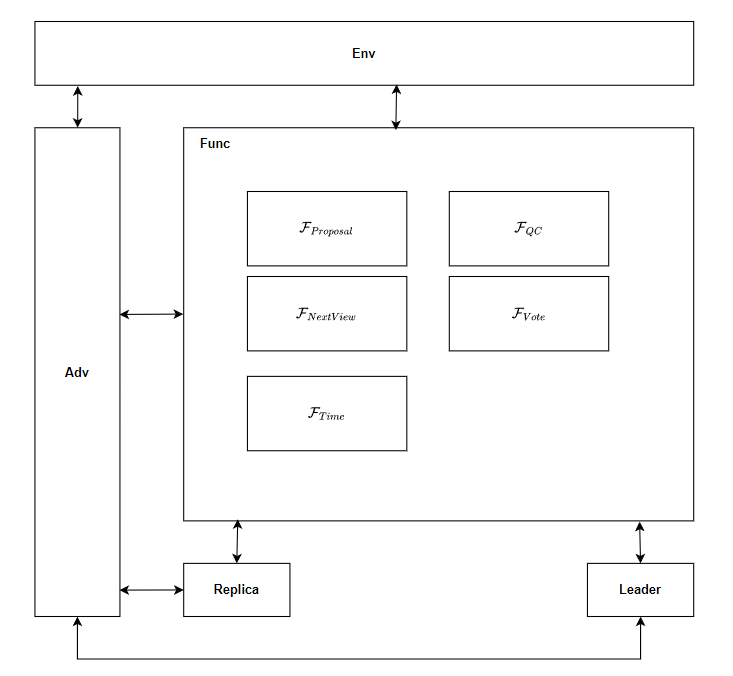
**Hotstuff建模进度**

**摘要：**

这篇文档是关于Hotstuff共识协议的12.04建模进度报告。根据事件驱动的hotstuff模型，对F\_ideal作了细致描述。

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，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。

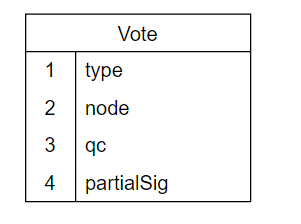
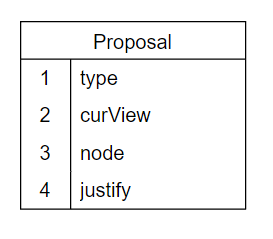
**Commit**: 根据收到的Pre-commitQC，更新lockedQC，调用。将投票发送给leader。

**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：



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
* 创世节点：（被所有正确副本都知道，并且自身包含一个硬编码的QC）
* 最后投票节点的高度：
* 锁定节点：
* 最后执行的节点：
* 由Pacemaker保存的叶节点：
* 由Pacemaker保存的最高已知 QC（类似于 genericQC）：

### OnPropose phase

当视图发生更替后领导者收到各个副本发送的：

* 调用：

-执行，如果则更新和

* 调用：

-将发送给

* 调用：

-调用

-执行

-将提案消息 广播给所有副本

### OnVote phase

当副本收到广播的提案消息时：

* 调用：

-如果 对执行安全性和活性检测：如果 ，更新，发送投票SEND(GetLeader(),。

-对调用,更新本地树。

* 调用时：

-对节点执行初始化：,

-Pacemaker更新和：

-接着对进行检测，若形成双链则更新,若形成三链则更新并且提交节点b的proposal：b)

* 调用时：

-如果,对b节点的父节点进行递归处理：,接着执行节点的cmd：

当Leader收到副本的投票消息时：

* 调用：

-先对投票进行检查，防止收到重复的投票，接着收集投票，如果收集到的投票超过n-f，Pacemaker更新QC：。

1. **终止性证明**

**定理一：在 GST (全局稳定时间)之后，如果满足以下条件：**

**所有正确的副本都保持在同一个视图（view v）中；**

**该视图的领导者是正确的； 那么一定会在有界的时间内达成决策。**

**证明过程**

1. **新视图的开始**： 领导者调用，并且计算其 **highQC**。然后，领导者会广播一个**prepare**消息。
2. ： 假设在所有副本（包括领导者本身）中，保持的最高 。说明至少有f+1个正确的副本曾投票支持过这个，并且已经通过发送给了领导者。
3. **领导者获取匹配的 prepareQC**： 因此，领导者在接收到这些新视图消息时，必定能找到一个匹配的，并将其作为自己的 **highQC** 调用广播到其他副本。
4. **副本投票**：假设所有正确的副本都与领导者同步，而且领导者是非故障的。在这种情况下，所有正确的副本都会在 **prepare** 阶段投票，因为在 **safeNode** 函数中，条件Line 27会得到满足，即使消息中的节点与副本的过时冲突，Line26条件不满足。
5. **完成决策**： 当领导者组装出一个有效的 **prepareQC** 后，所有副本都会在后续的阶段投票，最终达成决策。由于这些阶段是有序的，且每个阶段的长度是有界的，因此，在 GST 之后，所有阶段完成所需的时间也是有界的。