**TBFT建模进度**

**摘要**

本文档介绍了Tendermint-BFT（TBFT）共识机制的初步框架、核心功能以及协议描述。本文首先展示了协议框架，完善了功能描述，修改了理想功能的初步描述，展示了对于共识模块的描述。

1. **初步框架**

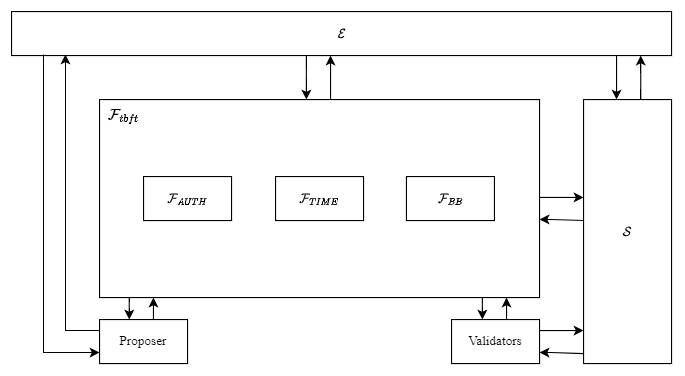


图 1 TBFT协议初步框架

**二、功能描述**

（一）功能

–当从参与方A收到( Send,sid, B, m )时，

将( Sent, sid, A, B, m )发送给A。

–当从A收到( Send, sid, B', m' )时，执行以下操作：

如果A是被破坏的，则向参与方B'输出( Sent, sid, A, m' )。

否则，向参与方B输出( Sent, sid, A, m )。终止操作。

（二）功能

初始化：设置Proposal := ⊥和Round := 0。

–当收到消息(startProposal)时，

* 通过Round-robin规则选定提议者Proposer H，H为V中诚实者的集合：

-初始化Validator的votingPower为其质押资金：

-按Round-robin规则依次选举Proposer，更新Round := Round+1。

* 更新votingPower：

-未被选中的Validator更新为：

-被选中为Proposer的Validator更新为：

–（超时处理）：当从敌手A接收到(timeout, T)消息时，如果T有效，增加Round，并选择新的提议者。

（三）功能

初始化：向发送(timeStart,)命令。若在任何阶段从收到(timeOver)消息，直接投票给nil块。

–当从验证者传入(Prevote, Proposal)消息时，

* 当收到Proposal时，向发送(, queryState)，获取PoLC。若锁定在上一轮Proposal，则签名并广播, B'为上一轮投票区块。否则广播当前轮Proposal，即签名并广播。
* 若未收到任何Proposal，则签名并广播。

–当从验证者传入(Precommit, Proposal)消息时，持续接收网络中的prevote投票，

* 若收到超过 的prevote投票，签名并广播 ，向发送(, unlock, B')解锁上一轮区块，然后向发送(, lock, B)锁定当前区块。
* 若收到超过 的空prevote投票，向发送(, unlock, ALL)释放所有锁定的区块。否则，不会锁定任何区块。

（四）功能

初始化：对于，设置，。表示Proposal是否已Commit。向发送(timeStart,)命令。若在任何阶段从收到(timeOver)消息，向发送(newRound)。

–当收到从验证者传入(Commit,Proposal)消息时，

* 持续接收网络中的precommit投票。若收到超过 的precommit投票，进入后续步骤。否则，向发送(newRound)，开启下一轮。
* Commit阶段：为区块B签名并广播commit投票 ，同时为其收集全网的commit投票。
* 若已为区块B广播commit投票且收集到超过的commit投票，则设置，设置commitTime为当前时间，向发送(newHeight)。
* 若为区块B收集到少于的commit投票，向发送(newRound)。
* 若在任何阶段收到超过的commit投票，立即进入Commit阶段。

–收到来自任意方的消息(request\_status)时：

返回集合C并指示区块B是否已完成。

（五）功能

初始化：设置Height := 0，Round := 0和PoLC := ⊥。

–当从任意验证者接收到(newHeight)消息时，

更新Height := Height+1并将Round重置为0，向发送(startProposal)消息。

–当从任意验证者接收到(newRound)消息时，

向发送(startProposal)消息。

–当从接收到(,lock,B)消息时，将加入到PoLC中 (Height,Round,B)对应的ValidatorSet中。

–当从接收到(,unlock,B)消息时，将在对应的PoLC中 (Height,Round,B)的ValidatorSet中删除。

–当从接收到(,unlock,ALL)消息时，则设置PoLC := ⊥。

–当从接收到(,queryState)消息时，返回PoLC。

（六）功能

初始化：设置， := ⊥。

–当从任意验证者接收到(timeStart,)请求时，将更新为 ← ，向验证者返回一个(timeOK)消息, 然后开始倒计时。

–当从任意验证者接收到(GetTime)请求时，它会将当前的返回给请求方。

–当从某一个时，它会向对应的验证者发送一个(timeOver)消息。

–当从任意验证者接收到(ResetTime)请求时，将重置为 ，向验证者返回一个(timeOK)消息。

**三、协议描述**

Tendermint-BFT协议通过轮次机制和投票阶段确保多个验证者之间就区块达成一致，并最终提交区块。该协议支持容忍少量恶意节点，依赖于消息广播、延迟处理和投票收集来实现共识。

–Party S:

**StartProposal**：开始共识，调用，选择并激活一个提议者Proposer。

–Party Proposer:

**Initialize**: 向发送(timeStart,)命令。若从收到(timeOver)消息，则直接跳转执行RoundOK部分。

**Input**: 从功能中接收并选择一个提案，确定其区块B有效后将其作为提议区块。

**Propose**: 将提议信息L(|Proposal|)发送给敌手A，然后签名并广播(Proposal)给验证者。

**RoundOK**: 调用更新轮次，重新选择提议者，开始新的轮次。

–Party Validator:

**Initialize**: 向发送自己的提案。

**Input**: 在收到来自Proposer的Proposal后，验证Proposal的完整性和有效性。

**Prevote**: 根据收到Proposal的，调用。

**Precommit**: 根据收到的Proposal，调用。若共识失败跳转执行RoundOK部分。

**Commit**: 根据收到的Proposal，调用。若共识失败跳转执行RoundOK部分。

**RoundOK**: 调用更新轮次，重新选择提议者，开始新的轮次。

|  |  |
| --- | --- |
| 图 2 Proposal数据结构 | 图 3 Vote数据结构 |

1. **理想功能**

### 理想功能

**网络延迟攻击**

**参数**：

* V：验证者集合
* ：最大超时时间
* ：理想时间功能
* ：理想广播功能

**符号解释**：

* ：当前区块高度，或当前正在执行的共识实例
* ：当前轮次
* ：当前轮次的共识阶段
* ：记录每个节点在不同区块高度下达成的最终共识值
* ：空参数

**当从** **接收到消息** ，**且** **时**：

1. 向 发送 并等待形式为 的响应。
2. 若 ：
   * 返回步骤 1。
3. 否则：
   * 从 中选出 。
   * 向 发送 并等待形式为 的响应。
   * 若 被破坏，则向 发送 。
   * 广播 。
   * 更新 。

**当从** **接收到消息** ，**且** **时**：

1. 向 发送 并等待形式为 的响应。
2. 若 ：
   * 广播 。
3. 否则：
   * 向 发送 。
   * 若 且尚未从 收到 ：
     + 广播 。
   * 否则，广播 。
   * 若收到来自 的 ，广播 。
4. 更新 。

**当从** **接收到消息** ，**且** **时**：

1. 向 发送 并等待形式为 的响应。
2. 若 ：
   * 广播 。
3. 否则：
   * 向 发送 。
   * 若 且尚未收到来自 的 ：
     + 广播 。
   * 否则，广播 。
   * 若收到来自 的 ，广播 。
4. 更新 。

**当从** **接收到消息** ，**且** **时**：

1. 向 发送 并等待形式为 的响应。
2. 若 ：
   * 广播 ，设置。
3. 否则：
   * 向 发送 。
   * 若 且尚未收到来自 的 ：
     + 广播 ，设置。
   * 否则，广播 ，设置。
   * 若收到来自 的 ，广播 。
4. 若，更新 。

**当从** **接收到消息** ，**且** **时**：

1. 若 ：
   * 设置 ，并将 。
   * 更新 且 。
2. **当前问题**

1.完善和适配 TBFT 协议的功能

针对 TBFT 协议，之前提出的功能设计需要进一步优化和完善，以确保它们能够适配并满足所有预期的功能需求。为了更准确地模拟 TBFT 协议的实际运行情况，有必要对现有功能进行调整和改进，确保其不仅符合协议的基本要求，还能有效支持协议的安全性和效率。为此，可以通过系统化的测试和验证来识别现有功能中的不足之处，并根据测试结果进行针对性的修改和优化。同时，借鉴其他类似协议的成功经验，结合实际应用场景的需求，逐步完善 TBFT 协议的各项功能。

2.确保 TBFT 协议的不可篡改性

关于 TBFT 协议的不可篡改性，我对“如何确保在共识达成之前，所有节点都无法提前得知最终的输出值”这一点存在疑问。根据我的理解，在 Propose 阶段，Proposer 会向全网广播包含 Block 字段的 Proposal。那么，当 Validator 接收到 Proposal 并读取其中的 Block 字段时，这是否意味着他们已经知道了最终的输出值。