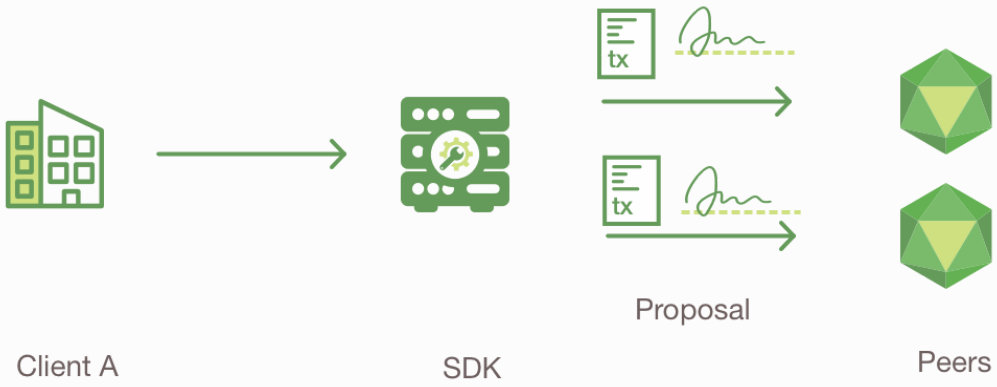
交易流程

本文讲解在一个标准的资产交换中的交易机制。该场景包含两个客户端 A 和 B，他们分别代表萝卜的买方和卖方。他们在网络上都有一个 Peer 节点，他们通过该节点来发送交易和与账本交互。

假设

该流程中，假设已经设置了一个通道，并且该通道正常运行。应用程序的用户已经使用组织的 CA 注册和登记完成，并且拿到了用于在网络中用确认身份的加密材料。

链码（包含了萝卜商店初始状态的键值对）已经安装在 Peer 节点上并在通道上完成了实例化。链码中的逻辑定义了萝卜的交易和定价规则。链码也设置了一个背书策略，该策略是每一笔交易都必须被 `peerA` 和 `peerB` 都签名。

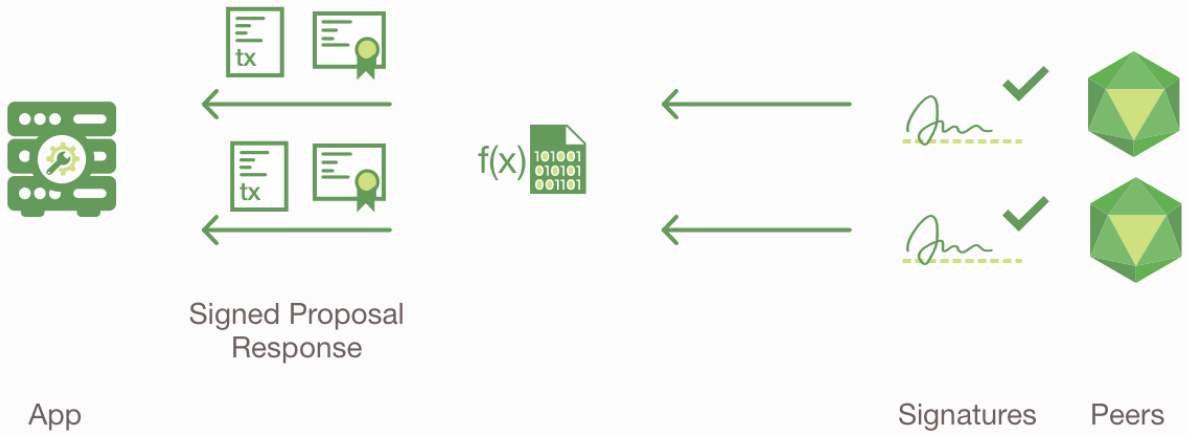


1. **客户端 A 发起一笔交易**

将会发生什么？客户端 A 发送一个采购萝卜的请求。该请求会到达 `peerA` 和 `peerB`，他们分别代表客户端 A 和客户端 B。背书策略要求所有交易都要两个节点背书，因此请求要到经过 `peerA` 和 `peerB`。

然后，要构建一个交易提案。应用程序使用所支持的 SDK（Node，Java，Python）中的 API 生成一个交易提案。提案是带有确定输入参数的调用链码方法的请求，该请求的作用是读取或者更新账本。

SDK 的作用是将交易提案打包成合适的格式（gRPC 使用的 protocol buffer）以及根据用户的密钥对交易提案生成签名。



2. **背书节点验证签名并执行交易**

背书节点验证（1）交易提案的格式完整，（2）且验证该交易提案之前没有被提交过（重放攻击保护），（3）验证签名是有效的（使用 MSP），（4）验证发起者（在这个例子中是客户端 A）有权在该通道上执行该操作（也就是说，每个背书节点确保发起者满足通道 *Writers* 策略）。背书节点将交易提案输入作为调用的链码函数的参数。然后根据当前状态数据库执行链码，生成交易结果，包括响应值、读集和写集（即表示要创建或更新的资产的键值对）。目前没有对账本进行更新。这些值以及背书节点的签名会一起作为“提案响应”返回到 SDK，SDK 会为应用程序解析该响应。

注解

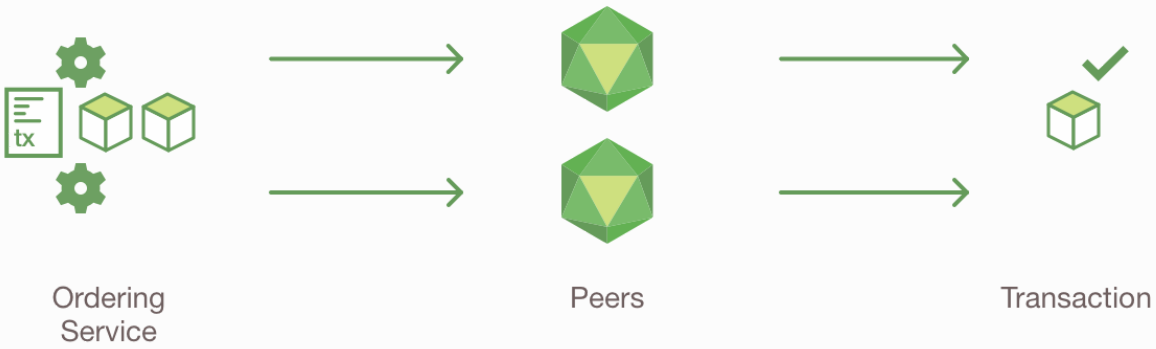
MSP 是节点的组件，它允许 Peer 节点验证来自客户端的交易请求，并签署交易结果（即背书）。写入策略在通道创建时就会定义，用来确定哪些用户有权向该通道提交交易。有关成员关系的更多信息，请查看 成员服务提供者 (MSP) 文档。

3. **检查提案响应**

应用程序验证背书节点的签名，并比较这些提案响应，以确定其是否相同。如果链码只查询账本，应用程序将检查查询响应，并且通常不会将交易提交给排序服务。如果客户端应用程序打算向排序服务提交交易以更新账本，则应用程序在提交之前需确定是否已满足指定的背书策略（即 peerA 和 peerB 都要背书）。该结构是这样的，即使应用程序选择不检查响应或以其他方式转发未背书的交易，节点仍会执行背书策略，并在提交验证阶段遵守该策略。

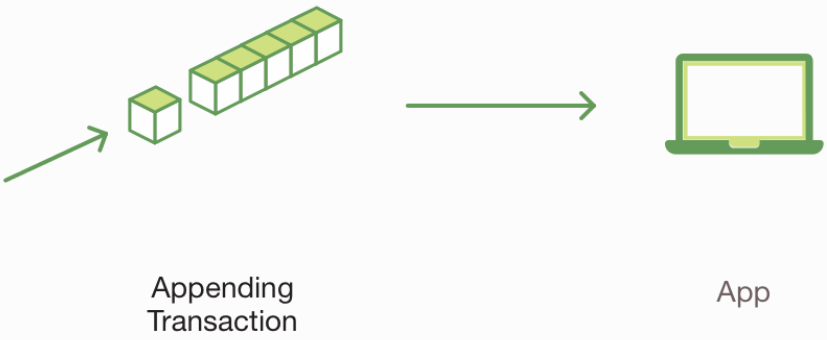
4. **客户端将背书结果封装进交易**

应用程序将交易提案和“交易消息”中的交易响应“广播”给排序服务。交易会包含读写集，背书节点的签名和通道 ID。排序服务不需要为了执行其操作而检查交易的整个内容，它只是从网络中的所有通道接收交易，将它们按时间按通道排序，并将每个通道的交易打包成区块。



5. **验证和提交交易**

交易区块被“发送”给通道上的所有 Peer 节点。对区块内的交易进行验证，以确保满足背书策略，并确保自交易执行生成读集以来，读集中变量的账本状态没有变化。块中的交易会被标记为有效或无效。



6. **账本更新**

6. **Ledger updated**

每个 Peer 节点都将区块追加到通道的链上，对于每个有效的交易，写集都提交到当前状态数据库。系统会发出一个事件，通知客户端应用程序本次交易（调用）已被不可更改地附加到链上，同时还会通知

交易验证结果是有效还是无效。

> **注解**
>
> 应用程序应该在提交交易后监听交易事件，例如使用 `submitTransaction` API，它会自动监听交易事件。如果不监听交易事件，您将不知道您的交易是否已经被排序、验证并提交到账本。

查看下边的泳道图来更好的理解交易流程。