Bchain Byzantine Replication with high throughput and embedded reconfiguration

Sisi Duan, Hein Meling, Sean Peisert, and Haibin Zhang

Presented by Ruben Romero

BChain Protocols

- Bchain3:
  - 3f+1 replicas
  - Sub protocols: (1) Chaining, (2) Re-chaining, (3) View Change, (4) Checkpoint and (5) Reconfiguration.

- BChain5:

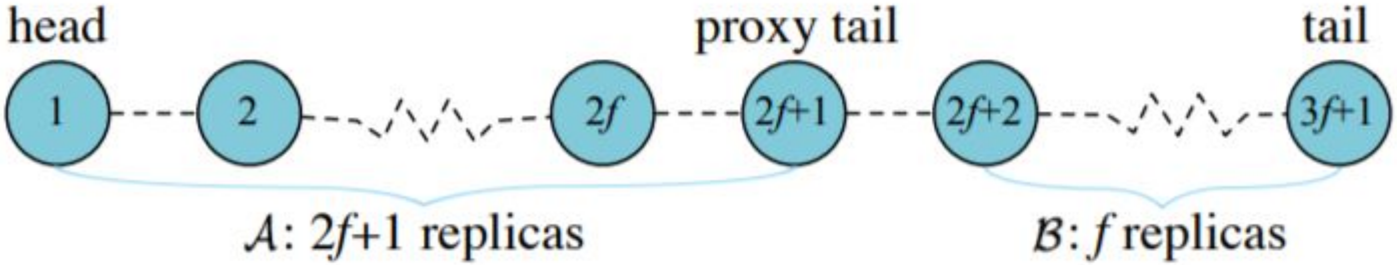
  - 5f+1 replicas

  - No Reconfiguration protocol

BChain

- Safety:
  - It is hold in any asynchronous environment where messages may be delayed, dropped, or deliver out of order
- Liveness
  - Assure assuming that synchrony holds after some unknown stabilization time

Bchain



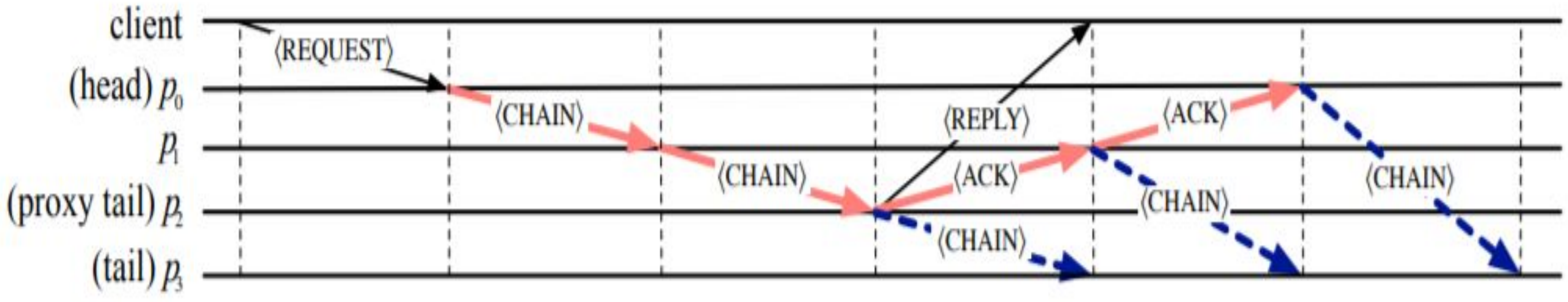
For each replica pj we define P(j), set predecessor, and S(pj), set successor, for replicas in the set A as:

-P(pj): if j < f+1 then P(pj) = {ph,p1,...,pj-1}, else P(j) = {pj-f-1,....,pj-1}

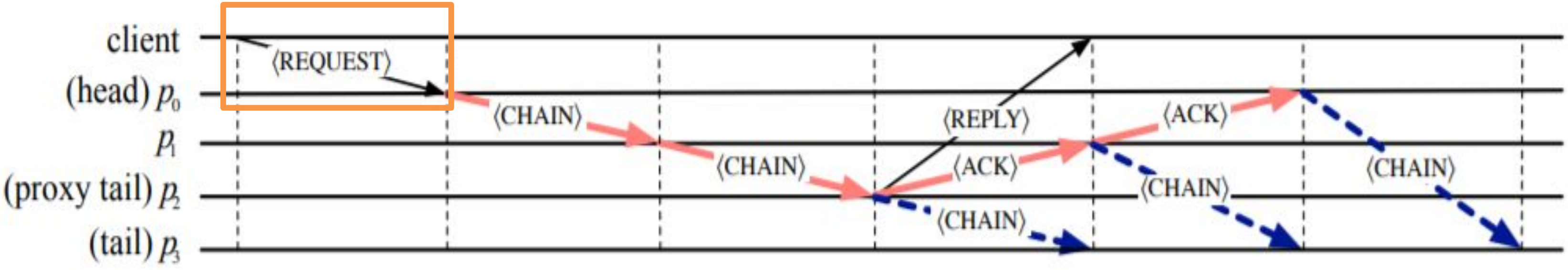
-S(pj): if 2f+1 < f < f+1 then S(pj) = {pj+1,....,p2f+1}, else S(j) = {pj+1,... pj+f+2 }

Chaining Protocol

- Orders clients requests



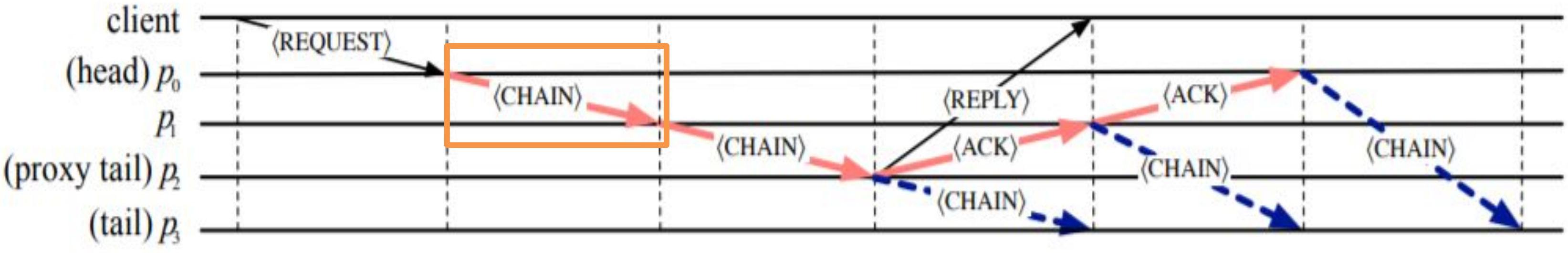
Chaining Protocol: Step 0



-Client c sends a request <Request, o, T, c> to the head p_h.

-o: state machine operation
-T: Timespan
-c: client id

Chaining Protocol: Step 1



-Head receives <Request, o, T, c> from C

-Head sends <Chain, v, ch, N, m, c, H, R, Λ> to its successor p1

-v: View number

-ch: Number of rechainning

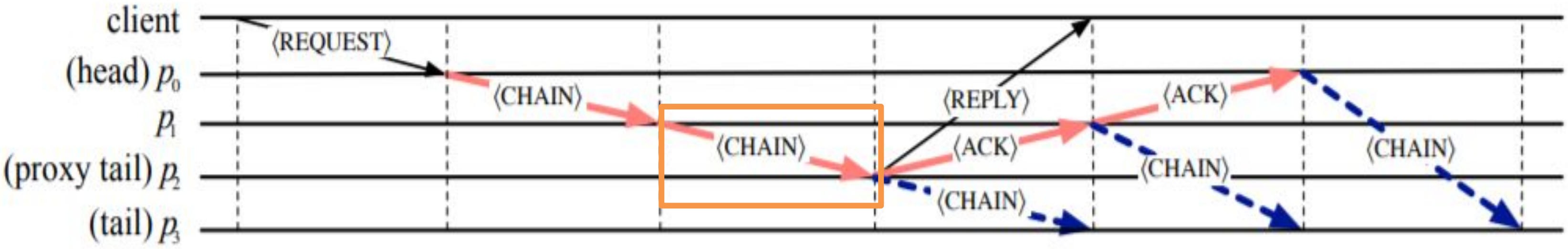
-c: client id

-H: Hash of its execution history

-R: Hash of reply r to the client containing the execution result

-Λ:: Current chain order

Chaining Protocol: Step 2



-Replica p_j receives <Chain, v, ch, N, m, c, H, R, Λ> from his predecessor p_{j-1} that contains valid signatures from P(p_j)

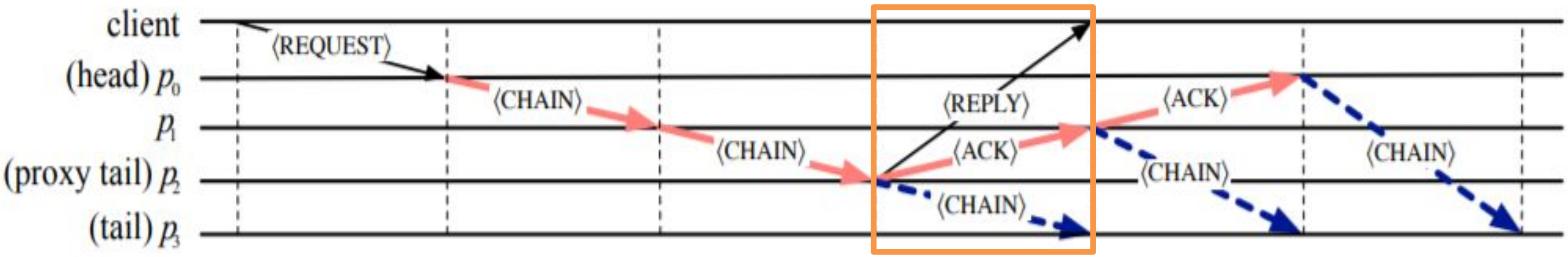
-if $p_j \in$ f+1 last replicas in A it updates H and R

-It appends its signature

-Send <Chain, v, ch, N, m, c, H, R, A> to its successor p_{j+1}.

-Set a timer Δ_1. Expecting ACK or SUSPECT message

Chaining Protocol: Step 3



-Proxy tail p2f+1 receives <Chain, v, ch, N, m, c, H, R, Λ> from its predecessor p2f that contain valid signatures from P(p2f+1)

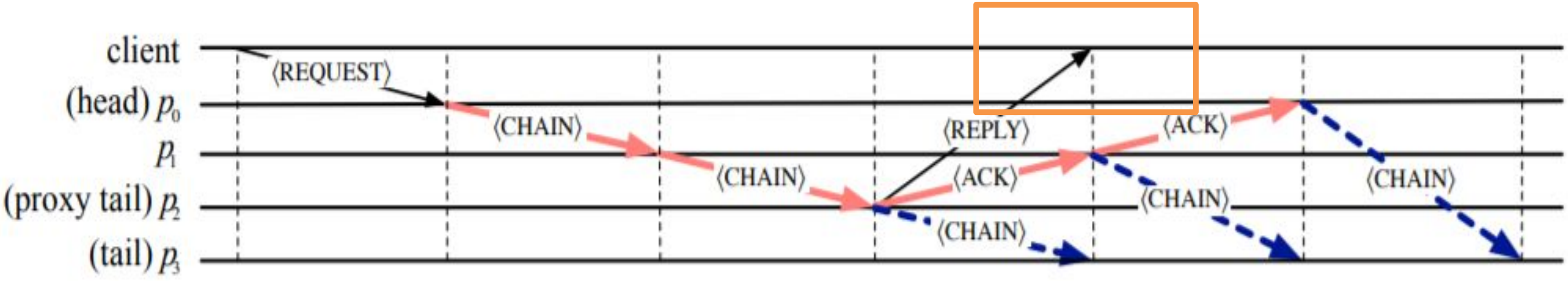
-Updates H and R and appends its signature.

-Sends reply to client

-Sends <ACK,v, ch, N, m, c, H, R, Λ> to its p2f

-Sends <Chain, v, ch, N, m, c, H, R, A> to its all replicas in B

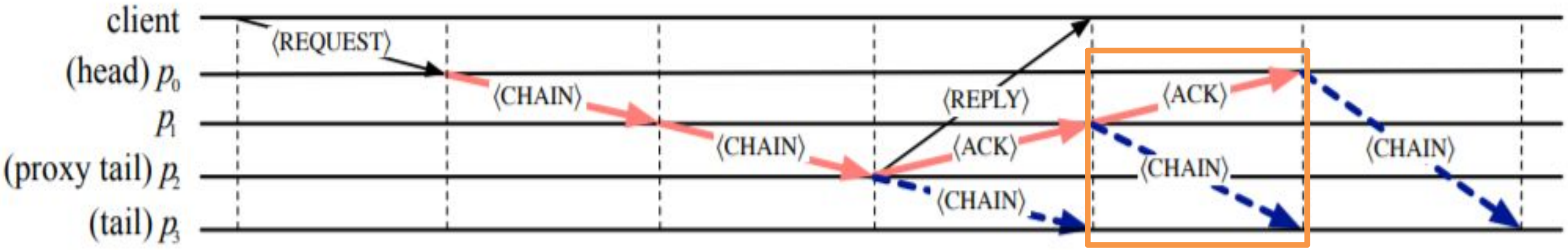
Chaining Protocol: Step 4



-Client complete the request if it receives reply from proxy that contains the signatures of last f+1 replicas in A.

-Otherwise. It retrasmites the request to all replicas.

Chaining Protocol: Step 5



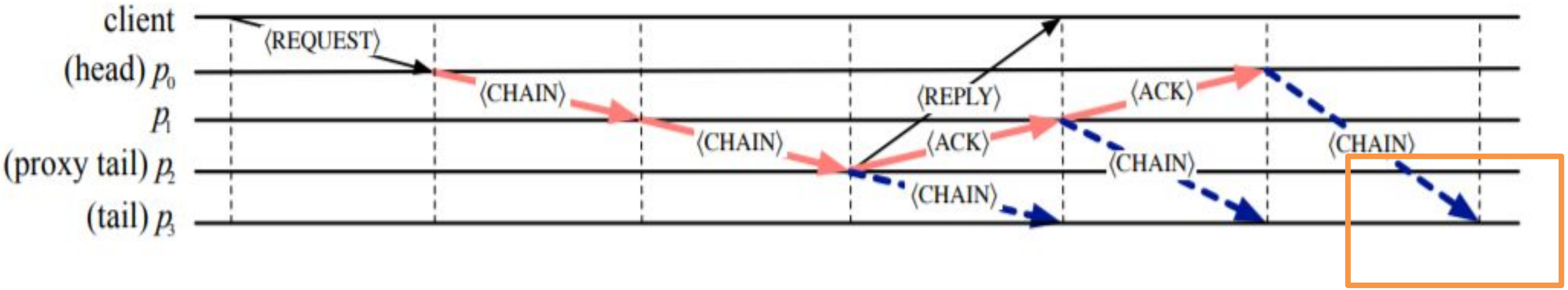
-Replica pj recieves <ACK,v, ch, N, m, c, H, R, Λ> from its successor pj+1. (ACK message contains valid signatures from S(pj)). Thus, it commits the request.

-Appends its signature.

-Sends <ACK,v, ch, N, m, c, H, R, Λ> to its predecessor pj-1.

-Sends <Chain, v, ch, N, m, c, H, R, Λ> to all replicas in B

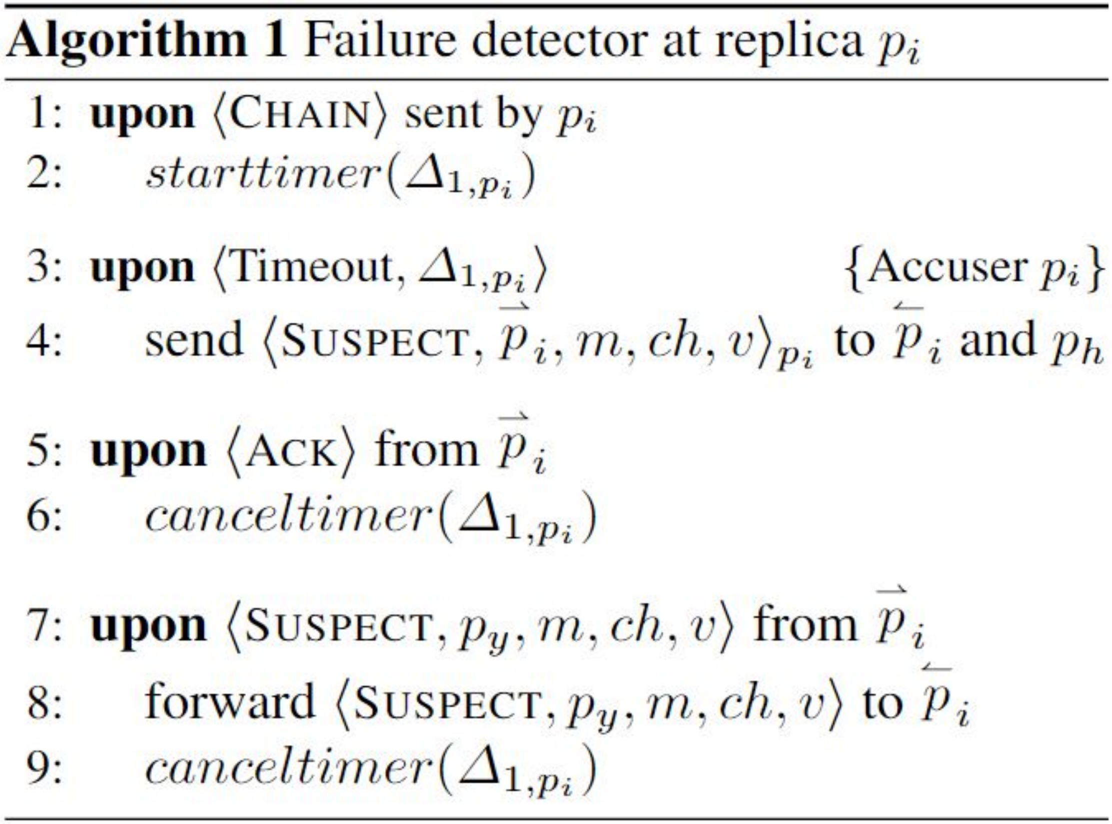
Chaining Protocol: Step 6



-Replicas in B collect f+1 CHAIN matching messages

-Then execute and commit the operation

Rechaining Protocol- Failure detector



Algorithm 1 Failure detector at replica p_i

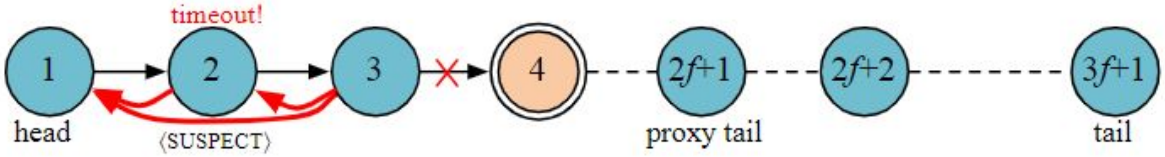
```
upon ⟨CHAIN⟩ sent by $p_i$
    $starttimer(\Delta_{1,p_i})$

upon ⟨Timeout, $\Delta_{1,p_i}$⟩                {Accuser $p_i$}
    send ⟨SUSPECT, $\overrightarrow{p}_i, m, ch, v$⟩$_{p_i}$ to $\overleftarrow{p}_i$ and $p_h$

upon ⟨ACK⟩ from $\overrightarrow{p}_i$
    $canceltimer(\Delta_{1,p_i})$

upon ⟨SUSPECT, $p_y, m, ch, v$⟩ from $\overrightarrow{p}_i$
    forward ⟨SUSPECT, $p_y, m, ch, v$⟩ to $\overleftarrow{p}_i$
    $canceltimer(\Delta_{1,p_i})$
```

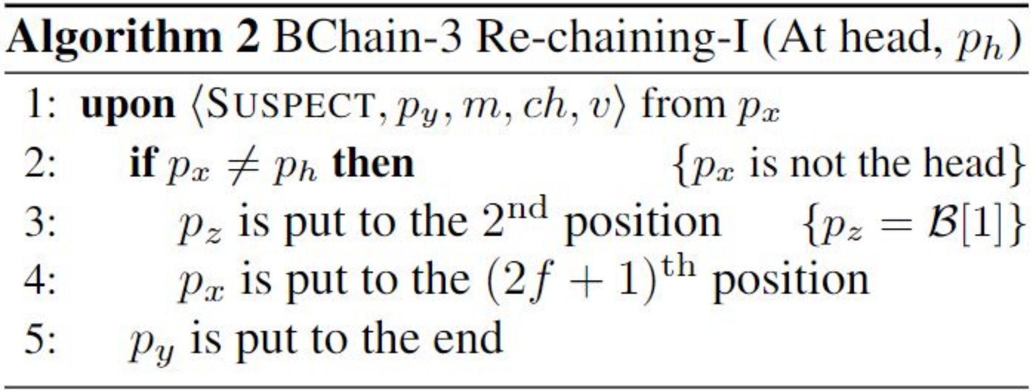
Head: Handling a suspect message:

- -increasing ch
- -new Λ
- -sending chain message

Forward also to the head

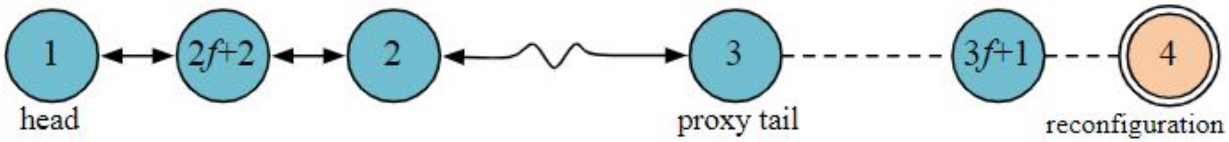
Rechaining Protocol



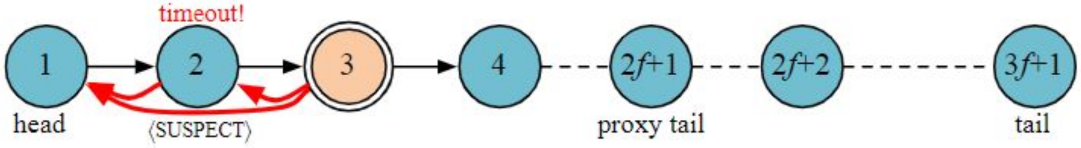


Algorithm 2 BChain-3 Re-chaining-I (At head, p_h)

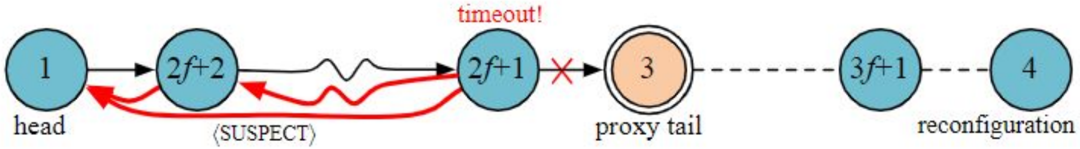
```
**upon** $\langle \textsc{Suspect}, p_y, m, ch, v \rangle$ from $p_x$
    **if** $p_x \neq p_h$ **then**                              $\{p_x$ is not the head$\}$
        $p_z$ is put to the $2^{\text{nd}}$ position            $\{p_z = \mathcal{B}[1]\}$
        $p_x$ is put to the $(2f+1)^{\text{th}}$ position
    $p_y$ is put to the end
```



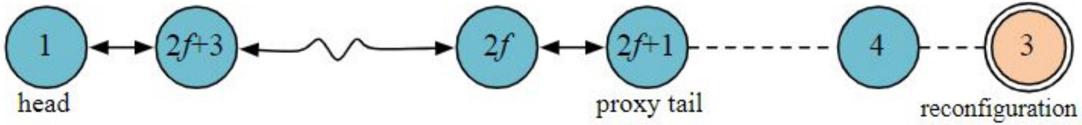
Rechaining Protocol



(a) p_3 generates a ⟨Suspect⟩ message to *maliciously* accuse p_4

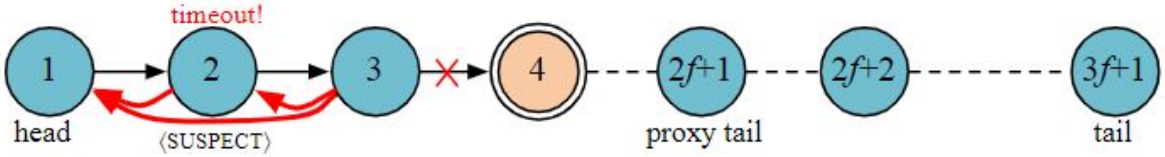


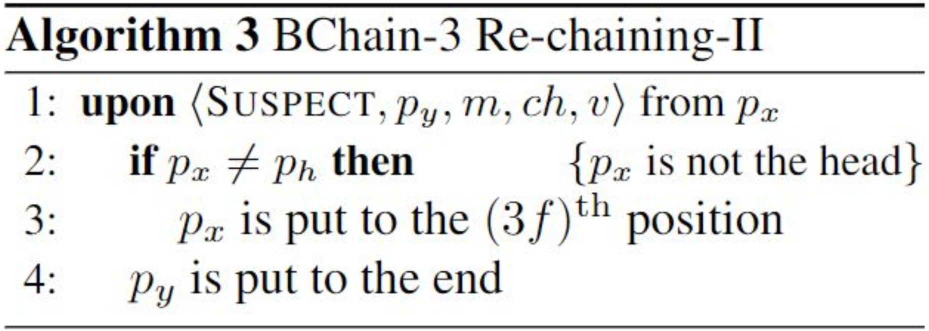
(b) p_{2f+1} generates a ⟨Suspect⟩ message to accuse p_3



(c) p_3 is moved to the tail and reconfigured

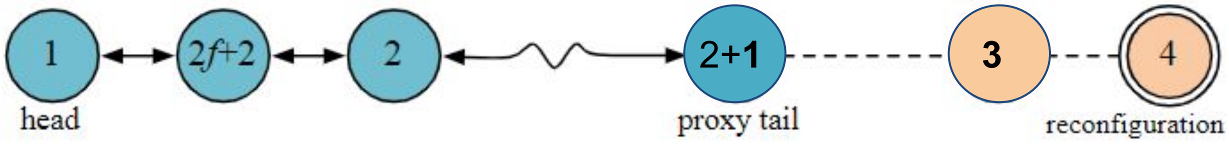
Rechaining Protocol





Algorithm 3 BChain-3 Re-chaining-II

```
upon ⟨SUSPECT, $p_y$, $m$, $ch$, $v$⟩ from $p_x$
    if $p_x \neq p_h$ then                {$p_x$ is not the head}
        $p_x$ is put to the $(3f)^{\text{th}}$ position
    $p_y$ is put to the end
```



Time setup and preventing performance attacks

Time setup:

$\Delta_{1,i}$ for each replica i= F(Δ1,i,l_i) such that if i = 0 , l_h = 1 and Δ1,h = F(Δ1,1) = Δ1,

If i = 2f+1, l_p = 2f+1 and Δ1,2f+1 = F(Δ1,2f+1) = 0

Performance threshold

$\Delta'_{1,pi} < \Delta_{1,pi}$

If average time answer is higher that $\Delta'_{1,pi}$. Replica starts suspect procedure

View Change protocol

1- Select a new head when the current one is deemed faulty

2- Adjust timers to ensure eventual progress

View Change protocol

1- Select a new head when the current one is deemed faulty:

A correct replica votes for VIEWCHANGE if:

1- It suspects the head to be faulty.

2- It receives f+1 <VIEWCHANGE> messages.

View Change protocol

1- Select a new head when the current one is deemed faulty:

If a replica votes for a VIEWCHANGE:

-Move to a new view

-Send <VIEWCHANGE,......> to all replicas

-Stop receiving messages except:

<CHECKPOINT>, <NEWVIEW>, and <VIEWCHANGE>

View Change protocol

1- Select a new head when the current one is deemed faulty:

When new head collect 2f+1 <VIEWCHANGES>:

-Send <NEWVIEW,......, newɅ, set valid viewChange messages, set of CHAIN messages> to all replica

-In the new Ʌ, the previous head was moved to the end of the chain

View Change protocol

2- Adjust timers to ensure eventual progress:

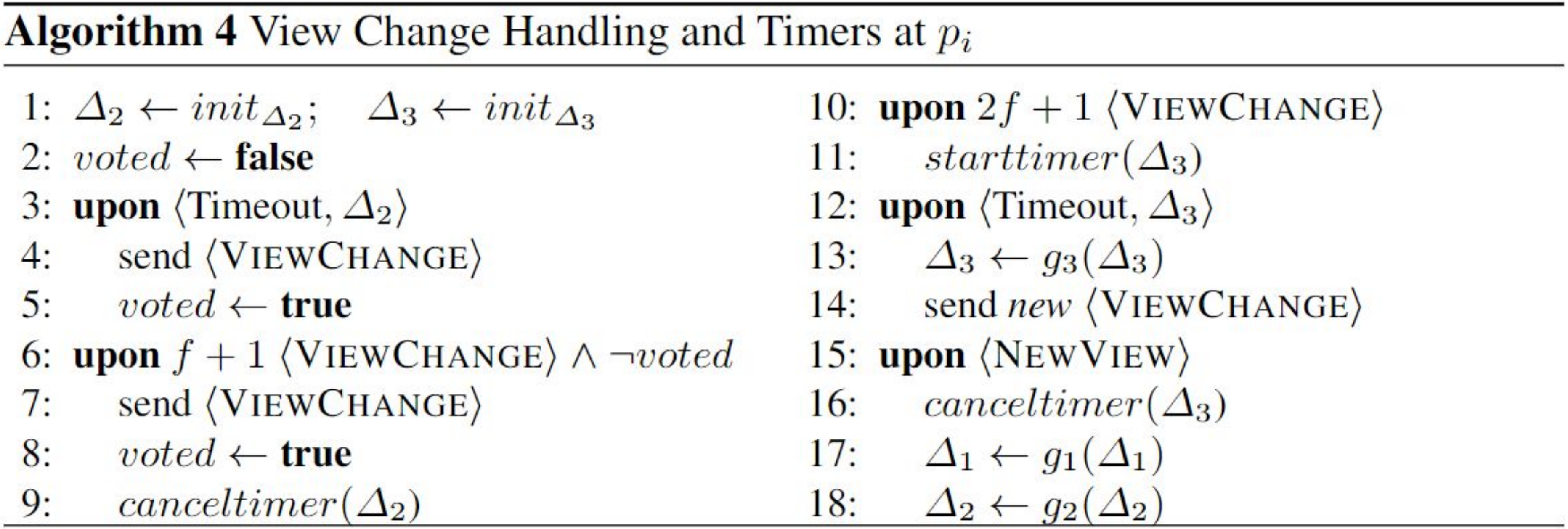
Δ1 = Timer for rechaining

Δ2 = Timer for current view when replica is waiting for a request to be committed

Δ3 = Timer for new view

View Change protocol

2- Adjust timers to ensure eventual progress:



Algorithm 4 View Change Handling and Timers at p_i

```
$\Delta_2 \leftarrow init_{\Delta_2}; \quad \Delta_3 \leftarrow init_{\Delta_3}$
$voted \leftarrow$ **false**
**upon** ⟨Timeout, $\Delta_2$⟩
    send ⟨VIEWCHANGE⟩
    $voted \leftarrow$ **true**
**upon** $f + 1$ ⟨VIEWCHANGE⟩ $\wedge \neg voted$
    send ⟨VIEWCHANGE⟩
    $voted \leftarrow$ **true**
    $canceltimer(\Delta_2)$
**upon** $2f + 1$ ⟨VIEWCHANGE⟩
    $starttimer(\Delta_3)$
**upon** ⟨Timeout, $\Delta_3$⟩
    $\Delta_3 \leftarrow g_3(\Delta_3)$
    send *new* ⟨VIEWCHANGE⟩
**upon** ⟨NEWVIEW⟩
    $canceltimer(\Delta_3)$
    $\Delta_1 \leftarrow g_1(\Delta_1)$
    $\Delta_2 \leftarrow g_2(\Delta_2)$
```

Reconfiguration protocol

It is a general technique, often abstract as stopping the current state machine and restarting with a new set of replicas.

BChain reconfiguration concerns with re-chainning to replaces faulty replicas with new ones.

Checkpoint Protocol

- Similar to the PBFT
- It is used to bound the growth of message log and reduce the cost of view changes

Questions