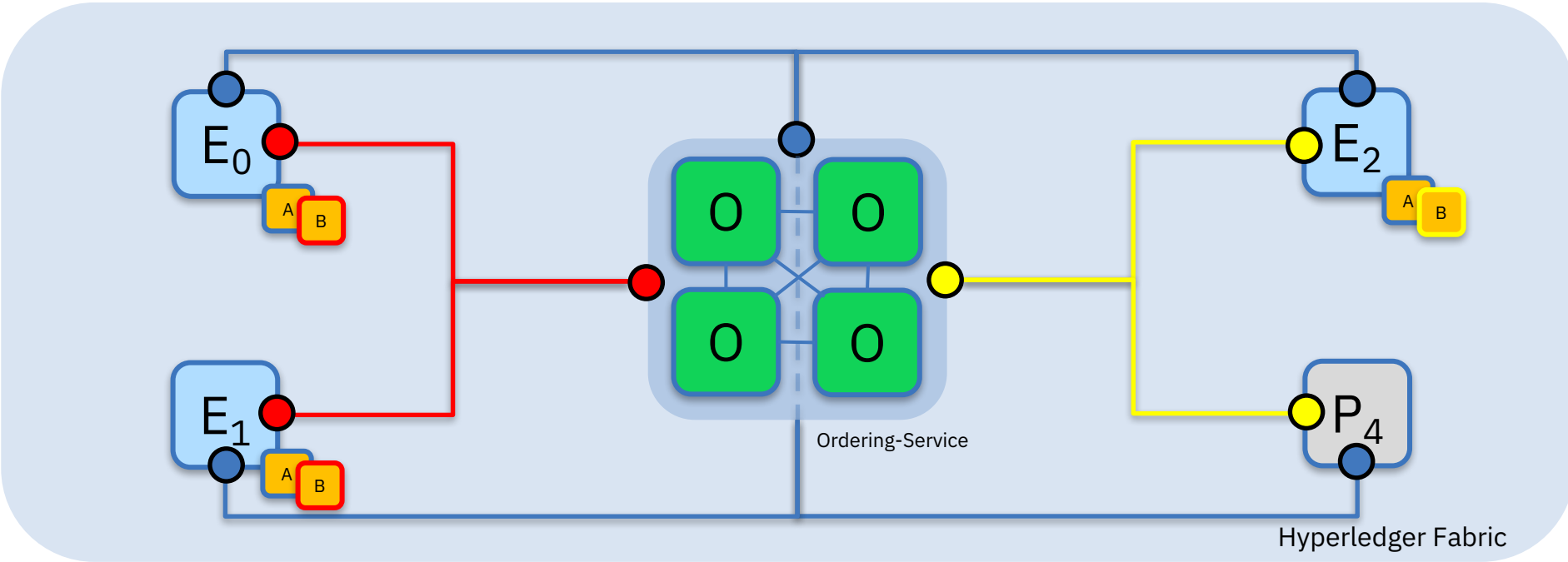
- Peers that are permissioned can then join the channels they want to transact on
  `$ peer channel join ...`

Bootstrapping the Network (6/7) – Install Chaincode



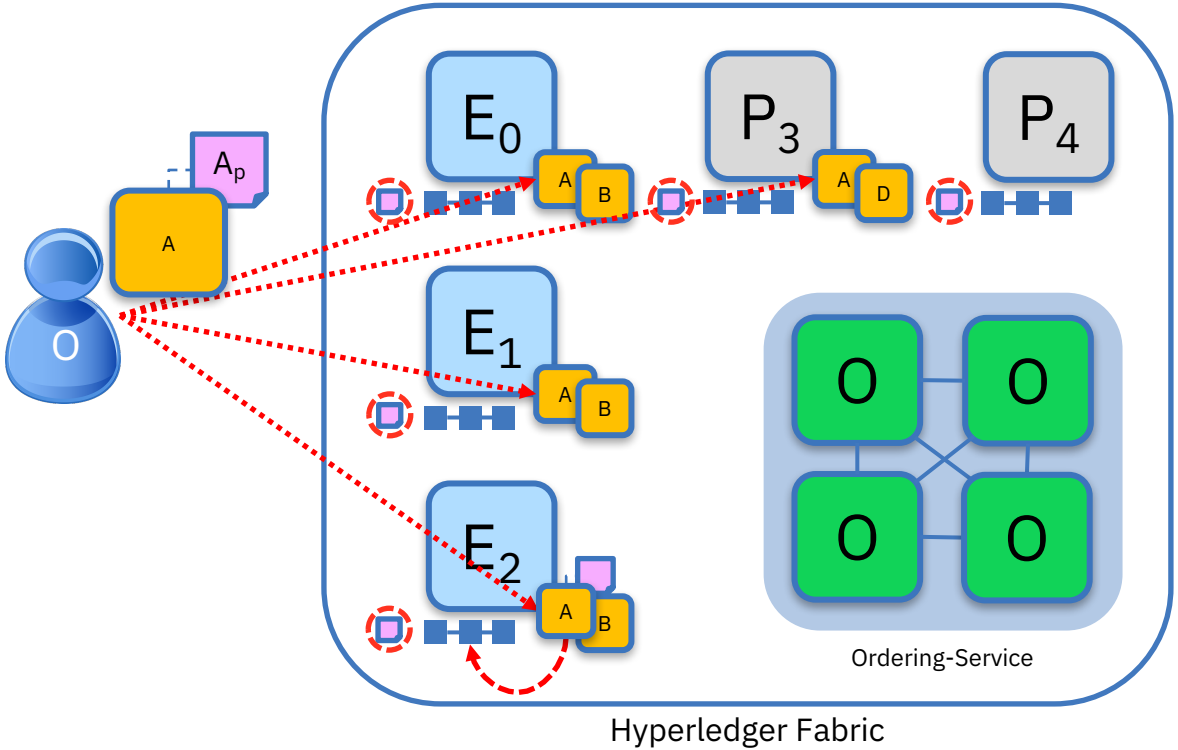
- Chaincode is installed onto each Endorsing Peer that needs to execute it
  `$ peer chaincode install ...`

Bootstrapping the Network (7/7) – Instantiate Chaincode



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

Installing and instantiating smart contract Chaincode

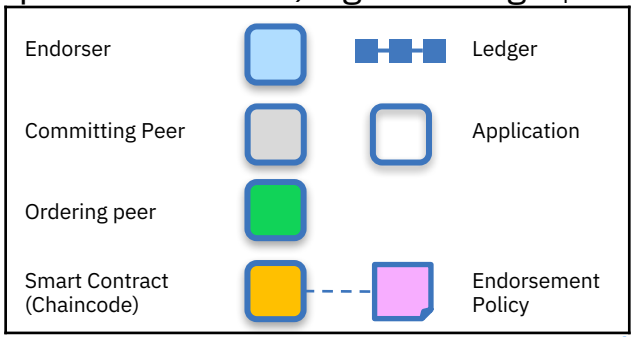


Operator installs then instantiates

Operator installs smart contracts with endorsement policies to appropriate peers: E_0, E_1, E_2, P_3, and not P_4

Operator instantiates smart contract on given channel. One-time initialization

Policy subsequently available to all peers on channel, e.g. including P_4



Endorsement Policies

Endorsement Policy Syntax

```
$ peer chaincode instantiate
-C mychannel
-n mycc
-v 1.0
-p chaincode_example02
-c '{"Args":["init","a", "100",
"b","200"]}'
-P "AND('Org1MSP.member')“
```

This command instantiates the chaincode mycc on channel mychannel with the policy AND('Org1MSP.member')

Policy Syntax: EXPR(E[, E...])

Where EXPR is either AND or OR and E is either a principal or nested EXPR.

Principal Syntax: MSP.ROLE

Supported roles are: member and admin.

Where MSP is the MSP ID required, and ROLE is either “member” or “admin”.

Endorsement Policy Examples

Examples of policies:

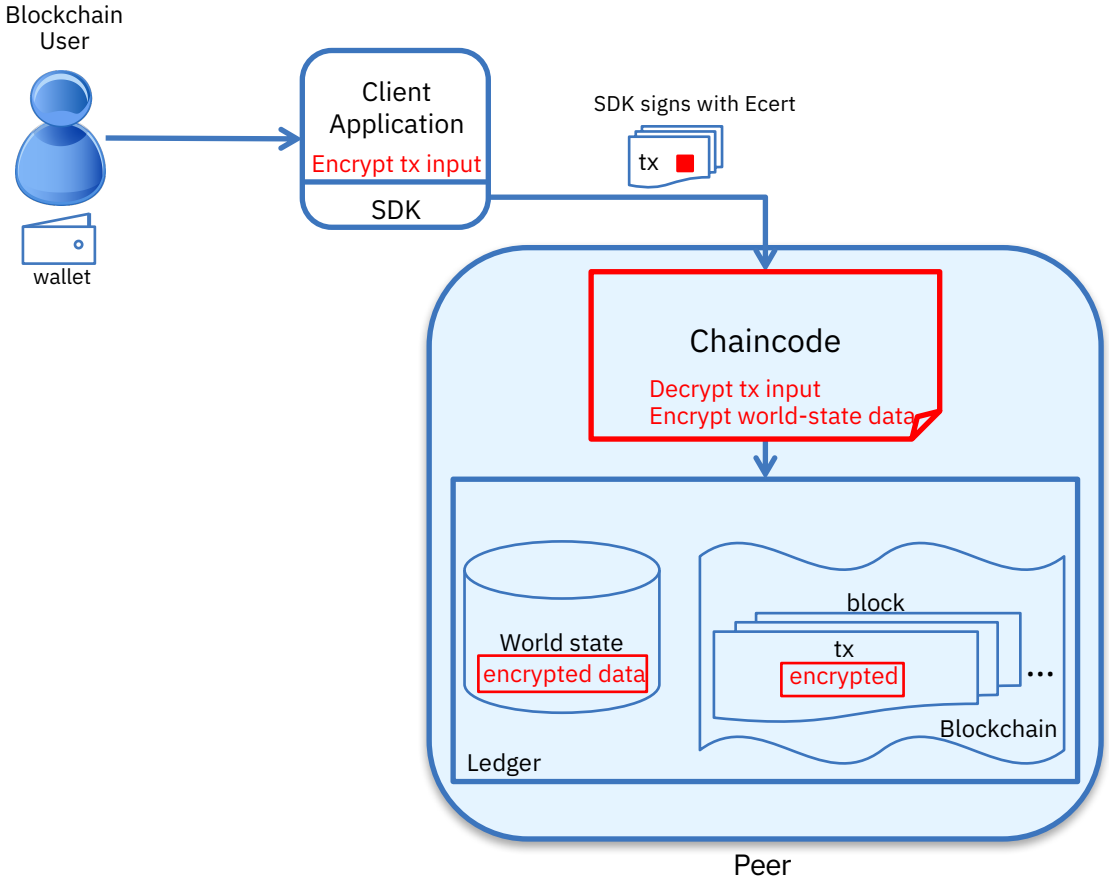
- Request 1 signature from all three principals
  - AND('Org1.member', 'Org2.member', 'Org3.member')
- Request 1 signature from either one of the two principals
  - OR('Org1.member', 'Org2.member')
- Request either one signature from a member of the Org1 MSP or (1 signature from a member of the Org2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

Permissioned Ledger Access

Transaction and Identity Privacy

- Enrollment Certificates, Ecerts
  - Long term identity
  - Can be obtained offline, bring-your-own-identity
- Permissioned Interactions
  - Users sign with their Ecert
- Membership Services
  - Abstract layer to credential providers

Application Level Encryption



Data Encryption

Handled in the application domain.

Multiple options for encrypting:

- Transaction Data
- Chaincode*
- World-State data

Chaincode optionally deployed with cryptographic material, or receive it in the transaction from the client application using the transient data field (not stored on the ledger).

*Encryption of application chaincode requires additional development of system chaincode.

Pluggable World State

WorldState Database

- Pluggable worldstate database
- Default embedded key/value implementation using LevelDB
  - Support for keyed queries, but cannot query on value
- Support for Apache CouchDB
  - Full query support on key and value (JSON documents)
  - Meets a large range of chaincode, auditing, and reporting requirements
  - Will support reporting and analytics via data replication to an analytics engine such as Spark (future)
  - Id/document data model compatible with existing chaincode key/value programming model