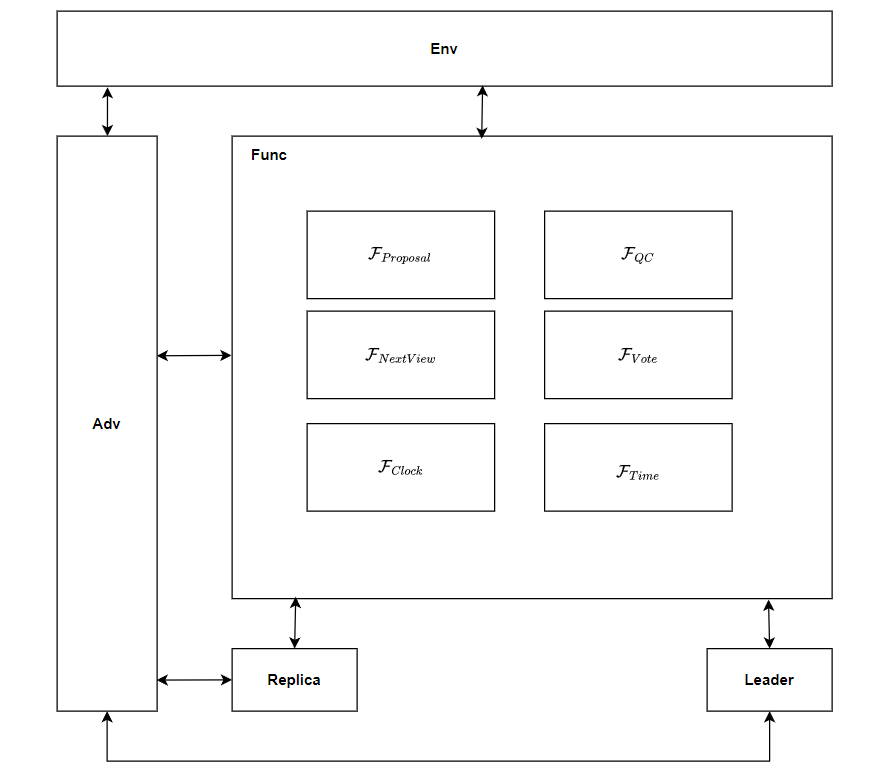
**Hotstuff建模进度**

**摘要：**

这篇文档是关于Hotstuff共识协议的12.04建模进度报告。针对上周发现的超时处理不够细化的问题，引入了新的理想功能F\_time,对之前不够细化的理想功能重新进行了描述，提出了一个简单的F\_ideal草稿。

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. **协议描述**

–Party Environment:

调用更新轮次，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。

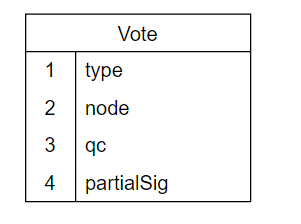
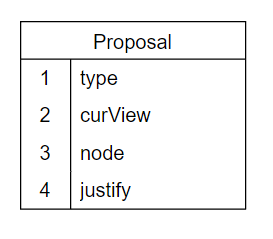
**Commit**: 根据收到的Pre-commitQC，更新lockedQC，调用。将投票发送给leader。

**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：



1. **理想功能**

–Hotstuff的理想功能::

### 初始化

初始化以下变量：

* 视图编号
* 初次超时时间T0:=300 ms
* 各副本当前prepareQC
* 各副本当前lockedQC

### Prepare 阶段

* 当节点收到提议块后，进入 Prepare 阶段，并广播 voteMsg(prepare, n.node, ⊥) 消息，表示自己对提议块的初步投票。
* 节点等待其他节点的投票消息。如果收到 n-f 个有效的 voteMsg(prepare, m.node, ⊥) 消息，且这些消息来自不同节点，且没有冲突，节点进入 Pre-commit 阶段。

**条件检查：**

* 如果收到 n-f 个有效的 voteMsg(prepare, m.node, ⊥) 消息，且这些消息有效且一致，节点认为提议块有效，进入 Pre-commit 阶段。

### Pre-commit 阶段

* 当节点从其他节点收到 n-f 个有效的 voteMsg(prepare, m.node, ⊥) 消息后，进入 Pre-commit 阶段，并广播 preCommitMsg(prepare, n.node, curView) 消息，表示自己对提议块的预提交。
* 节点继续等待 n-f 个有效的 new-view 消息，其中每个消息必须匹配当前视图的提议块，且视图号为 curView - 1。
* 如果节点收到 n-f 个有效的 matchingMsg(m, new-view, curView − 1) 消息，且这些消息匹配当前视图的提议块，节点进入 Commit 阶段。

**条件检查：**

* 如果节点收到 n-f 个有效的 matchingMsg(m, new-view, curView − 1) 消息，且这些消息有效，节点认为当前视图的提议块有效，进入 Commit 阶段。

**Commit 阶段**

* 当节点从其他节点收到 n-f 个有效的 preCommitMsg(prepare, m.node, curView) 消息后，进入 Commit 阶段，并广播 commitMsg(prepare, n.node, curView) 消息，表示自己确认该提议块。
* 节点继续等待其他节点的 commitMsg 消息，直到收到 n-f 个有效的 commitMsg 消息。如果这些消息来自不同节点，且确认的提议块一致，则节点将区块提升到更高的高度，进入下一轮的共识过程。

1. **共识模块**

# ****场景1：****网络延迟攻击

* **攻击目标：** 延缓共识达成时间，影响系统性能。
* **攻击手段：** 恶意节点通过操纵网络路径或发送延迟包来拖慢消息传递速度。
* **攻击复杂度：** 中等，依赖攻击者对网络拓扑的了解和干预能力。
* **影响范围：** 整体共识集群。
* **影响后果：** 导致出块时间增加，可能引发节点对协议的信任问题，甚至触发超时机制使协议失败。

**防御措施：**

1. **检测措施：**
   * **分布式网络监控：** 使用 Prometheus 或类似工具，监控各节点间消息延迟的实时状态，触发延迟告警机制。
   * **路径异常检测：** 使用 Traceroute 或分布式网络流量分析工具检测网络路径的异常。
2. **缓解措施：**
   * **动态超时调整：**
     + 引入自适应超时算法，根据实时网络条件动态调整协议的超时阈值。
   * **消息缓存与优先级：**
     + 在延迟情况下缓存高优先级的共识消息，确保消息不会丢失或过期。
3. **预防措施：**
   * **多路径传输协议（MPTCP）：** 为每条共识消息提供多条传输路径，降低单一路径延迟的影响。
   * **延迟签名校验：** 使用消息时间戳和签名校验机制（如 HMAC），确保恶意延迟的消息不被接收。

**适合 UC 证明的部分：**

* **终止性：** UC 框架证明协议在一定时间内仍能达成一致，即便存在延迟影响。
* **有效性：** 验证协议是否能确保由合法节点提出的数据值仍能被达成共识，而不因延迟被放弃。

# ****场景2：****女巫攻击（Sybil 攻击）

* **攻击目标：** 创建多个虚假节点控制投票权或破坏共识过程。
* **攻击手段：** 利用虚假身份注册或通过接管资源伪造多个节点身份。
* **攻击复杂度：** 较高，需耗费资源生成大量虚假节点并成功加入网络。
* **影响范围：** 单点或多个节点，但可扩展至整个共识集群。
* **影响后果：** 篡改共识投票结果，降低系统的容错性和数据完整性。

**防御措施：**

1. **检测措施：**
   * **行为分析：** 实施基于网络行为的异常检测，通过流量分析区分真实节点和虚假节点。
   * **节点验证：** 在节点加入时，使用零知识证明（ZKP）或可信计算环境（如 Intel SGX）验证其身份真实性。
2. **缓解措施：**
   * **权重机制：**
     + 按资源分配节点投票权，如采用工作量证明（PoW）或权益证明（PoS），限制虚假身份的投票权重。
   * **去中心化注册服务：**
     + 使用分布式身份协议（如 DID），确保身份生成分布式管理，减少集中化伪造身份的可能性。
3. **预防措施：**
   * **可信硬件：** 使用 TPM 或 HSM 设备确保节点身份的物理真实性。
   * **随机抽样机制：** 在共识过程中随机选取节点，降低攻击者影响范围。

**适合 UC 证明的部分：**

* **有效性：** 确保共识过程仅包含由真实节点提出的数据值，拒绝虚假节点的影响。
* **数据完整性：** 验证信息从合法节点传播至接收节点的过程中，未被篡改。

**场景3：**网络分区攻击

* **攻击目标：** 隔离部分节点，使其无法参与共识流程。
* **攻击手段：** 通过网络干扰技术（如路由篡改或拒绝服务攻击）断开特定节点与集群的连接。
* **攻击复杂度：** 高，需对目标网络和拓扑结构有较深了解。
* **影响范围：** 单个分区或整体集群（视攻击规模）。
* **影响后果：** 导致部分节点因无法通信而无法完成共识投票，影响系统的健壮性。

**防御措施：**

1. **检测措施：**
   * **拓扑分析：** 定期分析网络拓扑，识别分区风险。
   * **心跳机制：** 节点定期发送心跳信号，监测集群内的连通性。
2. **缓解措施：**
   * **跨链共识备份：**
     + 对重要状态数据进行备份，防止分区影响全网数据一致性。
   * **协议切换：**
     + 在分区检测到后，临时切换到基于局部分区的共识模式（如分区 PBFT）。
3. **预防措施：**
   * **多链路冗余：** 使用 SD-WAN 技术构建多路径连接，提高网络稳定性。
   * **中继节点：** 增加可信中继节点，确保不同分区间的连接不中断。

**适合 UC 证明的部分：**

* **终止性：** 在网络分区的条件下，协议仍然能确保余下节点最终达成一致性。
* **数据完整性：** 验证协议对消息分发的安全性，避免篡改或丢失。

**场景4：节点宕机攻击**

* **攻击目标：** 利用意外宕机事件干扰共识进程。
* **攻击手段：** 攻击者通过物理、网络或软件漏洞触发节点停机。
* **攻击复杂度：** 中等，视节点分布与备份机制而定。
* **影响范围：** 单点宕机影响有限，多点宕机可能危及整个集群。
* **影响后果：** 共识参与节点减少，可能导致共识无法达成或协议效率下降。

**防御措施：**

1. **检测措施：**
   * **状态监控：** 部署如 Nagios 的节点监控系统，自动识别宕机事件并告警。
   * **日志分析：** 收集并分析节点日志，判断宕机原因。
2. **缓解措施：**
   * **拜占庭容错机制（BFT）：**
     + 允许部分节点失效，其他节点继续达成共识。
   * **自动切换备份节点：**
     + 利用故障转移机制（如 Pacemaker），确保宕机节点的功能快速恢复。
3. **预防措施：**
   * **定期备份：** 通过节点热备份和容灾策略，确保系统快速恢复。
   * **分布式架构：** 提高节点分布冗余度，减少宕机对系统的整体影响。

**适合 UC 证明的部分：**

* **终止性：** 证明即使部分节点失效，协议仍能在剩余活跃节点中完成共识。
* **不可篡改性：** 宕机节点恢复后，验证其输出仍与共识数据一致。

**场景5：**DDOS攻击

* **攻击目标：** 消耗系统资源，使节点或整个集群无法正常参与共识过程。
* **攻击手段：** 攻击者通过大量伪造的流量请求，占用节点的带宽、计算能力或存储资源。
* **攻击复杂度：** 较高，需要攻击者拥有大量资源或僵尸网络。
* **影响范围：** 单点或整体集群，取决于攻击强度和目标节点的位置。
* **影响后果：** 导致节点超载无法响应合法请求，共识过程延迟甚至失败，影响系统的可用性和性能。

**防御措施：**

1. **检测措施：**
   * **流量监控：** 使用 Snort 或 Wireshark 等工具监控网络流量，检测异常高流量模式。
   * **异常行为分析：** 结合机器学习模型，分析流量来源、频率和特征，识别恶意请求。
2. **缓解措施：**
   * **分布式流量清洗：**
     + 部署 CDN 或 Anti-DDoS 服务（如 Cloudflare），过滤攻击流量并仅转发合法请求。
   * **速率限制：**
     + 实施每 IP 的请求速率限制，通过令牌桶算法限制访问频率，防止恶意请求耗尽资源。
   * **优先级分配：**
     + 针对合法共识消息设置高优先级队列，确保攻击流量不会影响关键通信。
3. **预防措施：**
   * **访问控制列表（ACL）：** 定义严格的白名单和黑名单策略，阻止可疑 IP 地址访问系统。
   * **弹性扩展：**
     + 通过 Kubernetes 和 Auto-scaling 技术动态调整系统资源，缓解高负载情况。
   * **去中心化架构：**
     + 增加分布式节点，避免单点过载，通过 DHT 技术（如 Kademlia）分散流量。

**适合 UC 证明的部分：**

* **终止性：** 验证协议在高流量冲击下，能否通过容错机制保持共识的完成。
* **数据完整性：** 确保攻击过程中，共识数据的传输与存储不受干扰或篡改。
* **有效性：** UC 框架证明合法节点提出的数据值仍能在攻击环境中成功通过共识。

1. **下周安排**
2. 对F\_ideal做细化
3. 继续对模型进行优化，考虑同步因素