**摘要**

### 建模版本8：1、将广播功能F\_bc嵌入理想功能具体流程中，并且对同步切换视图进行了修正，实现了所有节点同步进入下一视图。2、协议实现的UC建模完成度比例为90%左右

证明版本3：通过假设敌手控制有限数量的节点，并且消息延迟有界，尝试证明所有正确副本在规定时间内必定达成共识，且环境无法区分真实协议执行与理想功能模拟之间的差异。文中定义了敌手模型及其行为，比较了现实世界和理想世界中协议的执行过程，涵盖了消息传递、超时检测、视图切换及QC生成等环节以满足证明终止性的条件。

**目录**

**[Hotstuff协议UC建模 2](#_Toc26815)**

**[一、 整体框架图： 2](#_Toc21595)**

**[二、 功能描述 2](#_Toc22636)**

**[三、 理想功能 5](#_Toc22710)**

**[四、 协议描述 11](#_Toc20180)**

**[HotStuff协议UC终止性证明 16](#_Toc29242)**

**[一、敌手模型与行为定义 16](#_Toc30393)**

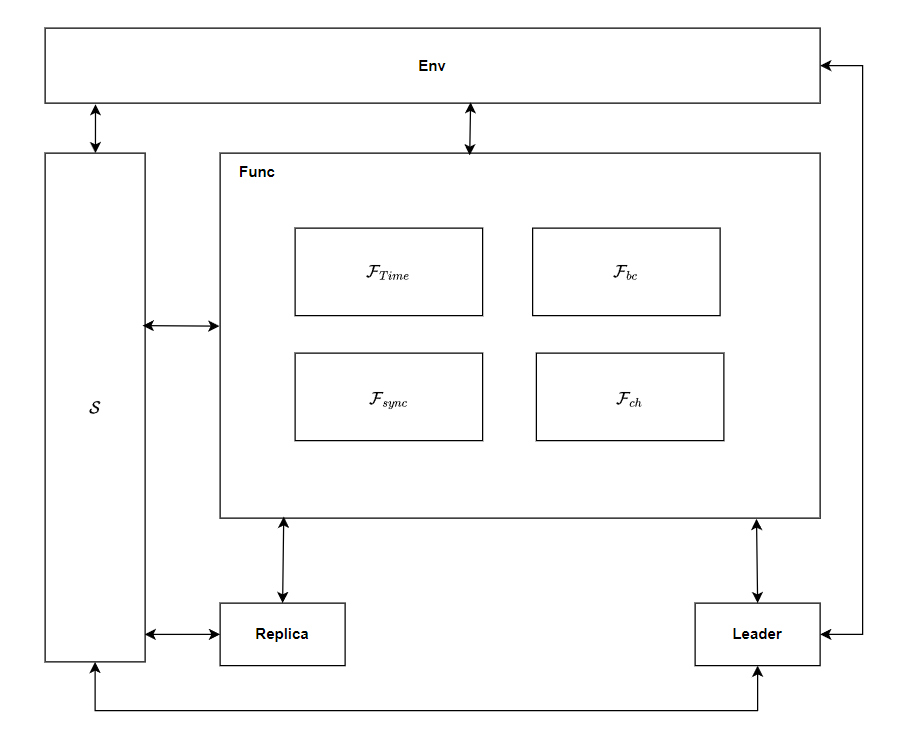
**[二、协议行为对应（现实世界 vs 理想功能） 16](#_Toc23894)**

**[三、模拟器设计 17](#_Toc19619)**

**[四、不可区分性证明（Game Hopping） 18](#_Toc22270)**

# Hotstuff协议UC建模

## 整体框架图：



## 功能描述

### F\_{Proposal}

初始化：设置。

–当收到消息时，

* 选取这些消息中最高的prepareQC最为highQC：
* 在highQC的节点的叶子上写入客户指令，提出新的提案B：

-将提案B、highQC封装在MSG中广播给replica：

### F\_{Vote} 初始化：设置。

–当收到来自的消息时，

* 先检查m是否与自己状态匹配：
* 检查叶子节点是否是本地lockedQC对应节点后继以及QC是否比本地lockedQC对应节点的视图更高：
* 在highQC的节点的叶子上写入客户指令，提出新的提案B：

-将投票信息结点m、自己的部分签名封装在VOTEMSG中发送给leader。

-当收到来自的消息时，

* 先检查QC是否与自己状态匹配：
* 如果决定投票且type是PREPARE阶段，更新本地状态：
* 如果决定投票且type是PRE-COMMIT阶段，更新本地状态：

-将投票信息m.justify.node、自己的部分签名封装在VOTEMSG中发送给leader：

### F\_{QC}

初始化：设置。

-当收到2f+1条投票消息时：

* 先检查m是否与自己状态匹配：
* 收集replica的投票，把部分签名组合：

-将QC封装在MSG中广播给replica：

### F\_{TIME}

初始化：设置，。

–当从任意replica 接收到请求时，将更新为 ← ，向replica 返回一个消息, 然后开始倒计时。

–当从某一个时，它会向对应的replica 发送一个消息。

1. **F\_{Next\_view}**

初始化：设置，。

–当从任意replica m收到 next\_view请求MSG(⊥，m,prepareQC)时，将更新为，将prepareQC更新为。

–将、封装在NEW-VIEW message中发送给下一视图的leader。

1. **F\_{Ch}**

我们定义一组参与方，其中 和 表示该集合中的两个参与方，分别是消息 的发送方和接收方。

是基于功能的参数进行定义的。消息标识符 由功能新选定。

在接收到来自 的输入 时，输出 给 。

在接收到来自 的 后，向发送 。

根据以下参数化函数设置 ：

– 对于 设置 , 。在接收到来自 的 后，向 发送 。

– 对于 设置 。

– 对于 设置 。

– 对于 设置 。

– 对于 设置 。在接收到来自 的 后，向 发送 。

– 对于 设置 。

在接收到来自 的 后，向 发送 。

在接收到来自 的 后，向 发送 。

在接收到来自 的 后，输出 给 。

在接收到来自 的 后，向 发送 。

1. **F\_{bc}**

广播功能 由集合 参数化，具体流程如下：

在接收到来自某方 的 后，向集合 中的所有实体以及 发送 。

## 理想功能

**Functionality**

**Network Delay Attack**

**Parameters**:

* : Replica Set
* ：the Maximum timeout duration
* : the Maximum timeout duration of Replica within current view.
* : Ideal functionality for timing.
* : Ideal functionality for broadcast.
* : Ideal functionality for synchronization.

**Symbol Explanation:**

#### **Upon receiving message** from :

1. .
2. If is corrupted:

* Send to .

1. Send to wait for a response of the form :
   * Set which .

#### **Upon receiving message** from :

1. Send to and wait for a response of the form :
   * If :
     + set
     + send to and .
     + Send to .
     + Send to ,receive its response .
       - If ,set .
       - else re-execute this step.
   * else,Send to .
2. If :
   * Set .
   * If ,Send to and wait for a response of the form :
     + Create.
     + If no is received from :
       - send to .
       - Set .
     + Else:
       - set .
       - send to and .
       - Send to .
       - Send to ,receive its response .
         * If ,set .
         * else re-execute this step.
3. Else，ignore this message.

#### **Upon receiving message** from :

1. Send to and wait for a response of the form :
   * If :
     + set
     + send to and .
     + Send to .
     + Send to ,receive its response .
       - If ,set .
       - else re-execute this step.
   * else,Send to .
2. If :
   * If the is higher than , update
   * If and no has been received from :
     + Send to .
     + Send to and wait for a response of the form :
       - Set .
   * else:
     + set .
     + send to and .
     + Send to .
     + Send to ,receive its response .
       - If ,set .
       - else re-execute this step.
3. Else，ignore this message.

#### **Upon receiving message** from :

1. Send to and wait for a response of the form :
   * If :
     + set
     + send to and .
     + Send to .
     + Send to ,receive its response .
       - If ,set .
       - else re-execute this step.
   * else,Send to .
     + If :
     + Set
     + If :
     + Create.
     + If no is received from :
       - send to .
       - Send to and wait for a response of the form :
         * Set .
     + Else:
       - set .
       - send to and .
       - Send to .
       - Send to ,receive its response .
         * If ,set .
         * else re-execute this step.
2. Else，ignore this message.

#### **Upon receiving message** from :

1. Send to and wait for a response of the form :
   * If :
     + set .
     + send to and .
   * else,Send to .
2. If :
   * If no has been received from :
     + Set .
     + Send to .
     + Send to and wait for a response of the form :
       - Set .
   * else:
     + set .
     + send to and .
     + Send to .
     + Send to ,receive its response .
       - If ,set .
       - else re-execute this step.
3. Else，ignore this message.

#### **Upon receiving message** from :

1. Send to and wait for a response of the form :
   * If :
     + set
     + send to and .
     + Send to .
     + Send to ,receive its response .
       - If ,set .
       - else re-execute this step.
   * else,Send to .
     + If :
     + Set
     + If :
     + Create.
     + If no is received from :
       - send to .
       - Send to and wait for a response of the form :
         * Set .
     + Else:
       - set .
       - send to and .
       - Send to .
       - Send to ,receive its response .
         * If ,set .
         * else re-execute this step.
2. Else，ignore this message.

#### **Upon receiving message** from :

1. Send to and wait for a response of the form :
   * If :
     + set .
     + send to and .
   * else,Send to .
2. If :
   * If no has been received from :
     + Set .
     + Send to .
     + Send to and wait for a response of the form :
       - Set .
   * else:
     + set .
     + send to and .
     + Send to .
     + Send to ,receive its response .
       - If ,set .
       - else re-execute this step.
3. Else，ignore this message.

#### **Upon receiving message** from :

1. Send to and wait for a response of the form :
   * If :
     + set
     + send to and .
     + Send to .
     + Send to ,receive its response .
       - If ,set .
       - else re-execute this step.
   * else,Send to .
     + If :
     + Set
     + If :
     + Create.
     + If no is received from :
       - send to .
       - send to and .
       - Send to .
       - Send to ,receive its response .
         * If ,set .
         * else re-execute this step.
     + Else:
       - set .
       - send to and .
       - Send to .
       - Send to ,receive its response .
         * If ,set . set .
         * else re-execute this step.
2. Else，ignore this message.

#### **Upon receiving message** from :

1. Send to and wait for a response of the form :
   * If :
     + set .
     + send to and .
   * else,Send to .
2. If :
   * If no has been received from :
     + Execute new commands through
     + Send to .
     + Send to ,receive its response .
       - If ,set ,set .
       - else re-execute this step.
   * else:
     + set .
     + send to and .
     + Send to .
     + Send to ,receive its response .
       - If ,set .
       - else re-execute this step.
3. Else，ignore this message.

## 协议描述

**The protocol**

|  | Leader |  | Replica |  |
| --- | --- | --- | --- | --- |
|  | 1: Send to |  |  |  |
|  | 2: Send to ,receive |  |  |  |
|  | 3: Send to |  |  |  |
|  | 4: If a message is received from ,Send to |  |  |  |
|  | 5: Else,broadcast to Replicas |  |  |  |
|  | 6: Send to |  |  |  |
|  |  |  | 7: Send to |  |
|  |  |  | :If a message is received from ,Send to |  |
|  |  |  | 8: Else,if ,call |  |
|  | 9: Send to |  |  |  |
|  | 10: Call |  |  |  |
|  |  |  | 11: Send to |  |
|  |  |  | 12: Set |  |
|  |  |  | 13: Call |  |
|  | 14: Send to |  |  |  |
|  | 15: Call |  |  |  |
|  |  |  | 16:Send to |  |
|  |  |  | 17: Set |  |
|  |  |  | 18: Call |  |
|  | 19: Send to |  |  |  |
|  | 20: Call |  |  |  |
|  | 21: Send to |  |  |  |
|  |  |  | 22: Send to |  |
|  |  |  | 23: If a message is received from ,Send to |  |
|  |  |  | 24: Else,execute new commands through |  |
|  |  |  | 25: Send to |  |
|  |  |  |  |  |

### **The Functionality**

| REPLICA |  |  |
| --- | --- | --- |
|  |  | Initialize: |
| 1: Send to |  |  |
|  |  | 2: set ,return |
|  |  | 3: When |
|  |  | 4: Send |

**The Functionality**

|  | Replica |  |  |
| --- | --- | --- | --- |
| Prepare | 1: Send to |  |  |
|  |  |  | 2 : Send to |
|  |  |  | 3 : If no is received from : send to Leader |
|  |  |  |  |
| Precommit | 1: Send to |  |  |
|  |  |  | 2 : Send to |
|  |  |  | 3 : If no is received from : send to Leader |
| Commit | 1: Send to |  |  |
|  |  |  | 2 : Send to |
|  |  |  | 3 : If no is received from : send to Leader |

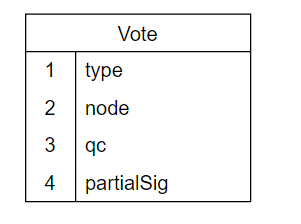
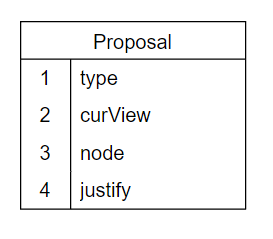
### **The Functionality**

|  | Leader |  |  |
| --- | --- | --- | --- |
| **PreCommit** | 1: Send to |  |  |
|  |  |  | 2: Send to |
|  |  |  | 3: If no is received from and more than prepare votes are received: broadcast |
| **Commit** | 1: Send to |  |  |
|  |  |  | 2: Send to |
|  |  |  | 3: If no is received from and more than precommit votes are received: broadcast |
| **Decide** | 1: Send to |  |  |
|  |  |  | 2: Send to |
|  |  |  | 3: If no is received from and more than commit votes are received: broadcast |

### **The Functionality**

| Leader/REPLICA |  |  |
| --- | --- | --- |
| 1: Send to |  |  |
|  |  | 2: Return n-f NEW-VIEW messages which |
| 1. Send to |  |  |
|  |  | 2: Set ,return |

相关数据结构：



# HotStuff协议UC终止性证明

## 定理（UC终止性）

在HotStuff协议中，假设敌手最多控制个节点（包括可能的领导者），且消息延迟有界（），则存在模拟器，使得对于任意环境，以下成立：

**终止性**：所有正确副本在时间内达成共识并输出相同的值。

**不可区分**：环境无法区分真实协议与理想功能的执行。

# **一、敌手模型与行为定义**

### 敌手能力

1. **控制节点**：
   * 敌手控制个节点，可包括**领导者（Leader）**或**普通副本（Replica）**。
   * 被控制的节点称为**腐败节点**，其余为**诚实节点**。
2. **攻击行为**：
   * **延迟消息**：腐败节点可延迟其发送或转发的消息，最大延迟为。
   * **偏离协议**：腐败节点可发送无效提案（Proposal）、拒绝投票（Vote）或生成非法QC。
3. **限制**：
   * 敌手无法永久阻止消息传递（消息最终必达）。
   * 敌手无法伪造合法签名（假设存在不可伪造的签名机制）。

# **二、协议行为对应（现实世界 vs 理想功能）**

### 1. 领导者行为

#### 腐败领导者（被敌手控制）

* **Prepare阶段**：
  + 发送无效提案（如不包含前序QC哈希）。
  + 故意延迟广播提案，触发超时视图切换。
* **PreCommit/Commit阶段**：
  + 不广播QC消息或生成非法QC。
* **理想功能模拟**：
  + 检测到无效提案后，广播超时信息给所有节点，调用触发同步视图切换。
  + 腐败领导者的恶意行为被限制为延迟而非破坏协议逻辑。

#### 诚实领导者（未被控制）

* **Prepare阶段**：
  + 发送合法提案，包含前序QC哈希。
  + 在时间内广播提案。
* **理想功能模拟**：
  + 调用BROADCAST(Proposal)正常广播提案。
  + 通过同步超时检测。

### 2. 普通副本行为

#### 腐败副本（被敌手控制）

* **投票阶段**：
  + 拒绝发送Vote消息。
  + 发送无效Vote（如签名不匹配）。
* **理想功能模拟**：
  + 标记腐败副本，其投票不计入QC生成。
  + 腐败副本的恶意行为被过滤。

#### 诚实副本（未被控制）

* **投票阶段**：
  + 收到合法提案后，发送签名正确的Vote消息。
* **理想功能模拟**：
  + 监控中的合法投票，调用GENERATE-QC生成QC。

### 3.敌手延迟攻击场景分类与影响

#### 场景1：（不触发超时）

* **行为分析**：  
   - 消息延迟未超过当前视图超时阈值。  
   - 所有正确副本在时间内收到消息，正常推进共识。
* **共识流程**：  
   1. **Prepare阶段**：诚实领导者发送提案，副本在内响应，生成PrepareQC。  
   2. **PreCommit阶段**：广播PreCommit消息，内生成PreCommitQC。  
   3. **Commit阶段**：同理生成CommitQC，最终提交决策。
* **时间上界**：（Prepare+PreCommit+Commit）。

#### 场景2：（触发超时）

* **行为分析**：  
   - 至少一个正确副本因消息未达触发超时，发起视图切换。  
   - 根据附件二规则，翻倍：。
* **视图切换流程**：  
   1. 副本广播NEW-VIEW消息，递增视图号。  
   2. 新视图中超时阈值更新为。

**递归约束**：  
 - 敌手最多触发次恶意视图切换。  
 - 第次视图必由诚实节点主导，且。

# **三、模拟器设计**

### 1. 消息传递模拟

* **接口调用**： Send(m),broadca(m)
* **操作**：
  + 当副本发送消息时，调用send(m)。
  + 按时间戳排序消息，若，在时间后调用broadca(m)

### 2. 超时检测与视图切换

* **接口调用**：TIMEOUT(view), NEW-VIEW(view)
* **操作**：
  + 通过检测到时：
    1. 调用TIMEOUT(curView)。
    2. 递增视图：curView := curView + 1。
    3. 更新超时参数：。
    4. 调用NEW-VIEW(curView)广播视图切换消息。

### 3. QC生成与阶段推进

* **接口调用**：GENERATE-QC(phase), ADVANCE-PHASE()
* **操作**：
  + 当合法投票数时，调用GENERATE-QC(phase)生成QC。
  + 调用ADVANCE-PHASE()进入下一阶段，确保QC链式链接。

## 终止性证明

#### 综合时间上界

#### 分项验证

1. **视图切换时间**：
   * 最多次切换，总时间为。
2. **最终诚实视图共识时间**：
   * Prepare/PreCommit/Commit阶段各耗时，总计。
3. **全局时间上界**：

* **结论：**在时间内，所有正确副本必然达成共识。

# **四、不可区分性证明（Game Hopping）**

### 1. 消息传递阶段（映射1）

#### 真实世界行为

* 腐败节点发送消息，可能延迟。
* 诚实节点按协议在时间内广播消息。

#### 理想世界模拟

* 调用BROADCAST(m)，将加入队列。
* 计算交付时间：

#### 不可区分性计算

* **消息时序一致性**：
  + 对任意消息，观测到的时间差为：
  + 若敌手尝试施加的延迟 **未超过时间上界**（即 ），则 ，因此 。这意味着真实世界与理想世界的消息延迟差值为零，环境 无法观察到任何差异。
    - 若敌手试图施加**超过上界的延迟**（即 ），则 。然而，这种情况被协议主动排除：超时机制会触发视图切换，腐败节点的消息将被标记为过期或无效，敌手的延迟攻击被中止。
* **通信内容一致性**：
  + 腐败节点的无效消息被过滤（通过签名验证），诚实消息内容完全一致。
  + 引入变量对延迟时间和超时时间进行记录，此修改存储其他信息，但不会改变通信，因此无法区分。

### 2. 超时检测与视图切换（映射2）

#### 真实世界行为

* 若副本在时间内未收到消息，触发超时并广播NEW-VIEW。
* 腐败节点可能伪造超时事件。

#### 理想世界模拟

* 通过检测到。
* 调用递增视图：curView := curView + 1。
* 更新超时参数：。
* 调用NEW-VIEW(curView)，广播视图切换消息，调用实现同步视图切换。

#### 不可区分性

* 视图切换次数、时序、消息内容在两种世界中完全一致
* 无法获取节点本地超时检测和达成视图同步的内部状态，仅能观测网络消息

### 3. QC生成与阶段推进（映射3）

#### 真实世界行为

* QC生成需满足个合法投票，且链式哈希链接。
* 腐败节点可能发送无效投票。

#### 理想世界模拟

* 监控中的投票，若合法数，调用GENERATE-QC。
* 腐败节点的无效投票被丢弃。

#### 不可区分性计算

* **QC生成条件**：
  + 真实世界合法投票数：
  + 理想世界合法投票数：
  + 由于，有，与真实世界一致。
  + 消息计数逻辑与真实协议一致，无法察觉监控行为
* **哈希链一致性**：
  + 真实世界的QC哈希链接由协议规则保证。
  + 理想世界中，强制所有QC必须包含前序哈希，否则拒绝生成。