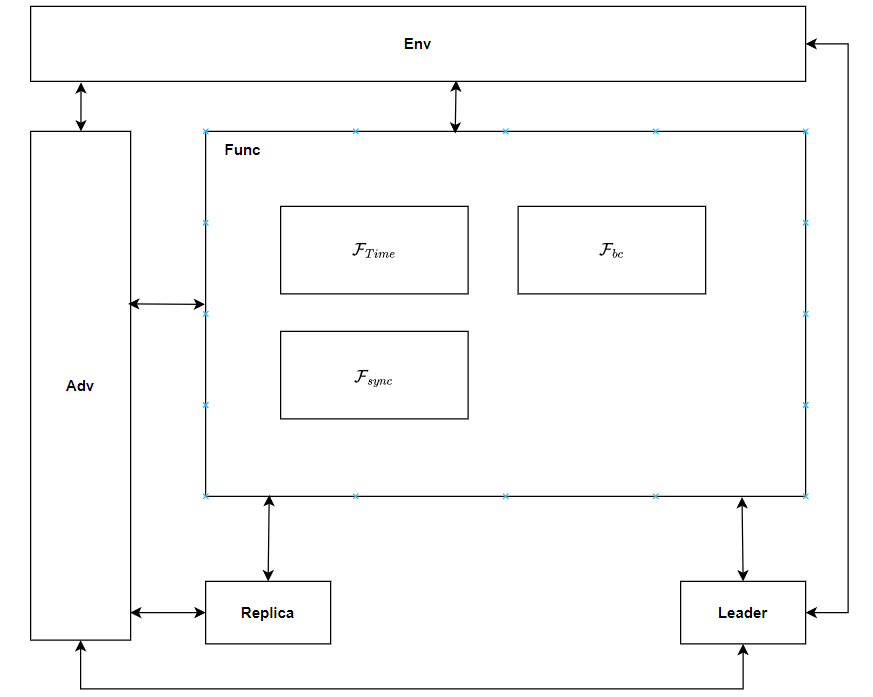
**Hotstuff建模进度**

**摘要：**

这篇文档是关于Hotstuff共识协议的v5建模进度报告。对框架图进行了修改，引入了新的理想功能Fsync保证节点同步，同时对Fhotstuff进行了修改，引入了同步机制，保证所有节点可以同步进入同一视图。

1. **整体框架图：**



1. **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. **理想功能**

# **功能**

**网络延迟攻击**

**参数**：

* : 副本集
* ：最大超时持续时间
* : 当前视图中副本 的最大超时持续时间。
* : 理想时间功能。
* : 理想广播功能。
* : 理想同步功能。

**符号说明**：

### **接收消息** 来自 ：

1. 。
2. 如果 被破坏：

* 发送 给 。

1. 发送 给 ，等待响应 ：
   * 设置 ，其中 。

### **接收消息** 来自 ：

1. 发送 给 ，并等待响应 ：
   * 如果 ：
     + 设置
     + 发送 给 和 。
     + 发送 给 。
     + 发送 给 ，接收其响应 。
       - 如果 ，设置 。
       - 否则，重新执行此步骤。
   * 否则，发送 给 。
2. 如果 ：
   * 设置 。
   * 如果 ，发送 给 ，等待响应 ：
     + 创建 。
     + 如果没有收到来自 的 ：
       - 发送 给 。
       - 设置 。
     + 否则：
       - 设置 。
       - 发送 给 和 。
       - 发送 给 。
       - 发送 给 ，接收其响应 。
         * 如果 ，设置 。
         * 否则，重新执行此步骤。
3. 否则，忽略此消息。

### **接收消息** 来自 ：

1. 发送 给 ，并等待响应 ：
   * 如果 ：
     + 设置
     + 发送 给 和 。
     + 发送 给 。
     + 发送 给 ，接收其响应 。
       - 如果 ，设置 。
       - 否则，重新执行此步骤。
   * 否则，发送 给 。
2. 如果 ：
   * 如果 并且没有收到来自 的 ：
     + 发送 给 。
     + 发送 给 ，并等待响应 ：
       - 设置 。
   * 否则：
     + 设置 。
     + 发送 给 和 。
     + 发送 给 。
     + 发送 给 ，接收其响应 。
       - 如果 ，设置 。
       - 否则，重新执行此步骤。

### **收到消息** 来自 时：

1. 向 发送 ，并等待回复，回复形式为 ：
   * 如果 ：
     + 设置
     + 向 和 发送 。
     + 向 发送 。
     + 向 发送 ，接收其响应 。
       - 如果 ，设置 。
       - 否则重新执行此步骤。
   * 否则，向 发送 。
     + 如果 ：
       - 设置 。
       - 如果 ：
         * 创建 。
       - 如果没有收到来自 的 ：
         * 向 发送 。
         * 向 发送 ，并等待回复，回复形式为 ：

设置 。

* + - 否则：
      * 设置 。
      * 向 和 发送 。
      * 向 发送 。
      * 向 发送 ，接收其响应 。
        + 如果 ，设置 。
        + 否则重新执行此步骤。

1. 否则，忽略此消息。

### **收到消息** 来自 时：

1. 向 发送 ，并等待回复，回复形式为 ：
   * 如果 ：
     + 设置 。
     + 向 和 发送 。
   * 否则，向 发送 。
2. 如果 ：
   * 如果没有收到来自 的 ：
     + 设置 。
     + 向 发送 。
     + 向 发送 ，并等待回复，回复形式为 ：
       - 设置 。
   * 否则：
     + 设置 。
     + 向 和 发送 。
     + 向 发送 。
     + 向 发送 ，接收其响应 。
       - 如果 ，设置 。
       - 否则重新执行此步骤。
3. 否则，忽略此消息。

### **收到消息** 来自 时：

1. 向 发送 ，并等待回复，回复形式为 ：
   * 如果 ：
     + 设置
     + 向 和 发送 。
     + 向 发送 。
     + 向 发送 ，接收其响应 。
       - 如果 ，设置 。
       - 否则重新执行此步骤。
   * 否则，向 发送 。
     + 如果 ：
       - 设置 。
       - 如果 ：
         * 创建 。
       - 如果没有收到来自 的 ：
         * 向 发送 。
         * 向 发送 ，并等待回复，回复形式为 ：

设置 。

* + - 否则：
      * 设置 。
      * 向 和 发送 。
      * 向 发送 。
      * 向 发送 ，接收其响应 。
        + 如果 ，设置 。
        + 否则重新执行此步骤。

1. 否则，忽略此消息。

### **收到消息** 来自 时：

* 1. 向 发送 ，并等待回复，回复形式为 ：
     + 如果 ：
       - 设置 。
       - 向 和 发送 。
     + 否则，向 发送 。
  2. 如果 ：
     + 如果没有收到来自 的 ：
       - 设置 。
       - 向 发送 。
       - 向 发送 ，并等待回复，回复形式为 ：
         * 设置 。
     + 否则：
       - 设置 。
       - 向 和 发送 。
       - 向 发送 。
       - 向 发送 ，接收其响应 。
         * 如果 ，设置 。
         * 否则重新执行此步骤。
  3. 否则，忽略此消息。

### **收到消息** 来自 时：

* 1. 向 发送 ，并等待回复，回复形式为 ：
     + 如果 ：
       - 设置 。
       - 向 和 发送 。
       - 向 发送 。
       - 向 发送 ，接收其响应 。
         * 如果 ，设置 。
         * 否则重新执行此步骤。
     + 否则，向 发送 。
       - 如果 ：
       - 设置 。
       - 如果 ：
         * 创建 。
       - 如果没有收到来自 的 ：
         * 向 发送 。
         * 向 发送 ，并等待回复，回复形式为 ：

设置 。

* + - * 否则：
        + 设置 。
        + 向 和 发送 。
        + 向 发送 。
        + 向 发送 ，接收其响应 。

如果 ，设置 。

否则重新执行此步骤。

* 1. 否则，忽略此消息。

### **收到消息** 来自 时：

* 1. 向 发送 ，并等待回复，回复形式为 ：
     + 如果 ：
       - 设置 。
       - 向 和 发送 。
     + 否则，向 发送 。
  2. 如果 ：
     + 如果没有收到来自 的 ：
       - 通过 执行新命令。
       - 向 发送 。
       - 向 发送 ，接收其响应 。
         * 如果 ，设置 ，并设置 。
         * 否则重新执行此步骤。
     + 否则：
       - 设置 。
       - 向 和 发送 。
       - 向 发送 。
       - 向 发送 ，接收其响应 。
         * 如果 ，设置 。
         * 否则重新执行此步骤。
  3. 否则，忽略此消息。

1. **协议描述**

–Party Environment:

调用更新轮次，根据轮次取模GETLEADER()=curView%n选取某个副本作为本轮的leader。

–Party Leader:

**New\_view**：新领导者从功能收集来自(n - f)个副本的"new-view"消息。这些消息包含每个副本在上一轮的（prepareQC）。

**Proposal**: 领导者调用从这些New\_view消息中选择具有最高视图编号prepareQC，（如果没有的话，为⊥)并基于此创建一个新的提案（Proposal）

**Broadcast MSG**: 领导者向所有副本广播这个提案，并附带其选择的最高prepareQC作为安全证明。

**QC**:领导者调用，对来自replica的部分签名进行组合生成QC。并且将其广播给replica。

–Party Replica:

**safeNode**: 在收到来自Leader的提议消息m后，它首先调用功能检查提案消息m，m携带QC的正确性参数（justification）m.justify，检查后确定m.node是否可以安全接受。

**Prepare**: 根据收到的Proposal消息m，调用，将投票发送给leader。

**Pre-commit**: 根据收到的PrepareQC，更新自身PrepareQC，调用。将投票发送给leader。

**Commit**: 根据收到的Pre-commitQC，更新lockedQC，调用。将投票发送给leader。

**Next\_view**:

在所有阶段中，副本在视图viewNumber处等待消息的超时时间，超时时间由辅助的确定。如果nextView（viewNumber）中断等待，副本调用,增加viewNumber并开始下一个视图。

**RoundOK**: 副本将等待轮次更新，开始新的轮次。

相关数据结构：

