CSE446: Blockchain & Cryptocurrencies

Lecture – 16: Hyperledger Fabric - 1

Agenda

- Private Blockchain Systems
- 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 the 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/

HF: Modularity

- Fabric is highly modular and configurable
- A pluggable *ordering service* allows different types of consensus algorithms to be integrated with HF to fit particular use cases and trust models
- Smart contracts ("chaincode") run within a container environment (e.g. Docker) for isolation
  - They can be written in standard programming languages such as Java, Go and JavaScript, rather than constrained domain-specific languages (DSL)
- 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, allows only authorised entities to participate in the blockchain platform
- It also 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
- This “permissioned” notion of Hyperledger Fabric, coupled with the existence and capabilities of channels, helps address scenarios where privacy and confidentiality are of paramount concerns

Fabric privacy

- To further obfuscate the data, values within chaincode can be encrypted (in part or in total)
  - using common cryptographic algorithms such as AES before sending transactions to the ordering service and appending blocks to the ledger
- Once encrypted data has been written to the ledger, it can be decrypted only by a user in possession of the corresponding key that was used to generate the cipher text

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
- A pluggable *membership service provider* is responsible for associating entities in the network with cryptographic identities and authenticating them
- 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

Fabric components

- Users and identities (Membership services)
- Nodes
- Chaincode
- Transactions, Blocks and Ledger
- Consensus

Users and identities

- Hyperledger Fabric underpins a network where all participants have known identities
- Remember: users on a blockchain network are represented using public keys
- For any private blockchain, we need verifiable identities
- It means, there must be a way to guarantee that a certain public key represents an authorised entity in a network
- Based on this assurance we can create rules which will dictate who can participate in the blockchain network
- Public Key Infrastructure (PKI) is used in Fabric to generate cryptographic certificates which are tied to organisations, network components, and end users or client applications

Fabric network: CA and PKI

- But, how can we ensure this association?
- We use a Trusted Third Party (TTP) which vouches that a certain public key belongs to a certain entity
- In security, such TTPs are known as Certificate Authorities (CAs)

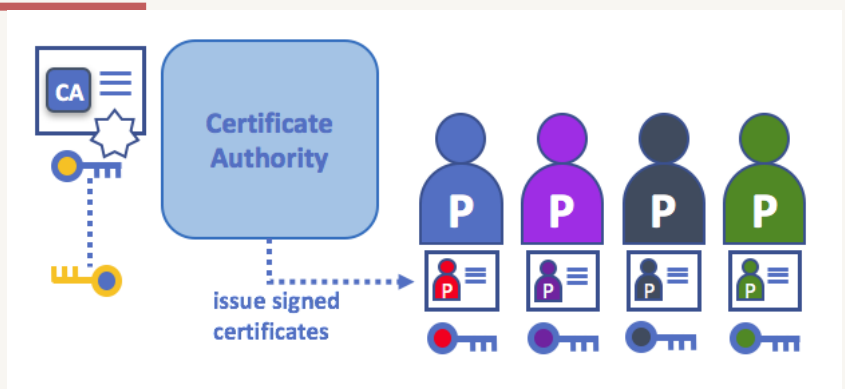
https://medium.com/codechain/modified-merkle-patricia-trie-how-ethereum-saves-a-state-e6d7555078dd

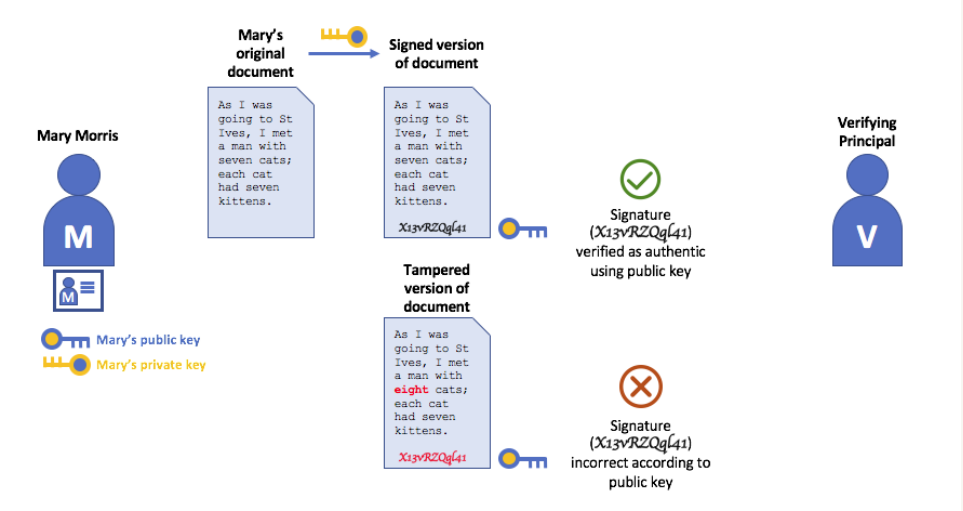
Fabric network: CA and PKI

- A CA issues a digital certificate which is used to bind a public key with an entity
- This association looks something like this:
  - "I, Mango CA, certify that Mr. Rahim belongs to Organisation O4 has this public key P_k and this is valid till 19 December, 2026"

Fabric network: CA and PKI

- Every digital certificate is then signed by the private key of the CA
- Anybody can utilise the public key of the CA to verify the signature in the certificate
- Once verified, everybody trusts the association between the public key and the entity
- If the certificate is tampered, the signature verification will fail





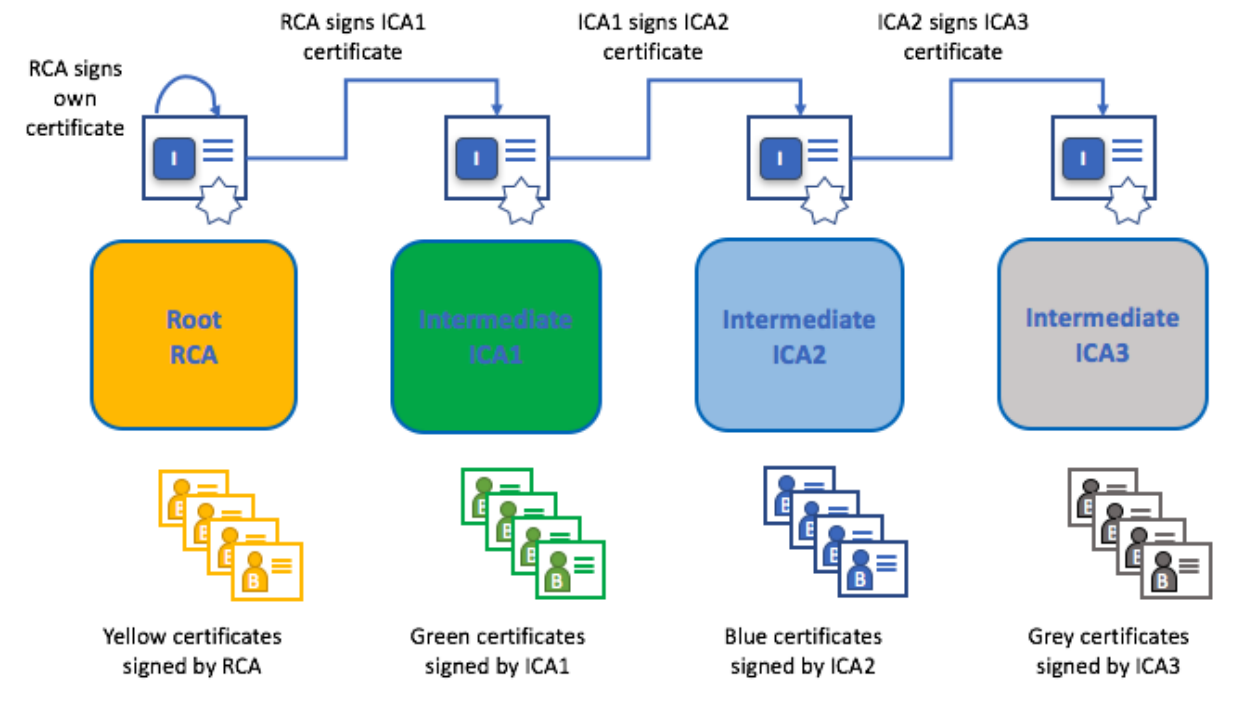
Fabric network: CA and PKI

- But, the problem is how do we get the public key of the CA to verify the signature in the certificate?
  - We used the signed certificate to get the public key of an entity
  - Does it mean we will need another certificate to get a CA's public key?
  - Who will sign such certificates? How the signature verification will work?
  - It is a circular problem!

Fabric network: CA and PKI

- Solution:
  - Create a hierarchy: Root CAs and Intermediate CAs
- Intermediate CAs distribute certificates to millions of users
- Root CAs distribute certificates for the intermediate CAs
- Embed the public keys of Root CAs to the OS and devices
  - In order to verify the root CA certificates
- Such a hierarchy is known as PKI (Public Key Infrastructure)

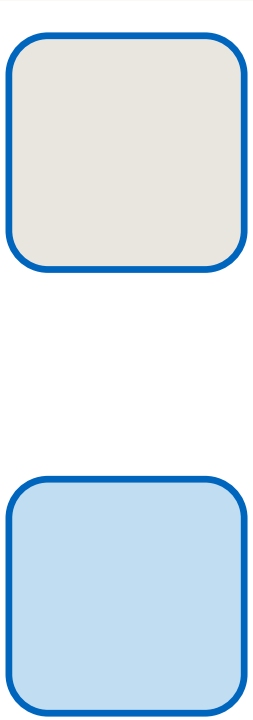
Fabric network: CA and PKI



Fabric CA is a private root CA provider capable of managing digital identities of Fabric participants that have the form of **X.509** certificates for the Fabric network

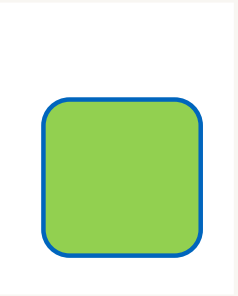
Nodes and roles

- Committing Peer
  - Maintains the state of the Blockchain and commits transactions
  - It verifies endorsements (rules and policies) and validates 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)



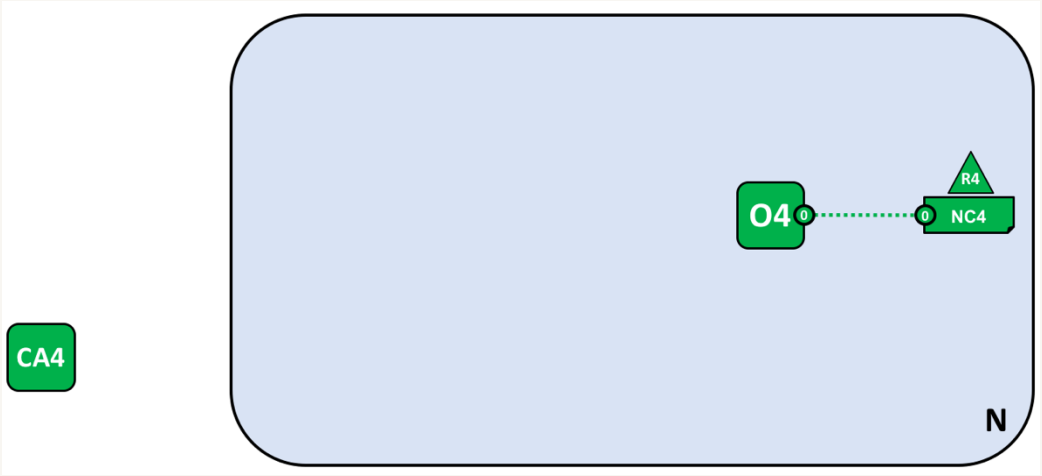
Nodes and roles

- 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



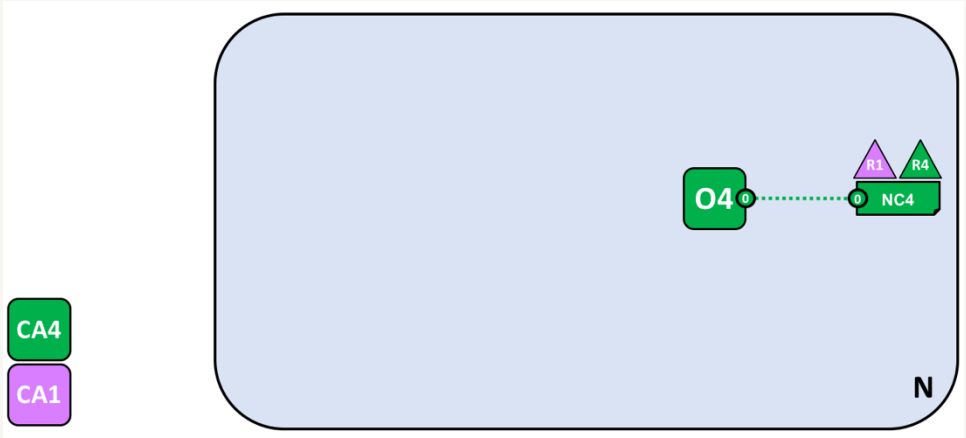
Fabric network: an illustrative example

- We start a network with the initialisation of an orderer
- Our example network, **N**, consists of
  - a single orderer **O4**
  - a single organisation **R4**
  - a certificate authority **CA4**
- **O4** is configured according to a network configuration **NC4**, which gives administrative rights to organisation **R4**
- **CA4** is used to dispense identities to the administrators and network nodes of the **R4** organisation



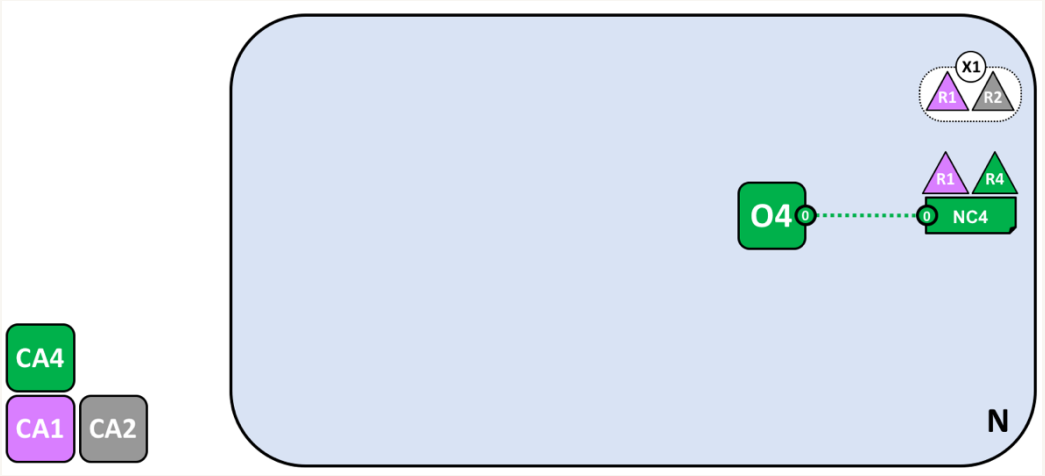
Fabric network

- In this next phase, we are going to allow organisation **R1** users to administer the network
- R4 updates the network configuration to make R1 an administrator too
- After this point R1 and R4 have equal rights over the network configuration
- Also notice that certificate authority CA1 has been added
  - it can be used to identify users from R1



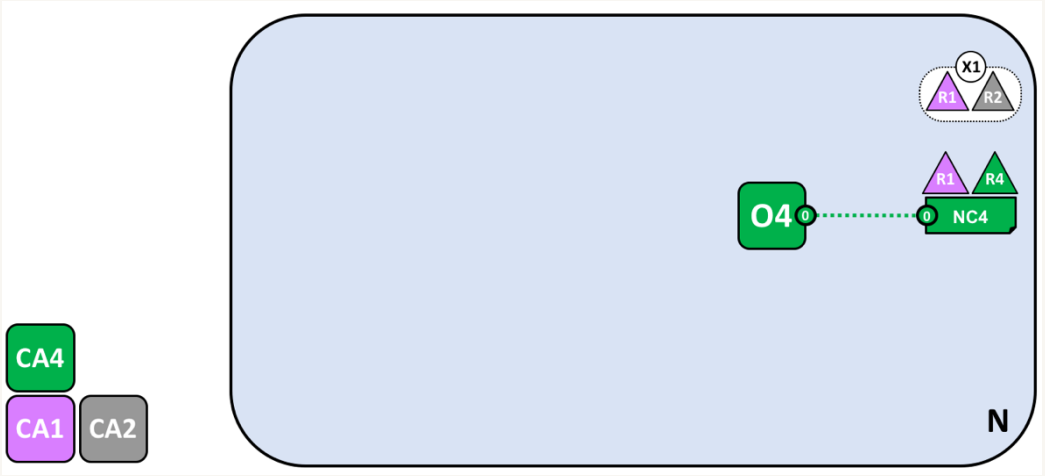
Fabric network

- Next, define a consortium
- This word literally means “a group with a shared destiny”, e.g. trying to utilise the blockchain for a particular use case
- To enable this, we need to combine a set of organisations in our blockchain network



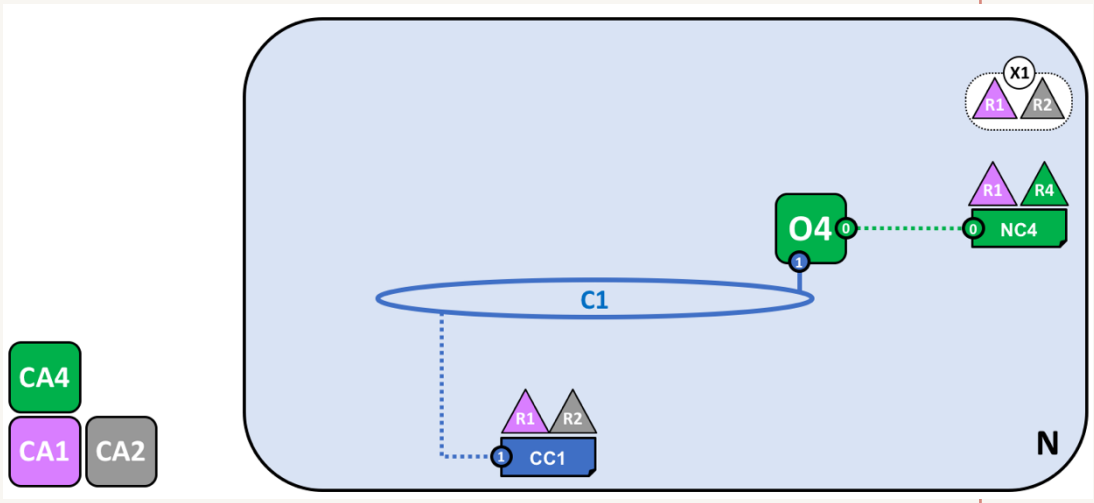
Fabric network

- A network administrator defines a consortium X1 that contains two members
  - the organisations R1 and R2
- This consortium definition is stored in the network configuration NC4
- Because of the way NC4 is configured, only R1 or R4 can create a new consortium
- CA1 and CA2 are the respective CAs for these organisations



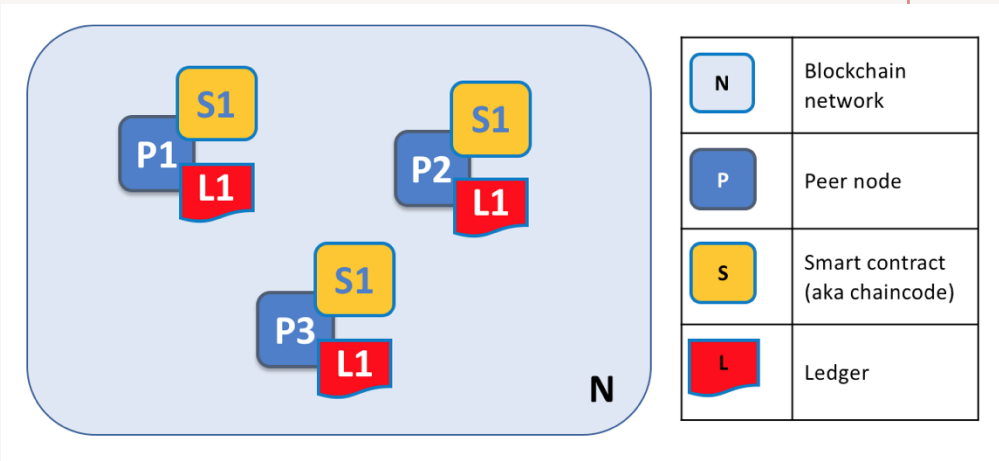
Fabric network

- A channel is a primary communications mechanism by which the members of a consortium can communicate with each other
- A channel **C1** has been created for R1 and R2 using the consortium definition **X1**
- The channel is governed by a channel configuration **CC1**, completely separate to the network configuration
- **CC1** is managed by R1 and R2 who have equal rights over C1
- R4 has no rights in CC1 whatsoever
  - As it is not a part of the consortium



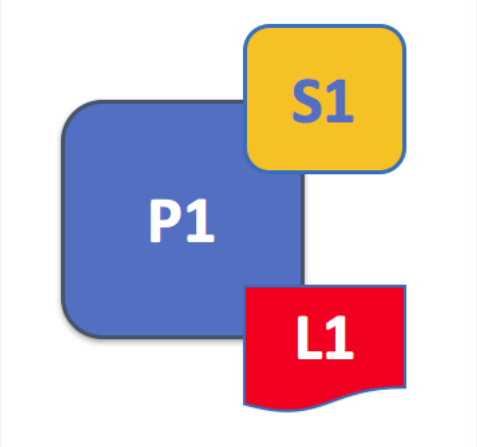
Fabric network: peers

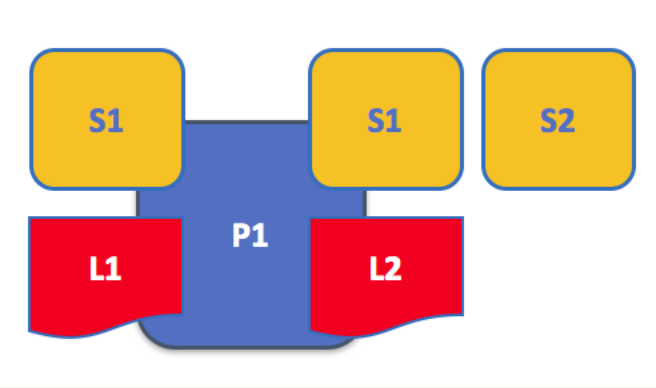
- Next, we will add Peers in the network
- A blockchain network is comprised primarily of a set of peer nodes (or, simply, peers)
- Peers are a fundamental element of the network because they host ledgers and smart contracts
- In this example, the network N consists of peers P1, P2 and P3, each of which maintain their own instance of the distributed ledger L1
- P1, P2 and P3 use the same chaincode, S1, to access their copy of that distributed ledger



Fabric network: peers

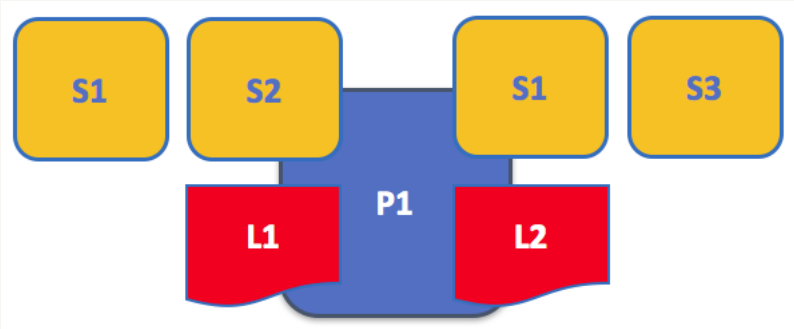
- In this example, P1 hosts an instance of ledger L1 and an instance of chaincode S1
- There can be many ledgers and chaincodes hosted on an individual peer
- In this example, we can see that the peer P1 hosts ledgers L1 and L2
- Ledger L1 is accessed using chaincode S1
- Ledger L2 on the other hand can be accessed using chaincodes S1 and S2





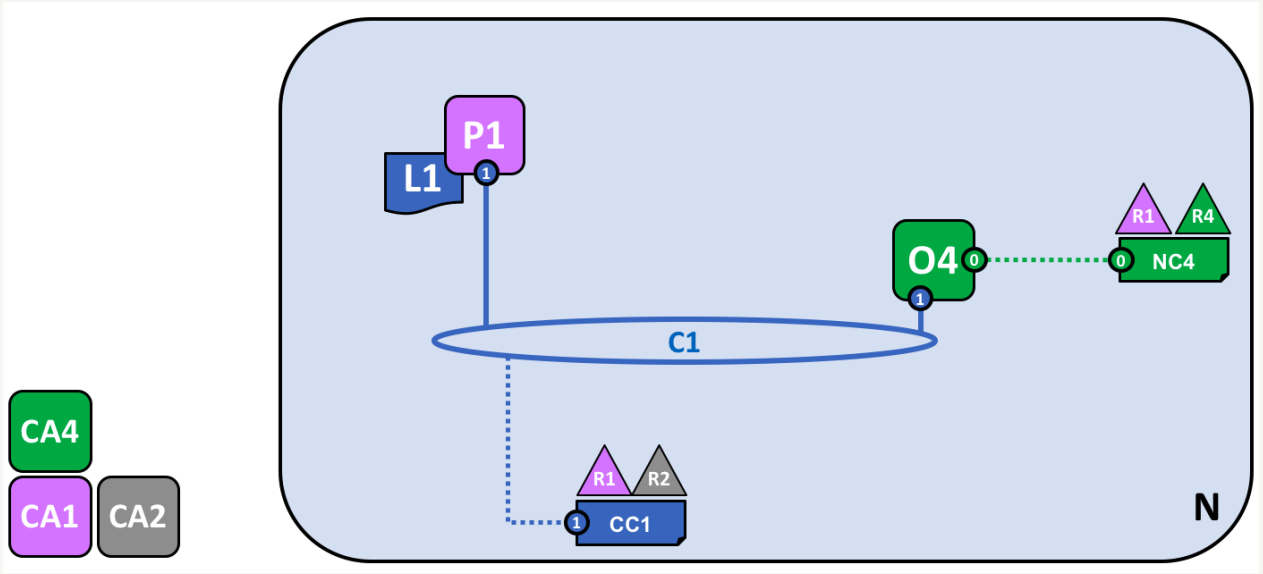
Fabric network: peers

- Each ledger can have many chaincodes which access it
- In this example, we can see that peer P1 hosts ledgers L1 and L2, where L1 is accessed by chaincodes S1 and S2, and L2 is accessed by S1 and S3
- We can see that S1 can access both L1 and L2



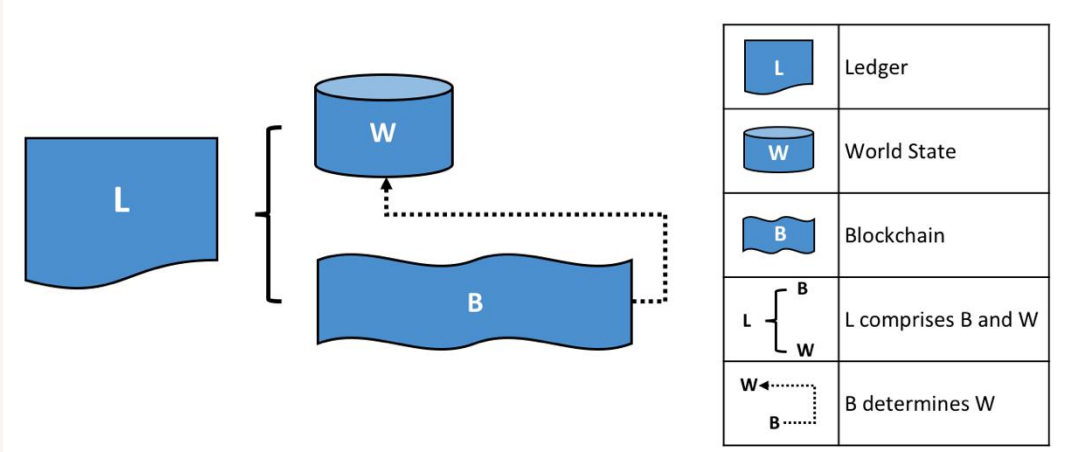
Fabric network

- Next, a peer node P1 has joined the channel C1
- P1 hosts a copy of the ledger L1
- P1 and O4 can communicate with each other using channel C1



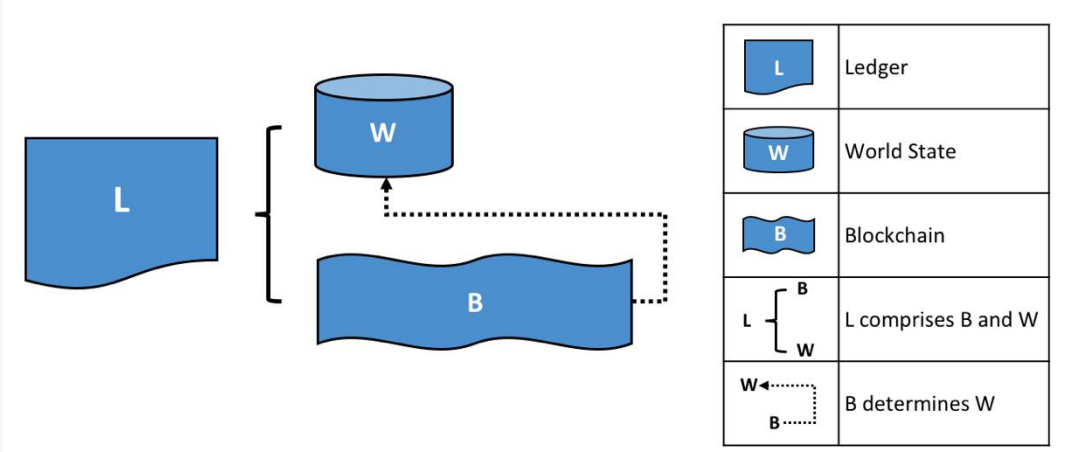
Fabric network: ledger

- 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 network: 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



Question?