Gossip 数据传播协议

Hyperledger Fabric 通过将工作负载拆分为交易执行（背书和提交）节点和交易排序节点的方式来优化区块链网络的性能、安全性和可扩展性。这样对网络的分割就需要一个安全、可靠和可扩展的数据传播协议来保证数据的完整性和一致性。为了满足这个需求，Fabric 实现了 **Gossip 数据传播协议** 。

Gossip 协议

Peer 节点通过 gossip 协议来传播账本和通道数据。Gossip 消息是持续的，通道中的每一个 Peer 节点不断地从多个节点接收当前一致的账本数据。每一个 gossip 消息都是带有签名的，因此拜占庭成员发送的伪造消息很容易就会被识别，并且非目标节点也不会接受与其无关的消息。Peer 节点会受到延迟、网络分区或者其他原因影响而丢失区块，这时节点会通过从其他拥有这些丢失区块的节点处同步账本。

基于 gossip 的数据传播协议在 Fabric 网络中有三个主要功能：

1. 通过持续的识别可用的成员节点来管理节点发现和通道成员，还有检测离线节点。
2. 向通道中的所有节点传播账本数据。所有没有和当前通道的数据同步的节点会识别丢失的区块，并将正确的数据复制过来以使自己同步。
3. 通过点对点的数据传输方式，使新节点以最快速度连接到网络中并同步账本数据。

Peer 节点基于 gossip 的数据广播操作接收通道中其他的节点的信息，然后将这些信息随机发送给通道上的一些其他节点，随机发送的节点数量是一个可配置的常量。Peer 节点可以用“拉”的方式获取信息而不用一直等待。这是一个重复的过程，以使通道中的成员、账本和状态信息同步并保持最新。在分发新区块的时候，通道中 **主** 节点从排序服务拉取数据然后分发给它所在组织的节点。

主节点选举

主节点的选举机制用于在每一个组织中 **选举** 出一个用于链接排序服务和开始分发新区块的节点。主节点选举使得系统可以有效地利用排序服务的带宽。主节点选举模型有两种模式可供选择：

1. **静态模式**：系统管理员手动配置一个节点为组织的主节点。
2. **动态模式**：组织中的节点自己选举出一个主节点。

静态主节点选举

静态主节点选举允许你手动设置组织中的一个或多个节点节点为主节点。请注意，太多的节点连接到排序服务可能会影响带宽使用效率。要开启静态主节点选举模式，需要配置 `core.yaml` 中的如下部分：

```
peer:
    # Gossip related configuration
    gossip:
        useLeaderElection: false
        orgLeader: true
```

另外，这些配置的参数可以通过环境变量覆盖：

```
export CORE_PEER_GOSSIP_USELEADERELECTION=false
export CORE_PEER_GOSSIP_ORGLEADER=true
```

注解

下边的设置会使节点进入 **旁观者** 模式，也就是说，它不会试图成为一个主节点：

```
export CORE_PEER_GOSSIP_USELEADERELECTION=false
export CORE_PEER_GOSSIP_ORGLEADER=false
```

不要将 `CORE_PEER_GOSSIP_USELEADERELECTION` 和 `CORE_PEER_GOSSIP_ORGLEADER` 都设置为 `true`，这将会导致错误。

使用静态配置时，主节点失效或者崩溃都需要管理员进行处理。

动态主节点选举

动态主节点选举使组织中的节点可以 **选举** 一个节点来连接排序服务并拉取新区块。这个主节点由每个组织单独选举。

动态选举出的主节点通过向其他节点发送 **心跳** 信息来证明自己处于存活状态。如果一个或者更多的节点在一个段时间内没有收到 **心跳** 信息，它们就会选举出一个新的主节点。

在网络比较差有多个网络分区存在的情况下，组织中会存在多个主节点以保证组织中节点的正常工作。在网络恢复正常之后，其中一个主节点会放弃领导权。在一个没有网络分区的稳定状态下，会只有 **唯一** 一个活动的主节点和排序服务相连。

下边的配置控制主节点 **心跳** 信息的发送频率：

```
peer:
    # Gossip related configuration
    gossip:
        election:
            leaderAliveThreshold: 10s
```

开启动态节点选举，需要配置 `core.yaml` 中的以下参数：

```
peer:
    # Gossip related configuration
    gossip:
        useLeaderElection: true
        orgLeader: false
```

同样，这些配置的参数可以通过环境变量覆盖：

```
export CORE_PEER_GOSSIP_USELEADERELECTION=true
export CORE_PEER_GOSSIP_ORGLEADER=false
```

锚节点

gossip 利用锚节点来保证不同组织间的互相通信。

当提交了一个包含锚节点更新的配置区块时，Peer 节点会连接到锚节点并获取它所知道的所有节点信息。一个组织中至少有一个节点连接到了锚节点，锚节点就可以获取通道中所有节点的信息。因为 gossip 的通信是固定的，而且 Peer 节点总会被告知它们不知道的节点，所以可以建立起一个通道上成员的视图。

例如，假设我们在一个通道有三个组织 *A*、*B* 和 *C*，组织 *C* 定义了锚节点 *peer0.orgC*。当 *peer1.orgA* 连接到 *peer0.orgC* 时，它将会告诉 *peer0.orgC* 有关 *peer0.orgA* 的信息。稍后等 *peer1.orgB* 连接到 *peer0.orgC* 时，后者也会告诉前者关于 *peer0.orgA* 的信息。在这之后，组织 *A* 和组织 *B* 可以开始直接交换成员信息而无需借助 *peer0.orgC* 了。

由于组织间的通信依赖于 gossip，所以在通道配置中必须至少有一个锚节点。为了系统的可用性和冗余性，我们强烈建议每个组织都提供自己的一些锚节点。注意，锚节点不一定和主节点是同一个节点。

外部和内部端点（endpoint）

为了让 gossip 高效地工作，Peer 节点需要包含其所在组织以及其他组织的端点信息。

当 Peer 节点启动的时候，它会使用 `core.yaml` 文件中的 `peer.gossip.bootstrap` 来宣传自己并交换成员信息，同时建立所属组织中可用节点的视图。

`core.yaml` 文件中的 `peer.gossip.bootstrap` 属性用于在 **一个组织内部** 启动 gossip。如果你要使用 gossip，通常要为组织中的所有节点配置一组启动节点（使用空格隔开的节点列表）。内部端点通常是由 Peer 节点自动计算的，或者在 `core.yaml` 中的 `core.peer.address` 指明。如果你要覆盖该值，你可以设置环境变量 `CORE_PEER_GOSSIP_ENDPOINT`。

启动信息也同样需要建立 **跨组织** 的通信。初始的跨组织启动信息通过上面所说的“锚节点”设置提供。如果想让其他组织知道你所在组织中的其他节点，你需要设置 `core.yaml` 文件中的 `peer.gossip.externalendpoint`。如果没有设置，节点的端点信息就不会广播到其他组织的 Peer 节点。

这些属性的设置如下：

```
export CORE_PEER_GOSSIP_BOOTSTRAP=<a list of peer endpoints within the peer's org>
export CORE_PEER_GOSSIP_EXTERNALENDPOINT=<the peer endpoint, as known outside the org>
```

Gossip 消息

在线的节点通过持续广播“存活”消息来表明其处于可用状态，每一条消息都包含了“公钥基础设施（PKI）” ID 和发送者的签名。节点通过收集这些存活的消息来维护通道成员。如果没有节点收到某个节点的存活信息，这个“死亡”的节点会被从通道成员关系中剔除。因为“存活”的消息是经过签名的，恶意节点无法假冒其他节点，因为他们没有根 CA 签发的密钥。

除了自动转发接收到的消息之外，状态协调进程还会在每个通道上的 Peer 节点之间同步 **世界状态**。每个 Peer 节点都持续从通道中的其他节点拉取区块，来修复他们缺失的状态。因为基于 gossip 的数据分发不需要固定的连接，所以该过程可以可靠地提供共享账本的一致性和完整性，包括对节点崩溃的容忍。

因为通道是隔离的，所以一个通道中的节点无法和其他通道通信或者共享信息。尽管节点可以加入多个通道，但是分区消息传递通过基于 Peer 节点所在通道的应用消息的路由策略，来防止区块被分发到其他通道的 Peer 节点。

注解

1. 通过 Peer 节点 TLS 层来处理点对点消息的安全性，不需要使用签名。Peer 节点通过 CA 签发的证书来授权。尽管没有使用 TLS 证书，但在 gossip 层使用了经过授权的 Peer 节点证书。账本区块经过排序服务签名，然后被分发到通道上的主节点。
2. 通过 Peer 节点的成员服务提供者来管理授权。当 Peer 节点第一次连接到通道时，TLS 会话将与成员身份绑定。这就利用网络和通道中成员的身份来验证了与 Peer 节点相连的节点的身份。