

TBFT 建模进度

摘要

本文档针对 TBFT 共识机制进行了初步建模。相较于之前建模进度，文档更新了框架结构图，增加了 Validator 与环境的交互；文档完善了功能描述，细化了对于理想功能 \mathcal{F}_{TBFT} 的描述，增加了必要的参数，并完善了参数说明。

一、初步框架

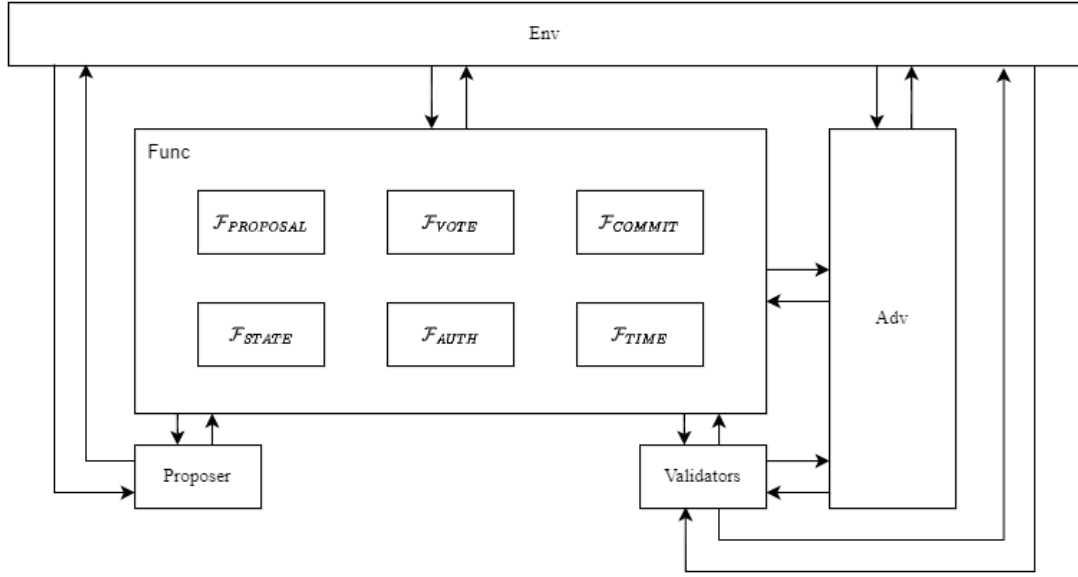


图 1 TBFT 协议初步框架

二、功能描述

(一) 功能 \mathcal{F}_{AUTH}

- 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 $\mathcal{F}_{PROPOSAL}$

初始化：设置 Proposal := \perp 和 Round := 0。

- 当收到消息(startProposal)时，
 - 通过 Round-robin 规则选定提议者 Proposer $\in H$ ，H 为 V 中诚实者的集合，
 - 初始化 Validator 的 votingPower 为其质押资金：
$$\text{votingPower}_i = \text{stake}_i, \quad \forall i \in \{1, \dots, N\}$$
 - 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：
$$\text{votingPower}_i \leftarrow \text{votingPower}_i + \text{stake}_i$$
 - 被选中的 Validator 更新为：

$$\text{votingPower}_i \leftarrow \text{votingPower}_i - \sum_{j \neq i} \text{stake}_j$$

- (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 $\mathcal{F}_{\text{VOTE}}$

初始化：向 $\mathcal{F}_{\text{TIME}}$ 发送(timeStart, δ)命令。若在任何阶段从 $\mathcal{F}_{\text{TIME}}$ 收到(timeOver)消息，直接投票给 nil 块。

- 当从验证者 $v_i \in V$ 传入(Prevote, Proposal)消息时，
 - 若收到 Proposal，则向 $\mathcal{F}_{\text{STATE}}$ 发送(v_i , queryState)，获取 PoLC。
 - 查询 PoLC，若 v_i 锁定在上一轮 Proposal，则签名并广播上一轮区块(v_i , prevote, Vote(B'))。
 - 否则，签名并广播当前轮区块(v_i , prevote, Vote(B))。
 - 否则，则签名并广播(v_i , prevote, Vote(nil))。
- 当从验证者 $v_i \in V$ 传入(Precommit, Proposal)消息时，
 - 若收到超过 $2f+1$ 的 prevote 投票，
 - 签名并广播 (v_i , precommit, Vote(B))，向 $\mathcal{F}_{\text{STATE}}$ 发送(v_i , unlock, B')解锁上一轮区块，然后向 $\mathcal{F}_{\text{STATE}}$ 发送(v_i , 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 (v_i , precommit, Vote(nil))，向 $\mathcal{F}_{\text{STATE}}$ 发送(v_i , unlock, ALL)释放所有锁定的区块。
 - 否则，不锁定任何区块。

(四) 功能 $\mathcal{F}_{\text{COMMIT}}$

初始化：对于 $v_i \in V$ ，设置 $c_i := 0$ ， $c_i \in C$ 。表示 Proposal 是否已 Commit。向 $\mathcal{F}_{\text{TIME}}$ 发送(timeStart, δ)命令。若在任何阶段从 $\mathcal{F}_{\text{TIME}}$ 收到(timeOver)消息，向 $\mathcal{F}_{\text{STATE}}$ 发送(newRound)。

- 当收到从验证者 $v_i \in V$ 传入(Commit, Proposal)消息时，
 - 若收到超过 $2f+1$ 的 precommit 投票，
 - 签名并广播(v_i , commit, Vote(B))，同时收集全网的 commit 投票。
 - 若 v_i 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 $c_i := 1$ ，设置 commitTime 为当前时间，向 $\mathcal{F}_{\text{STATE}}$ 发送(newHeight)。
 - 否则，向 $\mathcal{F}_{\text{STATE}}$ 发送(newRound)。
 - 否则，向 $\mathcal{F}_{\text{STATE}}$ 发送(newRound)，开启下一轮。
- 收到来自任意方 v_k 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 $\mathcal{F}_{\text{STATE}}$

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

- 当从任意验证者 $v_i \in V$ 接收到(newHeight)消息时，
 - 更新 Height := Height+1 并将 Round 重置为 0，向 $\mathcal{F}_{\text{PROPOSAL}}$ 发送(startProposal)消息。
- 当从任意验证者 $v_i \in V$ 接收到(newRound)消息时，
 - 向 $\mathcal{F}_{\text{PROPOSAL}}$ 发送(startProposal)消息。
- 当从 v_i 接收到(v_i , lock, B)消息时，
 - 将 v_i 加入到 PoLC 中 (Height, Round, B) 对应的 ValidatorSet 中。
- 当从 v_i 接收到(v_i , unlock, B)消息时，
 - 将 v_i 在对应的 PoLC 中 (Height, Round, B) 的 ValidatorSet 中删除。
- 当从 v_i 接收到(v_i , unlock, ALL)消息时，设置 PoLC := \perp 。

- 当从 v_i 接收到 $(v_i, \text{queryState})$ 消息时, 返回 PoLC。

(六) 功能 \mathcal{F}_{TIME}

初始化: 设置 $t_i \in T$, $t_i := \perp$ 。

- 当从任意验证者 $v_i \in V$ 接收到 $(\text{timeStart}, \delta)$ 请求时,
将 t_i 更新为 $t_i \leftarrow \delta$, 向验证者 v_i 返回一个 (timeOK) 消息, 然后开始倒计时。
- 当从任意验证者 $v_i \in V$ 接收到 (GetTime) 请求时, 将当前的 t_i 返回给请求方。
- 当从某一个 $t_i \in T$, $t_i = 0$ 时, 向对应的验证者 v_i 发送一个 (timeOver) 消息。
- 当从任意验证者 $v_i \in V$ 接收到 (ResetTime) 请求时,
将 t_i 重置为 $t_i := \perp$, 向验证者 v_i 返回一个 (timeOK) 消息。
- 当从理想功能 \mathcal{F}_{tbft} 接收到 $(\text{timeStart}, \text{sid}, \text{step}_p, \delta)$ 请求时,
将 t_{sid} 更新为 $t_{\text{sid}} \leftarrow \delta$, 向理想功能 \mathcal{F}_{tbft} 返回一个 (timeOK) 消息, 然后开始倒计时。
- 当从某一个 $t_{\text{sid}} \in T$, $t_{\text{sid}} = 0$ 时,
它会向理想功能 \mathcal{F}_{tbft} 发送一个 $(\text{timeOver}, \text{sid}, \text{step}_p, \delta)$ 消息。

三、理想功能 \mathcal{F}_{TBFT}

理想功能 $\mathcal{F}_{TBFT}^{V, \Delta, \Sigma}[\mathcal{F}_{TIME}, \mathcal{F}_{BB}, \mathcal{F}_{CLOCK}]$

参数:

- V : 验证者集合.
- Δ : 每个阶段的最大执行时间.
- Σ : 延迟攻击中的最大延迟.
- \mathcal{F}_{TIME} : 时间的理想功能.
- \mathcal{F}_{BB} : 广播的理想功能.
- \mathcal{F}_{CLOCK} : 同步的理想功能.

符号解释:

δ : 实际执行时间

σ : 实际延迟

h_p : 当前区块高度或正在执行的共识实例.

round_p : 当前轮次.

$\text{step}_p \in \text{propose, prevote, precommit, commit}$: 当前轮次的共识阶段.

$\text{count}_{\text{step}_p}$: 记录每个阶段投票的数量.

$\text{decision}_p[\]$: 记录各个节点在不同高度达成的最终共识值.

$\text{isVote}_{\text{COMMIT}}$: 标记提交阶段是否已完成投票.

$*$: 空参数.

当收到消息 $\langle \text{NEWROUND}, h_p, \text{round}_p, * \rangle$ 来自 \mathcal{E} , 且 $\text{step}_p = \text{propose}$ 时:

1. 发送 $\langle \text{Sleep}, \text{sid}, \text{step}_p \rangle$ 到 \mathcal{S} 并等待 $\langle \text{Wake}, \text{sid}, \text{step}_p, \delta, \sigma \rangle$ 的响应。
2. 如果 $(\delta + \sigma > \Delta) \vee \sigma > \Sigma$:
 - 返回第 1 步。
3. 否则:
 - 发送 $\langle \text{timeStart}, \text{sid}, \text{step}_p, \sigma \rangle$ 到 \mathcal{F}_{TIME} , 并暂停执行。
 - 收到来自 \mathcal{F}_{TIME} 的 $\langle \text{timeOver}, \text{sid}, \text{step}_p, \sigma \rangle$ 后恢复执行。
 - 发送 $\langle \text{timeStart}, \text{sid}, \text{step}_p, \delta \rangle$ 到 \mathcal{F}_{TIME} 。
 - 发送 $\langle \text{CreateProposal}, \text{sid}, \text{step}_p \rangle$ 到 \mathcal{S} 并等待 $\langle \text{StartProposal}, \text{sid}, \text{step}_p \rangle$ 的响应。
 - 发送 $\langle \text{StartProposal}, \text{sid}, \text{step}_p \rangle$ 到 $\text{Proposer}(h_p, \text{round}_p)$ 并等待 $\langle \text{PROPOSAL}, \text{sid}, \text{step}_p, v \rangle$ 的响应。

- 如果 $\text{Proposer}(h_p, \text{round}_p)$ 被破坏,
 - 发送 $\langle \text{Input}, \text{sid}, h_p, \text{round}_p, v \rangle$ 到 \mathcal{S} 。
- 如果 $\text{valid}(v)$ 且尚未收到来自 $\mathcal{F}_{\text{TIME}}$ 的 $\langle \text{timeOver}, \text{sid}, \text{step}_p, \delta \rangle$:
 - 广播 $\langle \text{PROPOSAL}, h_p, \text{round}_p, v \rangle$ 。
- 否则:
 - 返回第 1 步。
- 发送 $\langle \text{RoundOK} \rangle$ 到 $\mathcal{F}_{\text{CLOCK}}$ 。
- 发送 $\langle \text{RequestRound} \rangle$ 到 $\mathcal{F}_{\text{CLOCK}}$ 并接收其响应 d_i ,
 - 如果 $d_i = 0$, 更新 $\text{step}_p \leftarrow \text{prevote}$ 。
 - 否则重新执行此步骤。

在接收到来自 $\text{Validator}(h_p, \text{round}_p)$ 的消息 $\langle \text{PREVOTE}, h_p, \text{round}_p, \text{id}(v) \rangle$ 且 $\text{step}_p = \text{precommit}$ 时:

1. 设置 $\text{count}_{\text{prevote}} \leftarrow \text{count}_{\text{prevote}} + 1$
2. 发送 $\langle \text{Sleep}, \text{sid}, \text{step}_p \rangle$ 到 \mathcal{S} 并等待形式为 $\langle \text{Wake}, \text{sid}, \text{step}_p, \delta, \sigma \rangle$ 的响应。
3. 如果 $(\delta + \sigma > \Delta) \vee \sigma > \Sigma$:
 - 广播 $\langle \text{PRECOMMIT}, h_p, \text{round}_p, \text{nil} \rangle$ 。
4. 否则:
 - 发送 $\langle \text{timeStart}, \text{sid}, \text{step}_p, \sigma \rangle$ 到 $\mathcal{F}_{\text{TIME}}$ 并暂停执行。
 - 在接收到来自 $\mathcal{F}_{\text{TIME}}$ 的 $\langle \text{timeOver}, \text{sid}, \text{step}_p, \sigma \rangle$ 后恢复执行。
 - 发送 $\langle \text{timeStart}, \text{sid}, \text{step}_p, \delta \rangle$ 到 $\mathcal{F}_{\text{TIME}}$ 。
 - 如果 $\text{valid}(v) \wedge (\text{count}_{\text{prevote}} > 2f + 1)$ 且尚未从 $\mathcal{F}_{\text{TIME}}$ 收到 $\langle \text{timeOver}, \text{sid}, \text{step}_p, \delta \rangle$:
 - 广播 $\langle \text{PRECOMMIT}, h_p, \text{round}_p, \text{id}(v) \rangle$ 。
 - 否则:
 - 广播 $\langle \text{PRECOMMIT}, h_p, \text{round}_p, \text{nil} \rangle$ 。
5. 向 $\mathcal{F}_{\text{CLOCK}}$ 发送 $\langle \text{RoundOK} \rangle$ 。
6. 向 $\mathcal{F}_{\text{CLOCK}}$ 发送 $\langle \text{RequestRound} \rangle$ 并接收响应 d_i ,
 - 如果 $d_i = 0$, 更新 $\text{step}_p \leftarrow \text{commit}$ 。
 - 否则重新执行此步骤。

在接收到来自 $\text{Validator}(h_p, \text{round}_p)$ 的消息 $\langle \text{PRECOMMIT}, h_p, \text{round}_p, \text{id}(v) \rangle$ 且 $\text{step}_p = \text{commit}$ 时:

1. 设置 $\text{count}_{\text{precommit}} \leftarrow \text{count}_{\text{precommit}} + 1$
2. 发送 $\langle \text{Sleep}, \text{sid}, \text{step}_p \rangle$ 到 \mathcal{S} 并等待形式为 $\langle \text{Wake}, \text{sid}, \text{step}_p, \delta, \sigma \rangle$ 的响应。
3. 如果 $(\delta + \sigma > \Delta) \vee \sigma > \Sigma$:
 - 广播 $\langle \text{COMMIT}, h_p, \text{round}_p, \text{nil} \rangle$, 并设置 $\text{isVote}_{\text{COMMIT}} \leftarrow \text{false}$ 。
4. 否则:
 - 发送 $\langle \text{timeStart}, \text{sid}, \text{step}_p, \sigma \rangle$ 到 $\mathcal{F}_{\text{TIME}}$ 并暂停执行。
 - 在接收到来自 $\mathcal{F}_{\text{TIME}}$ 的 $\langle \text{timeOver}, \text{sid}, \text{step}_p, \sigma \rangle$ 后恢复执行。
 - 发送 $\langle \text{timeStart}, \text{sid}, \text{step}_p, \delta \rangle$ 到 $\mathcal{F}_{\text{TIME}}$ 。
 - 如果 $\text{valid}(v) \wedge (\text{count}_{\text{precommit}} > 2f + 1)$ 且尚未从 $\mathcal{F}_{\text{TIME}}$ 收到 $\langle \text{timeOver}, \text{sid}, \text{step}_p, \delta \rangle$:
 - 广播 $\langle \text{COMMIT}, h_p, \text{round}_p, \text{id}(v) \rangle$, 并设置 $\text{isVote}_{\text{COMMIT}} \leftarrow \text{true}$ 。
 - 否则:
 - 向 $\mathcal{F}_{\text{CLOCK}}$ 发送 $\langle \text{RoundOK} \rangle$ 。
 - 向 $\mathcal{F}_{\text{CLOCK}}$ 发送 $\langle \text{RequestRound} \rangle$ 并接收响应 d_i ,
 - 如果 $d_i = 0$, 更新 $\text{step}_p \leftarrow \text{propose}$ 且 $\text{round}_p \leftarrow \text{round}_p + 1$ 。

- 否则重新执行此步骤。

在接收到来自 Validator(h_p, round_p) 的消息 $\langle \text{COMMIT}, h_p, \text{round}_p, \text{id}(v) \rangle$ 且 $\text{step}_p = \text{commit}$ 时:

1. 设置 $\text{count}_{\text{commit}} \leftarrow \text{count}_{\text{commit}} + 1$
2. 如果 $\text{valid}(v) \wedge (\text{count}_{\text{commit}} > 2f + 1) \wedge \text{isVote}_{\text{COMMIT}}$:
 - 设置 $\text{decision}_p[h_p] = v$, 并设置 $\text{isVote}_{\text{COMMIT}} \leftarrow \text{false}$ 。
 - 更新 $h_p \leftarrow h_p + 1$, 设置 $\text{step}_p \leftarrow \text{propose}$ 并重置 $\text{round}_p, \text{count}_{\text{step}_p}$ 。

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 E:

StartProposal: 开始共识, 调用 $\mathcal{F}_{\text{PROPOSAL}}$, 选择并激活一个提议者 Proposer。

– Party Proposer:

Initialize: 向 $\mathcal{F}_{\text{TIME}}$ 发送 $(\text{timeStart}, \delta)$ 命令。若从 $\mathcal{F}_{\text{TIME}}$ 收到 (timeOver) 消息, 则直接跳转执行 RoundOK 部分。

Input: 从功能 $\mathcal{F}_{\text{STATE}}$ 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 $L(|\text{Proposal}|)$ 发送给敌手 A, 然后签名并广播 (Proposal) 给验证者。

RoundOK: 调用 $\mathcal{F}_{\text{PROPOSAL}}$ 更新轮次, 重新选择提议者, 开始新的轮次。

– Party Validator:

Initialize: 向 $\mathcal{F}_{\text{STATE}}$ 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 $\mathcal{F}_{\text{VOTE}}(\text{Prevote}, \text{Proposal})$ 。

Precommit: 根据收到的 Proposal, 调用 $\mathcal{F}_{\text{VOTE}}(\text{Precommit}, \text{Proposal})$ 。若共识失败跳转执行 RoundOK 部分。

Commit: 根据收到的 Proposal, 调用 $\mathcal{F}_{\text{COMMIT}}(\text{Commit}, \text{Proposal})$ 。若共识失败跳转执行 RoundOK 部分。

RoundOK: 调用 $\mathcal{F}_{\text{PROPOSAL}}$ 更新轮次, 重新选择提议者, 开始新的轮次。