**TBFT建模进度**

**摘要**

版本6：本文完善了TBFT共识机制的建模框架：优化协议框架图，细化理想功能，新增匿名通信信道理想功能，并初步完成协议的形式化建模。

1. **初步框架**

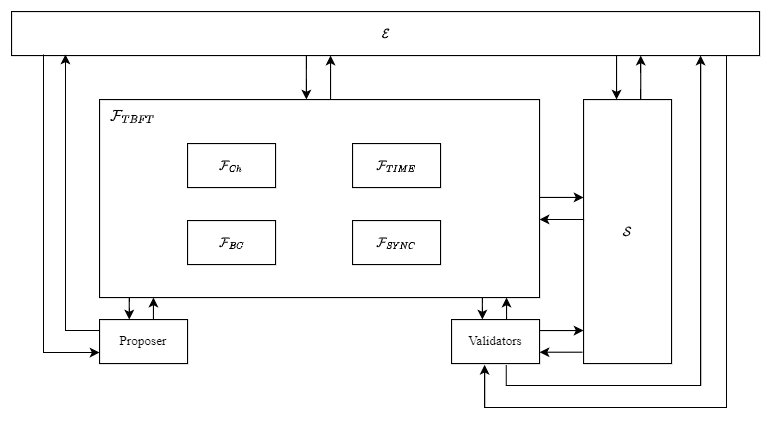


图 1 TBFT协议初步框架

**二、功能描述**

（一）功能

初始化：定义一组参与方，其中 和 分别表示该组中的两个参与方，作为消息 的发送方和接收方。

根据功能参数定义如下。消息标识符 由功能随机选择。

1. 当从 接收到输入 时，向 输出 。
2. 当从 接收到 时，向 发送 。

根据以下参数化函数设置 ：

– 对于 ，设置 , 。当从 接收到 时，向 发送 。

– 对于 ，设置 。

– 对于 ，设置 。

– 对于 ，设置 。

– 对于 ，设置 。当从 接收到 时，向 发送 。

– 对于 ，设置 。

1. 当从 接收到 时，向 发送 。当从 接收到 时，向 发送 。
2. 当从 接收到 时，向 输出 。当从 接收到 时，向 发送 。

a 这赋予了对手 更多的权力，因为 UC 模型中需要顺序发送消息， 决定发送方何时可以继续。

（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，设置Round = Round + 1，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，

* 若收到Proposal，则向发送(, queryState)，获取PoLC。
  + 查询PoLC，若锁定在上一轮Proposal，则签名并广播上一轮区块。
  + 否则，签名并广播当前轮区块。
* 否则，则签名并广播。

–当从验证者传入(Precommit, Proposal)消息时，

* 若收到超过2f+1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2f+1的空prevote投票，
  + 签名并广播 ，向发送(, unlock, ALL)释放所有锁定的区块。
* 否则，不锁定任何区块。

（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，

* 若收到超过2f+1的precommit投票，
  + 签名并广播，同时收集全网的commit投票。
  + 若已为区块B广播commit投票且收集到超过2f+1的commit投票，则设置，向验证者发送(allowCommit,Proposal)消息，向发送(newHeight)。
  + 否则，向验证者发送(denyCommit,Proposal)消息，向发送(newRound)。
* 否则，向发送(newRound)，开启下一轮。

–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。

–当从任意验证者接收到(newRound)消息时，

更新Round := Round +1。

–当从出块人Proposer接收到(getProposal, sid, , )消息时，

从配置文件中获取Proposals，然后将其返回给调用者。

–当从接收到 (updateProposal, sid, , Proposals)消息时，

将Proposals更新到配置文件中。

–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，设置PoLC := ⊥。

–当从接收到(,queryState)消息时，返回PoLC。

（六）功能

初始化：设置， := ⊥。

–当收到(GetTime)请求时，将当前的返回给请求方。

–当收到(ResetTime)请求时，

将重置为 ，向调用者返回一个(timeOK)消息。

–当收到(timeStart, sid, , )请求时，

将更新为 ，向理想功能返回一个(timeOK)消息, 然后开始倒计时。

–当从时，

会向对应的调用者发送一个(timeOver, sid, , )消息。

1. **理想功能**

# 功能

**参数：**

**符号说明：**

**接收到来自 的消息 ，当 时：**

1. 向 发送 ，并等待返回形式为 的响应。
2. 如果 ：
   * 返回第 1 步。
3. 否则：
   * 向 发送 ，并挂起执行。
   * 收到 返回的 后恢复执行。
   * 向 发送 。
   * 向 发送 ，并等待返回形式为 的响应。
   * 向 发送 ，并等待返回形式为 的响应。
   * 如果 被篡改，
     + 向 发送 。
   * 如果 并且没有收到来自 的 ：
     + 广播 。
   * 否则：
     + 返回第 1 步。
   * 更新 。

**接收到来自 的消息 ，当 时：**

1. 向 发送 ，并等待返回形式为 的响应。
2. 如果 ：
   * 广播 。
3. 否则：
   * 向 发送 ，并挂起执行。
   * 收到 返回的 后恢复执行。
   * 向 发送 。
   * 如果 且没有收到来自 的 ：
     + 广播 。
   * 否则：
     + 广播 。
4. 向 发送 。
5. 更新 。

**接收到来自 的消息 ，当 时：**

1. 设置 。
2. 如果 :
   * 广播 。
3. 否则：
   * 广播 。

**接收到来自 的消息 ，当 时：**

1. 设置 。
2. 发送 到 ，并等待返回 格式的响应。
3. 如果 ：
   * 广播 。
4. 否则：
   * 发送 到 ，并挂起执行。
   * 接收到来自 的消息 后，恢复执行。
   * 发送 到 。
   * 如果 且没有收到来自 的 消息：
     + 设置 , 。
     + 广播 。
     + 设置 , 。
   * 否则：
     + 广播 。
5. 更新 。

**接收到来自 的消息 ，当 时：**

1. 设置 。
2. 发送 到 ，并等待返回 格式的响应。
3. 如果 ：
   * 广播 ，并设置 。
4. 否则：
   * 发送 到 ，并挂起执行。
   * 接收到来自 的消息 后，恢复执行。
   * 发送 到 。
   * 如果 且没有收到来自 的 消息：
     + 广播 ，并设置 。
   * 否则：
     + 更新 和 。

**接收到来自 的消息 ，当 时：**

1. 设置 。
2. 如果 ：
   * 设置 ，并更新 , 。
3. 发送 到 。
4. 发送 到 ，接收其响应 ，
   * 如果 ：
     + 更新 ，并重置 。
     + 发送 到 。
   * 否则重新执行此步骤。

**接收到消息 时：**

1. 设置 。
2. 如果 ：
   * 发送 到 。

**四、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 Z:

**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。

## **协议**

|  | 提议者（Proposer） |  | 验证者（Validator） |  |
| --- | --- | --- | --- | --- |
|  | 1: 发送 给 |  |  |  |
|  | 2: 发送 给 |  |  |  |
|  | 3: 从 获取 |  |  |  |
|  | 4: 从提案（Proposals）中选择一个提案值 。 |  |  |  |
|  | 5: 发送 给 |  |  |  |
|  | 6: 如果 并且从 接收到消息 ，则调用 以开始新一轮。 |  |  |  |
|  | 7: 否则广播 |  | 8: 发送 给 |  |
|  |  |  | 9: 如果 ，则调用 |  |
|  |  |  | 10: 发送 给 |  |
|  |  |  | 11: 如果 ，则调用 |  |
|  |  |  | 12: 发送 给 |  |
|  |  |  | 13: 如果 ，则调用 |  |
|  |  |  | 14: 调用 以开始新一轮。 |  |