



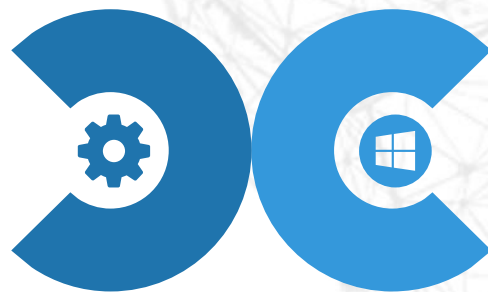
联盟链技术 Alliance-chain Technology

北京交通大学
计算机与信息技术学院
信息安全系

李超 (li.chao@bjtu.edu.cn)
段莉 (duanli@bjtu.edu.cn)

内容

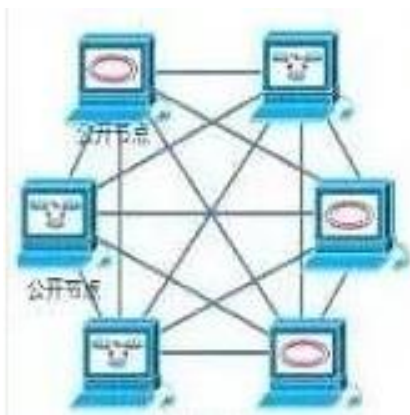
联盟链



Hyperledger Fabric

联盟链

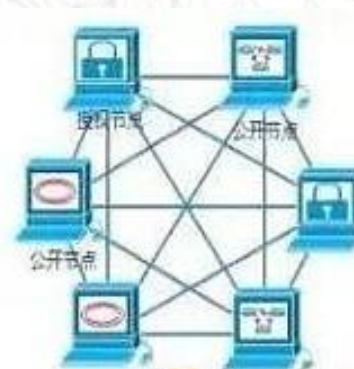
- 根据去中心化程度的不同，分化出3种不同应用场景下的区块链：



公有链



私有链



联盟链

- 公有链：全网公开，广大用户可参与
- 私有链：所有网络中的节点都掌握在一家机构手中
- 联盟链：用于多个机构之间，允许授权的节点加入网络，可根据权限查询或修改信息

联盟链的优缺点

- 相比于公有链，联盟链在效率和灵活性上更有优势
 - 交易成本更低，交易只需被几个受信的高算力节点验证就可以，无需全网确认。
 - 节点规模小，可以很好地连接，故障可以迅速通过人工干预来修复。
 - 可以使用确定型的共识算法并缩短区块生成时间，从而更快完成交易。
 - 读写权限可以控制，从而提供更好的隐私保护。
 - 联盟链的参与者可以更容易地达成一致来更新区块链的规则、还原交易、修改余额等。
- 相比于公有链，联盟链去中心化程度不够



联盟链的发展前景

- 联盟链受政策支持
 - 不依赖发币来激励用户参与，无监管问题
 - 不需要耗费大量电力资源挖矿
- 联盟链是“区块链+”的技术载体
 - 支持已有业务系统中部分数据的上链需求
 - 因联盟而产生的信任创造新的业务方向



联盟链的应用探索



--图片来自于清华大学徐恪教授课件

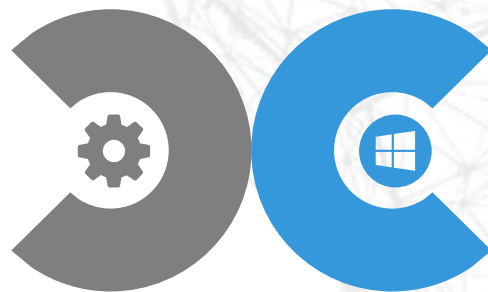
联盟链技术平台



- 当前最著名的的联盟链基础平台就是“超级账本（Hyperledger）”项目。

内容

联盟链



Hyperledger Fabric

超级账本



Hyperledger 是 **Linux** 基金会主持的一个开源项目，启动于 **2015** 年，核心目标是建立开放的、标准化的、企业级的、能支持商业交易的分布式账本的框架与基础代码

<https://github.com/hyperledger/fabric>

超级账本



- Hyperledger具有高度模块化和可配置的体系结构，可支持银行、金融、保险、医疗等广泛的行业用例。
- 支持以通用编程语言（如Go/Java/Node.js）而非受约束的领域特定语言（DSL）编写智能合约。
- 支持可插拔的共识协议以适应特定的信任模型，节点的授权加入方便网络治理，且无需发行加密货币来激励记账。

目前，超过 **100** 家全球知名企业和机构（大部分均为各自行业的领导者）宣布加入 **Hyperledger** 项目，包括 **30** 多家中国本土企业

艾亿新融旗下的艾亿数融科技公司（**2016.05.19**）、**Onchain**（**2016.06.22**）、比邻共赢（**Belink**）信息技术有限公司（**2016.06.22**）、**BitSE**（**2016.06.22**）、布比（**2016.07.27**）、三一重工（**2016.08.30**）、万达金融（**2016.09.08**）、华为（**2016.10.24**）等

超级账本

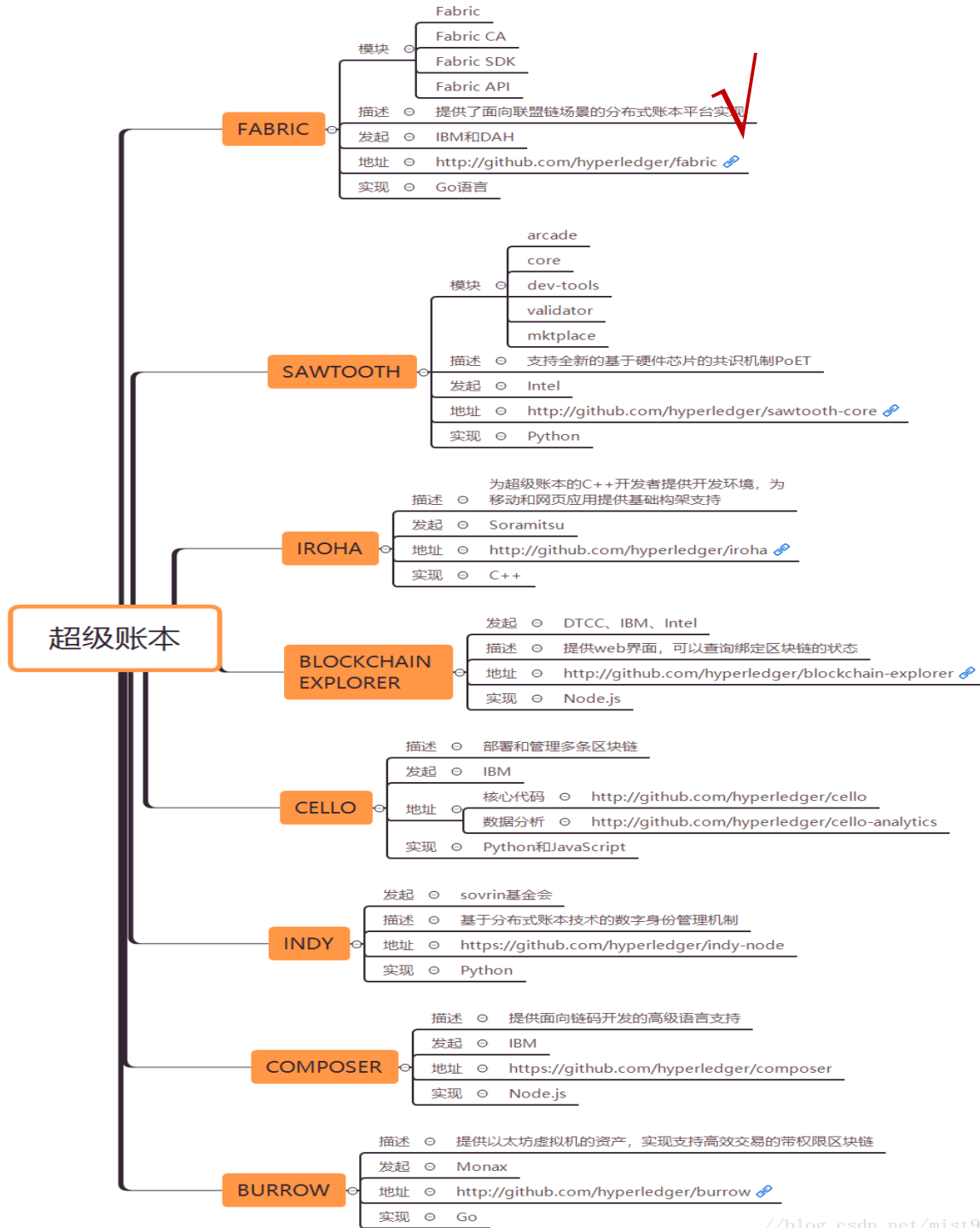


Premier Member

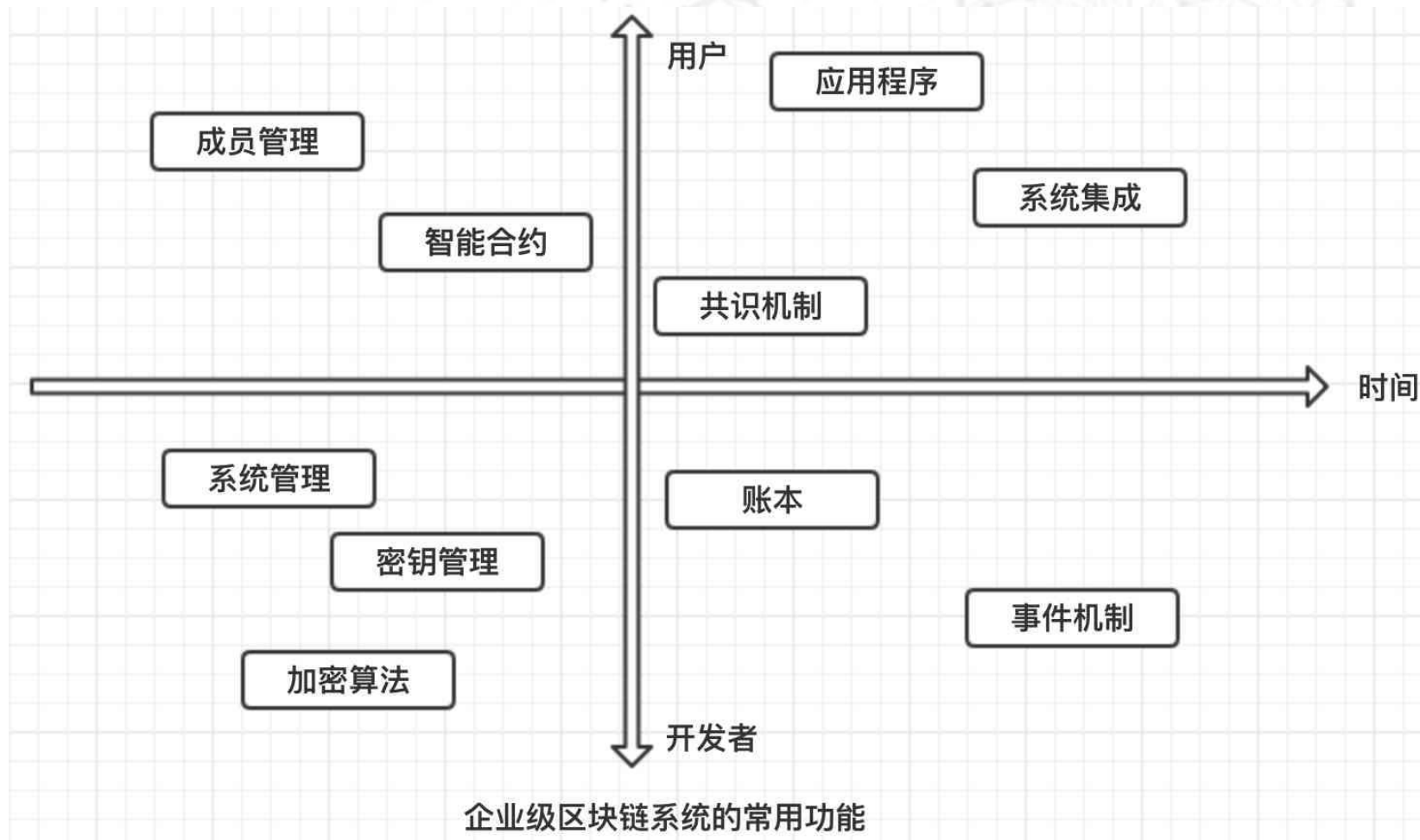
General Member

超级账本的组成:

- 作为一个联合项目，超级账本由面向不同目的和场景的子项目组成；
- 目前主要包含8大顶级项目，所有项目都遵守Apache V2许可



超级账本系统架构



Hyperledger Fabric 1.0

- 在通用的区块链技术当中，**Hyperledger Fabric 1.0** 是一个目前发展不错的技术，其设计的目标是利用一些成熟的技术**实现分布式账本技术**（Distributed Ledger Technology, DLT）。
- 超级账本其采用了**模块化的**架构设计，**复用通用的功能模块和接口**。
模块化的方法有以下好处：**可扩展性、高灵活性**；减少升级和修改过程中的影响。

Hyperledger Fabric 1.0

● 模块插件化

- 很多功能模块都实现了可插拔，比如CA模块、ESCC、VSCC、BCCSP、共识算法以及状态数据库存储等
- 系统提供了通用的接口以及默认的实现方式

● 容器技术

- 节点和链码都使用了容器技术，默认运行在安全的容器中
- 外部系统或应用程序要想操作链码，必须通过背书节点提供的接口转发给链码进行执行

● 可扩展性

- 对 Peer 节点的角色进行了拆分

● 高安全性

- 提供的区块链网络是授权访问网络
- 多链和多通道的设计，有利于实现数据隔离非常有利，提供了应用程序和链码之间的安全通道，完善了隐私保护

超级账本架构设计

超级账本包括三大组件：**区块链**（Blockchain）、**链码**（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**，并**对外提供接口**，外部通过调用链码接口来改变世界观。世界观是一个键值数据库，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。

Fabric逻辑架构

□ 身份管理

- 提供准入机制

□ 账本管理

- 授权的用户是可以查询账本数据



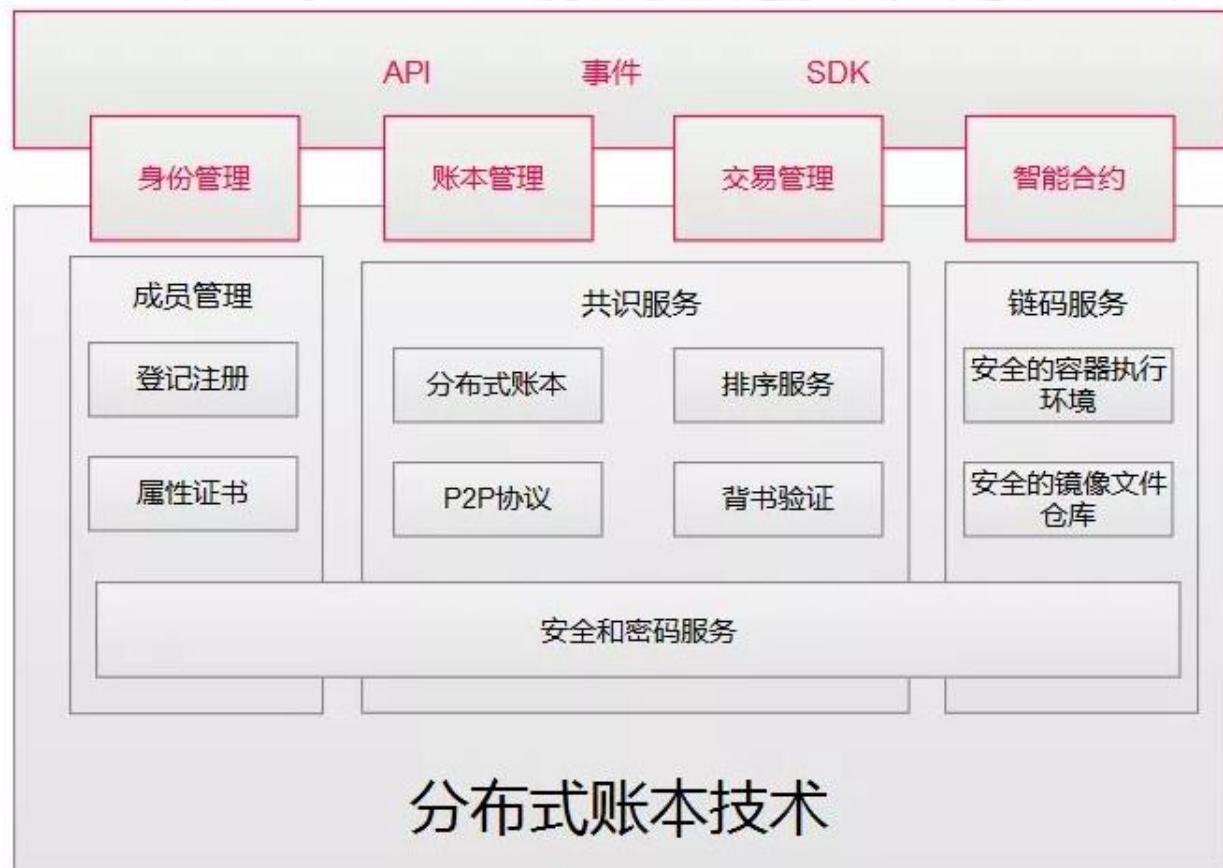
Fabric逻辑架构

交易管理

- 负责管理用户标识、隐私、以及网络的保密性和可审计性

智能合约

- 实现可编译的交易账本通过链码执行提交的交易



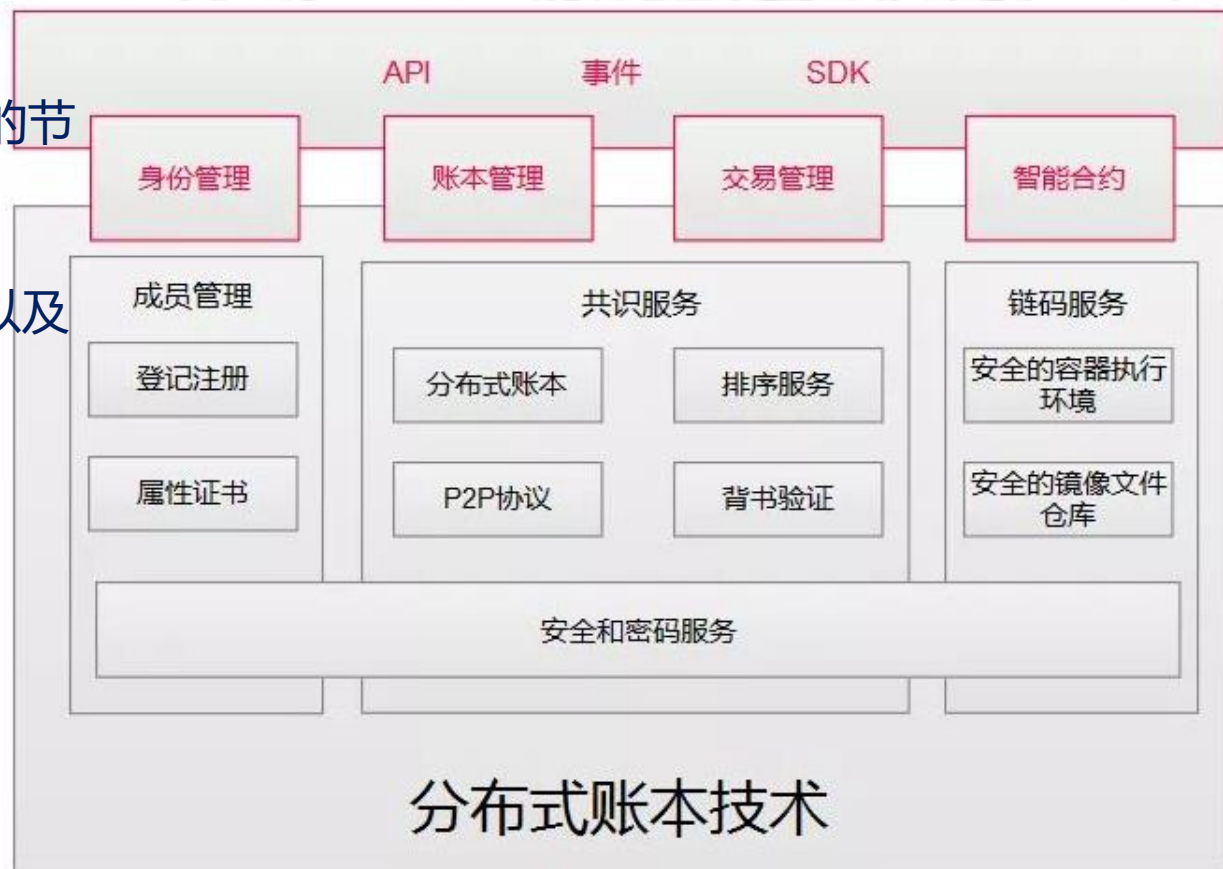
Fabric逻辑架构

MemberShip-成员服务

- 基于PKI，平台可以对接入的节点和客户端的能力进行限制。
- 负责管理用户标识、隐私、以及网络的保密性和可审计性

共识服务

- CAP原理：一致性 (Consistency)，可用性 (Available)，分区容错性 (Tolerance of network Partition)



Fabric逻辑架构

链码服务

- 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。

安全和密码服务

- 提供加密服务



超级账本Fabric项目

业务层

交易

区块

链码

通道

链结构

账本

业务开发人员

共识机制

背书

排序

验证

权限管理

成员服务提供者 MSP

组织

联盟

身份证书

组织管理人员

网络层

节点

排序者

客户端

CA

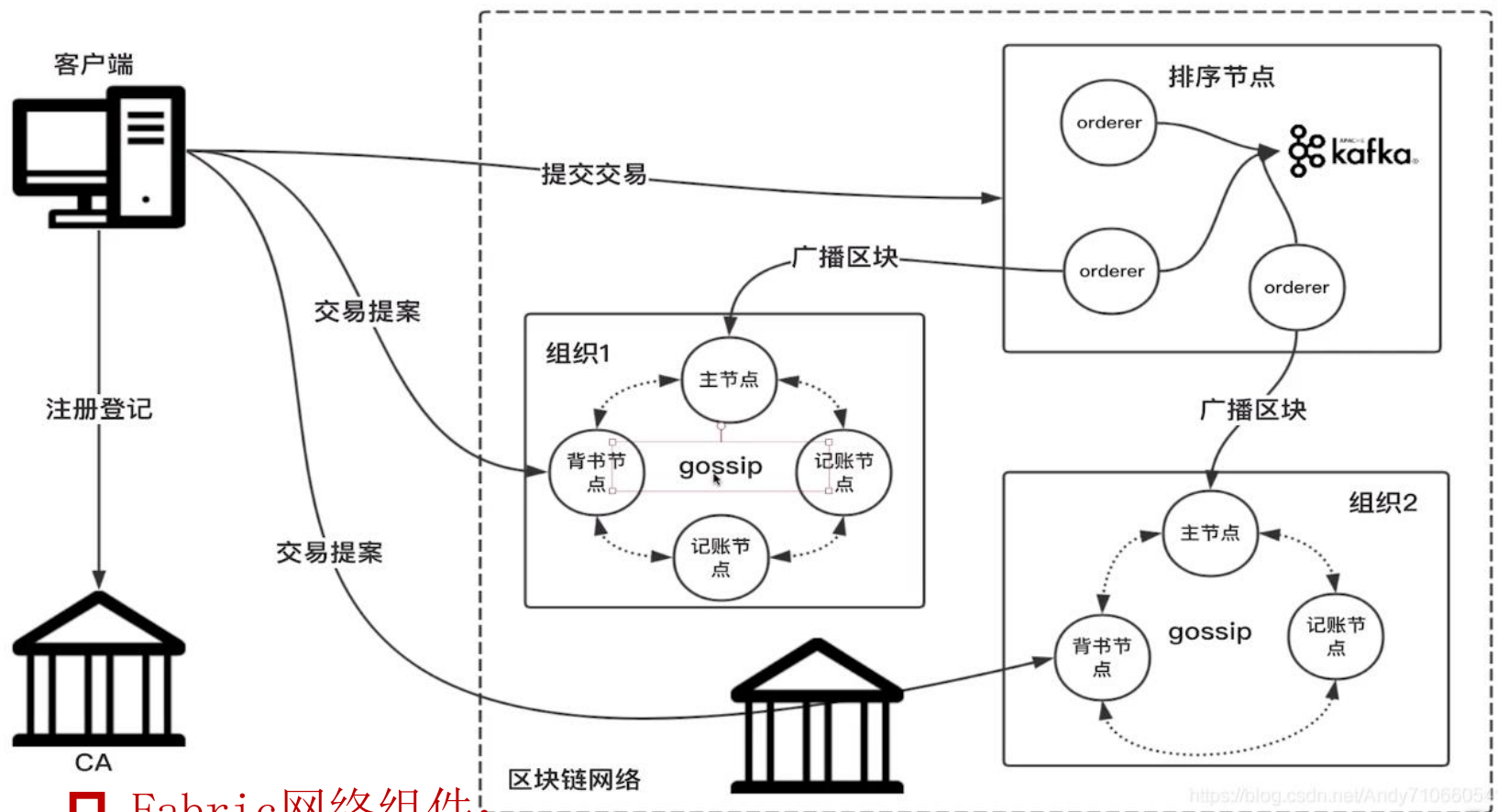
Gossip

系统管理人员

超级账本Fabric面向不同的开发人员提供了不同层面的功能，自下而上可以分为三层：

- **网络层**：面向系统管理人员。实现P2P网络，提供底层构建区块链网络的基本能力，包括代表不同角色的节点和服务；
- **共识机制和权限管理**：面向联盟和组织的管理人员。基于网络层的连通，实现共识机制和权限管理，提供分布式账本的基础；
- **业务层**：面向业务应用开发人员。基于分布式账本，支持链码、交易等跟业务相关的功能模块，提供更高一层的应用开发支持。

Fabric网络拓扑图



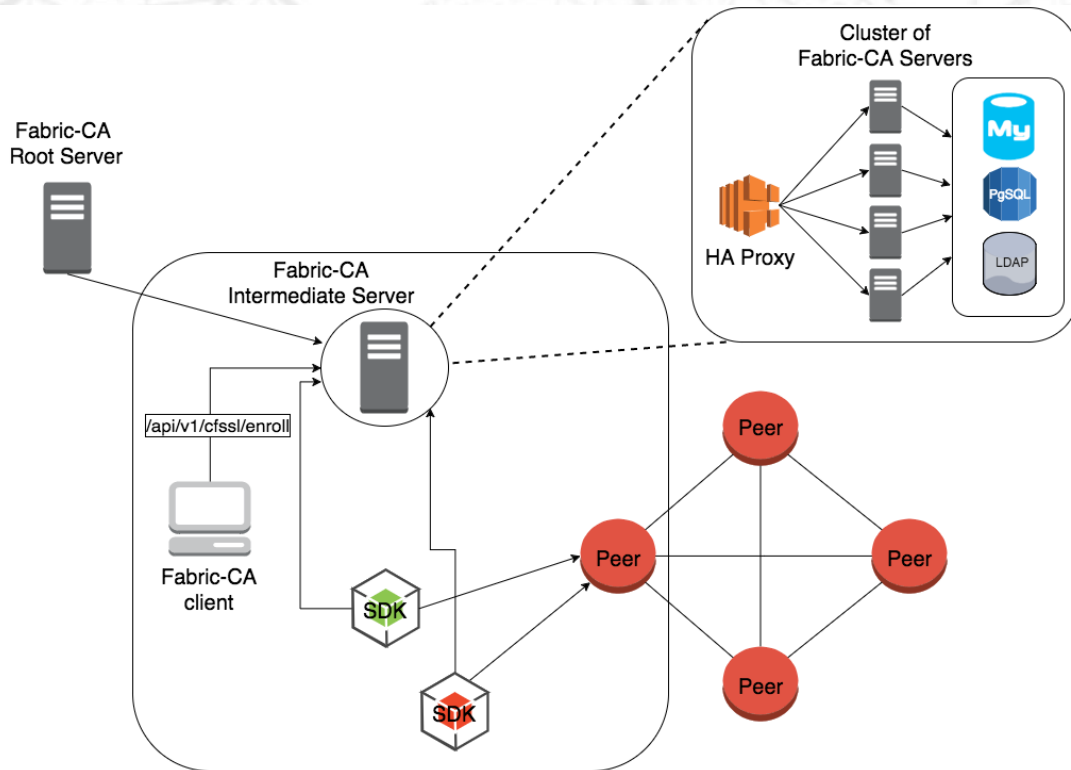
❑ Fabric网络组件:

- 账本
- Peer节点
- 通道
- 智能合约
- 排序节点
- Fabric证书颁发机构

Fabric CA

CA主要功能

- ✓ 身份注册，或者将连接到LDAP作为用户注册；
- ✓ 颁发登录证书(ECerts)；
- ✓ 颁发交易证书(TCerts)，
保证链上交易的匿名性与
不可连接性；
- ✓ 证书续期与撤销。



Peer节点

- 对等节点（简称**Peer**）组件是**Fabric**区块链网络的基本元素，它托管着账本和智能合约，且可以同时托管多个账本和多个智能合约。
- 区块链网络拥有多个组织下的多个对等节点，对等方节点具有通过特定**CA**颁发的数字证书分配的身份。

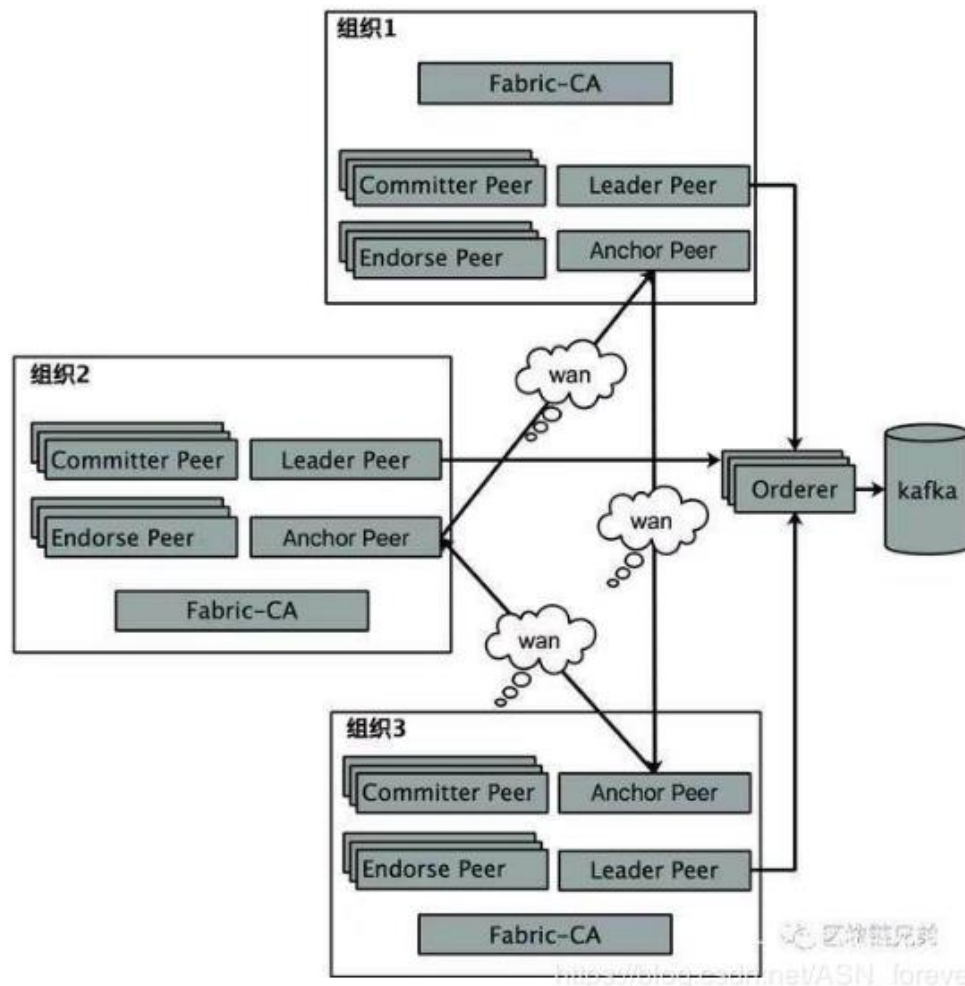
Peer节点

➤ Endorser Peer（背书节点）：和Contract绑定，供客户端调用，完成对交易提案的背书（目前主要是签名）处理。

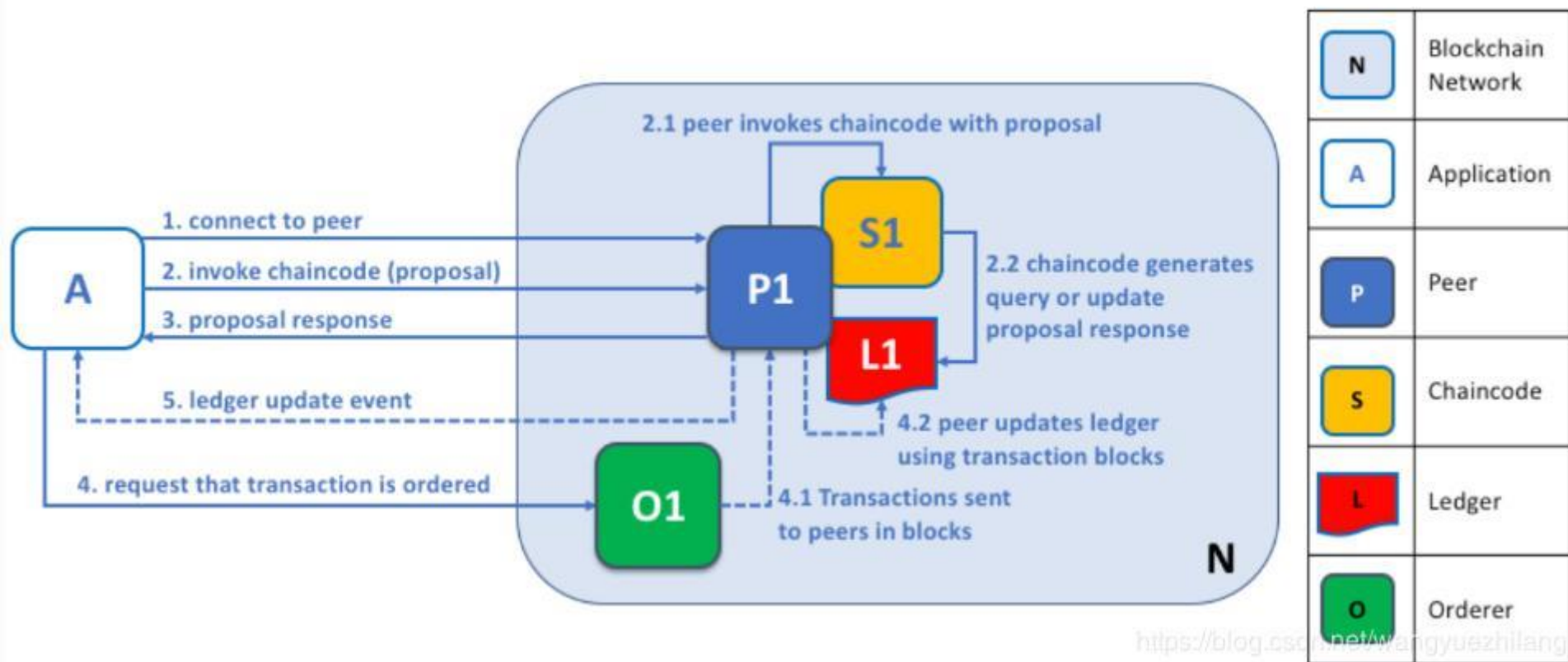
❑ Leader Peer（主节点）：主节点连接到排序服务，负责把接受到的批量区块转发给其他节点。

➤ Committer Peer（记账节点）：负责维护区块链和账本结构（包括状态DB、历史DB、索引DB等）；

❑ Anchor Peer（锚节点）：在一个channel上可以被所有其他peer发现的peer



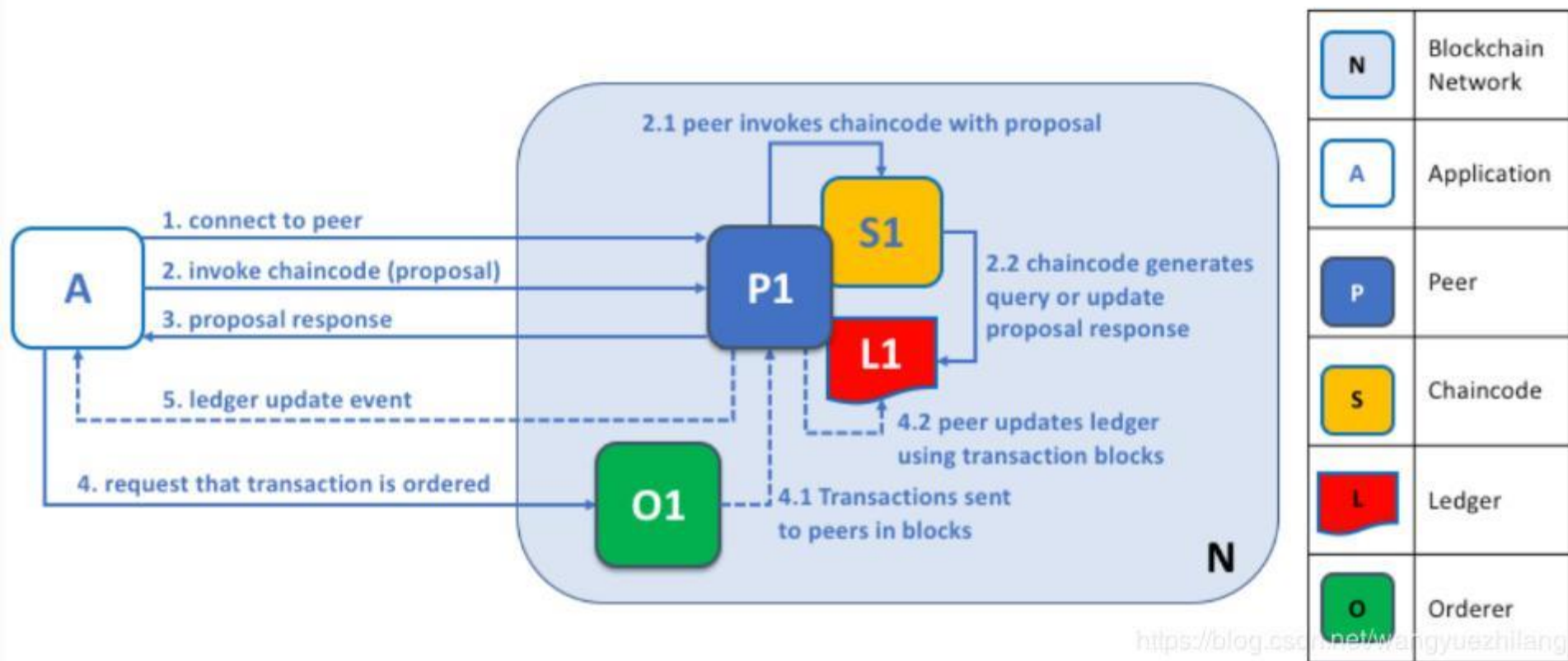
Peer节点



<https://blog.csdn.net/wangyuezhilang>

排序节点

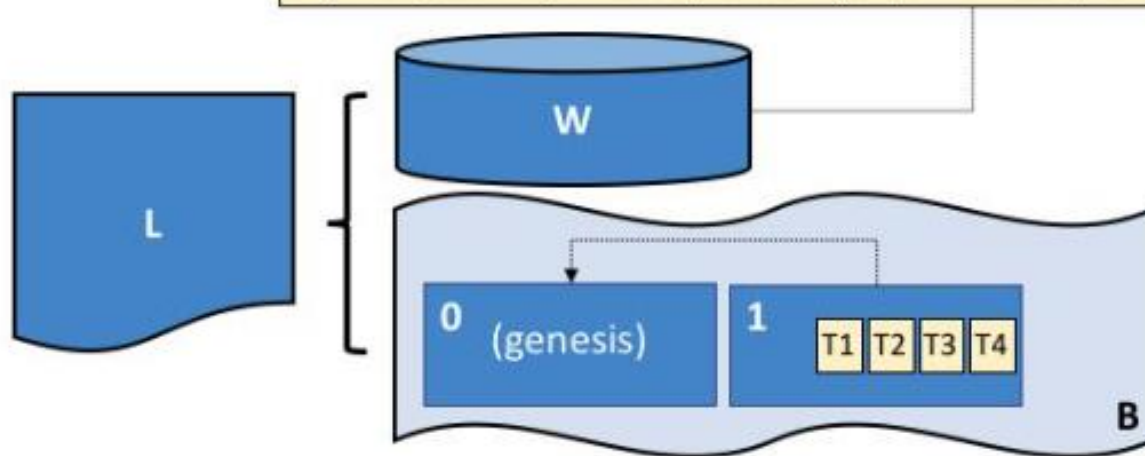
➤ **Orderer节点**：负责接收包含背书签名的交易，对未打包的交易进行排序生成区块，广播给peer节点。



账本 Ledger

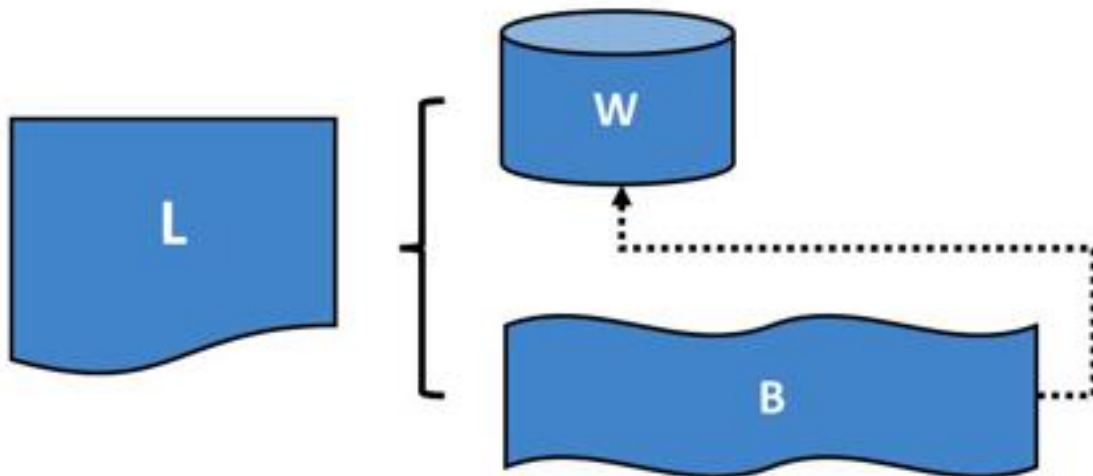
- Fabric的账本由两个不同但相关的部分组成：世界状态和区块链。
- 世界状态(World-Stat) 保存了账本数据的当前值，可以频繁更改，便于应用程序直接访问，本质是一个KV数据库，当前支持LevelDB和CouchDB。

key=CAR3, value={color: yellow, make: Volkswagen, model: Passat, owner: Max}	version=0
key=CAR2, value={color: green, make: Hyundai, model: Tucson, owner: Jin Soo}	version=0
key=CAR1, value={color: red, make: Ford, model: Mustang, owner: Brad}	version=0
key=CAR0, value={color: blue, make: Toyota, model: Prius, owner: Tomoko}	version=0



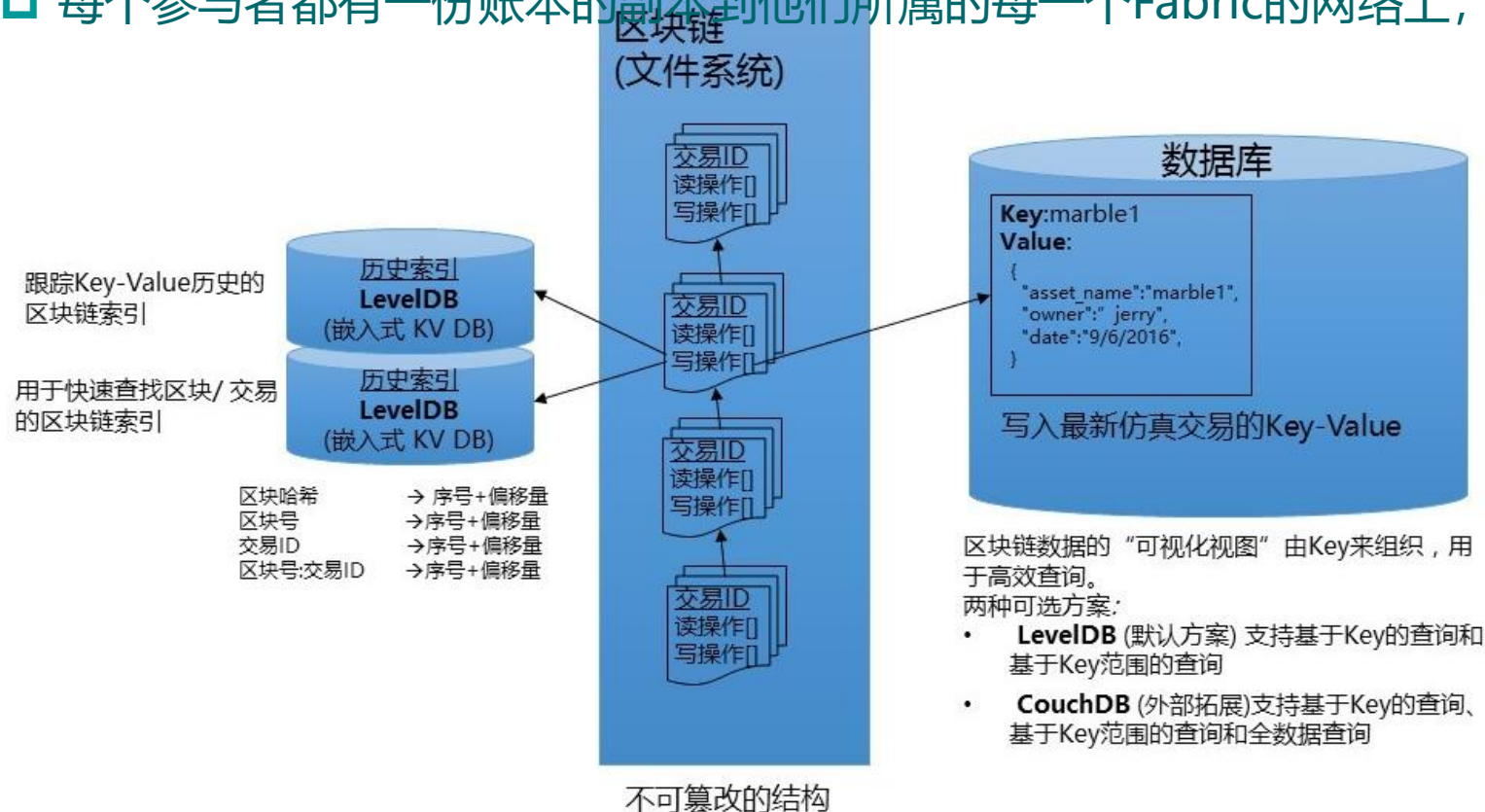
账本 Ledger

- 区块链(Blockchain) 记录了导致当前世界状态的所有更改日志，写入后无法修改，便于历史追踪。



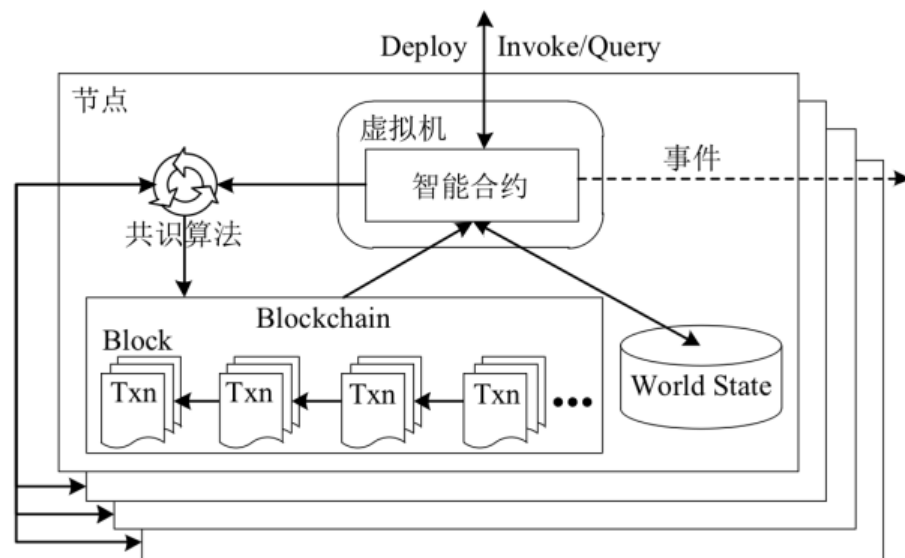
	Ledger
	World State
	Blockchain
	L comprises B and W
	B determines W

- ❑ 区块链数据，这是用文件系统存储在Committer节点上的。区块链中存储了Transaction的读写集；
- ❑ 每个参与者都有一份账本的副本到他们所属的每一个Fabric的网络上；



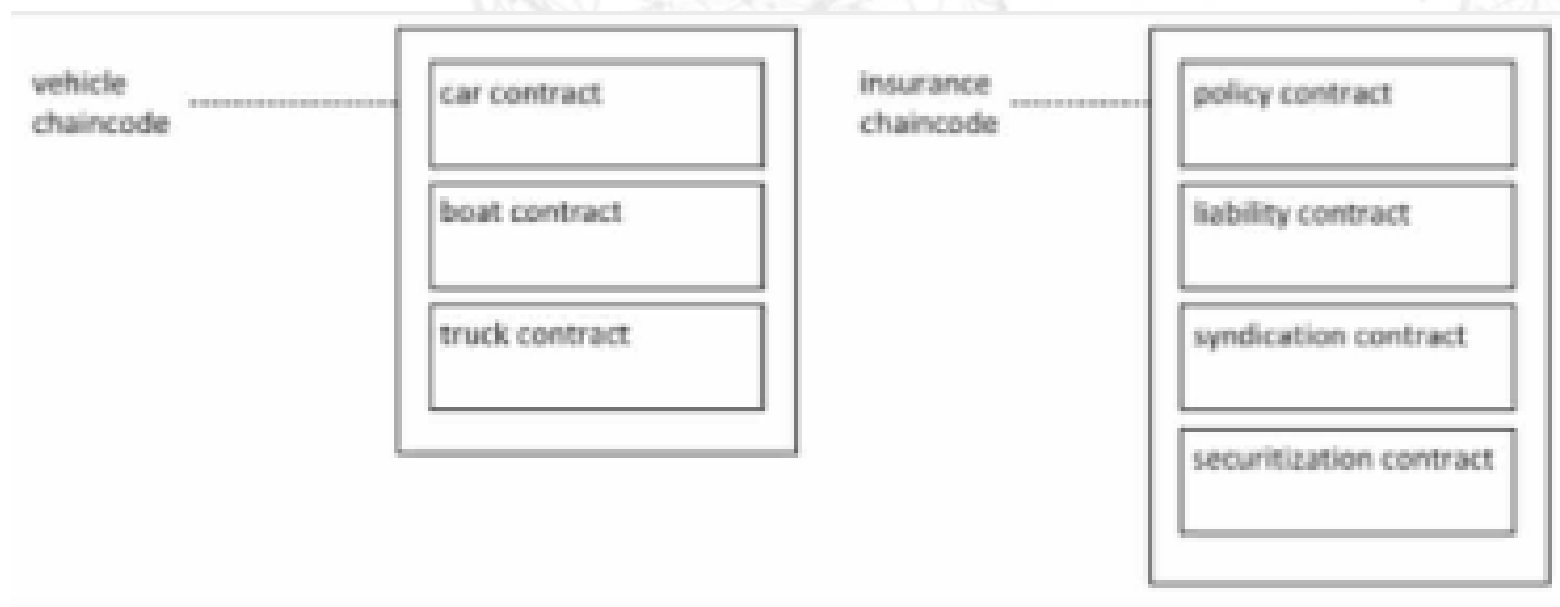
智能合约与链码

- ❑ 超级账本支持**基于主流编程语言的智能合约（链码）设计**，极大地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



智能合约与链码

- 智能合约定义了特定业务流程的交易逻辑，主要在世界状态下放置、获取、删除状态，或查询区块链交易记录，是应用程序开发的重点。
- 链码是一组用于安装和实例化智能合约的技术容器，一个链码可以包含多个智能合约。



智能合约

超级账本



1. 搭建hyperledger环境(docker)
2. 更改配置文件, 启动多个节点 (pbft共识)
3. 编写合约, 编译成二进制文件
4. 把二进制文件放在每个节点中
5. 在各个节点部署智能合约
6. 初始化智能合约

注意与以太坊的区别, 以太坊是通过发送交易, 交易中附带部署代码

公有链与联盟链的区别

智能合约



VS

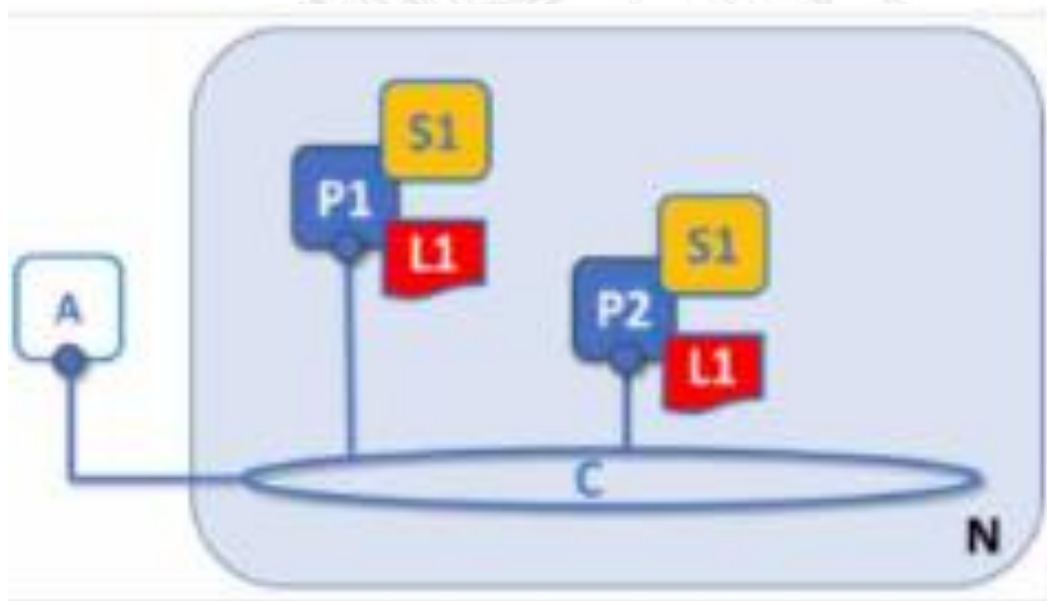


HYPERLEDGER

	以太坊	超级账本
执行环境	EVM	Docker
平台开发语言	Go	Go
智能合约开发	Solidity	Go,Java,Nodejs
部署难度	容易，通过交易部署	相对复杂
合约更新	不能更新*	支持chaincode更新
代币	矿工奖励与Gas机制	无

通道 Channel

- 通道允许一组特定的对等节点和应用程序在区块链网络内相互通信，通道并不实际存在，而是由物理对等节点集合形成的逻辑结构。
- 每个通道都有一个完全独立的账本，这意味着完全独立的区块链，以及完全独立的世界状态。
- 一个组织可以加入多个通道，从而参与多个独立的区块链网络。



Fabric通道

□ 数据隔离和保密

- ✓ 在共识服务上支持多通道消息传递，使得Peer节点可以基于应用访问控制策略来订阅任意数量的通道；也就是说，应用程序指定Peer节点的子集中架设通道。这些peer组成提交到该通道交易的相关者集合，而且只有这些peer可以接收包含相关交易的区块，与其他交易完全隔离
- ✓ 此外，peers的子集将这些私有块提交到不同的账本上，允许它们保护这些私有交易，与其他peers子集的账本隔离开来。应用程序根据业务逻辑决定将交易发送到1个或多个通道。这不是内置的限制，区块链网络不知道并假设不同通道上的交易之间没有关系

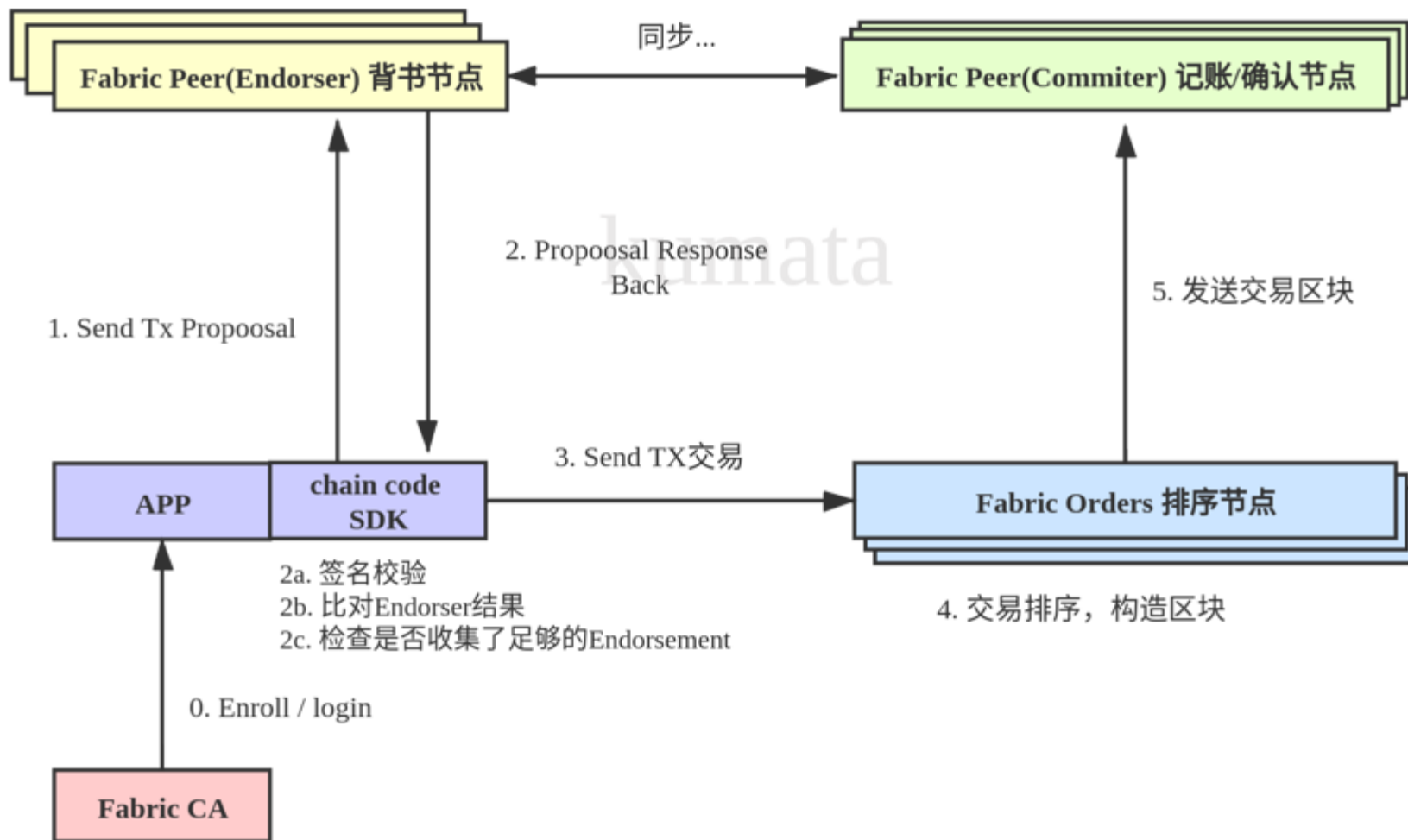
排序服务 Ordering Service

- **Fabric** 采用称为排序服务的特殊节点来执行交易的排序并生成区块，形成一种确定性的共识机制，由排序服务生成的任何块都可以保证是最终且正确的，不会产生分叉。
- 排序服务还维护着允许创建通道（**Channel**）的组织的列表，还对通道实施基本的访问控制，从而限制了谁可以对其进行数据读写和配置。
- 排序服务有三种实现，使得排序节点之间对严格的交易顺序达成共识：
 - **Solo**: 只有一个排序节点，无法容错，但可以用于开发测试环境
 - **Kafka**: 崩溃容错（**CFT**）机制，选举领导者节点，跟随者复制其决策
 - **Raft**: 也是崩溃容错（**CFT**）机制，比**Kafka**易于配置和管理

Fabric交易流程

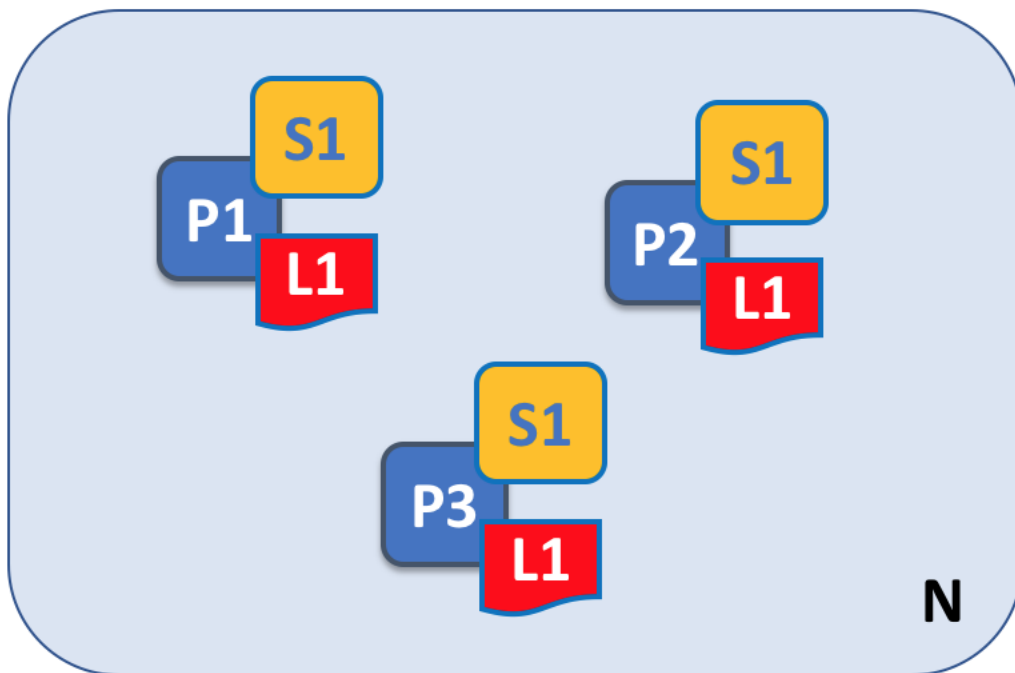
- 1a. 校验Proposal签名
- 1b. 是否满足channel ACL
- 1c. 模拟执行交易，结果签名

- 5a. 检查交易结构/签名/是否重复
- 5b. 检验交易是否符合Endorsement策略
- 5c. 检查读集中版本是否与账本一致
- 5d. 执行区块中的合法交易，账本status更新



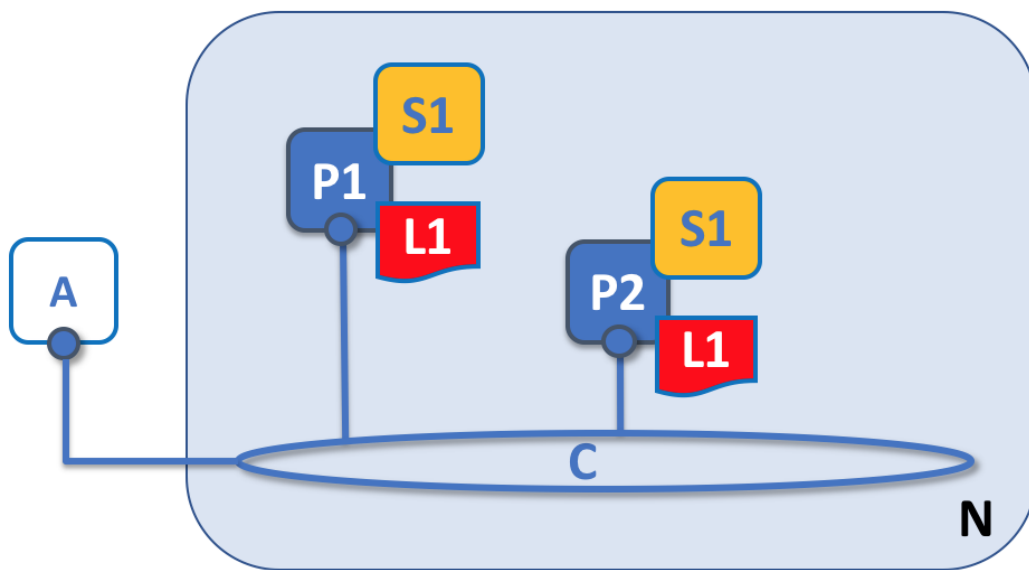
Fabric交易提交流程


- 节点部署



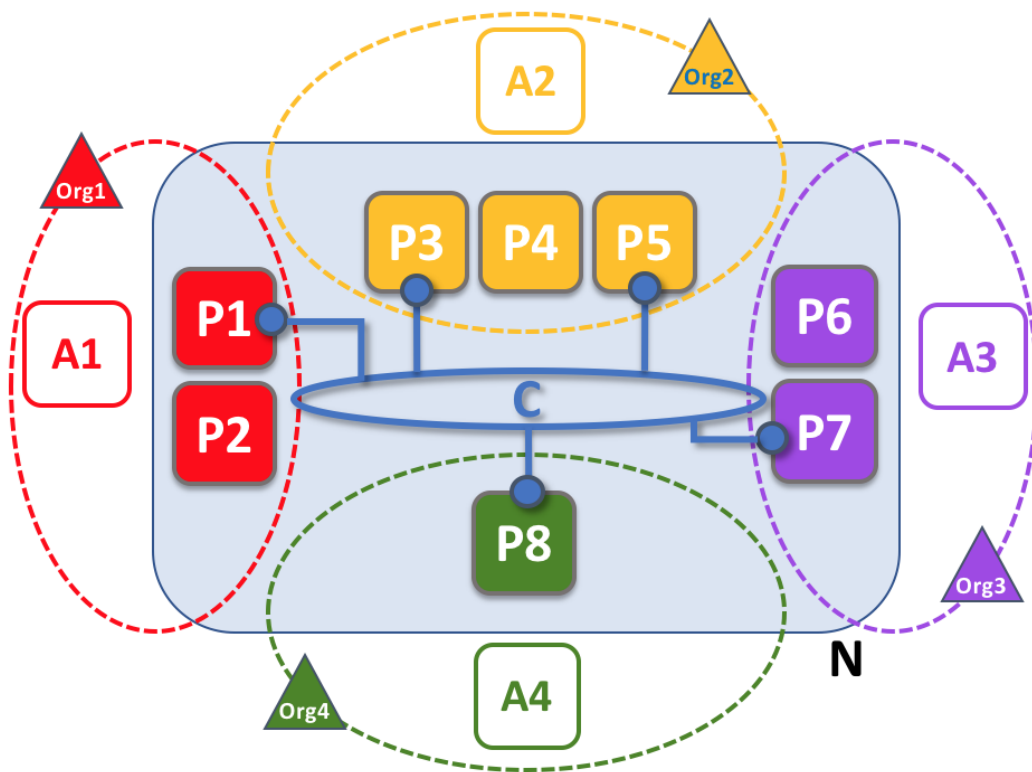
N	Blockchain network
P	Peer node
S	Smart contract (aka chaincode)
L	Ledger

Fabric交易提交流程



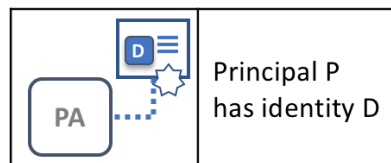
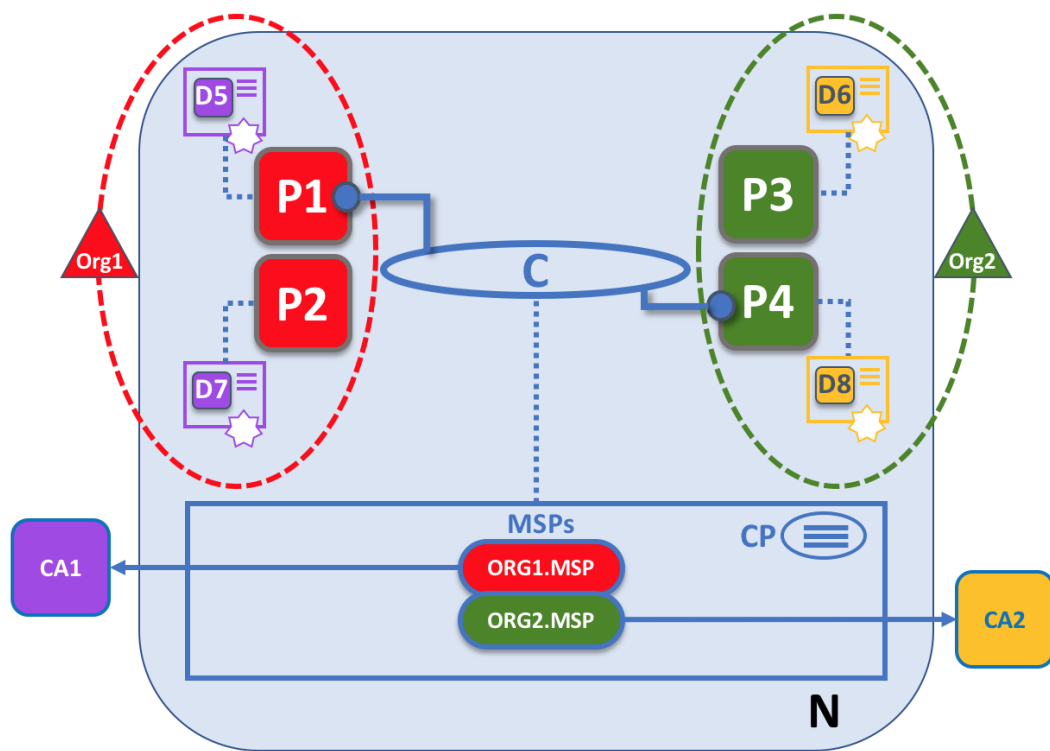
N	Blockchain Network	L	Ledger
C	Channel	A	Application
P	Peer		Principal PA (e.g. A, P1) communicates via channel C.
S	Chaincode		

Fabric交易提交流程



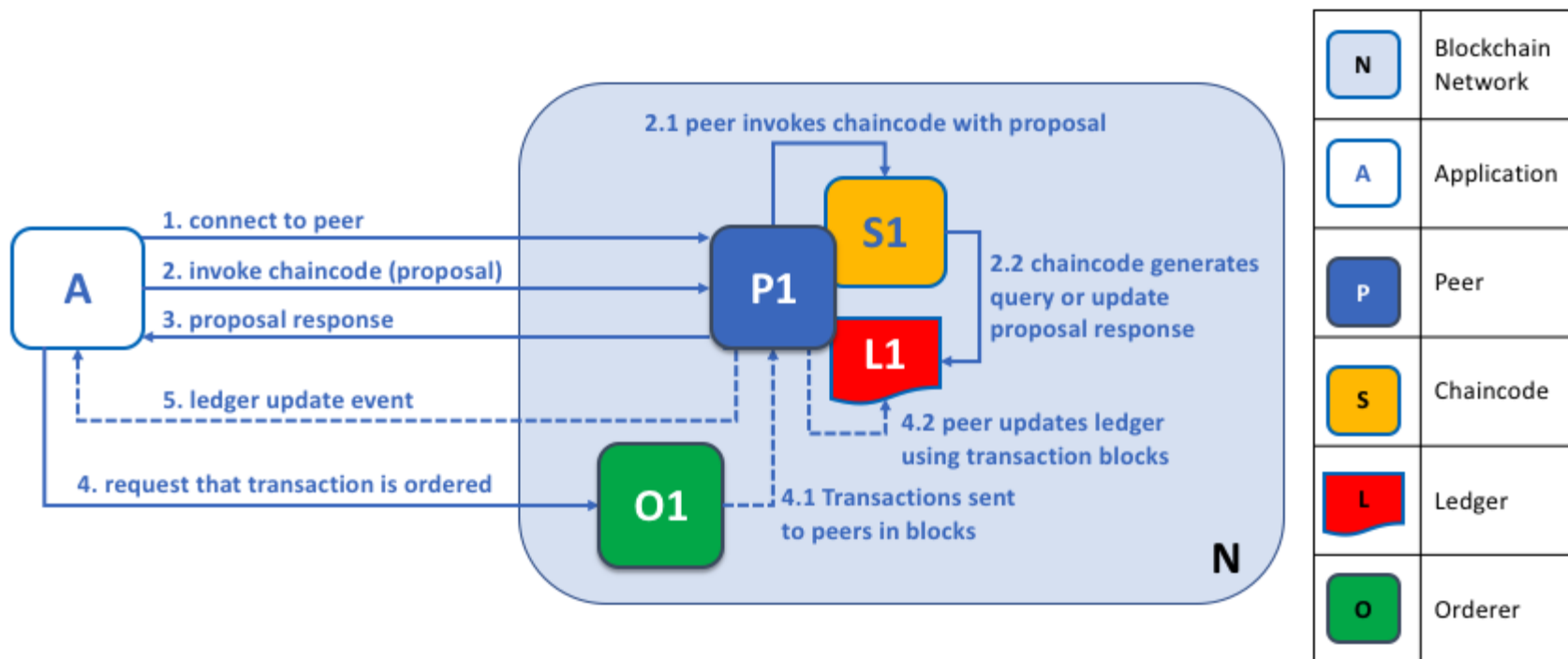
	Blockchain Network		Ledger
	Channel		Application
	Peer		Principal PA (e.g. A1, P5) communicates via channel C.
			Organization
		Organization R owns application A1 and peers P1, P2.	

Fabric交易提交流程

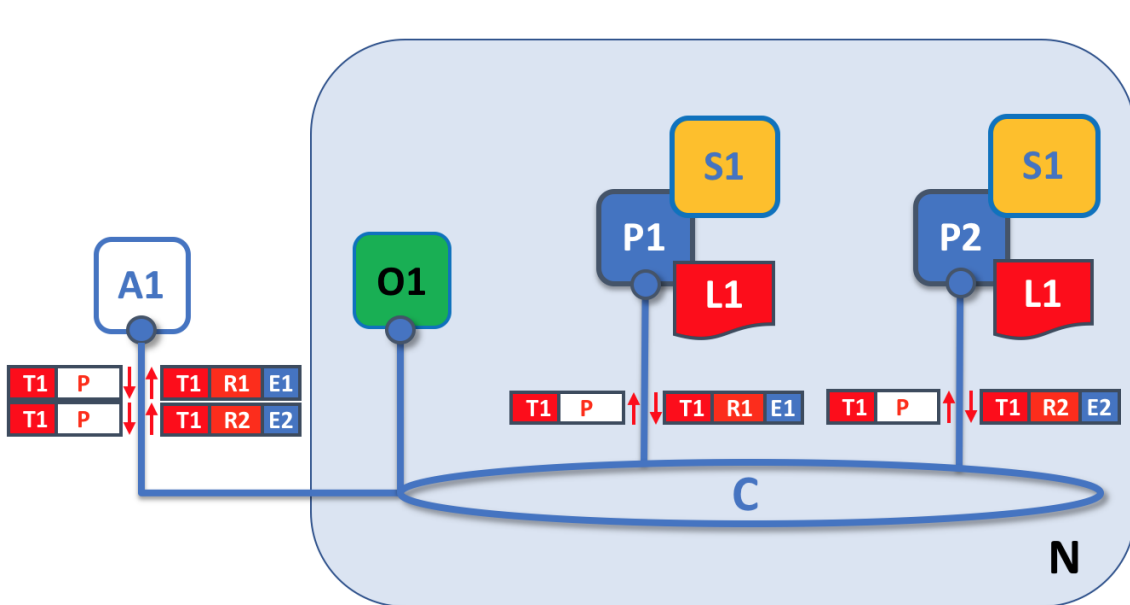


N	Blockchain Network	P	Peer
C	Channel	Org	Organization
I	Identity	PA	Principal PA (e.g. P1,P4) communicates via channel C.
CP	Channel policy	C	
CA	Certificate Authority	MSP	Membership Service Provider
		Organization R owns application A1 and peers P1, P2.	
	Channel C subject to policy CP.		Channel policy CP contains MSPs: MSP1 and MSP2.
		MSP1 selects the Certificate Authority CA1 to provide certificates for it.	

Fabric交易提交流程

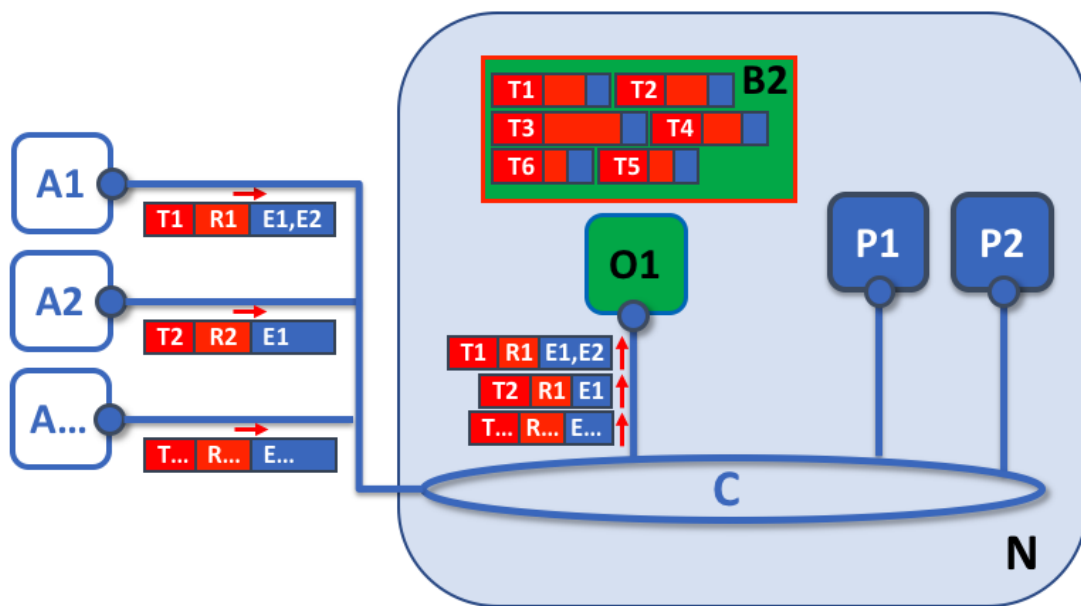











Proposal提案阶段



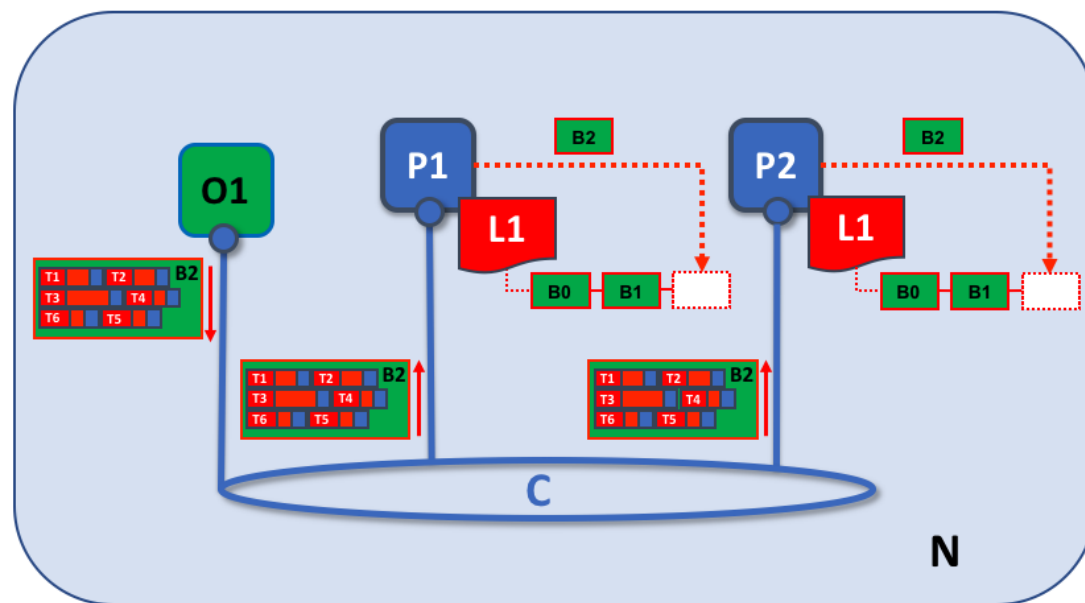
	Blockchain Network		Chaincode
	Channel		Orderer
	Peer		Ledger
	Transaction T1, proposal P		Transaction T1, response R2 endorsed with E2
	Ledger transaction T1 flows on channel C		Principal PA (P1,P2) communicates via channel C.

package打包阶段



	Blockchain Network		Peer
	Block B1		Orderer
	Transaction T1, response R2a endorsed with E2		Channel
	Block B1 contains transactions T1, T2, T3...		
	Ledger transaction T1 flows on channel C		Principal PA (P1,P2) communicates via channel C.

验证阶段



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

Fabric环境搭建

□ 配置Fabric环境的docker-compose文件

- 配置Orderer
- 配置Peer
- 配置CLI

□ 初始化Fabric环境

- 启动Fabric环境的容器
- 创建Channel
- 各个Peer加入Channel
- 更新锚节点

□ 链上代码的安装与运行

- Install ChainCode安装链上代码
- Instantiate ChainCode实例化链上代码
- 发起和查询交易

□ 生成公私钥和证书

- 编译生成cryptogen
- 配置crypto-config.yaml
- 生成公私钥和证书

□ 生成创世区块和Channel配置区块

- 编译生成configtxgen
- 配置configtx.yaml
- 生成创世区块
- 生成Channel配置区块

总结

- Fabric逻辑架构
- Fabric网络架构
- Fabric交易流程