



区块链技术与应用

(2024年春季)

计算机科学与技术学院 李京

10章 超级账本

Hyperledger Fabric

公有链的不足

- Consensus is hard-coded within the platform, which contradicts the well-established understanding that there is no “one-size-fits-all” (BFT) consensus protocol;
- The trust model of transaction validation is determined by the consensus protocol and cannot be adapted to the requirements of the smart contract;
- Smart contracts must be written in a fixed, non-standard, or domain-specific language, which hinders wide-spread adoption and may lead to programming errors;
- The sequential execution of all transactions by all peers limits performance, and complex measures are needed to prevent denial-of-service attacks against the platform originating from untrusted contracts (such as accounting for runtime with “gas” in Ethereum);
- Transactions must be deterministic, which can be difficult to ensure programmatically;
- Every smart contract runs on all peers, which is at odds with confidentiality, and prohibits the dissemination of contract code and state to a subset of peers.

-Hyperledger Fabric: A Distributed Operating-System for Permissioned Blockchains, EuroSys'18: Thirteenth EuroSys Conference 2018, April 23–26, 2018, Porto, Portugal

公有链最大的优点就是去中心化和安全性，最大的缺点是非许可和一套很严格的共识机制直接导致了公有链处理数据的速度问题和隐私保护问题。

联盟链

- 联盟链是由多个机构共同参与管理的区块链，每个组织或机构管理一个或多个节点，其数据只允许系统内不同的机构进行读写和发送。联盟链的各个节点通常有与之对应的实体机构组织，通过授权后才能加入与退出网络，是许可链。各机构组织组成利益相关的联盟，共同维护区块链的健康运转。
- 联盟链内部会指定多个预选的节点为记账人，每个块的生成由所有的预选节点共同决定，其他接入节点可以参与交易，但不过问记账过程，其他第三方可以通过该区块链开放的API进行限定查询。

联盟链的特点

- 部分去中心化。与公有链不同，联盟链在某种程度上只属于联盟内部的成员所有，由于其节点不多，达成共识容易，交易速度自然快很多。
- 可控性较强。公有链的节点一般是海量的（如比特币、以太坊），要篡改区块数据，几乎是不可能的，而联盟链，只要所有机构中的大部分达成共识，即可更改区块数据。
- 数据不默认公开，可以提供更好的隐私保护，可以分级授权访问。
- 节点是（基本）可信任的。公有链：匿名的，以出块为激励的区块链。联盟链：实名的，以其上运行的商业业务价值为激励的区块链。

联盟链平台

- 超级账本(Hyperledger)
- 企业以太坊联盟(EEA)
- R3区块链联盟
- 区块链服务网络BSN
- 中国分布式总账基础协议联盟(ChinaLedger)
- 中国区块链研究联盟(CBRA)
- 金链盟



目 录

- 10.1 超级账本项目简介
- 10.2 Fabric系统结构
- 10.3 Fabric核心概念

10.1 超级账本项目简介

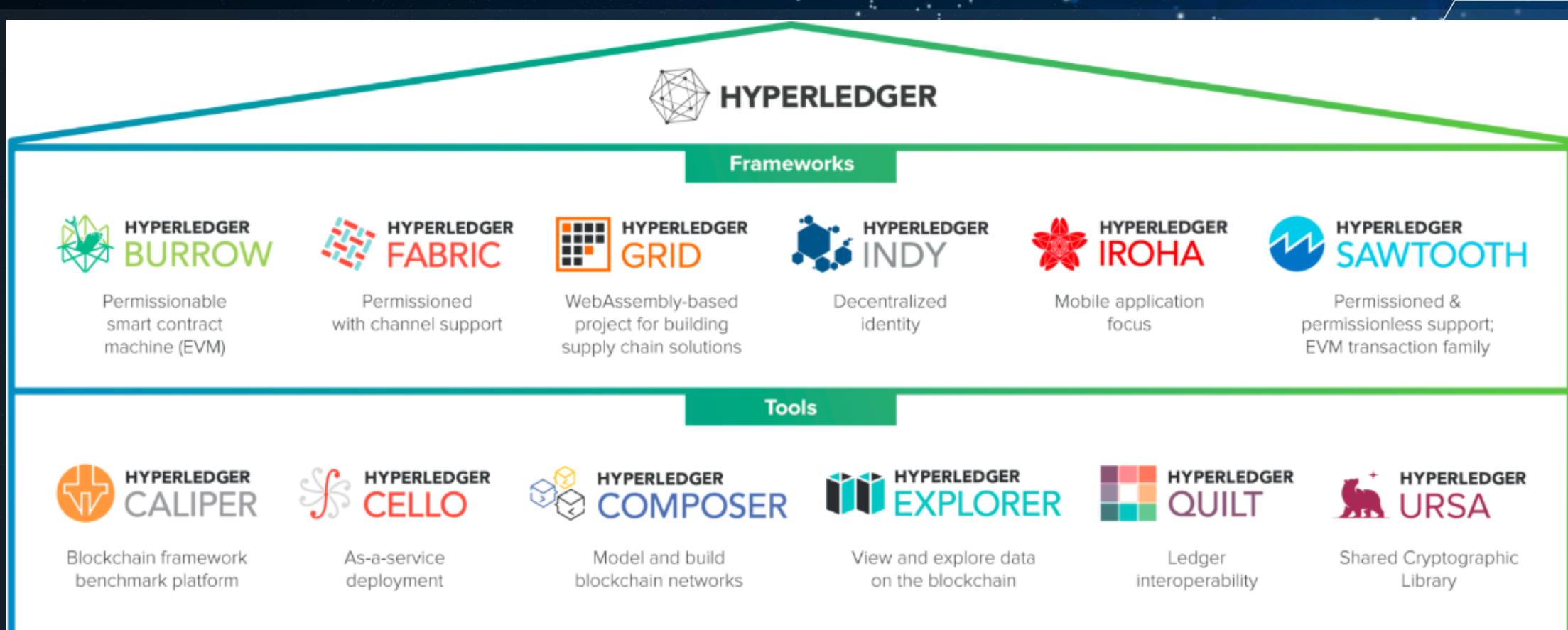
Hyperledger Project

2015年12月，开源世界的旗舰组织——Linux基金会牵头，联合30家初始企业成员（包括IBM、Accenture、Intel、J.P. Morgan、R3、DAH、DTCC、FUJITSU、HITACHI、SWIFT、Cisco等），共同宣布了超级账本（Hyperledger）联合项目的成立。超级账本项目致力为透明、公开、去中心化的企业级分布式账本技术提供开源参考实现，并推动区块链和分布式账本相关协议、规范和标准的发展。项目官方网站为hyperledger.org。

作为一个联合项目（Collaborative Project），超级账本由面向不同目的和场景的子项目构成。目前包括Fabric、SawToothLake、Iroha、Blockchain Explorer、Cello、Indy、Composer、Burrow、Quilt、Caliper等十多个顶级项目，所有项目都遵守Apache v2许可，并约定共同遵守如下的基本原则：

- 重视模块化设计：包括交易、合同、一致性、身份、存储等技术场景；
- 重视代码可读性：保障新功能和模块都可以很容易添加和扩展；
- 可持续的演化路线：随着需求的深入和更多的应用场景，不断增加和演化新的项目。

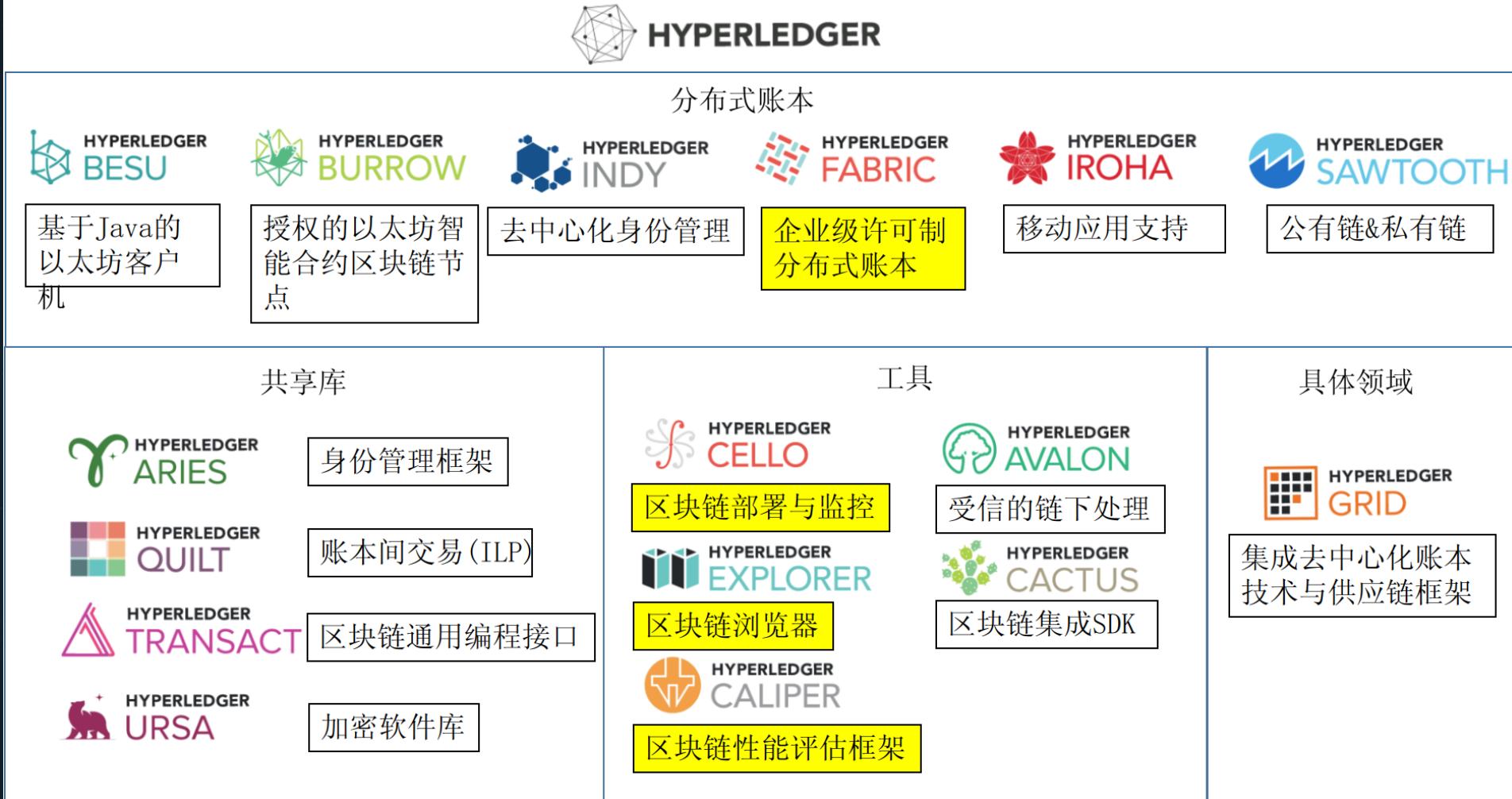
Hyperledger开源项目



- Hyperledger 所有项目代码托管在Gerrit和Github上。主要包括若干顶级项目：Besu、Burrow、Fabric、Indy、Iroha、Sawtooth、Aries、Quilt、Transact、Ursa、Avalon、Caliper、Cello、Explorer、Grid。这些顶级项目相互协作，构成了Hyperledger的生态系统。

Hyperledger开源项目

Hyperledger（超级账本）是一组开源工具，旨在构建一个强大的、业务驱动的区块链框架。



Hyperledger Fabric-Blockchain Technology for Business

- Hyperledger Fabric从一开始就是为企业使用而设计的。
- 对于企业使用，需要考虑以下需求：
 - 参与者必须被识别/可识别
 - 网络需要获得许可
 - 高交易吞吐量性能
 - 交易确认延迟低
 - 与商业交易有关的交易和数据的隐私和机密性

字段	含义	默认值
BatchTimeout	出块时间	默认2秒
MaxMessageCount	区块容纳的交易条数	默认10条
AbsoluteMaxBytes	区块大小	默认10MB
PreferredMaxBytes	每条交易的大小	默认1512KB © 51CTO 博客

Hyperledger Fabric

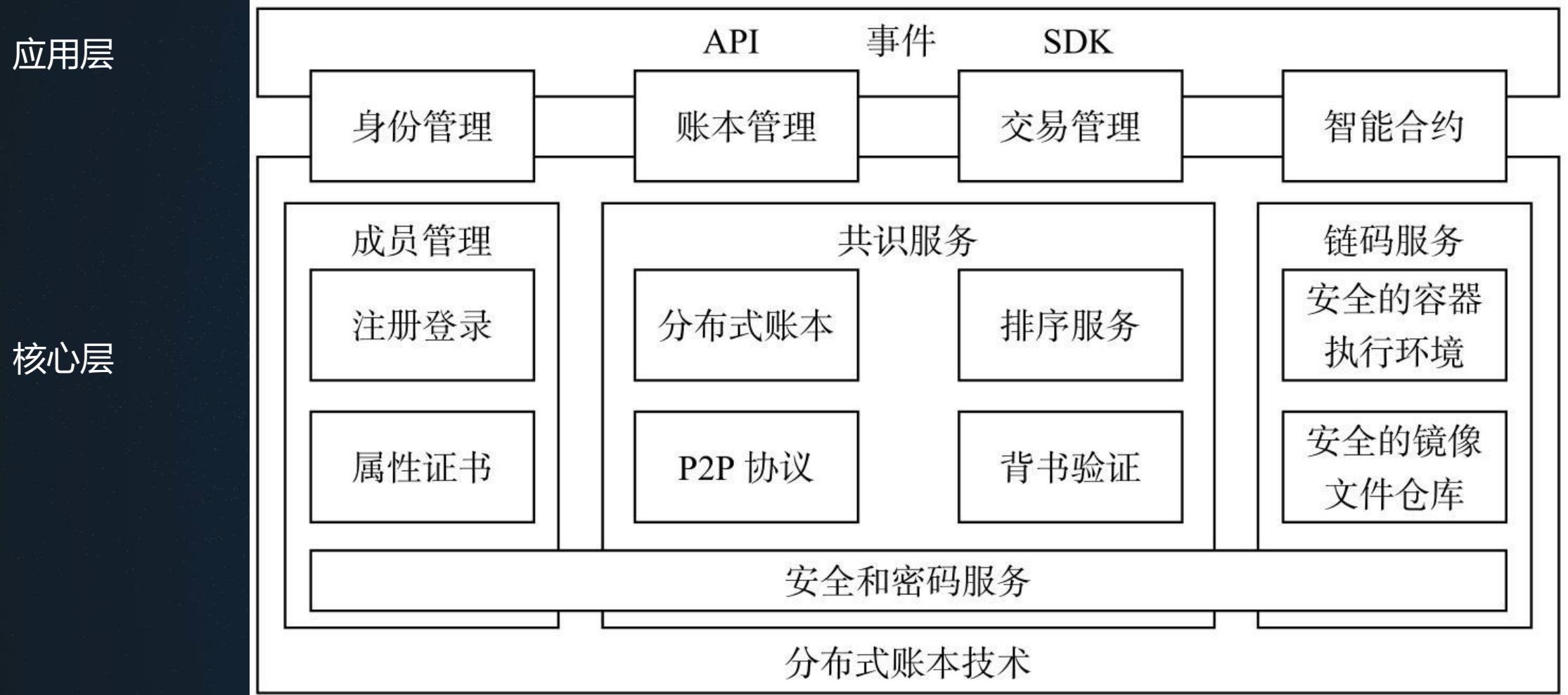
- Hyperledger Fabric是Hyperledger中的区块链项目之一。与其他区块链技术一样，它有一个账本，使用智能合约，是一个参与者管理交易的系统。
- Hyperledger Fabric 是一个许可区块链平台。与允许未知身份参与网络的开放式非许可系统（需要诸如“工作量证明”之类的协议来验证交易并保护网络）不同，Hyperledger Fabric网络的成员需要从可信赖的 成员服务提供者（MSP）注册。
- 许可区块链在一组已知、已识别且经常经过审查的参与者之间运作，这些参与者在产生一定程度信任的治理模式下运作。许可区块链提供了一种方法来保护具有共同目标但可能不完全信任彼此的一组实体之间的交互。通过依赖参与者的身份，许可区块链可以使用更传统的崩溃容错（CFT）或拜占庭容错（BFT）共识协议，这些协议不需要昂贵的挖矿。

Hyperledger Fabric

- Fabric是一个区块链应用支持框架，是一种许可链，组织的成员可以通过一个MSP来注册。
- Fabric经过特别设计，具有模块化的体系结构。作为开发模块化体系结构的企业区块链应用程序的基础，支持可插拔共识、可插拔身份管理协议（如LDAP或OpenID Connect）、密钥管理协议或加密库。
- Fabric提供创建通道（Channel）的功能，允许一组参与者创建一个单独的共同维护的交易账本。
对于某些应用而言，这是一个特别重要的选择。这些应用中，一些参与者可能是竞争对手，并且不希望他们做出的每笔交易都被每个参与者知晓，例如，他们只向某些参与者提供的特殊价格，而其他人不是。如果两个参与者组成一个通道，那么只有这两个参与者拥有该通道的账本副本，而其他参与者没有。
- Fabric支持用Java、Go和Node.js等通用编程语言编写智能合约（chaincode），而不是受限的领域特定语言DSL。这意味着大多数企业已经具备了开发智能合约所需的技能，不需要额外的培训来学习新的语言或DSL。

10.2 Fabric系统结构

Fabric系统结构



Fabric应用层

应用层提供了API、事件、SDK

- Fabric提供了gRPC接口，gRPC是一个跨语言的RPC框架，是一个比http更有效率的接口。
- 在API的基础上，Fabric官方给出了针对不同语言的SDK，如Node.js、Golang、Python和Java等，开发人员可以利用SDK开发基于区块链的应用。
- Fabric采用异步通信的模式进行开发。通过在链码中定义事件，然后使用应用程序去监听这些事件。当某个事件被触发的时候，就可以执行预先定义的回调函数。

应用层可以使用API、SDK调用智能合约、与底层身份服务交互、查询区块链账本，并构建和提交交易提案给背书节点，获得背书后，交易会被提交给排序服务进行排序，并最终被打包成区块并分发到整个网络（交易管理）。

Fabric核心层

- 超级账本Fabric核心层包括三大组件：
 - 区块链服务（Blockchain）
 - 链码服务（Chaincode）
 - 成员权限管理（Membership）
- 区块链提供一个分布式账本平台，一般地，多个交易被打包进区块中，多个区块构成一条区块链。区块链代表的是账本状态机发生变更的历史过程。
- 链码包含所有的处理逻辑
 - 对外提供接口，外部通过调用链码接口来改变世界状态。
 - 世界状态是一个键值数据库，用于存放链码执行过程中涉及到的状态变量。
- 成员权限管理基于PKI，平台可以对接入的节点和客户端的能力进行限制。

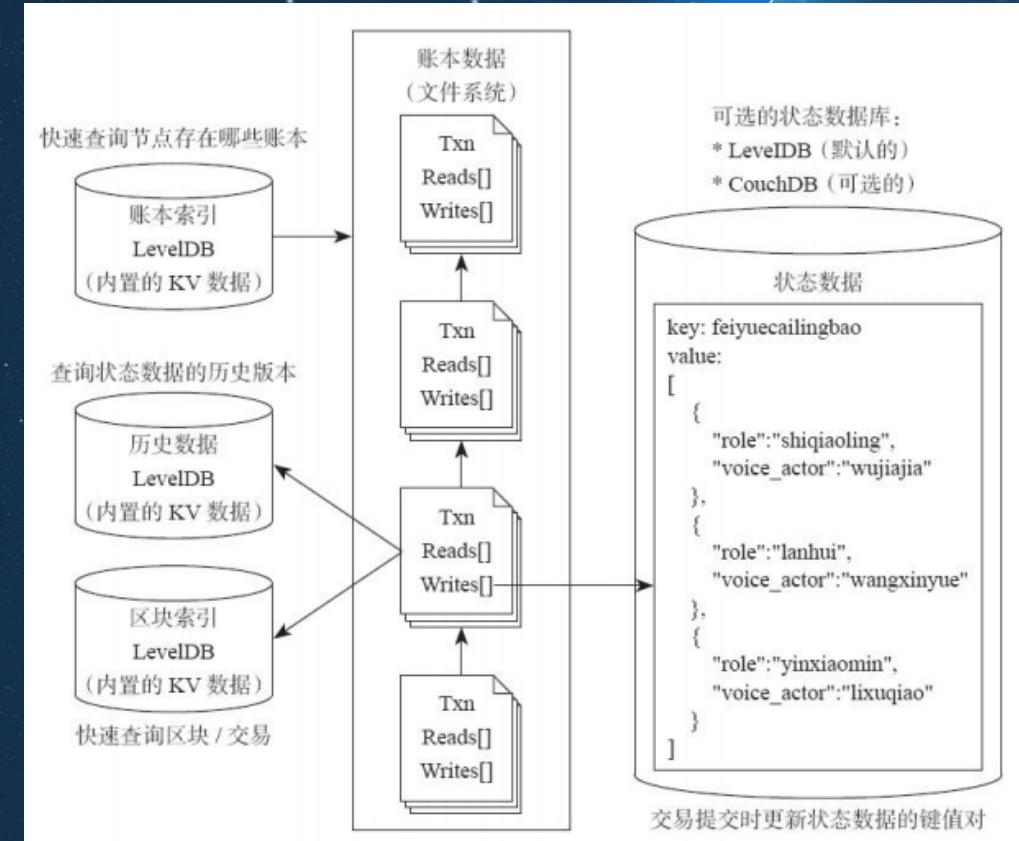


安全和成员服务MSP

- 超级账本交易网络中的所有参与者都知道身份。
- 公钥基础设施用于生成绑定到组织、网络组件和最终用户或客户端应用程序的加密证书。
- 数据访问控制可以在更广泛的网络和通道级别上进行操作和管理。
- 超级账本结构的这种“许可”概念，再加上通道的存在和功能，有助于解决隐私和保密性是首要问题的场景。

分布式账本

- 账本是Fabric中所有状态转换的有序、防篡改的记录。状态转换是由参与方提交的链码调用（称作“交易”）的结果。每个交易都会产生一组资产键值对，这些资产键值对在创建、更新或删除时提交到账本。
- 账本组织成一个区块链和一个状态数据库。区块链用于将不可变的、按顺序排列的记录存储在区块中，状态数据库来维护当前Fabric的状态。每个通道有一个账本。每个节点为其所属的每个通道保留一份账本副本。

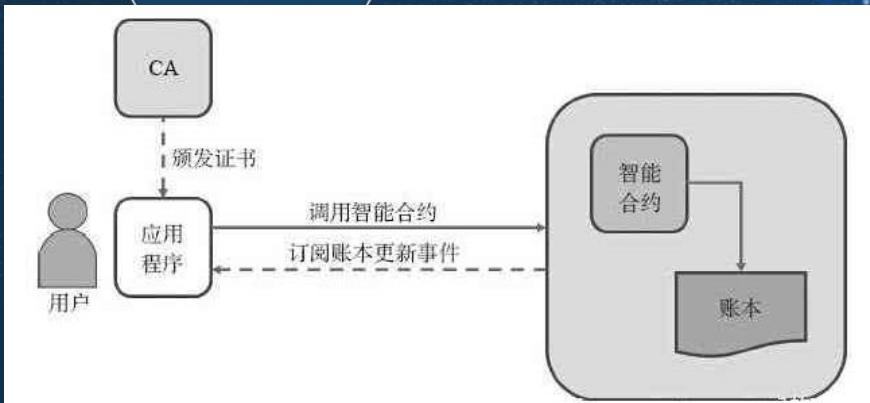
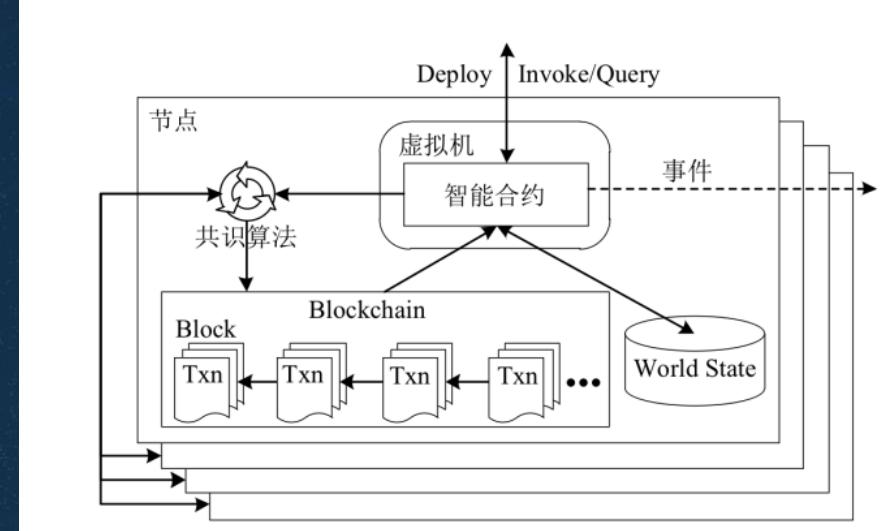


Fabric账本功能

- 使用基于键的查找、范围查询和组合键查询进行账本的查询和更新
- 使用富查询语言进行只读查询（如果使用CouchDB作为状态数据库）
- 只读历史查询-查询某个关键字的账本历史，数据溯源
- 交易包含链码中读取的键/值的版本（**read set**）和链码中写入的键/值的版本（**write set**）
- 交易包含每个背书节点的签名，并提交给排序服务
- 交易被按序装入区块，并从排序服务上被“传递”到通道上的节点
- 节点根据背书策略验证交易，并执行这些策略
- 在扩展块之前，执行版本检查，以确保所读取的资产状态在链码执行期间未发生变化
- 一旦交易被验证和提交，就不可改变
- 通道的账本包含配置区块的定义策略、访问控制列表和其他相关信息
- 通道包含成员服务提供程序实例，允许从不同的证书颁发机构派生加密材料

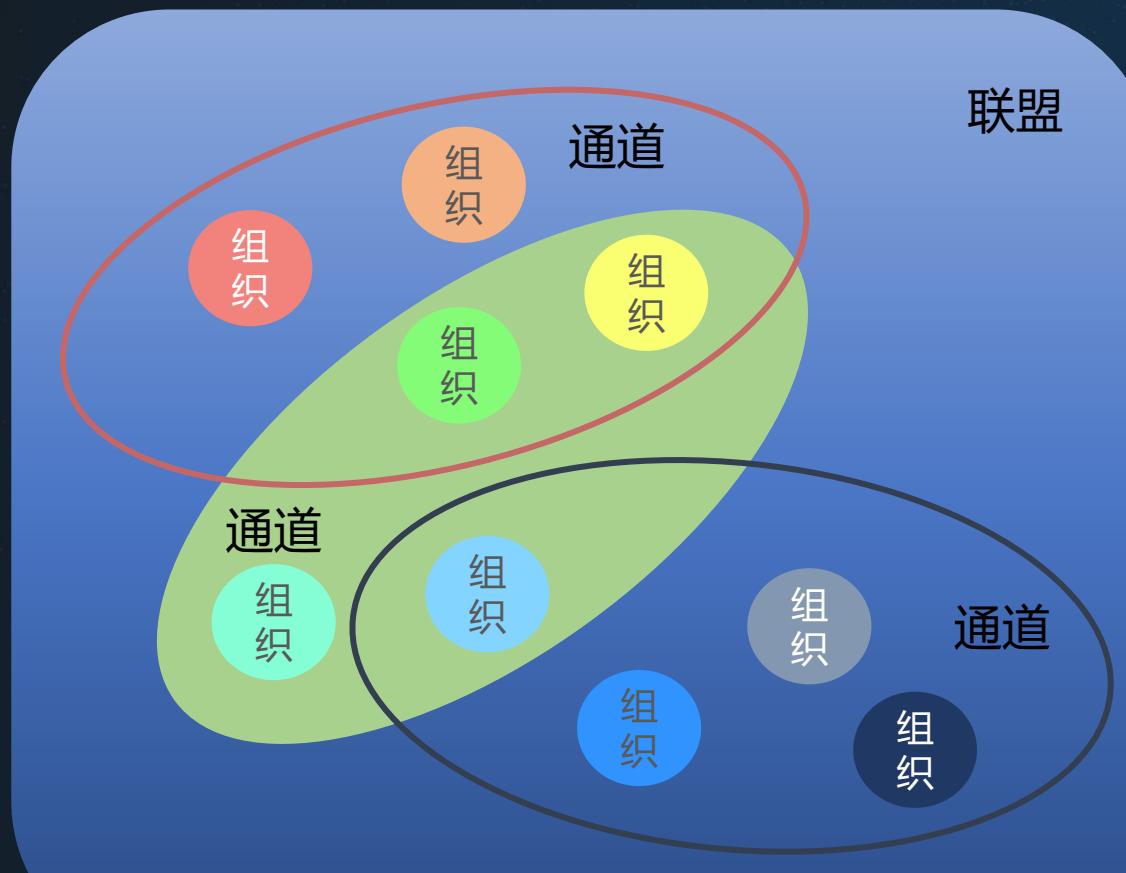
Fabric智能合约（链码Chaincode）

- 超级账本支持使用主流编程语言的智能合约（链码）设计，极大方便了应用开发人员快速开发新型的分布式应用，或将已有应用迁移到区块链系统上。
- 区块链应用，一般由若干部署在区块链网络中的智能合约，以及调用这些智能合约的应用程序组成。
 - ✓ 用户专注于与业务本身相关的应用程序；
 - ✓ 智能合约则封装了与区块账本直接交互的相关过程，被应用程序调用
- 智能合约直接与账本结构打交道，处于十分核心的位置。
 - ✓ 智能合约代码本质上是为了对上层业务逻辑进行支持；
 - ✓ 智能合约最终会部署在区块链网络中与账本进行交互。
- 应用程序通过调用智能合约提供的方法接口来实现业务逻辑。



典型的区块链应用程序的工作过程

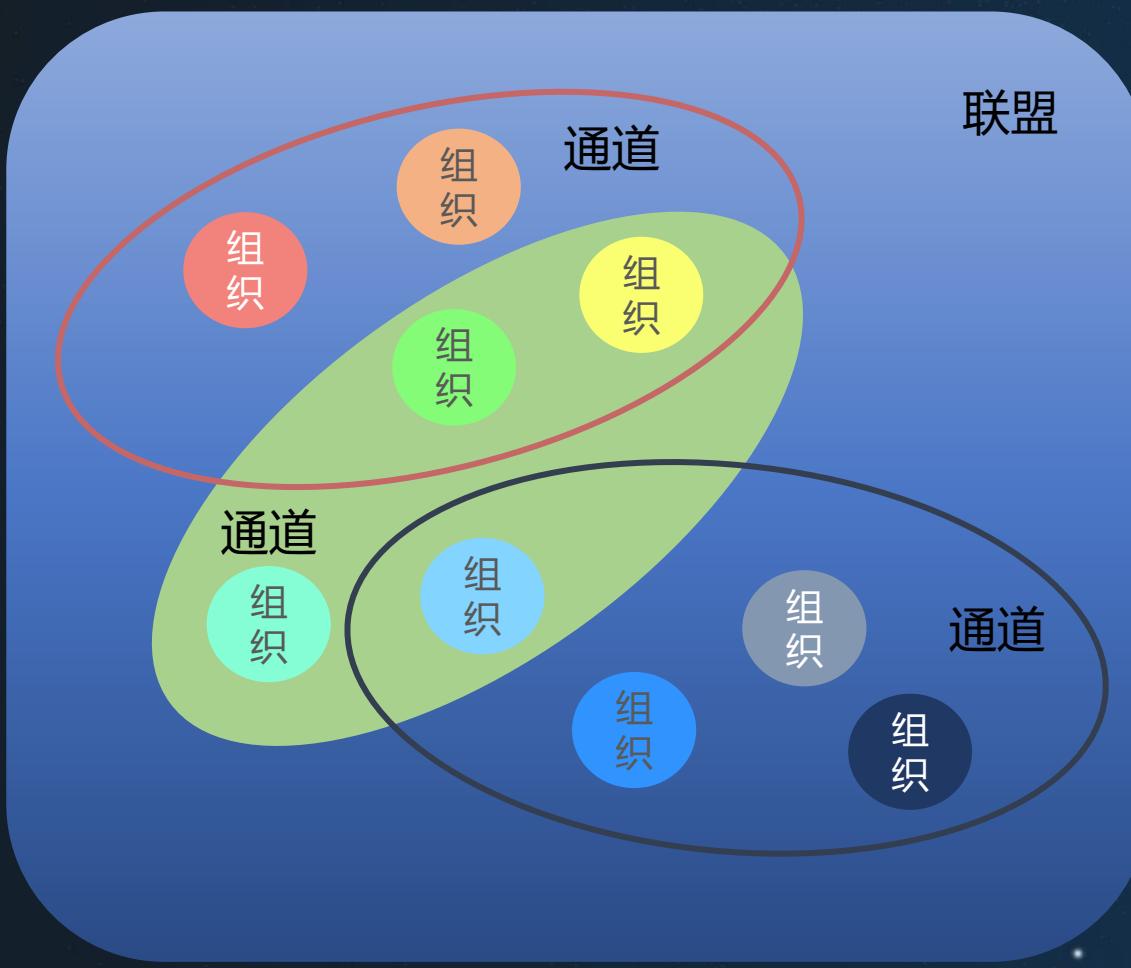
Fabric的联盟结构



联盟

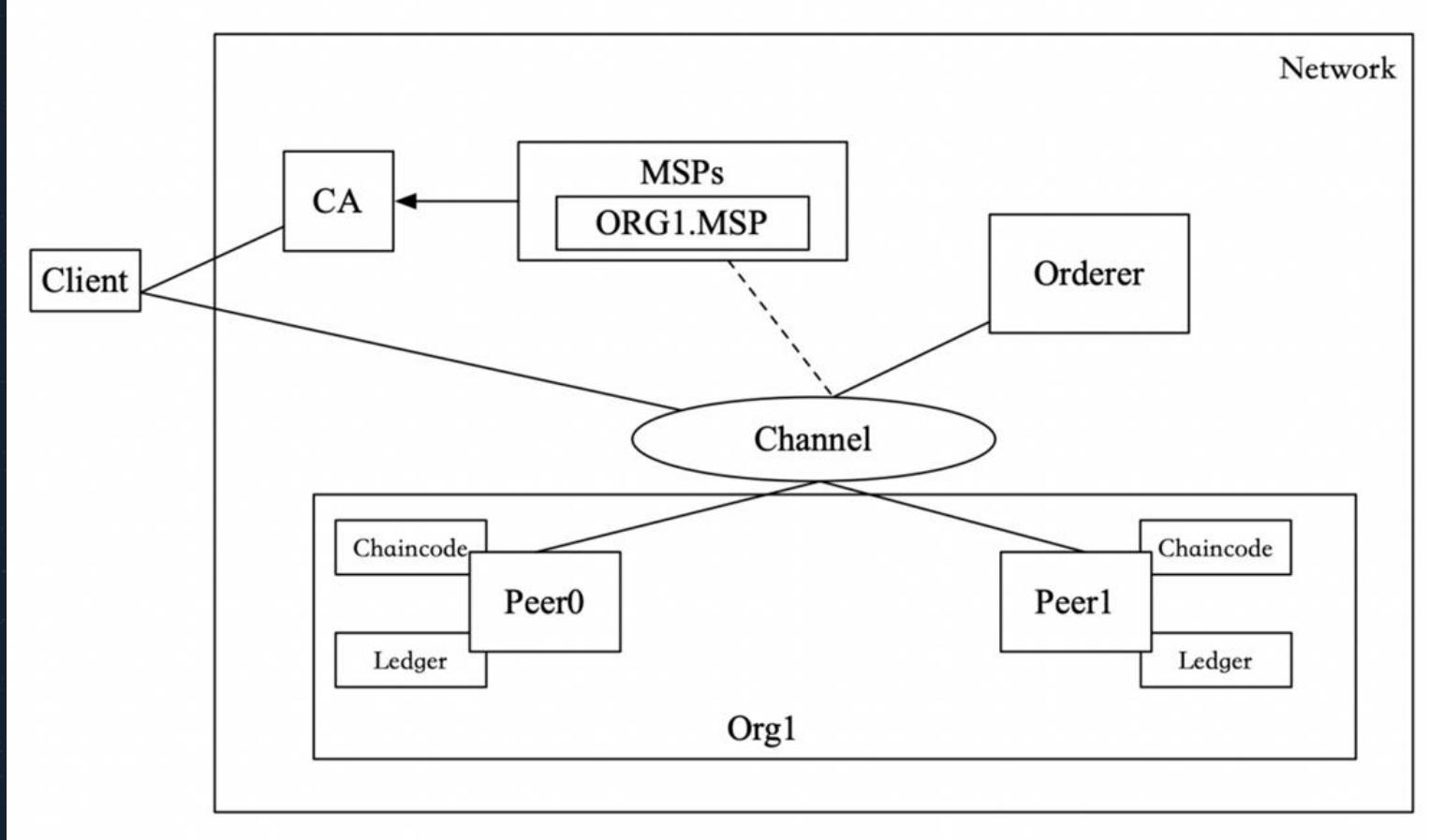


Fabric的联盟结构

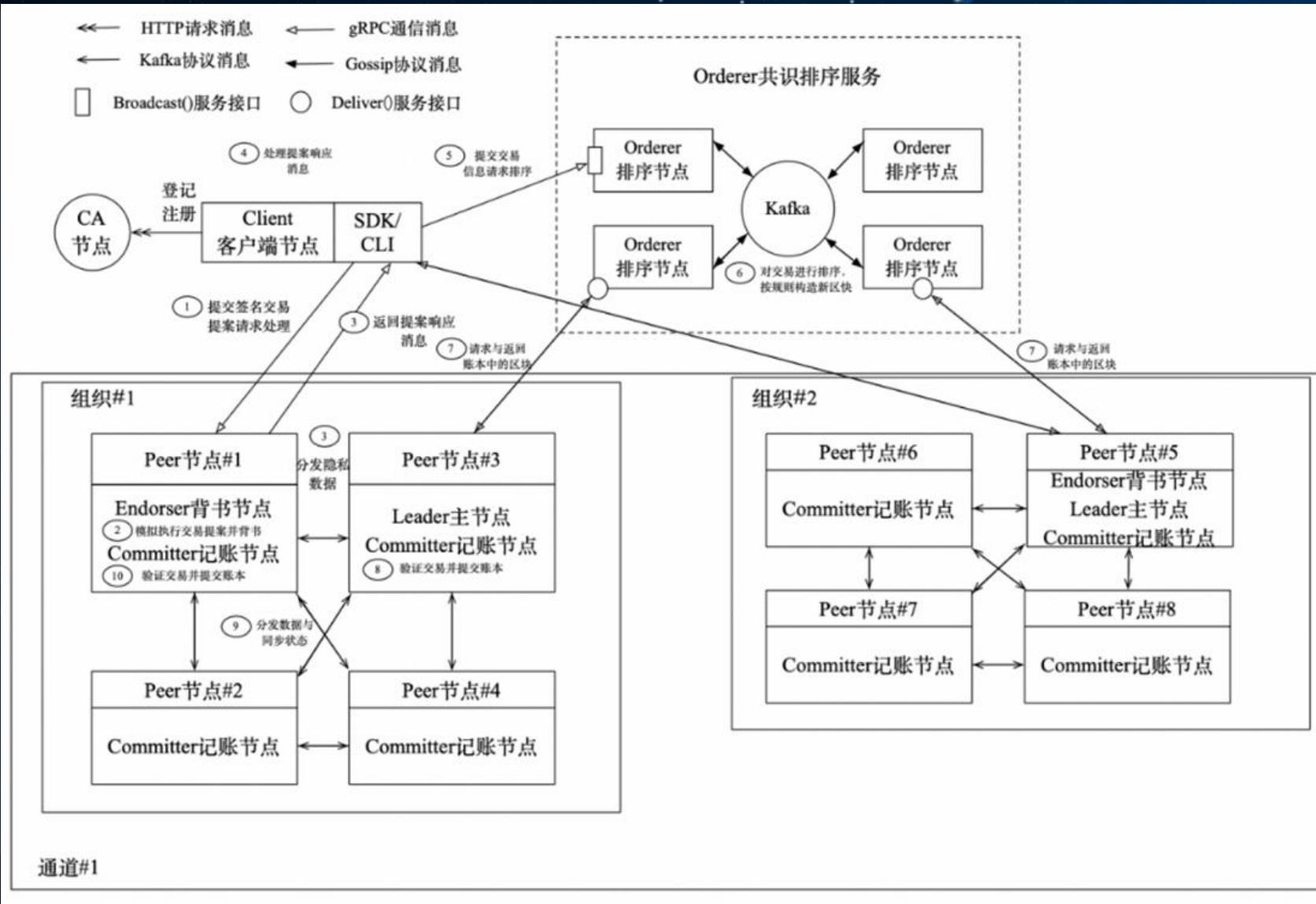


- 联盟：所有有业务往来的组织的集合称为一个联盟
- 组织：一个逻辑上的主体，可以包含了很多节点，这些节点之间互相信任
- 通道：一个具体的应用其对应的组织节点，账本数据等称为一个通道。通道的设立主要是为了提供数据隔离，可以进一步保护隐私
 - 节点：区块链网络上的具体节点。由于Fabric的交易流程的独特性，节点在功能上分为orderer和peer，分别用来产生区块、处理交易和区块。
 - 链码：链码就是智能合约的实例，一个链码可能包含多个合约。
 - 账本：Fabric账本相比于普通区块链的账本范围更广，除了包含链式数据结构（区块链），还包含了有状态数据库，另外还包含了通道和节点本身的一些状态信息

Fabric的运行结构



交易流程

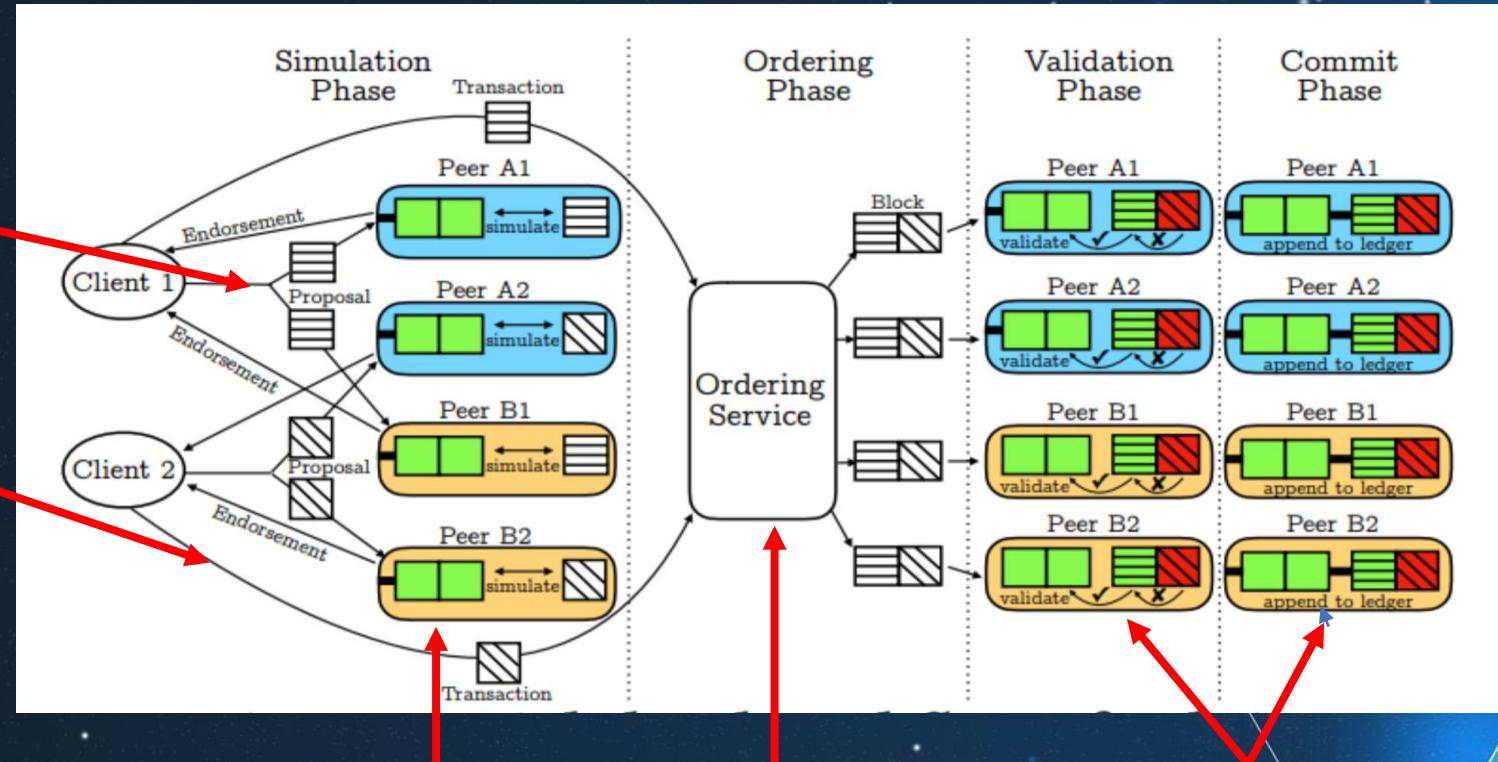


Fabric交易流程-EOV (共识机制)

客户发起交易

检查提案响应
将背书组合进
交易中并提交
给Orderer节点

E-Endorsing/Executing
O-Ordering
V-Validation/Commit



背书节点验
证签名并执
行交易

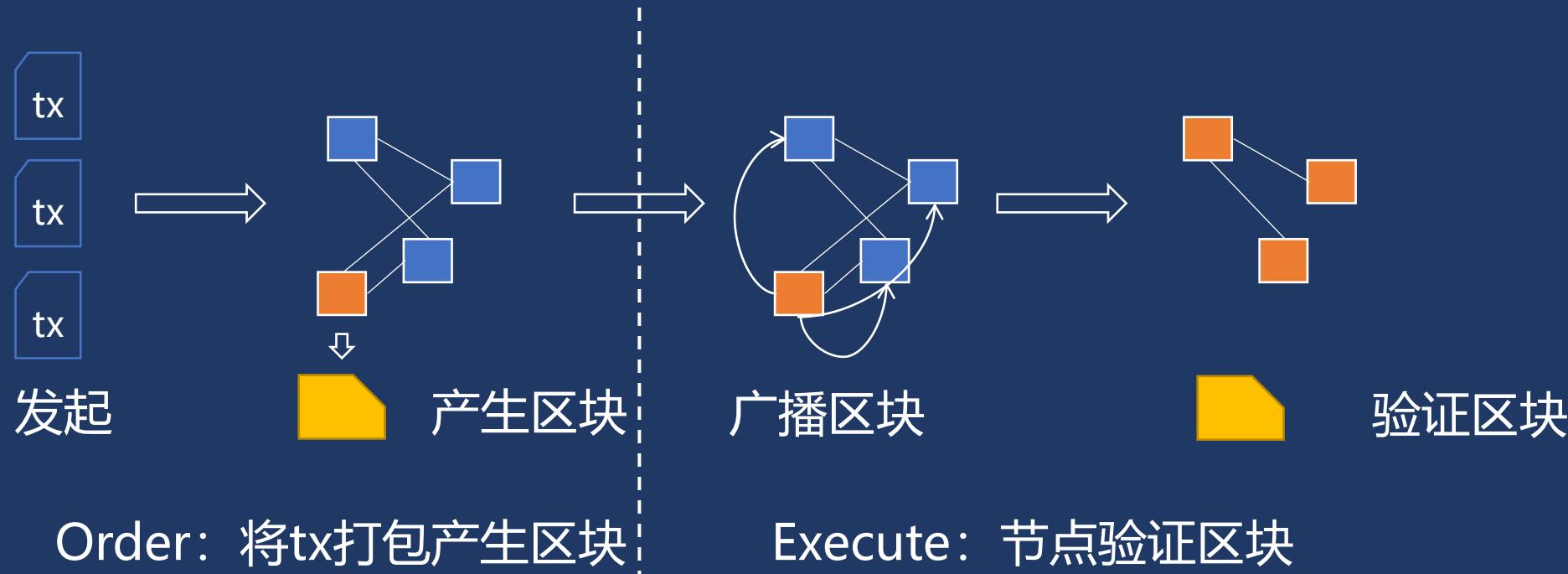
接收交易并
排序、创建
交易区块

验证交易并提交

共识机制对比

Order-Execute(Bitcoin, Ethereum) vs Execute-Order-Validation(Fabric)

- Order排序：交易有序加入账本，然后扩散到所有节点
- Execute执行：交易在所有节点上按顺序依次执行并更新账本



Order: 将tx打包产生区块

Execute: 节点验证区块

共识机制对比

Order-Execute (OE) :

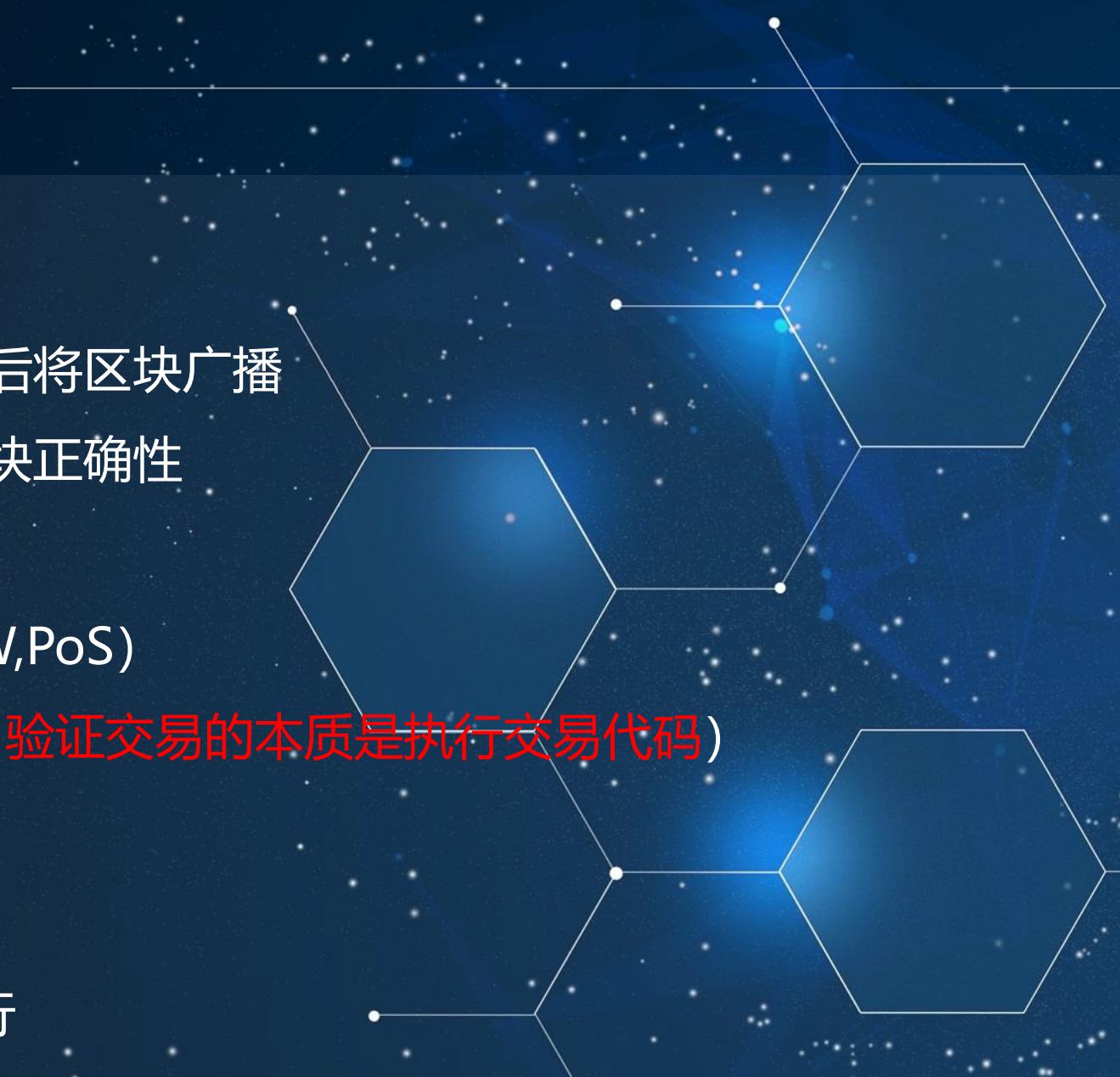
- 1) 先由矿工将交易打包产生区块，之后将区块广播
- 2) 接收到区块的节点会在本地验证区块正确性

特点：

- 矿工验证交易，满足共识算法 (PoW,PoS)
- 节点验证交易和共识算法解（注意：验证交易的本质是执行交易代码）

缺点：

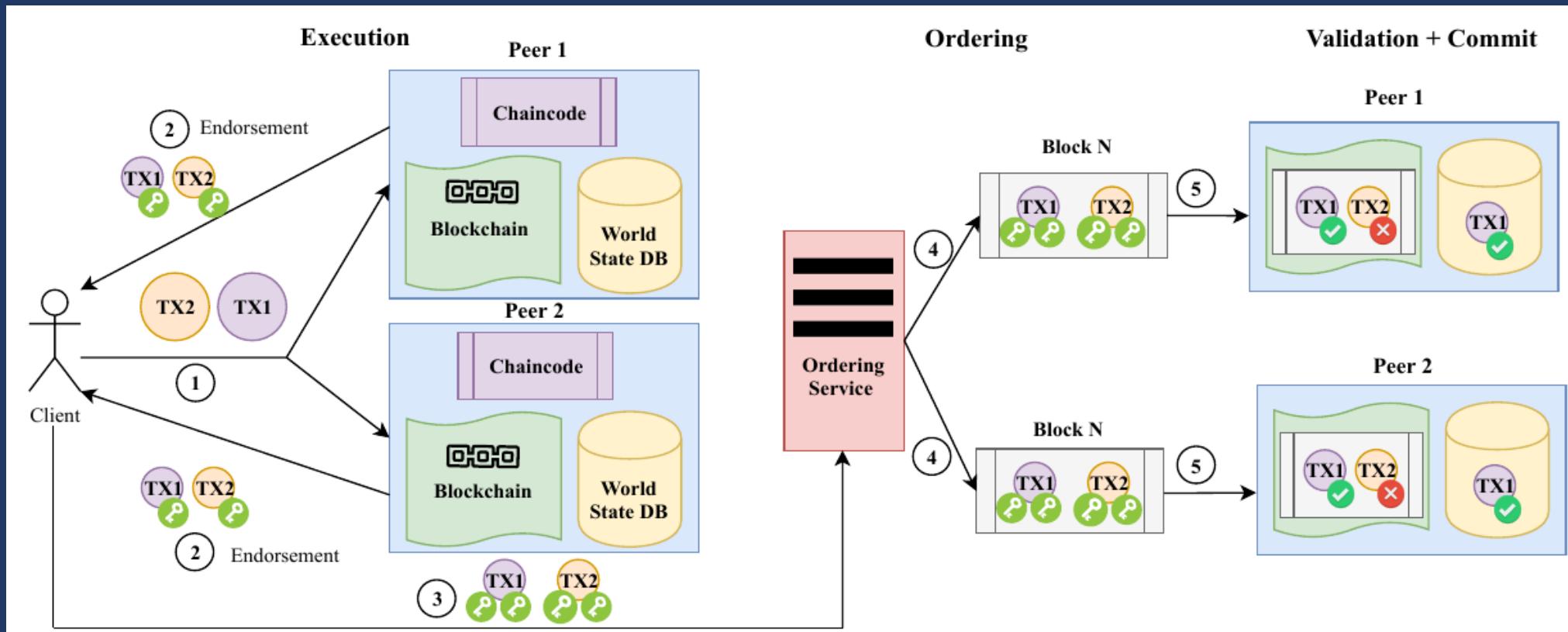
- 共识算法安全，性能低和开销大
- 交易执行需要是deterministic，串行



共识机制对比

Order-Execute(Bitcoin, Ethereum) vs Execute-Order-Validation(Fabric)

- Execute执行：交易（通过智能合约）以任意顺序执行，甚至可以并行执行
- Order排序：当足够数量的节点对交易结果达成一致，该交易就会被加入账本并扩散给所有节点。
- Validation验证：每个节点验证并按顺序确认交易，从而更新账本



共识机制对比

Execute-Order-Commit

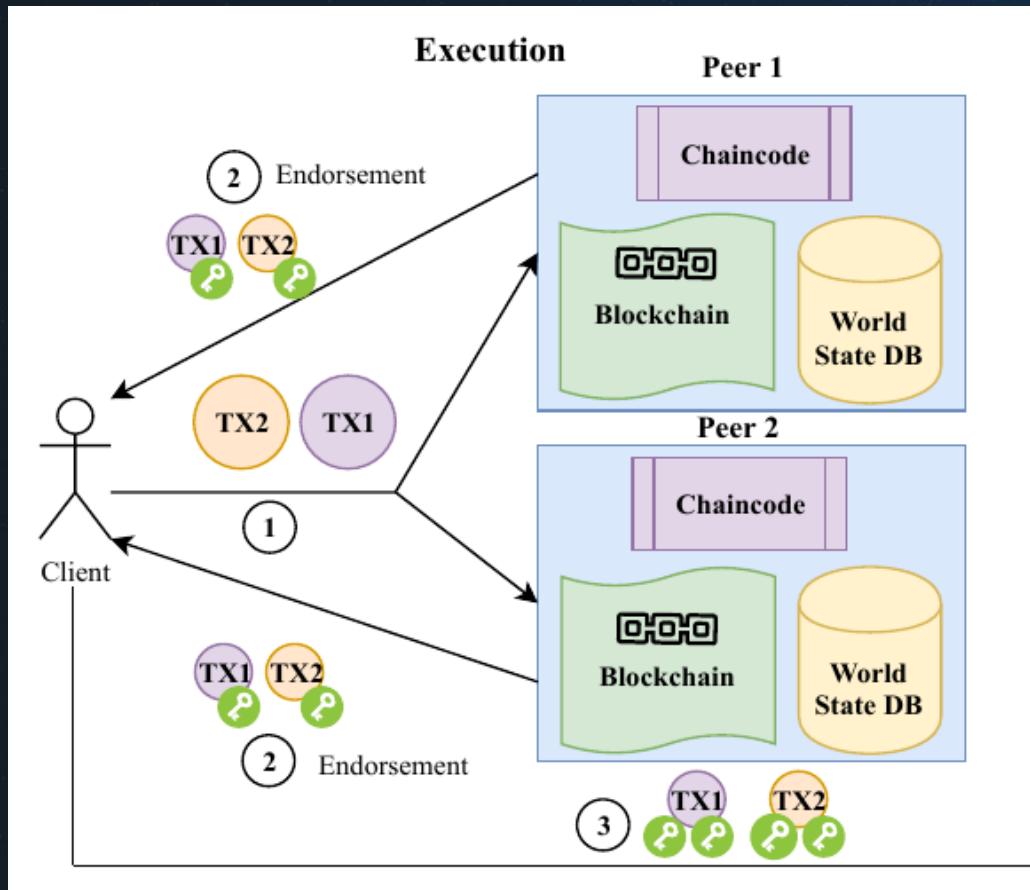
- 1) 客户端发起交易，多个节点会先执行交易并返回结果（**不会更改本地账本**）
- 2) 客户端检测是否满足共识算法，若满足则发给区块产生的节点产生区块
- 3) 所有节点接收到区块后提交执行结果到本地

Fabric共识算法：

主要检查交易执行结果是否一致，是否收到的足够数量的交易结果（**背书策略**）

优点：

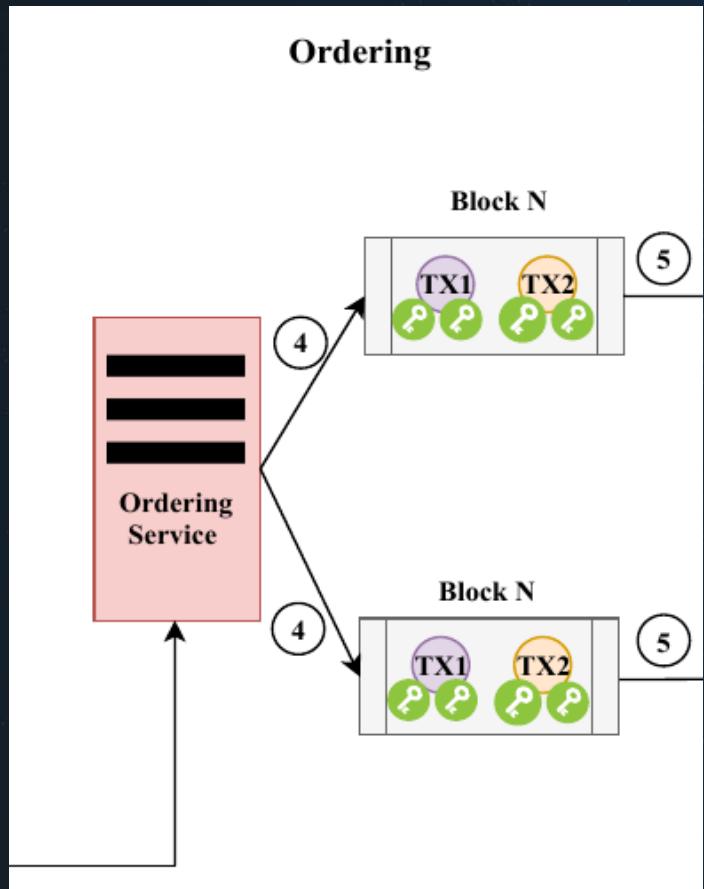
- 共识算法非常快
- 支持non-deterministic的交易
- 交易并发

**Execute:**

- 客户端发送交易请求到若干个背书节点 (背书策略)
- 客户端验证请求响应
- 若验证通过则发送给Order节点

一笔合法交易需要被多少个组织同意的规则，例如
 $\text{And}(\text{'Org1'}; \text{'Org2'}; \text{'Org3'})$
 $\text{Or}(\text{'Org1'}; \text{'Org2'}; \text{'Org3'})$

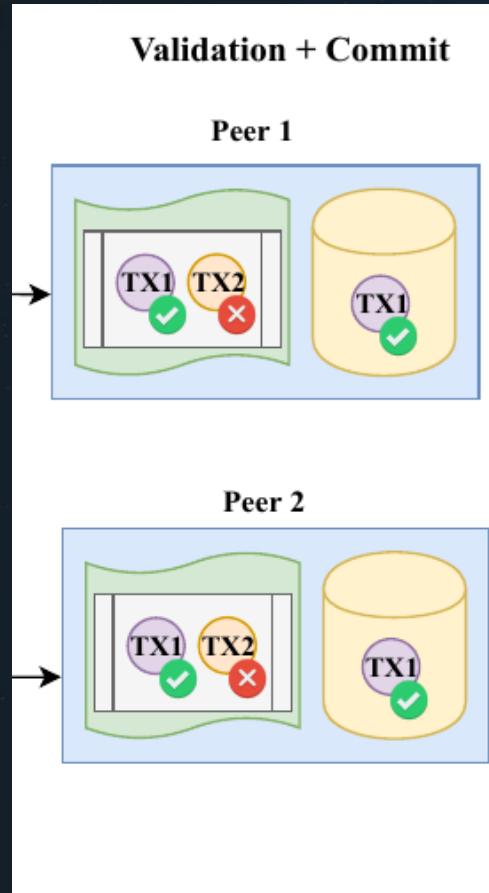
- 签名
- 所有响应读写集是否一致
- 收集到的响应数量是否满足背书策略



Order阶段:

4. Order节点验证接收到的交易请求，并打包成区块
5. Order节点将区块发送给Peer节点

1. 签名
2. 是否满足背书策略



Validate:

Peer节点对区块中的交易进行验证，并添加到链末尾，同时更改状态数据库

1. 背书策略验证
2. 读写冲突检查

交易请求tx1

读集合	写集合
(R_1, V_1)	(W_1, V_1)
(R_2, V_1)	(W_2, V_1)

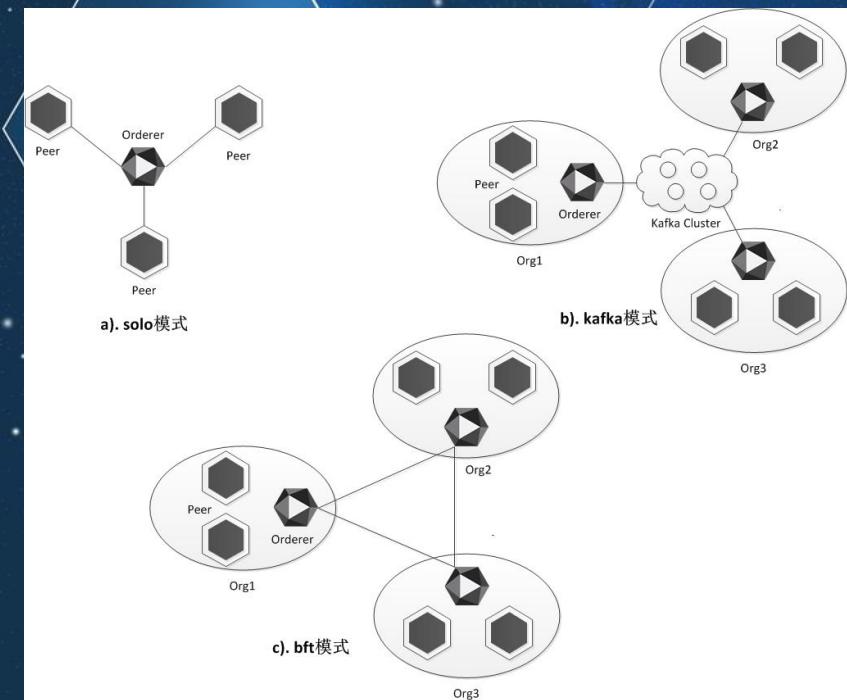
版本是否一致?

节点KV-DB

(R_1, V_1)
(R_2, V_1)
(R_3, V_1)
(R_4, V_1)
(W_1, V_1)
(W_2, V_1)

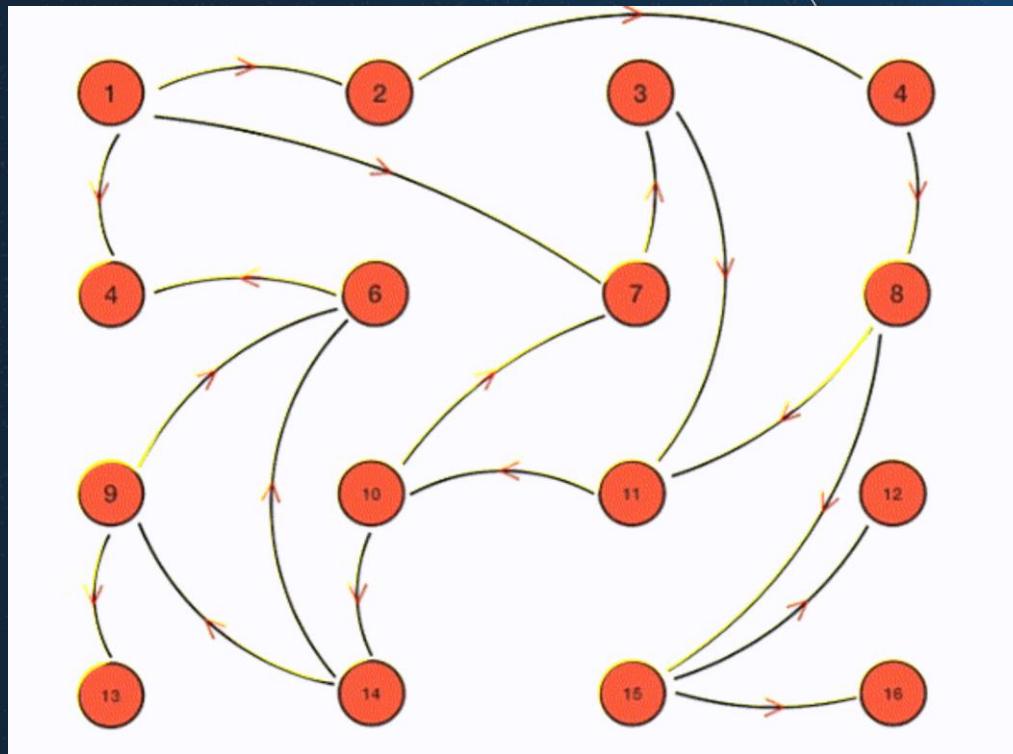
交易排序

- 排序服务在Fabric中建立了所有交易的顺序，在执行阶段每个交易包含了状态更新和计算的依赖，并带有背书节点的签名。Orderers不知道应用状态，也不参与交易的执行或者校验。这种设计使Fabric中的共识尽可能模块化，并简化了Fabric中共识协议的替换。
- Solo模式：整个Fabric网络依赖于一个Order节点。
- Kafka/Raft：模式依赖于一个Kafka/Raft Order集群。
- BFT（拜占庭容错）模式则是去中心化的Order集群。



Fabric的P2P协议—gossip protocol

- 需要一种通信方式来保证各个节点之间的传播可信、可扩展，并且可以支持拜占庭容错。Fabric在其框架中使用的就是gossip协议。



10.3 Fabric核心概念

- 身份和MSP
- Fabric网络
- 通道
- 策略
- 节点
- 账本
- 链码

Hyperledger Fabric 关键概念



通道 (Channel)

- 通道Channel是Fabric中非常重要的概念，是用于实现区块链网络中业务的隔离，一个联盟多个通道，每个通道可代表一项业务，并且对应一套账本，通道内的成员为业务参与方（即联盟内的组织），一个组织可以加入多个通道。通道分为：
 - 系统通道
 - 应用通道
- 排序节点通过系统通道来管理应用通道，用户的交易信息通过应用通道传递，对一般用户来说，通道是指应用通道。
- 通道也可以理解为子链
- 1个通道+1个账本+N个成员

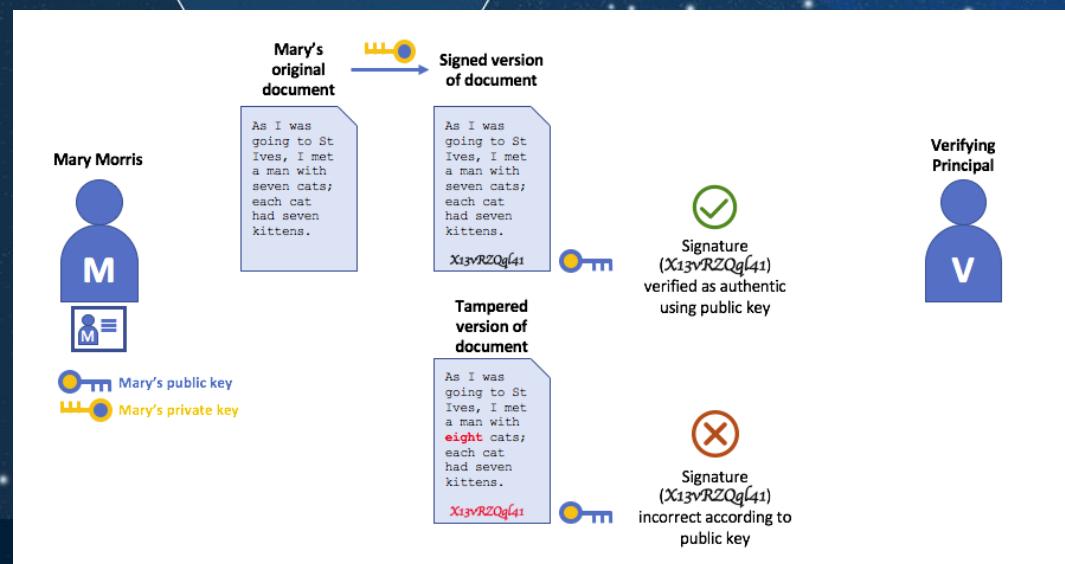
系统通道 (System Channel)

- 系统通道是一个特殊的通道，它与应用通道 (Application Channel) 相对。系统通道具有以下特点和作用：
 - 网络配置：系统通道用于配置整个区块链网络的初始设置，包括定义网络的基本参数和属性。
 - 组织管理：系统通道中包含了网络中所有组织的MSP（成员服务提供者）定义，这些定义确定了哪些实体是网络中的合法成员。
 - 共识配置：系统通道中配置了整个网络的共识参数，包括共识类型（如Kafka、Raft等）和相关的共识节点。
 - 创世区块：系统通道拥有网络的创世区块（Genesis Block），这是一个特殊的区块，包含了网络的初始状态和配置信息。
 - 根权限：系统通道通常由网络的初始创建者或管理员控制，具有网络中的最高权限。
 - 锚节点：系统通道可以配置锚节点（Anchor Peer），这些节点作为其他通道中组织之间的桥梁，用于跨通道通信。
 - 网络更新：对网络进行重大更新（如添加或移除组织、更改共识机制等）时，这些变更需要在系统通道中进行。
 - 身份认证：系统通道中的身份信息用于网络中其他通道的身份验证和授权。
 - 网络治理：系统通道是网络治理的基础，网络的规则和政策在这里被定义和维护。

系统通道是Hyperledger Fabric网络中不可或缺的一部分，它为网络的初始化、配置和管理提供了必要的机制。通过系统通道，网络管理员可以对整个区块链网络进行有效的控制和管理。

认证和身份管理

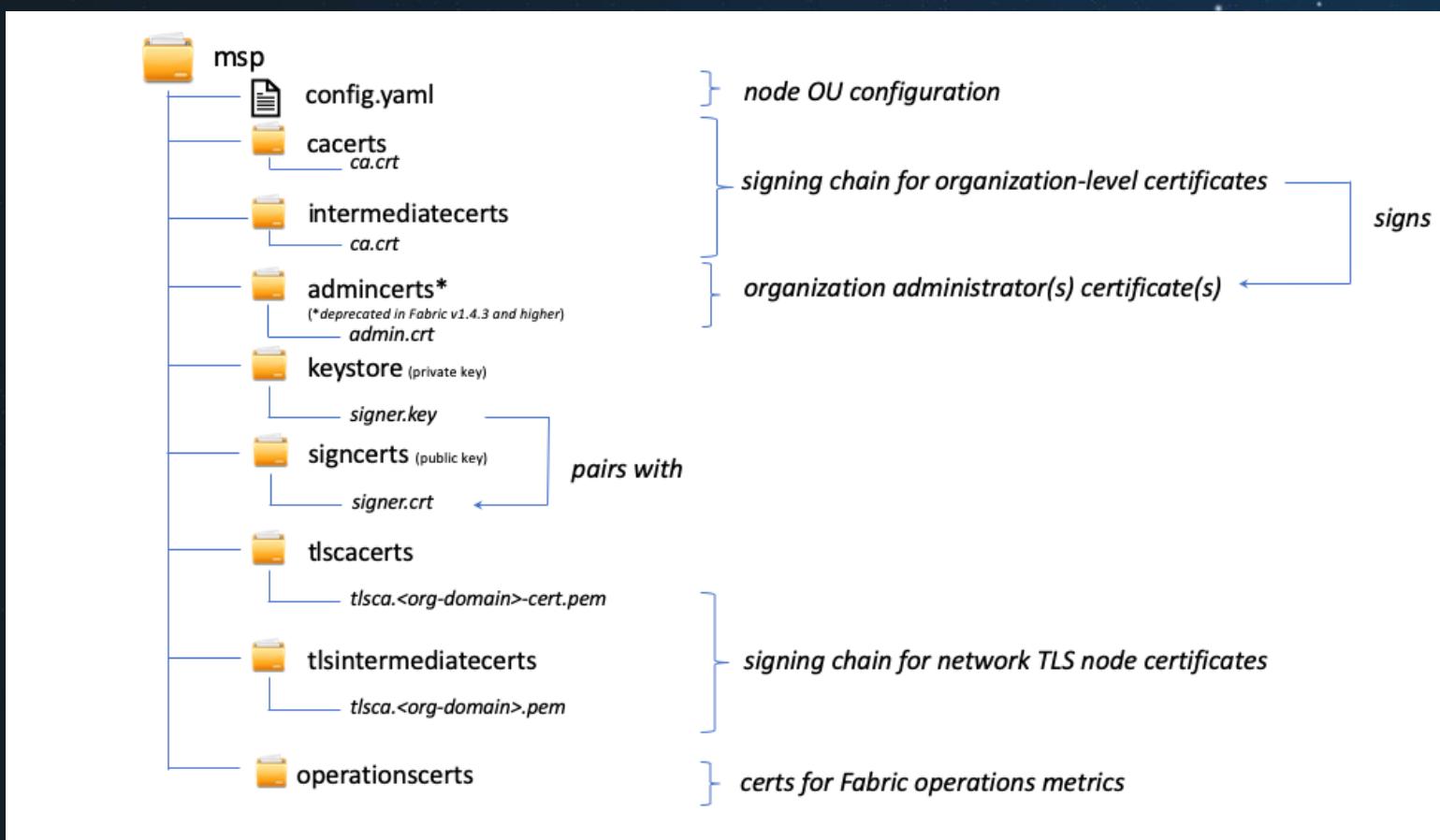
- Fabric是一个许可网络，区块链参与者需要一种方法来向网络的其余部分证明他们的身份，以便在网络上进行交易。在一个Fabric网络中，不同参与者包括Peer节点、排序节点、客户端应用程序、管理员等都需要确定**对资源的确切权限以及对参与者在区块链网络中拥有的信息的访问权限**，同时，与参与者身份相关联的主体信息，也需要**可被验证**。
- Fabric采用**Membership Service Provider (MSP)** 来抽象出颁发证书、验证证书和用户身份验证背后的所有加密机制和协议。MSP可以定义自己的身份概念，以及管理这些身份（身份验证）和认证（签名生成和验证）的规则。MSP使用X.509证书作为身份，采用传统的公钥基础结构（Public Key Infrastructure,PKI）分层模型实现。
 - 公钥加密，私钥解密；
 - 私钥签名，公钥验证；
 - 数字签名证书的机构（CA），是负责发放和管理数字证书的权威机构，并作为电子商务交易中受信任的第三方，承担公钥体系中公钥的合法性检验的责任；
 - CA通过中间CA组建信任链，规避风险；
 - Fabric提供了内置了Fabric CA组件帮助用户在区块链网络中创建CA。



认证和身份管理

- 根CA和中间CA构成的认证链，提供了身份证明；Fabric提供了一个内置的CA组件，允许在所形成的区块链网络中创建CA。
- Membership Service Provider (MSP)将为区块链网络中的参与者提供一个允许的身份列表；
- 在区块链网络中，MSP主要在本地节点 (local MSP) 和通道 (channel MSP) 两个域内生效；
- 本地MSP是为客户端和节点 (peer节点和排序节点) 定义的，每个节点都必须定义一个本地MSP，定义该**节点的管理权和参与权**。组织、组织的管理员、节点管理员和节点本身都必须有相同的信任根；
- 通道MSP定义了**通道级别的管理权和参与权**，识别谁拥有通道级别的权限；
- local MSP只在本地节点或用户的文件系统实例化，channel MSP会在通道的每个节点实例化，并通过**共识保持同步**；
- 如果组织希望加入通道，则需要将包含该组织成员信任链的MSP包含在通道配置中，否则，来自该组织身份的交易将被拒绝；
- 建议每个参与通道的组织都定义其MSP，以提升权限管理的颗粒度，更好的保证组织和用户隐私；
- 系统级别的通道包含参与服务的所有组织的MSP；

本地MSP结构和通道MSP

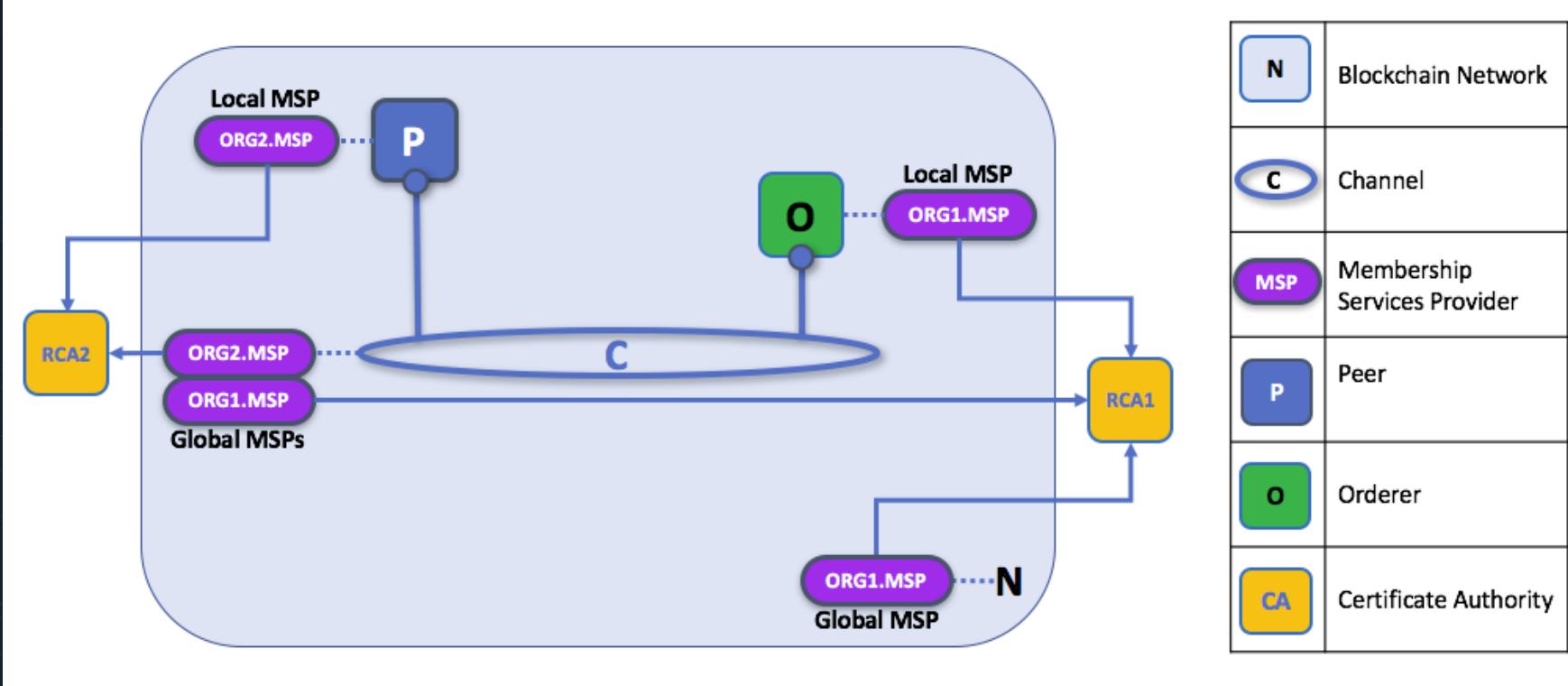


本地 MSP 表示为文件系统上的文件夹结构

```
"channel_group": {  
    "groups": {  
        "Applicat...": {  
            "groups": {  
                "Org1MSP": {  
                    "Org2MSP": {}  
                }  
            }  
        }  
    }  
}
```

通道MSP则在通道配置中进行描述

MSP和区块链网络



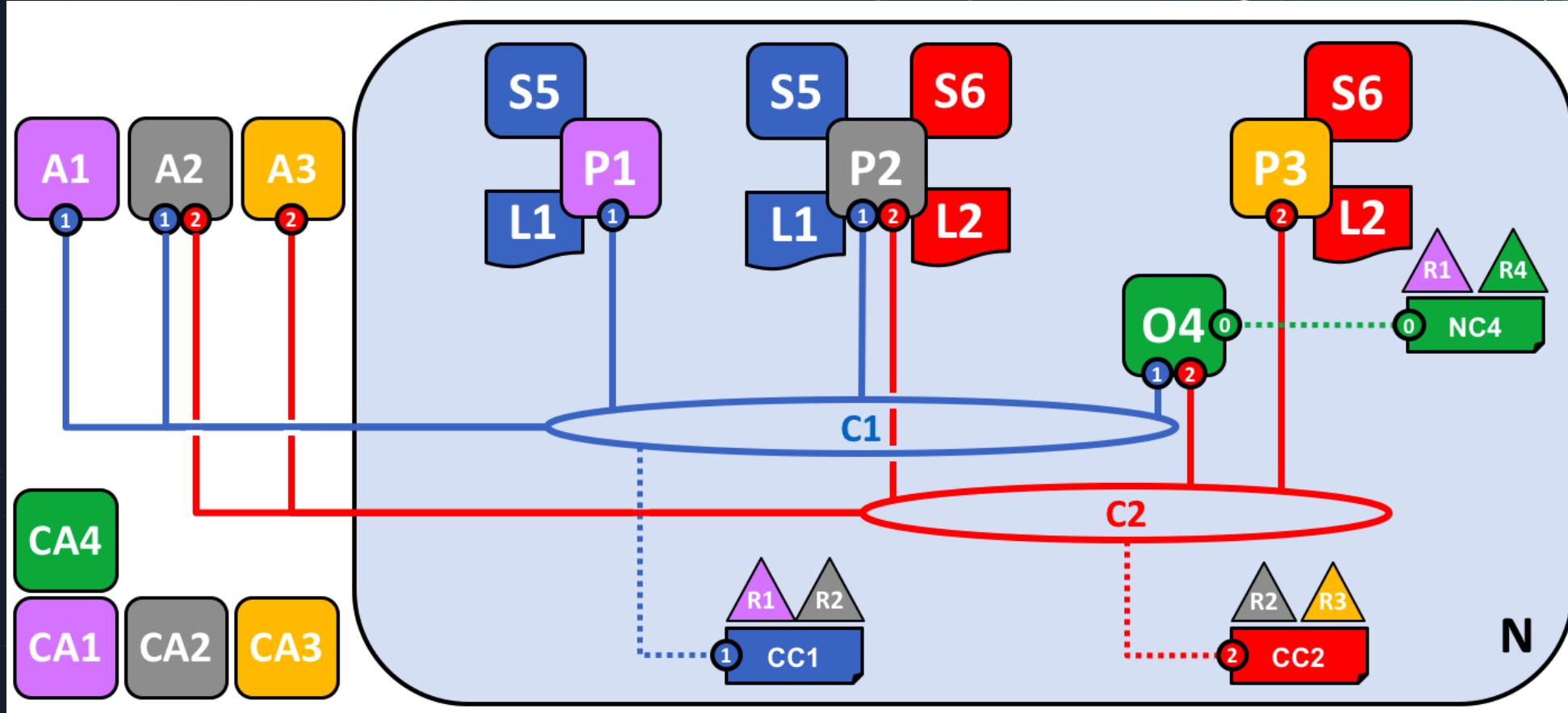
peer和orderer的MSP是本地的，而通道的MSP是全局的，由该通道的所有参与者共享。网络系统通道由 ORG1管理，但另一个应用程序通道可以由ORG1和ORG2管理。peer是ORG2的成员并由ORG2管理，而 ORG1管理orderer。ORG1信任来自RCA1的身份，而ORG2信任来自RCA2的身份。需要注意的是，这些是管理身份，反映了谁可以管理这些组件。

Fabric Network

Fabric网络是一个为应用程序提供账本及智能合约（Chaincode）服务的技术基础设施。首先，智能合约被用来生成交易，接下来这些交易会被分发给网络中的每个节点，这些交易会被记录在它们的账本副本上并且是不可篡改的。这个应用程序的用户可能是使用客户端应用的终端用户，或者是一个区块链网络的管理员。

在大多数情况下，多个组织会聚集到一起作为一个联盟来形成一个网络，并且它们的权限是由一套在网络最初配置的时候联盟成员都同意的规则来决定的。网络的规则也可以在联盟中的组织同意的情况下随时被改变。

Fabric Network



A: Application-应用

S: Smart Contract 智能合约

NC: Network Config

CA: Certificate Authorities 证书颁发机构

P: Peer 节点

CC: Channel Config

C: Channel 通道

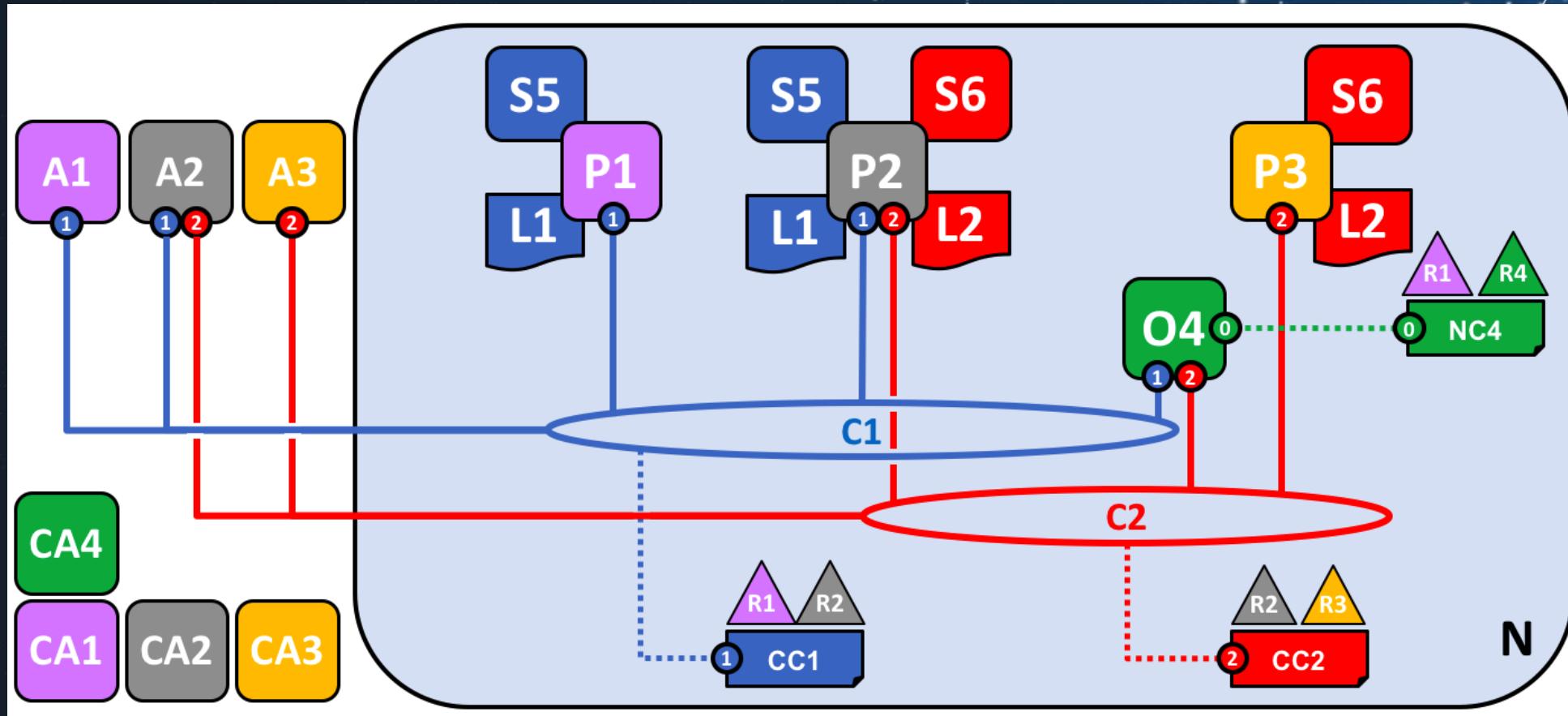
L: Ledger 账本

R: Organization 组织

背景设定（假定）

- 科大、滁州学院、工大和省网中心 (R1-R4) 共同决定搭建一个Hyperledger Fabric网络；
- 省网中心 (R4) 为网络的初始创建者，负责配置网络的初始版本（网络配置NC和排序O），省网中心不参与网络中的数据交易；
- 省网中心 (R4) 指派科大为网络的管理员，并共同管理网络配置 (NC) 和排序服务 (O)；
- 科大 (R1) 和滁州学院 (R2) 因“共建共享在线课程”项目 (A1) 有私有通讯 (C1) 的需求，并由两个组织协商制定了网络规范 (CC1)；
- 滁州学院 (R2) 和工大 (R3) 因“E会学”项目 (A3) 有私有通讯 (C2) 的需求，并由两个组织协商制定了网络规范 (CC2)；
- 科大、滁州学院和工大使用“线上课程资格证书颁发”项目 (A2)，A2同时拥有在C1和C2通信的权限；
- 各通道 (C1和C2) 遵循排序规则并使用排序规则 (O)；
- 科大、滁州学院、工大分别部署了一个节点 (P1-P3)，其中P1同步了C1的账本、P2同步了C1和C2的账本、P3同步了C2的账本；
- 应用 (A1-A3) 通过智能合约 (S) 访问账本 (L1-L2)；
- 组织之间通过CA来辨识用户、应用等的合法性；

Fabric Network



A: Application-应用

CA: Certificate Authorities 证书颁发机构

C: Channel 通道 L: Ledger 账本

S: Smart Contract 智能合约

P: Peer 节点

O: Ordering 排序

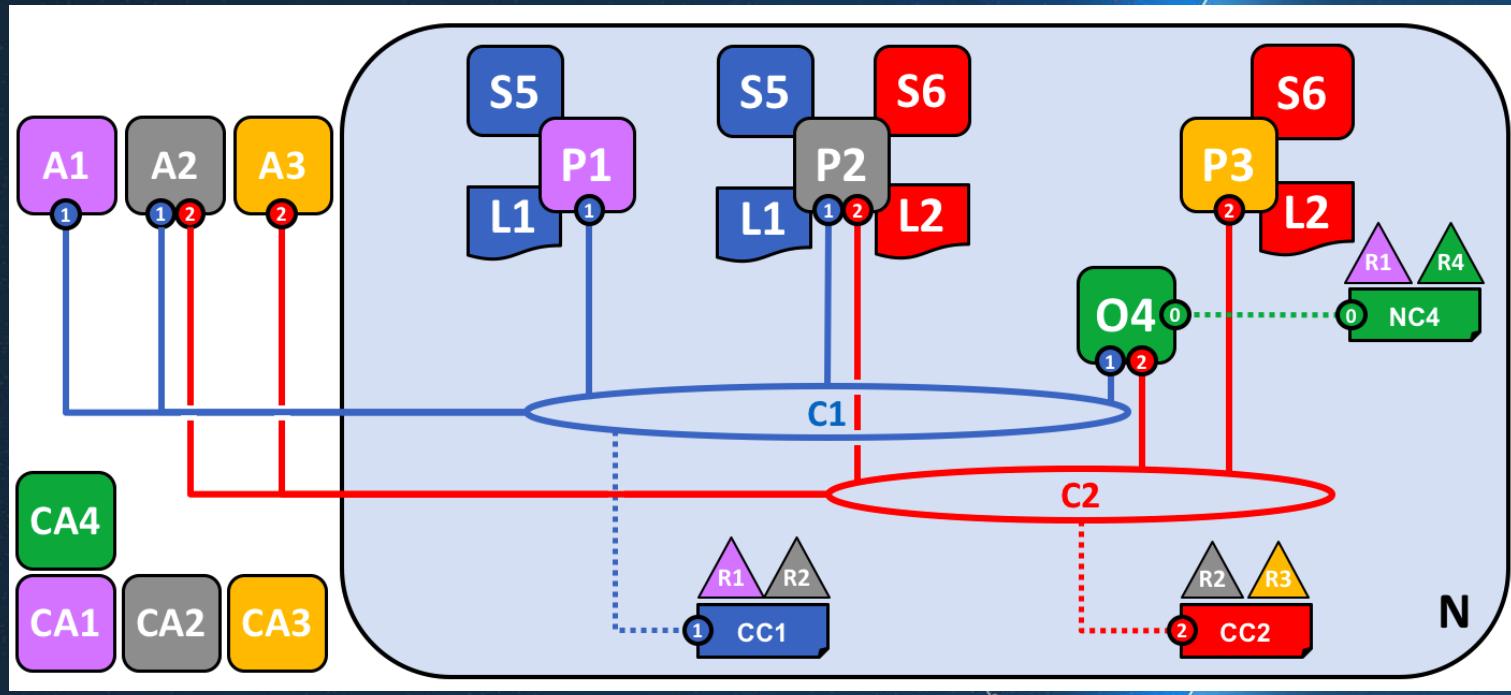
R: Organization 组织

NC: Network Config

CC: Channel Config

Fabric Network网络演进

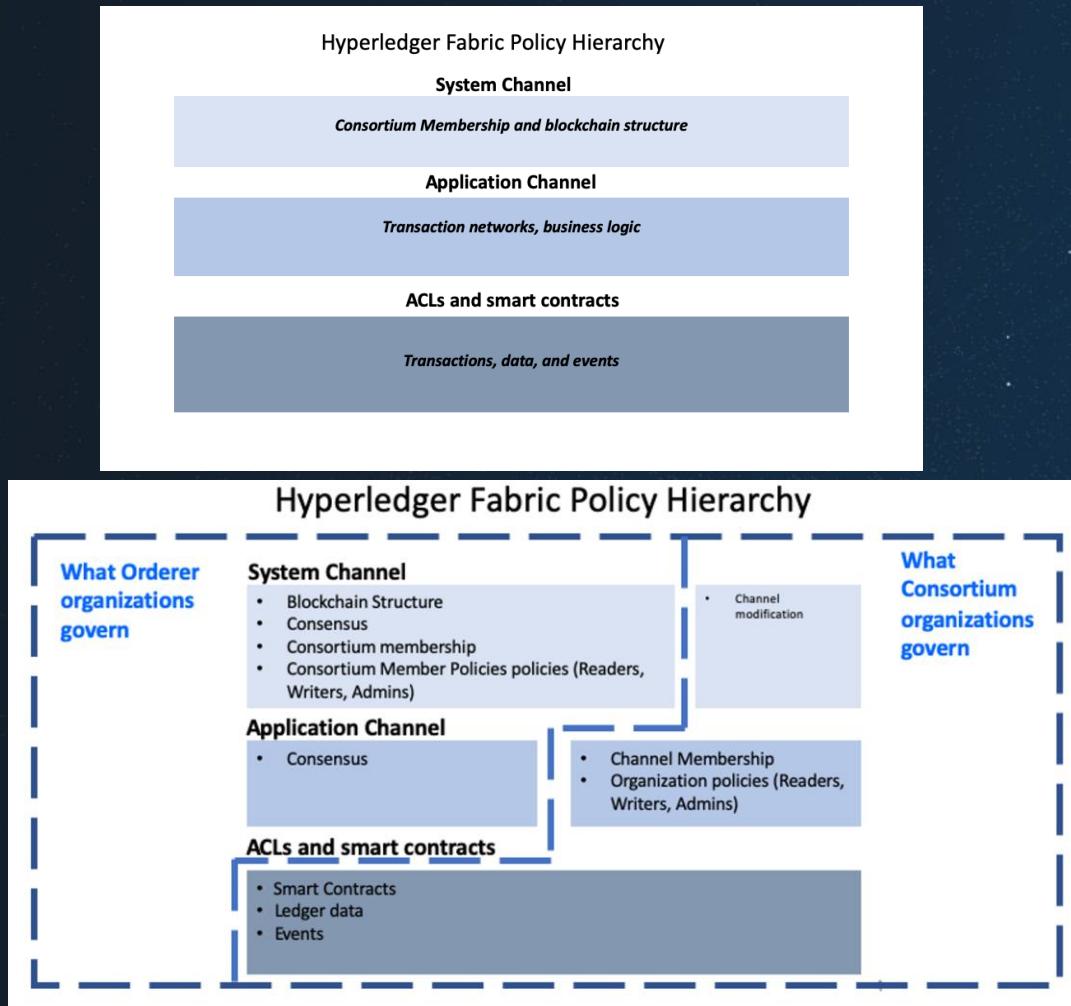
40. 智能合约通过链码逻辑
确保账本逻辑与通道逻辑实现
一致性，由管理员签发证书，
并初始化Fabric网络，
智能合约S5分别被部署在序
号为1和2的节点上。Fabric
将智能合约部署到不同的节点。
C1和通道首CC1配置本机构的
数据端应用服务CA1，保证方
块与通道C1进行交互；
智能合约S5提供给整个网
络，其获得了通道C1的
账本逻辑L1，账本逻辑上
存于C1，但物理上还是存
在各节点的拷贝L上的；



策略-Policies

- 认证和身份管理解决了区块链网络上的身份认证和管理权、参与权确权问题；
- Fabric策略是一组定义了**如何制定决策和达到特定结果的规则，是Fabric基础设施上的管理机制，负责管理网络成员在修改网络、通道或智能合约时如何达成一致**；例如，它们描述了在通道中添加或删除成员的标准，更改区块的形成方式，或指定背书智能合约所需的组织数量。
- 所有这些操作都由定义谁可以执行操作的策略来描述。简单的说，Fabric网络上的任何操作都受策略控制；
- 策略是Fabric与以太坊和比特币等其他区块链有所不同的原因之一：
 - 在以太坊和比特币区块链中，交易可以由任意节点生成和验证；管理网络的策略在任何时间点都是固定的，并且只能使用管理代码的相同过程进行更改。
 - Fabric是一个许可的区块链，其用户被底层基础设施识别，这些用户有能力在网络启动之前决定网络的治理，并改变正在运行的网络的治理。

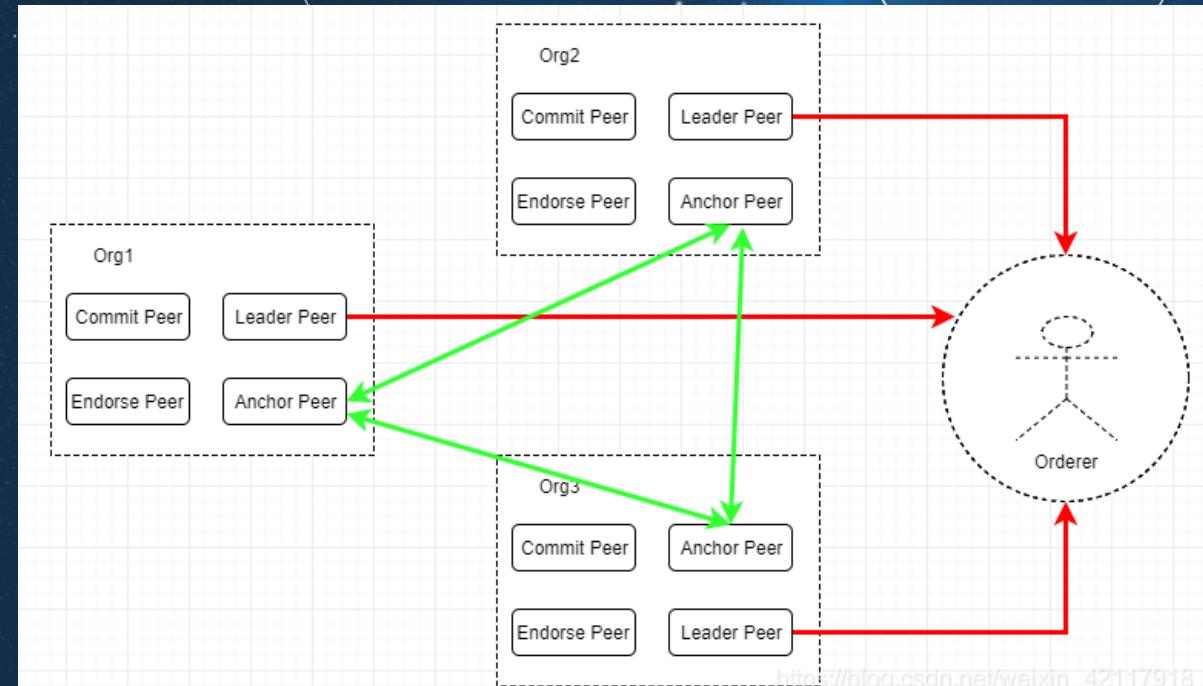
Fabric策略域



- 系统通道配置：每个网络都以一个排序系统通道开始，系统通道还包含作为排序服务成员的组织（排序组织）和在网络上进行交易的组织（联盟组织）。排序系统通道配置块中的策略管理排序服务使用的共识，并定义如何创建新块。系统通道还管理允许联盟的哪些成员创建新通道。
- 应用通道配置：应用程序通道用于在联盟中的组织之间提供私有通信机制。其策略控制从通道中添加或删除成员的能力，还管理在定义链码并将其提交到通道之前需要哪些组织批准链码。初始创建应用程序通道时，默认从orderer系统通道继承所有排序服务参数。但是，可以在每个通道中自定义这些参数（以及管理它们的策略）。
- ACL和智能合约：ACL提供了通过将资源与现有策略相关联来配置对这些资源的访问能力。每个智能合约都有一个背书策略，该策略指定属于不同通道成员的多少对等方需要针对给定的智能合约执行和验证交易，才能使交易被视为有效。

Fabric节点

- 节点Peers：表示组织中的节点，Peer以区块的形式从Orderer节点接收有序的状态更新，维护状态和账本。Peer节点有以下几种：
 - 确认节点Committer：验证数据并保存到账本
 - 背书节点Endorser：对结果进行背书，返回提案响应给客户端
 - 锚节点Anchor：通道中每个组织（Org）都有一个锚节点，用于同一通道中不同组织的Peer节点发现通道内所有Peer节点。
 - Leader节点：节点代表，连接到Orderer节点，从Orderer节点的批量区块广播给其他节点。

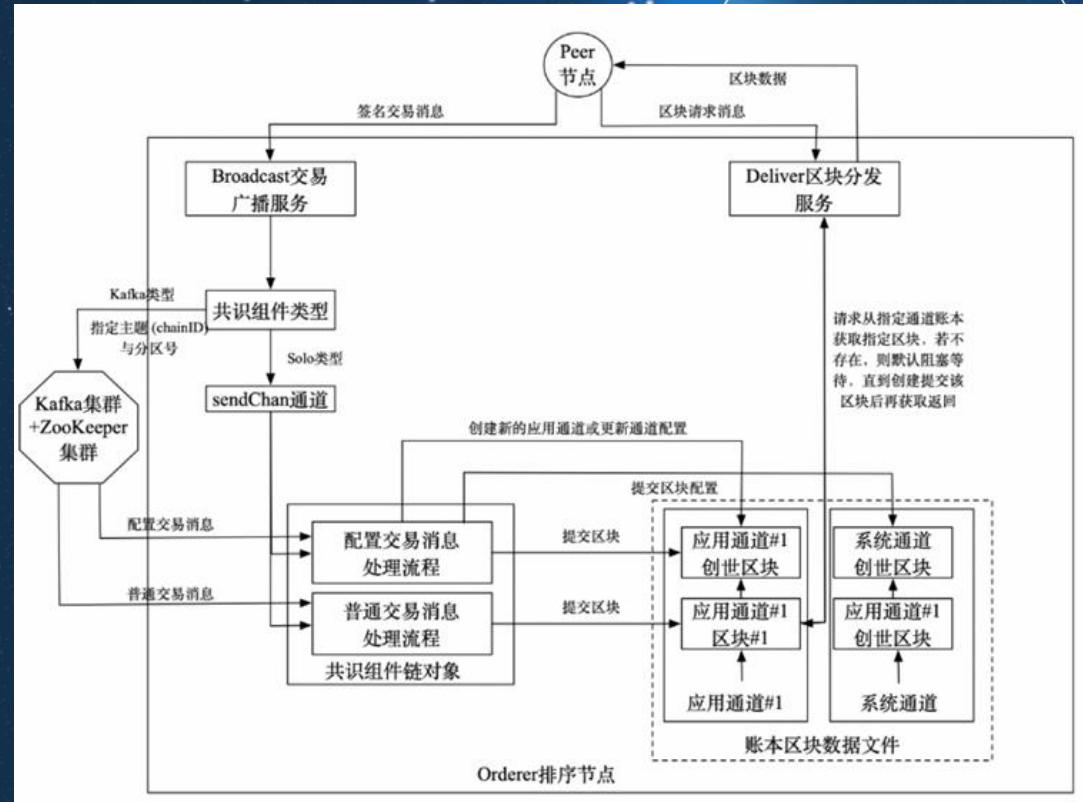


Fabric节点

- 排序节点Orderer: 为网络中所有合法交易进行全局排序，并将一批排序后的交易组合生成区块结构。
- 证书颁发机构CA节点: 负责组织内部成员用户生成和颁发数字证书。

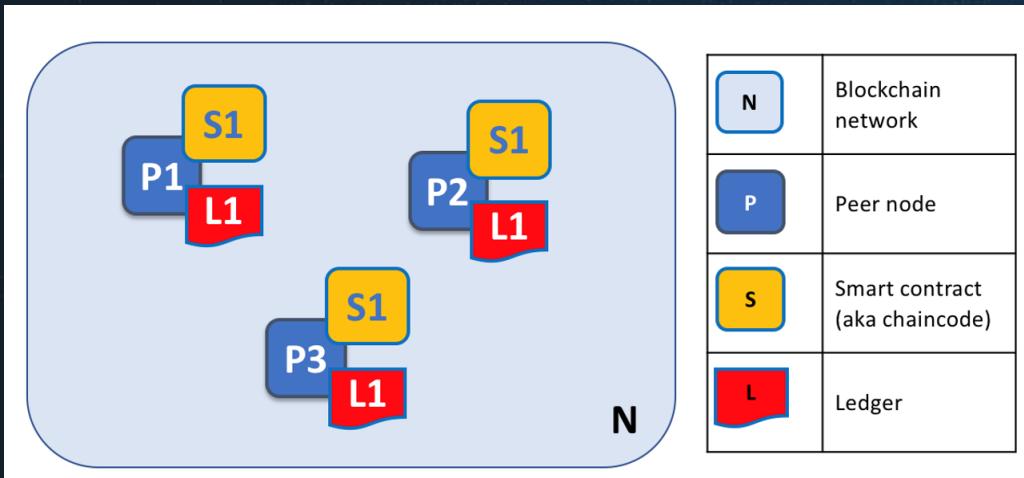
排序节点和排序服务 Ordering

- 排序服务在超级账本Fabric网络中起到十分核心的作用。所有交易在发送给Committer节点进行验证接受之前，需要先经过排序服务进行全局排序。
- 因为Fabric的设计依赖于确定性的共识算法，所以任何由普通节点验证的块都是最终的和顺序一致的。
- 除此之外，将链码执行（发生在普通节点上）的背书与排序分离，使Fabric在性能和可伸缩性方面具有优势，消除了在执行和排序由同一节点执行时可能出现的瓶颈。

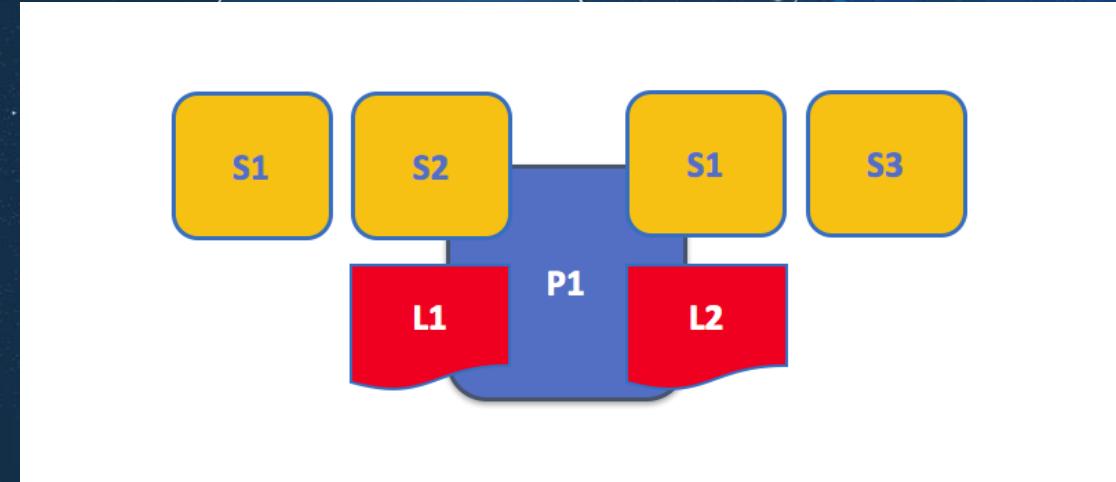


节点-Peers

- 节点是Fabric的基本元素，可以被创建、启动、停止、重新配置甚至删除。它们曝露了一系列的API，让管理者和应用程序可以同这些API提供的服务互动；
- 节点维护了一份或多份账本，并且每个账本都有一个或多个智能合约（链码）使用这个账本；
- 账本数量和访问账本的链码的数量之间没有固定的关系。一个Peer节点可能会有很多链码和账本。
- 节点维护的账本和链码具有冗余性，这是Fabric网络为了避免单点失效所采用的策略，也是Fabric网络去中心化和分布式的特点；



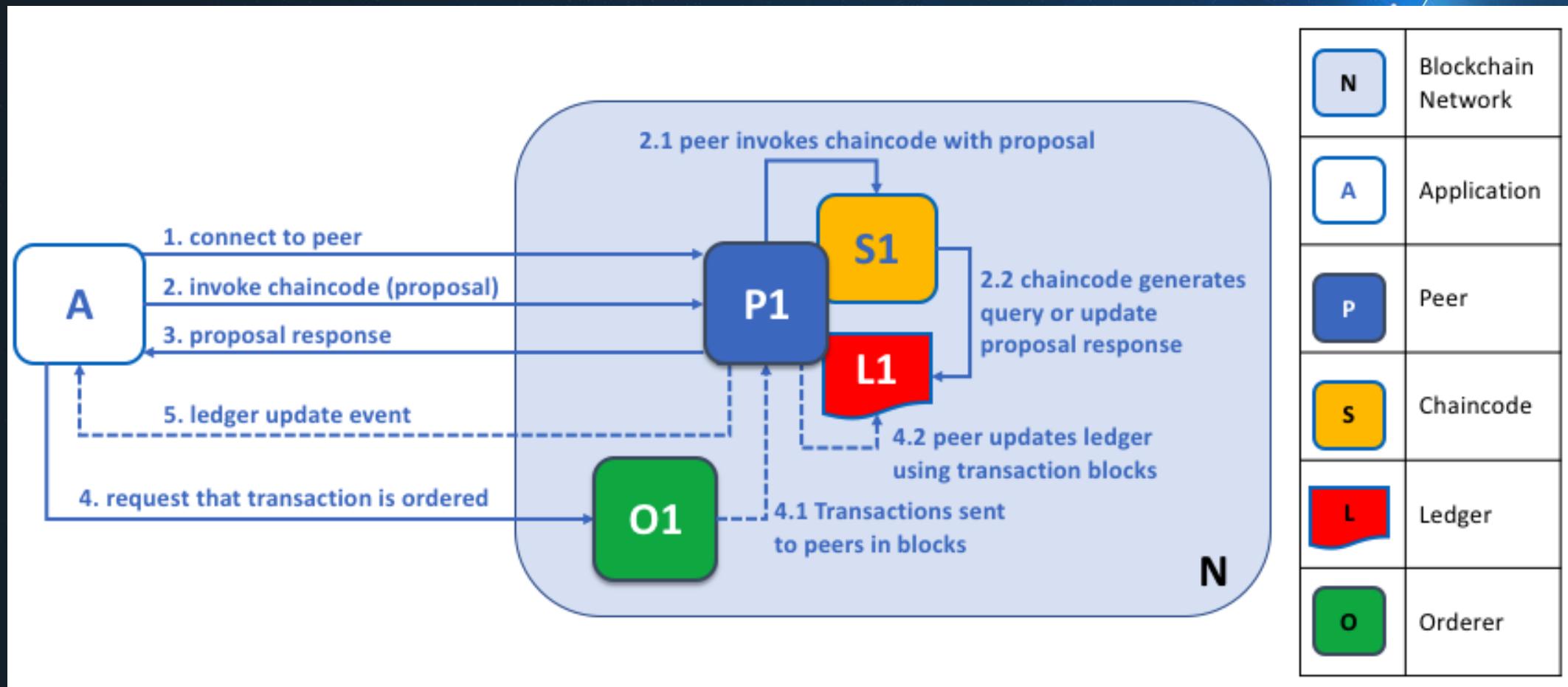
网络N由节点P1、P2和P3组成，它们各自维护自己的账本L1的实例。P1、P2和P3使用相同的链码S1访问其账本的副本。



节点P1维护账本L1和L2，其中L1由链码S1和S2访问，L2由S1和S3访问。可以发现S1可以同时访问L1和L2。

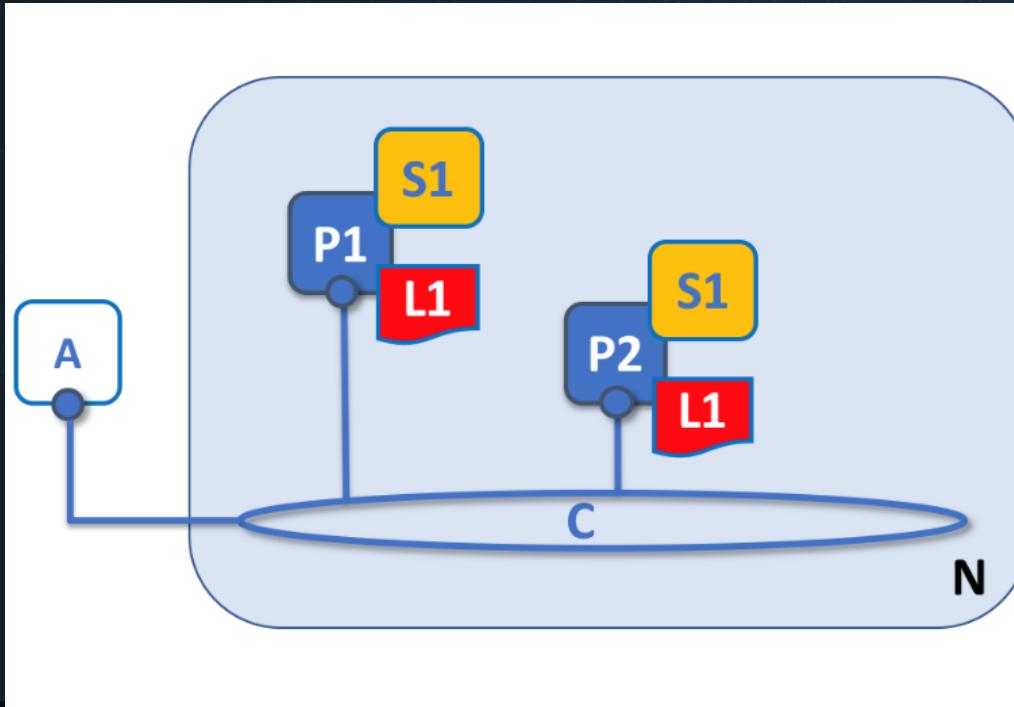
节点和应用程序

- 应用程序与节点交互



节点和通道

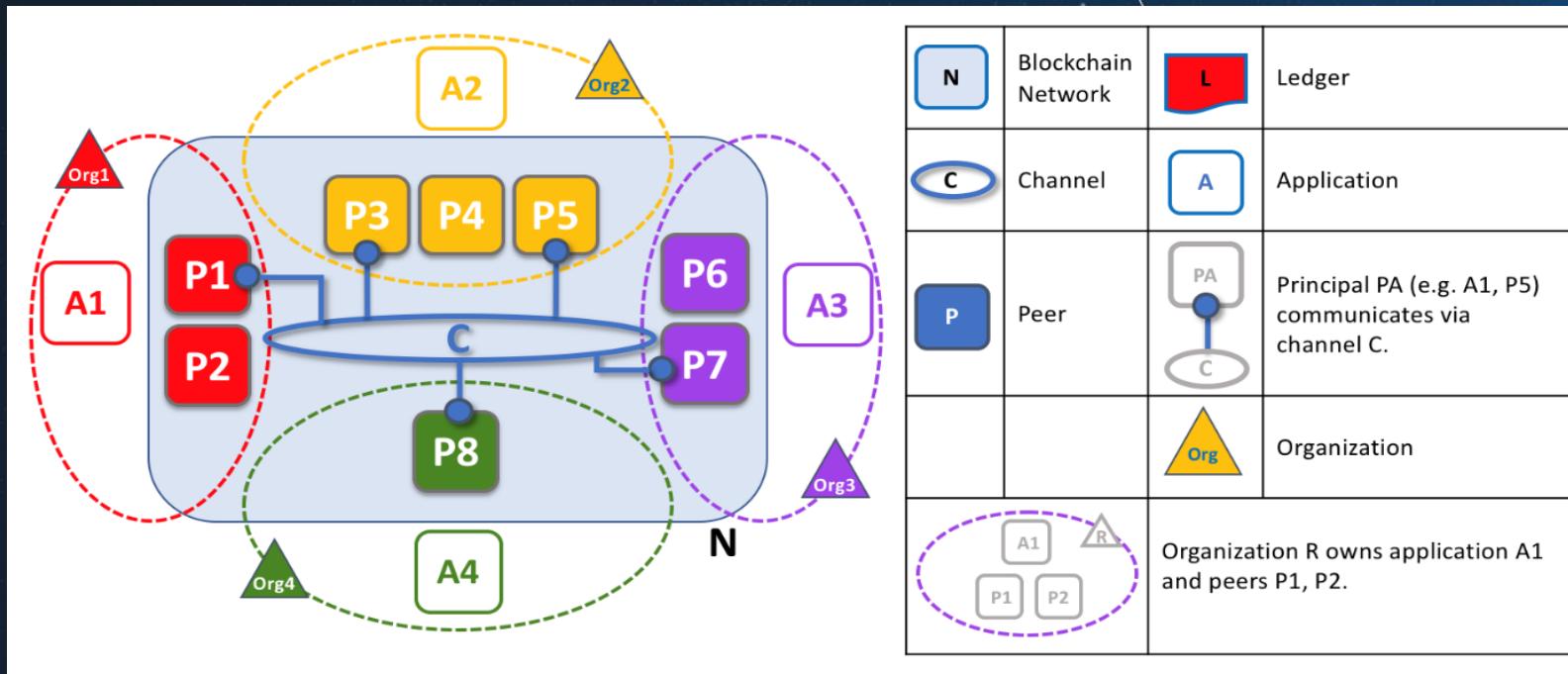
- 通道允许一组特定的节点和应用程序在区块链网络中彼此通信。例如：应用程序A可以使用通道C直接与节点P1和P2通信。您可以将通道视为特定应用程序和节点之间通信的路径。
- 将通道视为一个由物理节点集合构成的逻辑结构，节点提供了对通道的访问和管理的控制点。



N	Blockchain Network	L	Ledger
C	Channel	A	Application
P	Peer	PA C	Principal PA (e.g. A, P1) communicates via channel C.
S	Chaincode		

节点和组织

- Fabric网络由多个组织而不是单个组织管理。节点是构建这种分布式网络的核心，因为它们属于这些组织，并且是这些组织的网络连接点。

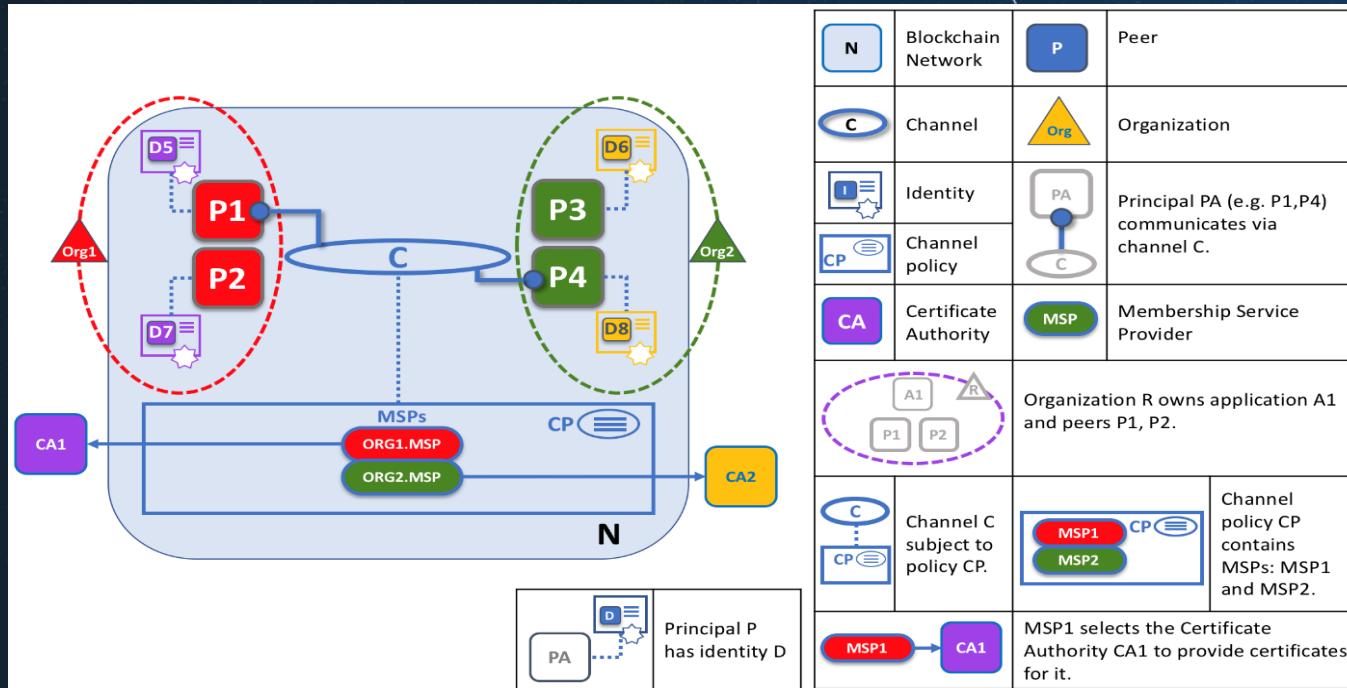


四个组织贡献了八个节点来组成一个网络。

通道C连接网络N-P1、P3、P5、P7和P8中的五个节点。

节点和身份

- 网络中的每个节点都由其所属组织的管理员分配一个数字证书，当节点连接到通道时，其数字证书通过通道MSP标识其所属组织。。



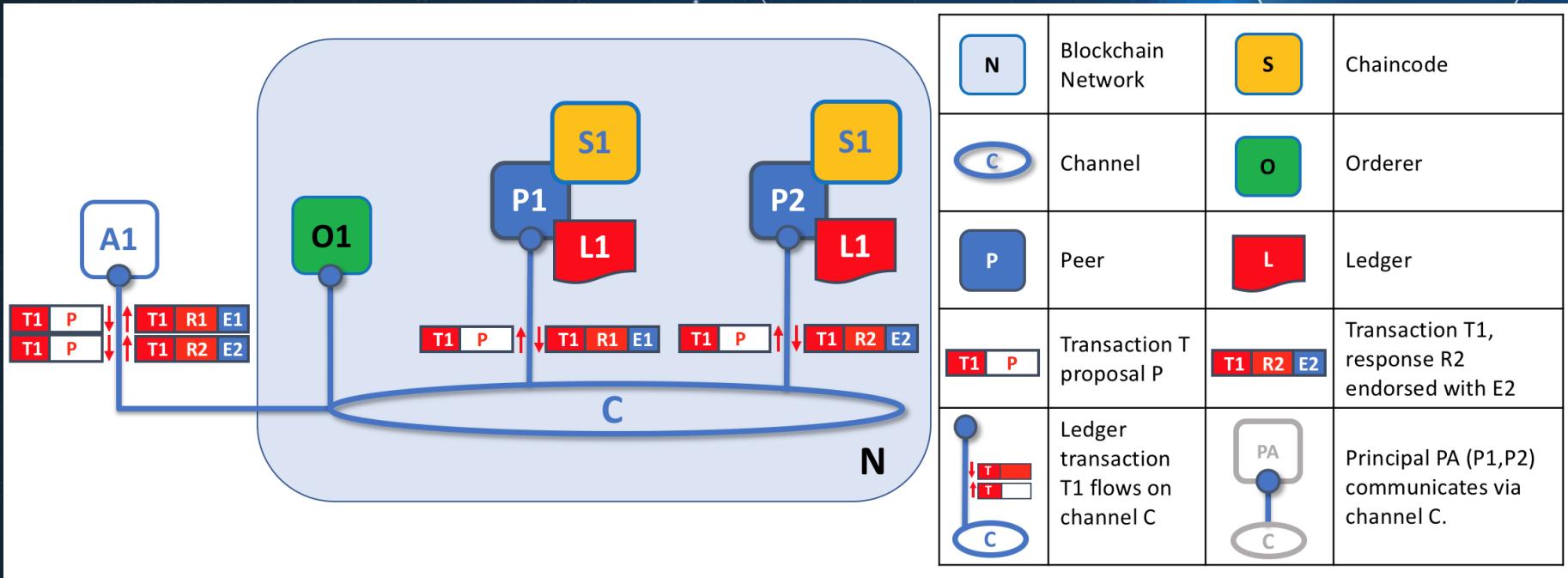
P1和P2具有由CA1发出的标识。通道C根据其通道配置中的策略确定来自CA1的标识应使用Org1.MSP与Org1关联。类似地，P3和P4被ORG2.MSP标识为ORG2的一部分

节点、应用程序和排序节点

接下来，我们通过交易的三个阶段和之前所讲述的模块，详细介绍一次交易的流程

- 第一阶段：提案阶段

3. 提案阶段：组织 O1 的 Peer P1 生成交易 T1，执行链码 S1，生成对于交易 T1 的响应 R2，提供背书 E2。这是因为背书策略定义了该交易提案被认可的前提是不需要对交易响应的认购添加自己的数字签名的方式提供背书，并且使用它的私钥为整个交易提案提供签名。背书 E1 相当于“在账本 L1 上的交易 T1 的响应 R1 已经被 Org1 的 Peer 节点 P1 同意了！”



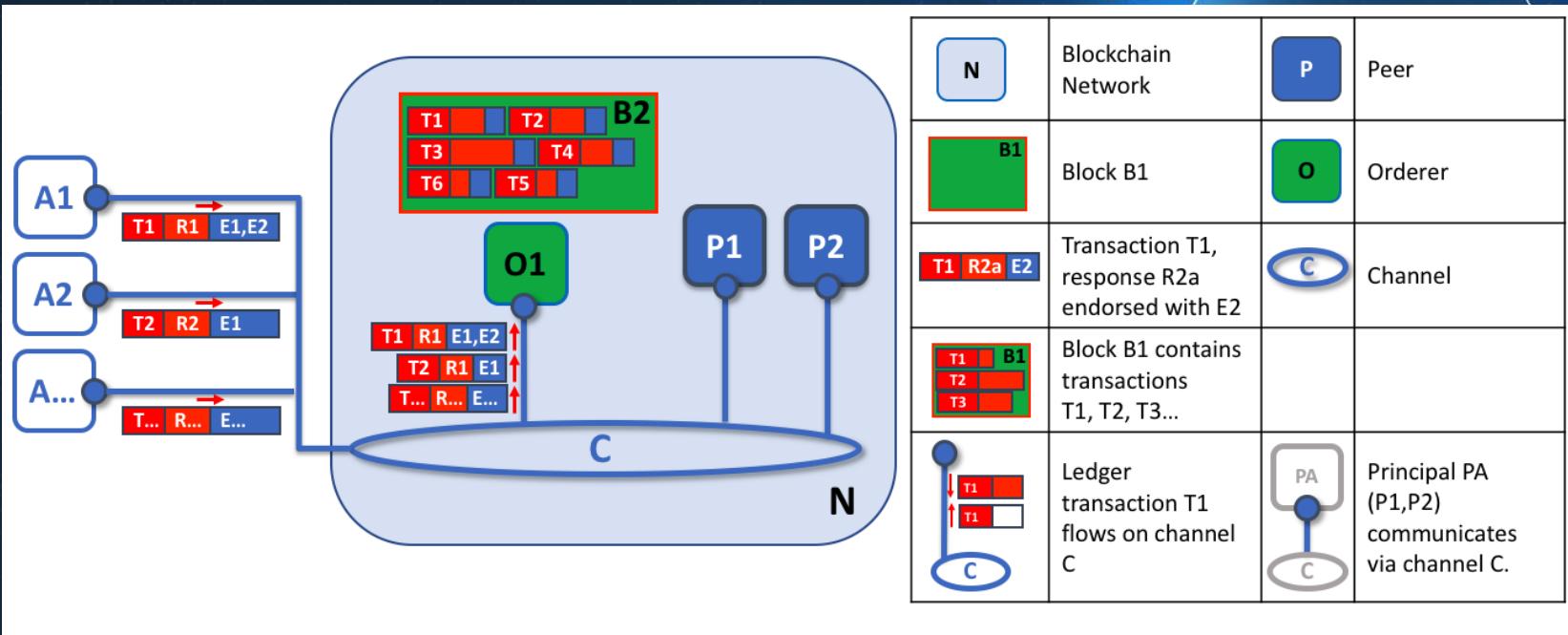
节点、应用程序和排序节点

- 第二阶段：排序和将交易打包到区块

- 应用程序 A1 向排序节点 O1 发送由 E1 和 E2 背书的交易 T1；
- 同时，应用程序 A2 将 E1 背书的交易 T2 发送给排序节点 O1；
- O1 将来自应用程序 A1 的交易 T1 和来自应用程序 A2 的交易 T2 以及来自网络中其他应用程序的交易打包到区块 B2 中；

一个区块中交易的顺序不一定与排序服务接收的顺序相同：O1接受的顺序可能是T1-T6，但在B2中交易的顺序是T1,T2,T3,T4,T6,T5，因为可能有多个排序节点；

重要的是，排序服务将交易放入严格的顺序中，并且 Peer 节点在验证和提交交易时将使用这个顺序。



在 Fabric 中，排序服务生成的区块是最终的，永远不会被重写或删除，确保了没有账本分叉；

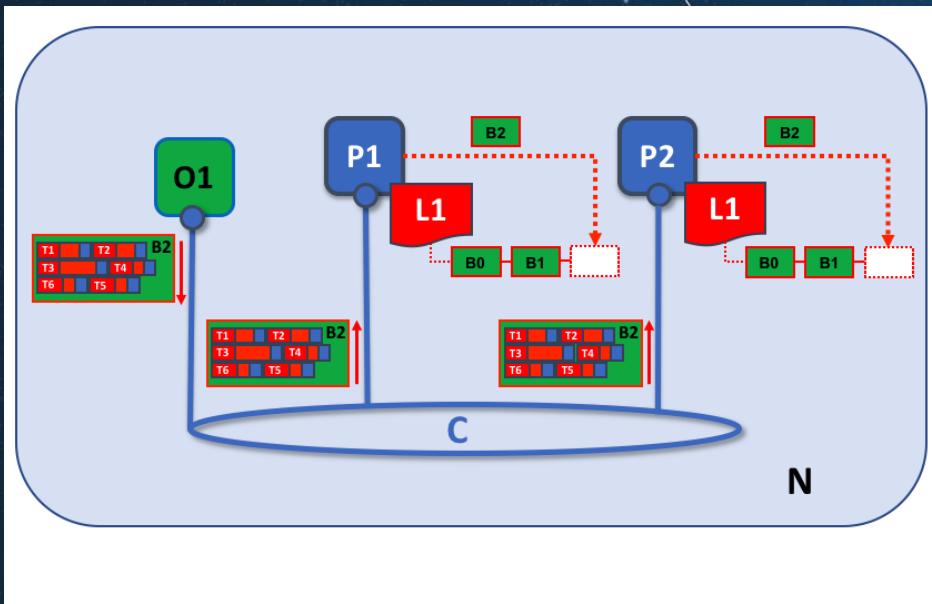
排序节点只做排序，不判断交易内容。

节点、应用程序和排序节点

• 第三阶段：验证和提交

3. 节点 O1 将区块 B2 发送给连接至它的所有 Peer 上的账本 L1 中添加一个新区块。同时，节点 P2 处理区块 B2，从而将一个新区块添加并存储到 P2 的账本中。所有的账本都需要连接到一个排序节点，Peer 节点可以用 gossip 协议将区块关联到其他节点。节点 P1 和 P2 上的账本 L1 就会保持一致的更新，并且每个节点都可以通知与之连接的应用程序交易已经处理。验证区块，以确保账本保持一致；

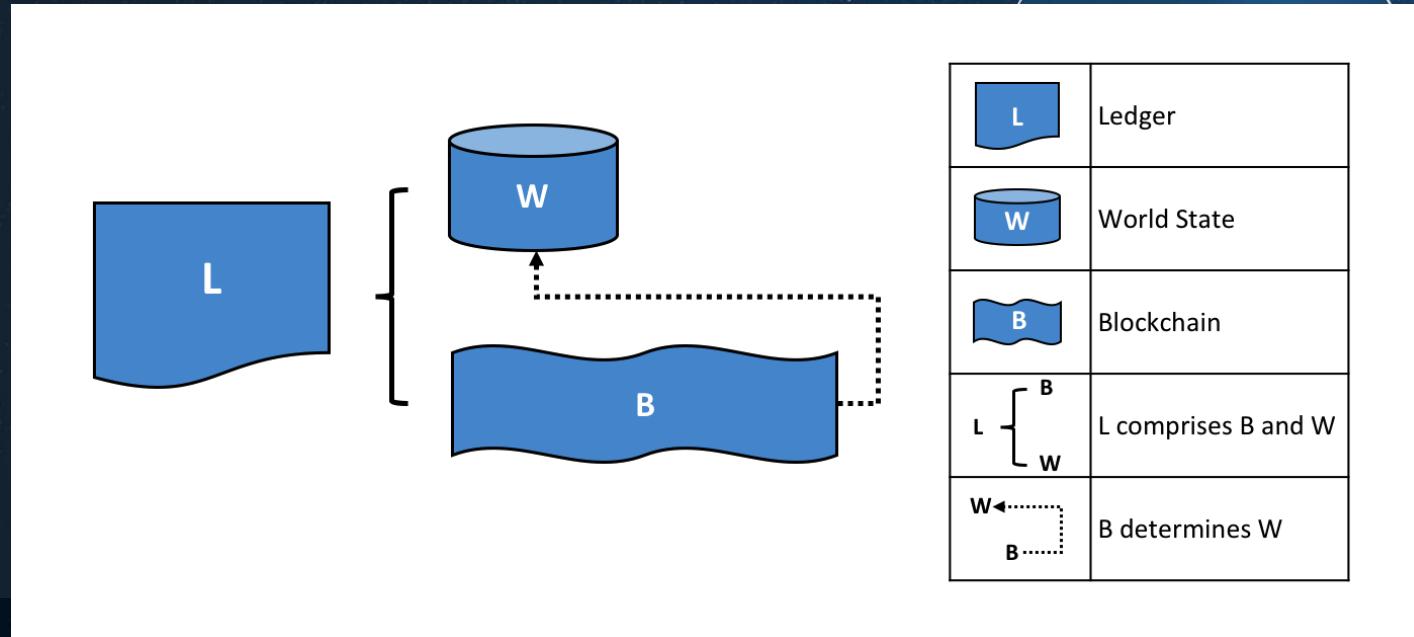
每个节点都验证区块中的每笔交易，确保得到了所需组织的背书和背书策略相匹配



N	Blockchain Network	P	Peer
C	Channel	O	Orderer
L	Ledger	B	Block B
L1	Ledger L1 has blockchain with blocks B0, B1	B1	Block B1 contains transactions T1, T2, T3...
	Block B1 flows on channel C	PA	Principal PA (P1, P2) communicates via channel C.

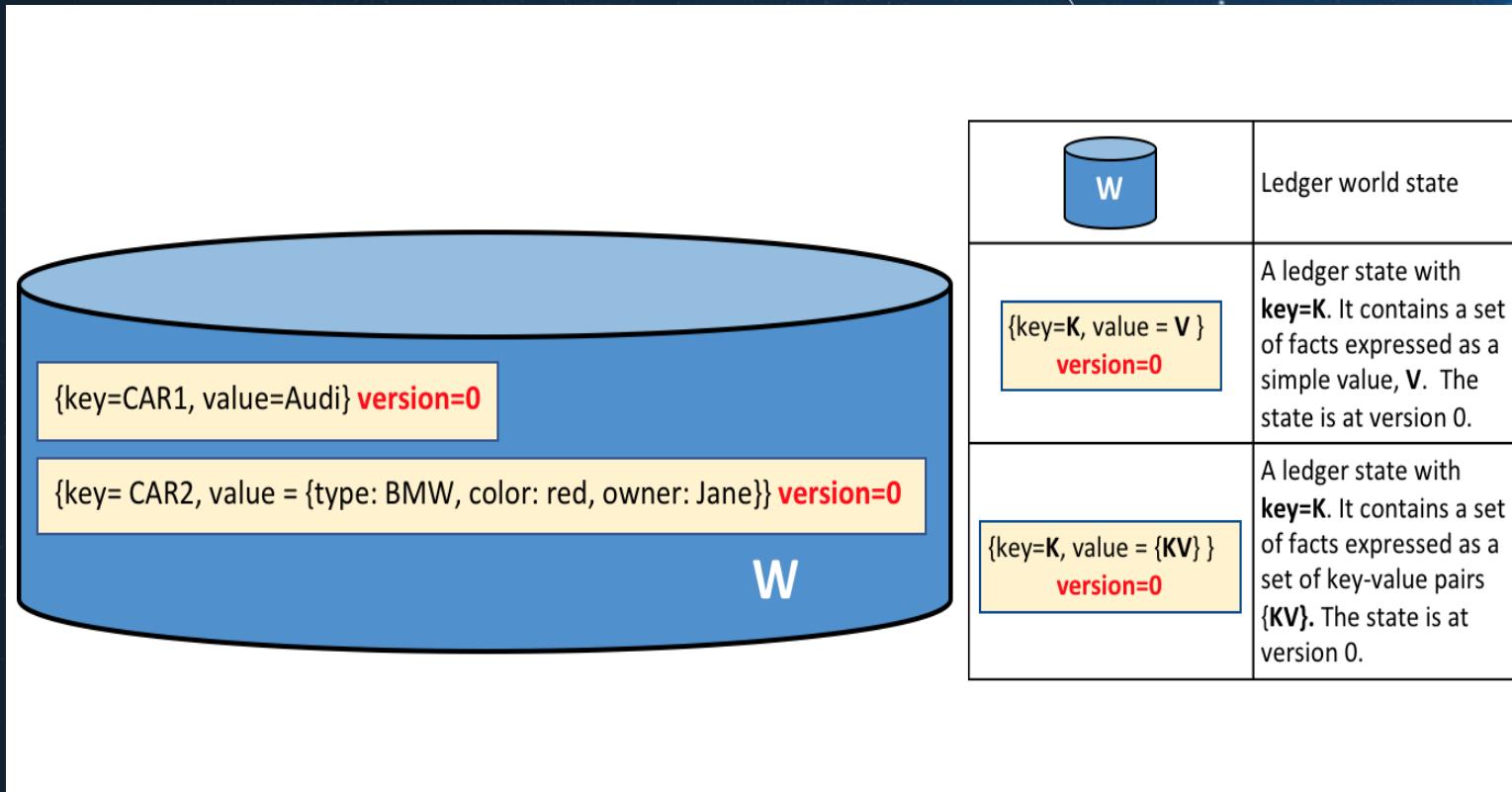
账本ledger

- Fabric中，分类帐由两个不同但相关的部分组成-**世界状态**和**区块链**；
- 区块链被构造为互连块的顺序**日志**，其中每个块包含一系列**交易**，每个交易代表对世界状态的查询或更新。
- 世界状态被实现为**数据库**，保存一组账本状态的当前值，账本状态表示为键值对。世界状态可以频繁更改，因为可以创建、更新和删除状态，世界状态中每次账本变更，都会有相应的版本号递增。
- 应用程序提交导致世界状态变化的交易，并提交到分类账区块链，但只有获得所需背书签名的交易才会导致对世界状态的更新，如果没有得到足够的背书人的签名，则不会导致世界状态的改变。



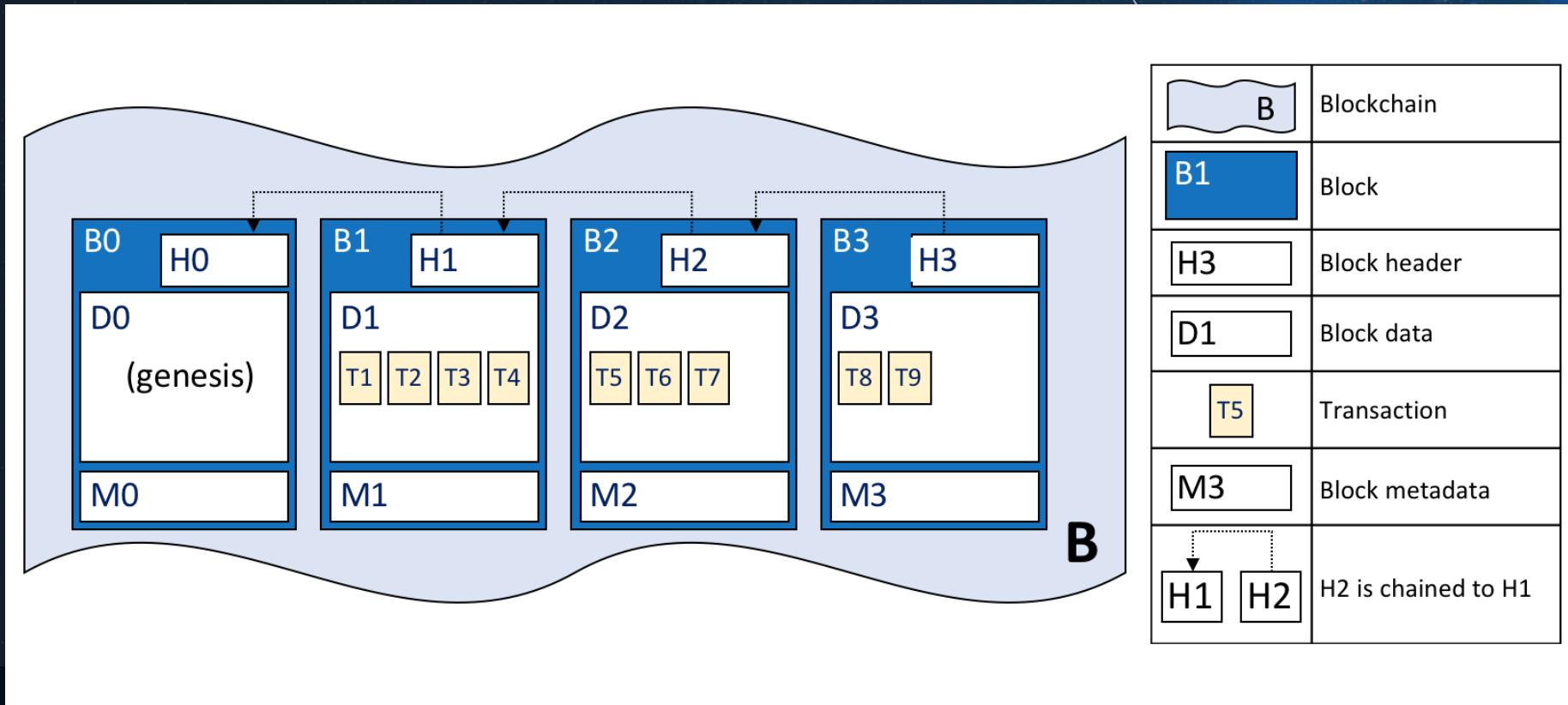
世界状态

- 世界状态将业务对象属性的当前值保存为唯一的账本状态。

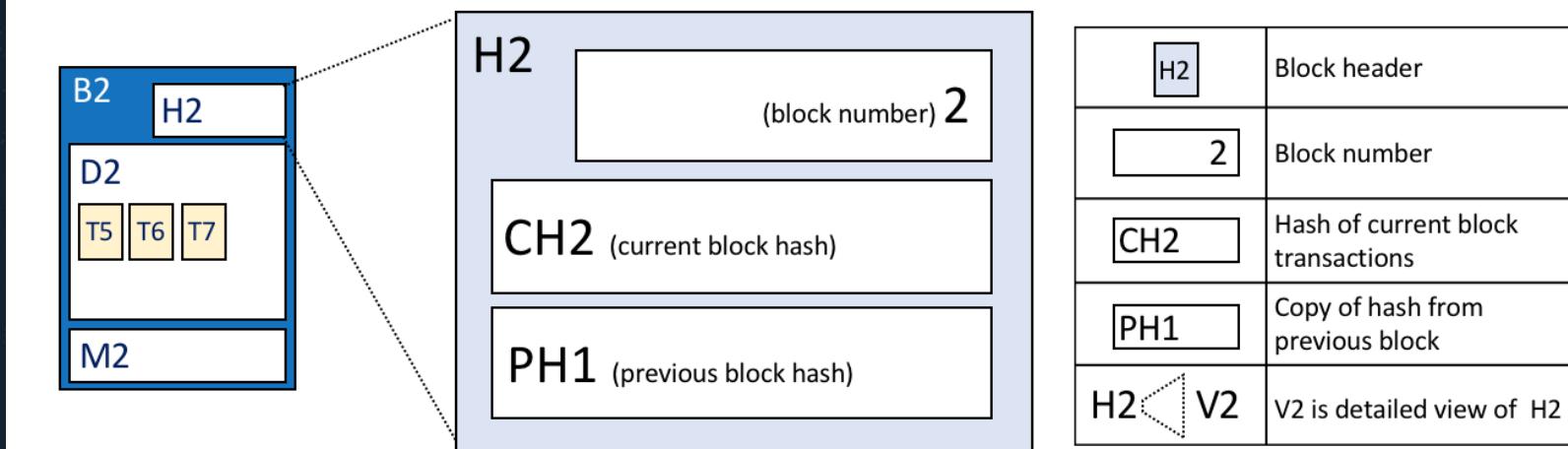


区块链

- 区块链则是世界状态中的业务对象如何达到其当前状态的历史记录，记录了每个账本状态的所有以前版本及其更改方式。
- B_0 是区块链中的第一个区块，即创世区块； B_n 具有块头 H_n ，其中包含 D_n 中所有事务的加密哈希以及 H_{n-1} 的哈希，并由此将区块前后相接，组成区块链。

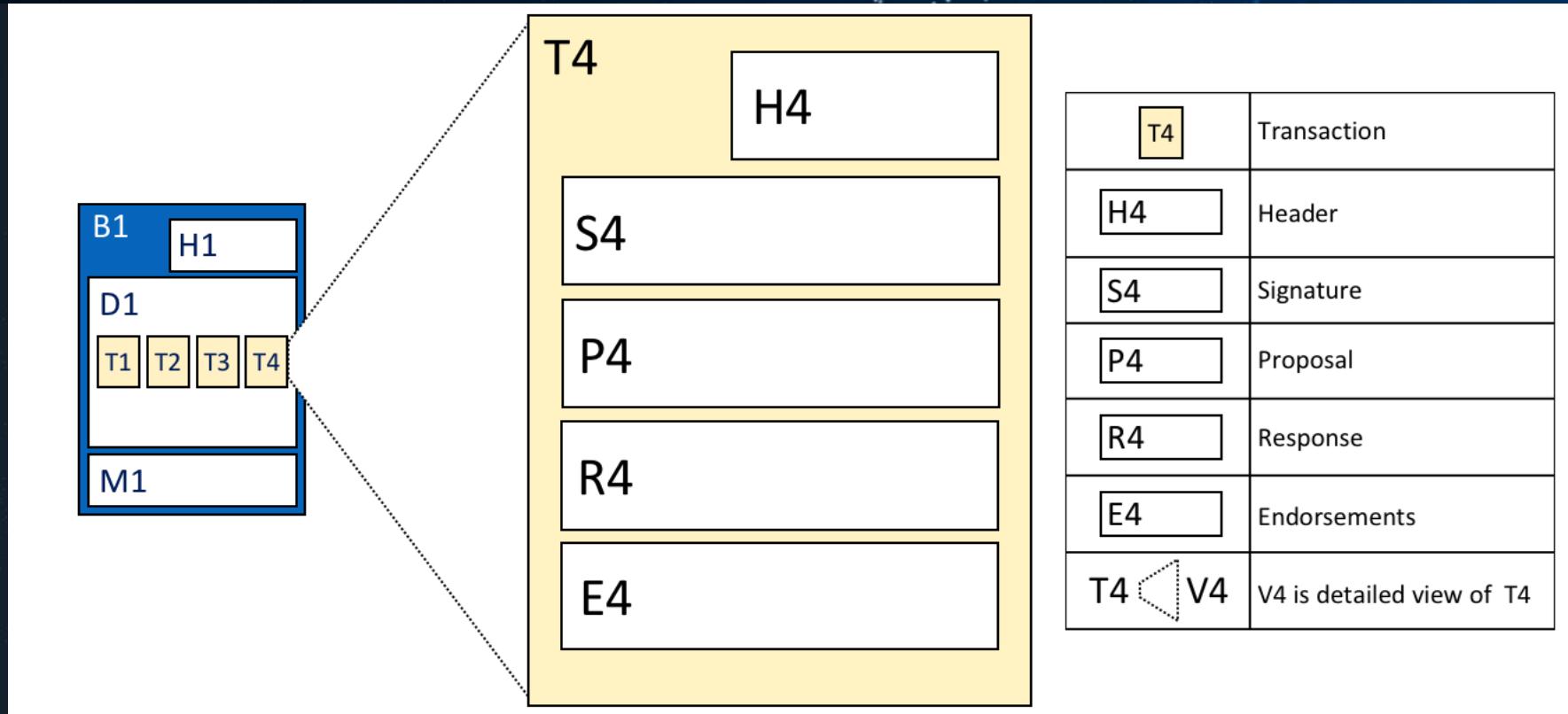


区块



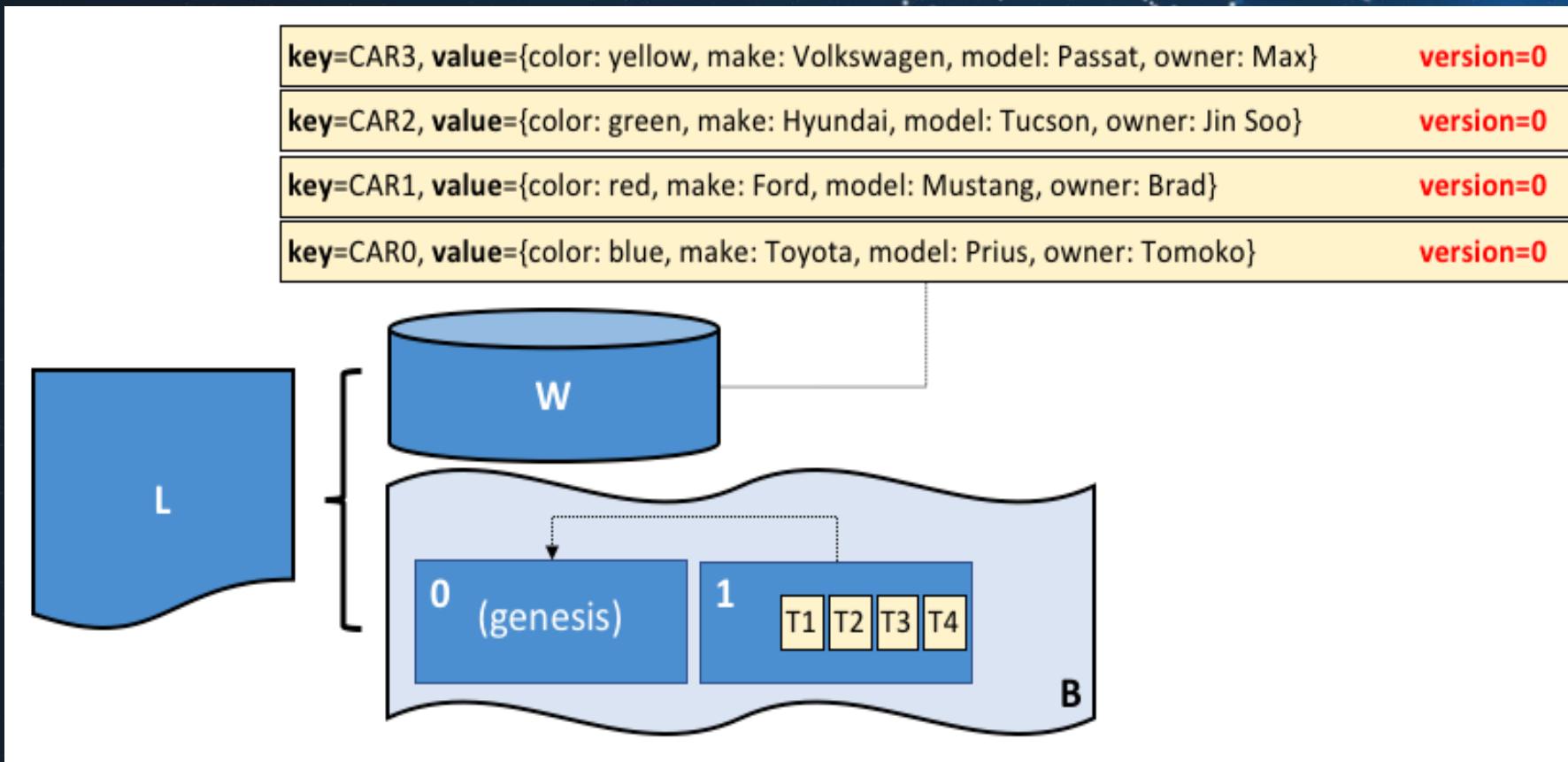
- 区块头：包含三个字段，区块编号、当前块哈希、上一个块头哈希。
- 区块数据：包含按顺序排列的交易列表，它是在排序服务创建块时写入的。
- 区块元数据：包含区块创建者的证书和签名，用于被网络节点验证区块，区块确认节点将每个交易的有效/无效指示符添加到位图中，该位图也位于区块元数据中，以及直到并包括该块的累积状态更新的哈希，以便检测状态分叉。与区块头和区块数据不同，此部分不是区块哈希计算的输入。

交易



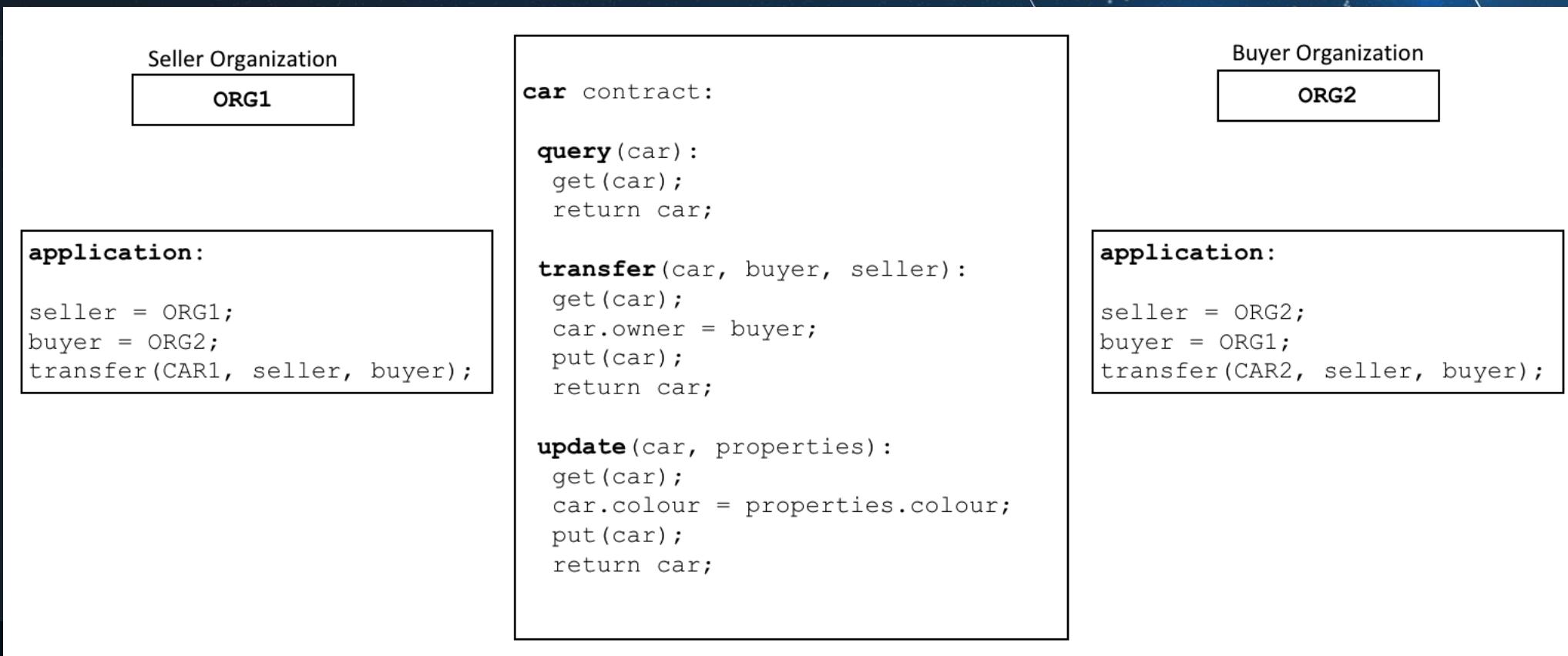
- 块B1的区块数据D1中的交易T4由交易头H4、交易签名S4、交易提案P4、交易响应R4和背书列表E4组成。
- 交易头保存有关交易的一些基本元数据，例如，相关链码的名称及其版本。

账本示例：fabcar



智能合约和链码

- 智能合约定义了被添加到账本中的新事实的可执行逻辑；
- 链码是智能合约的集合，管理者将智能合约打包用于部署；
- 类比于传统的http协议，一个Method和支持的Action相当于一个智能合约， Method的组合部署类似于链码；

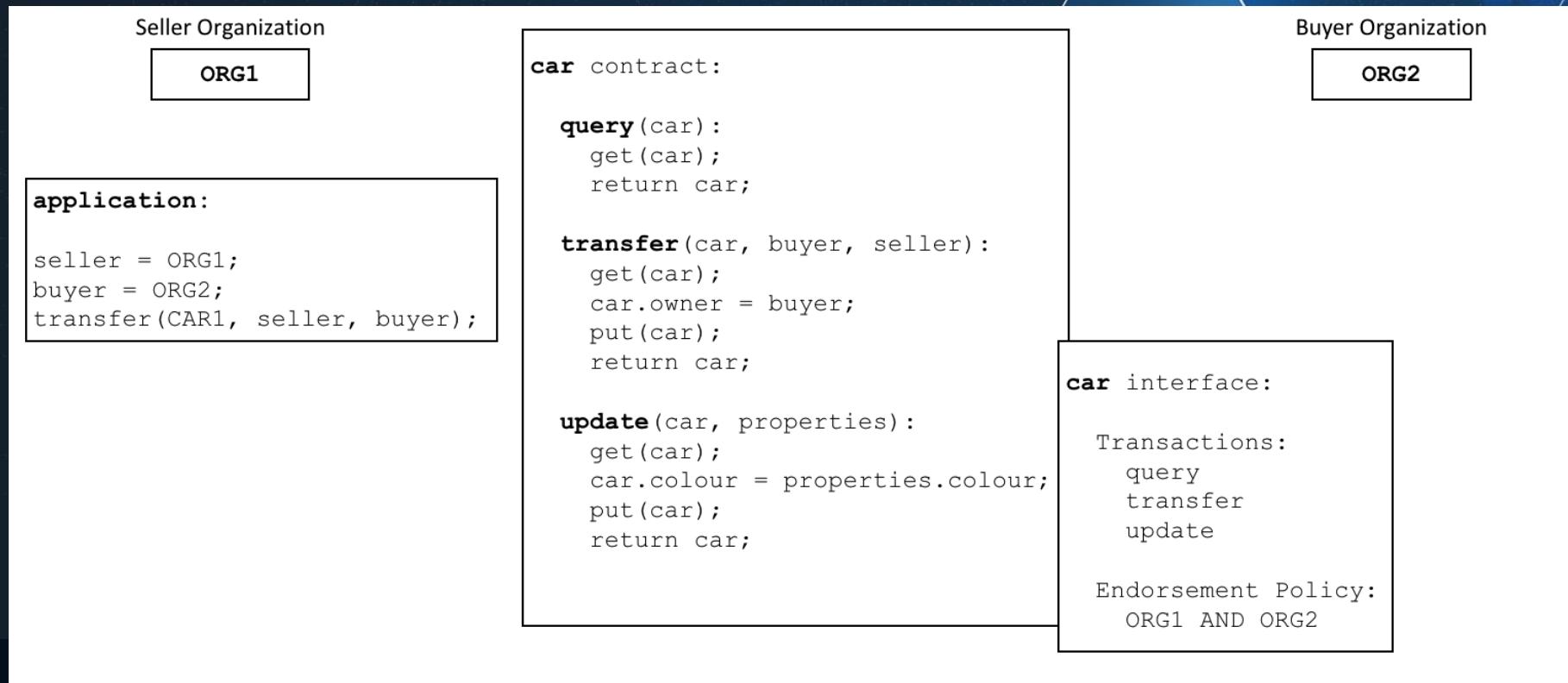


智能合约和链码

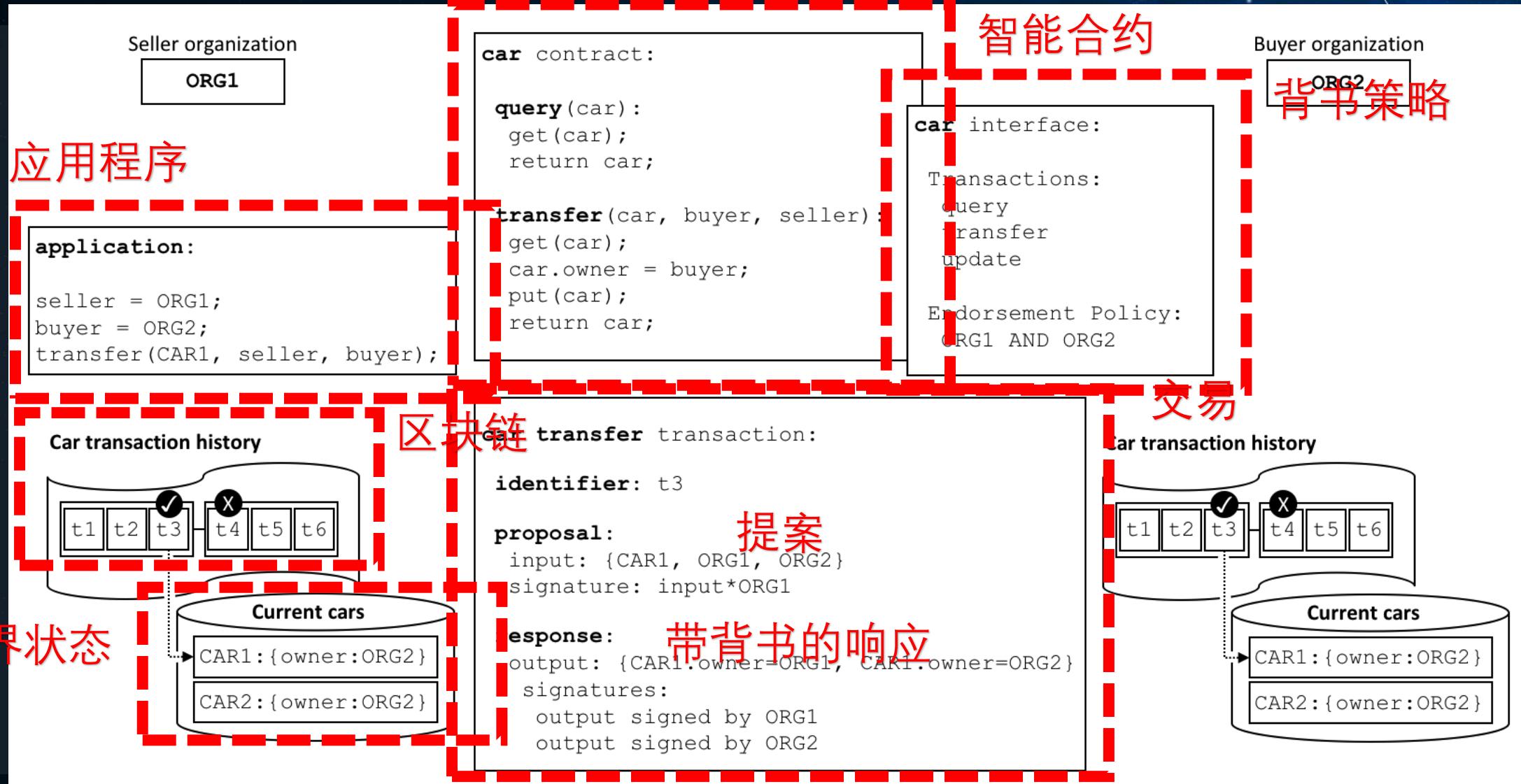
- 智能合约以编程方式访问账本两个不同的部分：
 - 区块链：记录所有交易的历史，且记录不可篡改
 - 世界状态：保存这些状态当前值的缓存，是一个经常需要用到的对象的当前值
- 智能合约主要在世界状态中将状态写入（put）、读取（get）和删除（delete），还可以查询不可篡改的区块链交易记录。
 - 读取（get）操作一般代表的是查询，目的是获取关于交易对象当前状态的信息。
 - 写入（put）操作通常生成一个新的业务对象或者对账本世界状态中现有的业务对象进行修改。
 - 删除（delete）操作代表的是将一个业务对象从账本的当前状态中移除，但不从账本的历史中移除。
- 开发人员的工作是将一个现有的业务流程用JavaScript、GOLANG 或 Java等编程语言来表示成一个智能合约。

链码和背书策略

- 每个链码都有一个背书策略与之相关联，该背书策略适用于此链码中定义的所有智能合约，指明了区块链网络中哪些组织必须对一个既定智能合约所生成的交易进行签名，以此来宣布该交易有效。
- 所有的交易，无论是有效的还是无效的，都会被添加到分布式账本中，但仅有效交易会更新世界状态。
- 背书策略是 Hyperledger Fabric 与以太坊（Ethereum）或比特币（Bitcoin）等其他区块链的区别所在。



有效交易



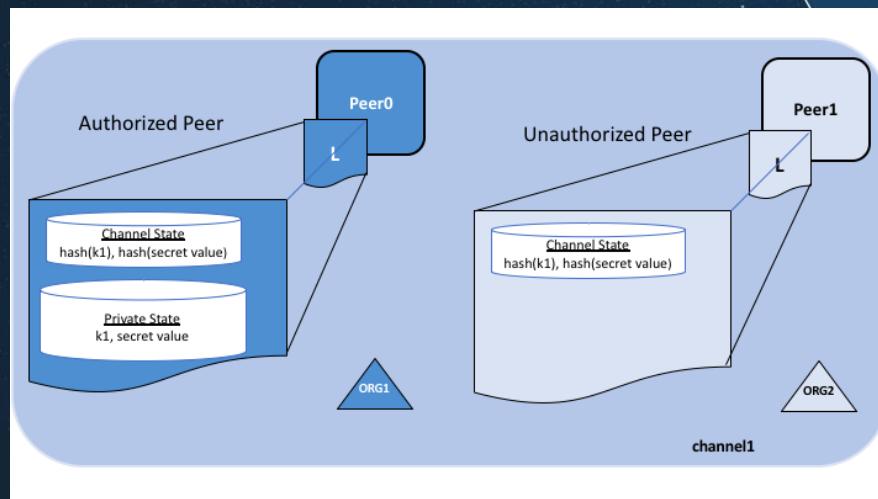
系统链码

Fabric链码分为系统链码和用户链码。系统链码用于协助节点完成包括系统配置、背书、校验等在内的工作。

- 配置系统链码 (Configuration System Chaincode, CSCC) : 负责处理记账节点上的各类配置信息。
- 生命周期系统链码 (Lifecycle System Chaincode, LSCC) : 负责用户链码的生命周期管理。
- 查询系统链码 (Query System Chaincode, QSCC) : 提供查询记账节点的账本数据。
- 背书管理系统链码 (Endorsement System Chaincode, ESCC) : 负责背书流程，支持对背书策略的管理，在对提交的交易提案的模拟执行结果进行签名后创建响应消息返回给客户端。
- 验证系统链码 (Validation System Chaincode, VSCC) : 负责在记账前对区块和交易进行验证。

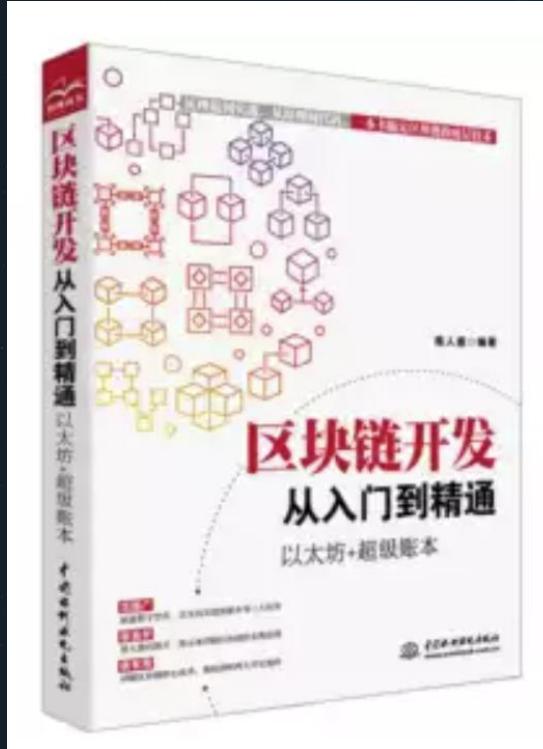
私有数据-Private Data

- 如果一个通道上的一个组织需要对该通道上的其他组织保持数据私有，则可以选择创建一个新通道，其中只包含需要访问数据的组织。但是，在每种情况下创建单独的通道会产生额外的管理开销(维护链码版本、策略、MSPs等)，并且在保留一部分数据私有的同时，希望所有通道参与者看到事务的情况是不允许的，从v1.2开始，Fabric提供了创建私有数据集合的功能，它允许在通道上定义的组织子集能够背书、提交或查询私有数据，而无需创建单独的通道。
- 下图显示了一个通道上被授权和未被授权拥有私有数据的节点的账本内容。



参考资料

- https://hyperledger-fabric.readthedocs.io/_/downloads/en/release-2.2/pdf/ Fabric官方文档



谢谢!