It’s not easy to relax:

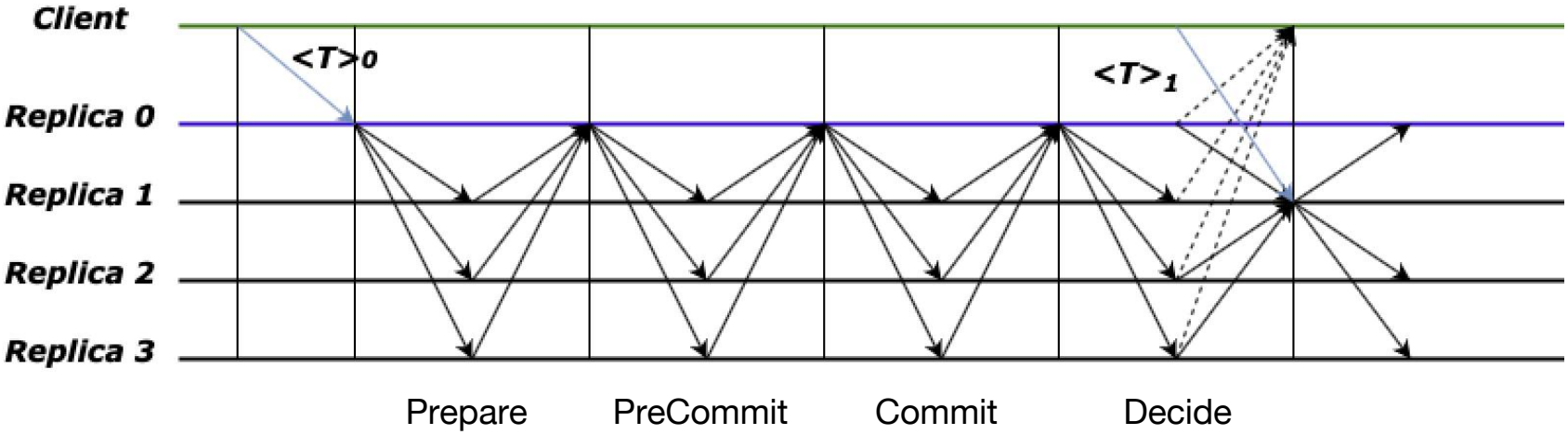
liveness in chained BFT protocols

Presenter:
Dakai Kang, Musheng He, Zizhong Li, Xiaoxing Chen, Piaopiao Long

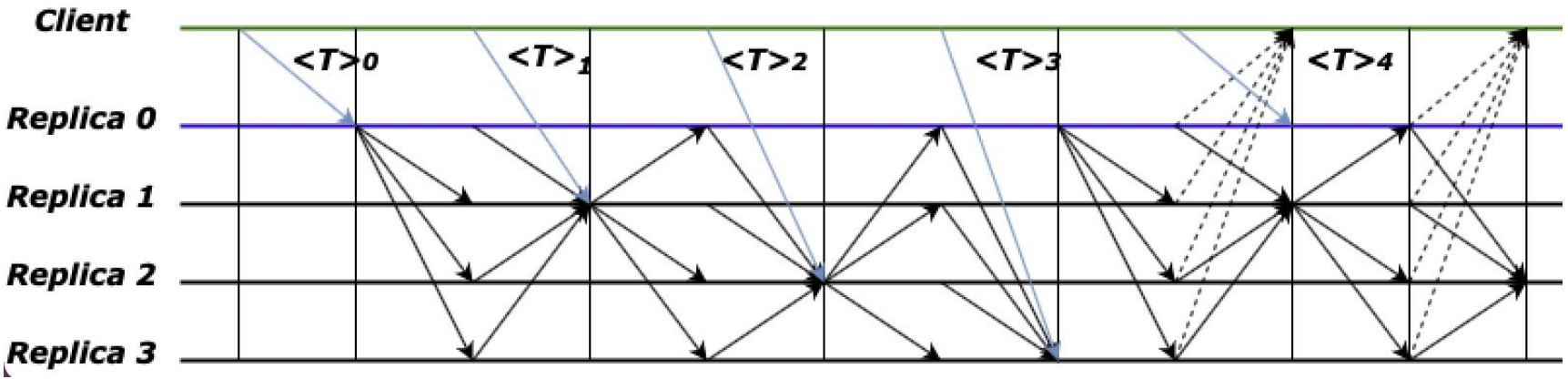
Part 1

- Basic HotStuff & Chained HotStuff
- Liveness Concern
- Siesta
- HotStuff vs Siesta

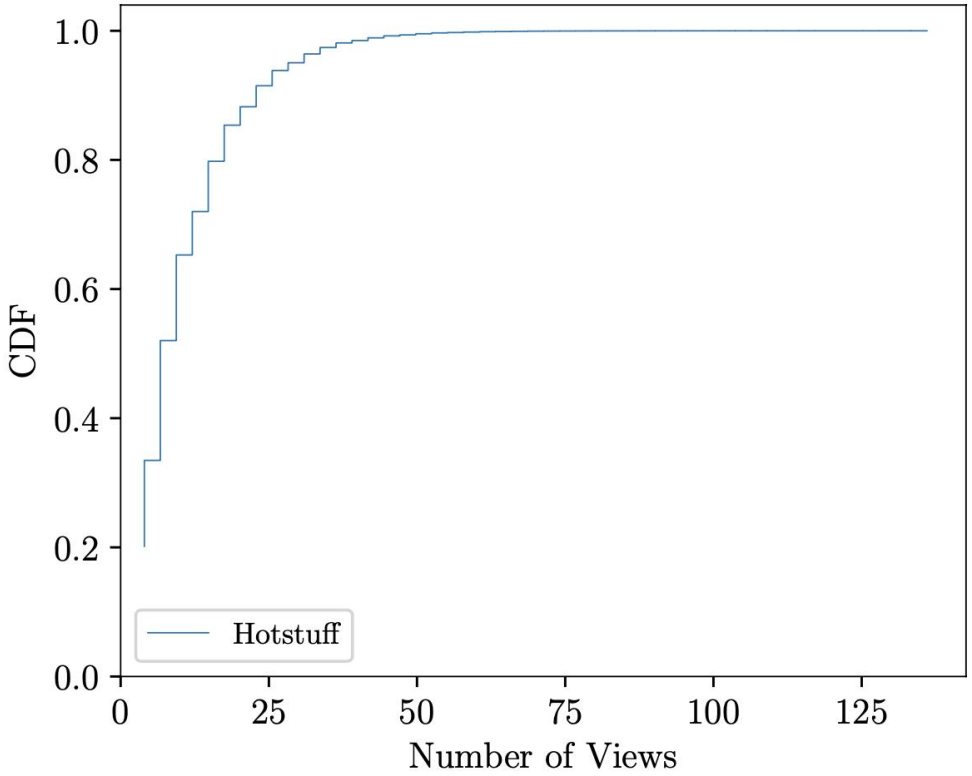
Basic HotStuff



Chained HotStuff



Liveness Concern



Consecutive-leader requirement:

- One to propose a block
- Another to collect votes and form QC

Hotstuff needs **4 non-faulty consecutive leaders** to commit!

Result:

- Averagely, 12 views to commit
- worst-case latency of 129 views

UCDAVIS

Siesta

- No protocol can commit in **non-consecutive views** in the presence of Byzantine faults (proof in Appendix C, terrible)

- In the presence of omission faults (crash fault or bad network condition), Siesta commits in **non-consecutive views** as long as one can prove that **no QC** for a conflicting block could have formed in between views.

- Detects **equivocation** and **slashes** a Byzantine leader.

HotStuff vs Siesta

- N = 3f+1
- Partial Synchrony
- Threshold Signature
- 4 consecutive non-faulty leaders
- No slashing

- N = 3f+1
- Partial Synchrony
- Digital Signature
- 3 non-consecutive non-faulty leaders
- Slashing

Hotstuff vs Siesta

1. Local State Variables

- **Hotstuff:**
  - ***viewNumber*** *: current view number*
  - ***lockedQC*** *: highest precommittedQC*
  - ***prepareQC*** *: highest preparedQC*

- **Siesta:**
  - v_r *: current view number*
  - QC_r *: the QC received with the highest view*
  - vr_r *: highest (by view) VOTE-REQ message received*
  - vp_r *: highest (by view) VOTE-RESP message sent*

UCDAVIS

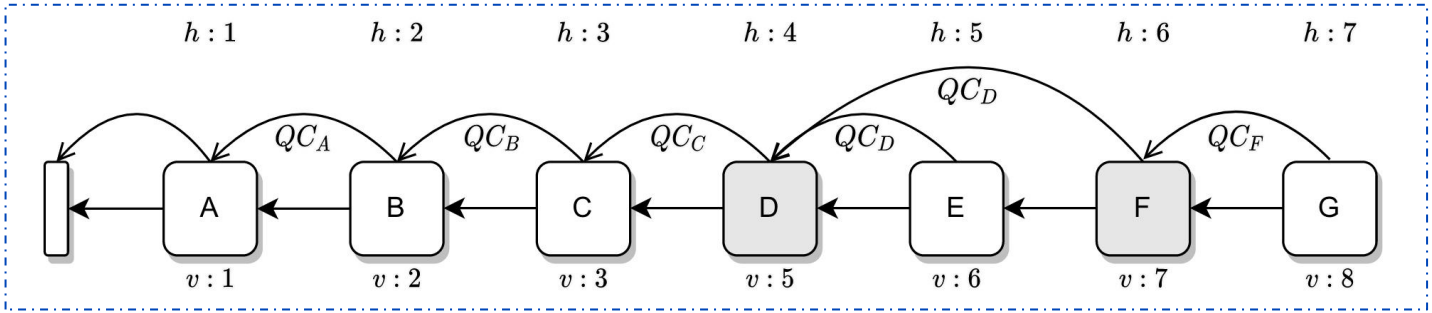
Hotstuff vs Siesta

2. Block

- h: the height of the block
- $H(B_{h-1})$: hash pointer to parent block
- b_h: batch of client transactions
- v: view in which block was proposed
- QC: QC attesting validity of parent block
- σ_l: leader signature

Additional state for a slow block:

- S: set of NEWVIEW messages
- π: optional equivocation proof



UCDAVIS

Hotstuff vs Siesta

3. *GENERIC vs VOTE-REQ*

VOTE-REQ for fast blocks

- b_h*: set of client transactions*
- ***h****: height*
- ***v****: current view*
- ***H****: hash pointer to the parent block*
- ***QC****: QC attesting the validity of the parent block*

VOTE-REQ for slow blocks

- ***S****: set of NEW-VIEW Message*
- π*: optional equivocation proof*

Hotstuff vs Siesta

4. NewView & VOTE-RESP

- **VOTE-RESP in Siesta:**
  - same as the NewView message in HotStuff
  - A vote for the block in the VOTE-REQ message received

- **NewView in Siesta:**
  - $\langle \text{NEWVIEW}, v, vr_r, vp_r, QC_r \rangle_{\sigma_r}$
  - the highest received VOTE-REQ vr_r
  - the highest sent VOTE-RESP vp_r

Part 2

- Fast View Change
- Slow View Change

View Change

Fast view change:

- As views are contiguous, the new leader is guaranteed to learn the latest certified block and can thus easily extend the chain.

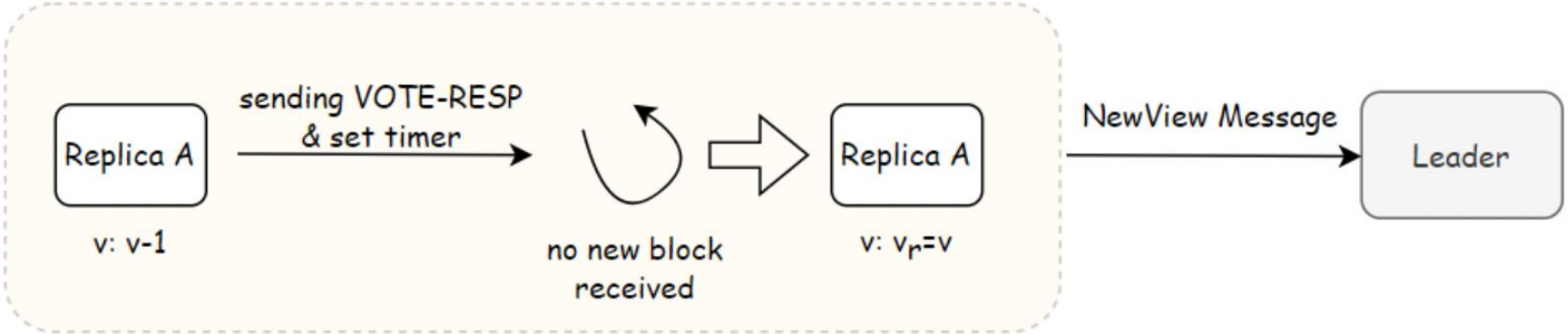
Slow view change:

- Triggered when sufficiently many (2f+1) replicas fail to receive a new block proposal before a specific timeout value
- Replicas move on to the next view and transfer necessary state information to new leader
- So it is no longer possible for the new leaders to simply extend a block, as conflicting blocks could have been proposed during views that did not complete.

Trigger a NewView

Step 1: R to L_v

Send $\langle \text{NEWVIEW}, v, QC_r, vr_r, vp_r \rangle_{\sigma_R}$



New Leader Logic:

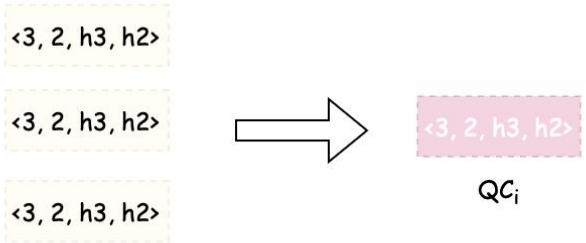
Decide the QC in VOTE-REQ

- In HotStuff, new leaders only take existing QCs into consideration
- In Siesta, new leaders attempt to generate new QCs from NEW-VIEW messages received
- We represent VOTE-RESP in NEW-VIEW in the form of a tuple:

<v, w, hv, hw>

<current view, view of parent block, block hash, parent block hash>

New Leader Logic

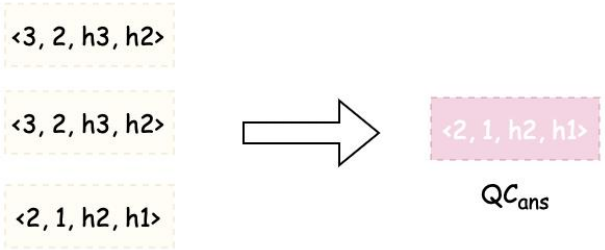


N =4, f = 1

upon event S = 2f+1 $\langle$NewView, $v, vr_r, vp_r, QC_r\rangle$ **do**
  if $2f+1$ matching vp_r **then**
    QC_i = set of matching vp_r
  else
    B_{anc} = earliest ancestor block of $2f+1$ matching vp_r
    if B_{anc} uncommitted **then**
      QC_{anc} = set of matching vp_r
    else
      $QC_{high} = max(QC_r)_{r \in R}$
      $QC_{choice} = max(QC_i, QC_{anc}, QC_{high})$
      $highVote = max(vr_r)_{r \in R}$
      if $QC_{choice} \longleftarrow highVote.B$ **then** ▷ highVote is the correct block to extend
      else
        if $highVote.B.S$ contains $n-f$ NewView messages that do not contain QC_{choice} **then** ▷ QC_{choice} could not have committed
          $QC_{choice} = highVote.QC$
        else
          $highVote = vr_r$ that contains QC_{choice}

UCDAVIS

New Leader Logic



N =4, f = 1

upon event S = 2f+1 $\langle$NewView, $v, vr_r, vp_r, QC_r\rangle$ **do**
  if $2f+1$ matching vp_r **then**
    QC_i = set of matching vp_r
  else
    B_{anc} = earliest ancestor block of $2f+1$ matching vp_r
    if B_{anc} uncommitted **then**
      QC_{anc} = set of matching vp_r
    else
      $QC_{high} = max(QC_r)_{r \in R}$
      $QC_{choice} = max(QC_i, QC_{anc}, QC_{high})$
      $highVote = max(vr_r)_{r \in R}$
      if $QC_{choice} \longleftarrow highVote.B$ **then** ▷ highVote is the correct block to extend
      else
        if $highVote.B.S$ contains $n-f$ NewView messages that do not contain QC_{choice} **then** ▷ QC_{choice} could not have committed
          $QC_{choice} = highVote.QC$
        else
          $highVote = vr_r$ that contains QC_{choice}

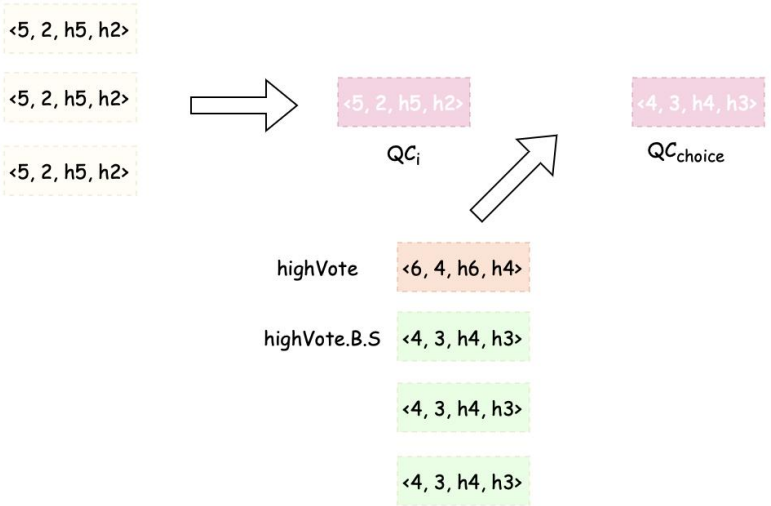
New Leader Logic

- QC_{high} is highest QC in the NEW-VIEW messages
- QC_{choice} = max(QC_i, QC_{ans}, QC_{high})

upon event S = 2f+1 $\langle$NewView, $v, vr_r, vp_r, QC_r\rangle$ **do**
  if $2f+1$ matching vp_r **then**
    QC_i = set of matching vp_r
  else
    B_{anc} = earliest ancestor block of $2f+1$ matching vp_r
    if B_{anc} uncommitted **then**
      QC_{anc} = set of matching vp_r
    else
      $QC_{high} = max(QC_r)_{r \in R}$
      $QC_{choice} = max(QC_i, QC_{anc}, QC_{high})$
      $highVote = max(vr_r)_{r \in R}$
      if $QC_{choice} \longleftarrow highVote.B$ **then** ▷ highVote is the correct block to extend
      else
        if $highVote.B.S$ contains $n - f$ NewView messages that do not contain QC_{choice} **then** ▷ QC_{choice} could not have committed
          $QC_{choice} = highVote.QC$
        else
          $highVote = vr_r$ that contains QC_{choice}

UCDAVIS

New Leader Logic



upon event S = 2f+1 $\langle$NewView, $v, vr_r, vp_r, QC_r\rangle$ **do**
  if $2f+1$ matching vp_r **then**
    QC_i = set of matching vp_r
  else
    B_{anc} = earliest ancestor block of $2f+1$ matching vp_r
    if B_{anc} uncommitted **then**
      QC_{anc} = set of matching vp_r
    else
      $QC_{high} = max(QC_r)_{r \in R}$
      $QC_{choice} = max(QC_i, QC_{anc}, QC_{high})$
      $highVote = max(vr_r)_{r \in R}$
      if $QC_{choice} \longleftarrow highVote.B$ **then** ▷ highVote is the correct block to extend
      else
        if $highVote.B.S$ contains $n-f$ NewView messages that do not contain QC_{choice} **then** ▷ QC_{choice} could not have committed
          $QC_{choice} = highVote.QC$
        else
          $highVote = vr_r$ that contains QC_{choice}

New Leader Logic

Propose a new block extending from the block of QC_{choice}.

b_h = batch of transactions
$votereqs$ = set of vr_r messages with the same view $vr_r.B.v$ but different $vr_r.B$
if $votereqs \neq \perp$ **then** ▷ Equivocation proof
    $\pi = votereqs$
$B = (h, H(highVote.B), b_h, v, QC_{choice}, S, \pi)$
σ_{L_v} = signature on B
Multicast $\langle$VOTE-REQ, $B\rangle_{\sigma_{L_v}}$ to all replicas

Replica Logic

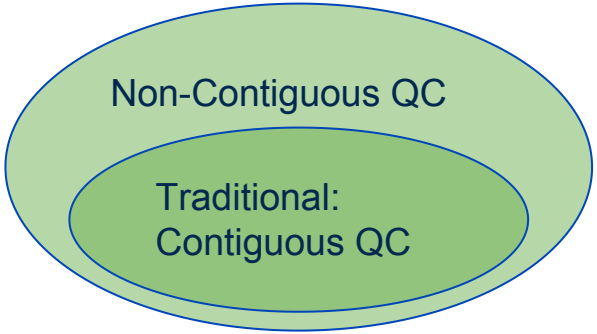
upon event $\langle$VOTE-REQ, $B\rangle$ **do**
  if $v == B.v$ **then**
    if $B.\pi \neq \perp$ **then**
      Ignore messages from identified faulty leader
    goto line 4 using $B.S$ ▷ Check if the leader did the view change correctly
    if result matches B **then**
      $vr_r = \langle$VOTE-REQ, $B\rangle$
      $QC_r = B.QC$
      Send $\langle$VOTE-RESP, $B\rangle_{\sigma_r}$ to L_{v+1}
      $vp_r = \langle$VOTE-RESP, $B\rangle_{\sigma_r}$
      $v_r = v_r + 1$
    else
      Ignore message
  else
    Ignore message
end event

UCDAVIS

Part 3

- Commit Rule
  - Contiguous QCs
  - Non-contiguous QCs: No-QC proof
- Slashing
- Advantage

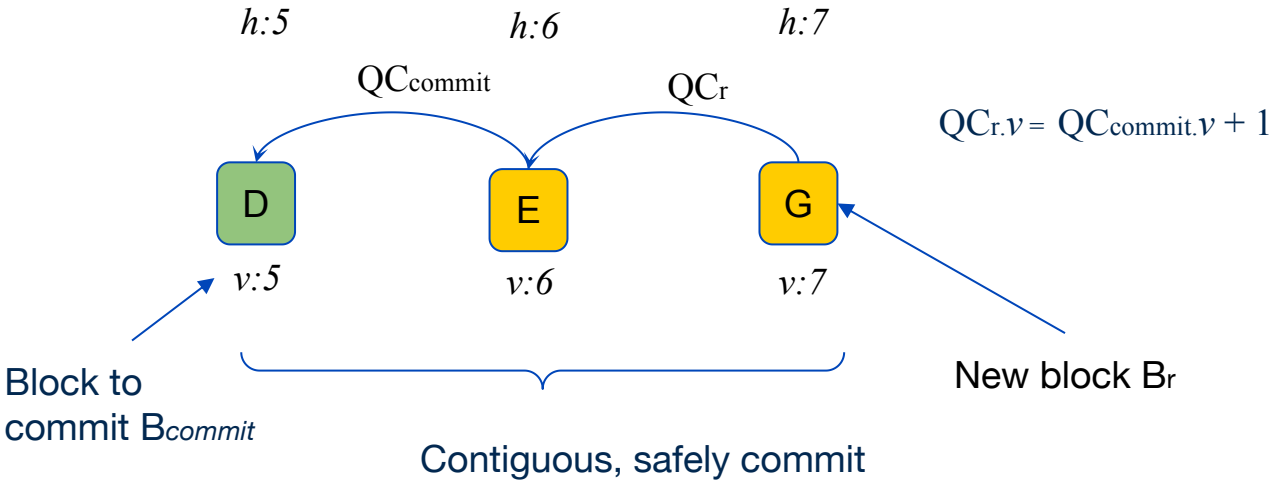
Commit Rule



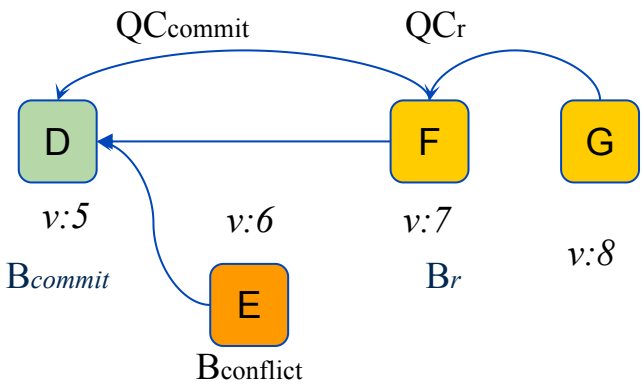
Siesta Support

Traditional way: contiguous QC

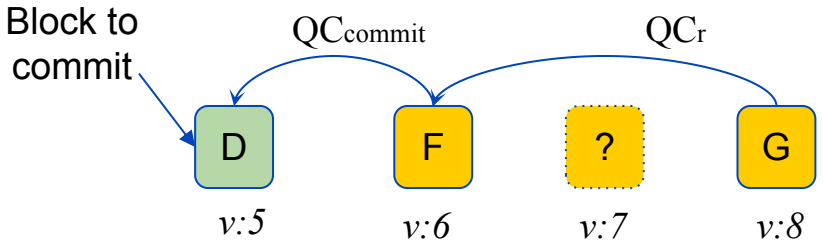
- **1st QC:** Certifices Block and achieve agreements across replicas
- **2nd QC**: achieves persistence across views



Non-Contiguous QC



Conflict

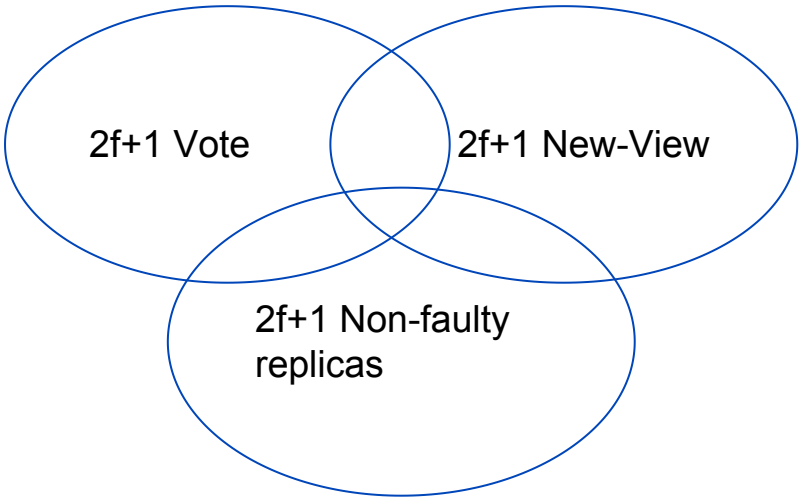


No conflict

$$QCr.v > QCcommit.v + 1$$

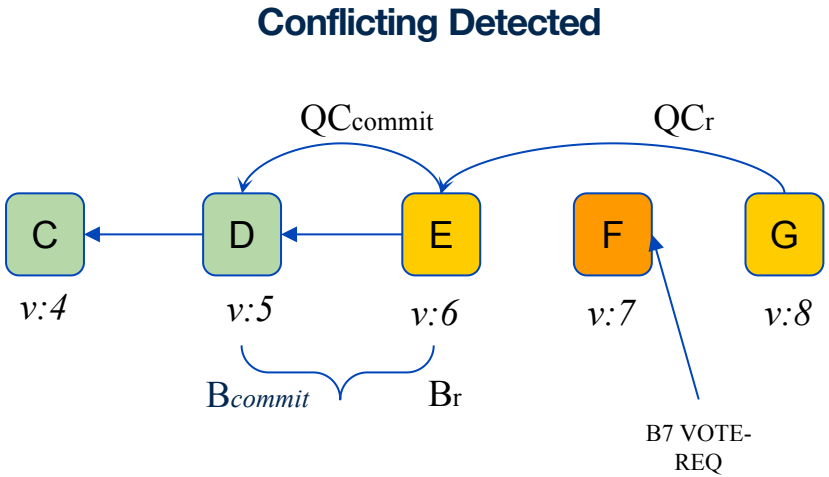
UCDAVIS

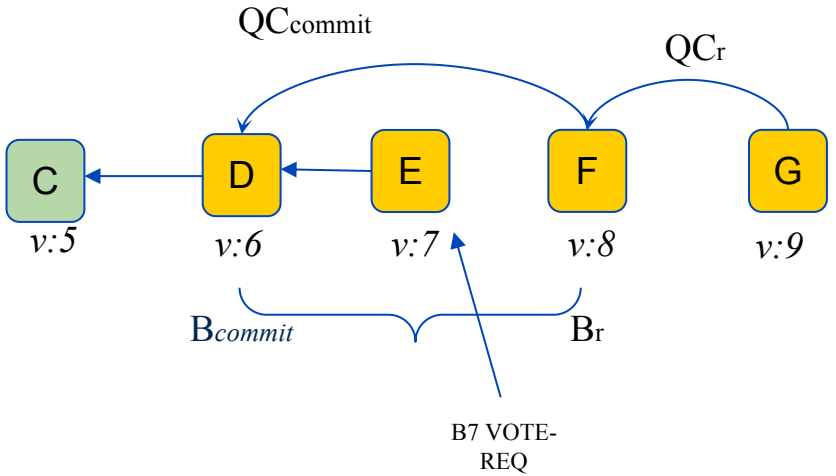
No-QC Proof in Detail



- It is safe to commit a block in non-consecutive views as long as one can prove that no QC for a conflicting block could have formed in between views. Siesta carefully designs such proofs to be efficient.
- If a QC for $B_{conflict}$ forms, all subsequent leaders will observe **at least one Vote-req message** for $B_{conflict}$
- Check every block on the chain between B_r (included) and B_{commit} to determine whether such a conflicting Vote-req exists.

How No-QC Works



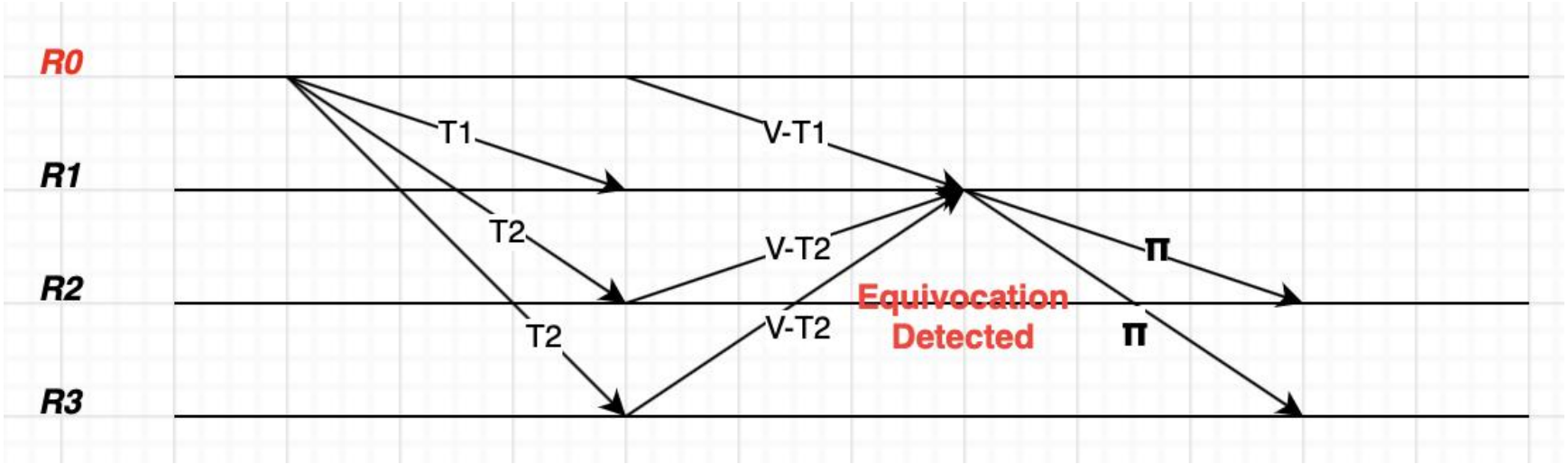


- *r* checks every block on the chain between B_r and B_{commit} to determine whether such a conflicting VOTE-REQ exists.

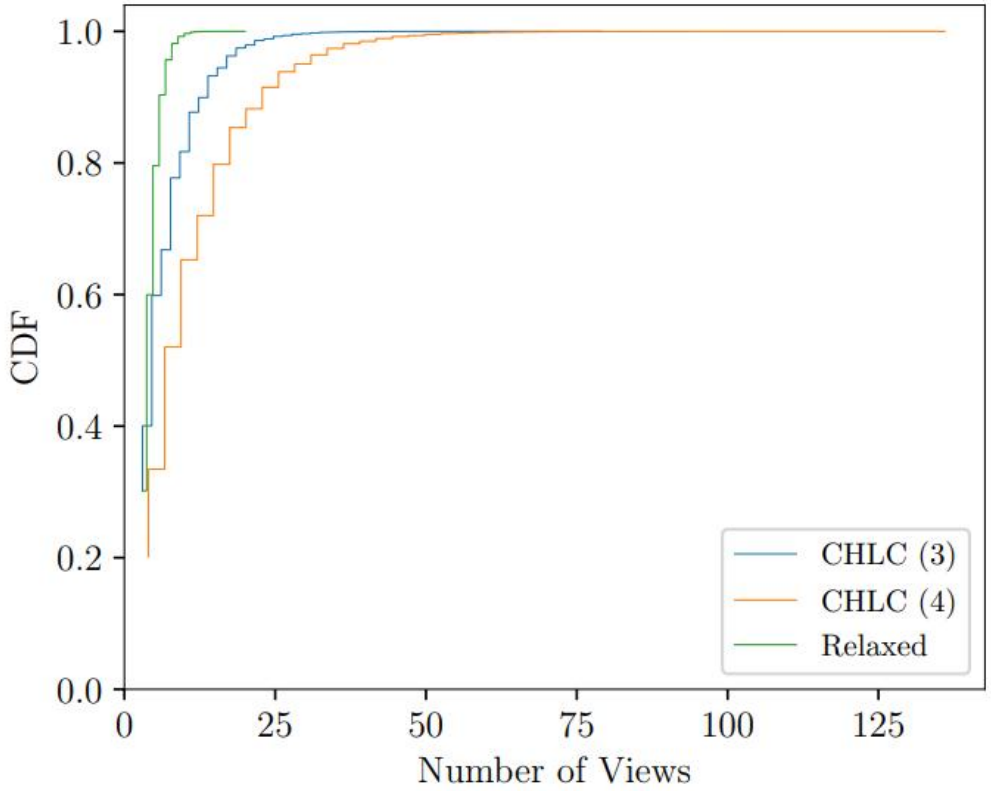
No B' VOTE-REQ => No Conflicting

UCDAVIS

Equivocation Proof & Slashing



The Advantage of Siesta



Average commit latency:

- Siesta: 4.5
- CHLC(3): about 7 rounds
- CHLC(4): 12 rounds

Worst-case commit latency:

- Siesta: 18
- CHLC(3): 76
- CHLC(4): 129

UCDAVIS

It’s not easy to relax:

liveness in chained BFT protocols

Q & A

Presenter:
Dakai Kang, Musheng He, Zizhong Li, Xiaoxing Chen, Piaopiao Long