**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**

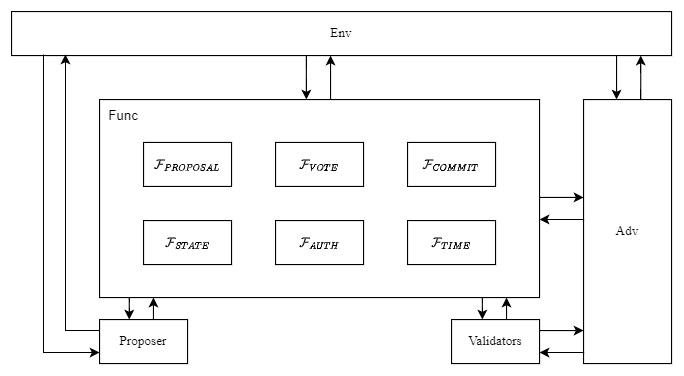


图 1 TBFT协议初步框架

**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是被腐化的，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**

TBFT的理想功能::

### 初始化

初始化以下变量：

* 当前共识实例的高度
* 当前轮次编号
* 当前共识流程的步骤
* 存储最终区块决策的数组
* 当前锁定的提议值（如果有）
* 锁定值的轮次 // 表示没有锁定
* 最后一个有效的提议值
* 最后一个有效提议的轮次

然后执行 ，开始第一轮次。

### 处理Propose消息

当从 接收到 ，且 ：

* 如果 ，那么则广播 ，否则广播 ，最后更新

### Pre-vote

当从 接收到 ，并收到 个 ，且 且 ：

* 如果 ，那么广播 ，否则广播 ，最后更新

### 处理Pre-vote结果

当首次接收到 个 时：

* 调度 在 后执行

### Pre-commit

接收到 ，并且有 个 ，且 ：

* 如果 ，则设置，，广播 ，更新
* 更新 和

接收到 个 ：

* 广播 并更新

### 处理Pre-commit消息

当首次接收到 个 时：

* 调度 在 后执行

### Commit

接收到 来自 ，且有 个 ，且 ：

* 如果 ，则确定决策 ，提升区块高度，开始下一个轮次 // 。然后重置 ，最后执行

### 启动下一轮

接收到 个 ，且 ： 启动新轮次，执行

### StartRound Function

函数 ：

* 如果 ：
  + 如果 ，则。否则。最后广播
* 否则：调度 在 后执行

### Timeout Functions

函数 ：

* 如果 ，则广播 ，然后更新

函数 ：

* 如果 ，则广播 ，然后更新

函数 ：

* 如果 ，则执行

1. **定理证明**

# 1、终止性证明（草稿）

**引理1** 算法 满足终止性。

**证明：**我们需要证明协议在GST(Global Stabilization Time) 之后的一个有限时间内，所有正确进程最终一定能够达成一致。

假设在时间 GST 时，最高启动的轮次是 ( )，并且存在一个正确的进程 ( p )，对于所有的正确进程 ( c )，有 ( )。进一步假设 ( p ) 将成为某个大于 ( ) 的轮次 ( ) 的提议者。

存在以下两种情况：

1. 对于所有轮次，没有正确的进程锁定任何值。

* 在这种情况下，当进入轮次时，通过下述第一种情况证明，所有正确的进程将在轮次内达成一致。

1. 对于所有轮次，存在一个正确的进程锁定了一个值（v）。

* 在这种情况下，假设是在之前的最高轮次，通过下述第二种情况证明，在轮次结束时，对于所有的正确的进程c，有 ( ) 和 ( )。因此，当进入轮次时，所有正确的进程将在轮次内达成一致。

### 第一种情况：没有正确的进程锁定任何值

1、初始化

首先初始化变量

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 变量含义 | 表示进程 ( c ) 最后一次发送非空 PRECOMMIT 消息的轮次。 | 表示进程 ( c ) 当前锁定的值。 | 表示进程 ( q ) 记录的最后一个可能的决策值所在的轮次。 | 表示进程 ( q ) 记录的最后一个可能的决策值。 |
| 更新时间 | 在发送 PRECOMMIT 消息之前，更新此变量 | 在发送 PRECOMMIT 消息之前，更新此变量 | 当接收到有效的 PROPOSAL 和对应的 2f+1 个 PREVOTE 消息时，更新此变量 | 当接收到有效的 PROPOSAL 和对应的 2f+1 个 PREVOTE 消息时，更新此变量 |

2、场景描述

我们假设下面的场景：

（a）进程 ( p ) 是第一个在时间 ( t > GST ) 进入轮次 ( r > 0 ) 的诚实进程

（b）轮次 ( r ) 的提议者是诚实进程 ( q )。

（c）对于每个诚实的进程 ( c )，在时间 ( t )内有 ( ) 。

（d）超时设置：（）

* Propose阶段超时设置为：
* Prevote阶段超时设置为：
* Precommit阶段超时设置为：

3、轮次r的具体进展

（a）**进入轮次 ( r )**：

* ( p ) 是首个进入轮次 ( r ) 的诚实进程，会在 ( ) 超时后，开始新轮次。
* 所有正确的进程将在 ( ) 时间前开始轮次 ( r )。

**（b）接收提案消息：**

* 提议者 ( q ) 在 ( ) 使用Propose功能发送 PROPOSAL 消息。
* 所有正确的进程最迟在 ( ) 收到此 PROPOSAL 消息，前提是 ( )。

**（c）预投票阶段 (PREVOTE)：**

* 在PREVOTE阶段，所有正确的进程将接受 PROPOSAL 消息并在 ( ) 时间内通过发送 PREVOTE 消息。
* 这些 PREVOTE 消息将在 ( ) 时间内被所有正确的进程收到。

**（d）预提交阶段 (PRECOMMIT)：**

* 在PRECOMMIT阶段，所有正确的进程将在 ( ) 时间内接收到 PROPOSAL 消息和 2f+1 PREVOTE 消息，然后通过发送 PRECOMMIT 消息。
* 这些 PRECOMMIT 消息将在 ( ) 时间内被所有正确的进程收到。
* 在 () 时间内，所有正确的进程都将收到 PROPOSAL 消息和 2f+1 PRECOMMIT 消息，然后通过正式提交区块。

最终，所有正确的进程将在轮次 ( r ) 内，在 ( ) 时间前达成一致，终止性得到证明。

**处理超时情况**

* **处理 timeoutPrevote(r)**：
  + 如果一个正确的进程 ( ) 是第一个触发 timeoutPrevote(r) 的进程，则意味着 ( ) 已经收到了 PROPOSAL 和一组 2f+1 PREVOTE 消息。
  + 通过 gossip 协议，所有正确的进程将在 ( ) 时间内收到这些消息，并在 ( ) 时间内发送自己的 PREVOTE 消息。
  + 因为 ( )，所以这确保了所有正确的进程在各自的本地 timeoutPrevote(r) 超时之前收到 PREVOTE 消息。
* **处理 timeoutPrecommit(r)**：
  + 如果一个正确的进程 ( ) 是第一个触发 timeoutPrecommit(r) 的进程，则意味着 ( ) 已经收到了至少 f+1 来自正确进程的 PRECOMMIT 消息。
  + 通过 gossip 协议，所有正确的进程将在 ( ) 时间内收到这些消息，并在 ( ) 时间内发送自己的 PRECOMMIT 消息。
  + 因为 ( )，所以这确保了所有正确的进程在各自的本地 timeoutPrecommit(r) 超时之前做出决定。

1. **当前问题**

1. TBFT协议满足性质的证明

对于 TBFT 协议，对于UC框架下的证明流程不熟悉，证明过程有点难。

2. TBFT终止性的第二种情况

TBFT终止性的第二种情况还需要时间去证明，后面将会首先完成此项证明。