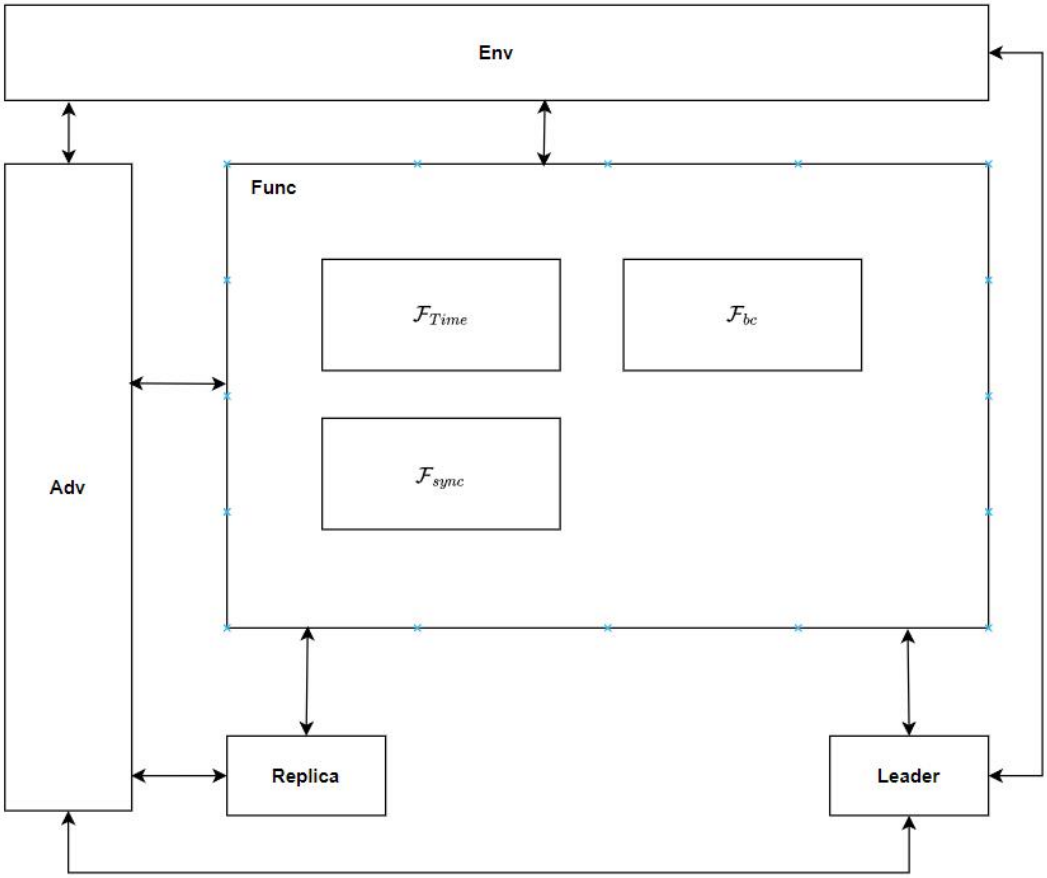
Hotstuff 建模进度

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 F_{sync} 保证节点同步，同时对 $F_{hotstuff}$ 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。

一、整体框架图：



二、功能描述

1. F_{Proposal}

初始化：设置 Proposal ≔ ⊥。

- 当收到消息$n - f\ NEW - VIEW\ messages$: $\{m_1, m_2, ..., m_{n-f}\}$时，
  - 选取这些消息中最高的 prepareQC 最为 highQC：

$$Determine\ highQC = max(QC_i | \forall m_i, QC_i.viewNumber)$$

- 在 highQC 的节点的叶子上写入客户指令，提出新的提案 B：

$$create\ proposal\ B = Leaf(highQC.node, command)$$

-将提案 B、highQC 封装在 MSG 中广播给 replica：

2. F_{Vote}

初始化：设置 VOTEMSG $\coloneqq \perp$。

- 当收到来自$\mathcal{F}_{Proposal}$的消息$MSG(PREPARE, CurProposal, highQC)$时，
  - 先检查 m 是否与自己状态匹配：

$$MATCHINGMSG(m, PREPARE, curView)$$

  - 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：

$$if\ m.node\ extends\ from\ m.justify.node\ \cap SAFENODE(m.node, m.justify)$$

  - 在 highQC 的节点的叶子上写入客户指令，提出新的提案 B：

$$create\ proposal\ B = Leaf(highQC.node, command)$$

-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$\mathcal{F}_{QC}$的消息$MSG(type, \perp, QC)$时，

- 先检查 QC 是否与自己状态匹配：

$$MATCHHINGQC(m.justify, type, curView)$$

- 如果决定投票且 type 是 PREPARE 阶段，更新本地状态：

$$prepareQC \leftarrow m.justify$$

- 如果决定投票且 type 是 PRE-COMMIT 阶段，更新本地状态：

$$lockedQC \leftarrow m.justify$$

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

3. F_{QC}

初始化：设置$QC \coloneqq \perp$。

-当收到 2f+1 条投票消息$VOTEMSG(type, m.justiy.node, \perp)$时：

- 先检查 m 是否与自己状态匹配：

$$MATCHINGMSG(m, type, curView)$$

- 收集 replica 的投票，把部分签名组合：

$$qc.type \leftarrow m.type : m \in V$$

$$qc.viewNumber \leftarrow m.viewNumber : m \in V$$

$$qc.node \leftarrow m.node : m \in V$$

$$qc.sig \leftarrow tcombine(qc.type, qc.viewNumber, qc.node, \{m.partialSig \mid m \in V\})$$

-将 QC 封装在 MSG 中广播给 replica：

4. F_{TIME}

初始化：设置$t_i \in T$，$t_i \coloneqq T0$。

- 当从任意 replica $v_i \in V$接收到$(timeStart, T)$请求时，将t_i更新为 $t_i \leftarrow T$，向 replica v_i返回一个$(timeOK)$消息，然后开始倒计时。
- 当从某一个$t_i \in T$, $t_i = 0$ 时，它会向对应的 replica v_i发送一个$(timeOver, 2T)$消息。

5. F_{Next_view}

初始化：设置$viewNumber: = \perp$，$prepareQC: = \perp$。

- 当从任意 replica m 收到 next_view 请求 MSG(⊥，m,prepareQC)时，将$viewNumber$更新为$m.viewNumber$，将 prepareQC 更新为 m.justify。
- 将$viewNumber$、$prepareQC$封装在 NEW-VIEW message 中发送给下一视图的 leader。

三、协议描述

- Party Environment:

调用$\mathcal{F}_{NextView}$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$\mathcal{F}_{NextView}$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$\mathcal{F}_{Proposal}$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$\mathcal{F}_{QC}$，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$\mathcal{F}_{Vote}$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$\mathcal{F}_{Vote}$(PREPARE, m. node, ⊥)，将投票发送给 leader。

Pre-commit: 根据收到的 PrepareQC，更新自身 PrepareQC，调用$\mathcal{F}_{Vote}$(PREPARE, m. justify. node, ⊥)。将投票发送给 leader。

Commit: 根据收到的 Pre-commitQC，更新 lockedQC，调用$\mathcal{F}_{Vote}$(COMMIT, m. justify. node, ⊥)。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$\mathcal{F}_{TIME}$确定。如果 nextView (viewNumber) 中断等待，副本调用$\mathcal{F}_{NextView}$,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：

Proposal	
1	type
2	curView
3	node
4	justify

Vote	
1	type
2	node
3	qc
4	partialSig

四、理想功能

Functionality $\mathcal{F}_{hotstuff}^{R,\Delta}$ $[\ \mathcal{F}_{time}, \mathcal{F}_{bb}, \mathcal{F}_{sync}\ ]$

Network Delay Attack

Parameters:

- R: Replica Set
- Δ：the Maximum timeout duration
- Δ_{vnum}^{i}: the Maximum timeout duration of Replica $r_i \in R$ within current view.
- $\mathcal{F}_{time}$: Ideal functionality for timing.
- $\mathcal{F}_{bb}$: Ideal functionality for broadcast.
- $\mathcal{F}_{sync}$: Ideal functionality for synchronization.

Symbol Explanation:

$curView_j$: the current Viewnumber of replica r_j

QC: Quorum certificates which combines a collection of signatures for the same tuple signed by (n − f) replicas.

m: Message or VoteMessage

$m.node$: The node that proposed this proposal within curView

$m.justify$: The QC carried by this message

$phase_j \in$ {Prepare, PreCommit, Commit, Decide}: the phase of replica $r_j$within curView

δ: actual execute time

Upon receiving message $MSG\langle \text{NEW} - \text{VIEW}, ViewNumber\rangle$ **from**

Leader_i**:**

1. $\text{curView}_i := \text{ViewNumber}+1$.
2. If Leader_i is corrupted:

- Send $\langle \text{Input}, \text{curView}_i, \text{cmd} \rangle$ to $\mathcal{S}$.

1. Send $\langle \text{Prepare}, \text{i}, \text{curView}_i, \text{sid} \rangle$ to $\mathcal{S}$ wait for a response of the form $\langle \text{startPrepare}, \text{i}, \text{curView}_i, \text{sid} \rangle$:
    - Set $\text{phase}_j := \text{Prepare}$ which $\text{curView}_j = \text{curView}_i$.

Upon receiving message $MSG\langle \text{NEW} - \text{VIEW}, \perp, \text{PrepareQC} \rangle$ from Replica r_j:

1. Send $\langle \text{Sleep}, \text{sid} \rangle$ to $\mathcal{S}$ and wait for a response of the form $\langle \text{Wake}, \text{sid}, \delta \rangle$:
    - If $\delta > \Delta^j_{vnum}$:
        - set $\Delta^j_{vnum} := \Delta^j_{vnum} * 2, \text{curView}_j := \text{curView}_j + 1$
        - send$\langle \text{NEW} - \text{VIEW}, \text{curView}_j, j \rangle$ to $\mathcal{S}$ and Leader_i.
        - Send $\langle \text{RoundOK} \rangle$ to $\mathcal{F}_{sync}$.
        - Send $\langle \text{RequestRound} \rangle$ to $\mathcal{F}_{sync}$,receive its response d_i.
            - If $d_i = 0$,set $\text{curView}_i := \text{curView}_i + 1$.
            - else re-execute this step.
    - else,Send $\langle \text{timeStart}, \text{sid}, \delta \rangle$ to $\mathcal{F}_{time}$.
2. If $MATCHINGMSG\langle \text{m}, \text{NEW} - \text{VIEW}, \text{curView}_i - 1 \rangle \cap \text{step}_i = \text{Prepare}$:
    - Set $\text{nums}_i(\text{NEW-VIEW}) := \text{nums}_i(\text{NEW-VIEW}) + 1$.
    - If $(\text{nums}_i(NEW - VIEW) > 2f + 1)$,Send $\langle \text{CreateProposal}, \text{i}, \text{sid} \rangle$ to $\mathcal{S}$ and wait for a response of the form $\langle \text{StartProposal}, \text{i}, \text{sid} \rangle$:
        - Create$\langle \text{Prepare}, \text{Curproposal}, \text{highQC} \rangle$.
        - If no$\langle \text{timeOver}, \text{sid} \rangle$ is received from $\mathcal{F}_{time}$:
            - send $\langle \text{Prepare}, \text{Curproposal}, \text{highQC} \rangle$ to $\mathcal{F}_{bb}$.
            - Set $\text{phase}_i = \text{PreCommit}$.
        - Else:
            - set $\Delta^i_{vnum} := \Delta^i_{vnum} * 2$.
            - send$\langle \text{NEW} - \text{VIEW}, \text{curView}_i, i \rangle$ to $\mathcal{S}$ and Leader_i.
            - Send $\langle \text{RoundOK} \rangle$ to $\mathcal{F}_{sync}$.
            - Send $\langle \text{RequestRound} \rangle$ to $\mathcal{F}_{sync}$,receive its response d_i.
                - If $d_i = 0$,set $\text{curView}_i := \text{curView}_i + 1$.
                - else re-execute this step.
3. Else，ignore this message.

Upon receiving message $MSG\langle \text{Prepare}, \text{CurProposal}, highQC \rangle$ from Leader_i:

1. Send $\langle \text{Sleep}, \text{sid} \rangle$ to $\mathcal{S}$ and wait for a response of the form $\langle \text{Wake}, \text{sid}, \delta \rangle$:

- If $\delta > \Delta^j_{vnum}$:
  - set $\Delta^j_{vnum} := \Delta^j_{vnum} * 2$, $\text{curView}_j := \text{curView}_j + 1$
  - send$\langle \text{NEW} - \text{VIEW}, \text{curView}_j, j \rangle$ to $\mathcal{S}$ and Leader_i.
  - Send $\langle \text{RoundOK} \rangle$ to $\mathcal{F}_{sync}$.
  - Send $\langle \text{RequestRound} \rangle$ to $\mathcal{F}_{sync}$,receive its response d_i.
    - If $d_i = 0$,set $\text{curView}_i := \text{curView}_i + 1$.
    - else re-execute this step.
- else,Send $\langle \text{timeStart}, \text{sid}, \delta \rangle$ to $\mathcal{F}_{time}$.

2. If $MATCHINGMSG\langle \text{m}, \text{Prepare}, \text{curView}_j \rangle \cap \text{step}_j = \text{Prepare}$:
   - If $\text{SafeNode}(m.node, m.justify) \cap \text{m.node}$ extends from m.justify.node and no $\langle \text{timeOver}, \text{sid} \rangle$ has been received from $\mathcal{F}_{time}$:
     - Send $\langle \text{Prepare}, m.node, \perp \rangle$ to Leader_i.
     - Send $\langle \text{updatePreCommit}, \text{j}, \text{sid} \rangle$ to $\mathcal{S}$ and wait for a response of the form $\langle \text{StartPreCommit}, \text{j}, \text{sid} \rangle$:
       - Set $\text{phase}_j = \text{PreCommit}$.
   - else:
     - set $\Delta^j_{vnum} := \Delta^j_{vnum} * 2$.
     - send$\langle \text{NEW} - \text{VIEW}, \text{curView}_j, j \rangle$ to $\mathcal{S}$ and Leader_i.
     - Send $\langle \text{RoundOK} \rangle$ to $\mathcal{F}_{sync}$.
     - Send $\langle \text{RequestRound} \rangle$ to $\mathcal{F}_{sync}$,receive its response d_i.
       - If $d_i = 0$,set $\text{curView}_i := \text{curView}_i + 1$.
       - else re-execute this step.
3. Else，ignore this message.

Upon receiving message $VOTEMSG\langle \text{Prepare}, m.node, \perp \rangle$ from Replicar_j:

1. Send $\langle \text{Sleep}, \text{sid} \rangle$ to $\mathcal{S}$ and wait for a response of the form $\langle \text{Wake}, \text{sid}, \delta \rangle$:
   - If $\delta > \Delta^i_{vnum}$:
     - set $\Delta^i_{vnum} := \Delta^i_{vnum} * 2$, $\text{curView}_i := \text{curView}_i + 1$
     - send$\langle \text{NEW} - \text{VIEW}, \text{curView}_i, i \rangle$ to $\mathcal{S}$ and Leader_i.
     - Send $\langle \text{RoundOK} \rangle$ to $\mathcal{F}_{sync}$.
     - Send $\langle \text{RequestRound} \rangle$ to $\mathcal{F}_{sync}$,receive its response d_i.
       - If $d_i = 0$,set $\text{curView}_i := \text{curView}_i + 1$.
       - else re-execute this step.
   - else,Send $\langle \text{timeStart}, \text{sid}, \delta \rangle$ to $\mathcal{F}_{time}$.
     - If $MATCHINGMSG\langle \text{m}, \text{Prepare}, \text{curView}_i \rangle \cap \text{phase}_i = \text{PreCommit}$:
     - Set $\text{nums}_i(\text{Prepare}) := \text{nums}_i(\text{Prepare}) + 1$
     - If $(\text{nums}_i(Prepare) > 2f + 1)$:
     - Create$\langle \text{PreCommit}, \perp, \text{PrepareQC} \rangle$.
     - If no$\langle \text{timeOver}, \text{sid} \rangle$ is received from $\mathcal{F}_{time}$:
       - send $\langle \text{PreCommit}, \perp, \text{PrepareQC} \rangle$to $\mathcal{F}_{bb}$.

- Send ⟨updateCommit, i, sid⟩ to $\mathcal{S}$ and wait for a response of the form ⟨StartCommit, j, sid⟩:
    - Set $\text{phase}_i = \text{Commit}$.
- Else:
    - set $\Delta^i_{vnum} := \Delta^i_{vnum} * 2$.
    - send⟨$\text{NEW} - \text{VIEW}, \text{curView}_i, i$⟩ to $\mathcal{S}$ and Leader_i.
    - Send ⟨RoundOK⟩ to $\mathcal{F}_{sync}$.
    - Send ⟨RequestRound⟩ to $\mathcal{F}_{sync}$,receive its response d_i.
        - If $d_i = 0$,set $\text{curView}_i := \text{curView}_i + 1$.
        - else re-execute this step.

2. Else，ignore this message.

Upon receiving message MSG⟨PreCommit, ⊥ , PrepareQC⟩ from Leader_i:

1. Send ⟨Sleep, sid⟩ to $\mathcal{S}$ and wait for a response of the form ⟨Wake, sid, δ⟩:
    - If $\delta > \Delta^j_{vnum}$:
        - set $\Delta^j_{vnum} := \Delta^j_{vnum} * 2$, $\text{curView}_j := \text{curView}_j + 1$.
        - send⟨$\text{NEW} - \text{VIEW}, \text{curView}_j, j$⟩ to $\mathcal{S}$ and Leader_i.
    - else,Send ⟨timeStart, sid, δ⟩ to $\mathcal{F}_{time}$.
2. If $MATCHINGQC$⟨$\text{m.justify}, \text{Prepare}, \text{curView}_j$⟩ ∩ $\text{phase}_j = \text{PreCommit}$:
    - If no ⟨timeOver, sid⟩ has been received from $\mathcal{F}_{time}$:
        - Set PrepareQC⬅m.justify.
        - Send VOTEMSG⟨PreCommit,m.justify.node,⊥⟩ to Leader_i.
        - Send ⟨updateCommit, j, sid⟩ to $\mathcal{S}$ and wait for a response of the form ⟨StartCommit, j, sid⟩:
            - Set $\text{phase}_j = \text{Commit}$.
    - else:
        - set $\Delta^j_{vnum} := \Delta^j_{vnum} * 2$.
        - send⟨$\text{NEW} - \text{VIEW}, \text{curView}_j, j$⟩ to $\mathcal{S}$ and Leader_i.
        - Send ⟨RoundOK⟩ to $\mathcal{F}_{sync}$.
        - Send ⟨RequestRound⟩ to $\mathcal{F}_{sync}$,receive its response d_i.
            - If $d_i = 0$,set $\text{curView}_i := \text{curView}_i + 1$.
            - else re-execute this step.
3. Else，ignore this message.

Upon receiving message $VOTEMSG$⟨$\text{PreCommit}, m.node, \perp$⟩ from Replica_j:

1. Send ⟨Sleep, sid⟩ to $\mathcal{S}$ and wait for a response of the form ⟨Wake, sid, δ⟩:

- If $\delta > \Delta^i_{vnum}$:
  - set $\Delta^i_{vnum} := \Delta^i_{vnum} * 2, \text{curView}_i := \text{curView}_i + 1$
  - send$\langle \text{NEW} - \text{VIEW}, \text{curView}_i, i \rangle$ to $\mathcal{S}$ and Leader_i.
  - Send $\langle \text{RoundOK} \rangle$ to $\mathcal{F}_{sync}$.
  - Send $\langle \text{RequestRound} \rangle$ to $\mathcal{F}_{sync}$,receive its response d_i.
    - If $d_i = 0$,set $\text{curView}_i := \text{curView}_i + 1$.
    - else re-execute this step.
- else,Send $\langle \text{timeStart}, \text{sid}, \delta \rangle$ to $\mathcal{F}_{time}$.
  - If $MATCHINGMSG\langle \text{m}, \text{PreCommit}, \text{curView}_i \rangle \cap \text{phase}_i = \text{Commit}$:
  - Set $\text{nums}_i(\text{PreCommit}) := \text{nums}_i(\text{PreCommit}) + 1$
  - If $(\text{nums}_i(\text{PreCommit}) > 2f + 1)$:
  - Create$\langle \text{Commit}, \perp, \text{PreCommitQC} \rangle$.
  - If no$\langle \text{timeOver}, \text{sid} \rangle$ is received from $\mathcal{F}_{time}$:
    - send $\langle \text{Commit}, \perp, \text{PreCommitQC} \rangle$to $\mathcal{F}_{bb}$.
    - Send $\langle \text{updateDecide}, \text{i}, \text{sid} \rangle$ to $\mathcal{S}$ and wait for a response of the form $\langle \text{StartDecide}, \text{j}, \text{sid} \rangle$:
      - Set $\text{phase}_i = \text{Decide}$.
  - Else:
    - set $\Delta^i_{vnum} := \Delta^i_{vnum} * 2$.
    - send$\langle \text{NEW} - \text{VIEW}, \text{curView}_i, i \rangle$ to $\mathcal{S}$ and Leader_i.
    - Send $\langle \text{RoundOK} \rangle$ to $\mathcal{F}_{sync}$.
    - Send $\langle \text{RequestRound} \rangle$ to $\mathcal{F}_{sync}$,receive its response d_i.
      - If $d_i = 0$,set $\text{curView}_i := \text{curView}_i + 1$.
      - else re-execute this step.

2. Else，ignore this message.

Upon receiving message $MSG\langle \text{Commit}, \perp, \text{PreCommitQC} \rangle$ from Leader_i:

1. Send $\langle \text{Sleep}, \text{sid} \rangle$ to $\mathcal{S}$ and wait for a response of the form $\langle \text{Wake}, \text{sid}, \delta \rangle$:
   - If $\delta > \Delta^j_{vnum}$:
     - set $\Delta^j_{vnum} := \Delta^j_{vnum} * 2, \text{curView}_j := \text{curView}_j + 1$.
     - send$\langle \text{NEW} - \text{VIEW}, \text{curView}_j, j \rangle$ to $\mathcal{S}$ and Leader_i.
   - else,Send $\langle \text{timeStart}, \text{sid}, \delta \rangle$ to $\mathcal{F}_{time}$.
2. If $MATCHINGQC\langle \text{m.justify}, \text{PreCommit}, \text{curView}_j \rangle \cap \text{phase}_j = \text{Commit}$:
   - If no $\langle \text{timeOver}, \text{sid} \rangle$ has been received from $\mathcal{F}_{time}$:
     - Set LockedQC⬅m.justify.
     - Send VOTEMSG$\langle$Commit,m.justify.node,⊥$\rangle$ to Leader_i.
     - Send $\langle \text{updateDecide}, \text{j}, \text{sid} \rangle$ to $\mathcal{S}$ and wait for a response of the form $\langle \text{StartDecide}, \text{j}, \text{sid} \rangle$:
       - Set $\text{phase}_j = \text{Decide}$.
   - else:

- set $\Delta_{vnum}^{j} := \Delta_{vnum}^{j} * 2$.
- send$\langle \text{NEW} - \text{VIEW}, \text{curView}_j, j \rangle$ to $\mathcal{S}$ and Leader_i.
- Send $\langle \text{RoundOK} \rangle$ to $\mathcal{F}_{sync}$.
- Send $\langle \text{RequestRound} \rangle$ to $\mathcal{F}_{sync}$,receive its response d_i.
    - If $d_i = 0$,set $\text{curView}_i := \text{curView}_i + 1$.
    - else re-execute this step.

3. Else，ignore this message.

Upon receiving message $VOTEMSG\langle \text{Commit}, m.node, \perp \rangle$ **from** Replicar_j**:**

1. Send $\langle \text{Sleep}, \text{sid} \rangle$ to $\mathcal{S}$ and wait for a response of the form $\langle \text{Wake}, \text{sid}, \delta \rangle$:
    - If $\delta > \Delta_{vnum}^{i}$:
        - set $\Delta_{vnum}^{i} := \Delta_{vnum}^{i} * 2, \text{curView}_i := \text{curView}_i + 1$
        - send$\langle \text{NEW} - \text{VIEW}, \text{curView}_i, i \rangle$ to $\mathcal{S}$ and Leader_i.
        - Send $\langle \text{RoundOK} \rangle$ to $\mathcal{F}_{sync}$.
        - Send $\langle \text{RequestRound} \rangle$ to $\mathcal{F}_{sync}$,receive its response d_i.
            - If $d_i = 0$,set $\text{curView}_i := \text{curView}_i + 1$.
            - else re-execute this step.
    - else,Send $\langle \text{timeStart}, \text{sid}, \delta \rangle$ to $\mathcal{F}_{time}$.
        - If $MATCHINGMSG\langle \text{m}, \text{Commit}, \text{curView}_i \rangle \cap \text{phase}_i = \text{Decide}$:
        - Set $\text{nums}_i(\text{Commit}) := \text{nums}_i(\text{Commit}) + 1$
        - If $(\text{nums}_i(\text{Commit}) > 2f + 1)$:
        - Create$\langle \text{Decide}, \perp, \text{CommitQC} \rangle$.
        - If no$\langle \text{timeOver}, \text{sid} \rangle$ is received from $\mathcal{F}_{time}$:
            - send $\langle \text{Decide}, \perp, \text{CommitQC} \rangle$to $\mathcal{F}_{bb}$.
            - send$\langle \text{NEW} - \text{VIEW}, \text{curView}_i, i \rangle$ to $\mathcal{S}$ and Leader_i.
            - Send $\langle \text{RoundOK} \rangle$ to $\mathcal{F}_{sync}$.
            - Send $\langle \text{RequestRound} \rangle$ to $\mathcal{F}_{sync}$,receive its response d_i.
                - If $d_i = 0$,set $\text{curView}_i := \text{curView}_i + 1$.
                - else re-execute this step.
        - Else:
            - set $\Delta_{vnum}^{i} := \Delta_{vnum}^{i} * 2$.
            - send$\langle \text{NEW} - \text{VIEW}, \text{curView}_i, i \rangle$ to $\mathcal{S}$ and Leader_i.
            - Send $\langle \text{RoundOK} \rangle$ to $\mathcal{F}_{sync}$.
            - Send $\langle \text{RequestRound} \rangle$ to $\mathcal{F}_{sync}$,receive its response d_i.
                - If $d_i = 0$,set $\text{curView}_i := \text{curView}_i + 1$. set $\text{phase}_i = \perp$.
                - else re-execute this step.

2. Else，ignore this message.

Upon receiving message $MSG\langle \text{Decide}, \perp, \text{CommitQC} \rangle$ **from** Leader_i**:**

1. Send ⟨Sleep, sid⟩ to $\mathcal{S}$ and wait for a response of the form ⟨Wake, sid, δ⟩:
    - If $\delta > \Delta_{vnum}^{j}$:
        - set $\Delta_{vnum}^{j} := \Delta_{vnum}^{j} * 2$, $\text{curView}_j := \text{curView}_j + 1$.
        - send⟨$\text{NEW} - \text{VIEW}, \text{curView}_j, j$⟩ to $\mathcal{S}$ and Leader_i.
    - else,Send ⟨timeStart, sid, δ⟩ to $\mathcal{F}_{time}$.
2. If $MATCHINGQC(\text{m.justify}, \text{Commit}, \text{curView}_j) \cap \text{phase}_j = \text{Decide}$:
    - If no ⟨timeOver, sid⟩ has been received from $\mathcal{F}_{time}$:
        - Execute new commands through m.justify.node
        - Send ⟨RoundOK⟩ to $\mathcal{F}_{sync}$.
        - Send ⟨RequestRound⟩ to $\mathcal{F}_{sync}$,receive its response d_i.
            - If $d_i = 0$,set $\text{curView}_i := \text{curView}_i + 1$,set $\text{phase}_j = \perp$.
            - else re-execute this step.
    - else:
        - set $\Delta_{vnum}^{j} := \Delta_{vnum}^{j} * 2$.
        - send⟨$\text{NEW} - \text{VIEW}, PrepareQC, \text{curView}_j, j$⟩ to $\mathcal{S}$ and Leader_i.
        - Send ⟨RoundOK⟩ to $\mathcal{F}_{sync}$.
        - Send ⟨RequestRound⟩ to $\mathcal{F}_{sync}$,receive its response d_i.
            - If $d_i = 0$,set $\text{curView}_i := \text{curView}_i + 1$.
            - else re-execute this step.
3. Else，ignore this message.