# Tx 广播优化方案

假设 Utopia 的网络由两种节点构成：

1. 普通节点：在网络中不挖矿也不发 assertion ，下图中用 A 集合表示，即 A = [A1,A2,...,An] ;

2. 委员会节点：负责出块的节点，下图中用 Q 集合表示，Q = [Q1,Q2,...,Qn] ;

## 优化前

下图模拟了优化前的交易发布流程，由共识算法的特点可以得知 Q 集合是相对稳定的节点，负责出块，交易在 A 集合的 txpool 中就浪费了 A 集合节点的内存，广播范围是 A + Q 集合也就浪费了全网的流量，降低了整体资源使用率；

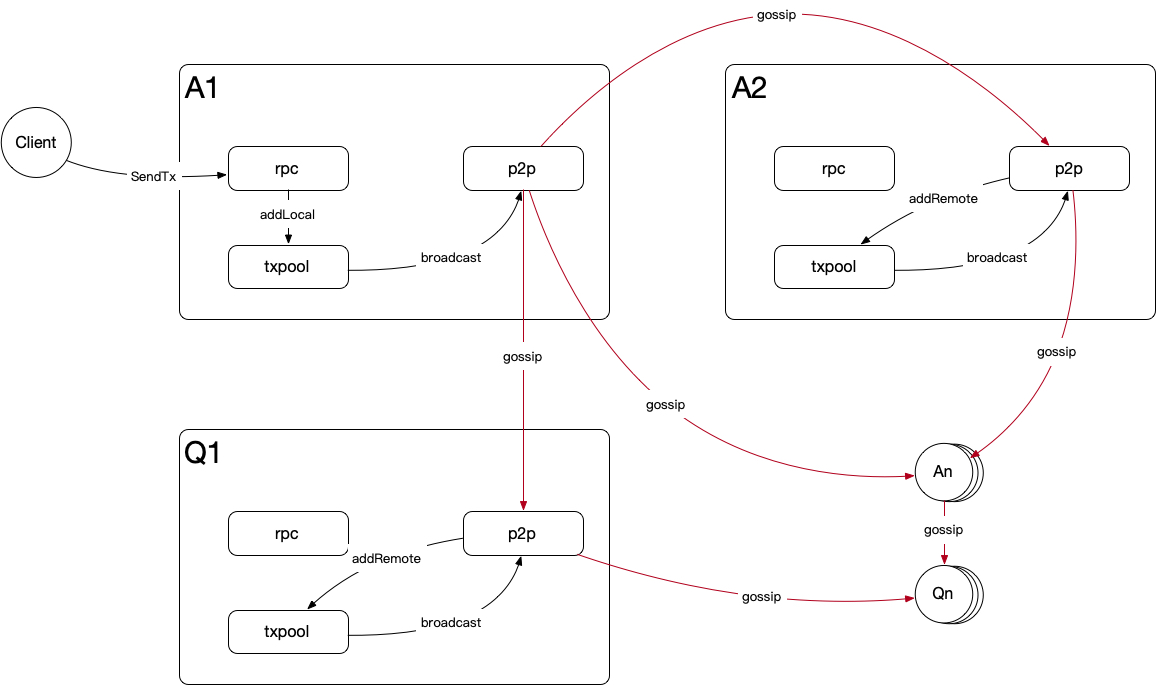


图1

1. 交易从 Client 端通过 rpc 接口发送到节点 A1；

2. 通过 txpool.addLocal 将交易放到交易池中；

3. 从 txpool 发出广播事件，通知 p2p 模块进行广播；

4. 筛选本地 peer 执行 gossip 逻辑，当 peer 中包含委员会成员时，优先广播给委员会成员；

## 优化后

优化的主要目标是缩小 tx 的广播范围，让 tx 只在委员会节点间 gossip ，并且交易只在发起节点和委员会节点的 txpool 中存在，这样节省不相关节点的内存使用率，也会降低全网广播交易所用的流量；

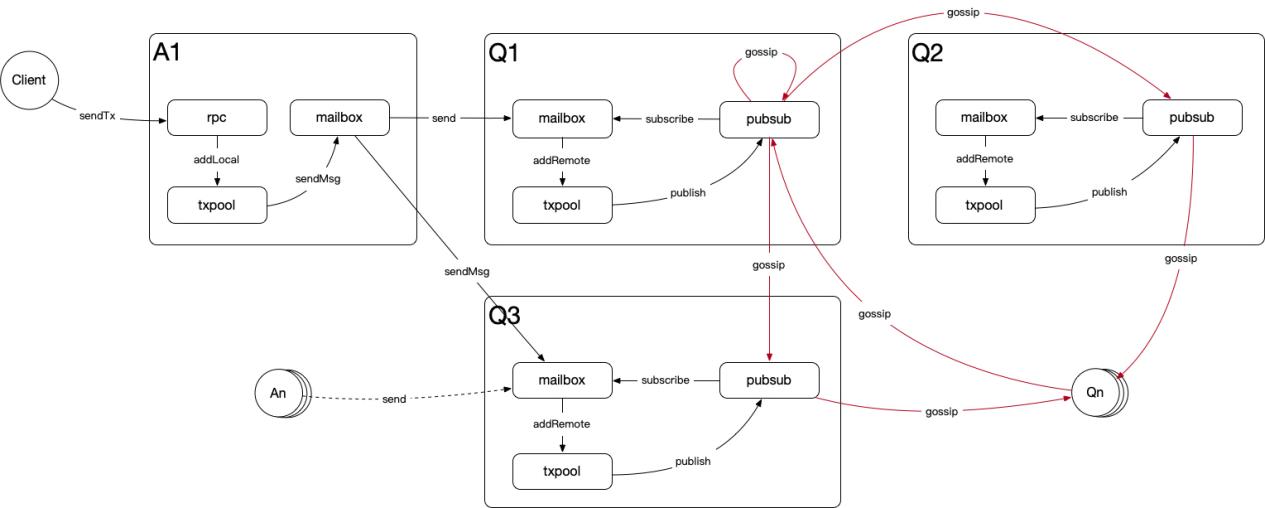


图2

1. 交易从 Client 端通过 rpc 接口发送到节点 A1；

2. 通过 txpool.addLocal 将交易放到交易池中；

3. 重写交易的广播逻辑，根据 txhash 匹配两个委员会成员并通过 mailbox 通道传递消息，图中是 Q1 和 Q3；

4. Q1 和 Q3 会执行相同的逻辑，我们以 Q1 为例，通过 addRemote 将交易放入 txpool ；

5. Q1 判断自己是委员会成员，直接将交易 publish 到 pubsub 模块；

6. 消息到达 pubsub 模块后会执行 gossip 逻辑，目标中也包含了自己；

7. Q1 节点 subscribe 到自己广播过来的消息后会进行忽略，Q2 节点 subscribe 到 Q1 的消息后会重复执行 1~5 步，在 publish 时会对消息进行判断，如果是来自 pubsub 通道，则忽略，如果消息来自 mailbox 通道则继续执行 publish；