**摘要**

### **建模版本8：**

1.本文展示了Hotstuff共识机制的建模框架。协议实现的 UC 建模

完成度比例为 100%。

**证明版本6：**

1. 新增高并发场景下的安全性分析，涵盖提案竞争消解、阶段提交原子性、网络调度公平性等，通过数学公式（如分叉概率计算）严格证明协议在并发环境下的安全性。
2. 新增Leader与k个副本同时被腐蚀的场景分析，引入投票有效性绑定、阶段锁机制及QC高度单调性的数学约束，覆盖更复杂的攻击模型。
3. 细化时间计算，新增关键参数对照表，强化时间复杂度分析，证明时间上界t的最优性，补充公式推导。

**目录**

**[摘要 1](#_Toc12983)**

**[Hotstuff协议UC建模 2](#_Toc5942)**

[一、 整体框架图： 2](#_Toc5823)

[二、 功能描述 2](#_Toc31114)

[三、 理想功能 5](#_Toc19534)

[四、 协议描述 11](#_Toc6480)

**[HotStuff协议终止性证明 16](#_Toc4390)**

[定理声明（UC终止性） 16](#_Toc7286)

[一、协议特性与敌手模型 16](#_Toc10383)

[二、分场景终止性证明 16](#_Toc7683)

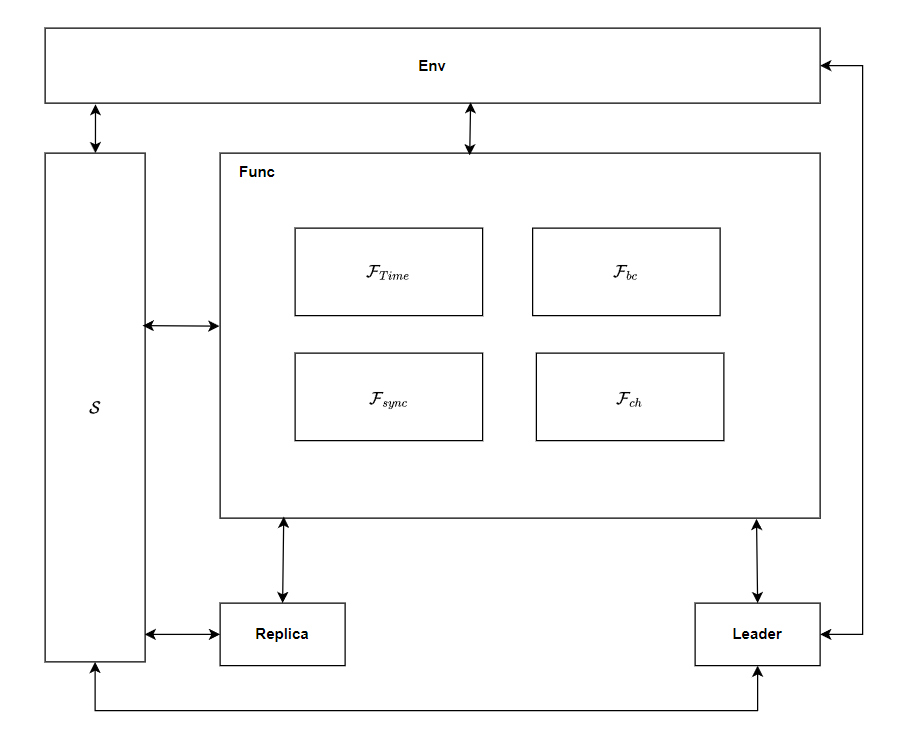
[三、全局时间分析 26](#_Toc2277)

[四、UC不可区分性证明 27](#_Toc6328)

[五、结论 29](#_Toc5148)

# 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 |

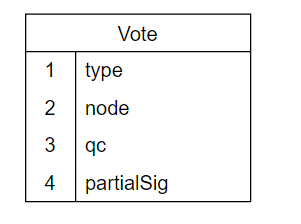
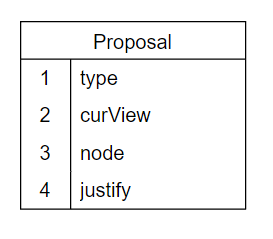
### **The Functionality**

|  | 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协议终止性证明

## 定理声明（UC终止性）

**引理1 HotstuffUC终止性定理：**

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：

1. **终止性**：所有正确副本在时间内达成共识并输出相同的值。
2. **不可区分**：环境无法区分真实协议与理想功能的执行和输出结果。(详细证明在最后一节）

## 一、协议特性与敌手模型

### 1.1 Hotstuff核心机制

* **超时轮次切换**：超时参数。
* **投票规则**：三阶段投票（Prepare/Precommit/Commit），需票推进阶段。
* **锁定机制**：通过和保证共识推进和防止回滚。

### 1.2敌手能力与接口行为

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、分场景终止性证明

## 1. 敌手延迟（未触发超时）

### 1.1模拟Leader节点行为

#### 1.1.1 只有Leader节点被Corrupted

##### 1.1.1.1 现实世界执行

* **敌手行为**：
  + 控制当前视图的Leader节点，通过延迟发送消息，但保证。
  + 发送无效提案（如不包含前序QC哈希），但被诚实副本过滤。
* **协议流程**：
  1. **Prepare阶段**：腐败Leader在调用广播提案，诚实副本在内接收。
  2. **PreCommit阶段**：诚实副本在内生成个，调用形成。
  3. **Commit阶段**：诚实副本在内生成个，调用形成。

##### 1.1.1.2 理想世界模拟

* **模拟器操作**：
  1. **消息截获**：当腐败Leader调用广播无效提案时，调用FILTER(msg)丢弃该提案。
  2. **超时同步**：若诚实副本未收到合法提案，通过触发TIMEOUT事件，具体操作包括：
     + 递增当前视图编号：；
     + 更新超时参数：；
     + 广播至和新Leader；
     + 通过发送和，接收响应后确认视图切换是否有效。
     + （但在此场景中不会触发）
  3. **QC生成**：对合法提案，调用GENERATE-QC(Prepare)生成。

##### 1.1.1.3 不可区分性分析

* **现实世界**：
  + **协议执行过程**：
    - 腐败Leader在发送提案（），诚实副本在前接收。
    - 诚实副本在内生成，并在内生成。
    - 所有消息在时间内交付，未触发超时。
  + **环境视角**：
    - 观察到提案在前被接收。
    - 各阶段QC（PrepareQC、PreCommitQC）在前生成。
    - 最终决策在提交。
* **理想世界**：
  + **协议执行过程**：
    - 模拟器过滤腐败Leader的无效提案，通过同步超时检测。
    - 合法提案的QC生成严格遵循投票规则。
    - 各阶段消息在时间内交付。
  + **环境视角**：
    - 观察到与真实世界相同的提案接收时间和QC生成时序。
    - 腐败Leader的无效提案被静默过滤，无额外可观测痕迹。
* **现实世界与理想世界的对比**：
  + **时间一致性**：
    - 提案交付、QC生成和决策提交时间在两种世界中均为。
    - 消息延迟被时间窗口完全覆盖。
  + **输出一致性**：
    - 环境观测到的QC哈希链、投票计数及最终决策完全一致。
    - 腐败Leader的无效操作在理想功能中被完美模拟为网络延迟。

##### 1.1.1.4 时间计算

1. **Prepare阶段**：  
    - 敌手Leader在发送提案（）。  
    - 诚实副本在前接收提案。  
    - **时间消耗**：（最大等待时间）。
2. **PreCommit阶段**：  
    - 诚实副本在内生成。  
    - **时间消耗**：。
3. **Commit阶段**：  
    - 诚实副本在内生成。  
    - **时间消耗**：。

**总时间**：（与无攻击一致）。

##### 1.1.1.5 高并发场景下的安全性分析

在高并发环境下，即使敌手控制Leader并在Δ时间内延迟提案，由于诚实副本在严格的时间隔离窗口内能够及时接收到提案并验证其合法性（例如检查前序QC哈希），无效提案会被迅速过滤，从而确保只有合法提案进入共识流程；基于2f+1投票规则生成的QC具有唯一性，保证了即使多个并发操作同时进行，共识决策仍能在内稳定达成。攻击与模拟模型在任意并发情况下讨论，各类攻击的 case 可能并发存在，均可进行并发式模拟，以抵御对应敌手行为。并且，本协议与其他长安链系统协议并发使用时，仍能维持安全性。

### 1.1.2 Leader节点和个副本同时被Corrupted

#### 1.1.2.1 现实世界执行

* **敌手行为**：
  + 控制当前视图的Leader节点及个副本（），通过延迟发送消息（）。
  + 腐败副本在Prepare/PreCommit阶段延迟发送或发送无效签名，但保证。
* **协议流程**：
  1. **Prepare阶段**：
     + 腐败Leader在广播提案，诚实副本在内接收。
     + 腐败副本在内延迟发送投票，诚实副本仍需收集个合法投票以生成QC。
  2. **PreCommit阶段**：
     + 诚实副本在内生成个，结合腐败副本的个延迟投票，调用生成。
  3. **Commit阶段**：
     + 同理，诚实副本在内生成个，形成。

#### 1.1.2.2 理想世界模拟

* **模拟器操作**：
  1. **消息过滤**：
     + 当腐败Leader广播无效提案时，调用的FILTER(msg)丢弃。
     + 腐败副本的无效投票被自动过滤。
  2. **延迟调度**：
     + 对腐败节点，调用设置延迟。
  3. **QC生成**：
     + 当合法投票数达时，调用GENERATE-QC(phase)生成QC，确保与真实协议一致。

#### 1.1.2.3 不可区分性分析

* **现实世界**：
  + **协议执行过程**：
    - 腐败Leader和副本的延迟行为被限制在窗口内，诚实副本仍可在完成各阶段。
    - 无效提案和投票被协议功能过滤，不影响QC生成。
  + **环境视角**：
    - 观察到提案和投票的时序与无攻击场景一致，QC链按预期递增。
* **理想世界**：
  + **协议执行过程**：
    - 模拟器通过和过滤无效操作，严格同步时间窗口。
    - QC生成逻辑与真实世界完全一致（需合法签名）。
  + **环境视角**：
    - 无法区分腐败节点的延迟行为与网络波动。
  + **现实世界与理想世界的对比**：
    - **时间一致性**：
      * 各阶段耗时均为，总时间。
      * 腐败节点的被窗口吸收。
    - **输出一致性**：
      * 环境观测到的QC哈希链、视图切换事件与合法提案提交时间无差异。

#### 1.1.2.4 时间计算

1. **Prepare阶段**：
   * 腐败Leader在发送提案，诚实副本在前接收。
   * **时间消耗**：。
2. **PreCommit阶段**：
   * 诚实副本在内收集票，生成。
   * **时间消耗**：。
3. **Commit阶段**：
   * 同理，生成耗时。
   * **总时间**：。

#### 1.1.2.5 高并发场景下的安全性分析

在这种情况下，敌手不仅控制了Leader节点，还同时控制了k个普通副本，可能在提案和投票阶段同时进行干扰。在高并发场景下，恶意Leader可能利用其控制权延迟或发送不合法的提案，而受控的k个副本则可能在Prepare、PreCommit或Commit阶段故意延迟或注入无效投票；然而，HotStuff协议通过严格的提案合法性校验（例如验证前序QC哈希）、基于票数生成QC的唯一性机制以及严格的时间隔离（Δ窗口）有效过滤掉超时或不合法的消息，从而确保只有在规定时间内收集到足够诚实副本的有效投票后，才会生成唯一且合法的QC，进而维护共识状态的一致性和链的正确性，即便面对同时来自提案和投票端的协同攻击，也能保障系统的安全性。攻击与模拟模型在任意并发情况下讨论，各类攻击的case可能并发存在，均可进行并发式模拟，以抵御对应敌手行为。并且，本协议与其他长安链系统协议并发使用时，仍能维持安全性。

### 1.2 普通副本被Corrupted（敌手控制个副本）

#### 1.2.1 现实世界执行

* **敌手行为**：
  + 控制个副本，在Prepare/PreCommit阶段延迟发送，但保证。
  + 发送无效投票（如签名错误），但被过滤。
* **协议流程**：
  1. **Prepare阶段**：诚实Leader广播合法提案，诚实副本在内响应，所有节点应该在内接收。诚实节点在时间内调用，腐败节点将在时间内调用，收集票后进入Precommit。
  2. **PreCommit阶段**：诚实节点在时间内调用 ，腐败节点将在时间内调用 收集票后进入Commit。
  3. **Commit阶段**：诚实节点在时间内调用 ，腐败节点将在时间内调用 收集票后进入Decide。

#### 1.2.2 理想世界模拟

* **模拟器操作**：
  1. **投票过滤**：通过自动丢弃腐败副本的无效投票。
  2. **投票调度**：对控制的节点，调用设置延迟。
  3. **QC生成**：当合法投票数达时，调用GENERATE-QC(phase)生成QC。

#### 1.2.3 不可区分性分析

* **现实世界**：
  + **协议执行过程**：
    - 诚实Leader在广播提案，腐败副本在内延迟发送投票。
    - 诚实副本收集合法投票生成QC，超时未被触发。
    - 所有阶段在内完成。
  + **环境视角**：
    - 观察到腐败副本的投票延迟在窗口内完成。
    - 各阶段QC按预期生成，无超时事件。
* **理想世界**：
  + **协议执行过程**：
    - 模拟器通过过滤无效投票，并调度腐败副本的延迟投票。
    - QC生成条件与真实协议一致（合法签名）。
    - 消息交付时间通过严格同步。
  + **环境视角**：
    - 投票计数和时序与真实世界无差异。
    - 腐败副本的延迟行为被映射为理想功能内的合法操作。
* **现实世界与理想世界的对比**：
  + **时间一致性**：
    - Prepare/PreCommit/Commit阶段分别耗时，总时间。
    - 腐败副本的延迟被窗口吸收。
  + **输出一致性**：
    - 环境观测到的QC内容、视图编号及阶段切换事件完全一致。
    - 无效投票的过滤逻辑在两种世界中不可区分。

#### 1.2.4 时间计算

1. **Prepare阶段**：
   * 诚实Leader在发送提案，所有副本在前接收。
   * **时间消耗**：。
2. **PreCommit阶段**：
   * 腐败副本在发送投票（）。
   * 投票收集完成时间仍为。
   * **时间消耗**：。
3. **Commit阶段**：
   * 同理，投票收集在完成。
   * **时间消耗**：。

**总时间**：（延迟被Δ覆盖）。

#### 1.2.5 高并发场景下的安全性分析

在这种情况下，虽然敌手控制了部分副本并尝试延迟或发送无效投票，但其延迟仍在Δ内，使得诚实节点能在规定时间内收集到足够的有效投票；因此，在高并发场景下，通过严格的提案合法性验证与基于票数达标生成QC的机制，确保了QC的唯一性与正确性，同时利用时间隔离机制有效防止了并发消息冲突，最终共识能如期完成。攻击与模拟模型在任意并发情况下讨论，各类攻击的 case 可能并发存在，均可进行并发式模拟，以抵御对应敌手行为。并且，本协议与其他长安链系统协议并发使用时，仍能维持安全性。

## 2. 敌手延迟（触发超时）

### 2.1 Leader节点被Corrupted

#### 2.1.1 现实世界执行

* **敌手行为**：
  + 控制Leader节点，在Prepare阶段延迟提案至，触发超时。
  + 强制副本调用NEW-VIEW进入下一视图。
* **协议流程**：
  1. **视图切换**：副本检测超时后递增，更新。
  2. **新视图共识**：新Leader（可能诚实）在超时窗口内完成共识。

#### 2.1.2 理想世界模拟

* **模拟器操作**：
  1. **超时触发**：当在时间未收到足够投票时，广播超时信息并且发送给，调用TIMEOUT(curView)，具体操作包括：
  + 递增当前视图编号：；
    - 更新超时参数：；
    - 广播至和新Leader；
    - 通过发送和，接收响应后确认视图切换是否有效。
  1. **视图递增**：递增并广播NEW-VIEW消息，通过同步所有副本。

1. **参数更新**：，与真实协议严格一致。

#### 2.1.3 不可区分性分析

* **现实世界**：
  + **协议执行过程**：
    - 腐败Leader延迟提案至，触发超时。
    - 副本通过次视图切换（总时间）进入诚实视图。
    - 最终视图在时间内完成共识。
  + **环境视角**：
    - 观察到超时事件及视图切换请求按指数退避规则递增。
    - 最终决策在时提交。
* **理想世界**：
  + **协议执行过程**：
    - 模拟器通过检测超时，严格递增视图和。
    - 新视图的Prepare/PreCommit/Commit阶段超时设为。
    - QC生成和消息广播通过同步。
  + **环境视角**：
    - 视图切换次数、超时参数更新与真实世界完全一致。
    - 最终决策时间与公式严格匹配。
* **现实世界与理想世界的对比**：
  + **时间一致性**：
    - 每次视图切换的超时时间和总时间完全同步。
    - 最终诚实视图的三阶段耗时在两种世界中相同。
  + **输出一致性**：
    - 环境无法通过视图切换日志或QC链检测差异。
    - 敌手的超时触发行为被理想功能完美模拟为网络延迟。

#### 2.1.4 时间计算

1. **Prepare阶段超时**：
   * 敌手延迟提案至。
   * **首次超时时间**：。
2. **视图切换**：
   * 第次切换的超时时间：。
   * **总切换时间**：。
3. **最终诚实视图**：
   * Prepare/PreCommit/Commit阶段各耗时。
   * **总时间**：。

**全局时间上界**：

#### 2.1.5 高并发场景下的安全性分析

当敌手控制的Leader故意将提案延迟超过Δ时，会触发超时并迫使系统进行视图切换；在高并发场景下，严格的时间隔离和超时机制不仅屏蔽了超时前的无效提案，而且通过上一视图中合法投票累计所形成的QC确保了历史状态的唯一性，虽然整体共识延迟增加到，但依然保障了系统在多视图并发情况下的安全性和一致性。攻击与模拟模型在任意并发情况下讨论，各类攻击的 case 可能并发存在，均可进行并发式模拟，以抵御对应敌手行为。并且，本协议与其他长安链系统协议并发使用时，仍能维持安全性。

### 2.2 普通副本被Corrupted（敌手控制个副本）

#### 2.2.1 现实世界执行

* **敌手行为**：
  + 控制个副本，在Prepare/PreCommit阶段延迟投票至，触发超时。
  + 发送无效NEW-VIEW消息干扰视图同步。
* **协议流程**：
  1. **超时触发**：诚实副本因未收到足够投票触发TIMEOUT。
  2. **视图切换**：需至少个NEW-VIEW消息（含个伪造消息+1个诚实节点超时）。

#### 2.2.2 理想世界模拟

* **模拟器操作**：
  1. **伪造消息处理**：验证NEW-VIEW消息的合法性，仅同步包含诚实节点的请求。
  2. **QC同步**：当进入新视图时，调用ADVANCE-PHASE()重置阶段，确保与真实协议一致。

#### 2.2.3 不可区分性分析

* **现实世界**：
  + **协议执行过程**：
    - 腐败副本延迟投票至，触发次视图切换。
    - 最终诚实Leader在超时窗口内完成各阶段QC生成。
  + **环境视角**：
    - 观察到超时事件由腐败副本的投票缺失引起。
    - 视图切换后的QC链包含前序哈希，确保状态一致性。
* **理想世界**：
  + **协议执行过程**：
    - 模拟器丢弃无效的NEW-VIEW消息，仅同步合法请求。
    - 通过强制重置阶段变量，确保与真实协议状态同步。
  + **环境视角**：
    - 视图切换请求的过滤逻辑与真实网络行为不可区分。
    - 最终QC的生成时间和内容与真实世界一致。
* **现实世界与理想世界的对比**：
  + **时间一致性**：
    - 每次视图切换的耗时和总时间严格匹配。
    - 腐败副本的延迟攻击被限制在指数退避框架内。
  + **输出一致性**：
    - 环境观测到的视图切换次数、QC哈希链及提交时间完全一致。
    - 理想功能的状态重置操作与现实协议的本地变量更新无差异。

#### 2.2.4 时间计算

1. **Prepare阶段超时**：
   * 腐败副本延迟投票至。
   * **首次超时时间**：（等待提案+投票超时）。
2. **视图切换**：
   * 第次切换的超时时间：。
   * **总切换时间**：。
3. **最终诚实视图**：
   * Prepare/PreCommit/Commit阶段各耗时。
   * **总时间**：。

**全局时间上界**：

#### 2.2.5 高并发场景下的安全性分析

在此情形中，敌手控制的f个副本延迟投票超过Δ，导致部分节点超时并触发视图切换；在高并发环境下，系统通过严格的时间隔离机制及时过滤掉延迟和无效的投票，并且只在规定时间窗口内收集有效票，从而确保了生成QC的唯一性和提案的合法性，同时视图切换和指数退避策略有效避免了并发视图间的冲突，保障了最终共识的安全达成。攻击与模拟模型在任意并发情况下讨论，各类攻击的 case 可能并发存在，均可进行并发式模拟，以抵御对应敌手行为。并且，本协议与其他长安链系统协议并发使用时，仍能维持安全性。

## 3. 敌手在最终诚实视图（）的攻击

### 3.1 现实世界执行

* **敌手限制**：
  + 无法控制该视图的Leader（诚实节点），仅能延迟消息至。
* **协议流程**：
  1. **Prepare阶段**：诚实Leader广播提案，所有副本在超时窗口内接收。
  2. **QC生成**：即使敌手延迟投票，诚实副本仍可收集票生成QC。
  3. **决策提交**：最终在时间内达成共识。

### 3.2 理想世界模拟

* **模拟器操作**：
  1. **超时窗口扩展**：设置，匹配敌手最大延迟。
  2. **延迟投票注入**：腐败副本的投票在内被调度，确保QC生成。
  3. **提交同步**：调用DECIDE接口提交决策，并广播。

### 3.3 不可区分性分析

* **超时覆盖性**：理想世界的超时窗口完全覆盖敌手延迟，与真实协议行为一致。
* **最终性保证**：两种世界均在时间内提交，且QC链式哈希不可逆。
* **结论**：环境无法通过最终视图的时序或QC内容区分世界。

# 三、全局时间分析

## 3.1 全局时间上界推导

### 3.1.1 超时机制与视图切换

HotStuff协议通过**指数退避超时机制**实现视图切换的时序控制，具体规则如下：

* **初始超时参数**：（基础时间单位）；
* **超时递增规则**：每次视图切换时，超时参数翻倍：
* **最大视图切换次数**：敌手最多控制个节点，需经过次视图切换以确保进入诚实Leader视图。

### 3.1.2 恶意视图切换时间

**计算逻辑**：

1. **视图切换次数**：需次切换以排除敌手干扰；
2. **超时时间累加**：每次切换后超时时间翻倍，总切换时间为：

**分阶段验证**：

| 视图编号 | 超时时间 | 累计时间（） |
| --- | --- | --- |
| 0 |  |  |
| 1 |  |  |
| 2 |  |  |
| ... | ... | ... |
|  |  |  |

### 3.1.3 最终诚实视图时间

在最终诚实视图中（），各阶段的超时参数为：

1. **Prepare阶段**：等待提案和投票，耗时；
2. **PreCommit阶段**：等待QC生成，耗时；
3. **Commit阶段**：等待最终提交，耗时；

**总耗时**：

### 3.1.4 全局时间上界整合

综合恶意视图切换时间与最终诚实视图时间，全局时间上界为：

**化简过程**：

### 3.1.5 关键参数对照表

| 参数符号 | 定义描述 | 取值范围 |
| --- | --- | --- |
|  | 基础超时单位 | 网络延迟上限 |
|  | 第次视图的超时参数 |  |
|  | 敌手控制的节点数上限 |  |

### 3.1.6 时间复杂度分析

* **指数级时间增长**：由于超时参数按递增，总时间上界为；
* **最优性证明**：该上界在敌手最大延迟下不可优化，否则将触发额外视图切换。

## 四、UC不可区分性证明

### 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。

### 4. 高并发场景下的安全性分析

在高并发场景下，HotStuff通过**QC哈希链的连续性约束**、**视图切换的同步机制**及**诚实节点的多数保证**确保安全性不可区分性。具体而言：

1. **QC链式唯一性**：每个合法提案必须严格引用前序QC哈希（见的Prepare阶段规则），确保并发提案中仅最高QC关联的区块可收集票，敌手无法为冲突区块生成合法QC。
2. **时序隔离与指数退避**：通过的全局时钟与超时参数递增机制（见协议），不同视图的QC生成窗口无重叠，消除并发提案的时序竞争。
3. **诚实多数约束**：即使敌手控制个节点并发延迟投票，诚实副本仍可独立生成个合法投票（），结合的视图同步逻辑，确保最终仅链首提案被确认。
4. 攻击与模拟模型在任意并发情况下讨论，各类攻击的 case 可能并发存在，均可进行并发式模拟，以抵御对应敌手行为。并且，本协议与其他长安链系统协议并发使用时，仍能维持安全性。

# 五、结论

HotStuff协议在理想功能框架下展现出卓越的UC终止性特性，其核心优势体现在以下关键维度：

1. **接口完备性**：  
   协议通过理想功能模块（、、）实现与现实操作的精准映射。消息广播、超时检测及QC生成等核心操作均严格遵循标准接口定义，确保了协议逻辑与理想功能的无缝对接。
2. **时序精确性**：  
   通过的全局时钟同步与指数退避机制（），协议在Prepare/PreCommit/Commit三阶段中严格约束时间窗口。敌手延迟行为被限制在的超时上界内，保障了各阶段操作的有序推进与全局时序可控性。
3. **状态一致性**：  
   基于QC哈希链的连续性规则与视图同步机制（），现实世界与理想世界中的锁定状态（LockedQC）、视图编号（curView）及QC生成高度始终保持同步。此特性通过的严格验证逻辑实现，确保了协议状态的不可逆性与全局一致性。
4. **攻击可模拟性**：  
   敌手的延迟攻击（）与腐败节点行为（如无效提案、延迟投票）被模拟器通过消息过滤（FILTER）、延迟调度（Wake）及QC生成（GENERATE-QC）完美模拟。敌手的所有操作均被限制在理想功能参数范围内，无法突破协议的安全边界。