Utopia Plume 轻节点方案

Plume 轻节点方案主要来为 Utopia 联盟链提供完全去中心化的轻节点方案；

在联盟链的实现中通常会使用 client / server 模式来为 Appliaction 提供区块链服务，其中 Server endpoint 由联盟链中某种角色的节点来提供，那么 client 通常固定在一个或几个 server 上使用联盟链上的数据和合约；

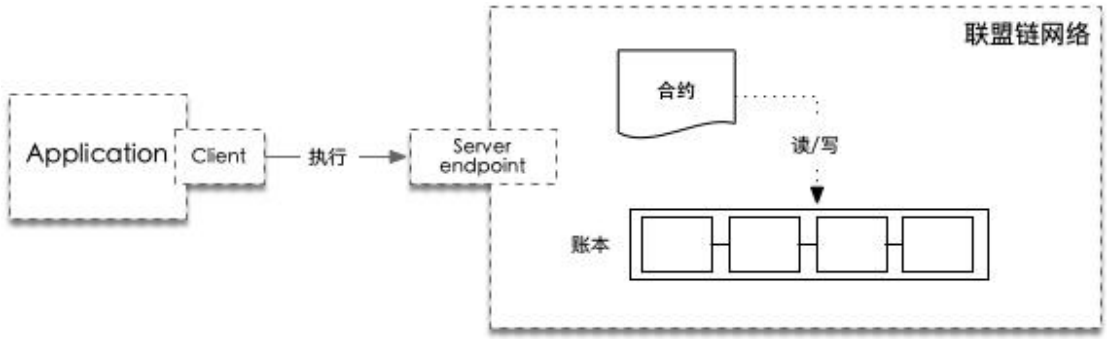

图 1

如“图 1”所示当 Application 想要使用区块链服务时，需要使用 Server 提供的中心化 endpoint ，其服务的可用性完全依赖于当前 Server，而并非一个可动态扩展和伸缩的区块链网络，当服务发生故障时甚至无法快速识别是联盟链网络故障还是 endporint 故障，对于提供服务的 Server 的维护也提出了很高的要求。

Utopia 提供的联盟链网络可以提供 ”图 1“ 所示的调用关系，同时也设计了一套轻节点方案，合理利用 Utopia p2p 网络特性，使 Application 能够更加便捷的使用区块链服务，其特点是“去中心、简单、低成本”；

Utopia 的网络运行方式是节点首先要通过 bootnode 加入到网络中，然后声明自己的角色，并与其他节点建立相应的关系，进而去使用服务或提供服务;

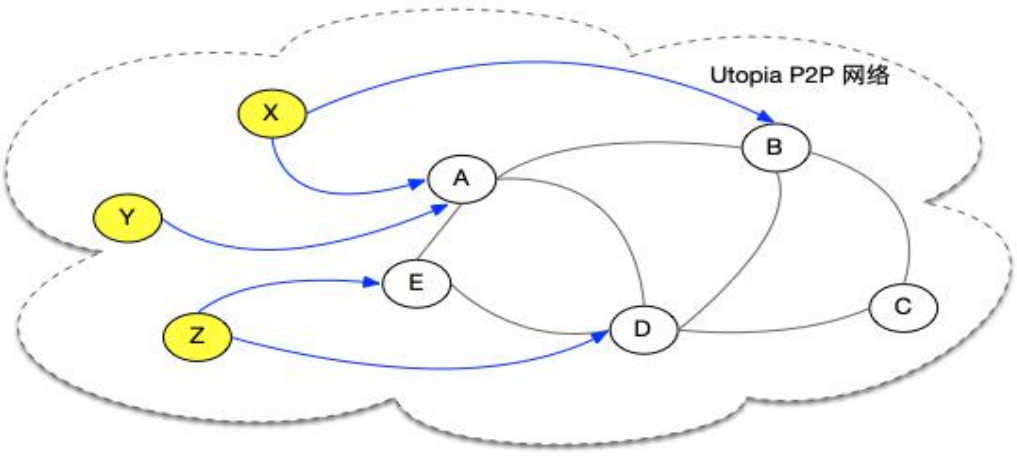

图 2

假设当前网络中存在着 ABCDEXYZ 这 8 个节点，其中 XYZ 为三个 Plumes 节点(轻节点)，假设其拓扑关系如“图 2”所示，假设图中的实线为物理连接关系，其中 XYZ 三个节点上的连线带箭头，表示 Plumes 这类节点并不接收 Inbound 的连接，其他不带箭头的节点是支持 Inbound/Outbound 两种类型连接的，当然 Plumes 节点也会拒绝承担路由节点的角色，除非启动节点时通过参数强行开启 Inbound 和 Relay 功能；

组网成功以后，节点们会按角色分成不同的关系组，每个节点加入网络成功后都会尝试进入不同的分组；

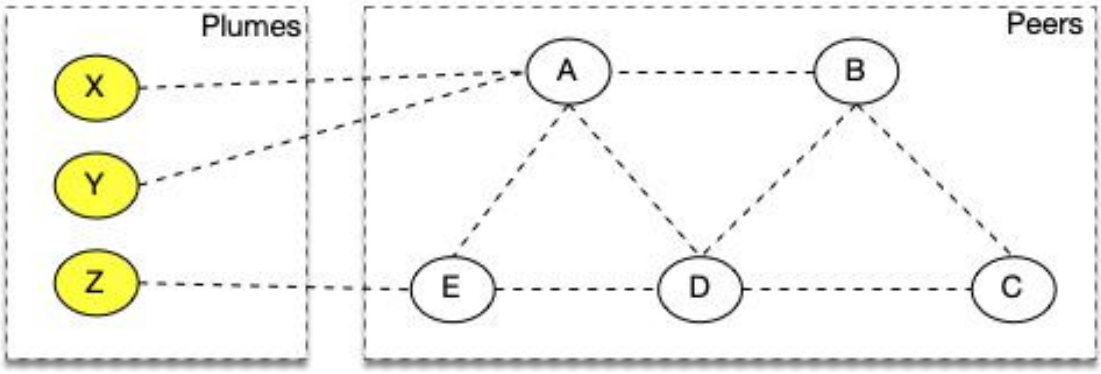

图 3

在“图 3”中描述了节点的分组和逻辑关系，节点间的虚线表示逻辑关系，图中 Peers 分组为区块链网络中的全节点集合，可以为 Plumes 组的节点提供数据读写服务；

可以看到 Plumes 组内 XYZ 三个节点间并没有虚线，在实际的网络中实线也没有，只要它们成功加入到网络中，就可以使用 Peers 提供的服务，并且对服务节点进行身份识别以及服务质量识别。

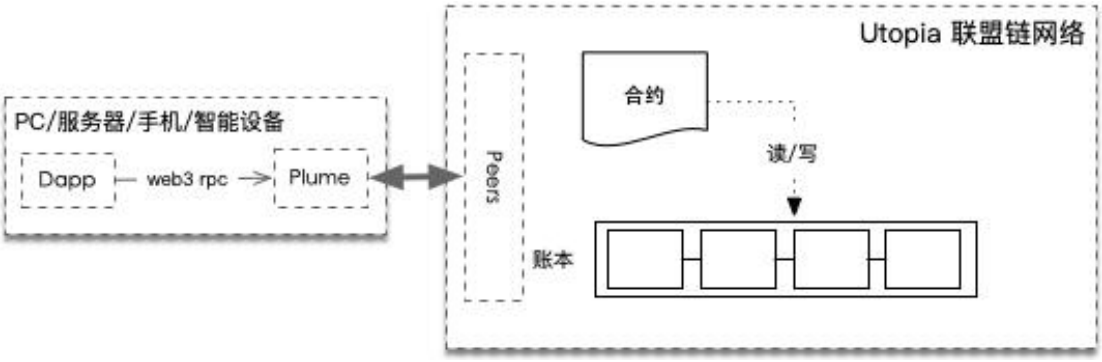

图 4

当一个终端用户想要使用 Utopia 区块链服务时，只需要简单的启动一个 Plumes 加入到网络中即可通过 Plume 提供的 RPC 来使用区块链服务，并不能确定是由哪个具体的 peer 在为 plume 节点提供服务，算法会替我们选择最适合的 peer，至此我们成功将 Application 转化为 Dapp 。

在 ”图 5” 中抽象了三个类型的节点交互过程:

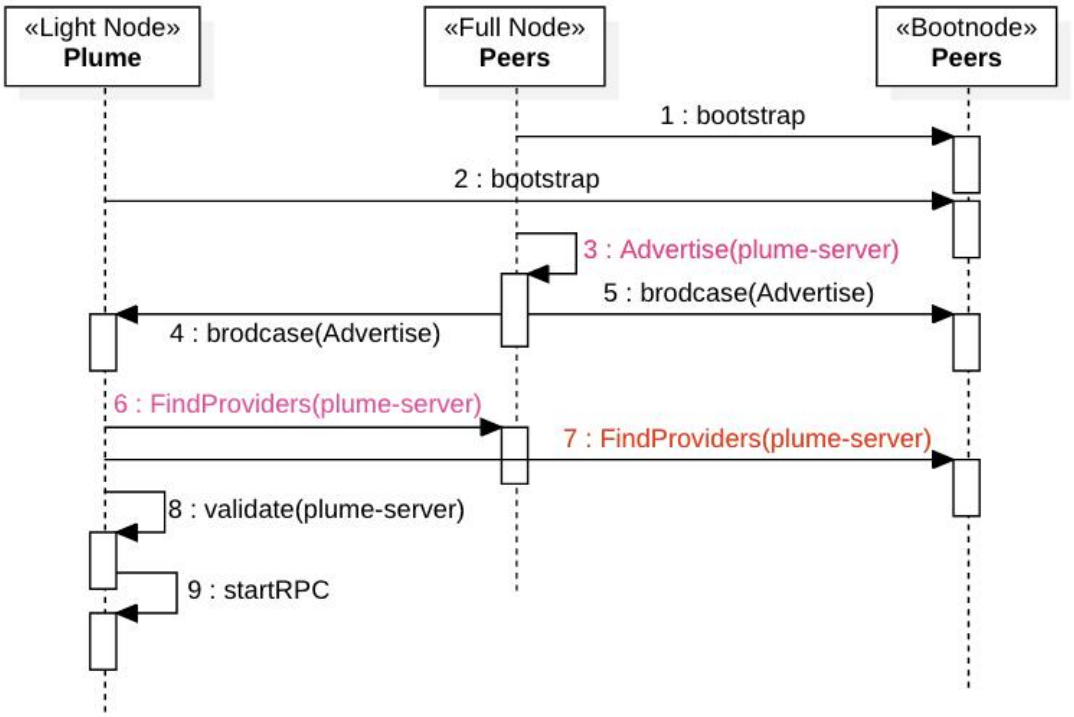

图 5

- 1、2 bootstrap：每个节点都要执行“bootstrap”操作来加入 Utopia p2p 网络；
- 3 Advertise：当“Full Node”类型节点入网后会尝试跟其他节点进行应用层面的握手，以便确立“Peer”关系，成功获取“Peer”身份并同步到最新块后通过执行“Advertise”来宣告自己的“plume-server”角色；
- 4、5 brodcase(Advertise)：这里 Advertise 并非以 GOSSIP 方式全网广播，而是仅广播到 DHT 上等待 Light 节点查询；
- 6、7 FindProviders(plume-server)：“Light Node”不需要与其他节点进行应用层握手，而是通过“FindProviders”来 Query 网络中扮演“plume-server”角色的节点，此查询只在 DHT 上进行而非全网 Query;
- 8 validate(plume-server)：找到目标节点后对其进行检查，包括可用性可靠性等等；
- 9 startRPC：检查通过后在本地执行 startRPC 启动服务，为 Dapp 提供 RPC 服务;

Plume 节点选择服务节点的规则：

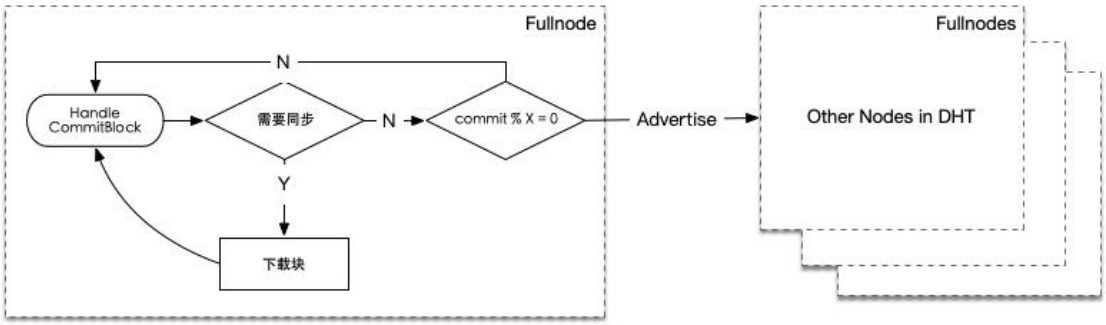

图 6-1

全节点在非同步状态下每隔 X 个 commit 块执行一次 Advertise 刷新自己的 Advertise 时间，其有效期为 X+1 个 commit 块的时间（图 6-1）；

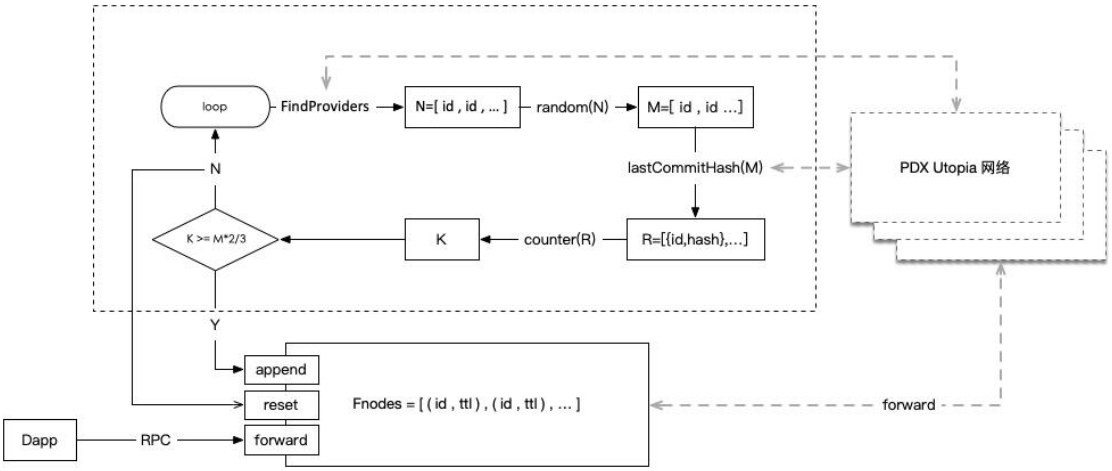

图 6-2

Plume 节点启动 loop 线程周期性执行 FindProviders 最多一次获取 N 个全节点的 ID（FindProviders 会过滤掉没有及时发送心跳的节点 ID）,再随机选出 M 个节点同时并发执行 lastCommitHash 询问最新的 commitHash 并得到结果集 R，对结果集 R 进行汇总，如果相同的 Hash 数 K 超过 M*2/3 个，则将提供了相同 Hash 的 K 个节点 通过 Fnodes.append 方法放入 Fnodes 集合备用，否则执行 Fnodes.reset 清空集合；

当 Dapp 将 RPC 请求发送到 Plume 节点时，按照 ttl 排序 Fnodes 集合并同步执行 forward 方法向集合中的节点转发请求，有节点接收请求并返回正确结果则终止（图 6-2）；

建议发送 sendRawTransaction 得到 txHash 时用 LRU Cache 记录提供 txHash 的节点信息，在查询执行状态时跳过这个节点，这么做的目的是将执行和查询分散在不同节点上；