Viewstamp Replication

Zhaoguo Wang

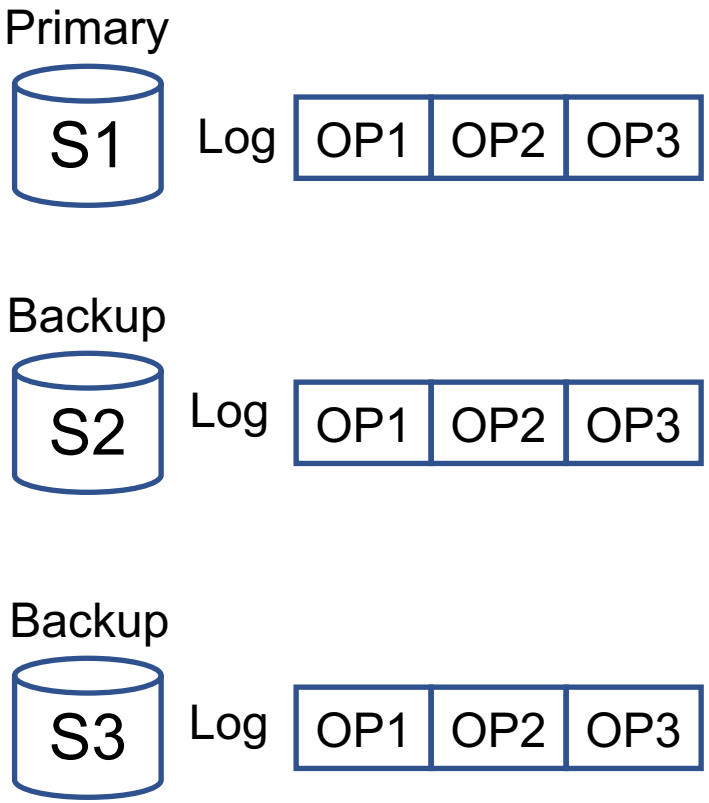
Some slides are adapted from Prof. Jinyang Li’s DS class

Viewstamp replication

A landmark work: The first (together w. Paxos) to handle failure correctly.

- Original paper Oki and Liskov, 1988
- Viewstamp revisited: Liskov and Cowling, 2012

VR overview: state machine replication



Correctness: servers execute the same sequence of log

- Operations must be deterministic

⇩

No two different ops are committed at the same log position

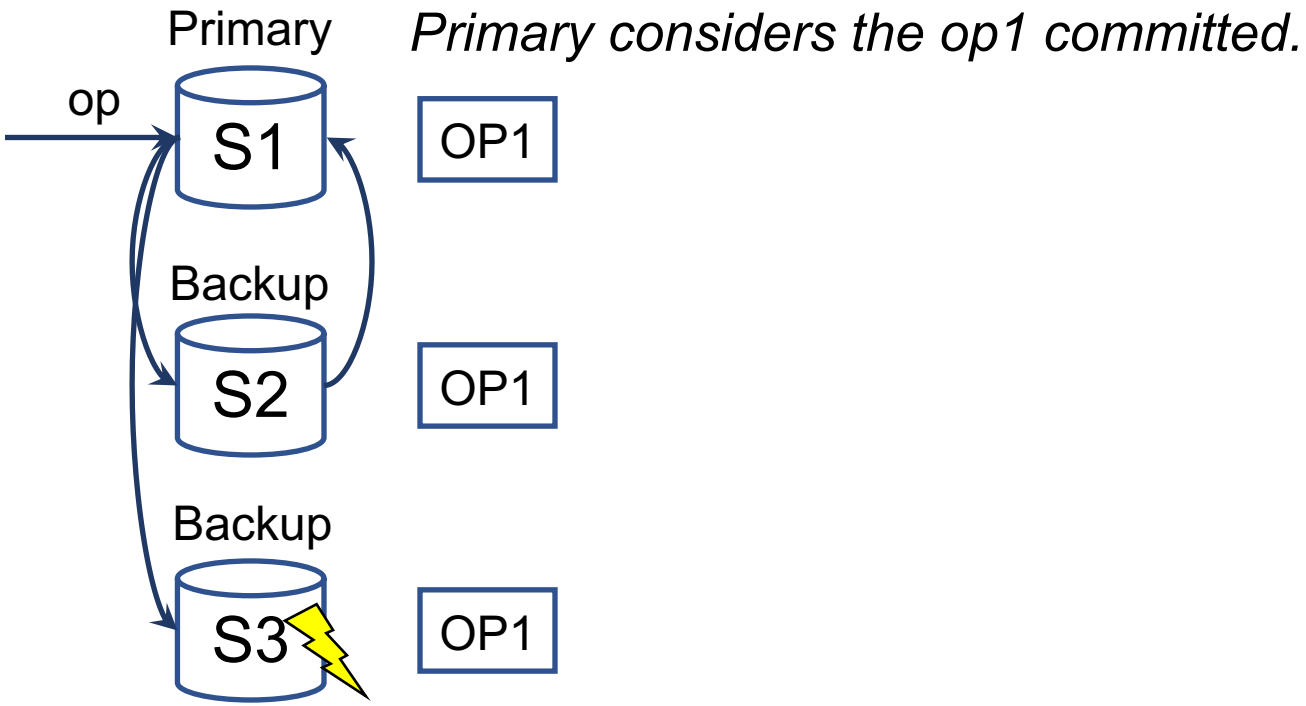
⇩

All committed ops in view V - 1 must be known to primary in view V

⇩

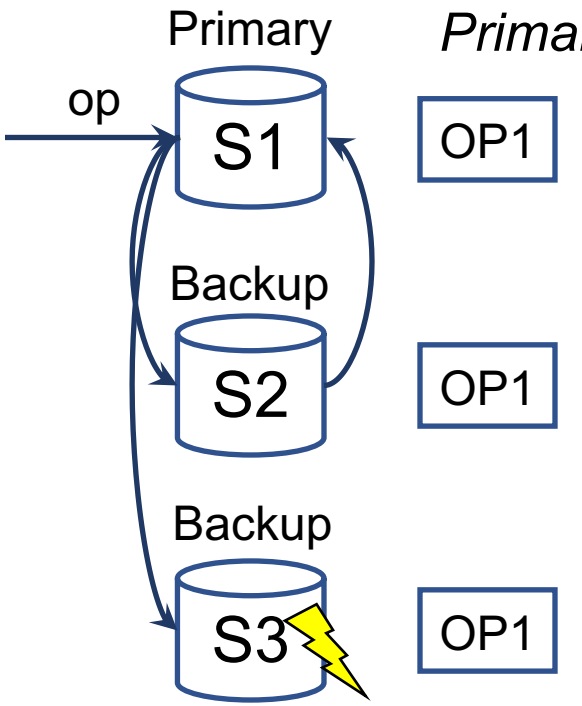
1. An op is “committed” if it is replicated by majority of nodes.
2. View v is only active after v’s primary has learned the log state of majority of nodes (at earlier views).

Normal Process



Primary considers the op1 committed.

Normal Process



Primary considers the op1 committed.

Question from Xingrong Wu:

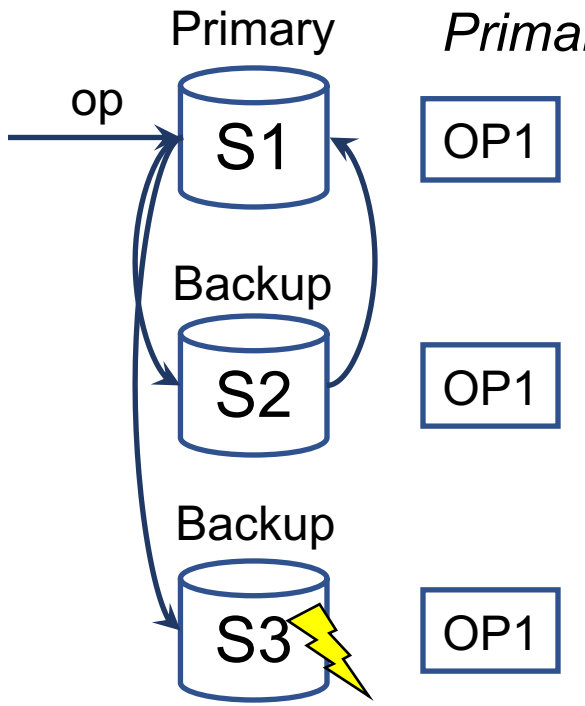
What if the primary cannot collect replies from a majority (F+1 / 2F + 1)?

Case 1. There are more than F + 1 failure

⇩

The system will stuck

Normal Process



Primary considers the op1 committed.

Question from Xingrong Wu:

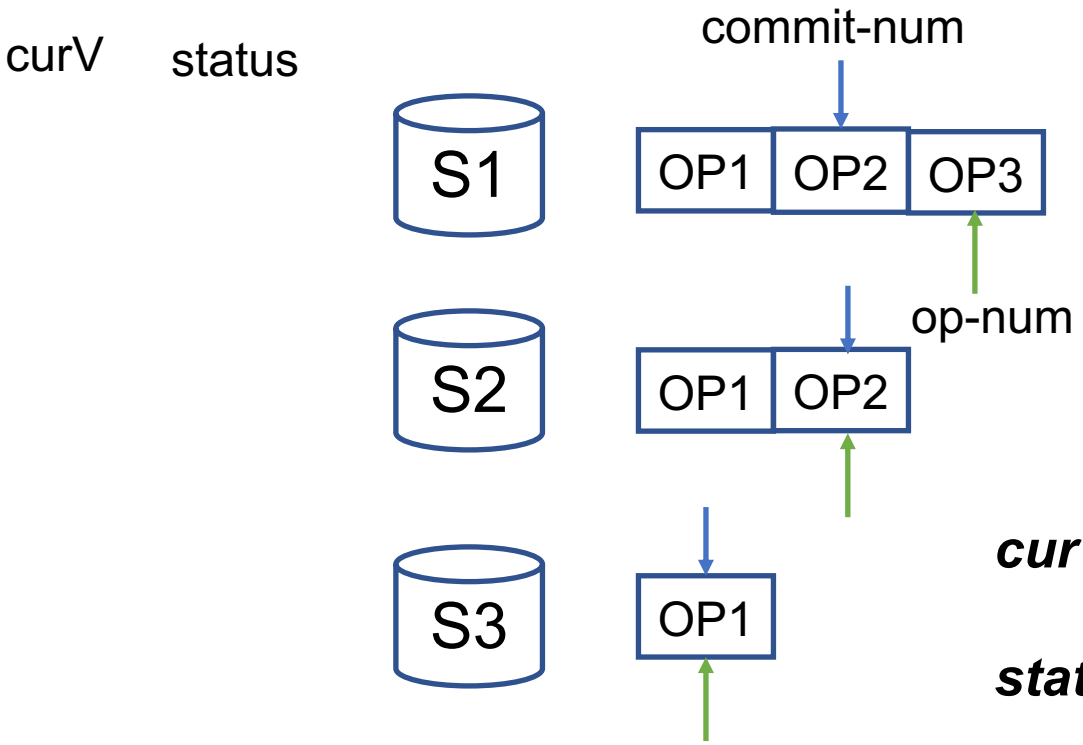
What if the primary cannot collect replies from a majority (F + 1 / 2F + 1)?

Case 2. There still have more than F + 1 alive

⇩

A backup starts a view change, the old primary will catch up after the network resumes.

Basic VR protocol



curV: current view number

status: NORMAL, VIEW-CHANGE, or RECOVERY

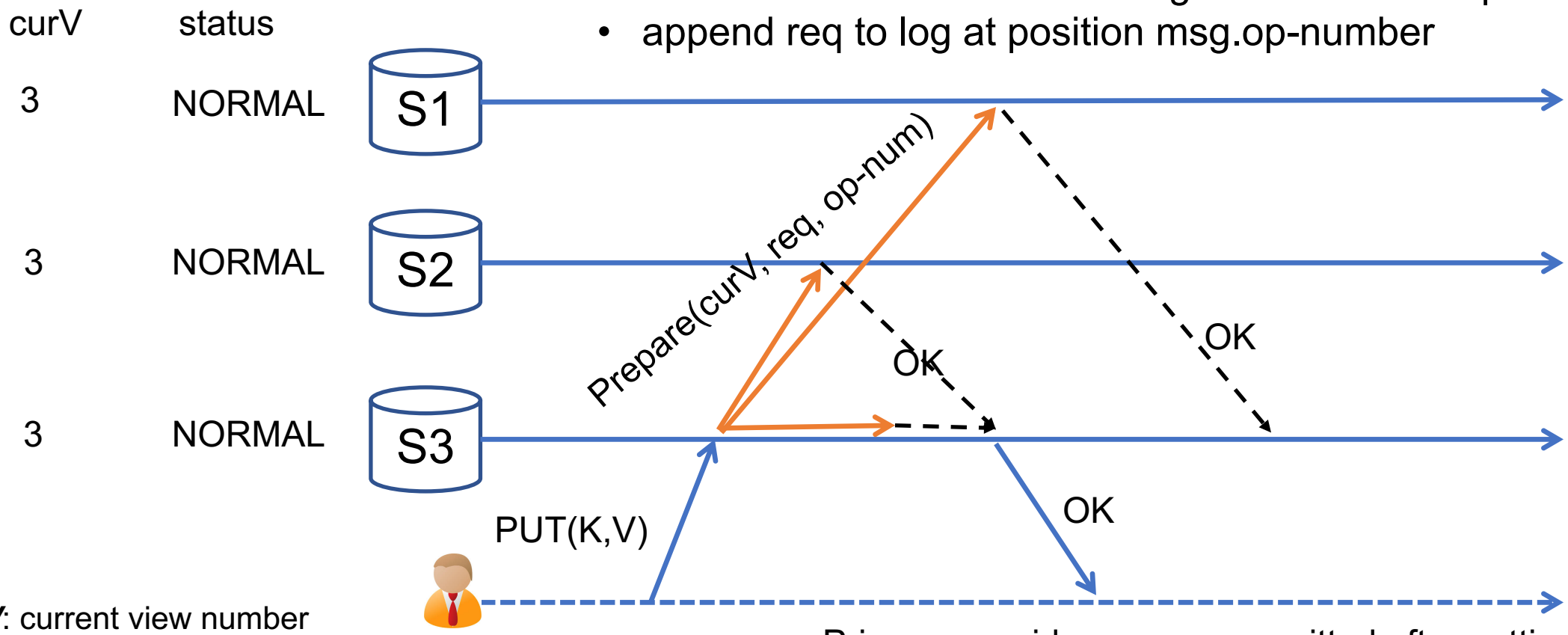
op-num: the number of last op in the log

commit-num: the number of last committed op

Basic VR protocol – Normal Processing

On receiving Prepare msg, server must:

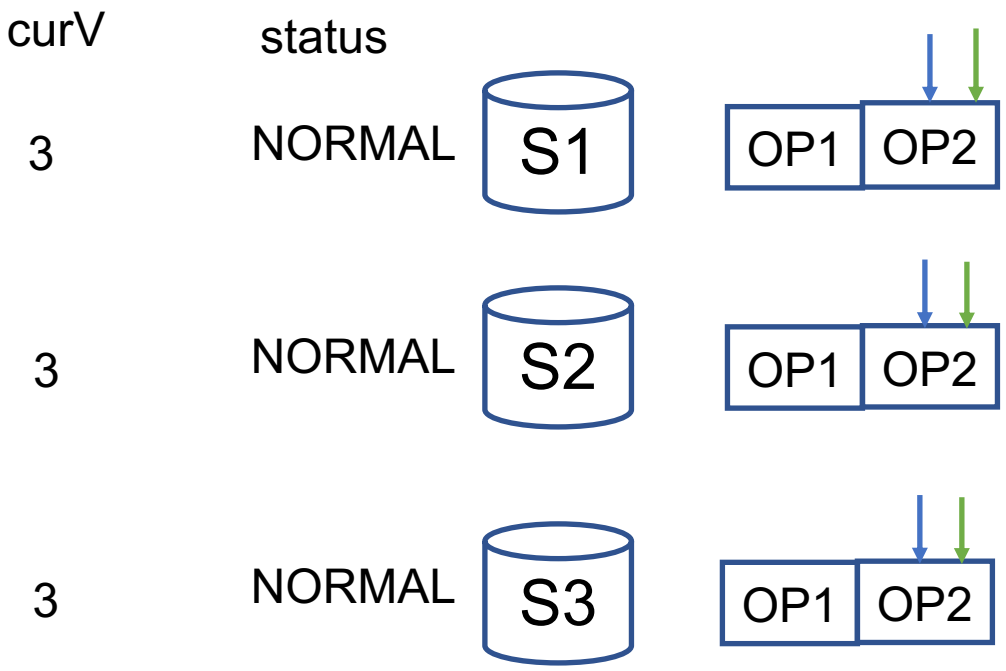
- ignore if its curV != msg.curV or status != NORMAL
- wait until it has entries in its log for all earlier requests
- append req to log at position msg.op-number



curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

Primary considers a req committed after getting majority OKs.

Basic VR protocol – Normal Processing

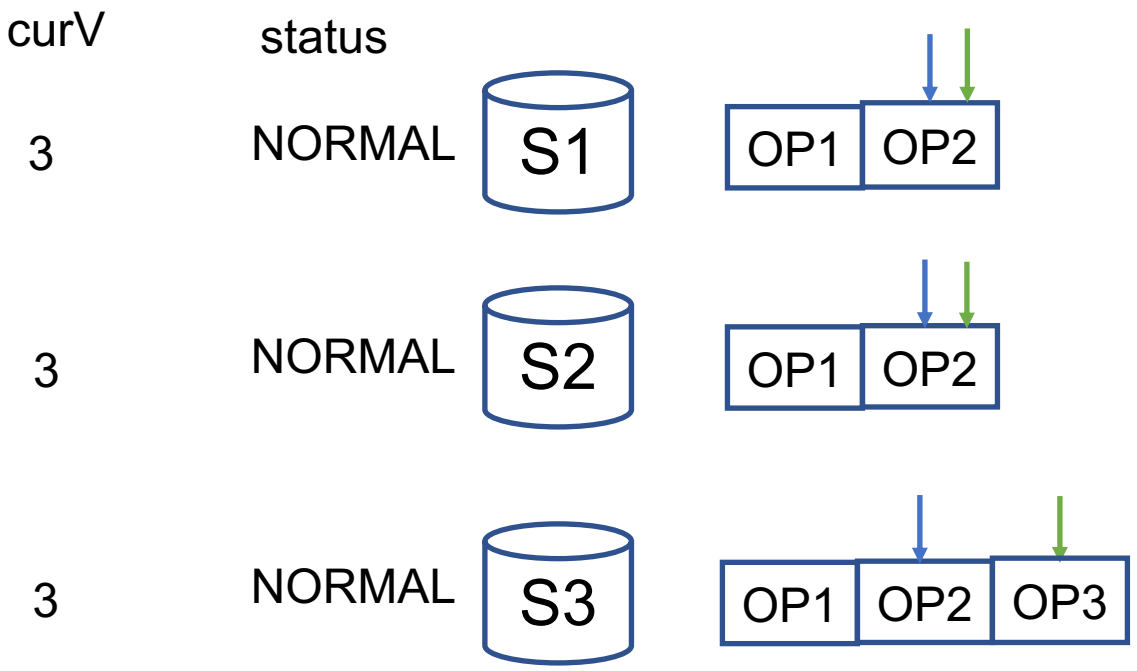


curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

On receiving Prepare msg, server must:

- ignore if its curV != msg.curV or status != NORMAL
- wait until it has entries in its log for all earlier requests
- append req to log at position msg.op-number

Basic VR protocol – Normal Processing



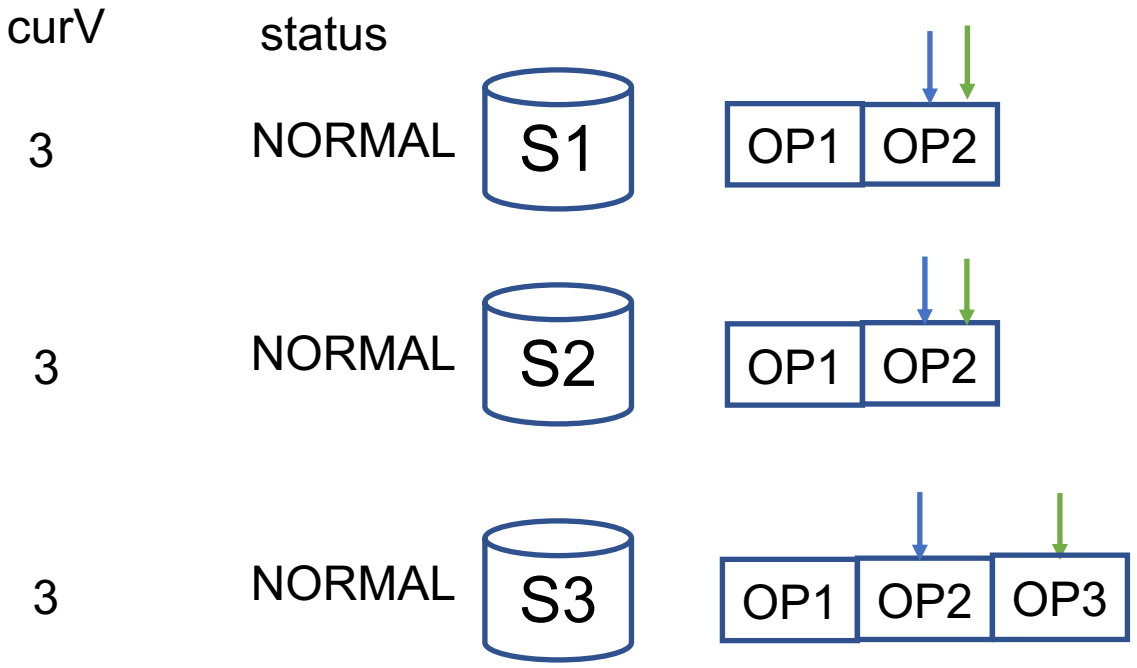
1. Primary appends the user's request to the log.

curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

On receiving Prepare msg, server must:

- ignore if its curV != msg.curV or status != NORMAL
- wait until it has entries in its log for all earlier requests
- append req to log at position msg.op-number

Basic VR protocol – Normal Processing



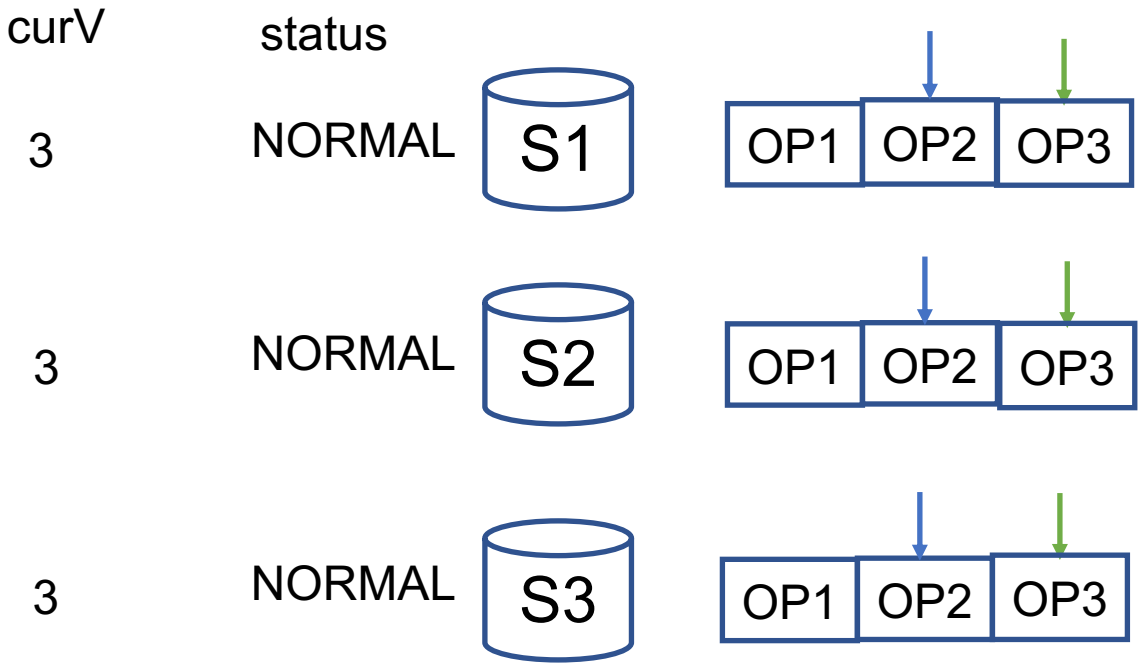
1. Primary appends the user's request to the log.
2. Primary sends prepare request to S1 and S2
3. S1 and S2 validate the prepare request

curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

On receiving Prepare msg, server must:

- ignore if its curV != msg.curV or status != NORMAL
- wait until it has entries in its log for all earlier requests
- append req to log at position msg.op-number

Basic VR protocol – Normal Processing



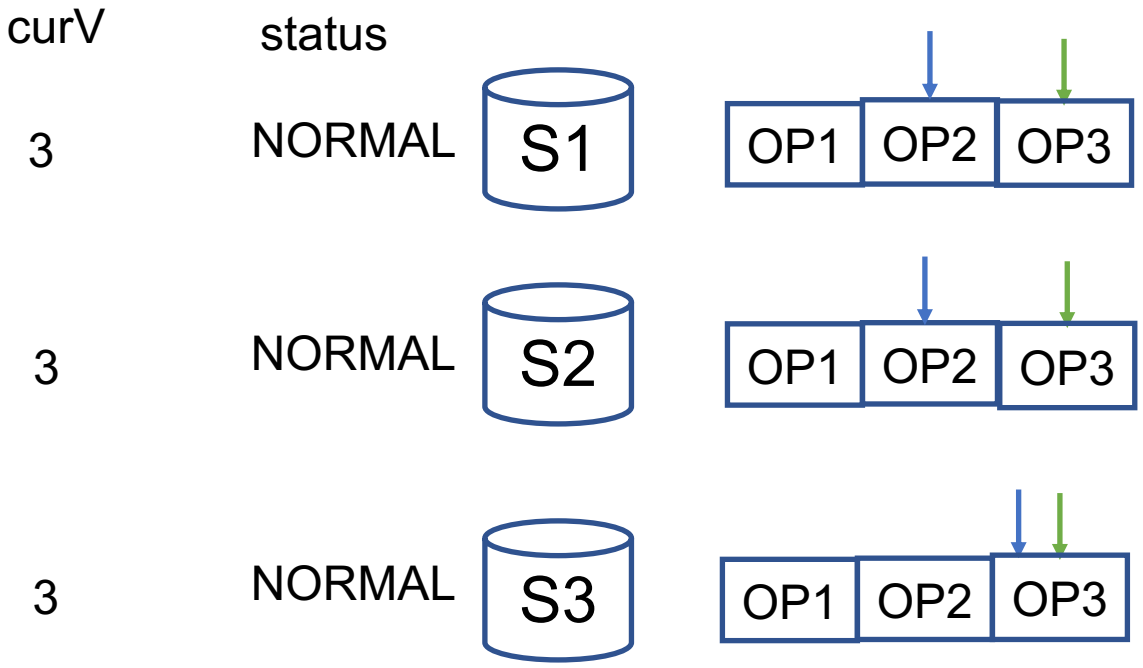
1. Primary appends the user's request to the log.
2. Primary sends prepare request to S1 and S2
3. S1 and S2 validate the prepare request
4. S1 and S2 append the request to the log and reply OK to the primary

curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

On receiving Prepare msg, server must:

- ignore if its curV != msg.curV or status != NORMAL
- wait until it has entries in its log for all earlier requests
- append req to log at position msg.op-number

Basic VR protocol – Normal Processing



1. Primary appends the user's request to the log.
2. Primary sends prepare request to S1 and S2
3. S1 and S2 validate the prepare request
4. S1 and S2 append the request to the log and reply OK to the primary
5. S3 commit the request after receiving one reply.

curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

On receiving Prepare msg, server must:

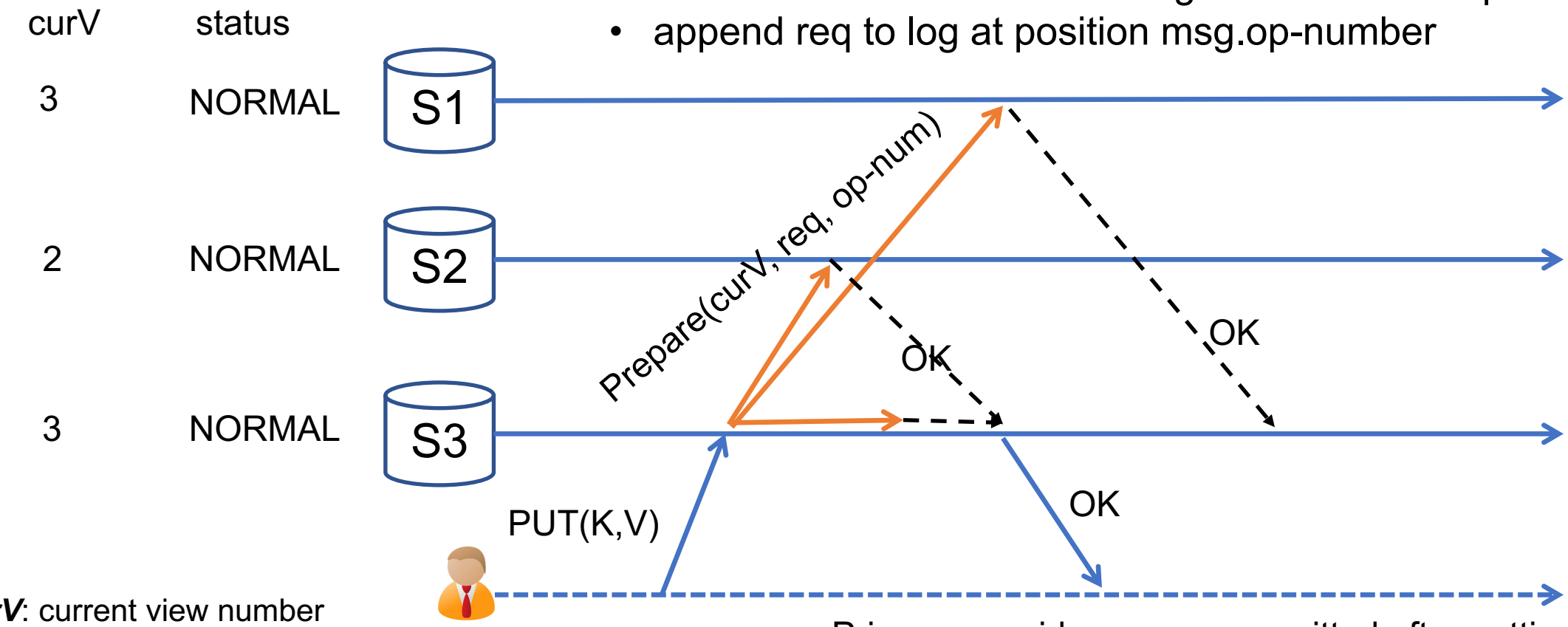
- ignore if its curV != msg.curV or status != NORMAL
- wait until it has entries in its log for all earlier requests
- append req to log at position msg.op-number

Basic VR protocol – Recovery

If curV < msg.curV run recovery

On receiving Prepare msg, server must:

- ignore if its curV != msg.curV or status != NORMAL
- wait until it has entries in its log for all earlier requests
- append req to log at position msg.op-number



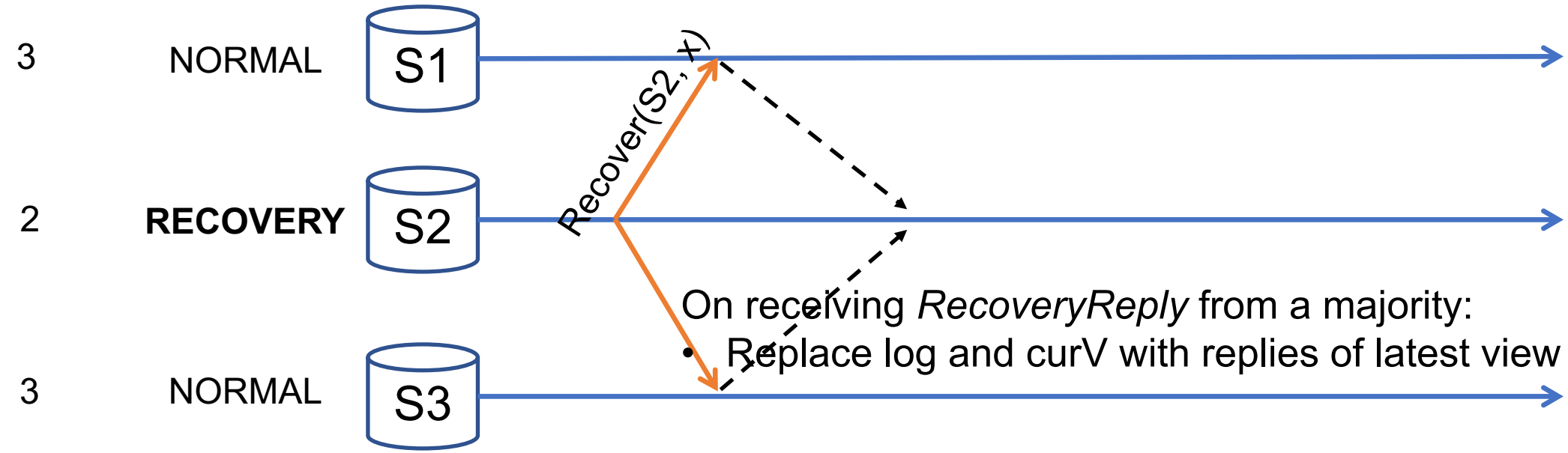
curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

Primary considers a req committed after getting majority OKs.

Basic VR protocol – Recovery

On receiving *Recovery* msg, server must:

- Ignore if status != NORMAL
- Reply <curV, x, log, op-number, commit-number>



On receiving *RecoveryReply* from a majority:

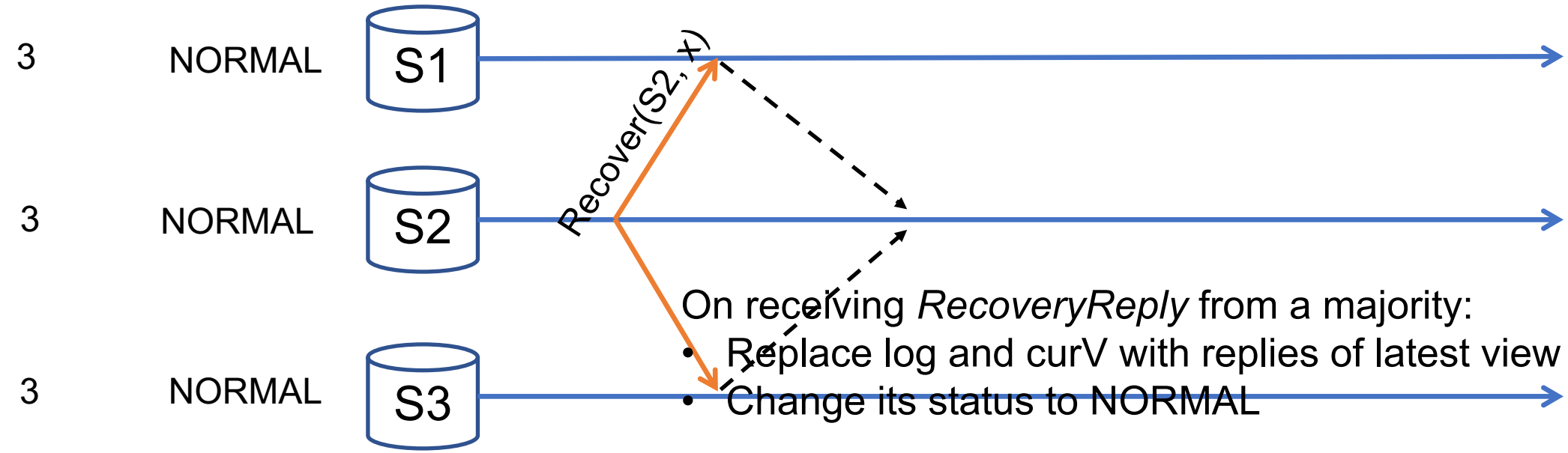
- Replace log and curV with replies of latest view (biggest curV)

curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

Basic VR protocol – Recovery

On receiving *Recovery* msg, server must:

- Ignore if status != NORMAL
- Reply <curV, x, log, op-number, commit-number>

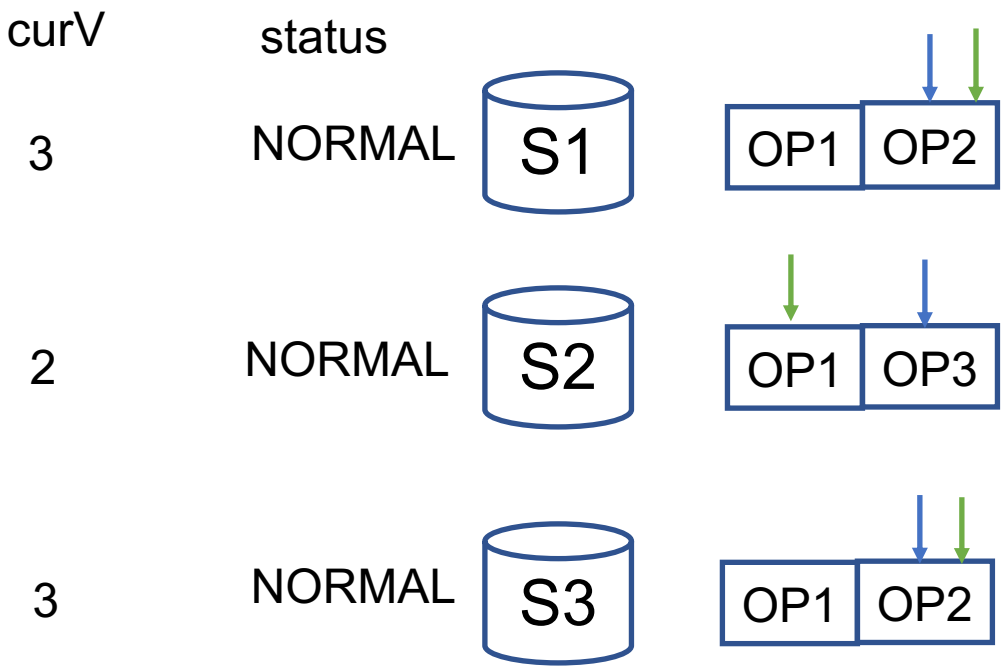


On receiving *RecoveryReply* from a majority:

- Replace log and curV with replies of latest view (biggest curV)
- Change its status to NORMAL

curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

Basic VR protocol – Recovery



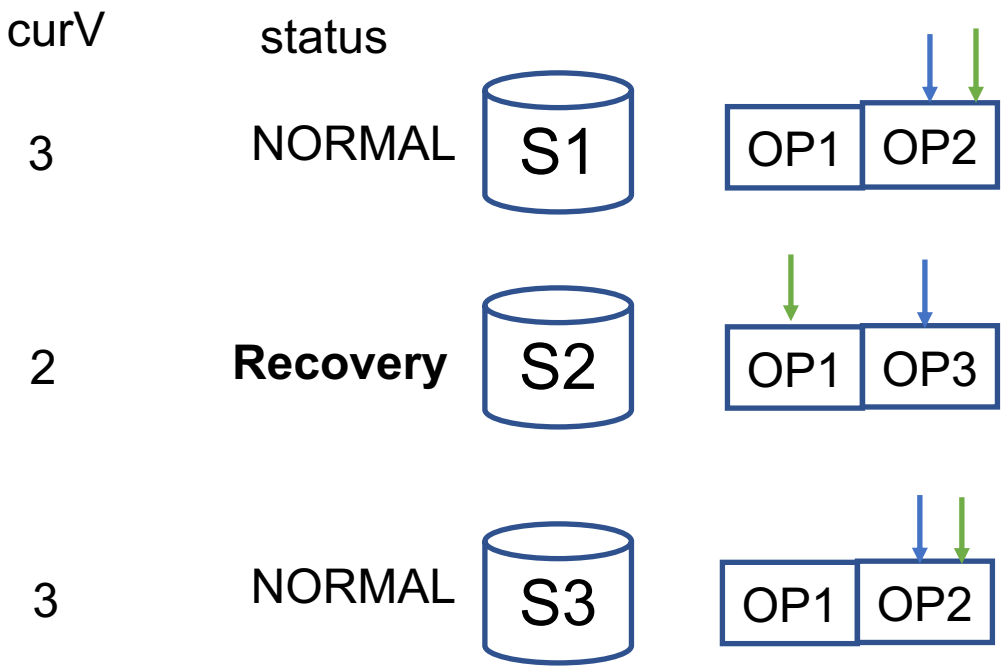
1. S2's network is resumed
2. S3 sends *Prepare* request to S2 with V3

curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

On receiving *Recovery* msg, server must:
- Ignore if status != NORMAL
- Reply <curV, x, log, op-number, commit-number>

On receiving *RecoveryReply* from a majority:
- Replace log and curV with replies of latest view
- Change its status to NORMAL

Basic VR protocol – Recovery



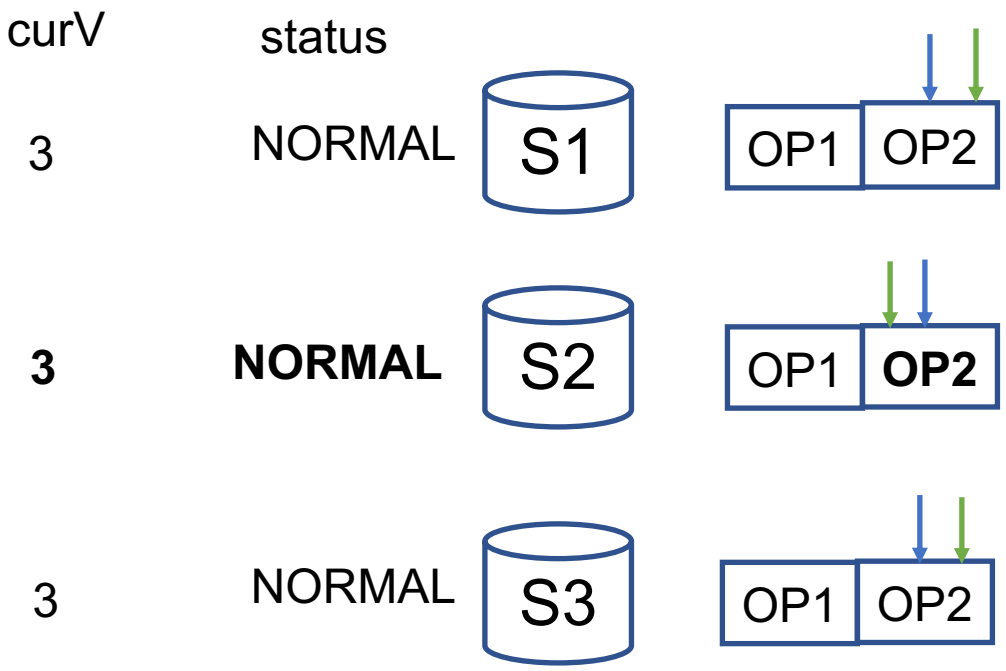
1. S2's network is resumed
2. S3 sends *Prepare* request to S2 with V3
3. S2 switches into Recovery state
4. S2 sends *Recovery* request to S1 and S3
5. S1 and S3 reply with its information

curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

On receiving *Recovery* msg, server must:
- Ignore if status != NORMAL
- Reply <curV, x, log, op-number, commit-number>

On receiving *RecoveryReply* from a majority:
- Replace log and curV with replies of latest view
- Change its status to NORMAL

Basic VR protocol – Recovery



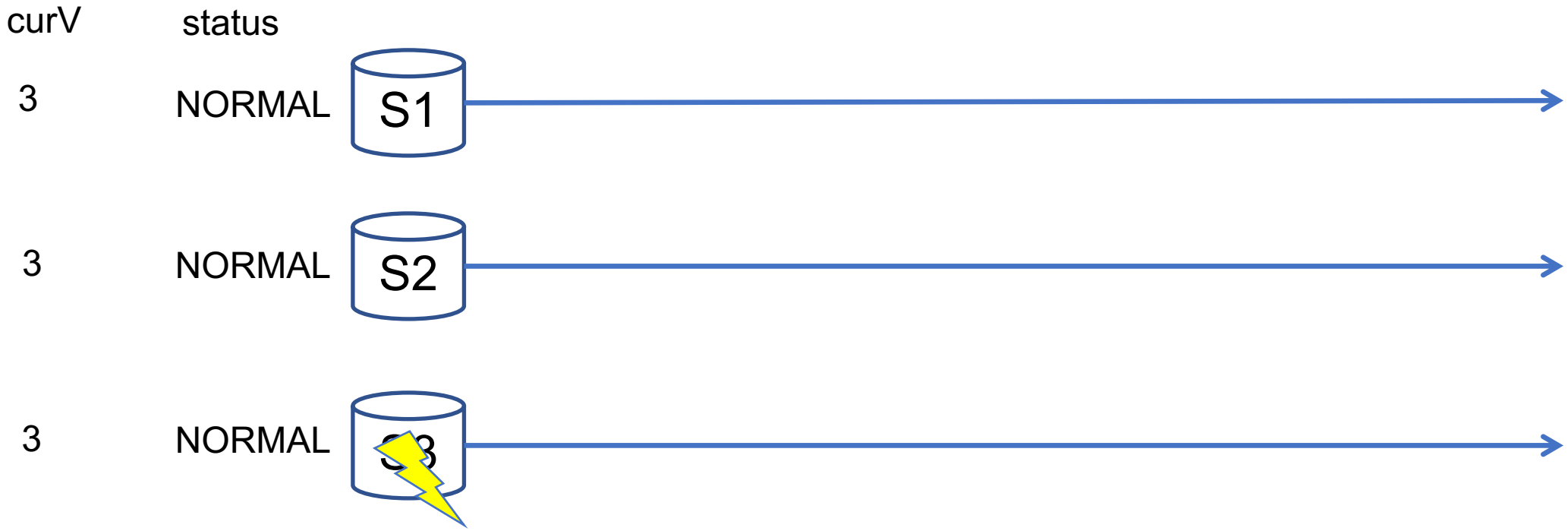
curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

1. S2's network is resumed
2. S3 sends *Prepare* request to S2 with V3
3. S2 switches into Recovery state
4. S2 sends *Recovery* request to S1 and S3
5. S1 and S3 reply with its information
6. S2 updates its log, curV and status

On receiving *Recovery* msg, server must:
- Ignore if status != NORMAL
- Reply <curV, x, log, op-number, commit-number>

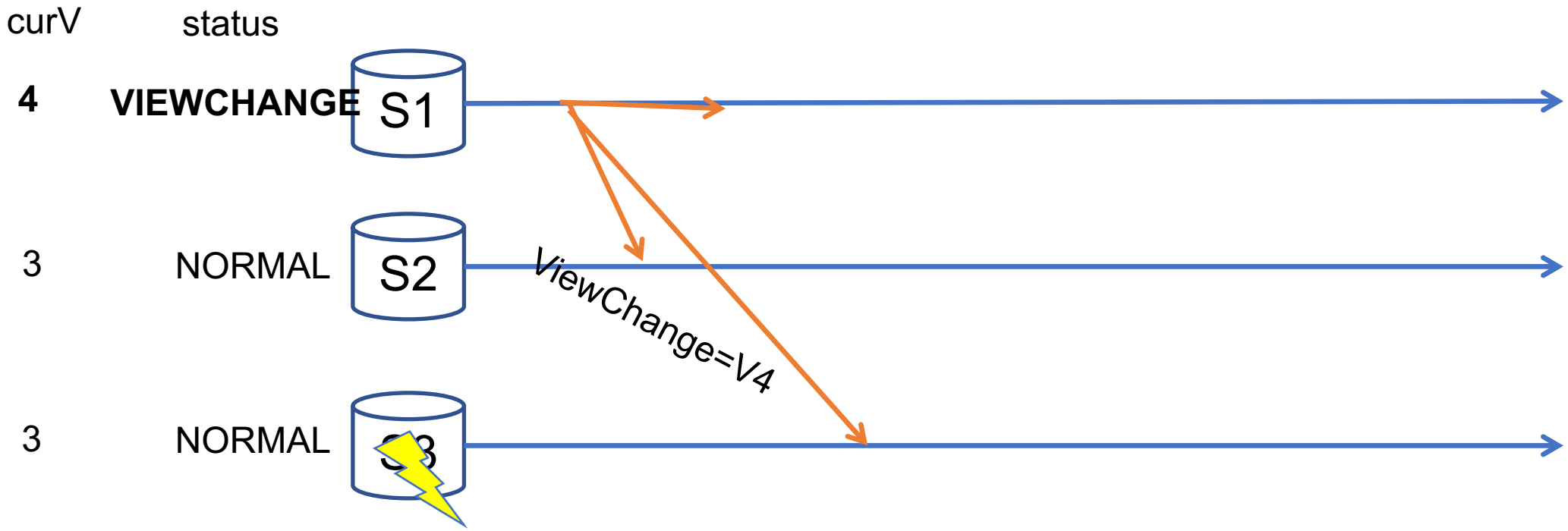
On receiving *RecoveryReply* from a majority:
- Replace log and curV with replies of latest view
- Change its status to NORMAL

Basic VR protocol – View Change



curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

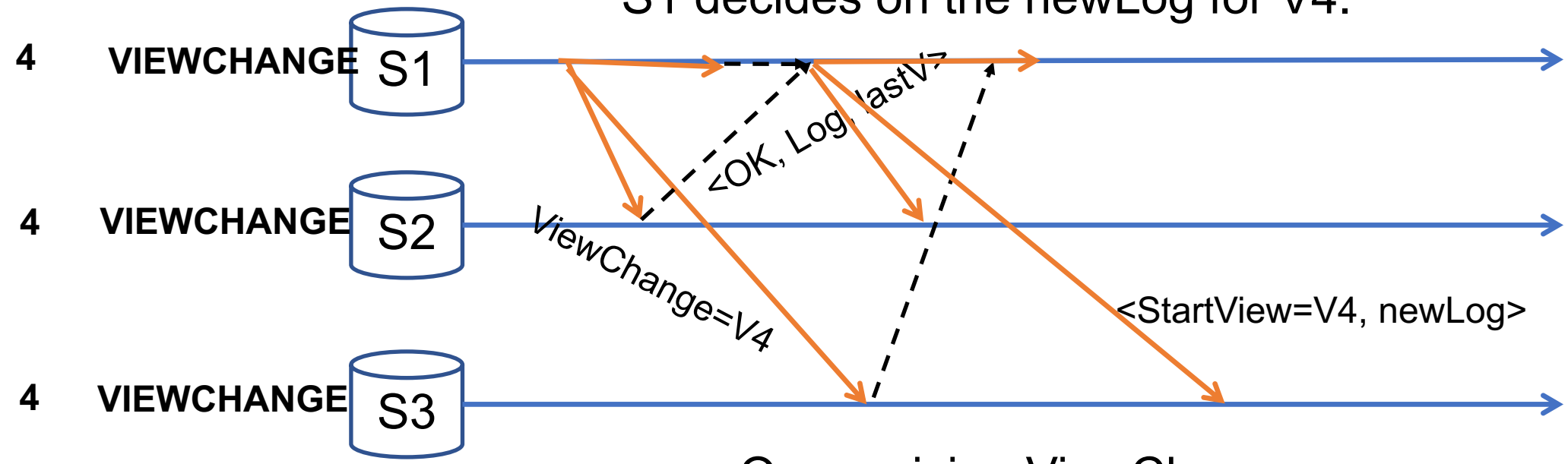
Basic VR protocol – View Change



curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

Basic VR protocol – View Change

View-change to v1 succeeds
If a majority replied OK.
S1 decides on the newLog for V4.



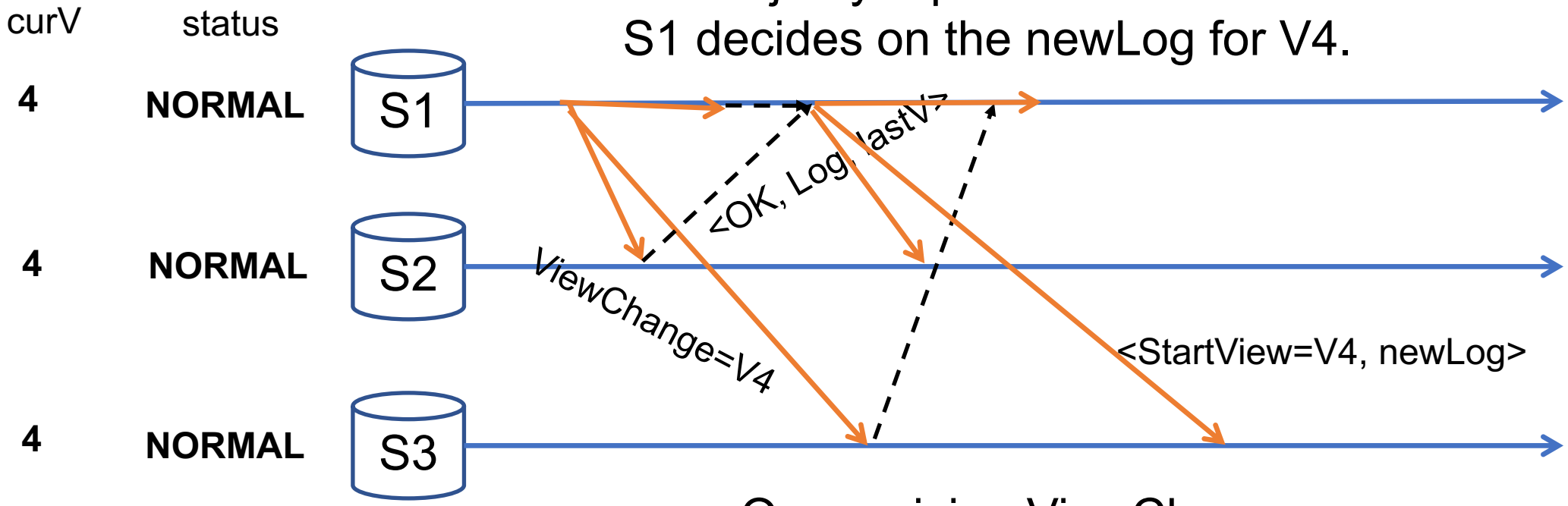
On receiving ViewChange msg, server does:

- reject if its curV > msg.V
- set curV = msg.V, status= VIEWCHANGE
- return its log and the curV.

curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

Basic VR protocol – View Change

View-change to v1 succeeds
If a majority replied OK.
S1 decides on the newLog for V4.

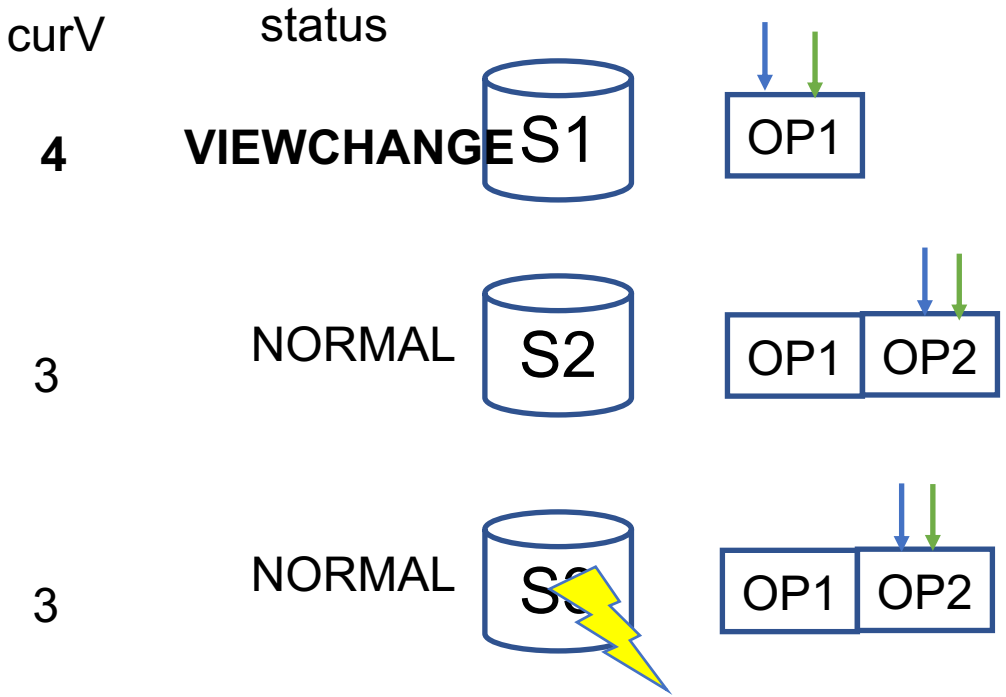


On receiving ViewChange msg, server does:

- reject if its curV > msg.V
- set curV = msg.V, status= VIEWCHANGE
- return its log and the curV.

curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

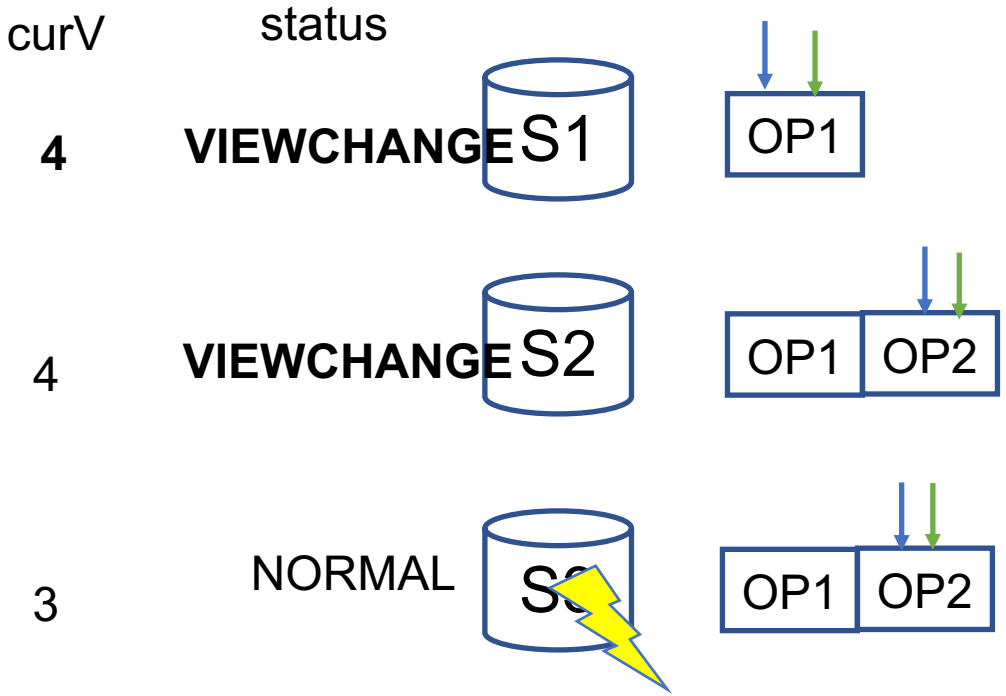
Basic VR protocol – View Change



1. S1 suspects primary crashes
2. S1 sends ViewChange request to S2

curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

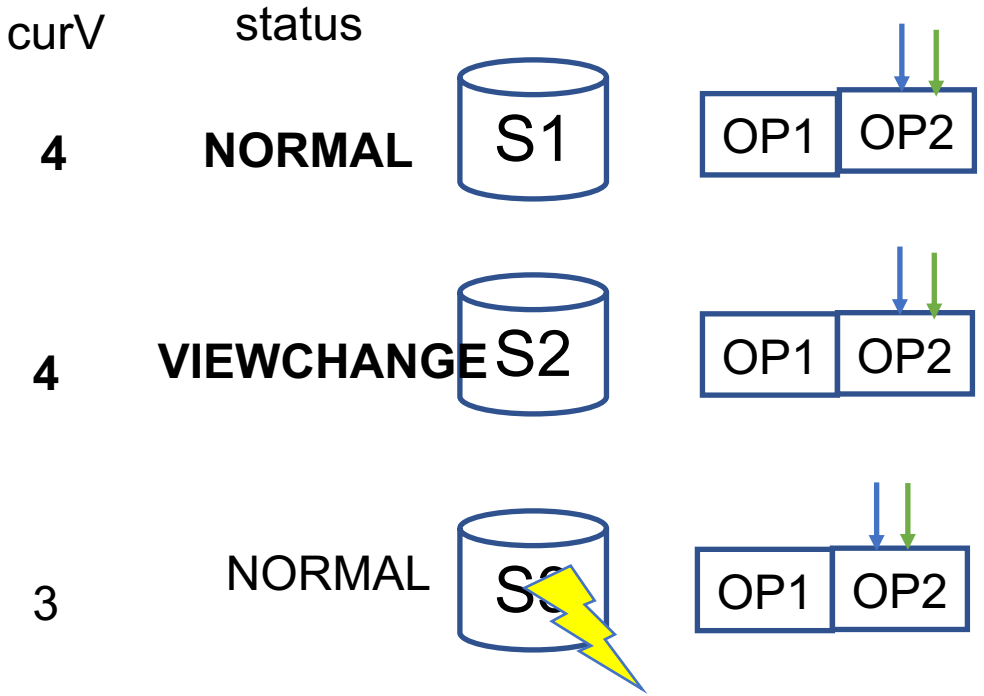
Basic VR protocol – View Change



1. S1 suspects primary crashes
2. S1 sends ViewChange request to S2
3. S2 replies S1 with its log and curV
4. S1 generate new log and start new view

curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

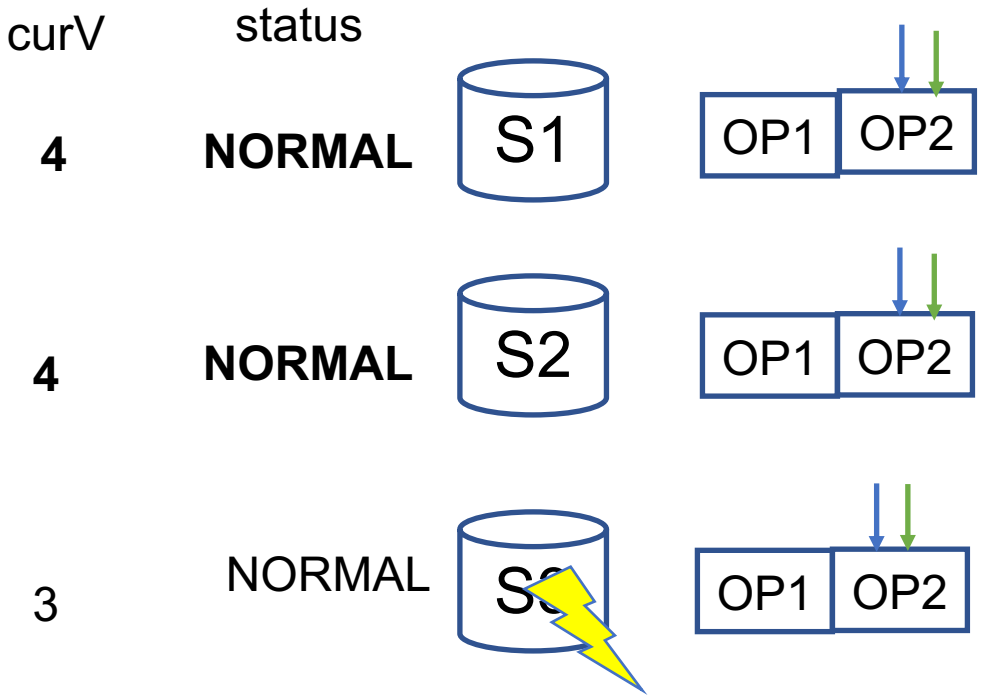
Basic VR protocol – View Change



1. S1 suspects primary crashes
2. S1 sends ViewChange request to S2
3. S2 replies S1 with its log and curV
4. S1 generate new log and start new view

curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

Basic VR protocol – View Change



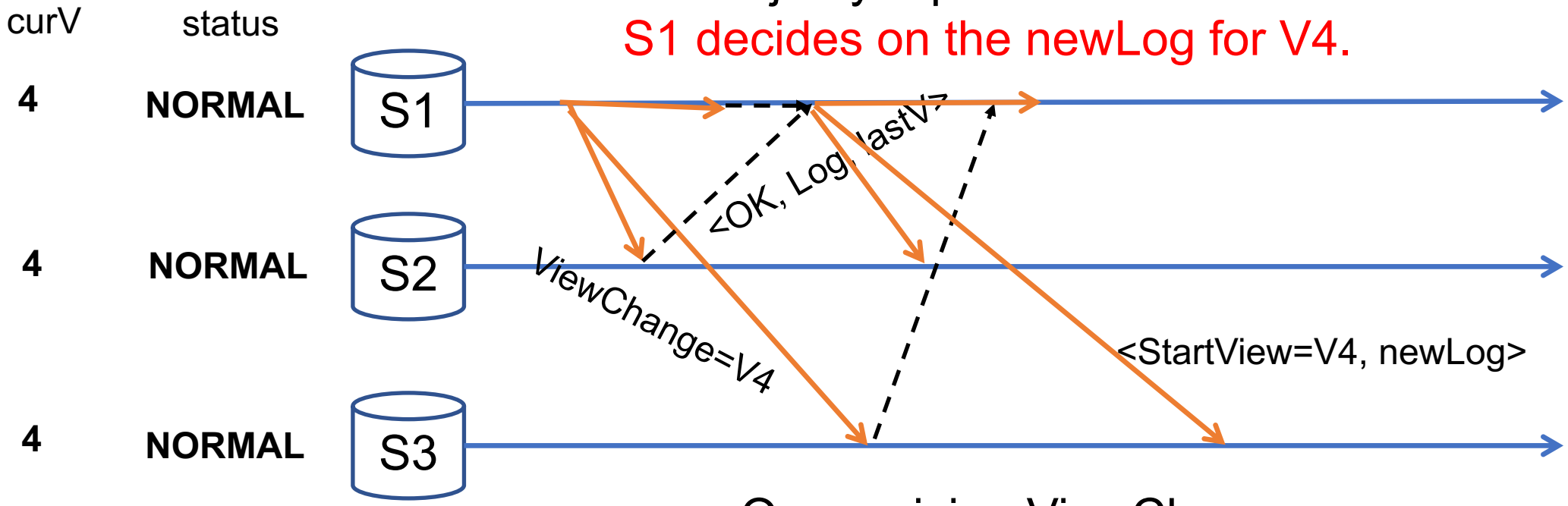
1. S1 suspects primary crashes
2. S1 sends ViewChange request to S2
3. S2 replies S1 with its log and curV
4. S1 generate new log and start new view
5. S1 sends StartView to others

curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

Basic VR protocol – View Change

But how???

View-change to v1 succeeds
If a majority replied OK.
S1 decides on the newLog for V4.



On receiving ViewChange msg, server does:

- reject if its curV > msg.V
- set curV = msg.V, status= VIEWCHANGE
- return its log and the curV.

curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

Basic VR protocol – View Change

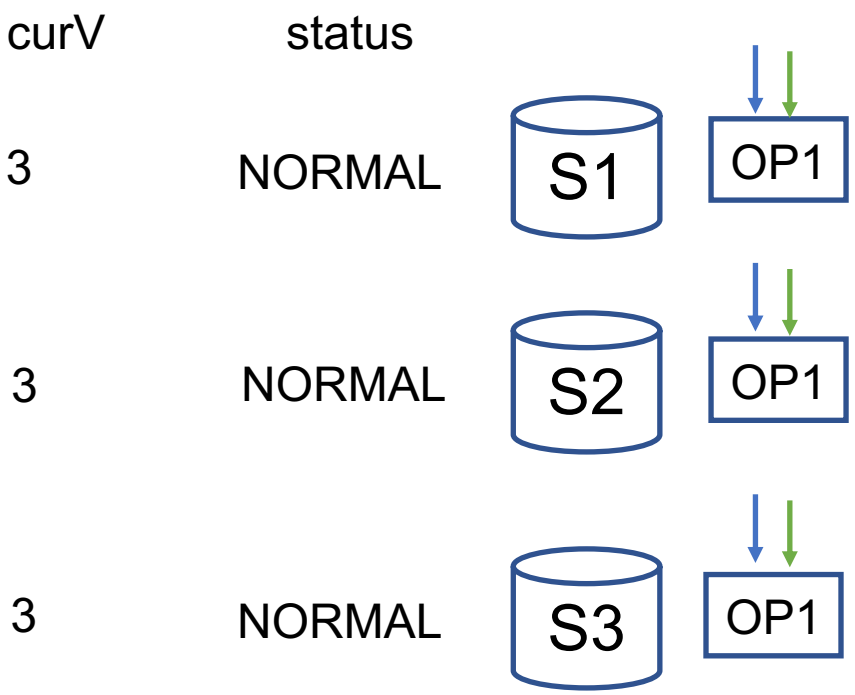
Question: how to decide on the new log???

Hypothesis: use the longest one???

Basic VR protocol – View Change

Question: how to decide on the new log???

Hypothesis: use the longest one???

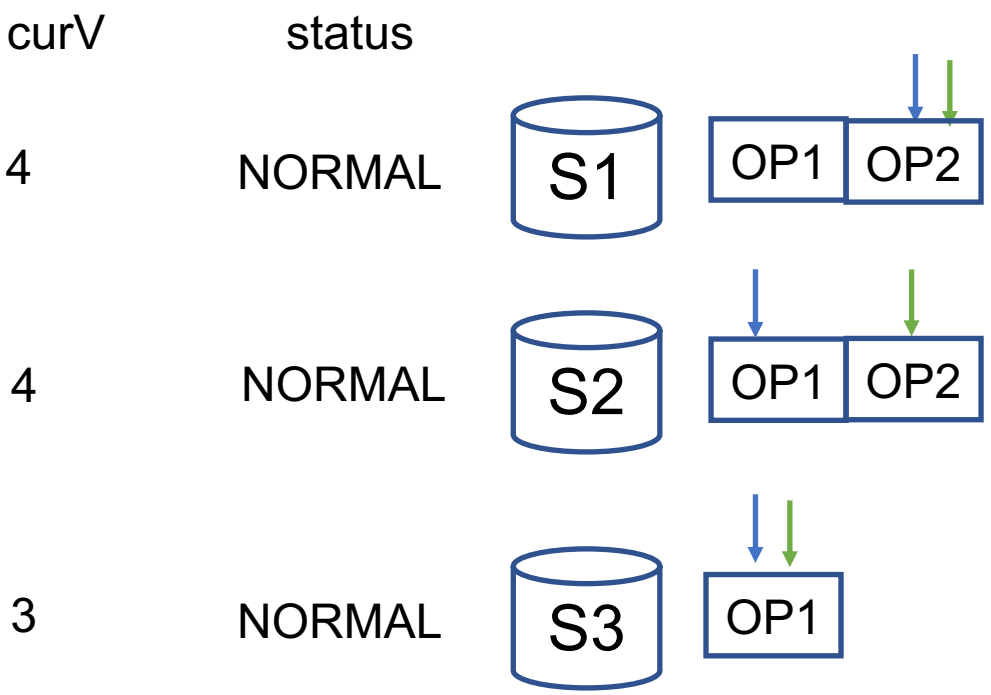


1. Network partitions S1, S2 from S3
2. V4 becomes active, S1, S2 replicate OP2
3. Primary S3 replicates OP3,OP4 at itself
4. Network heals, S1 crashes

Basic VR protocol – View Change

Question: how to decide on the new log???

Hypothesis: use the longest one???

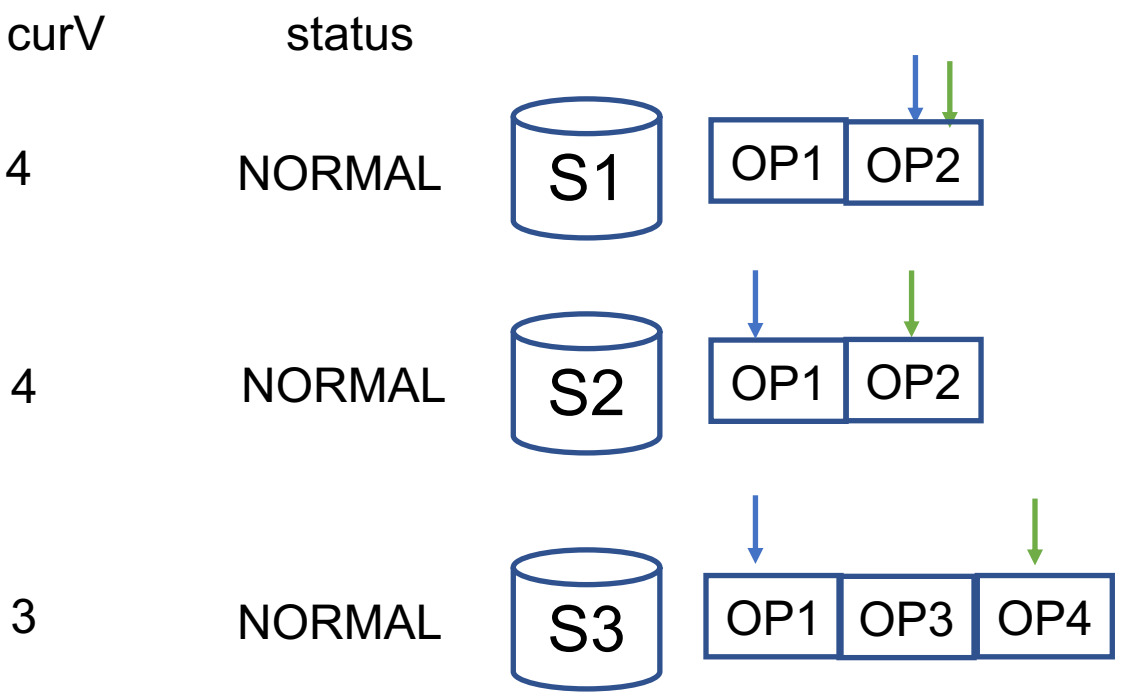


1. Network partitions S1, S2 from S3
2. V4 becomes active, S1, S2 replicate OP2

Basic VR protocol – View Change

Question: how to decide on the new log???

Hypothesis: use the longest one???

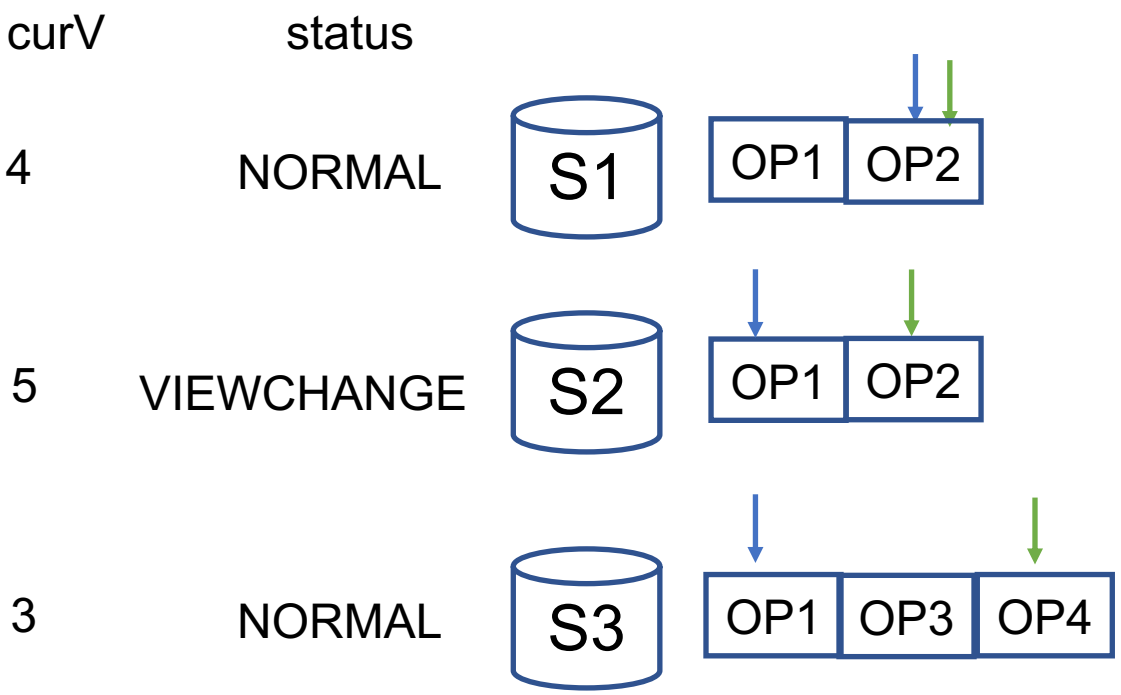


1. Network partitions S1, S2 from S3
2. V4 becomes active, S1, S2 replicate OP2
3. Primary S3 replicates OP3,OP4 at itself

Basic VR protocol – View Change

Question: how to decide on the new log???

Hypothesis: use the longest one???

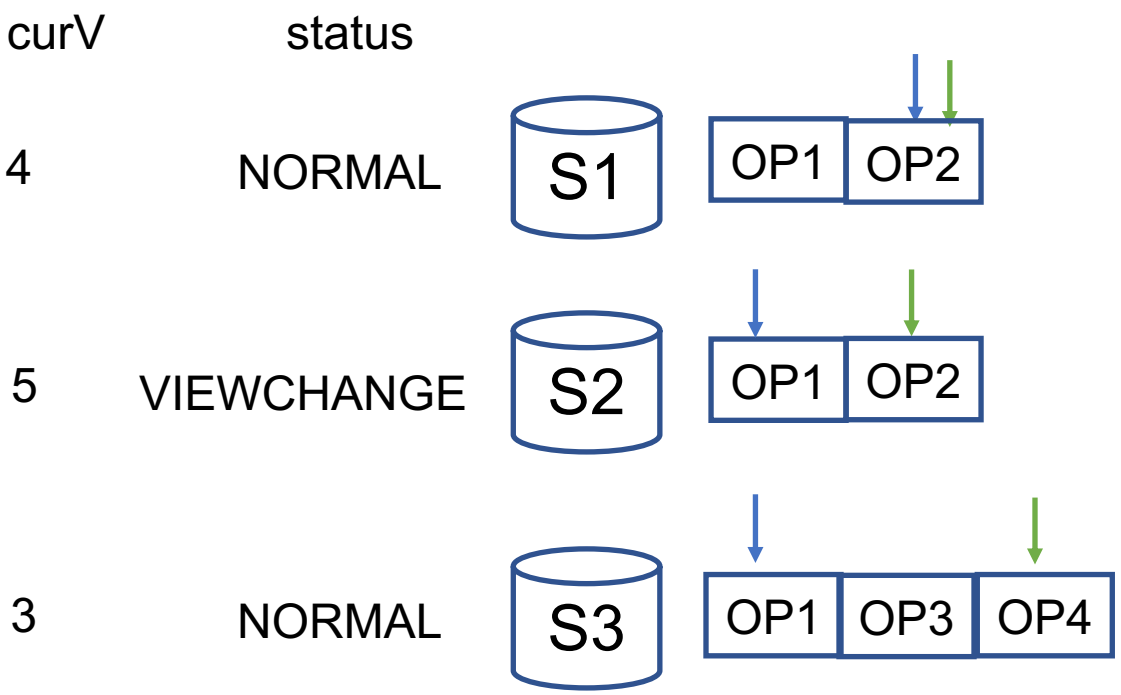


1. Network partitions S1, S2 from S3
2. V4 becomes active, S1, S2 replicate OP2
3. Primary S3 replicates OP3,OP4 at itself
4. Network heals, S1 crashes

Basic VR protocol – View Change

Question: how to decide on the new log???

Hypothesis: use one with biggest curV???

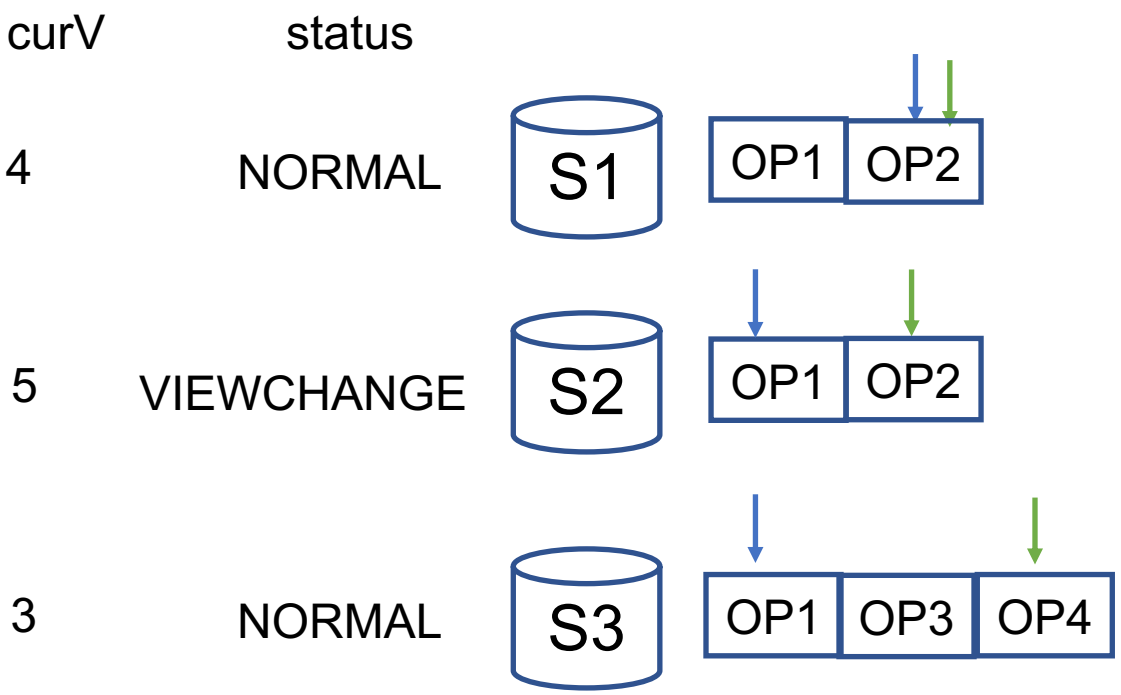


1. Network partitions S1, S2 from S3
2. V4 becomes active, S1, S2 replicate OP2
3. Primary S3 replicates OP3,OP4 at itself
4. Network heals, S1 crashes

Basic VR protocol – View Change

Question: how to decide on the new log???

Hypothesis: use one with biggest curV???

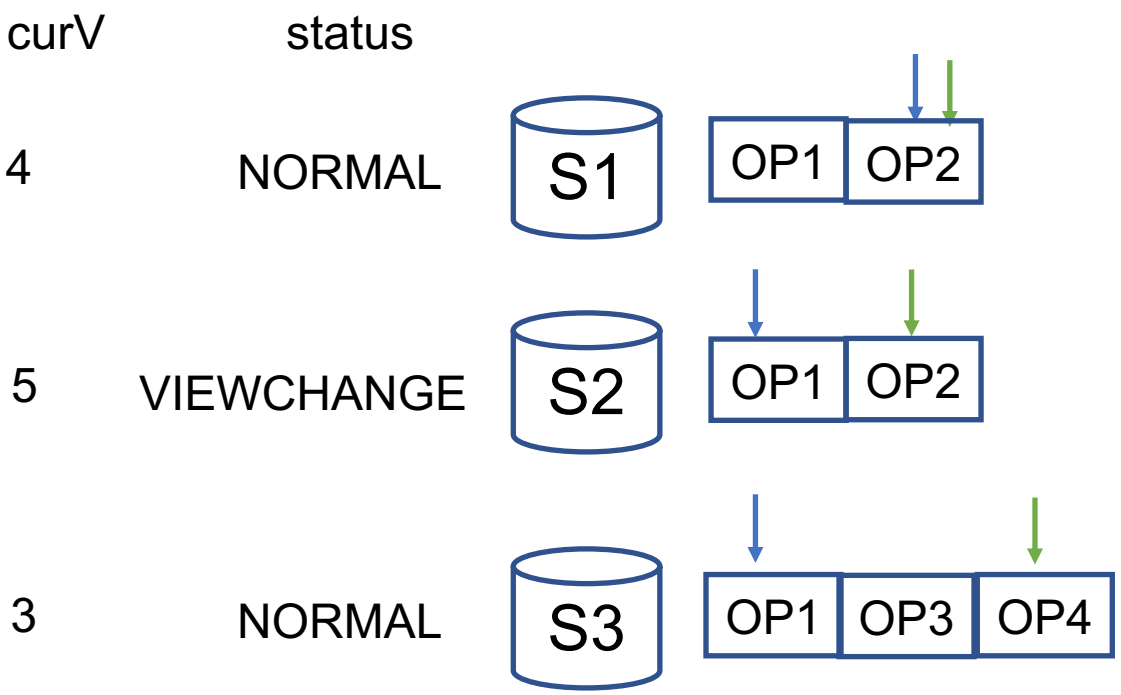


1. Network partitions S1, S2 from S3
2. V4 becomes active, S1, S2 replicate OP2
3. Primary S3 replicates OP3,OP4 at itself
4. Network heals, S1 crashes
5. S3 receives VIEWCHANGE request
6. S1 heals, S2 crashes.

Basic VR protocol – View Change

Question: how to decide on the new log???

Hypothesis: use one with biggest curV???

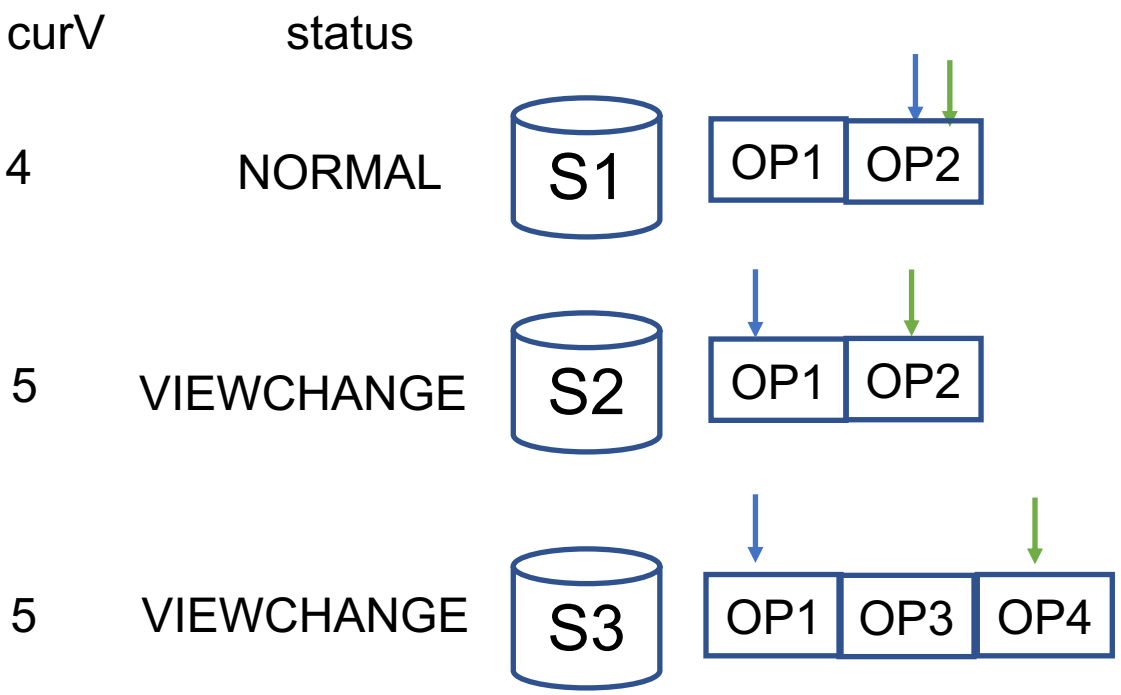


1. Network partitions S1, S2 from S3
2. V4 becomes active, S1, S2 replicate OP2
3. Primary S3 replicates OP3,OP4 at itself
4. Network heals, S1 crashes

Basic VR protocol – View Change

Question: how to decide on the new log???

Hypothesis: use one with biggest curV???

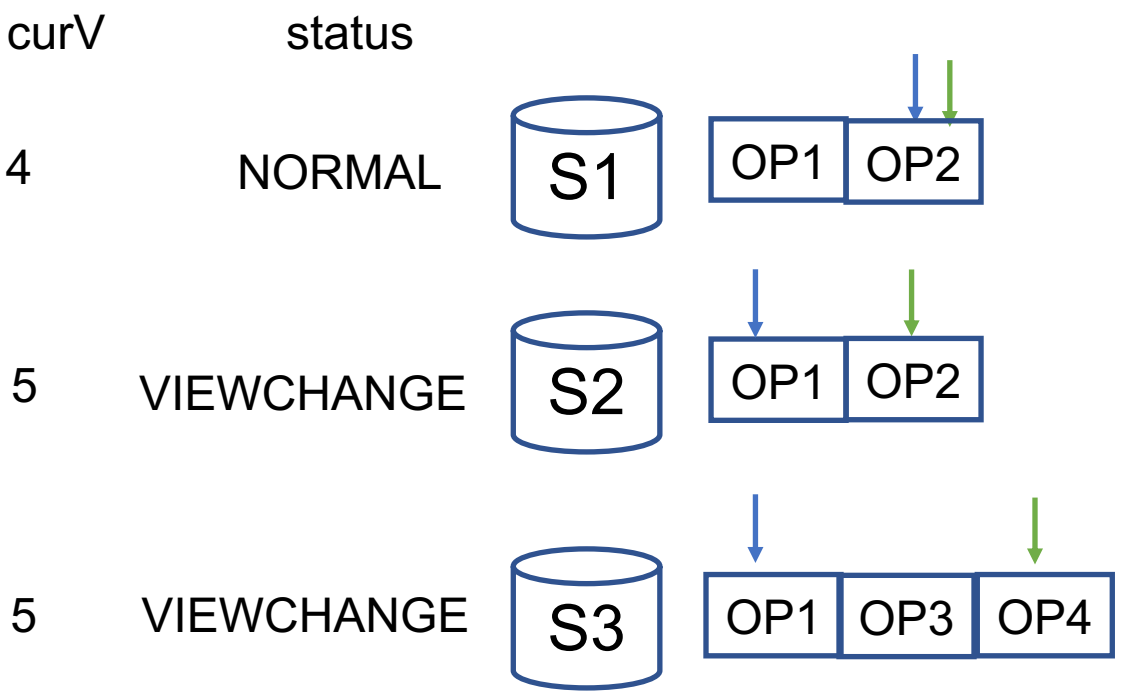


1. Network partitions S1, S2 from S3
2. V4 becomes active, S1, S2 replicate OP2
3. Primary S3 replicates OP3,OP4 at itself
4. Network heals, S1 crashes
5. S3 receives VIEWCHANGE request

Basic VR protocol – View Change

Question: how to decide on the new log???

Hypothesis: use one with biggest curV???



1. Network partitions S1, S2 from S3
2. V4 becomes active, S1, S2 replicate OP2
3. Primary S3 replicates OP3,OP4 at itself
4. Network heals, S1 crashes
5. S3 receives VIEWCHANGE request
6. S1 heals, S2 crashes.

Basic VR protocol – View Change

Rule 1: Pick the log w/ biggest lastV

curV	lastV	status		
3	3	NORMAL	S1	OP1
3	3	NORMAL	S2	OP1
3	3	NORMAL	S3	OP1

1. Network partitions S1, S2 from S3
2. V4 becomes active, S1, S2 replicate OP2
3. Primary S3 replicates OP3,OP4 at itself
4. Network heals, S1 crashes
5. S3 receives VIEWCHANGE request
6. S1 heals, S2 crashes.

Add a new state for each server: lastV – the last view number in which its normal status

Basic VR protocol – View Change

Rule 1: Pick the log w/ biggest lastV

curV	lastV	status		
4	3	VIEWCHANGE	S1	OP1
4	3	VIEWCHANGE	S2	OP1
3	3	NORMAL	S3	OP1

1. Network partitions S1, S2 from S3

Add a new state for each server: lastV – the last view number in which its normal status

Basic VR protocol – View Change

Rule 1: Pick the log w/ biggest lastV

curV	lastV	status		
4	4	NORMAL	S1	OP1 OP2
4	4	NORMAL	S2	OP1 OP2
3	3	NORMAL	S3	OP1

1. Network partitions S1, S2 from S3

2. V4 becomes active, S1, S2 replicate OP2

Add a new state for each server: lastV – the last view number in which its normal status

Basic VR protocol – View Change

Rule 1: Pick the log w/ biggest lastV

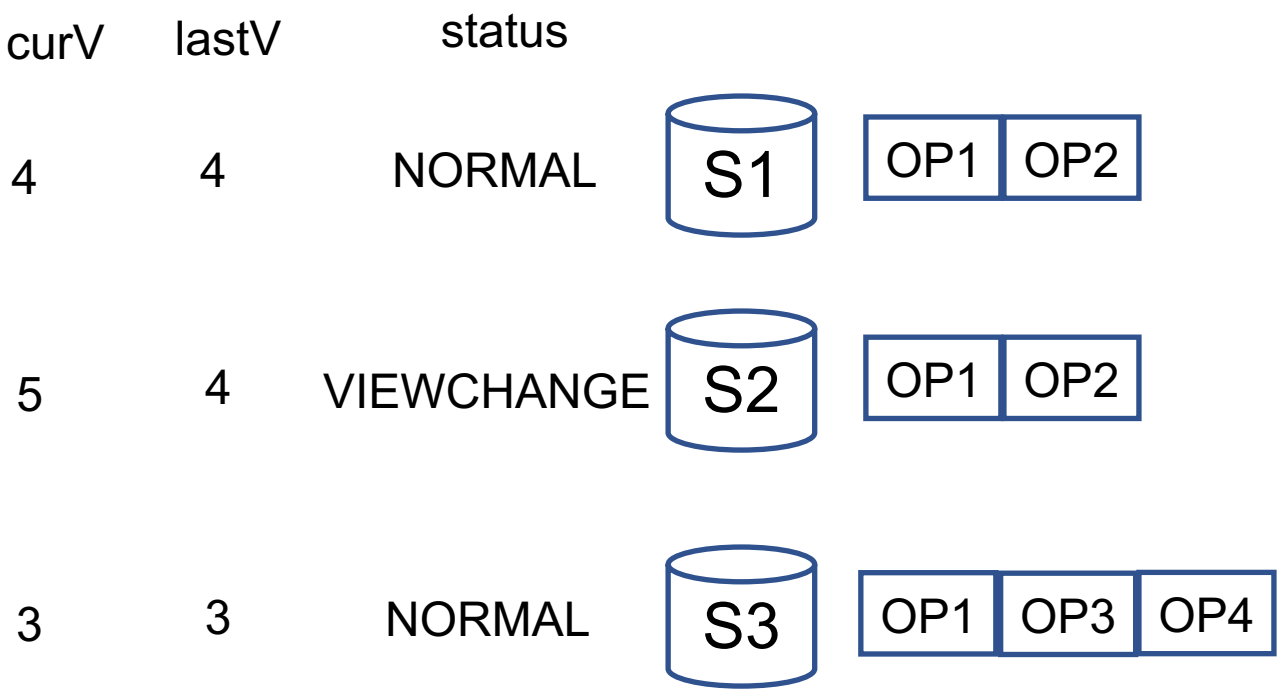
curV	lastV	status		
4	4	NORMAL	S1	OP1 OP2
4	4	NORMAL	S2	OP1 OP2
3	3	NORMAL	S3	OP1 OP3 OP4

1. Network partitions S1, S2 from S3
2. V4 becomes active, S1, S2 replicate OP2
3. Primary S3 replicates OP3,OP4 at itself

Add a new state for each server: lastV – the last view number in which its normal status

Basic VR protocol – View Change

Rule 1: Pick the log w/ biggest lastV



1. Network partitions S1, S2 from S3
2. V4 becomes active, S1, S2 replicate OP2
3. Primary S3 replicates OP3,OP4 at itself
4. Network heals, S1 crashes

Add a new state for each server: lastV – the last view number in which its normal status

Basic VR protocol – View Change

Rule 1: Pick the log w/ biggest lastV

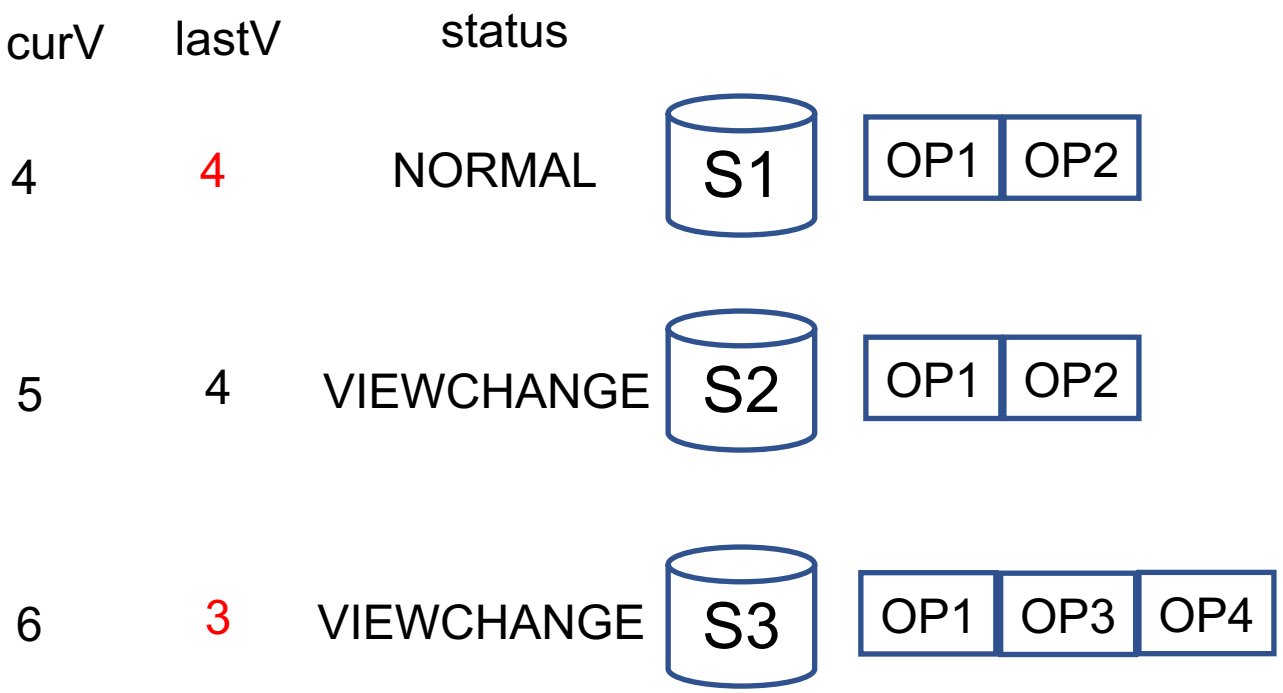
curV	lastV	status		
4	4	NORMAL	S1	OP1 OP2
5	4	VIEWCHANGE	S2	OP1 OP2
5	3	VIEWCHANGE	S3	OP1 OP3 OP4

1. Network partitions S1, S2 from S3
2. V4 becomes active, S1, S2 replicate OP2
3. Primary S3 replicates OP3,OP4 at itself
4. Network heals, S1 crashes
5. S3 receives VIEWCHANGE request

Add a new state for each server: lastV – the last view number in which its normal status

Basic VR protocol – View Change

Rule 1: Pick the log w/ biggest lastV



1. Network partitions S1, S2 from S3
2. V4 becomes active, S1, S2 replicate OP2
3. Primary S3 replicates OP3,OP4 at itself
4. Network heals, S1 crashes
5. S3 receives VIEWCHANGE request
6. S1 heals, S2 crashes.

Add a new state for each server: lastV – the last view number in which its normal status

Basic VR protocol – View Change

Rule 2: if >1 logs exist in rule-1, pick the longest one

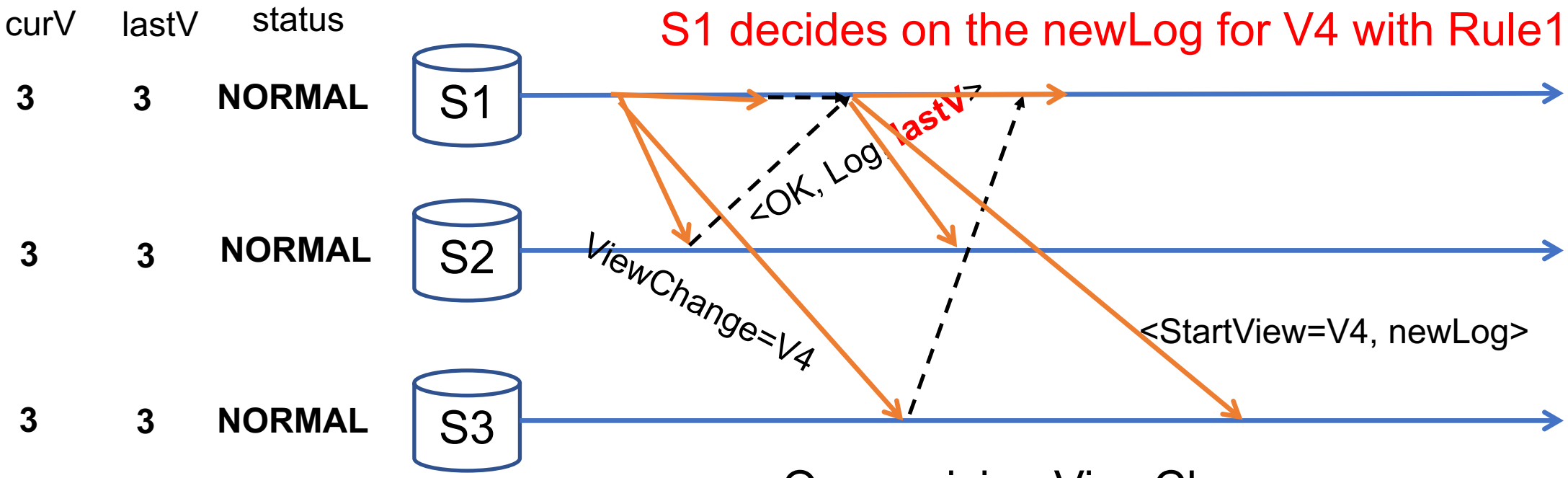
curV	lastV	status		
4	4	NORMAL	S1	OP1 OP2
5	4	VIEWCHANGE	S2	OP1 OP2
6	3	VIEWCHANGE	S3	OP1 OP3 OP4

1. Network partitions S1, S2 from S3
2. V4 becomes active, S1, S2 replicate OP2
3. Primary S3 replicates OP3,OP4 at itself
4. Network heals, S1 crashes
5. S3 receives VIEWCHANGE request
6. S1 heals, S2 crashes.

Add a new state for each server: lastV – the last view number in which its normal status

Basic VR protocol – View Change

View-change to v1 succeeds
If a majority replied OK.
S1 decides on the newLog for V4 with Rule1 and Rule2



On receiving ViewChange msg, server does:

- reject if its curV > msg.V
- set curV = msg.V, status= VIEWCHANGE
- return its log and the **lastV**.

curV: current view number
status: NORMAL, VIEW-CHANGE, or RECOVERY
op-num: the number of last op in the log
commit-num: the number of last committed op

Normal Process

Primary sends the req with curV to all

On receiving Prepare msg, server must:

- ignore if its curV != msg.curV or status != NORMAL
- wait until it has entries in its log for all earlier requests
- append req to log at position msg.op-number

Primary considers a req committed after getting majority OKs.

Recovery

A healed server sends Recovery request to all

On receiving *Recovery* msg, server must:

- Ignore if status != NORMAL
- Reply <curV, x, log, op-number, commit-number>

On receiving *RecoveryReply* from a majority:

- Replace log and curV with replies of latest view
- Change its status to NORMAL

ViewChange

A candidate sends a ViewChange request to all

On receiving ViewChange msg, server does:

- reject if its curV > msg.V
- set curV = msg.V, status= VIEWCHANGE
- return its log and the lastV.

On receiving OK from majority:
S1 decides on the newLog for V4 with following rules:

1. Rule 1: Pick the log w/ biggest lastV
2. Rule 2: if >1 logs exist in rule-1, pick the longest one

New primary sends StartView message to all

Backups replace log with primary's log,
update status and lastV

Normal Process

Primary sends the req with curV to all

On receiving Prepare msg, server must:

- ignore if its curV != msg.curV or status != NORMAL
- wait until it has entries in its log for all earlier requests
- append req to log at position msg.op-number

Primary considers a req committed after getting majority OKs.

Recovery

A healed server sends Recovery request to all

On receiving *Recovery* msg, server must:

- Ignore if status != NORMAL
- Reply <curV, x, log, op-number, commit-number>

On receiving *RecoveryReply* from a majority:

- Replace log and curV with replies of latest view
- Change its status to NORMAL

ViewChange

A candidate sends a ViewChange request to all

On receiving ViewChange msg, server does:

- reject if its curV > msg.V
- set curV = msg.V, status= VIEWCHANGE
- return its log and the lastV.

On receiving OK from majority:
S1 decides on the newLog for V4 with following rules:

1. Rule 1: Pick the log w/ biggest lastV
2. Rule 2: if >1 logs exist in rule-1, pick the longest one

New primary sends StartView message to all

Backups replace log with primary's log, update status and lastV

At the beginning of a view, backup's logs are identical with primary's log.

Normal Process

Primary sends the req with curV to all

On receiving Prepare msg, server must:

- ignore if its curV != msg.curV or status != NORMAL
- wait until it has entries in its log for all earlier requests
- append req to log at position msg.op-number

Primary considers a req committed after getting majority OKs.

Recovery

A healed server sends Recovery request to all

On receiving *Recovery* msg, server must:

- Ignore if status != NORMAL
- Reply <curV, x, log, op-number, commit-number>

On receiving *RecoveryReply* from a majority:

- Replace log and curV with replies of latest view
- Change its status to NORMAL

ViewChange

A candidate sends a ViewChange request to all

On receiving ViewChange msg, server does:

- reject if its curV > msg.V
- set curV = msg.V, status= VIEWCHANGE
- return its log and the lastV.

On receiving OK from majority:
S1 decides on the newLog for V4 with following rules:

1. Rule 1: Pick the log w/ biggest lastV
2. Rule 2: if >1 logs exist in rule-1, pick the longest one

New primary sends StartView message to all

Backups replace log with primary's log, update status and lastV

In the same view, req is always committed in the same position in a majority of servers.

Normal Process

Primary sends the req with curV to all

On receiving Prepare msg, server must:

- ignore if its curV != msg.curV or status != NORMAL
- wait until it has entries in its log for all earlier requests
- append req to log at position msg.op-number

Primary considers a req committed after getting majority OKs.

Recovery

A healed server sends Recovery request to all

On receiving *Recovery* msg, server must:

- Ignore if status != NORMAL
- Reply <curV, x, log, op-number, commit-number>

On receiving *RecoveryReply* from a majority:

- Replace log and curV with replies of latest view
- Change its status to NORMAL

ViewChange

A candidate sends a ViewChange request to all

On receiving ViewChange msg, server does:

- reject if its curV > msg.V
- set curV = msg.V, status= VIEWCHANGE
- return its log and the lastV.

On receiving OK from majority:
S1 decides on the newLog for V4 with following rules:

1. Rule 1: Pick the log w/ biggest lastV
2. Rule 2: if >1 logs exist in rule-1, pick the longest one

New primary sends StartView message to all

Backups replace log with primary's log, update status and lastV

In the same view, a backup's log is always the prefix of a primary's log.

Normal Process

Primary sends the req with curV to all

On receiving Prepare msg, server must:

- ignore if its curV != msg.curV or status != NORMAL
- wait until it has entries in its log for all earlier requests
- append req to log at position msg.op-number

Primary considers a req committed after getting majority OKs.

Recovery

A healed server sends Recovery request to all

On receiving *Recovery* msg, server must:

- Ignore if status != NORMAL
- Reply <curV, x, log, op-number, commit-number>

On receiving *RecoveryReply* from a majority:

- Replace log and curV with replies of latest view
- Change its status to NORMAL

ViewChange

A candidate sends a ViewChange request to all

On receiving ViewChange msg, server does:

- reject if its curV > msg.V
- set curV = msg.V, status= VIEWCHANGE
- return its log and the lastV.

On receiving OK from majority:
S1 decides on the newLog for V4 with following rules:

1. Rule 1: Pick the log w/ biggest lastV
2. Rule 2: if >1 logs exist in rule-1, pick the longest one

New primary sends StartView message to all

Backups replace log with primary's log, update status and lastV

In the same view, a backup's log is always the prefix of a primary's log.

Normal Process

Primary sends the req with curV to all

On receiving Prepare msg, server must:

- ignore if its curV != msg.curV or status != NORMAL
- wait until it has entries in its log for all earlier requests
- append req to log at position msg.op-number

Primary considers a req committed after getting majority OKs.

Recovery

A healed server sends Recovery request to all

On receiving *Recovery* msg, server must:

- Ignore if status != NORMAL
- Reply <curV, x, log, op-number, commit-number>

On receiving *RecoveryReply* from a majority:

- Replace log and curV with replies of latest view
- Change its status to NORMAL

ViewChange

A candidate sends a ViewChange request to all

On receiving ViewChange msg, server does:

- reject if its curV > msg.V
- set curV = msg.V, status= VIEWCHANGE
- return its log and the lastV.

On receiving OK from majority:
S1 decides on the newLog for V4 with following rules:

1. Rule 1: Pick the log w/ biggest lastV
2. Rule 2: if >1 logs exist in rule-1, pick the longest one

New primary sends StartView message to all

Backups replace log with primary's log, update status and lastV

In any majority, there exist a backup has all committed requests in its log.

Normal Process

Primary sends the req with curV to all

On receiving Prepare msg, server must:

- ignore if its curV != msg.curV or status != NORMAL
- wait until it has entries in its log for all earlier requests
- append req to log at position msg.op-number

Primary considers a req committed after getting majority OKs.

Recovery

A healed server sends Recovery request to all

On receiving *Recovery* msg, server must:

- Ignore if status != NORMAL
- Reply <curV, x, log, op-number, commit-number>

On receiving *RecoveryReply* from a majority:

- Replace log and curV with replies of latest view
- Change its status to NORMAL

ViewChange

A candidate sends a ViewChange request to all

On receiving ViewChange msg, server does:

- reject if its curV > msg.V
- set curV = msg.V, status= VIEWCHANGE
- return its log and the lastV.

On receiving OK from majority:

S1 decides on the newLog for V4 with following rules:

1. Rule 1: Pick the log w/ biggest lastV
2. Rule 2: if >1 logs exist in rule-1, pick the longest one

New primary sends StartView message to all

Backups replace log with primary's log, update status and lastV

The new primary of V is able to collect all committed requests in V – 1.

Correctness: servers execute the same sequence of log

↓

No two different ops are committed at the same log position

↓

All committed ops in view V - 1 must be known to primary in view V

↓

In any majority, there exist a backup has all committed requests in its log.
New primary collects safe log from a majority.

↓

In the same view, a backup's log is always the prefix of
a primary's log.

At the beginning of a view, backup's logs are identical with primary's log.

Request is committed in the same position of log in a majority of servers.

Before put a request into the log, wait until all earlier log entries are appended to the log

Recover by replacing the log from a majority