**TBFT Modeling Progress**

**Abstract**

Version 7: The modeling framework for the TBFT consensus mechanism has been refined.

1.The broadcast ideal functionality has been enhanced to ensure reliable transmission of information within the designated set and visibility to adversaries.

2.The functionality description of ​ has been improved by incorporating a random transaction-removal feature, making the system more flexible and secure when processing transactions.

3.The protocol description ​ has been detailed, with added descriptions of sub-functionalities, providing clearer guidance for protocol implementation.

4. The completion rate of UC modeling achieved by the protocol is about 80%.

**1. Overall Framework**

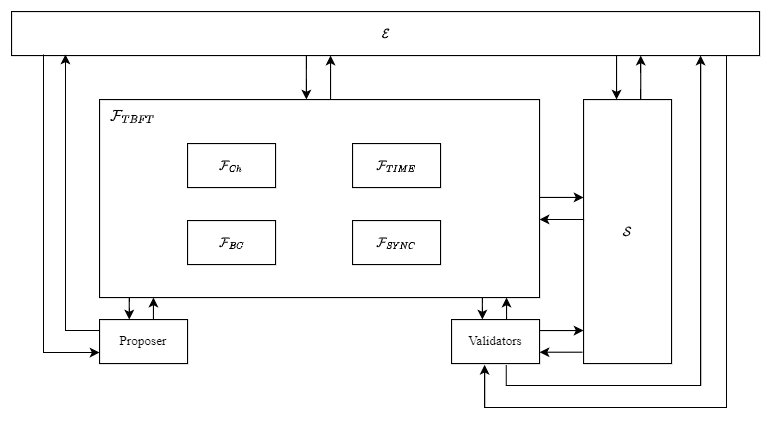


Figure 1: Overall Framework of the TBFT Protocol

**2. Function Description**

### a). Functionality

Initialization: Let define a set of parties where and denote two parties of the set as the sender and receiver of a message respectively.

is defined as follows based on parameters of functionality. Message identifier is selected freshly by the functionality.

1. Upon input from , output to .
2. Upon receiving from , send to .

Set based on the following parameterized functions:

– for set , . Upon receiving from , send to .

– for set .

– for set .

– for set .

– for set . Upon receiving from , send to .

– for set .

1. Upon receiving from , send to .

* Upon receiving from , send to .

1. Upon receiving from , output to .

* Upon receiving from , send to .

### b). Functionality

Initialization: Broadcast functionality parameterized by the set proceeds as follows:

Upon receiving from a party , send to all entities in the set and to .

### c). Functionality

Initialization: Set Proposal := ⊥and Round := 0。

–Upon receiving the message (startProposal)

* Select the proposer Proposer ∈ H through the Round-robin rule, where H is the set of honest validators in V.
  + Initialize the voting power of the validator to its staked funds:
  + Elect the Proposer in turn according to the Round-robin rule, and update Round := Round + 1.
* Update votingPower:
  + For validators not selected:
  + For the selected validator:

–(Timeout handling): Upon receiving the message (timeout, T) from the adversary A, if T is valid, set Round = Round + 1 and select a new proposer.

### d). Functionality

Initialization: Send the (timeStart, ) command to . If a (timeOver) message is received from at any stage, vote for the nil block directly.

–Upon receiving the message (Prevote, Proposal) from validator ,

* If a Proposal is received, send (, queryState) to to obtain the PoLC.
  + Query the PoLC. If v\_i is locked on the Proposal from the previous round, sign and broadcast the previous round's block .
  + Otherwise, sign and broadcast the current round's block.
* Otherwise, sign and broadcast .

–Upon receiving the message (Precommit, Proposal) from validator,

* If more than 2f+1 prevote votes are received,
  + Sign and broadcast , send (, unlock, B') to to unlock the previous round's block, then send (, lock, B) to to lock the current block.
* If more than 2f+1 null prevote votes are received,
  + Sign and broadcast , send (, unlock, ALL) to to release all locked blocks.
* Otherwise, do not lock any blocks.

### f). Functionality

Initialization: For , set ，, indicating whether the Proposal has been committed. Send the (timeStart,) command to . If a (timeOver) message is received from at any stage, send (newRound) to .

–Upon receiving the message (Commit, Proposal) from validator ,

* If more than 2f+1 precommit votes are received,
  + Sign and broadcast , and collect commit votes from the entire network.
  + If has already broadcast a commit vote for block B and has collected more than 2f+1 commit votes, set , send (allowCommit, Proposal) to the validator, and send (newHeight) to .
  + Otherwise, send (denyCommit, Proposal) to the validator and send (newRound) to .
* Otherwise, send (newRound) to to start the next round.

–Upon receiving the message (request\_status) from any party :

* Return the set C and indicate whether block B has been completed.

### g). Functionality

Initialization: Set Height := 0, Round := 0, and PoLC := ⊥.

–Upon receiving the message (newHeight) from any validator

Update Height := Height + 1 and reset Round to 0.

–Upon receiving the message (newRound) from any validator ,

Update Round := Round + 1.

–Upon receiving the message (getProposal, sid, , \*) from the proposer,

Retrieve the Proposals from the configuration file and return them to the caller.

–Upon receiving the message (updateProposal, sid, , Proposals),

Update the Proposals in the configuration file.

–Upon receiving the message (, lock, B) from ,

Add to the ValidatorSet corresponding to (Height, Round, B) in the PoLC.

–Upon receiving the message (, unlock, B) from ,

Remove from the ValidatorSet corresponding to (Height, Round, B) in the PoLC.

–Upon receiving the message (, unlock, ALL) from ,

Set PoLC := ⊥.

–Upon receiving the message (, queryState) from ,

Return the PoLC.

### h). Functionality

Initialization: Set ， := ⊥.

–Upon a (GetTime) request is received,

Return the current to the requester.

–Upon a (ResetTime) request is received,

Reset to := ⊥, and return a (timeOK) message to the caller.

–Upon a (timeStart, sid, , ) request is received,

Update to , return a (timeOK) message to the ideal functionality , and then start the countdown.

–Upon ,

Send a (timeOver, sid, , ) message to the corresponding caller.

1. **Ideal Functionality**

# Functionality

**Parameters**:

**Symbol Explanation:**

* : Total number of nodes in the validator set.

### Upon receiving message from , while :

1. Send to and wait for a response of the form .
2. If :
   * Return to step 1.
3. Otherwise:
   * Send to , and suspend execution.
   * Upon receiving from , resume execution.
   * Send to .
   * Send to and wait for a response of the form .
   * Send to and wait for a response of the form .
   * If is corrupted,
     + Send to .
   * If and no has been received from :
     + Broadcast .
   * Otherwise:
     + Return to step 1.
   * Update .

### Upon receiving message from , while :

1. Send to and wait for a response of the form .
2. If :
   * Broadcast .
3. Otherwise:
   * Send to , and suspend execution.
   * Upon receiving from , resume execution.
   * Send to .
   * If and no has been received from :
     + Broadcast and wait for a response .
     + If :
       - Broadcast and receive .
       - Broadcast .
     + Otherwise:
       - If :  
         Broadcast .
       - Otherwise:  
         Broadcast .
   * Otherwise:
     + Broadcast .
4. Send to .
5. Update .

### Upon receiving message from , while :

1. For each :
   * Set .
2. If there exists a such that :
   * Broadcast t to remove the transaction from the transaction pool.
   * Reset .

### Upon receiving message from , while :

1. Set .
2. If :
   * Broadcast .
3. Otherwise:
   * Broadcast .

### Upon receiving message from , while :

1. Set .
2. Send to and wait for a response of the form .
3. If :
   * Broadcast .
4. Otherwise:
   * Send to , and suspend execution.
   * Upon receiving from , resume execution.
   * Send to .
   * If and no has been received from :
     + Set , .
     + Broadcast .
     + Set , .
   * Otherwise:
     + Broadcast .
5. Update .

### Upon receiving message from , while :

1. Set .
2. Send to and wait for a response of the form .
3. If :
   * Broadcast , and set .
4. Otherwise:
   * Send to , and suspend execution.
   * Upon receiving from , resume execution.
   * Send to .
   * If and no has been received from :
     + Broadcast , and set .
   * Otherwise:
     + Update and .

### Upon receiving message from , while :

1. Set .
2. If :
   * Set , and update , .
3. Send to .
4. Send to , receive its response :
   * If :
     + Update and reset
     + Send to .
   * Otherwise re-execute this step.

### Upon receiving message :

1. Set .
2. If :
   * Send to .
3. **Protocol Description**

The Tendermint-BFT protocol ensures consensus among multiple validators on a block and eventually commits the block through a round-based mechanism and voting phases. The protocol supports tolerance of a small number of malicious nodes and relies on message broadcasting, delay handling, and vote collection to achieve consensus.  
– **Party Z**:

**StartProposal**: Initiates consensus by calling , selecting and activating a proposer (Proposer).

– **Party Proposer**:

**Initialize**: Sends the command to . Upon receiving the message from , directly proceeds to execute the RoundOK phase.

**Input**: Receives and selects a proposal from the function. After validating the proposal's block as valid, uses it as the proposed block.

**Propose**: Sends proposal information to adversary A, then signs and broadcasts to the validators.

**RoundOK**: Calls to update the round, reselect the proposer, and start a new round.  
– **Party Validator**:  
**Initialize**: Sends their own proposal to .

**Input**: Upon receiving the proposal from the Proposer, validates the proposal’s integrity and validity.

**Prevote**: Based on the received proposal, calls .

**Precommit**: Based on the received proposal, calls . If consensus fails, proceeds to the RoundOK phase.

**Commit**: Based on the received proposal, calls . If consensus fails, proceeds to the RoundOK phase.

**RoundOK**: Calls to update the round, reselect the proposer, and start a new round.

## **The Protocol**

|  | Proposer |  | Validator |  |
| --- | --- | --- | --- | --- |
|  | 1: Send to |  |  |  |
|  | 2: Send to |  |  |  |
|  | 3 : Get from |  |  |  |
|  | 4: Select a Proposal value from the Proposals. |  |  |  |
|  | 5: Send to |  |  |  |
|  | 6: If and get from , then call |  |  |  |
|  | 7: Otherwise:  broadcast |  |  |  |
|  |  |  | 8: Send to |  |
|  |  |  | 9: If , then call |  |
|  |  |  | 10: Send to |  |
|  |  |  | 11: If , then call |  |
|  |  |  | 12: Send to |  |
|  |  |  | 13: If , then call |  |
|  | Output to |  | 14 : If get from  Call |  |
|  | Output to |  | 15 : Otherwise: Call |  |

## **The Functionality**

| or Validators |  |  |
| --- | --- | --- |
| 1: Send to |  |  |
|  |  | 2: Send to |
|  |  | 3 : Get from |
|  |  | 4 : Each validator initializes their : |
|  |  | 5 : The validator is chosen as the proposer: |
|  |  | 6 : Send to |
|  |  | 7 : For unselected validators : |
|  |  | 8 : For the selected proposer : |
|  |  | 9 : Update based on |
|  |  | 10 : Send to |

## **The Functionality**

|  | Validator |  |  |
| --- | --- | --- | --- |
| **Prevote** | 1: Send to |  |  |
|  |  |  | 2 : Send to |
|  |  |  | 3 : If no is received from :  Sign and broadcast |
|  |  |  | 4 : Otherwise, If :  Send to to get PoLC |
|  |  |  | 5 : If is locked on Proposal from the previous round:  Sign and broadcast |
|  |  |  | 6 : Otherwise:  Sign and broadcast |
|  |  |  | 7 : If no :  Sign and broadcast |
| **Precommit** | 1: Send to |  |  |
|  |  |  | 2 : Send to |
|  |  |  | 3 : If no is received from :  Sign and broadcast |
|  |  |  | 4 : Otherwise, If :  Upon receiving more than prevote votes:  Sign and broadcast |
|  |  |  | 5 : Send to |
|  |  |  | 6 : Send to |
|  |  |  | 7 : Upon receiving more than null prevote votes:  Sign and broadcast |
|  |  |  | 8 : Send to |
|  |  |  | 9 : If no , Do not lock any blocks. |

## **The Functionality**

|  | Validator |  |  |
| --- | --- | --- | --- |
| **Commit** | 1: Send to |  |  |
|  |  |  | 2 : Send to |
|  |  |  | 3 : If no is received from :  Send to |
|  |  |  | 4 : Otherwise, If : If more than precommit votes are received:  Sign and broadcast |
|  |  |  | 5 : Collect commit votes from the network. |
|  |  |  | 6 : If node has already broadcast a commit vote for and collected more than commit votes:  Send to Validator |
|  |  |  | 7 : Send to |
|  |  |  | 8 : Otherwise: Send to Validator |
|  |  |  | 9 : Send to |

## **The Functionality**

|  | Proposer |  |  |
| --- | --- | --- | --- |
| **Get Proposals** | 1: Send to |  |  |
|  |  |  | 2: Get from Configuration File |
|  |  |  | 3: Send to Proposer |
|  |  |  |  |
| **Set Proposals** | 1 : Send to |  |  |
|  |  |  | 2: Update to Configuration File |
|  |  |  |  |
| **GetPoLC** | 1 : Send to |  |  |
|  |  |  | 2 : Return PoLC to the caller |
|  |  |  |  |
| **UnLock** | 1 : Send to |  |  |
|  |  |  | 2 : Delete from the ValidatorSet corresponding to in PoLC |
|  | 1 : Send to |  |  |
|  |  |  | 2 : Set PoLC := ⊥ |
| **Lock** | 1 : Send to |  |  |
|  |  |  | 2 : Add to the ValidatorSet corresponding to in PoLC. |
|  |  |  |  |
| **NewHeight** | 1 : Send to |  |  |
|  |  |  | 2 : Set , |
|  |  |  |  |
| **NewRound** | 1 : Send to |  |  |
|  |  |  | 2 : Set |

## **The Functionality**

|  | Proposer or Validator |  |  |
| --- | --- | --- | --- |
| **Countdown** | 1: Send to |  |  |
|  |  |  | 2: Set , start the countdown. Return |
|  |  |  | 3 : When and :  to the caller |
| **ResetTime** | 1: Send to |  |  |
|  |  |  | 2 : Set , stop the countdown Return |
| **GetTime** | 1: Send to |  |  |
|  |  |  | 2 : Return to the caller |