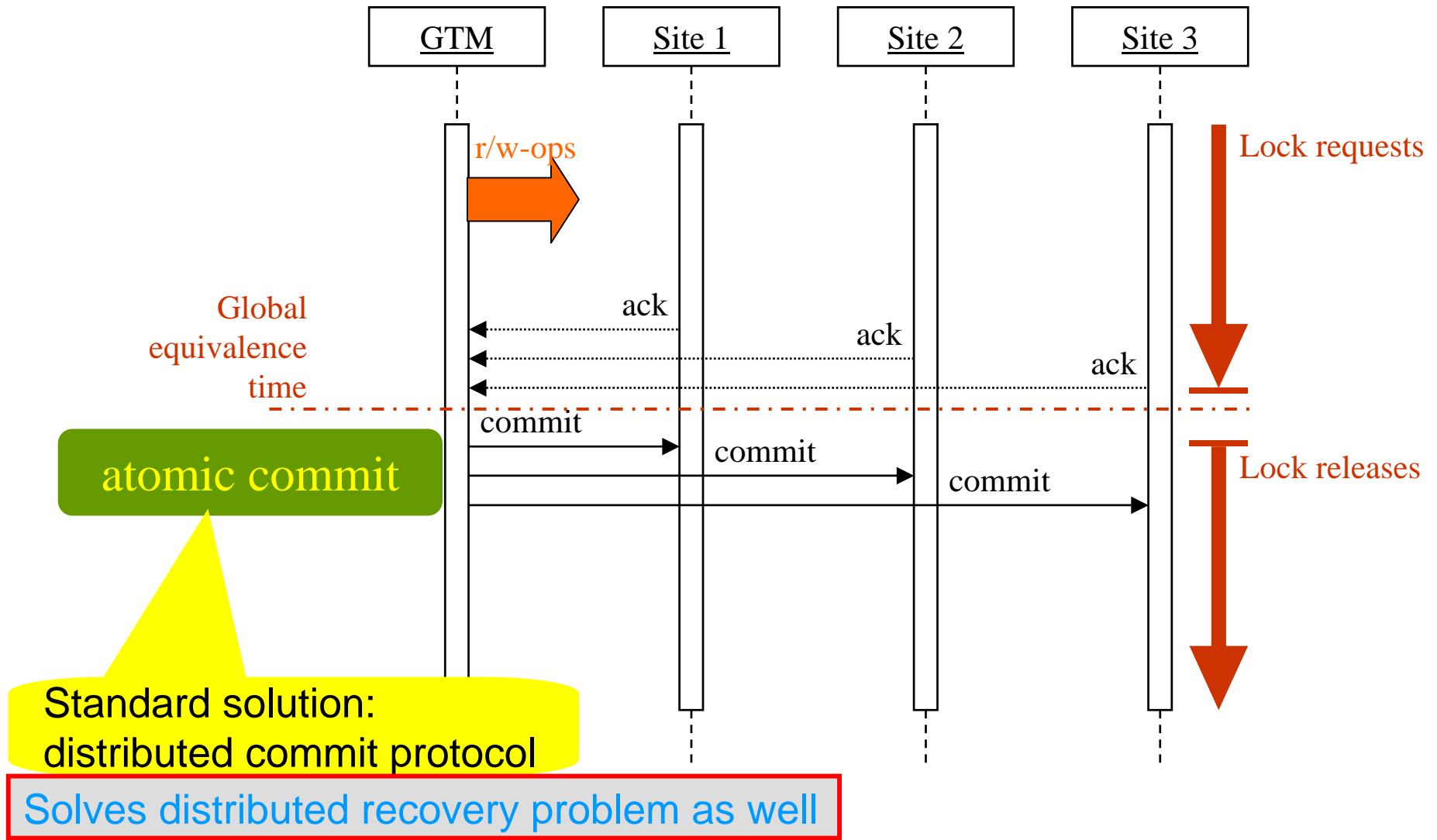


Chapter 11

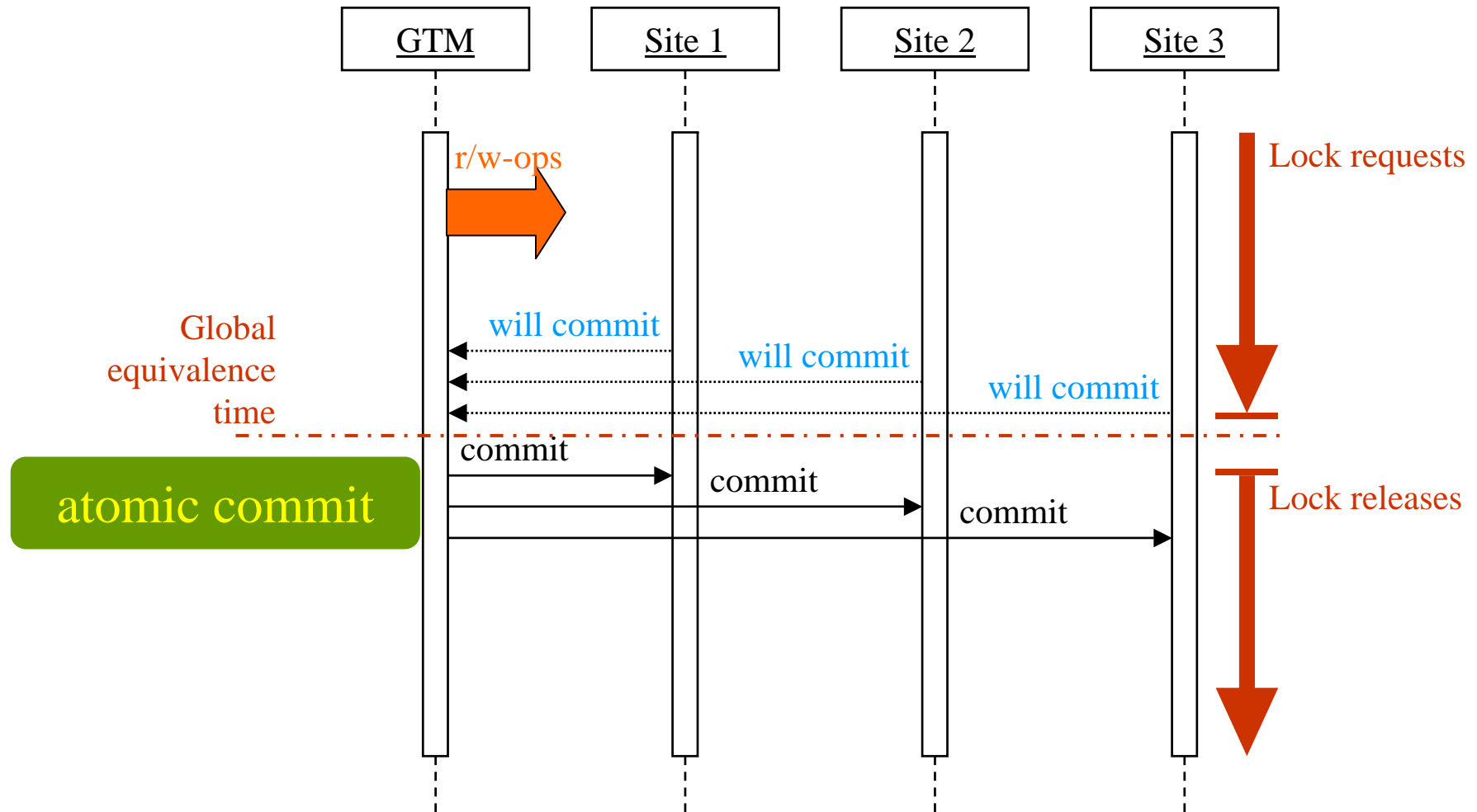
Distributed Transactions: Recovery

Global Atomicity

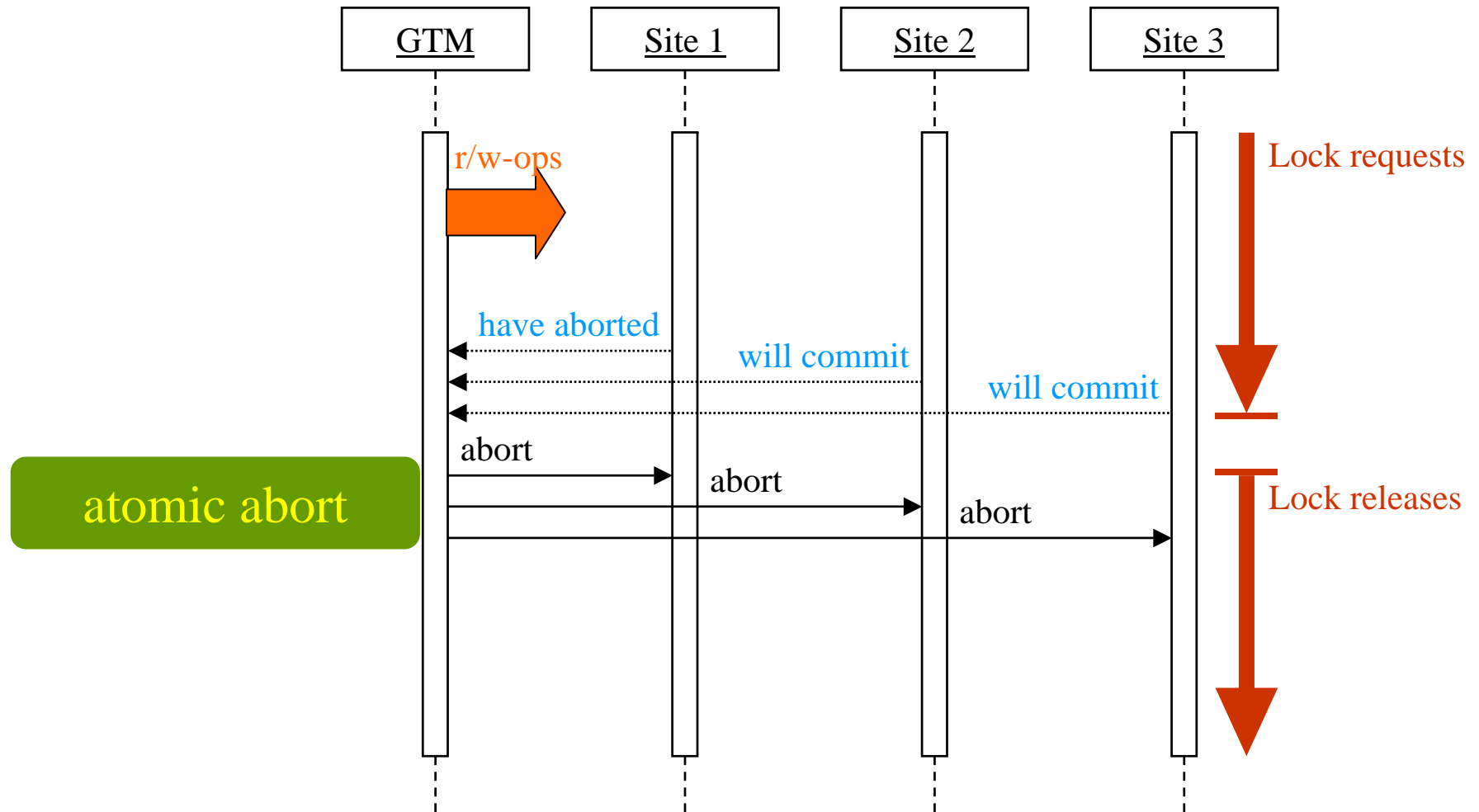
Remember: Rigorous / commit-deferred



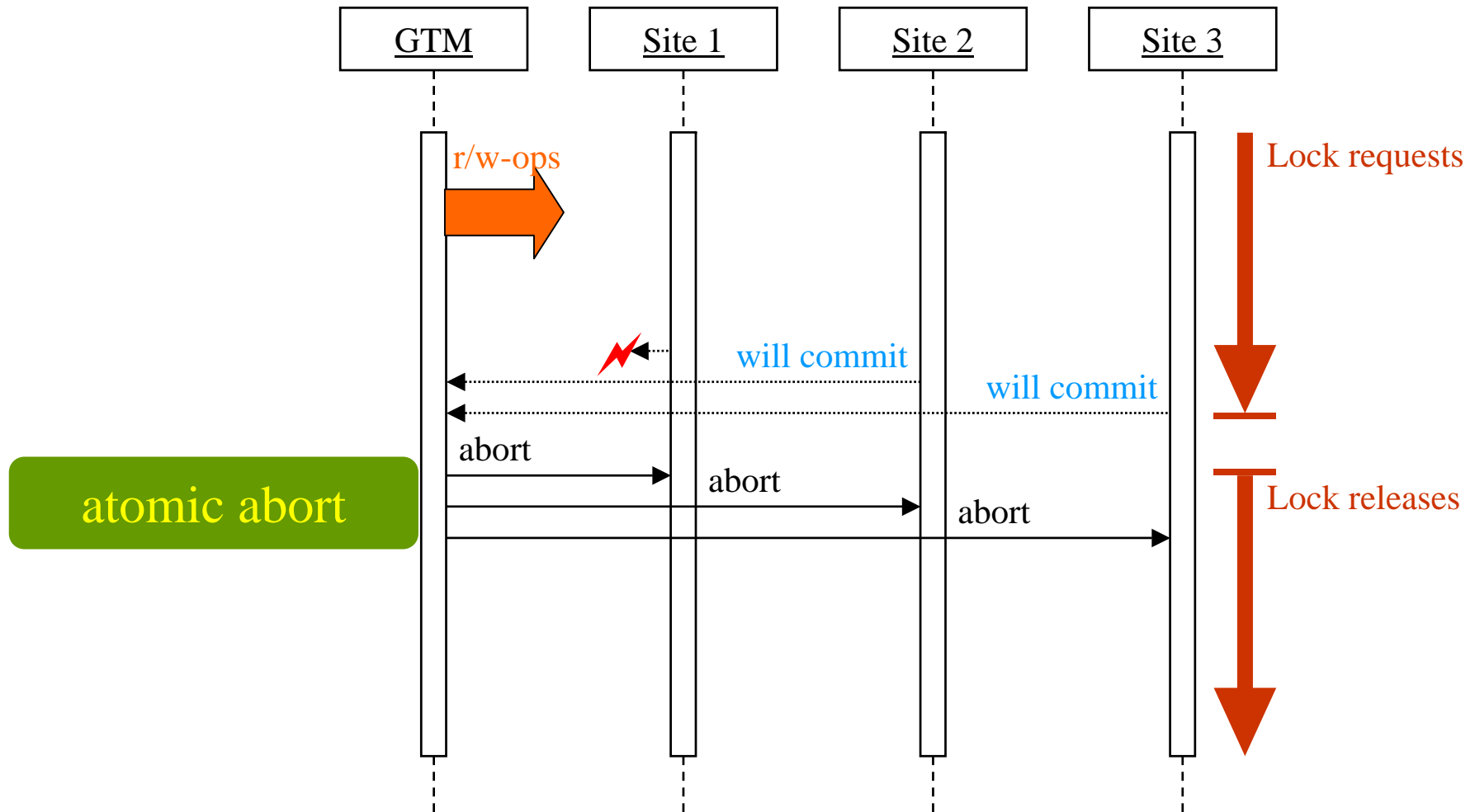
Atomic commitment



Atomic abort



Presumed abort



atomic abort

Atomicity

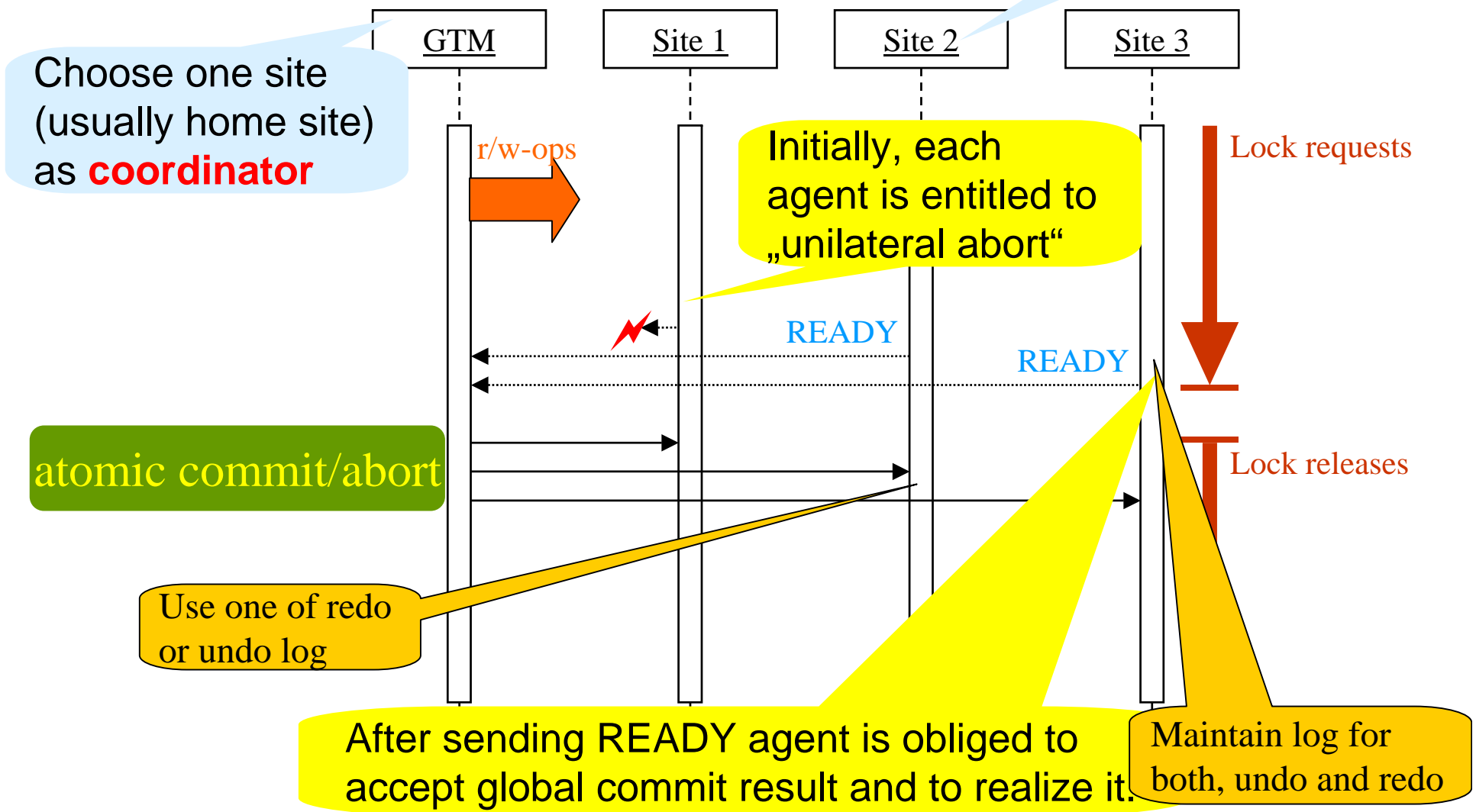
7

- Prerequisite of **local atomicity**: All transactions at all nodes are atomic. \Rightarrow Each node with a complete transaction manager as discussed earlier.
- Problem of **global atomicity**: All nodes decide exactly the same for all distributed transactions, i.e., global commit or abort, and enforce the decision locally.
- **Atomic Commit Protocols (ACP)** ensure that all nodes participating in a global transaction all either commit or abort this transaction.

Atomic commit protocol

Other sites are **participants** (or **agents**)

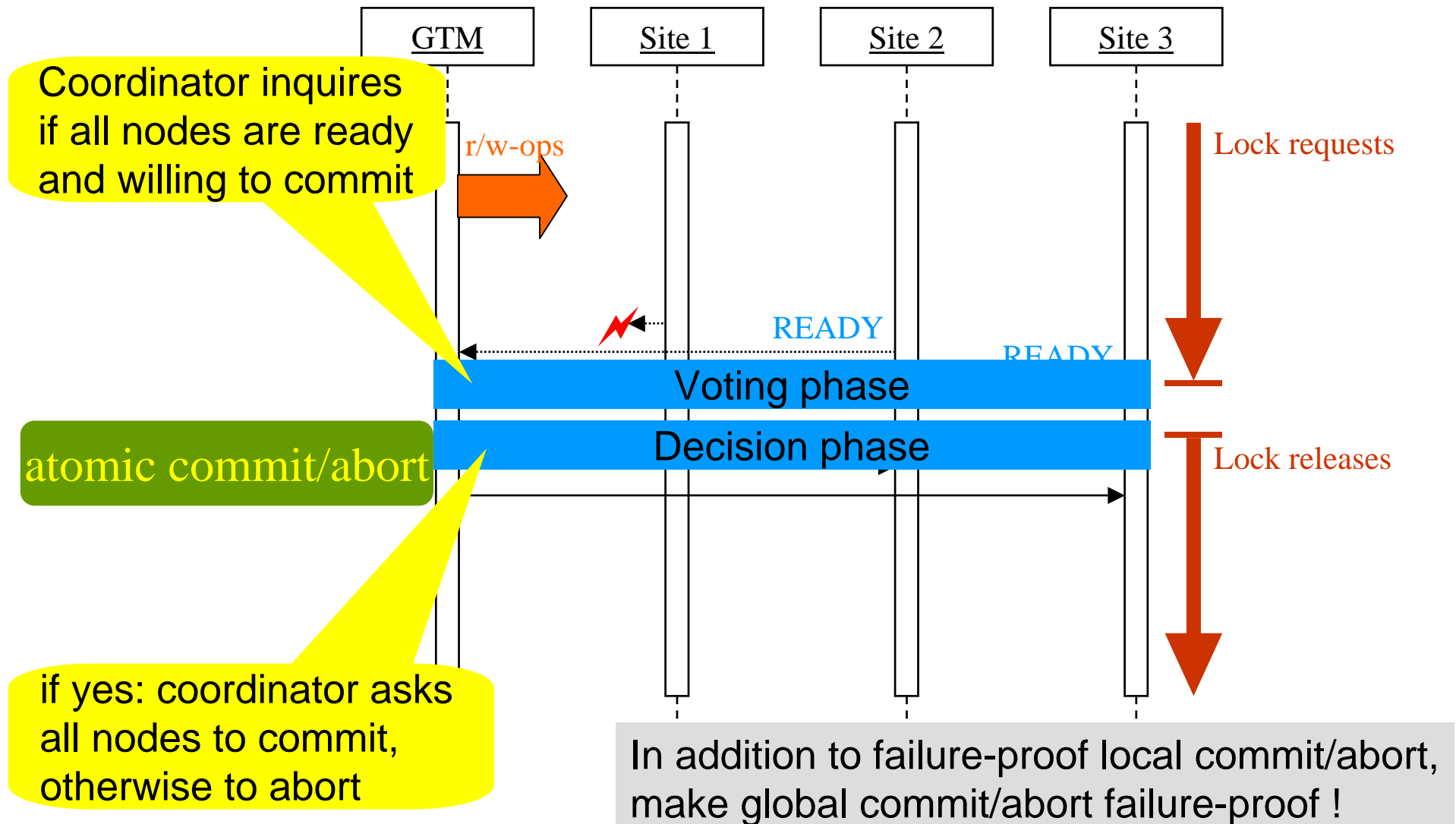
8



Basis: Two-phase commit protocol (2PC)

Two-phase commit protocol (2PC)

10



Preconditions (1)

Rule 1: No site may be forced to commit a transaction.

- ⇒ All involved participants must consent to a *global commit* for the TA coordinator to issue such a commit.
- ⇒ Otherwise the coordinator must issue a *global abort*.
- ◆ Participants where t only invoked read operations can safely be skipped.

Preconditions (2)

Rule 2: A transaction t may only be committed if all written items already are locally persistent.

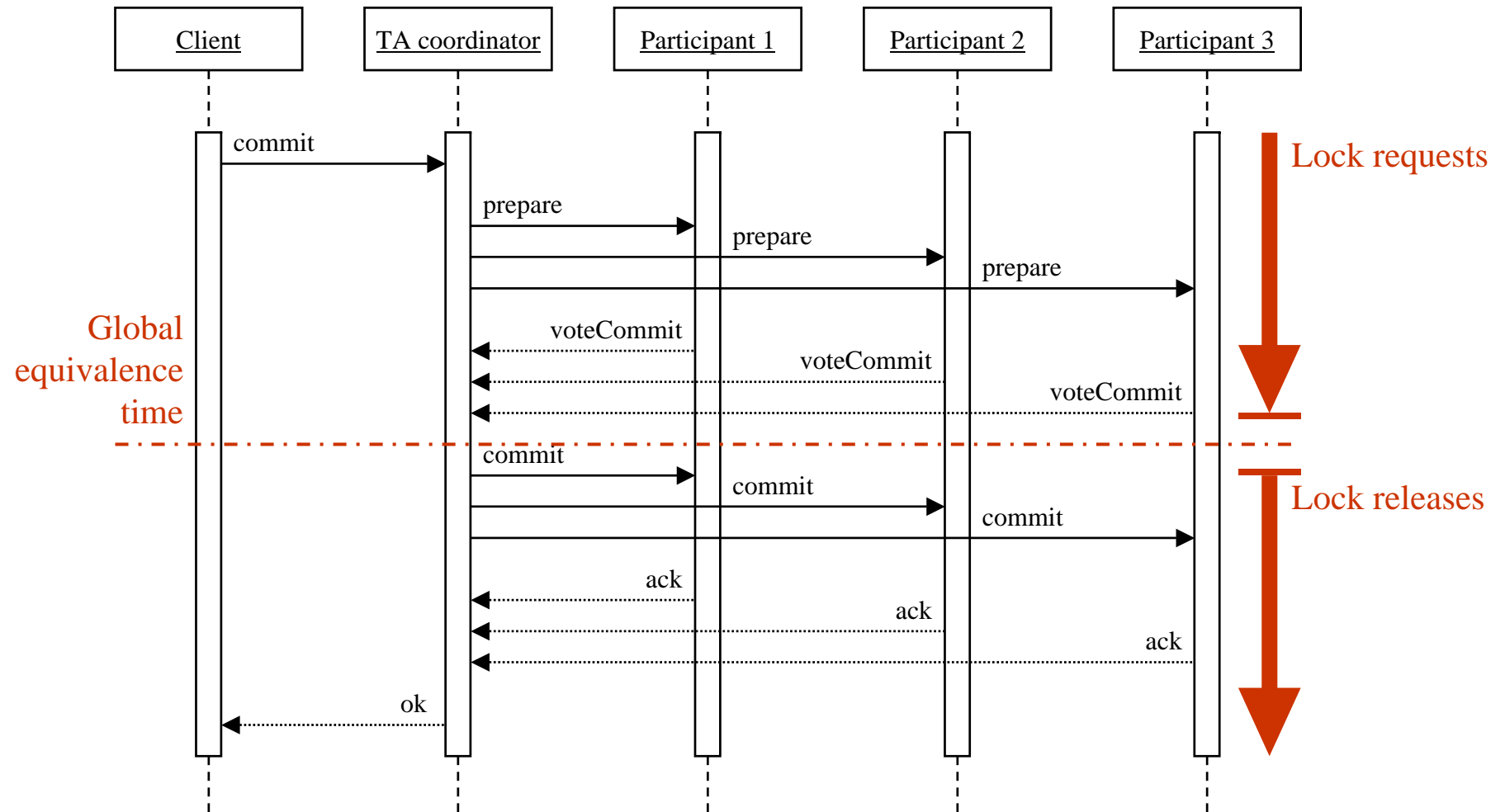
- Each site guarantees with its consent to t that
 - ◆ RC is satisfied,
 - ◆ Undo/Redo rule is satisfied,
 - ◆ the site will not commit other transactions that follow t in the local equivalent serial history before it received the coordinator's final decision,
 - ◆ If the site fails it can determine the outcome of t and if need be will be able to redo t .

Preconditions (3)

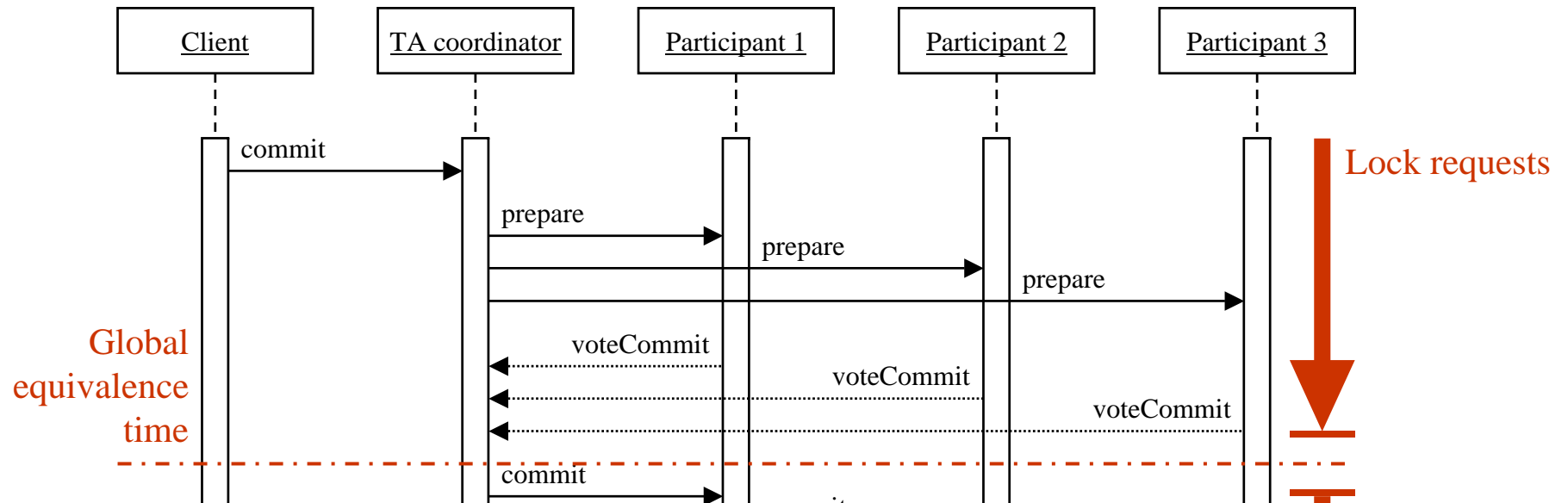
Rule 3: Participants must be informed of a global commit.

- Coordinator guarantees this to the consenting participants by maintaining a list of participants and the transaction commit in stable store until all participants have been informed.
- Presumed abort:
 - ◆ Participants that did not consent guarantee that they undo the local changes of the transaction and release the locks. They are not included in the further communication.
 - ◆ Participants that receive *global abort* from the coordinator do likewise.

Global commit via 2PC



Global commit via 2PC



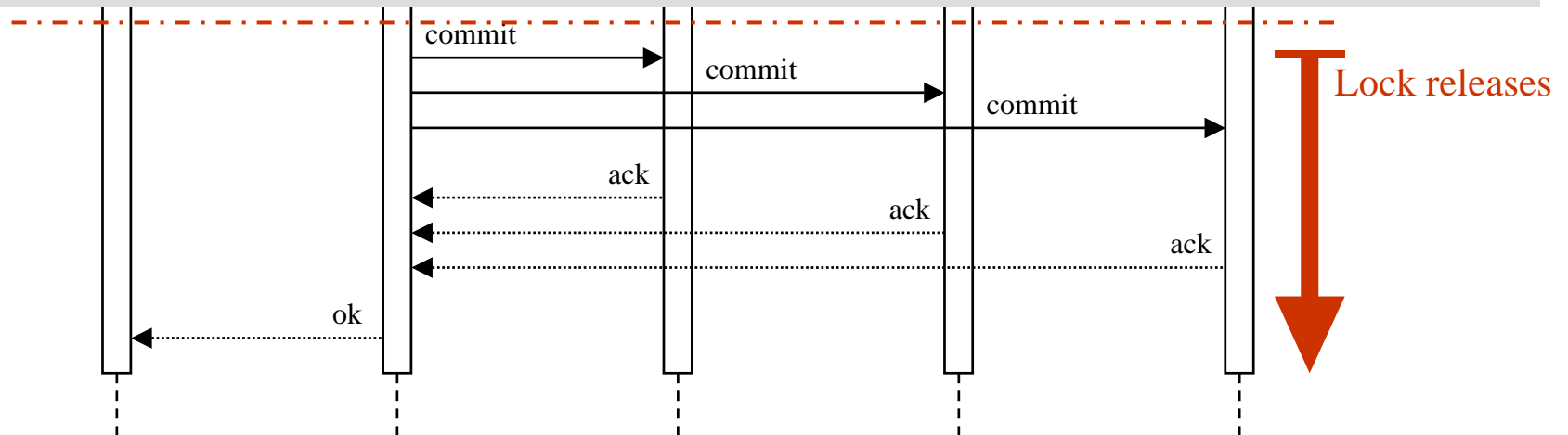
Voting phase:

- a) Coordinator sends *prepare* message to each participant.
- b) After arrival of *prepare*, the participant responds with *voteCommit* or *voteRollback*. In case of *voteRollback* the participant locally aborts the transaction.
 - Each participant has exactly one vote.
 - The participant must not revise its vote.

Global commit via 2PC

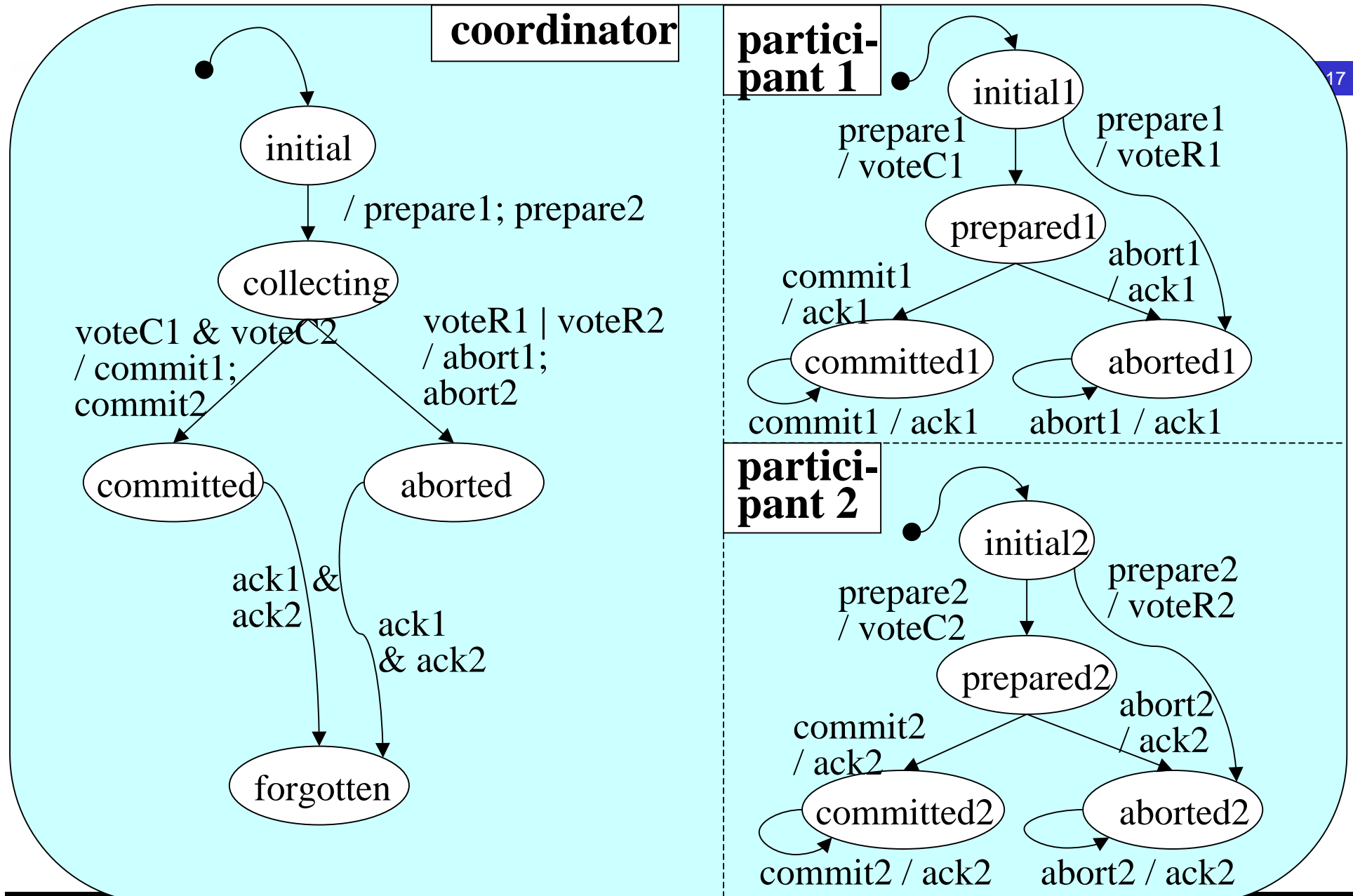
Decision phase:

- The coordinator collects the votes including its own. If all are *voteCommit* it decides on global commit and sends *commit* to all participants.
- Otherwise it decides on global abort and sends *abort* to all participants that voted *voteCommit*.
- Each participant that voted *voteCommit* waits for a *commit* or *abort* message, and after receiving it processes it accordingly.



Statechart for Basic 2PC

© Weikum, Vossen, 2002

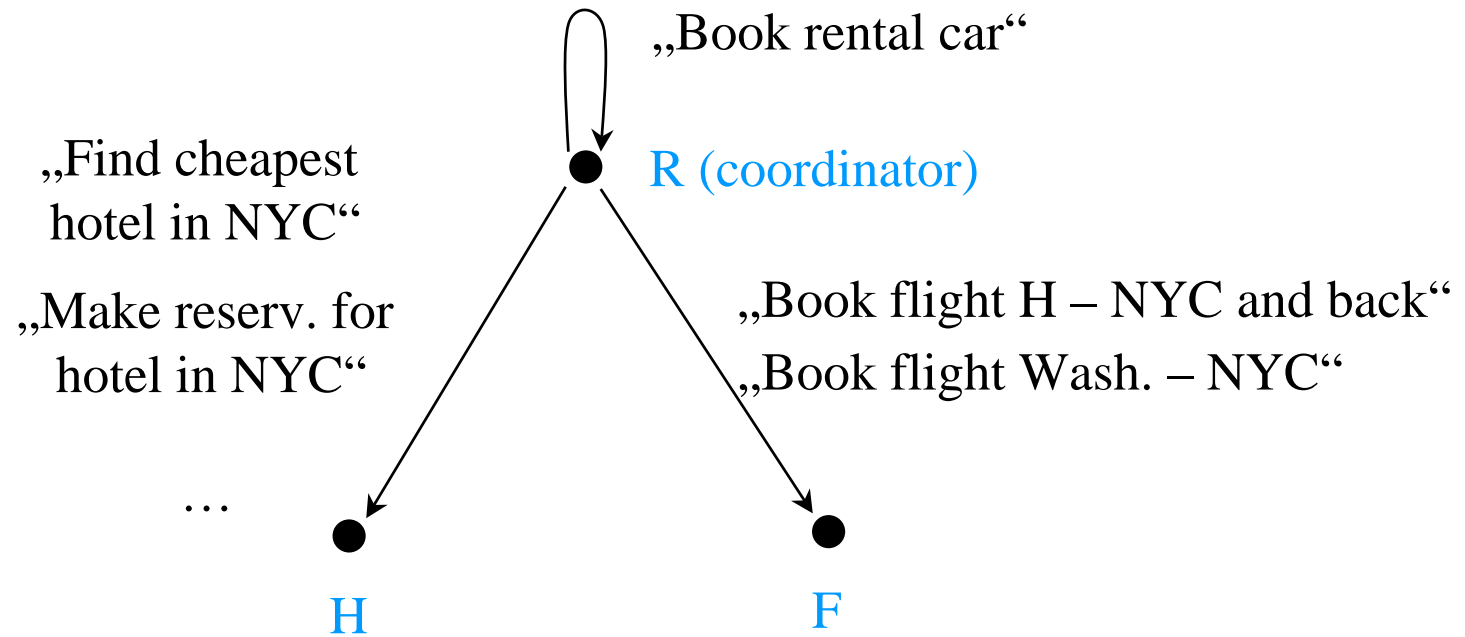


Scenario

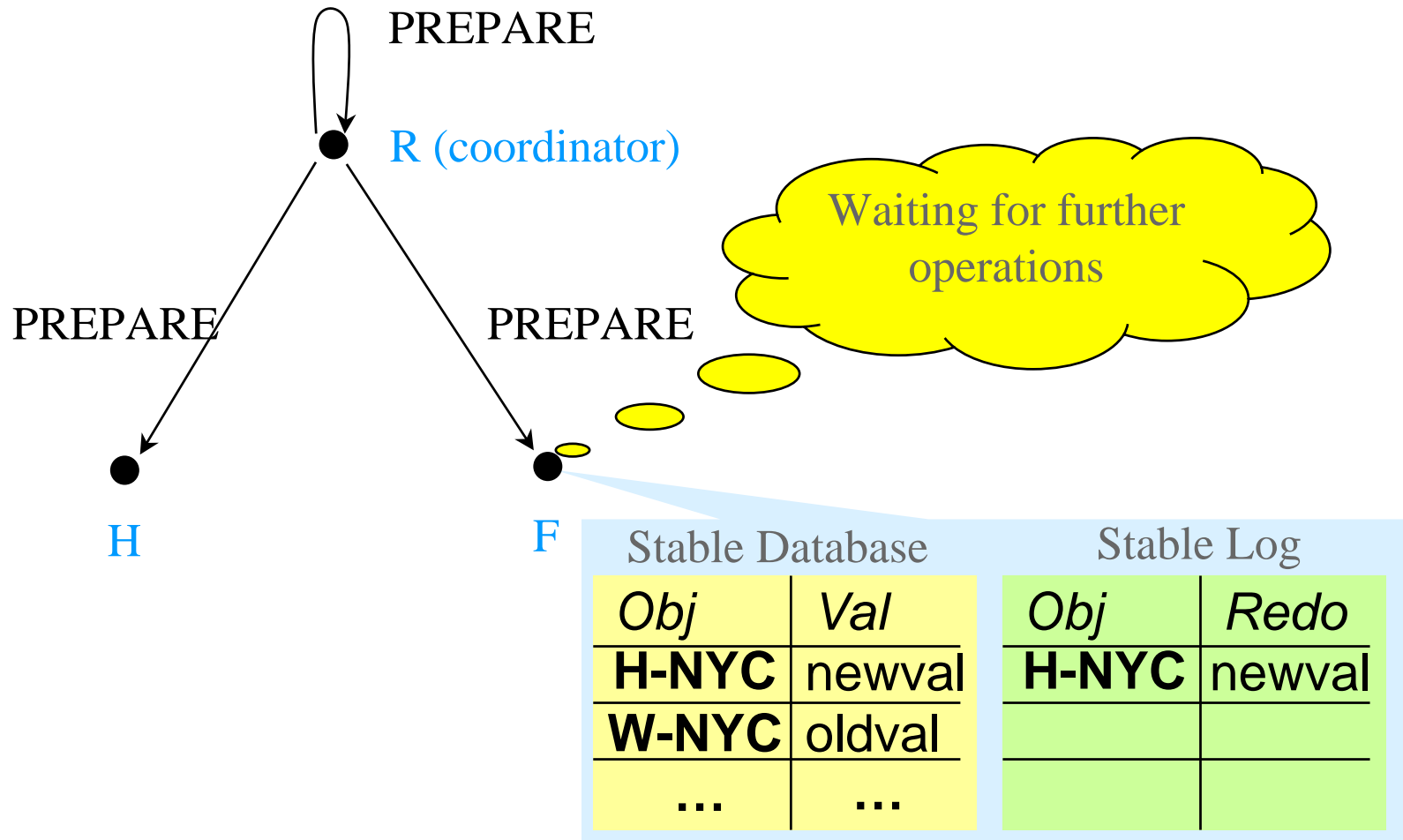
Scenario

- Transaction t that consists of several operations:
 - ◆ “Book flight from Hannover to New York City and back.”
 - ◆ “Book flight from Washington DC to NYC.”
 - ◆ “Find cheapest hotel in NYC.”
 - ◆ “Make reservation for it.”
 - ◆ “Make reservation for any hotel in Philadelphia.”
 - ◆ dto. Washington DC
 - ◆ “Book rental car NYC – Washington DC.”
-
- Flight DB
(Node F)
- Hotel DB
(Node H)
- Rental Car DB
(Node R)

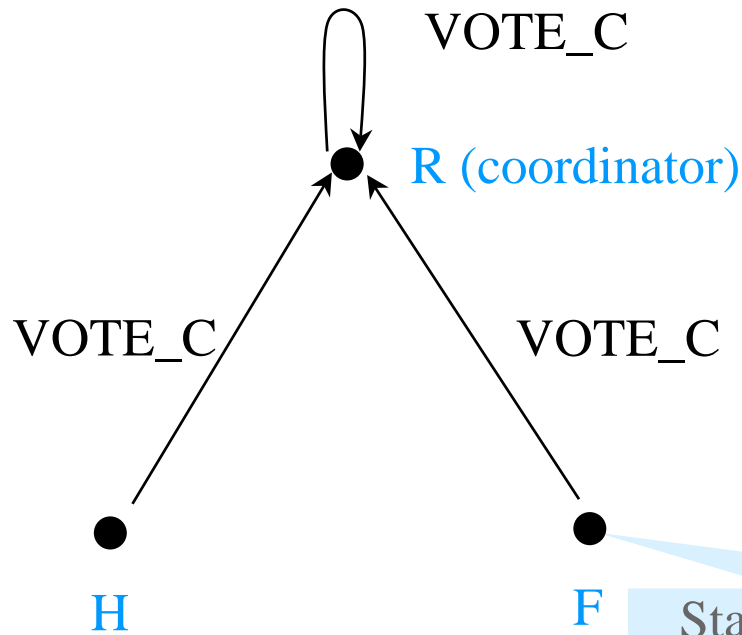
Transaction execution - no failures



2PC execution – regular commit (1)

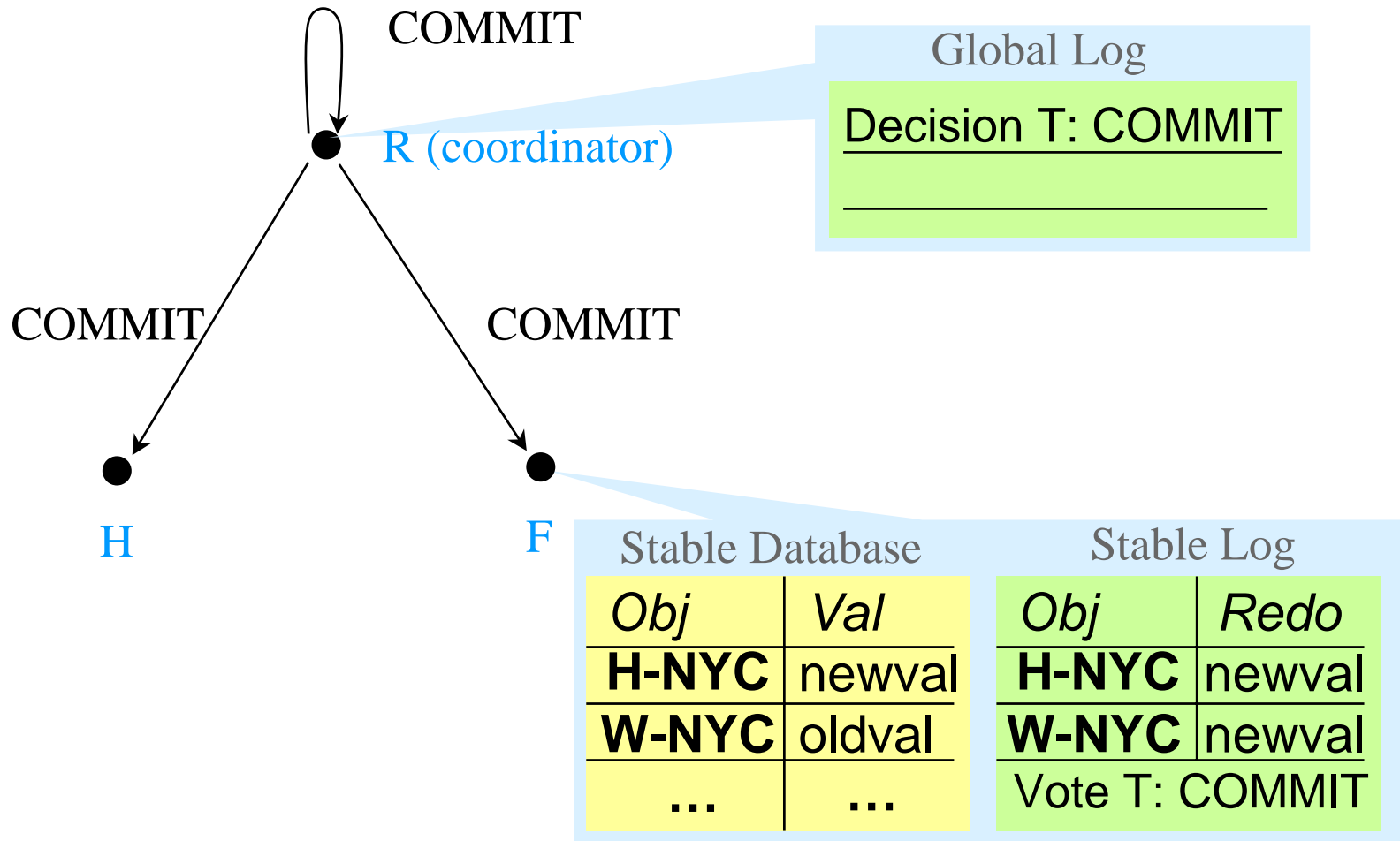


2PC execution – regular commit (2)

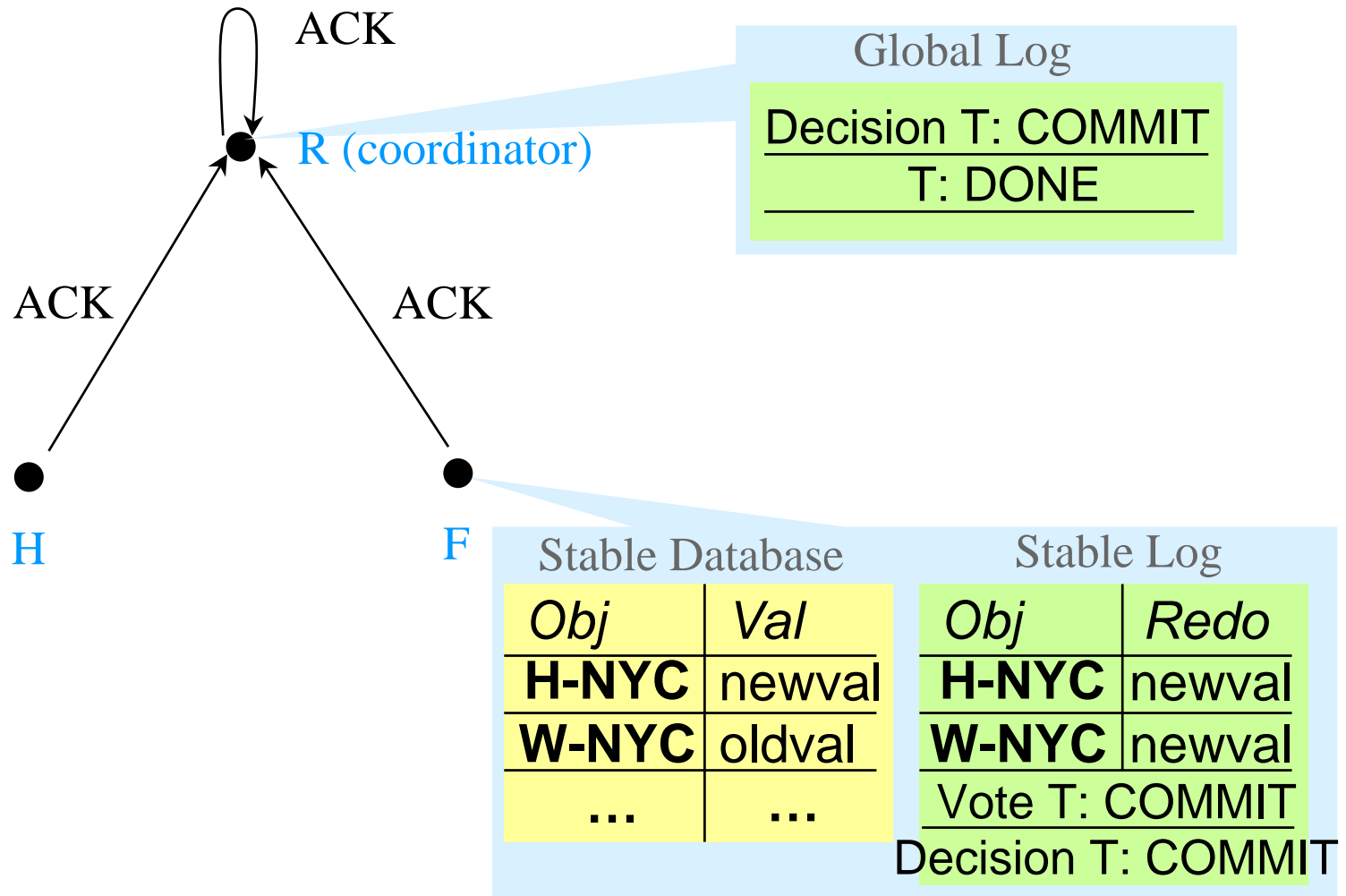


Stable Database		Stable Log	
<i>Obj</i>	<i>Val</i>	<i>Obj</i>	<i>Redo</i>
H-NYC	newval	H-NYC	newval
W-NYC	oldval	W-NYC	newval
...	...	Vote T: COMMIT	

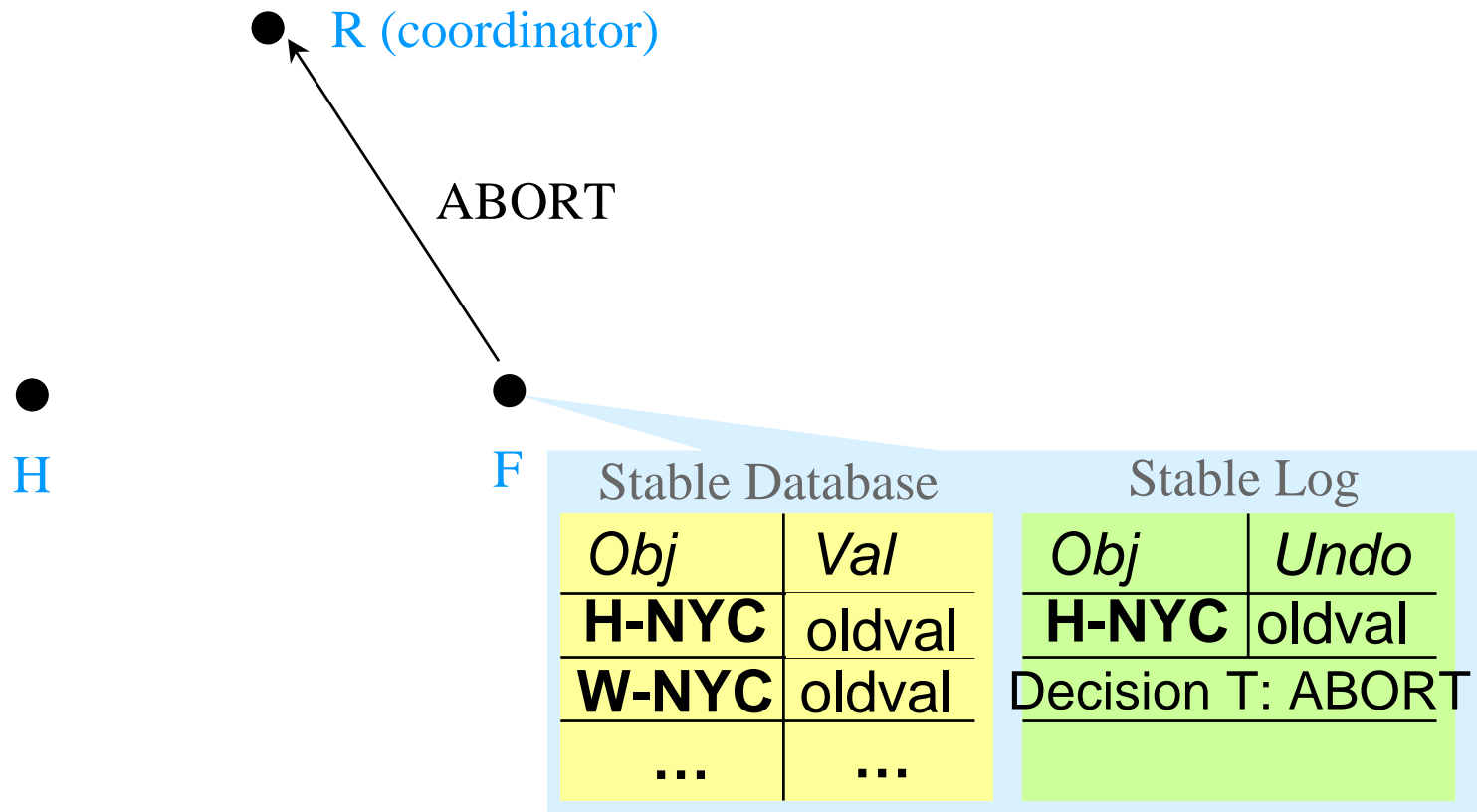
2PC execution – regular commit (3)



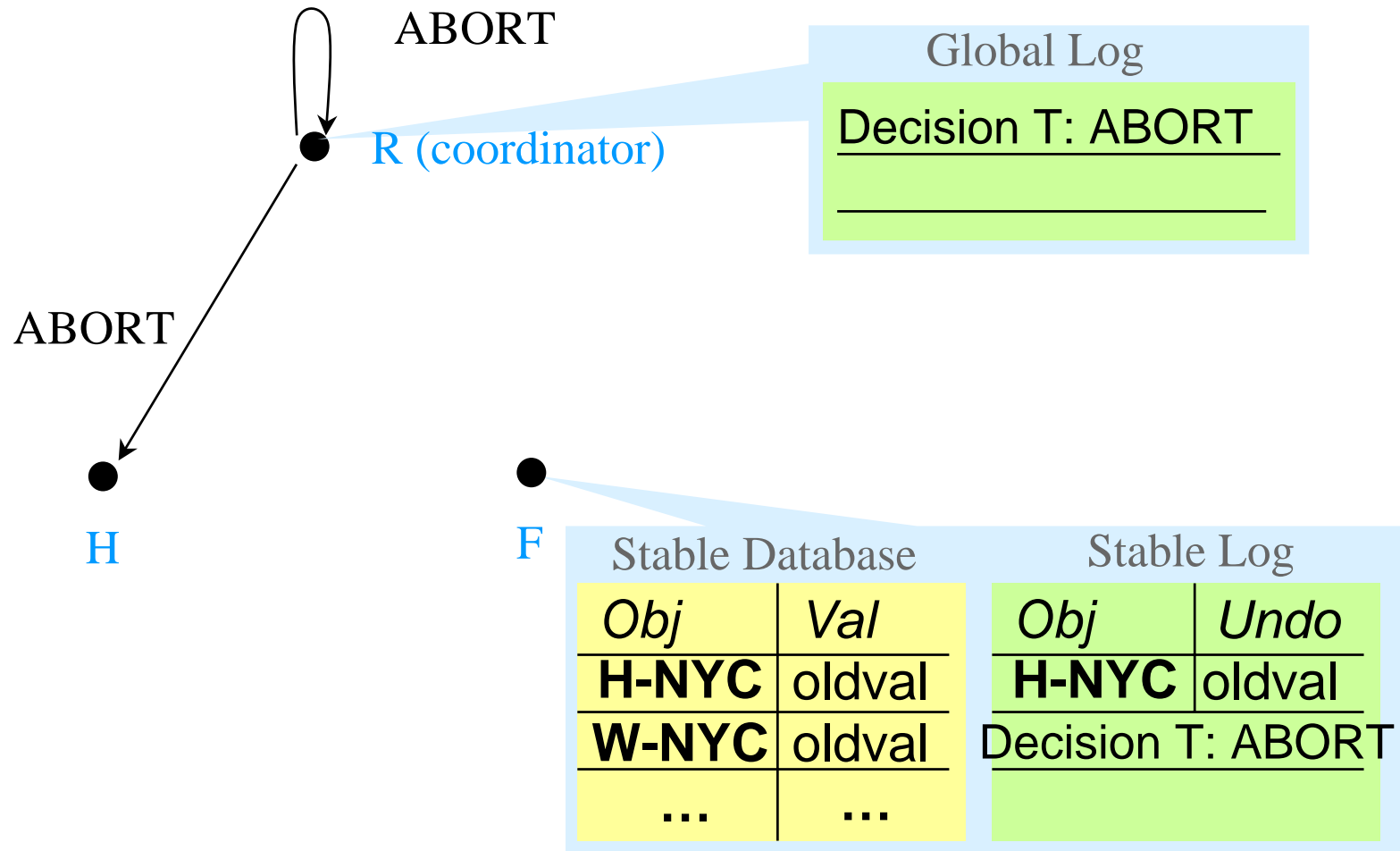
2PC execution – regular commit (4)



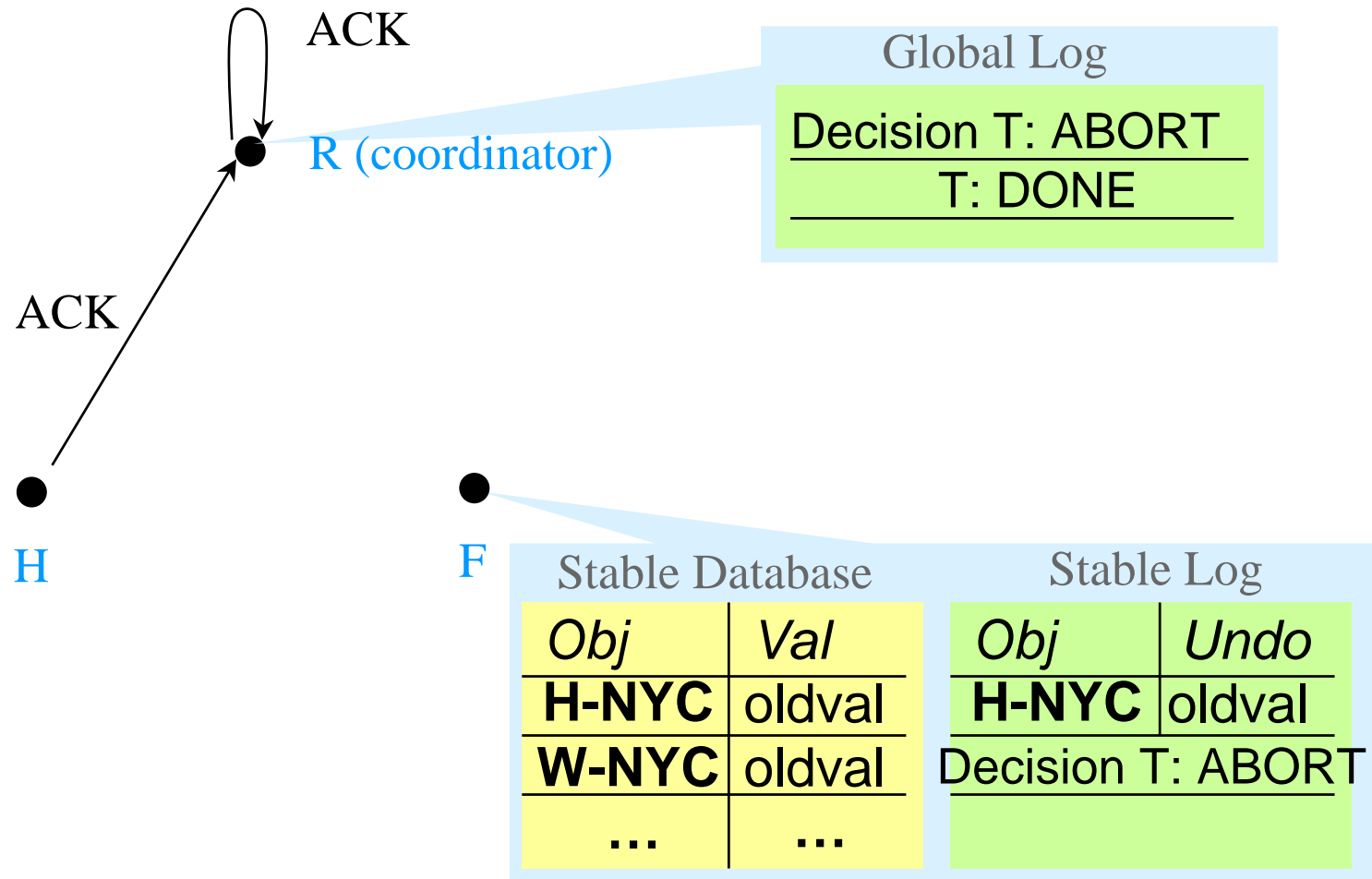
2PC execution – regular abort (1)



2PC execution – regular abort (2)



2PC execution – regular abort (3)



2PC under failures

Failure types

Site failure

- Local system crash. Correctly functioning sites are up, otherwise down.
- If all sites are down: **total failure**.
- If only some sites are down: **partial failure**.

Communication failure

- Connection loss
 - ◆ Assumption 1: Message simply disappears, no attempt at repeated delivery.
 - ◆ Assumption 2: No other failures: no message corruption or loss.

⇒ **Common effect: undeliverable message**

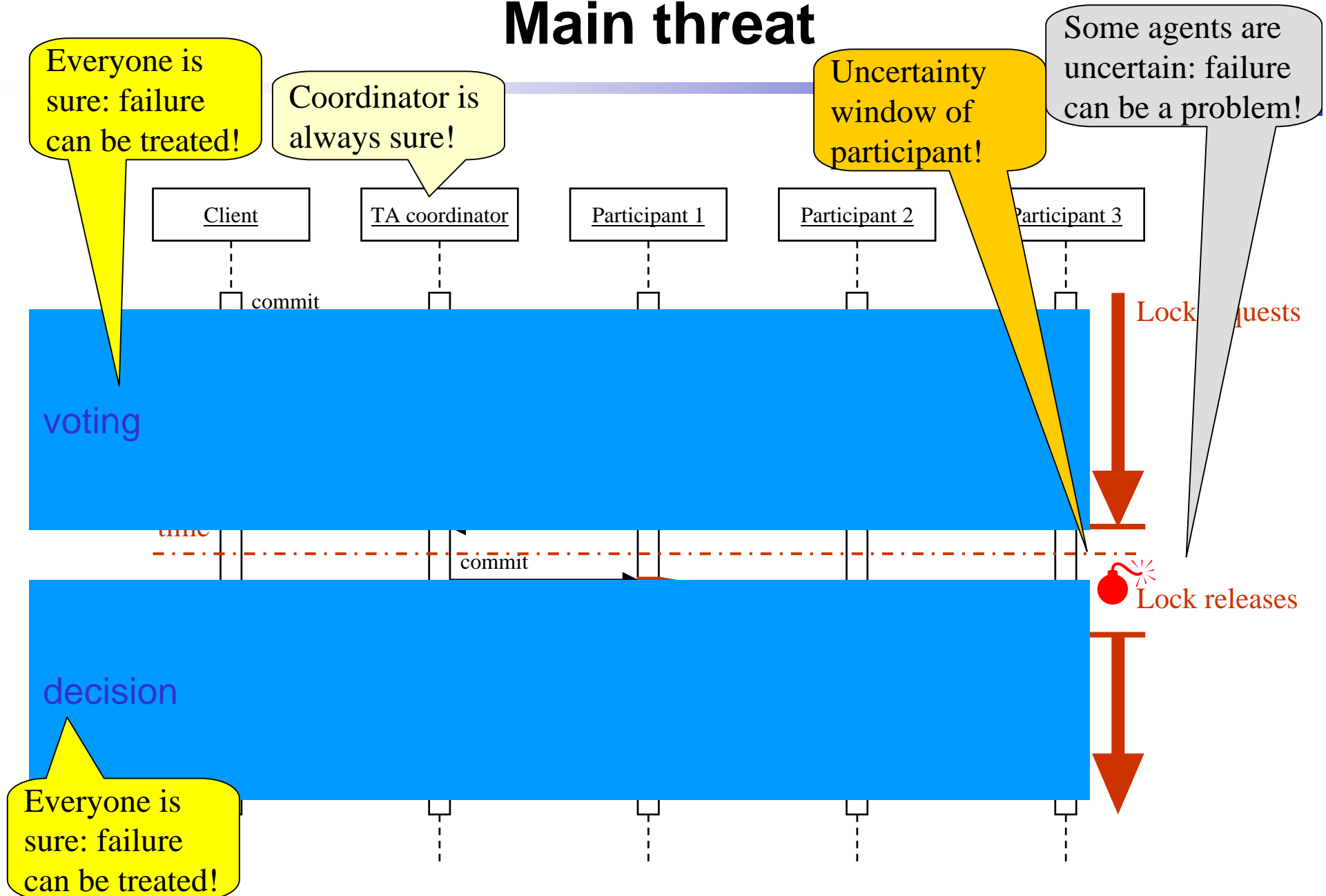
- Terminology: Site is **up** if it can be reached, site is **down** otherwise (site failure or connection loss).

Atomicity: constituent protocols

30

- **Completion protocol** aspects:
 - ◆ **Global commit.**
 - ◆ **Global abort.** } combined in 2PC ✓
- **Termination protocol** for failure recovery. ←

Main threat

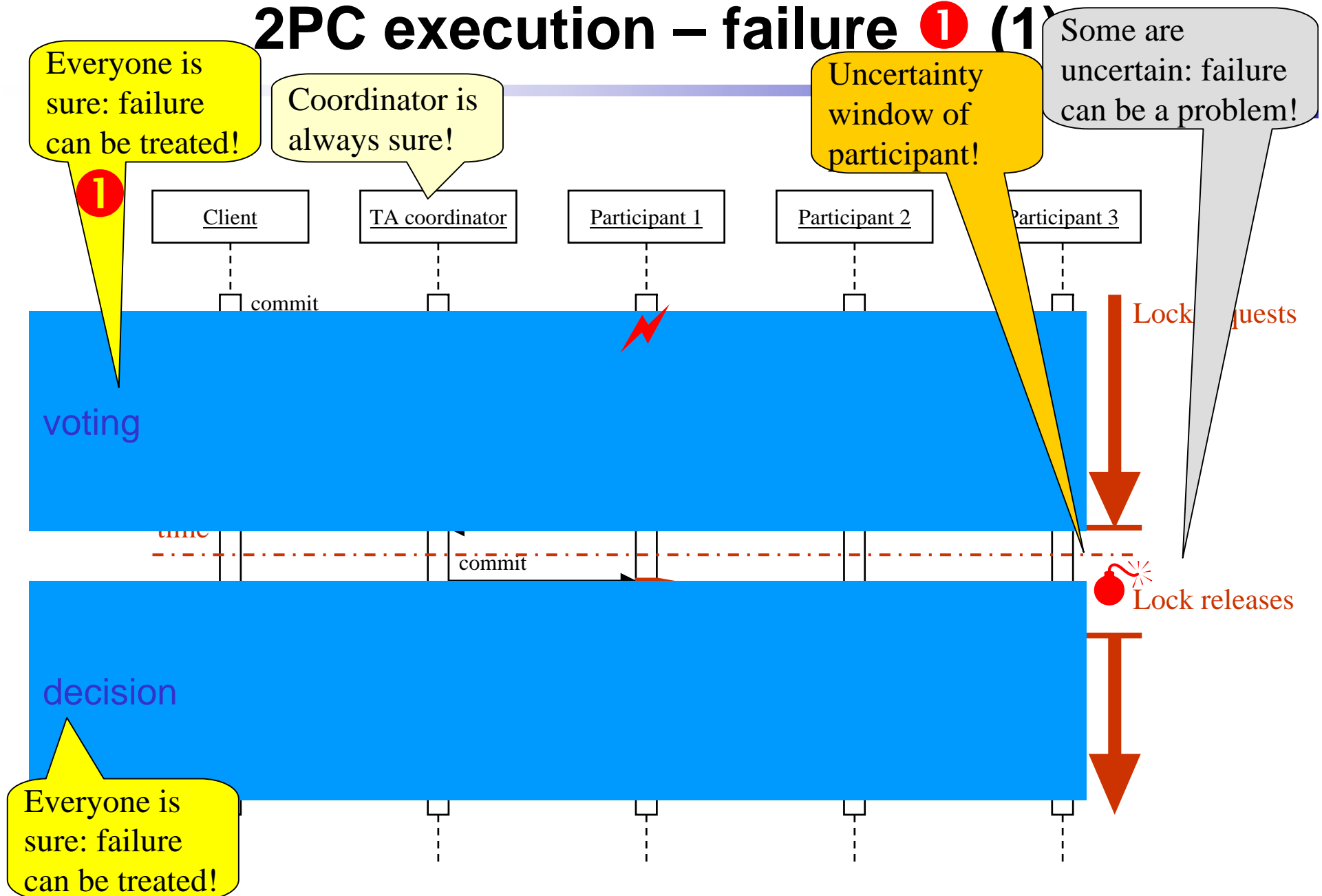


Distributed Transaction Log

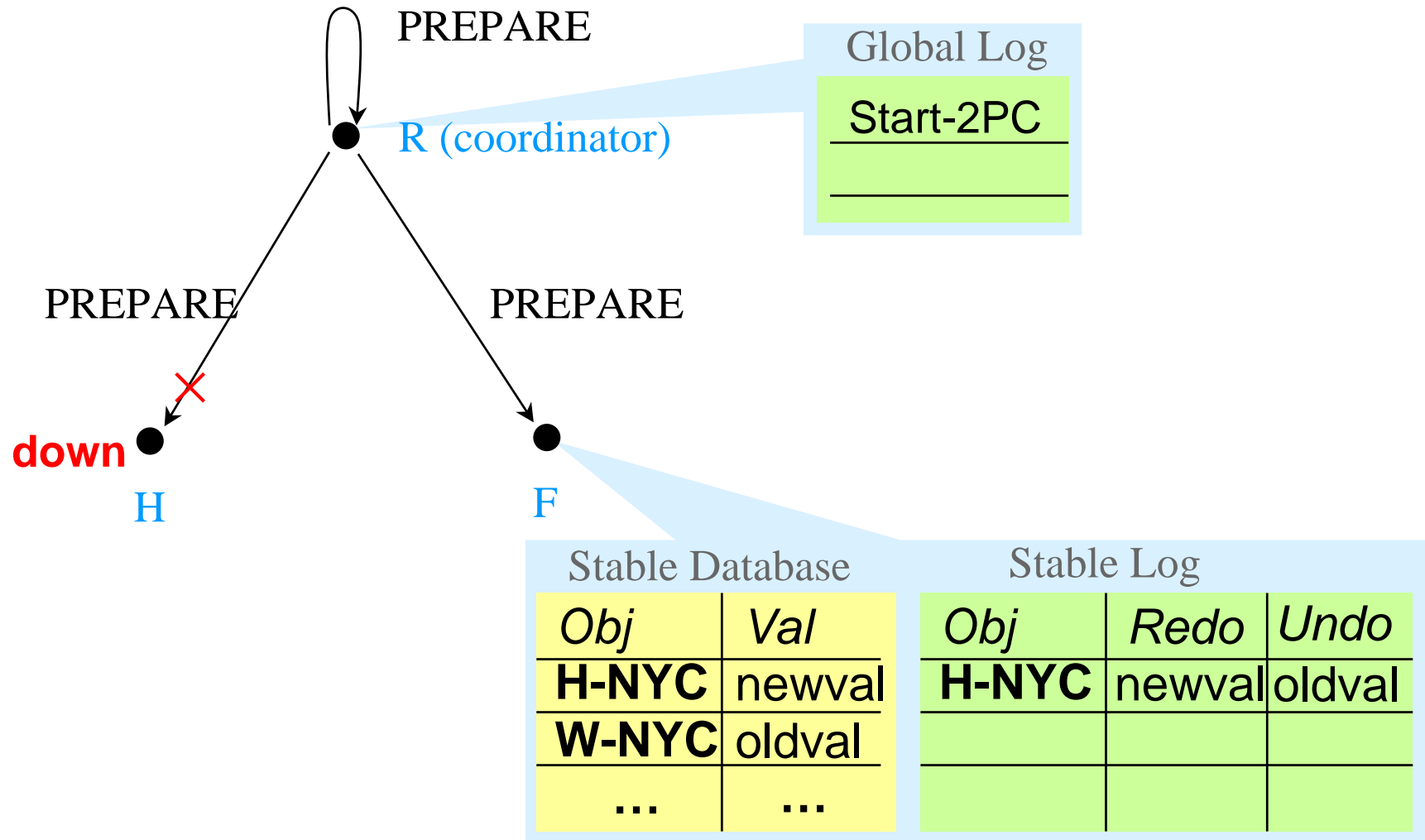
Failure of a site \Rightarrow After its restart its TM must take a decision (commit or abort) that agrees with the decision of all other nodes (coordinator and participants).

- **Refined:** Each site maintains a **Distributed Transaction Log** (DTL) and updates it as follows:
 - ◆ **Coordinator** writes **Start-2PC** with IDs of the participants into its DTL.
 - ◆ A consenting participant writes **voteCommit** and the IDs of the other nodes to its DTL. A disagreeing participant writes **abort** to its DTL.
 - ◆ **Coordinator** writes **commit** or **abort** to its DTL before sending the respective message.
 - ◆ The participants write the received decision to their DTL.
 - ◆ **Coordinator** writes **done** after receiving the acknowledgements of all participants.

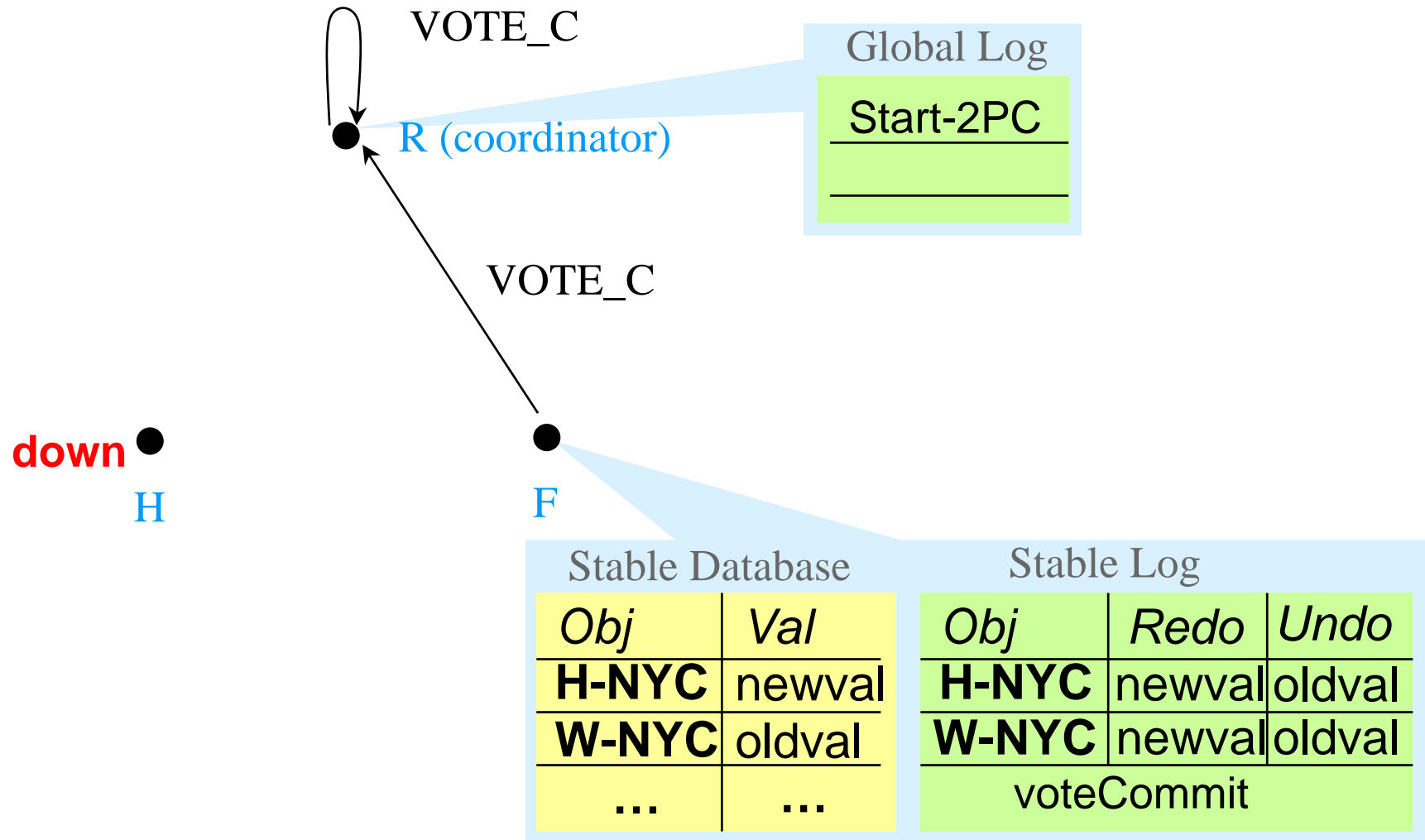
2PC execution – failure 1 (1)



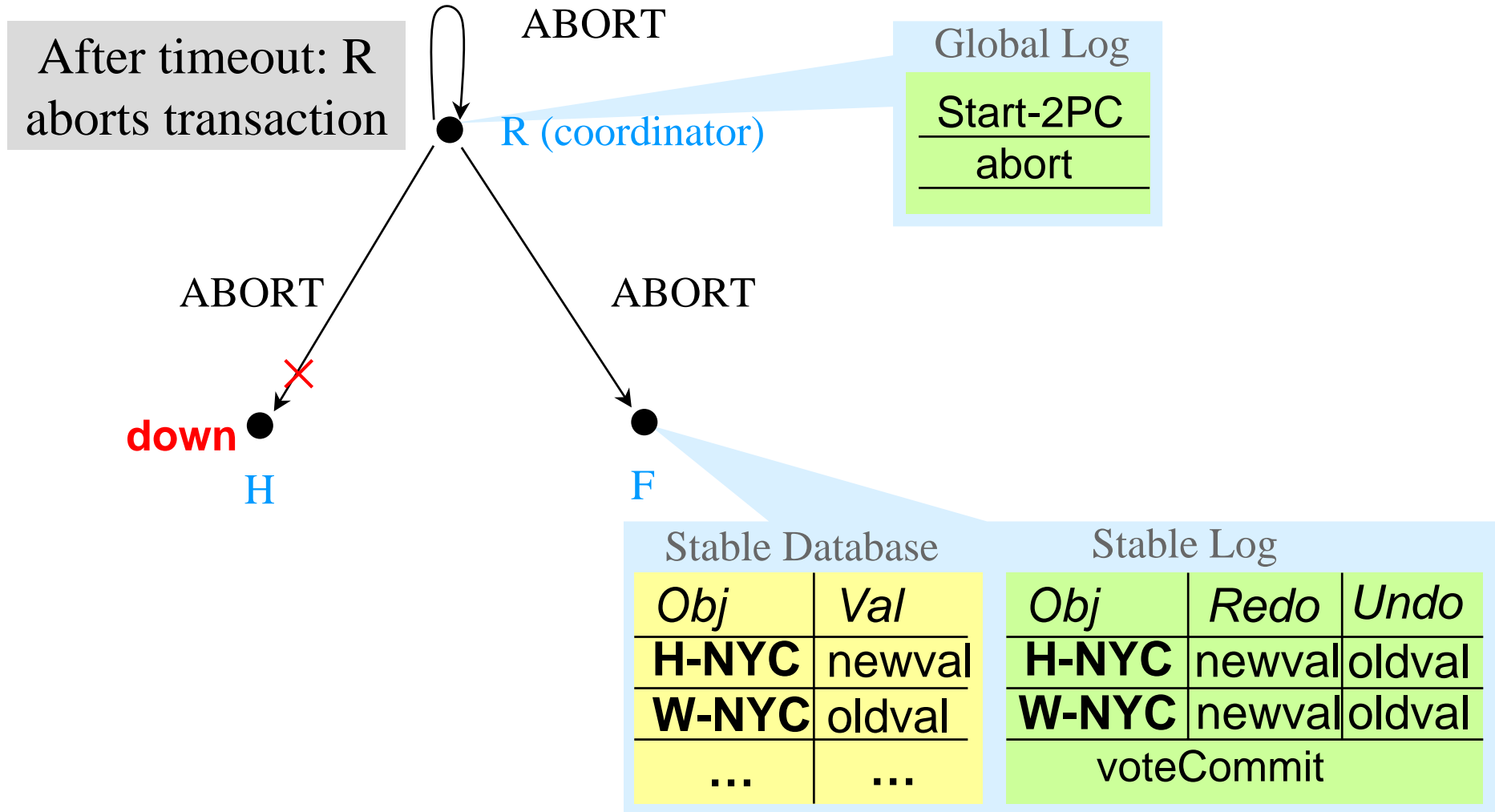
2PC execution – failure 1 (2)



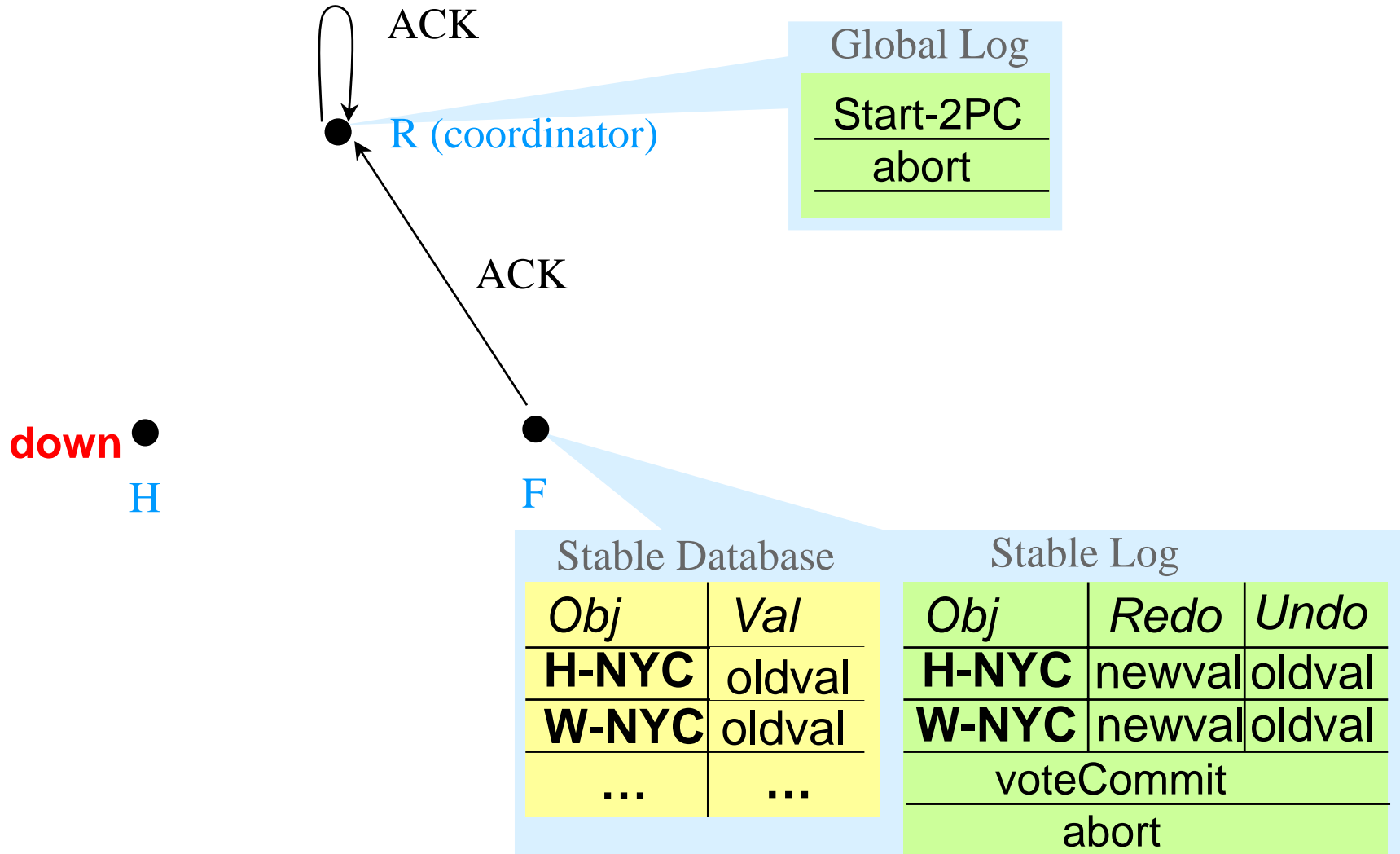
2PC execution – failure 1 (3)



2PC execution – failure 1 (4)



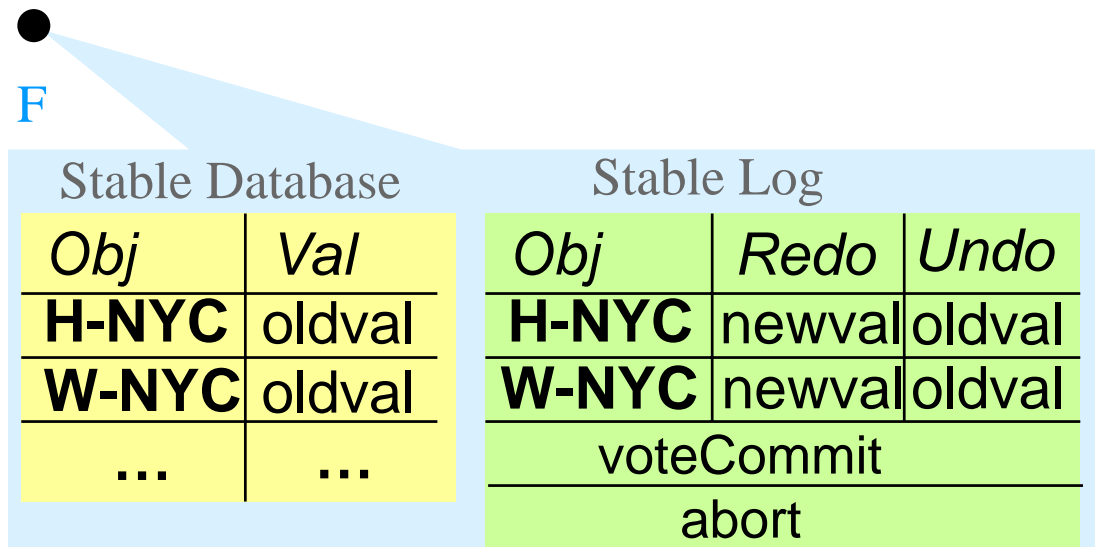
2PC execution – failure 1 (5)



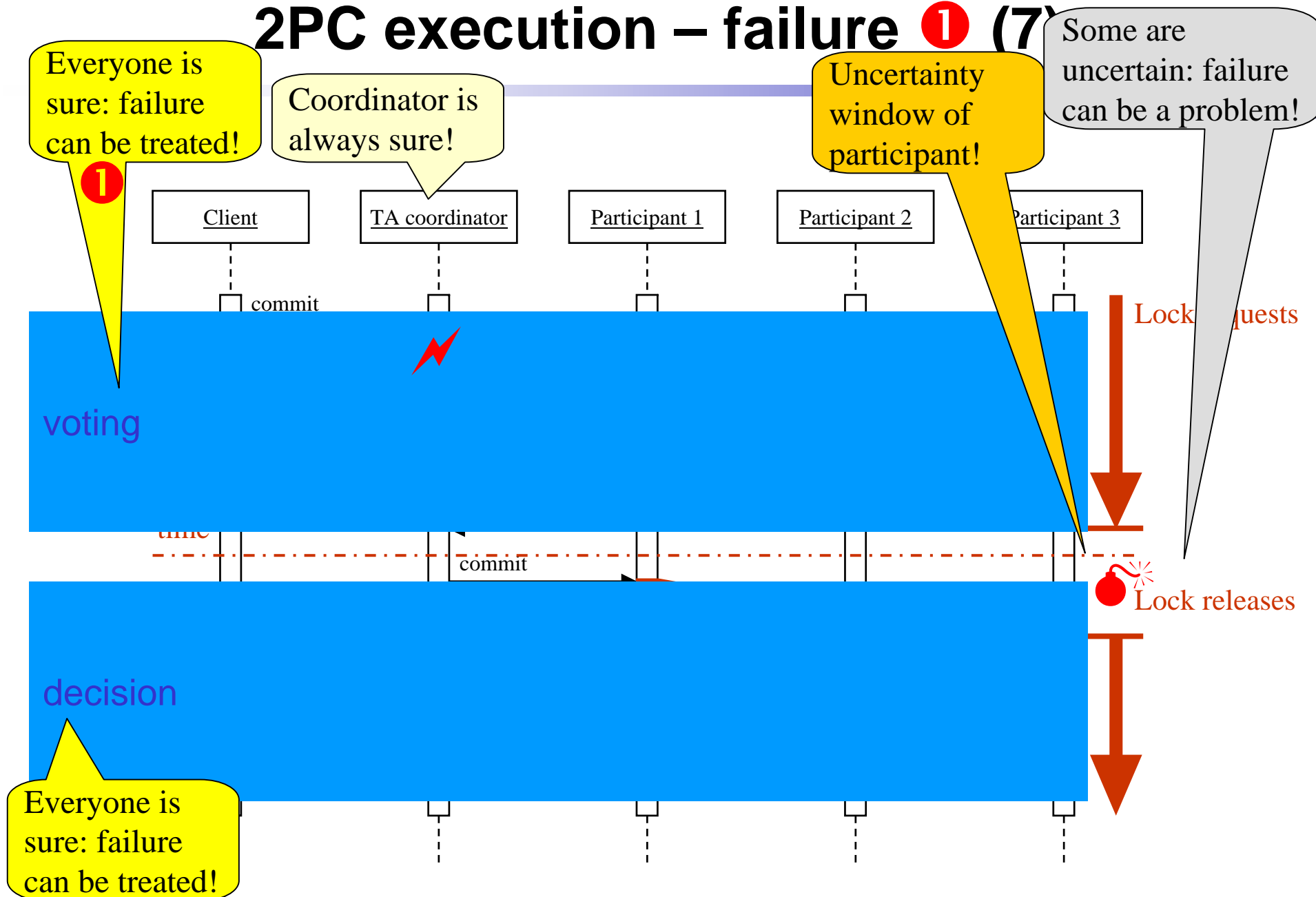
2PC execution – failure 1 (6)



Once H is up again: H decides on abort (*undo*)



2PC execution – failure 1 (7)



2PC execution – failure 1 (8)

down ● R (coordinator)

●
H

●
F

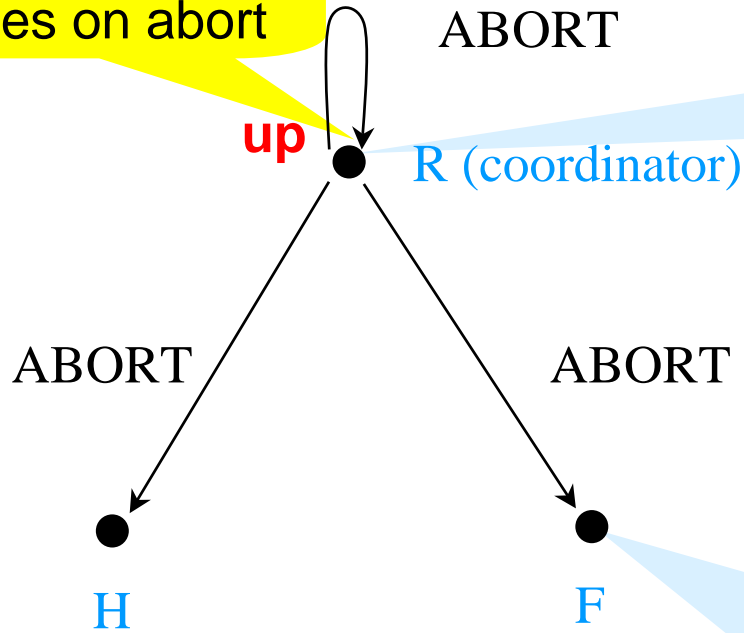
Waiting for further operations

After timeout:
Unilateral Abort

Stable Database		Stable Log		
<i>Obj</i>	<i>Val</i>	<i>Obj</i>	<i>Redo</i>	<i>Undo</i>
H-NYC	oldval	H-NYC	newval	oldval
W-NYC	oldval	abort		
...	...			

2PC execution – failure 1 (9)

Once R is up again:
R decides on abort

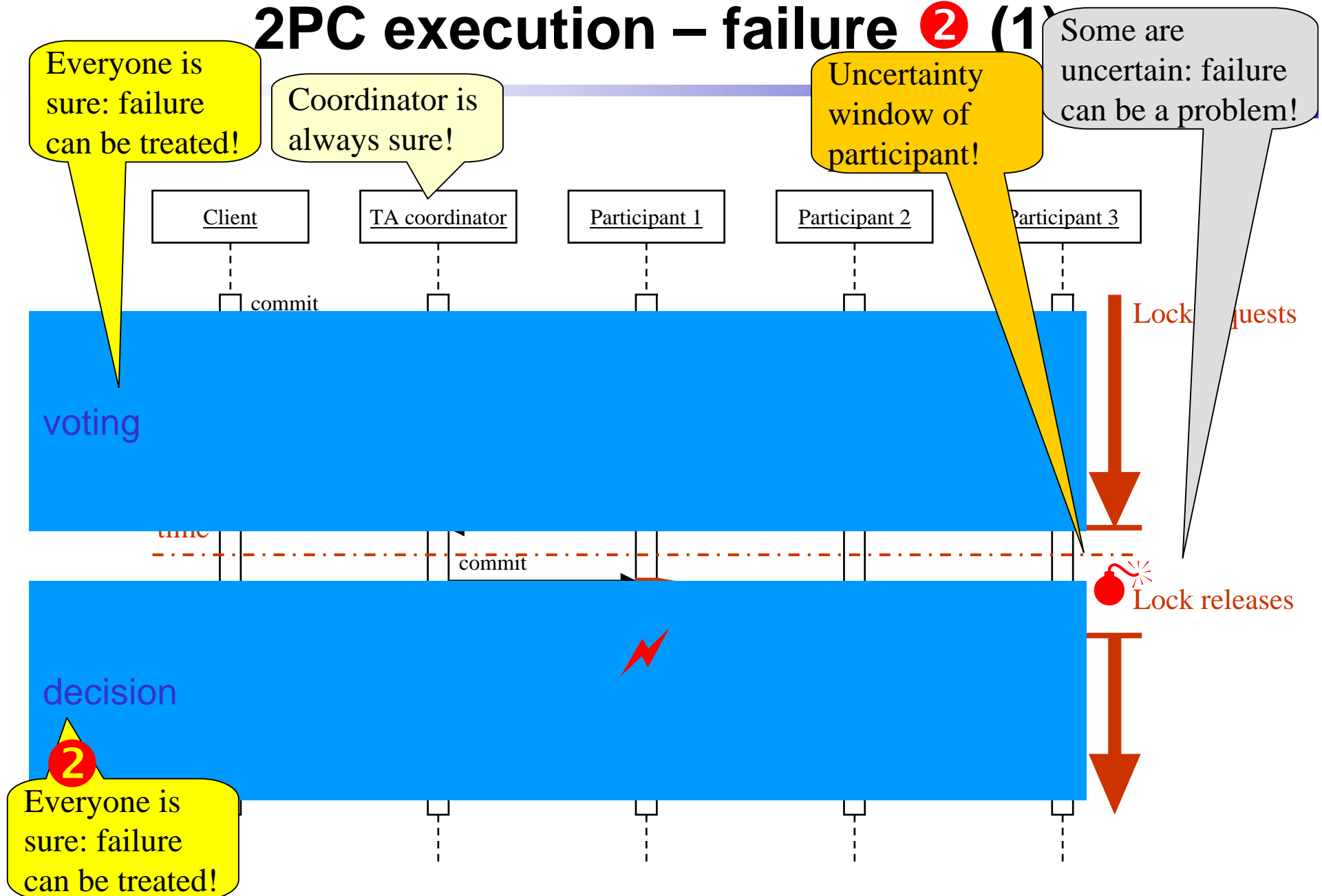


Global Log

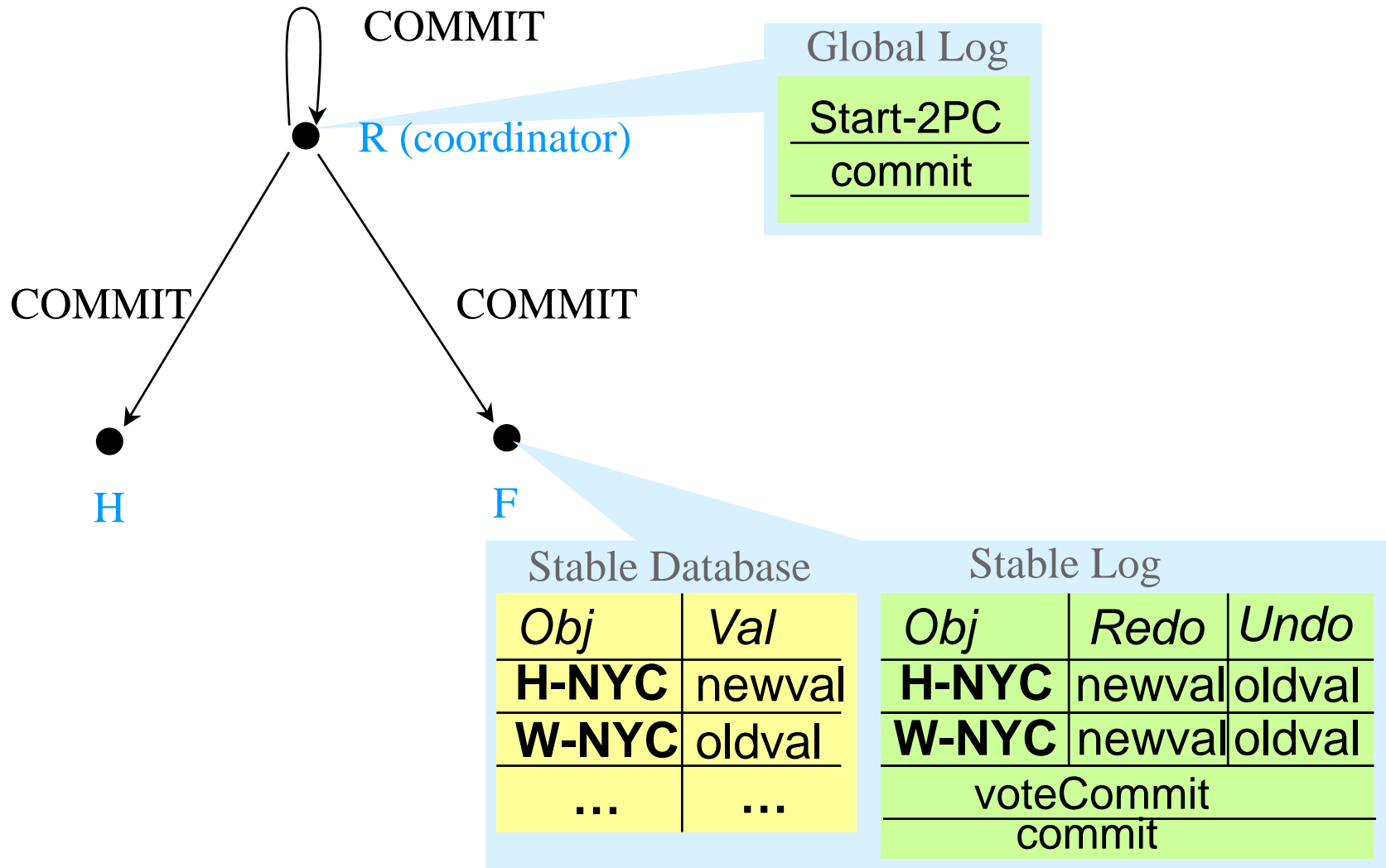
Start-2PC
abort

Stable Database		Stable Log		
<i>Obj</i>	<i>Val</i>	<i>Obj</i>	<i>Redo</i>	<i>Undo</i>
H-NYC	oldval	H-NYC	newval	oldval
W-NYC	oldval	abort		
...	...			

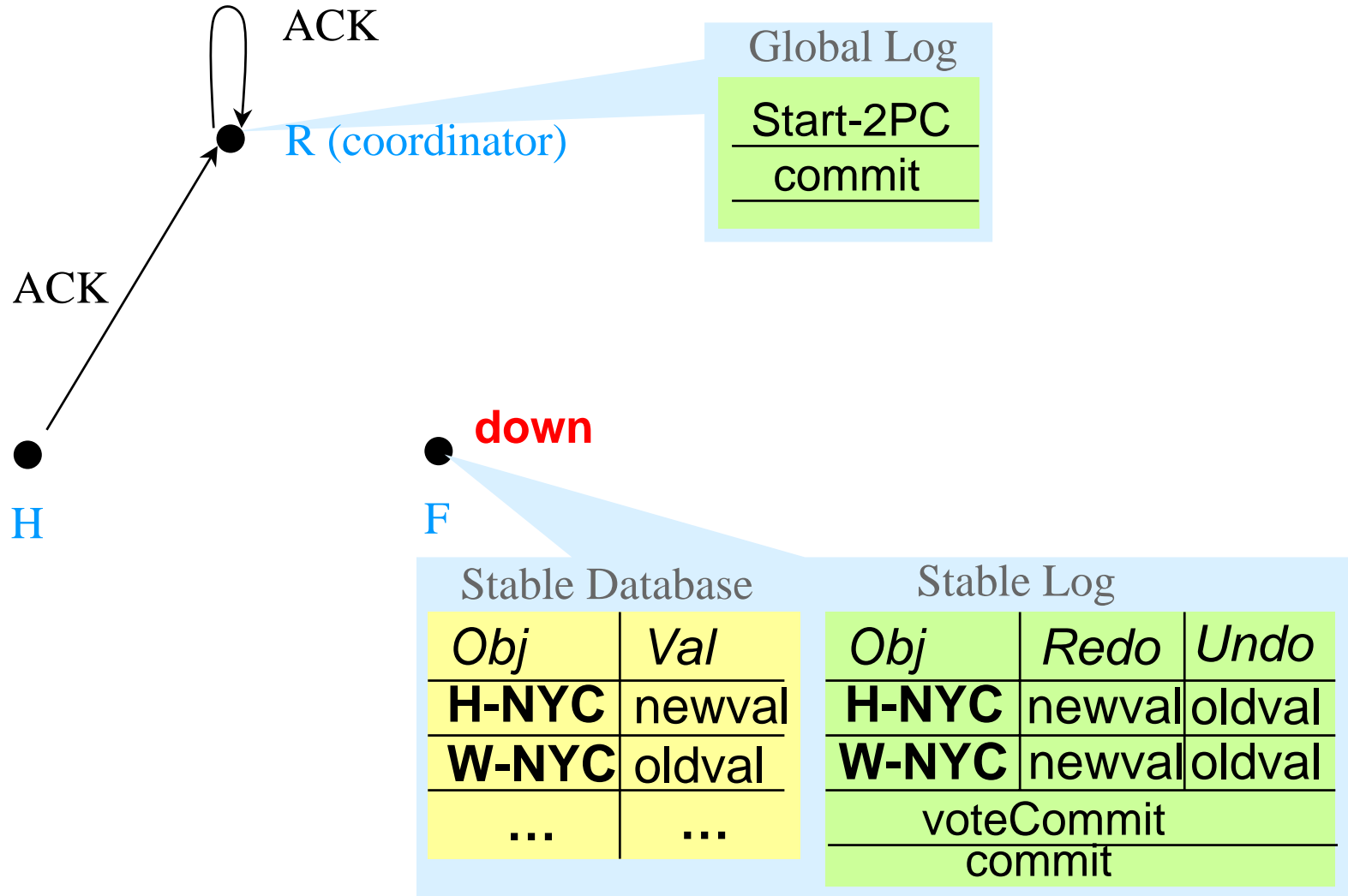
2PC execution – failure 2 (1)



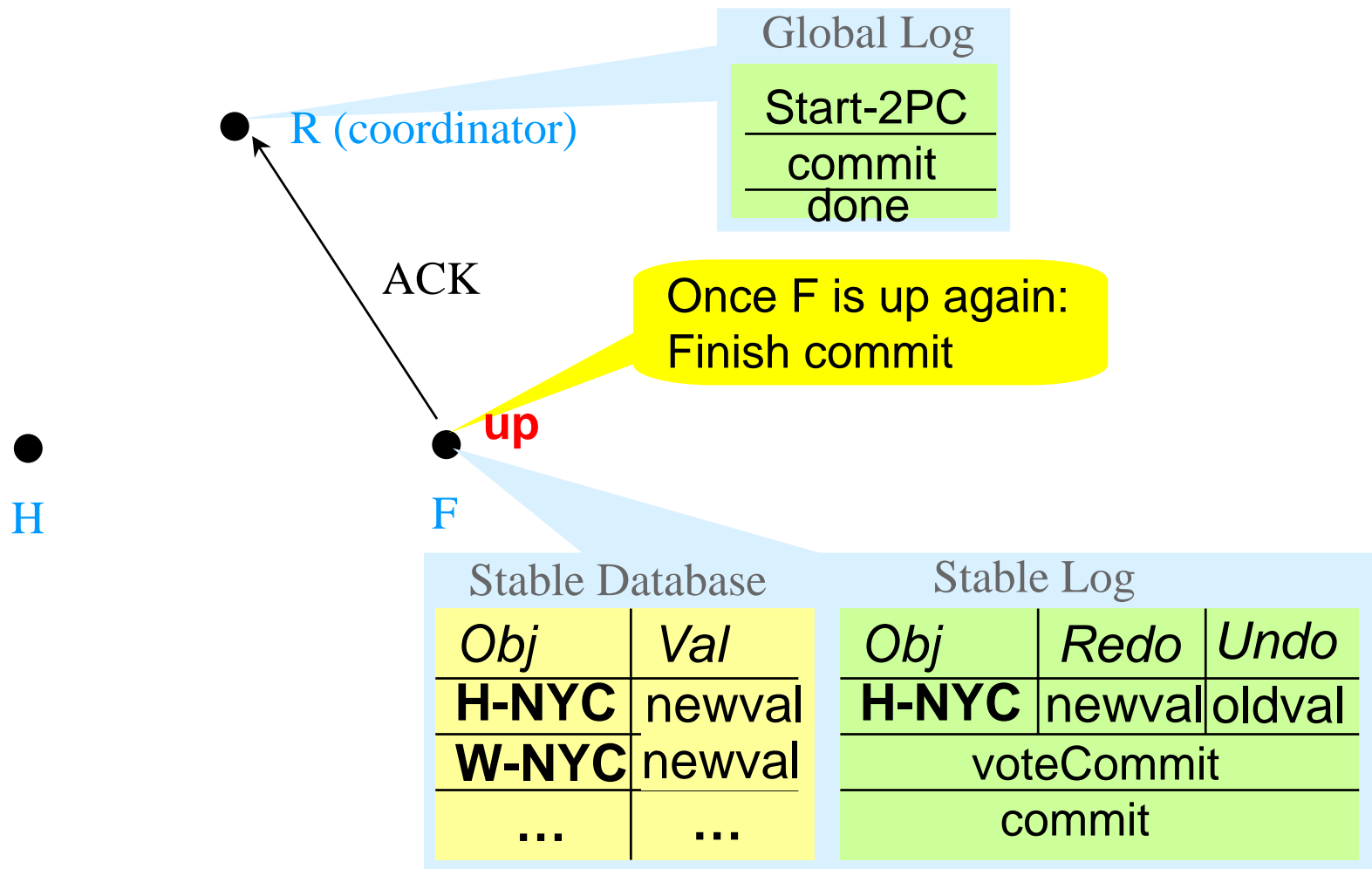
2PC execution – failure ② (2)



2PC execution – failure ② (3)



2PC execution – failure 2 (4)



2PC execution – site down

46

- While a site fails its TM cannot decide \Rightarrow site remains blocked.
- Sites waiting for responses from sites that are down:
 - ◆ To avoid waiting forever, set **timer**.
 - ◆ Coordinator: After timeout, if it waits for *voteCommit*, signal *abort* to all up-sites and later to the down-site. No timer set for *ack*, last *ack* causes *done* to be written to log.
 - ◆ Participant: After timeout, if waiting for *prepare*, decide on abort.

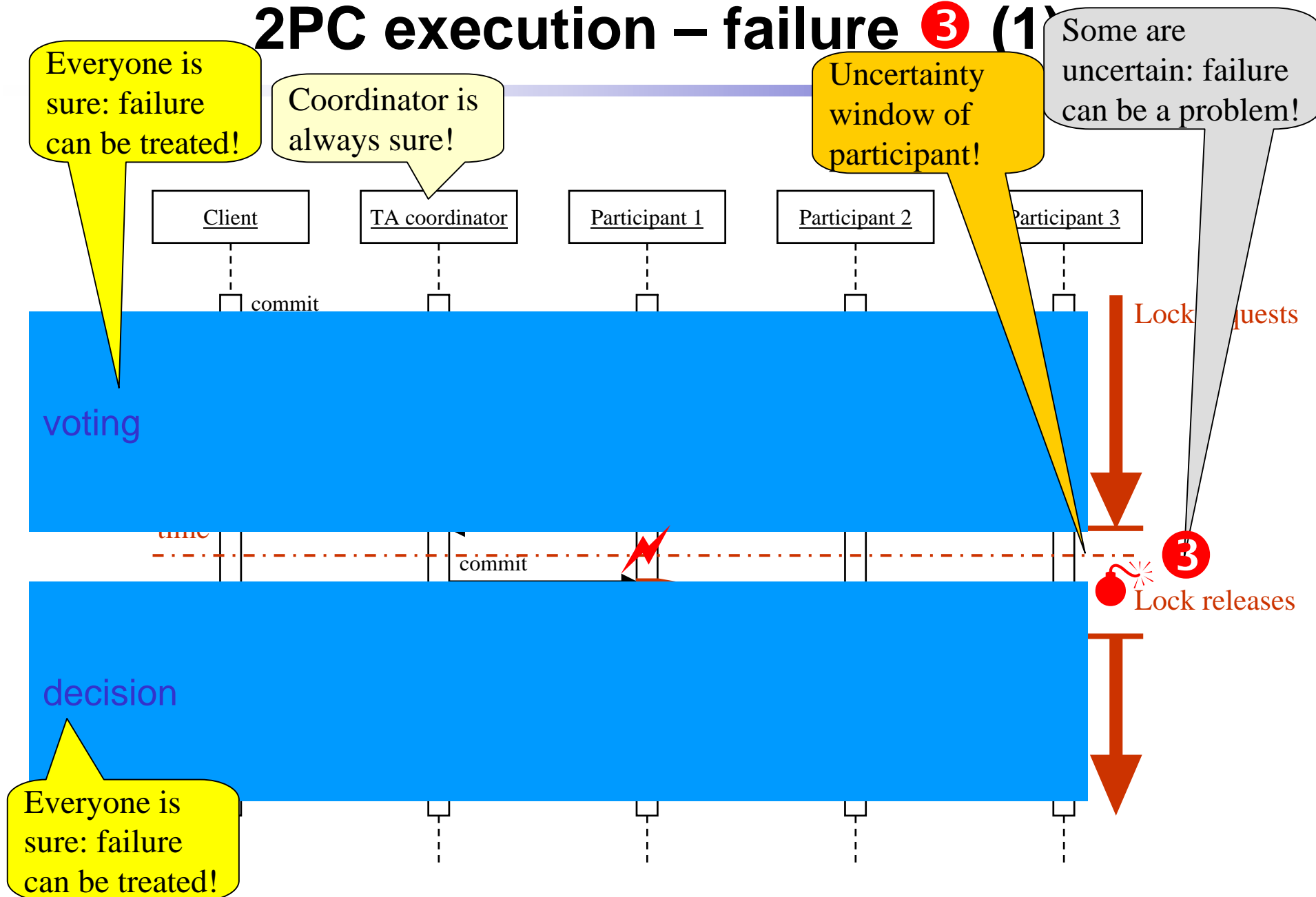
2PC execution – site recovery

Once site S has been reconnected or restarted:

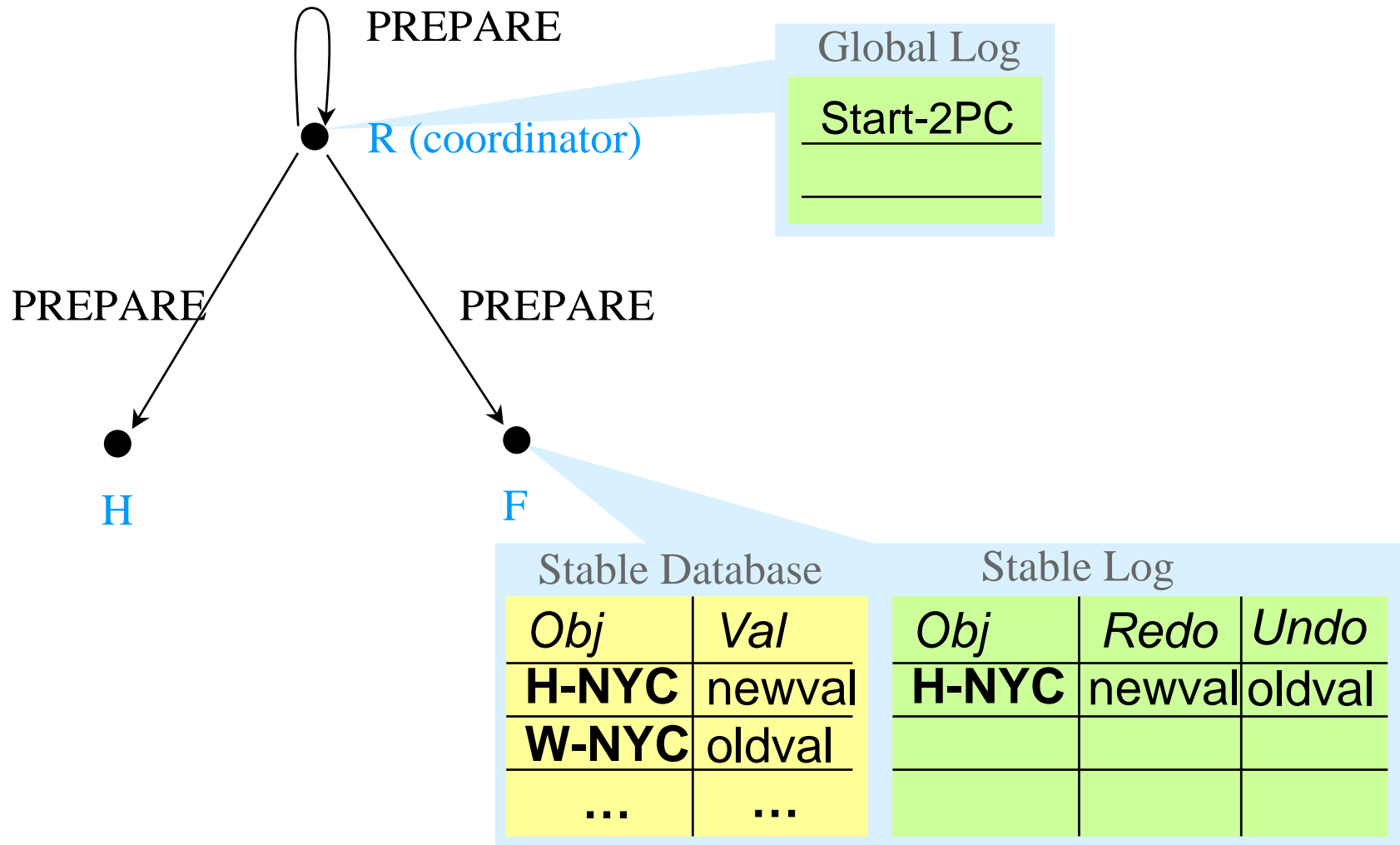
- DTL without **Start-2PC** for TA t . S was **participant in t** .
 - ◆ DTL contains **commit** or **abort**: Decision was taken. Repeat acknowledgement to coordinator or wait for reminder. ②
 - ◆ DTL contains **no voteCommit**: Decide on abort and inform coordinator of the decision. ①
- DTL with **Start-2PC** for TA t . S was **coordinator of t** .
 - ◆ DTL contains **only Start-2PC**: No decision was taken yet. Coordinator decides on **abort** and sends the decision. ①
 - ◆ DTL contains **Start-2PC and commit** or **abort but not done**: Decision was taken but not necessarily sent to all: Retransmit the decision. ②
 - ◆ DTL contains **Start-2PC, commit** or **abort and done**: Decision was taken and sent to all: Nothing left to do. ②

2PC Termination problems

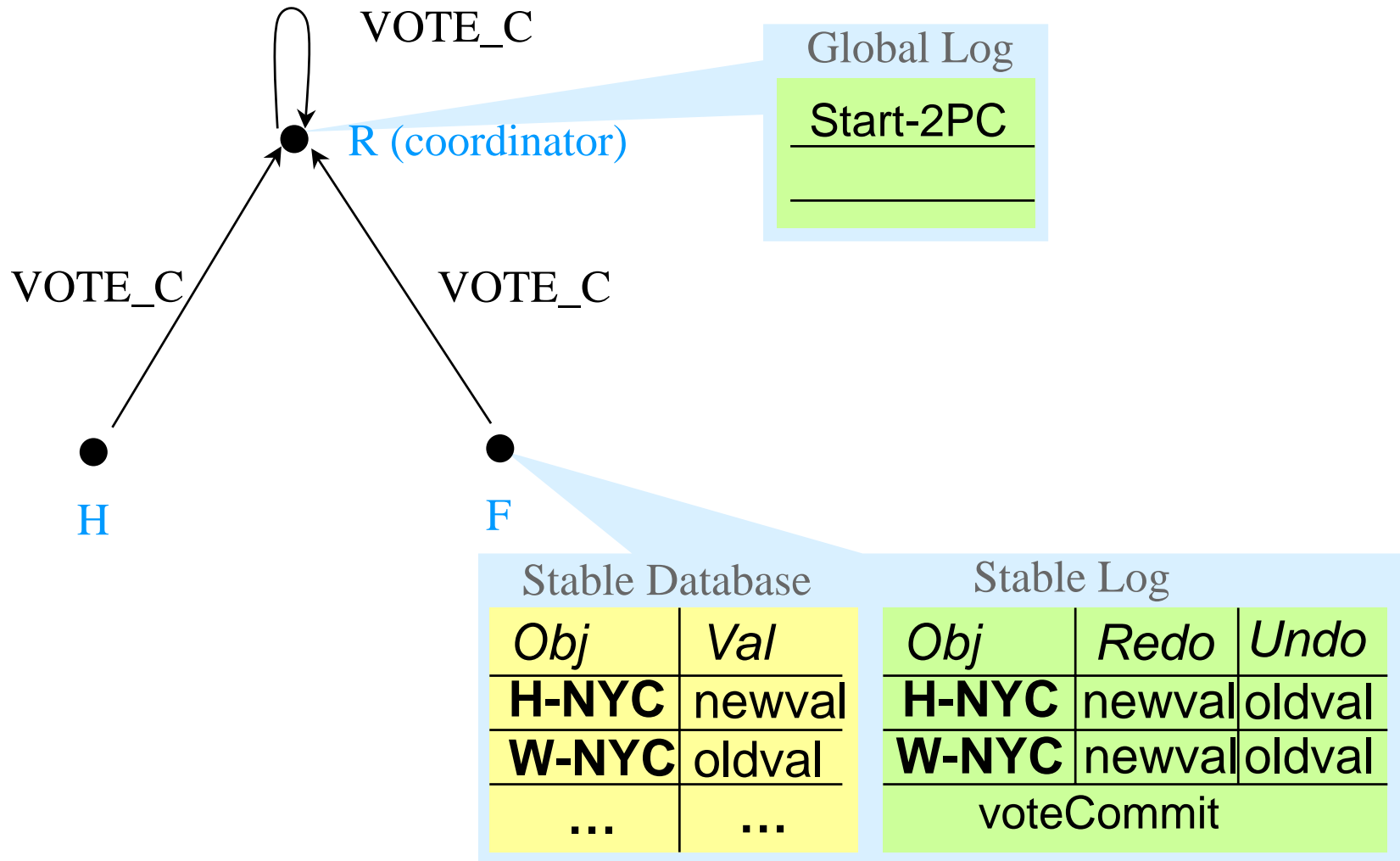
2PC execution – failure 3 (1)



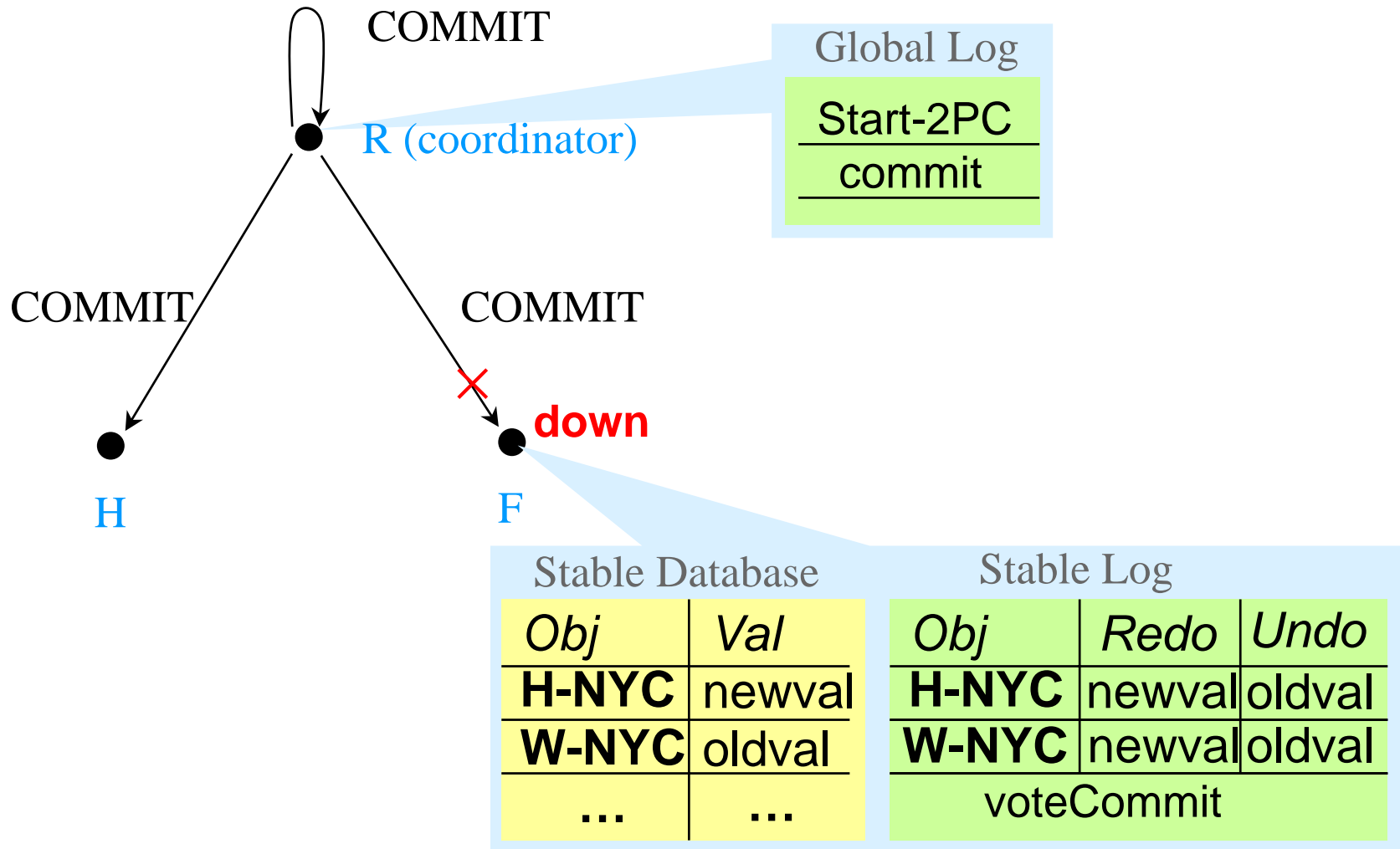
2PC execution – failure 3 (2)



2PC execution – failure ③ (3)

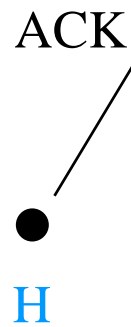


2PC execution – failure ③ (4)



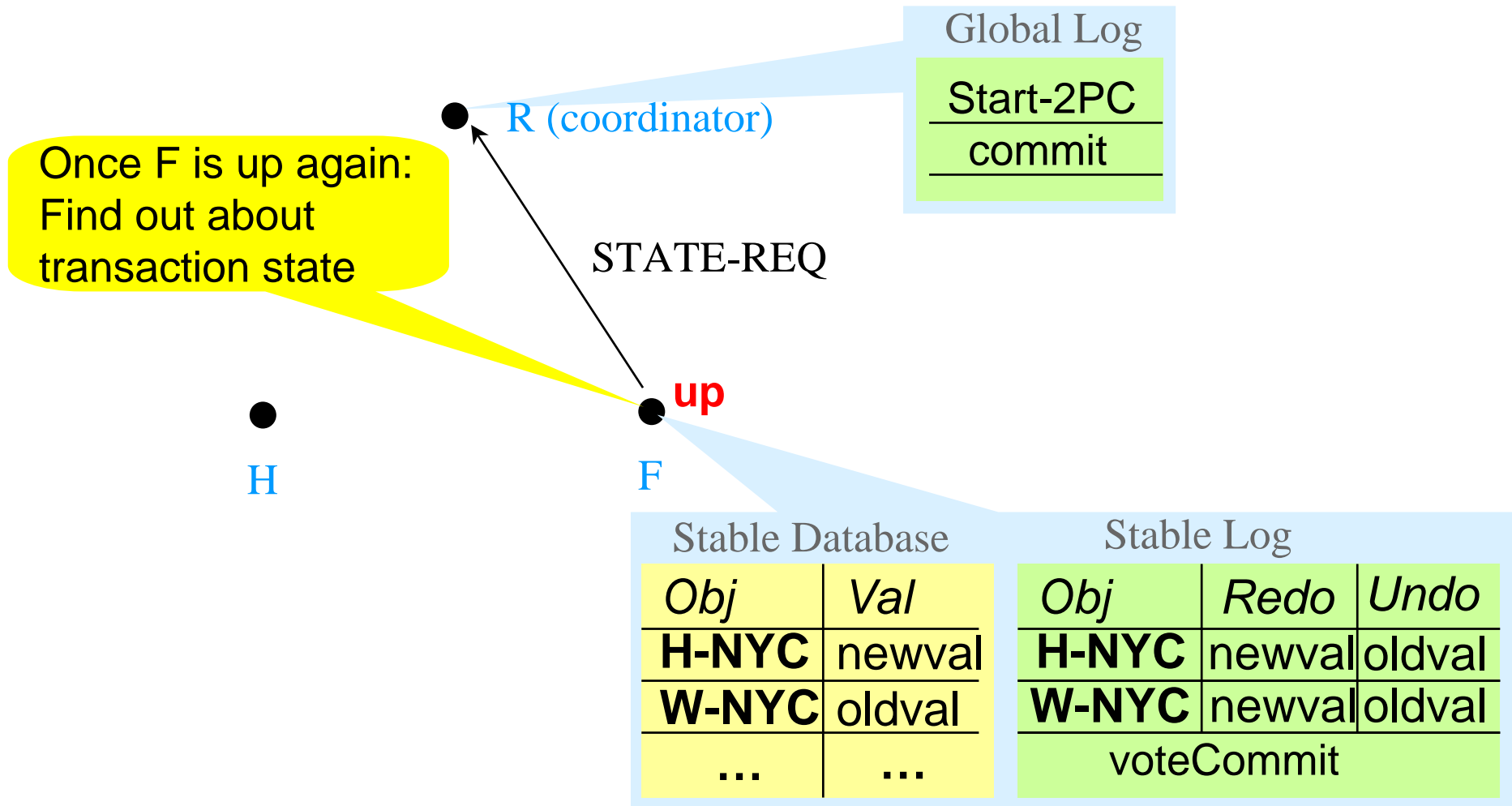
2PC execution – failure 3 (5)

R is already committed:
Can only wait

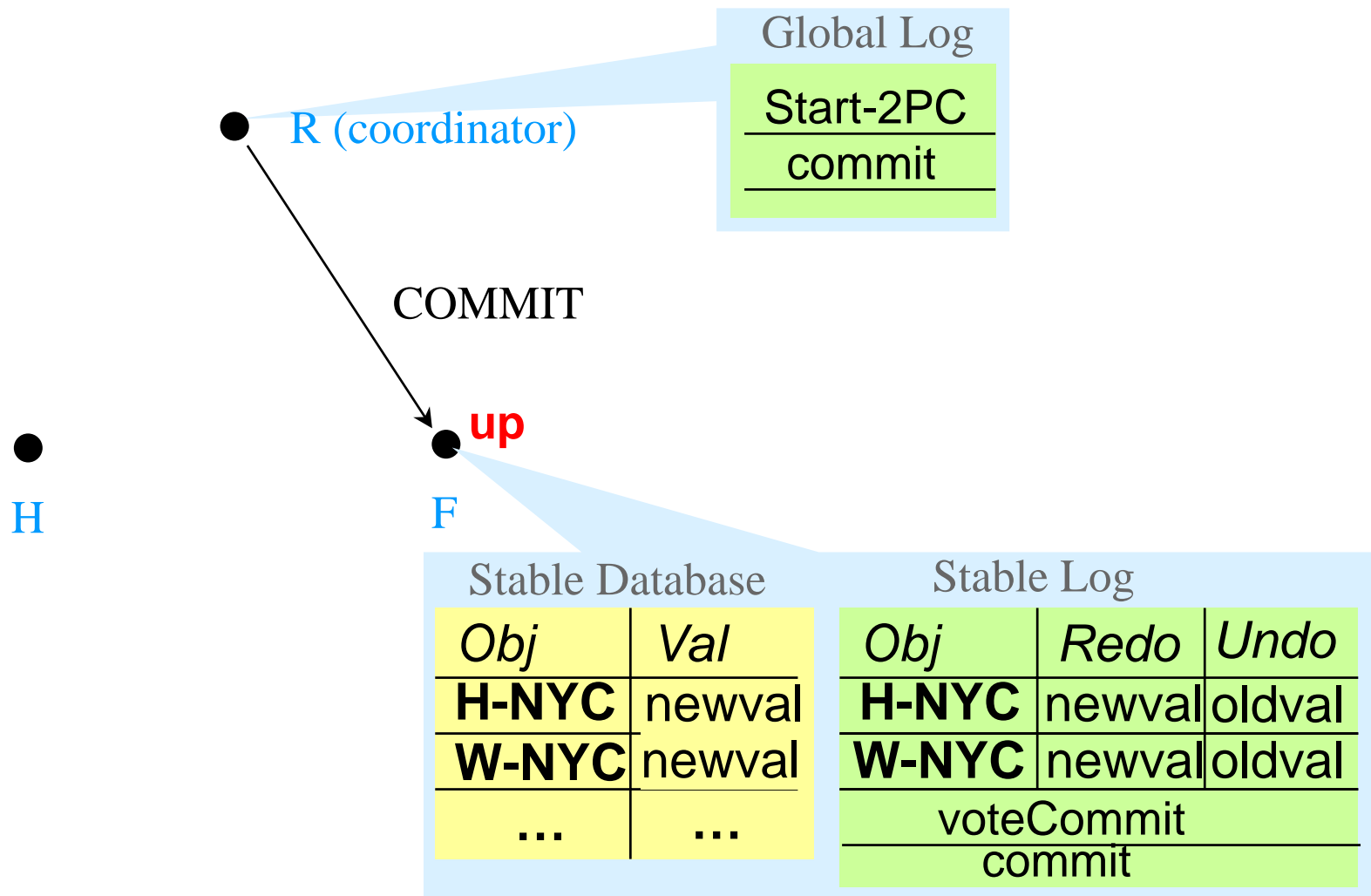


Stable Database		Stable Log		
<i>Obj</i>	<i>Val</i>	<i>Obj</i>	<i>Redo</i>	<i>Undo</i>
H-NYC	newval	H-NYC	newval	oldval
W-NYC	oldval	W-NYC	newval	oldval
...	...	voteCommit		

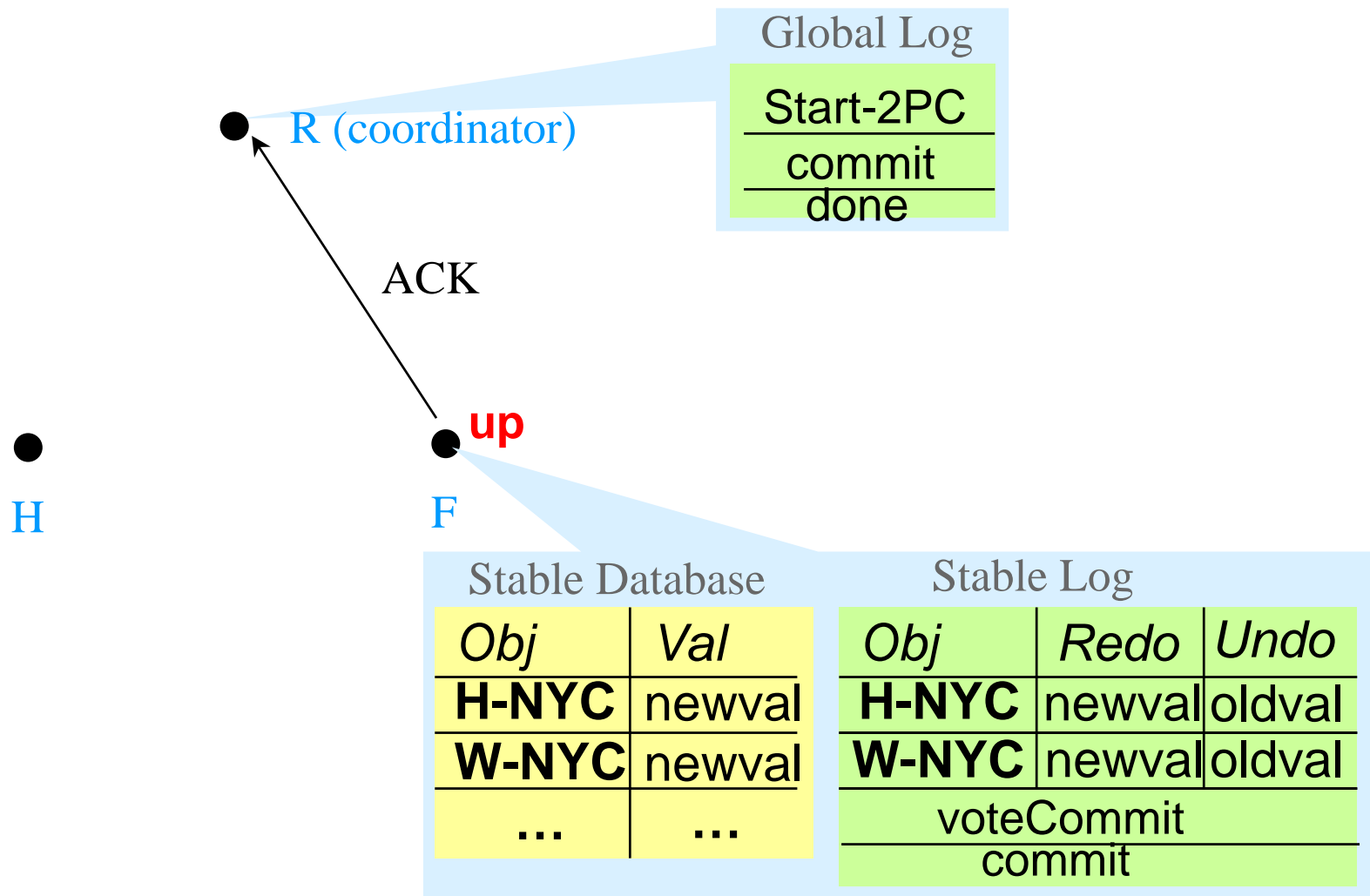
2PC execution – failure ③ (6)



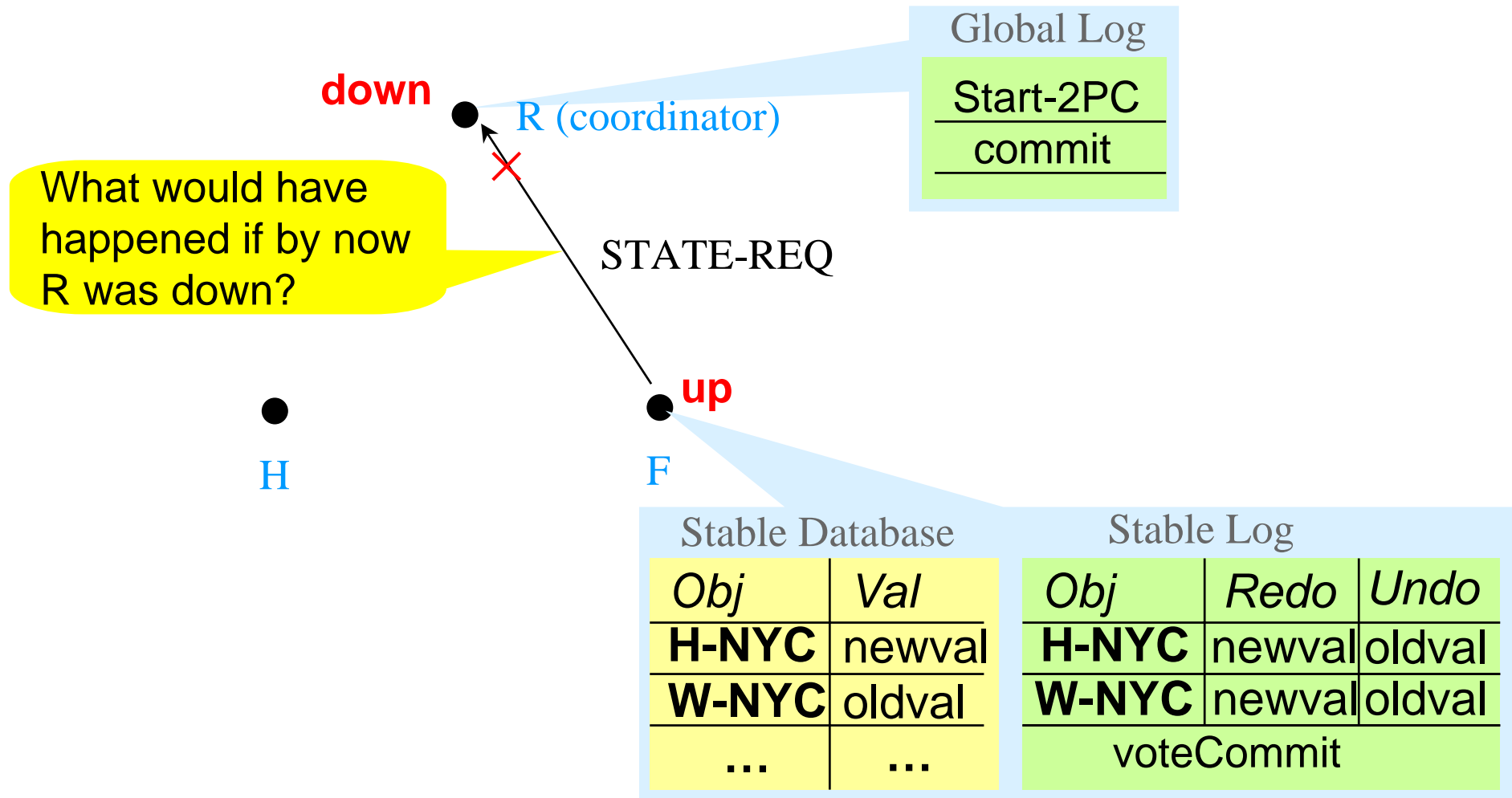
2PC execution – failure ③ (7)



2PC execution – failure ③ (8)



2PC termination problem ③



2PC execution – site recovery (cont'd)

58

Once site S has been reconnected or restarted:

- DTL without **Start-2PC** for TA t . S was participant in t .
 - ◆ DTL contains **commit** or **abort**: Decision was taken. Repeat acknowledgement to coordinator or wait for reminder. ②
 - ◆ DTL contains **no voteCommit**: Decide on abort and inform coordinator of the decision. ①
 - ◆ DTL contains **voteCommit** but **no commit** or **abort**: Participant is uncertain. ③

Ask others what the decision was:

- Try coordinator: It is always sure.
- Otherwise try to reach participants that are sure. If none or only uncertain nodes can be reached: **Protocol is blocked!**

Termination during uncertainty window (1)

59

S asks others what the decision was:

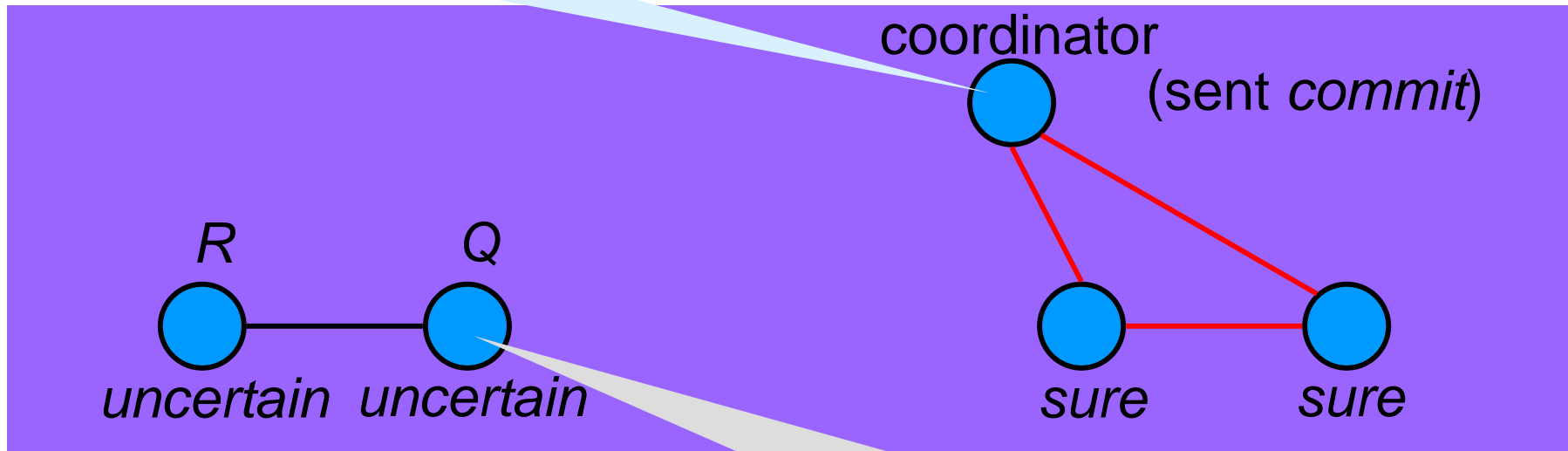
- Try coordinator: It is always sure.
- Otherwise try to reach participants that are sure. If none or only uncertain nodes can be reached: **Protocol is blocked!**

- Other site Q decided on **commit/abort** due to coordinator message: pass decision on to S.
- Q voted **abort**: pass decision on to S, S infers the global decision.
- Q did not yet vote: S decides on unilateral abort.

- Q voted **voteCommit** but received no answer and, hence, is uncertain.

Termination during uncertainty window (2)

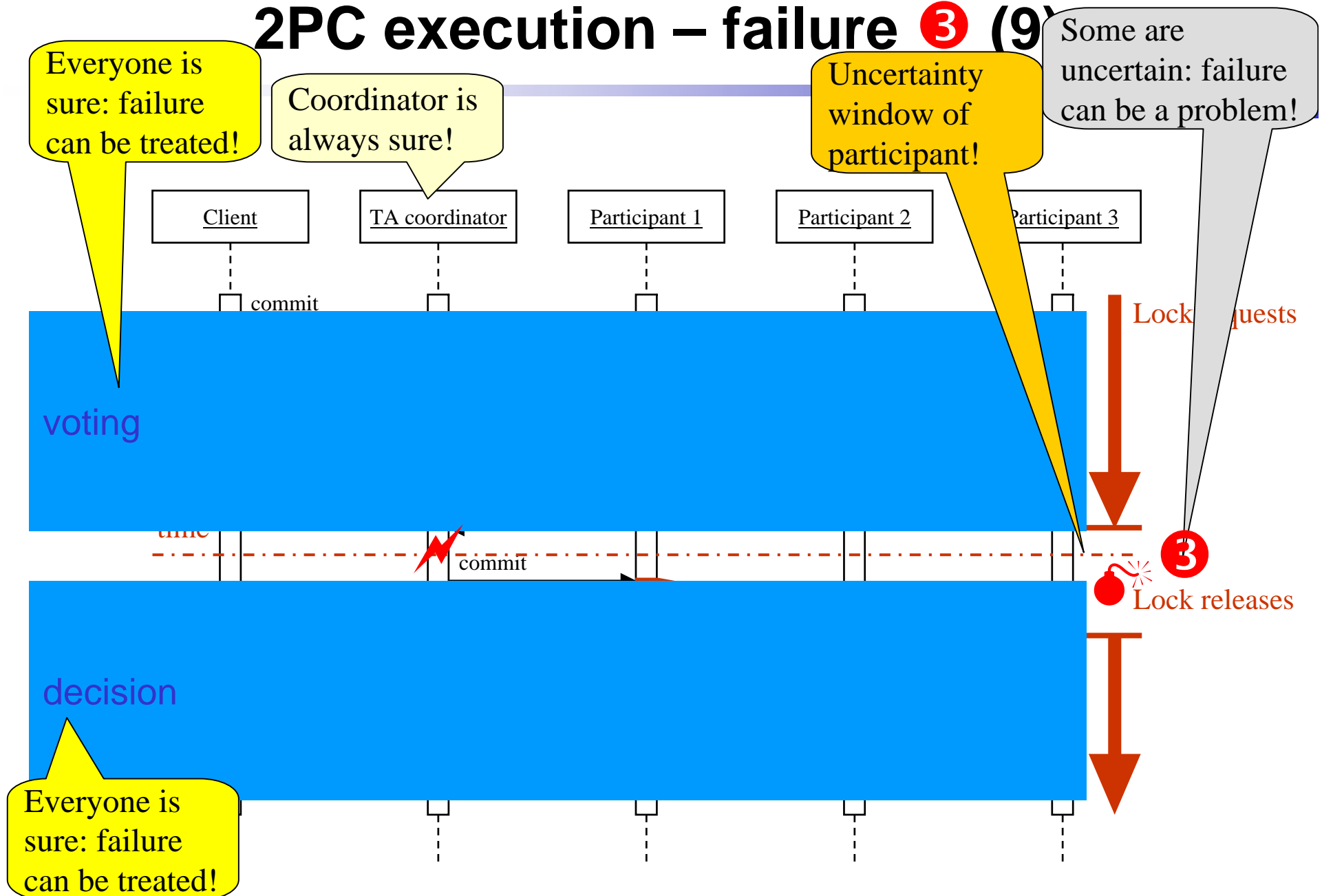
t remains blocked until all *acks* have been received



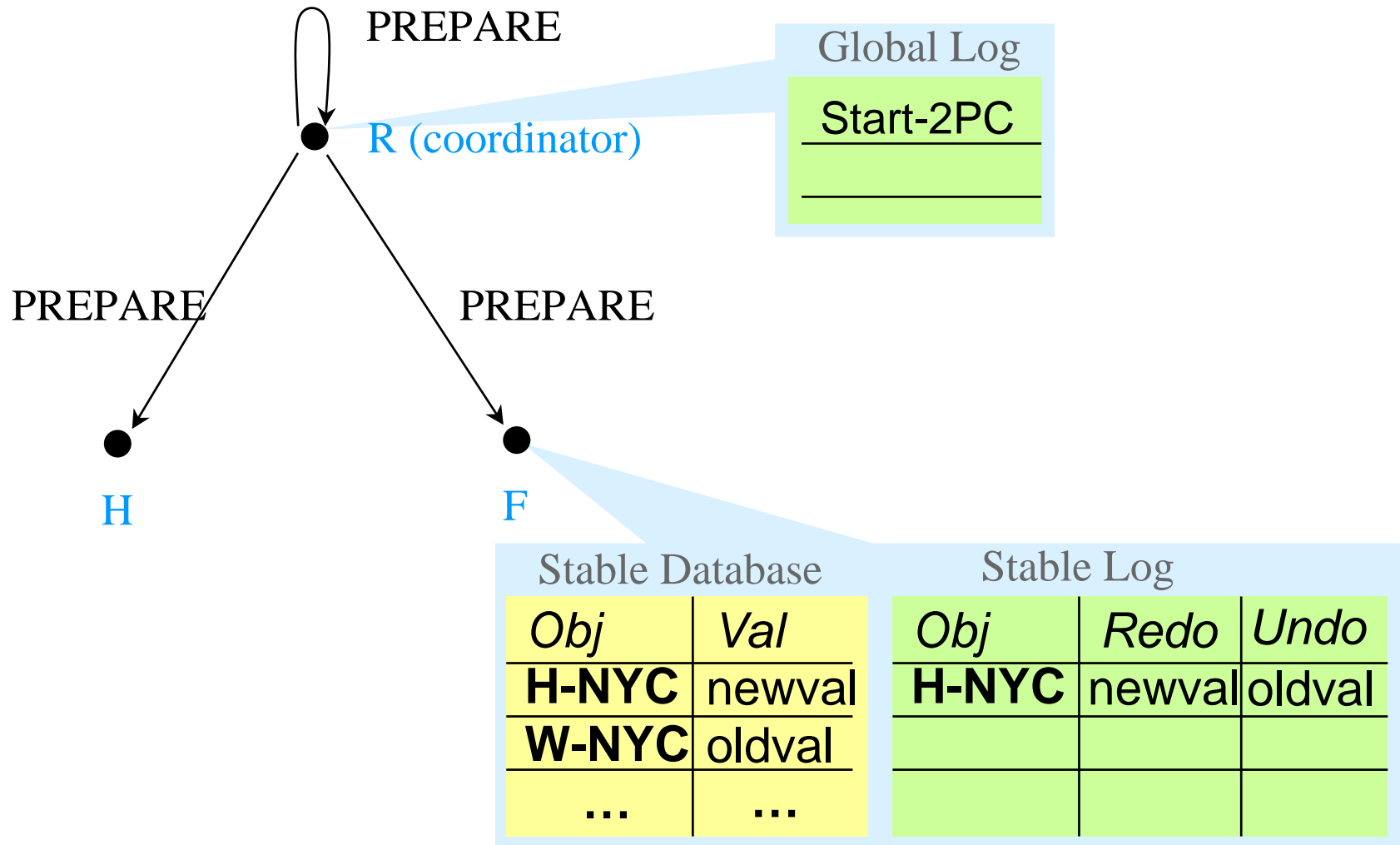
R and *Q* remain blocked until one of them has been reconnected to the other subnet

Blocking is of unknown duration !

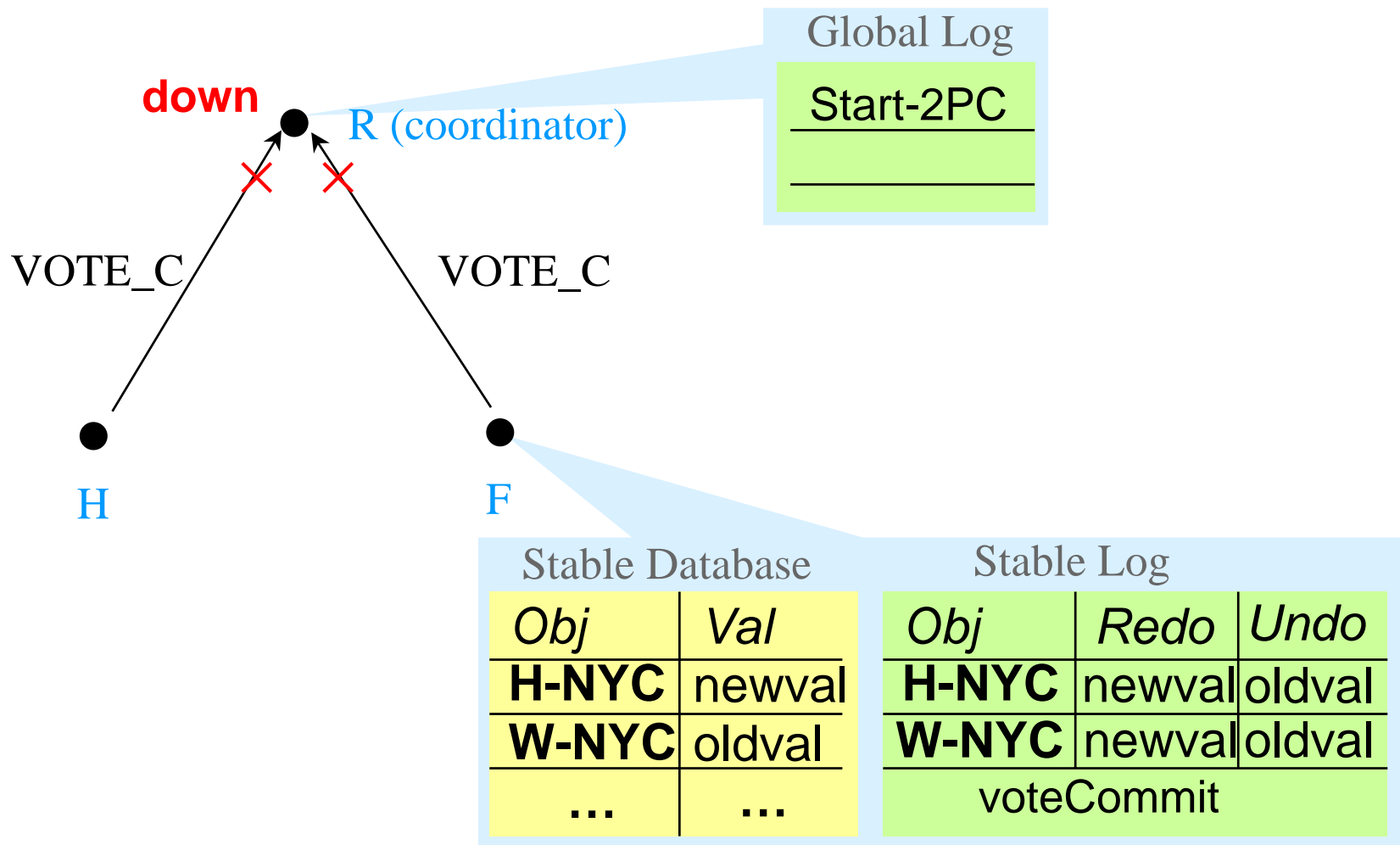
2PC execution – failure 3 (9)



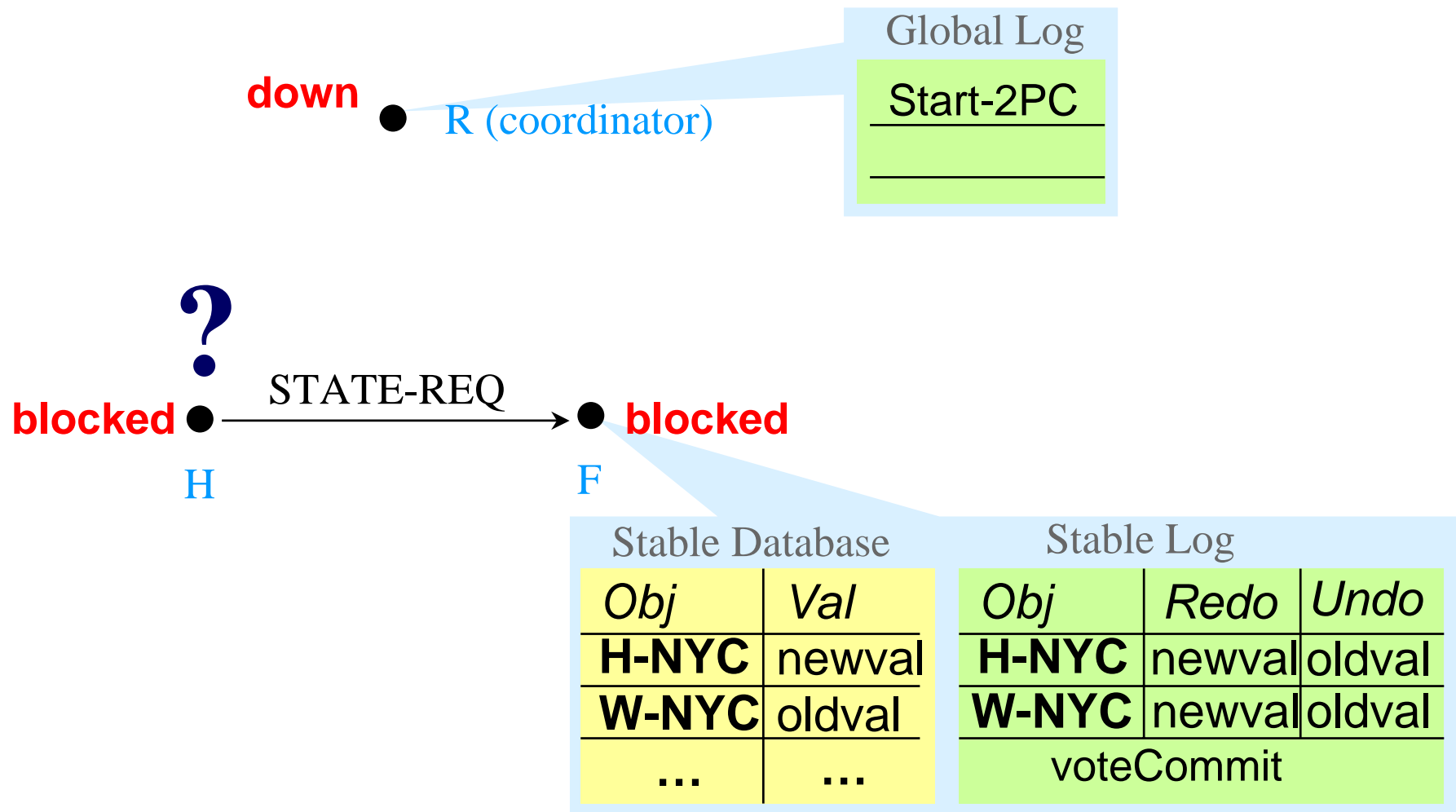
Termination during uncertainty window (3)



Termination during uncertainty window (4)

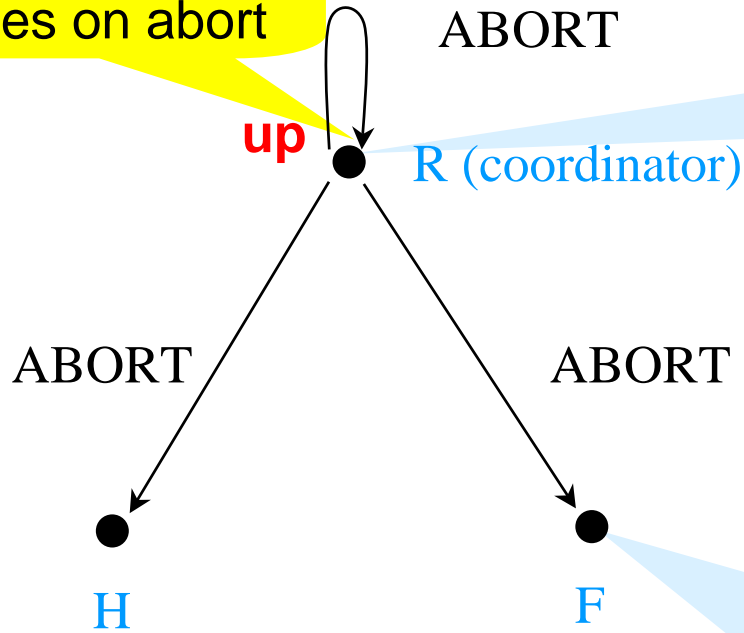


Termination during uncertainty window (5)



Termination during uncertainty window (6)

Once R is up again:
R decides on abort

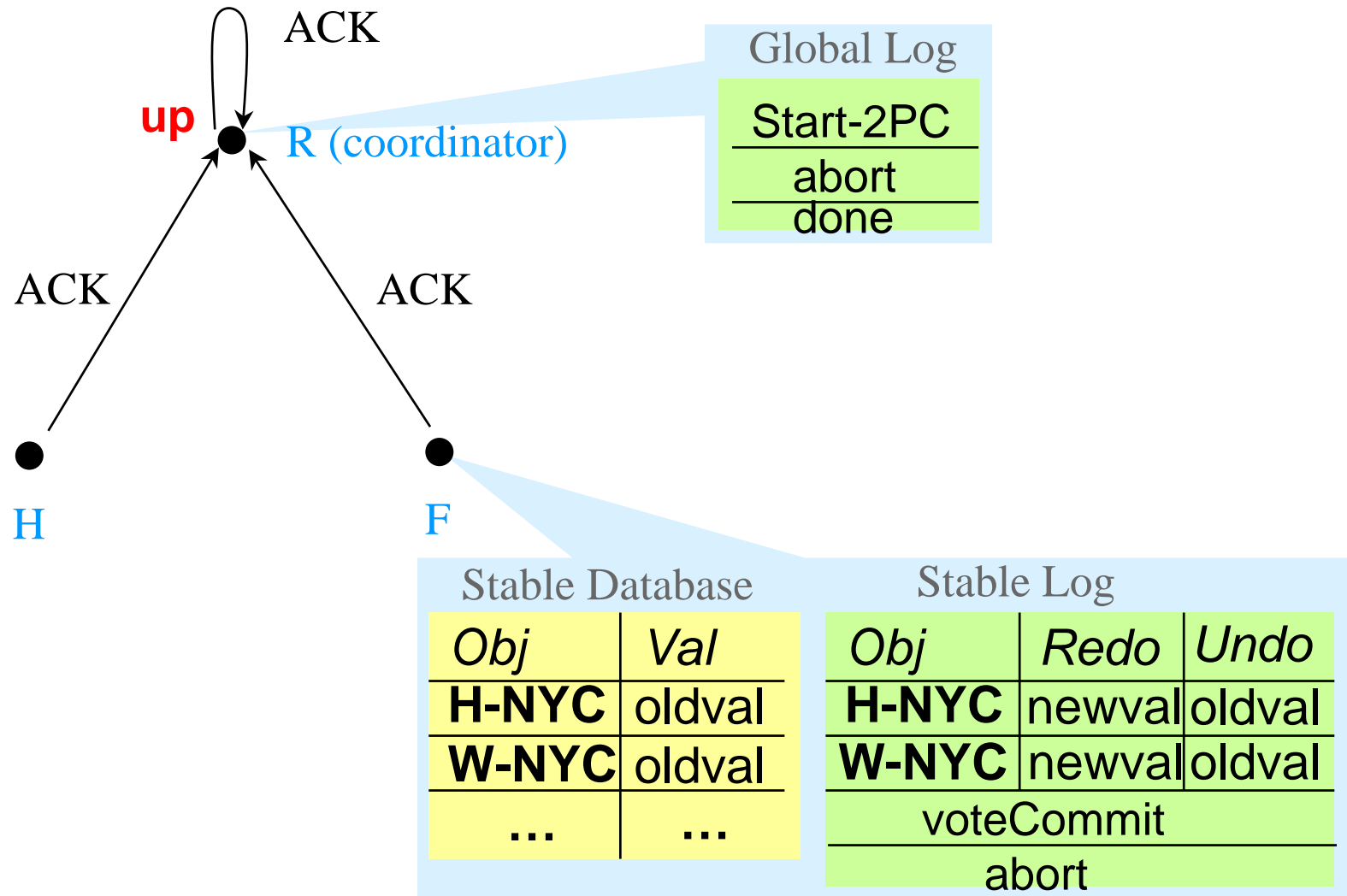


Global Log

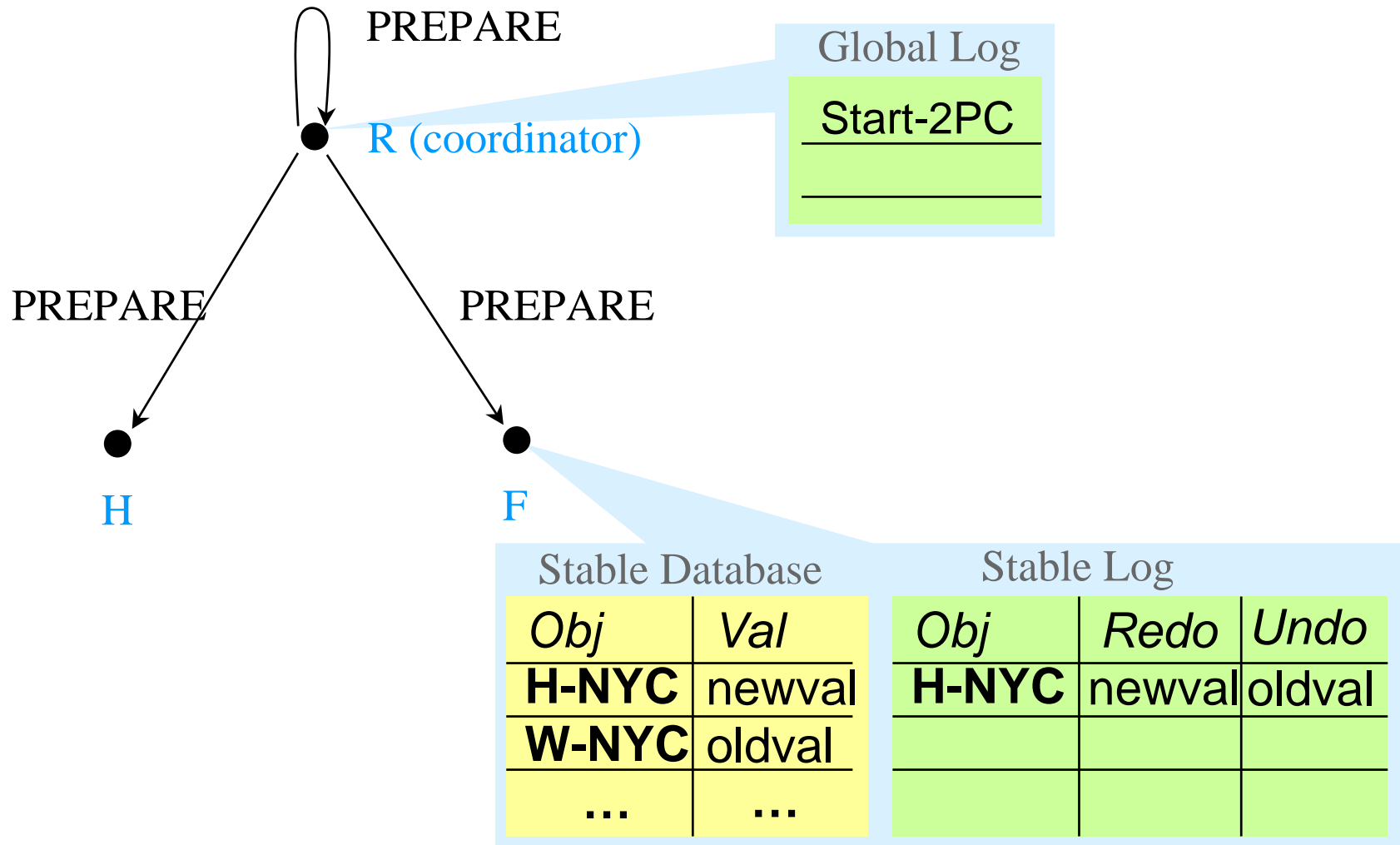
Start-2PC
abort

Stable Database		Stable Log		
<i>Obj</i>	<i>Val</i>	<i>Obj</i>	<i>Redo</i>	<i>Undo</i>
H-NYC	oldval	H-NYC	newval	oldval
W-NYC	oldval	W-NYC	newval	oldval
...	...	voteCommit		
abort				

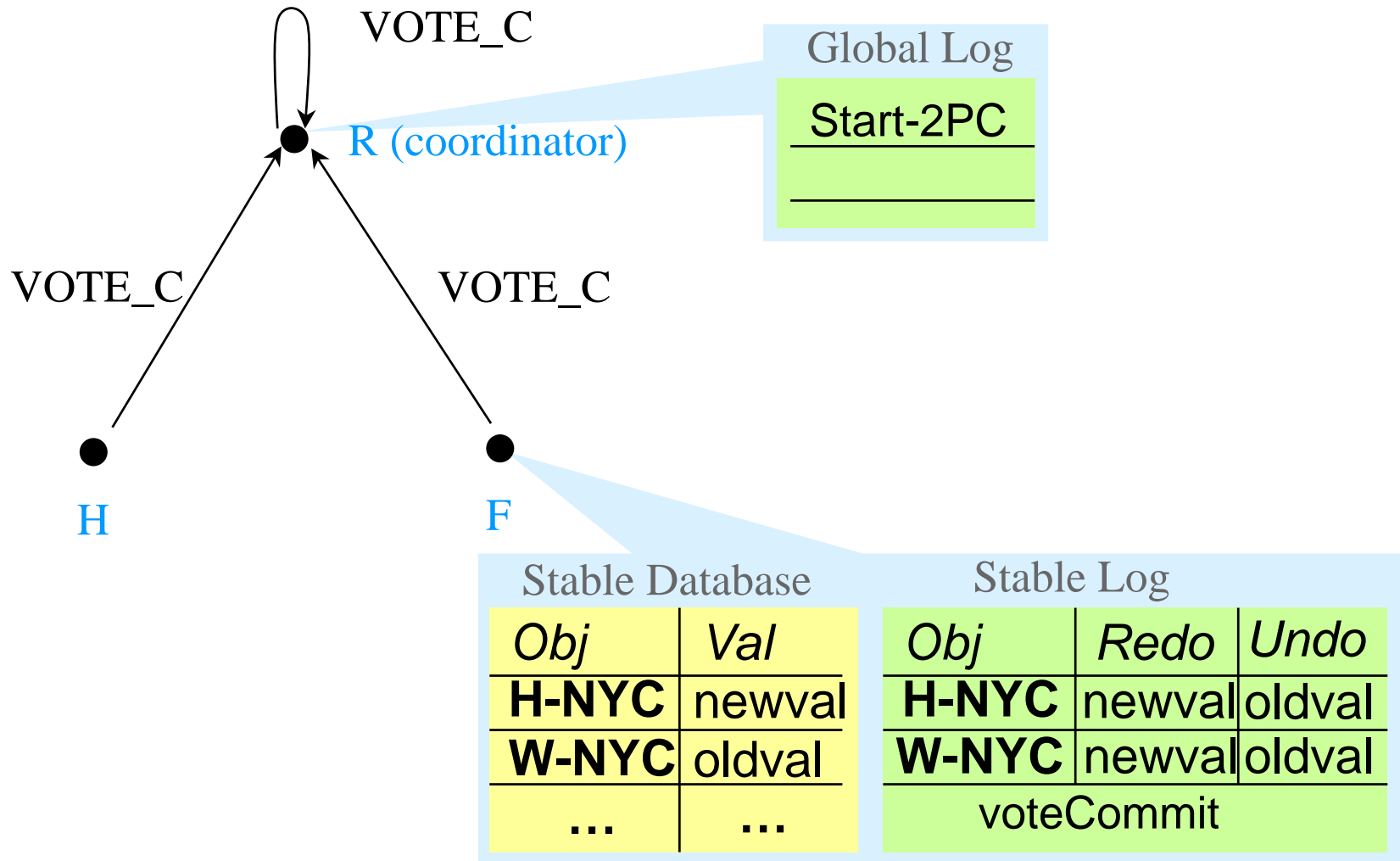
Termination during uncertainty window (7)



Termination during uncertainty window (8)



Termination during uncertainty window (9)



Termination during uncertainty window (10)

after decision but before sending messages

down ● R (coordinator)

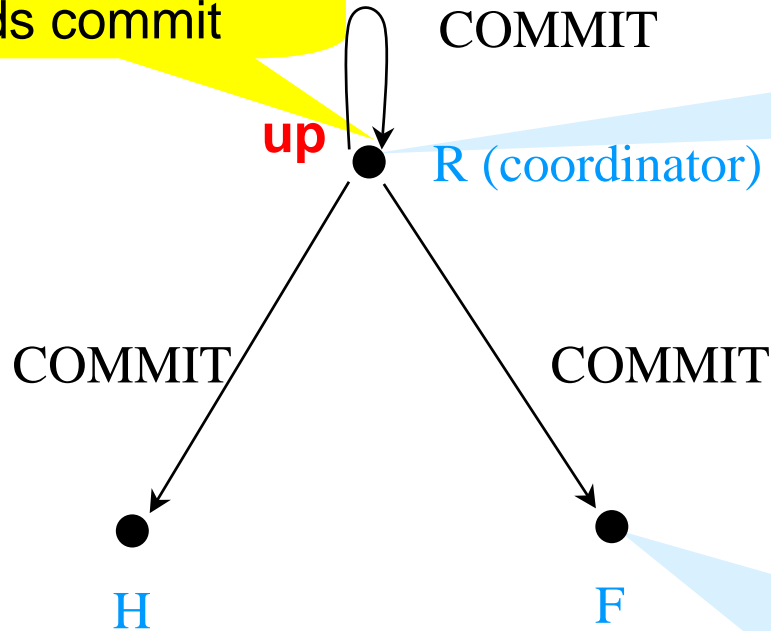
Global Log		
Start-2PC		
<hr/>		
commit		



Stable Database		Stable Log		
<i>Obj</i>	<i>Val</i>	<i>Obj</i>	<i>Redo</i>	<i>Undo</i>
H-NYC	newval	H-NYC	newval	oldval
W-NYC	oldval	W-NYC	newval	oldval
...	...	voteCommit		

Termination during uncertainty window (11)

Once R is up again:
R sends commit

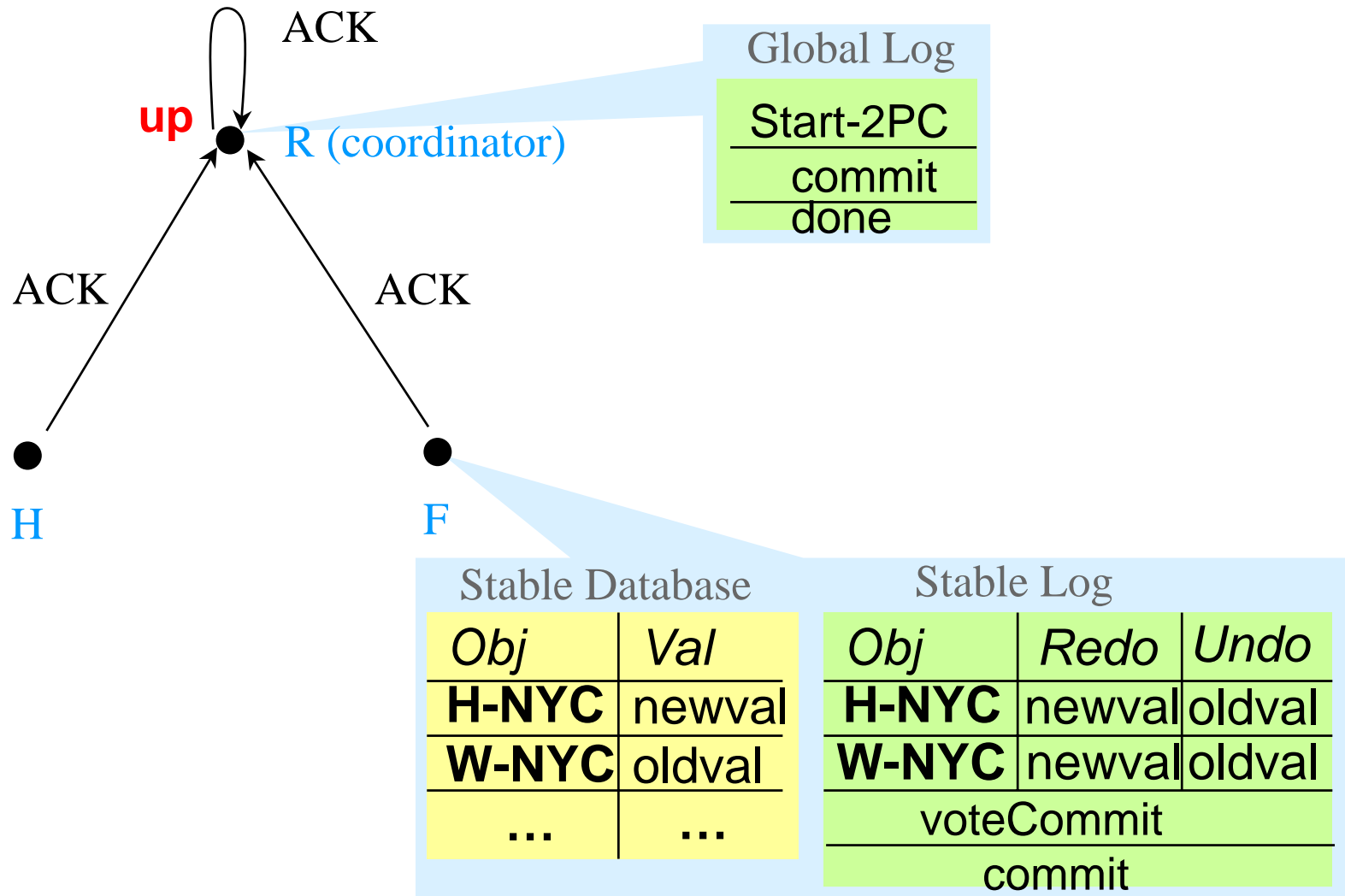


Global Log

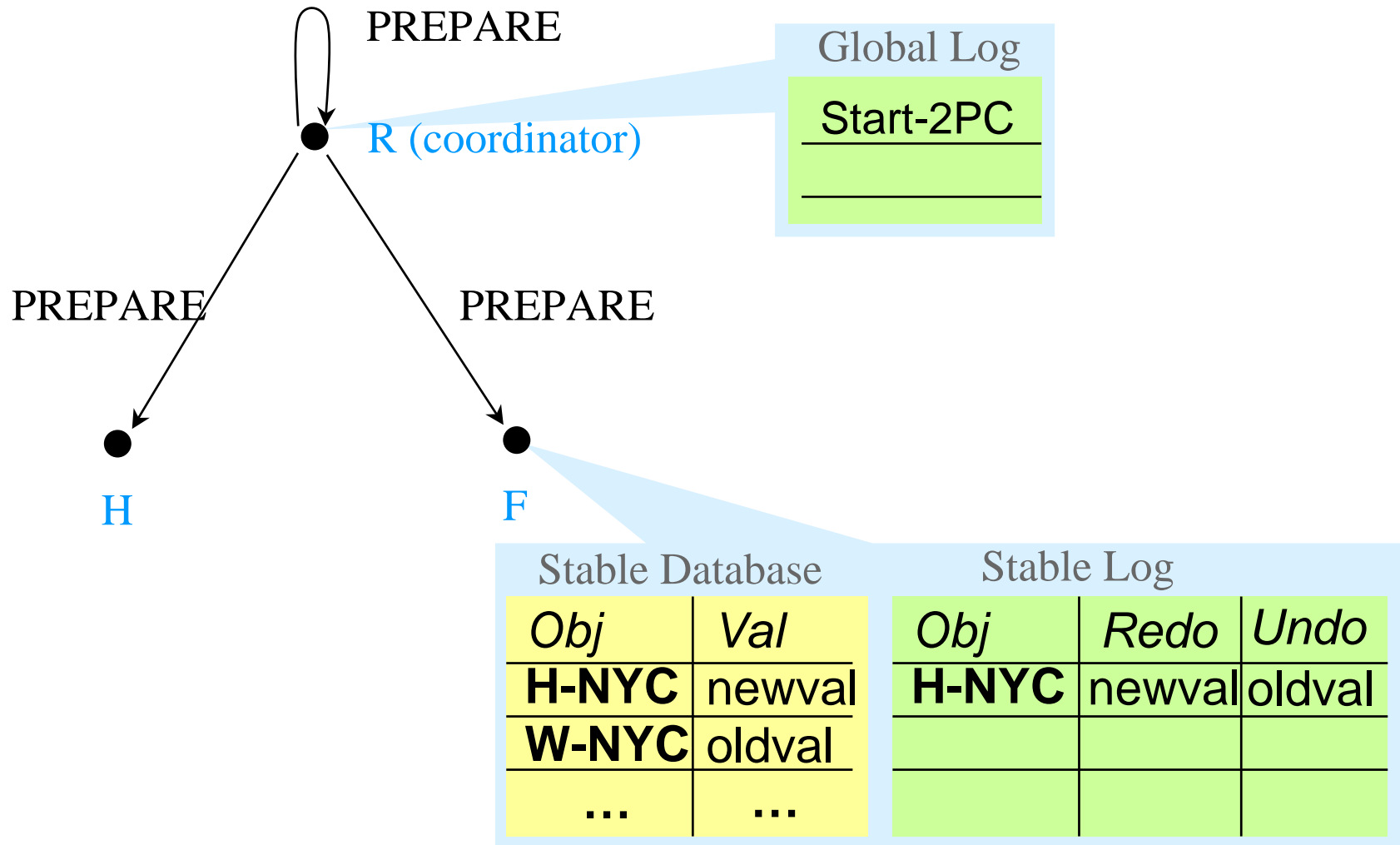
Start-2PC
commit

Stable Database		Stable Log		
<i>Obj</i>	<i>Val</i>	<i>Obj</i>	<i>Redo</i>	<i>Undo</i>
H-NYC	newval	H-NYC	newval	oldval
W-NYC	oldval	W-NYC	newval	oldval
...	...	voteCommit		
		commit		

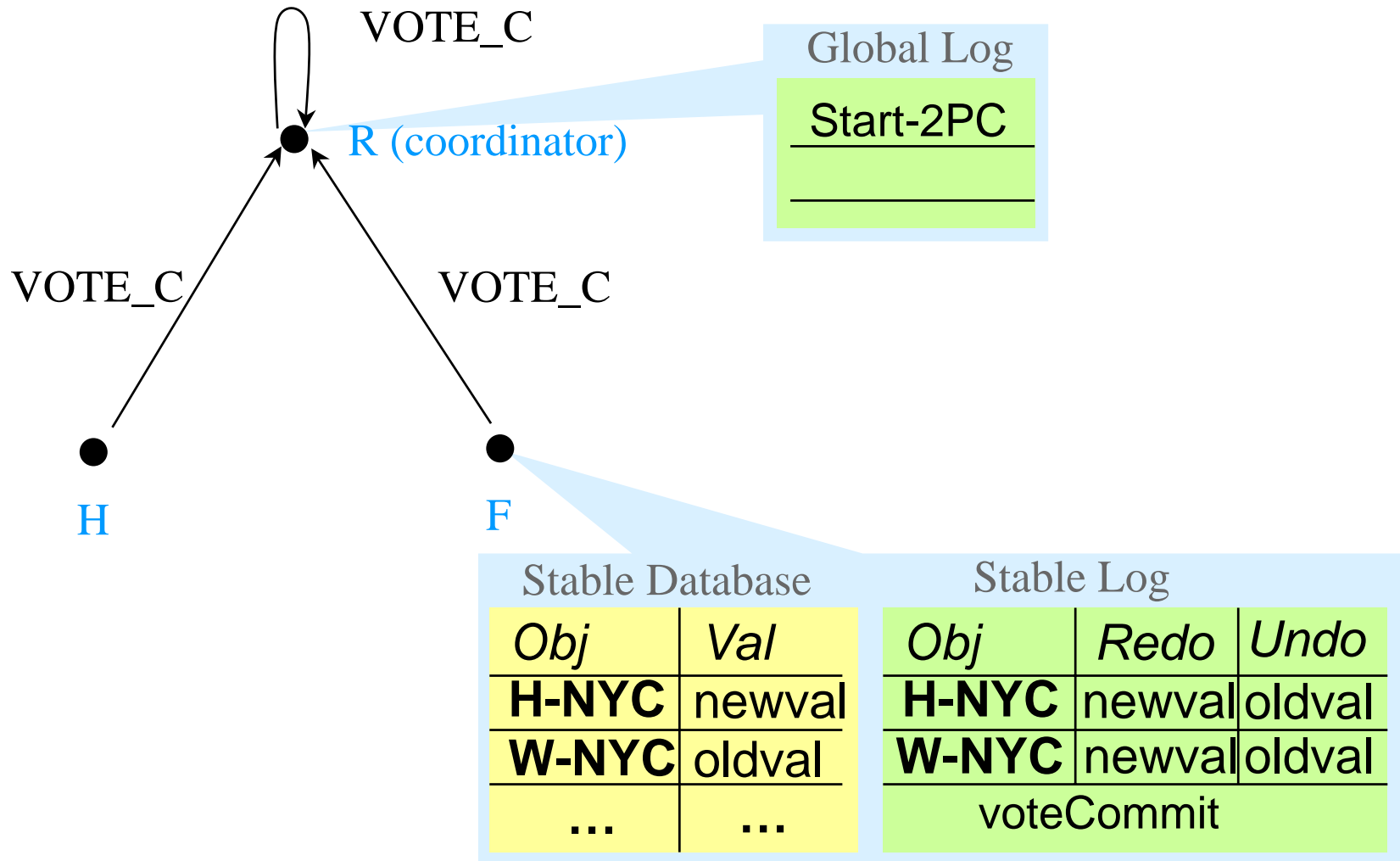
Termination during uncertainty window (12)



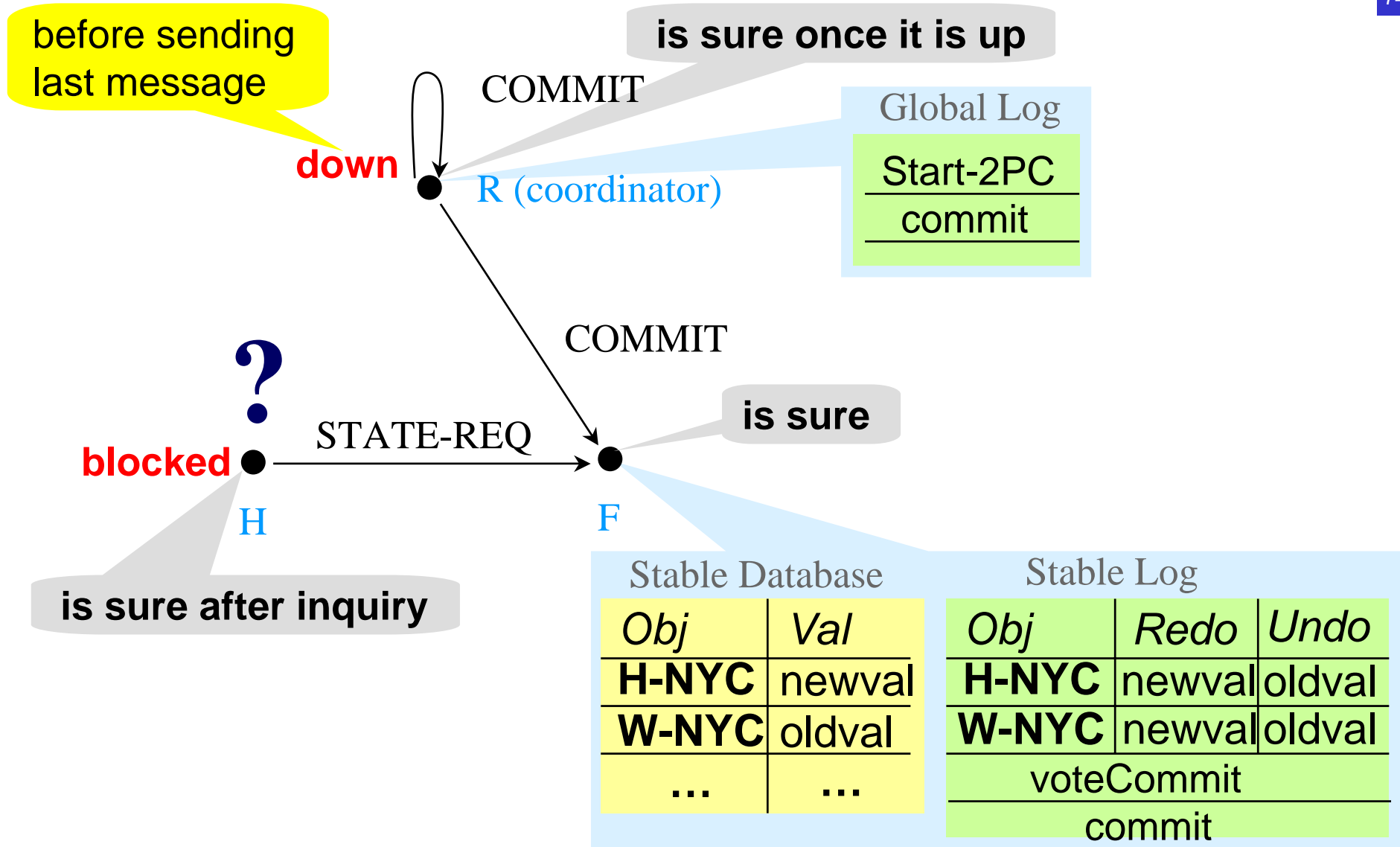
Termination during uncertainty window (13)



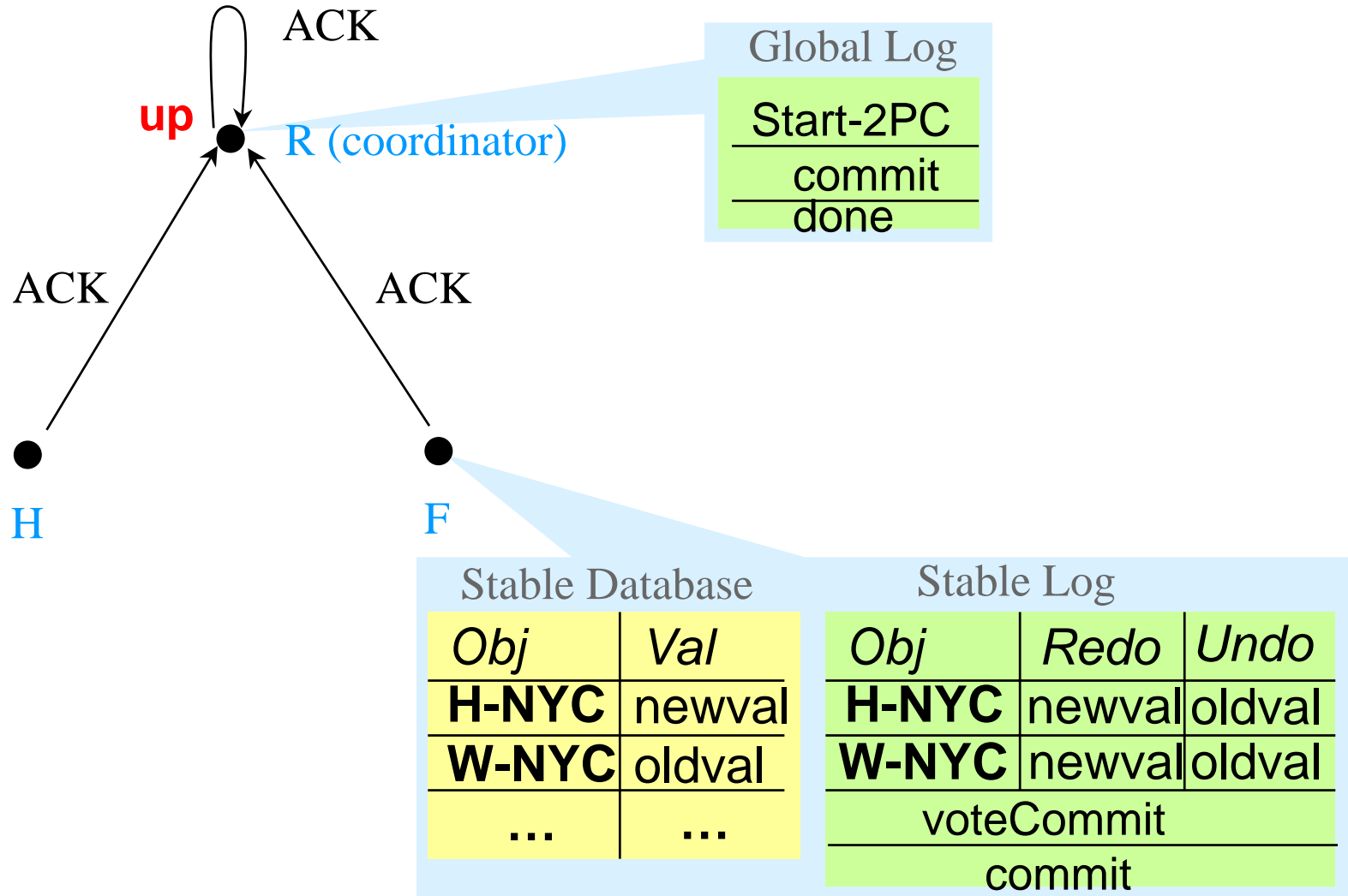
Termination during uncertainty window (14)



Termination during uncertainty window (15)



Termination during uncertainty window (16)



2PC: Performance

Performance

77

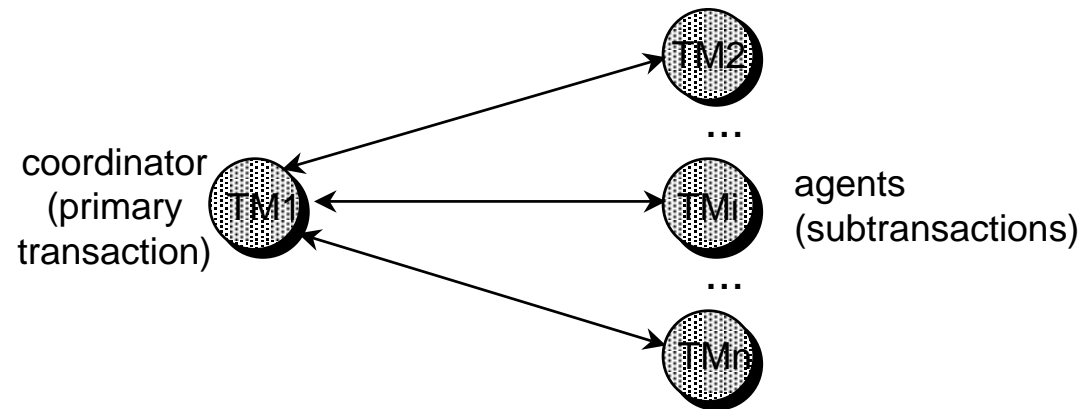
Requirements:

- As few messages as possible, and minimal number of writes to the log file.
- High robustness of protocol wrt failures: Keep probability of „blocking“ low.

Assumption: failures are rare
→ optimize design of system for normal mode.

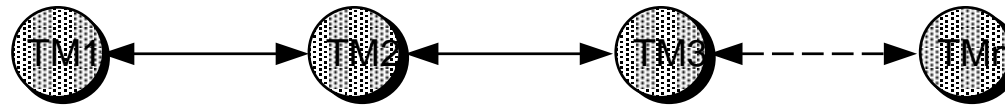
Centralized commit

Basic protocol (treated so far):

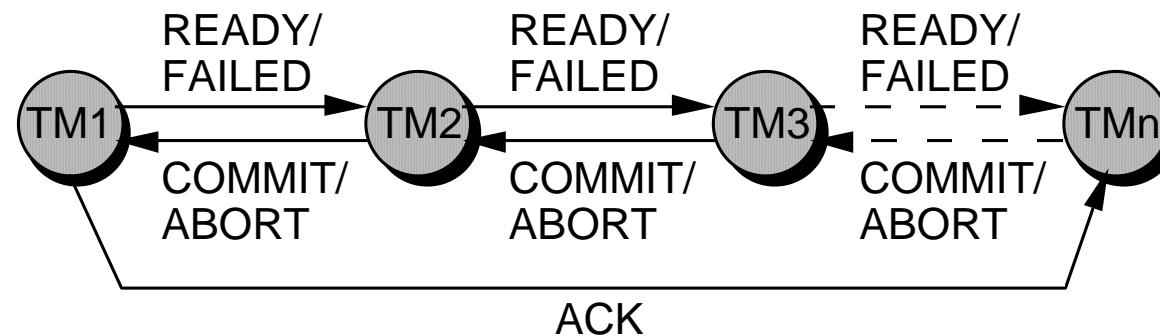


- Typically no communication between agents.
- Advantage: Commit processing in parallel at all agents.
- Four *messages* per agent $\rightarrow 4 \cdot (n-1)$ messages.
- For coordinator and each agent two *log writes* $\rightarrow 2 \cdot n$ log operations. (Processing delayed by duration of I/O.)

Linear commit (1)

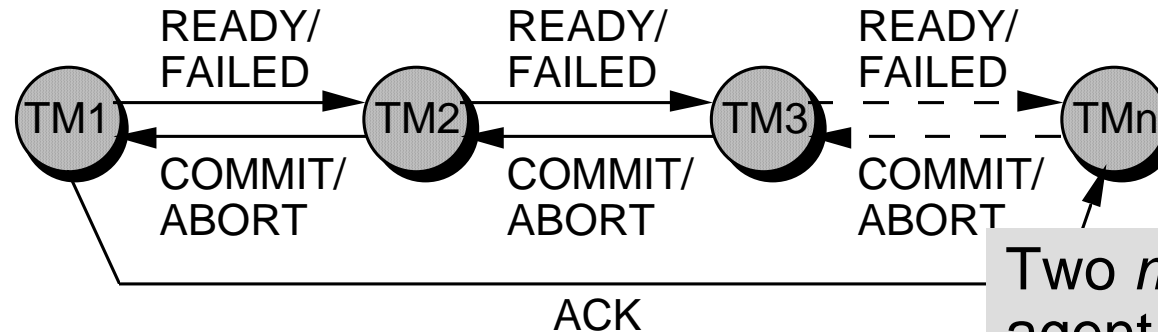


- Commit is processed sequentially → takes longer.
- Phase 1: communication downstream from coordinator (TM1) to last agent (TMn); Phase 2: upstream.



Linear commit (2)

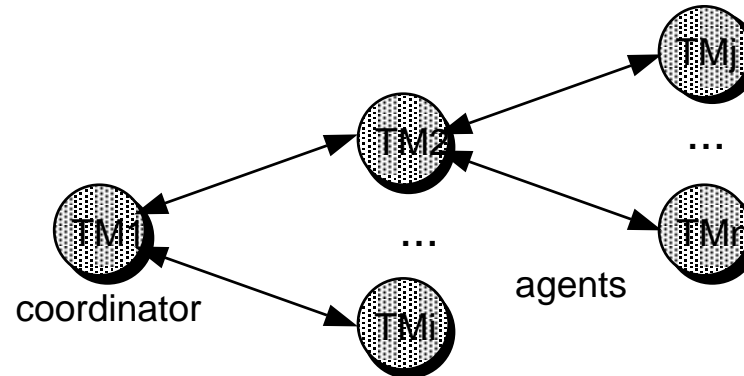
80



- Coordinator enters PREPARED state, passes its local commit decision (READY) to TM2.
 - ◆ Agent enters PREPARED state after having received READY, sends READY to next agent.
 - ◆ Successful termination of transaction is decided once last agent TMn has received READY and has written commit log entry.
 - ◆ Commit decision goes to agents in reverse order; logging and release of locks.
 - ◆ TM1 then sends ACK to TMn; TMn writes *done* log entry.
- Abort of transaction if one of the sites decides on abort; FAILED message is passed on.
 - ◆ Last agent (TMn) becomes coordinator: it logs global abort result and passes it on.

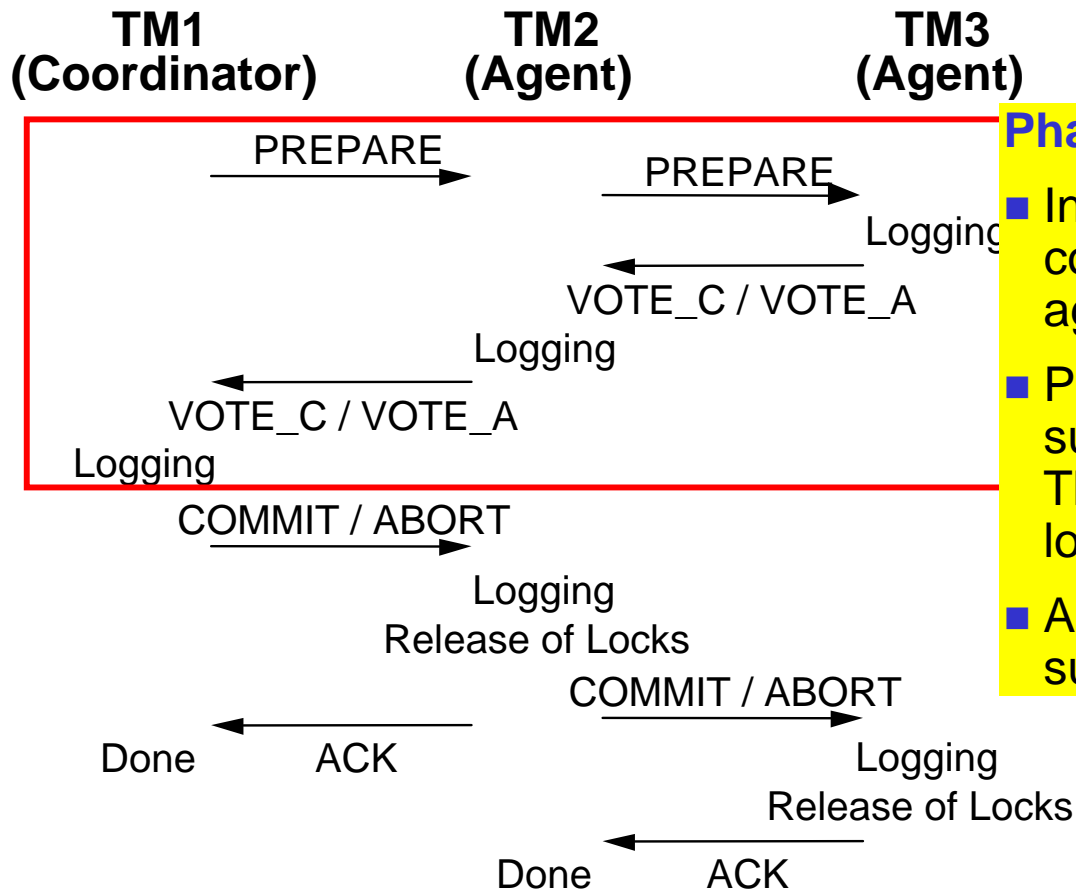
Hierarchical commit (1)

81



- Takes hierarchical invocation structure within global transaction into account, analogous to transaction tree.
- Each agent communicates only with direct ancestor and direct successors.
- Most general structure – previous ones are special case of this one.

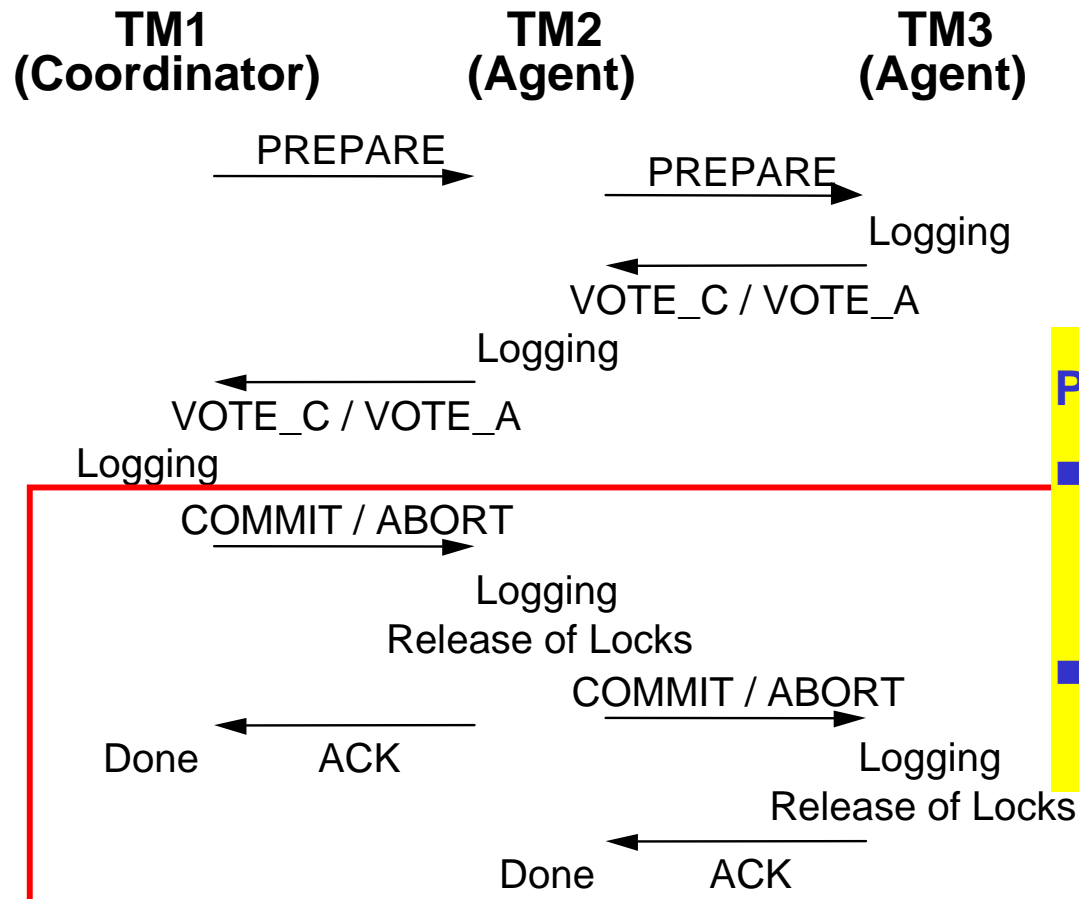
Hierarchical commit (2)



- Phase 1:**
- Intermediate nodes: coordinator for successors in tree, agent from its ancestor's perspective.
 - PREPARE messages go to all successors, wait for their commit votes. Then commit decision for entire subtree, logging + sending it to ancestor.
 - Abort – immediately inform all successors having voted commit.

Hierarchical commit (3)

83

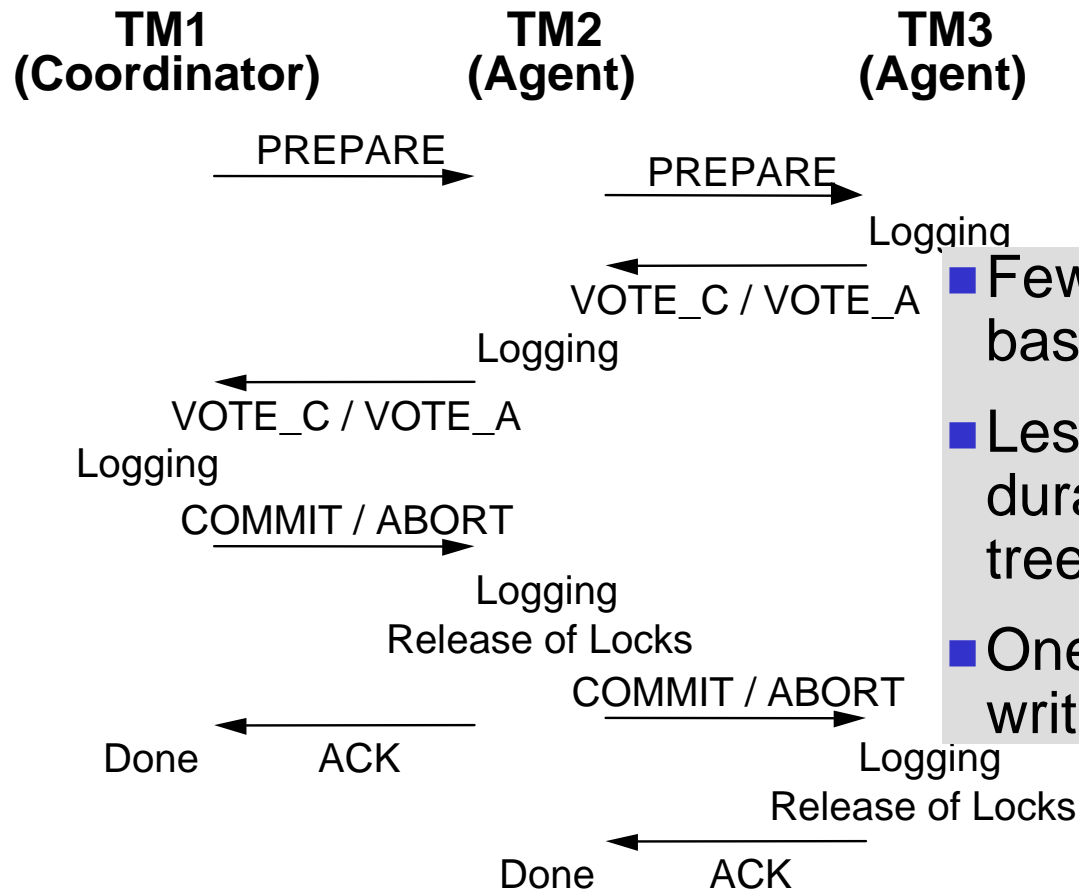


Phase 2:

- Receive decision from ancestor, log it, pass it on to successors, and confirm it immediately.
- After having received all *acks* from successors write done log entry.

Hierarchical commit (4)

84



- Fewer messages compared to basic scheme.
- Less parallelism and longer duration (proportional to height of tree).
- One additional (asynchronous) log write in intermediate node (done).

Presumed Abort

Also note:

Presumed abort improves performance on abort:

- Coordinator can write *abort* log entry asynchronously (no wait!).
- *ack* messages for failed transactions superfluous.
- *done* log entries superfluous at coordinator and intermediate sites.

Incorporated in several products and in standards (ISO/OSI TP, X/Open DTP).

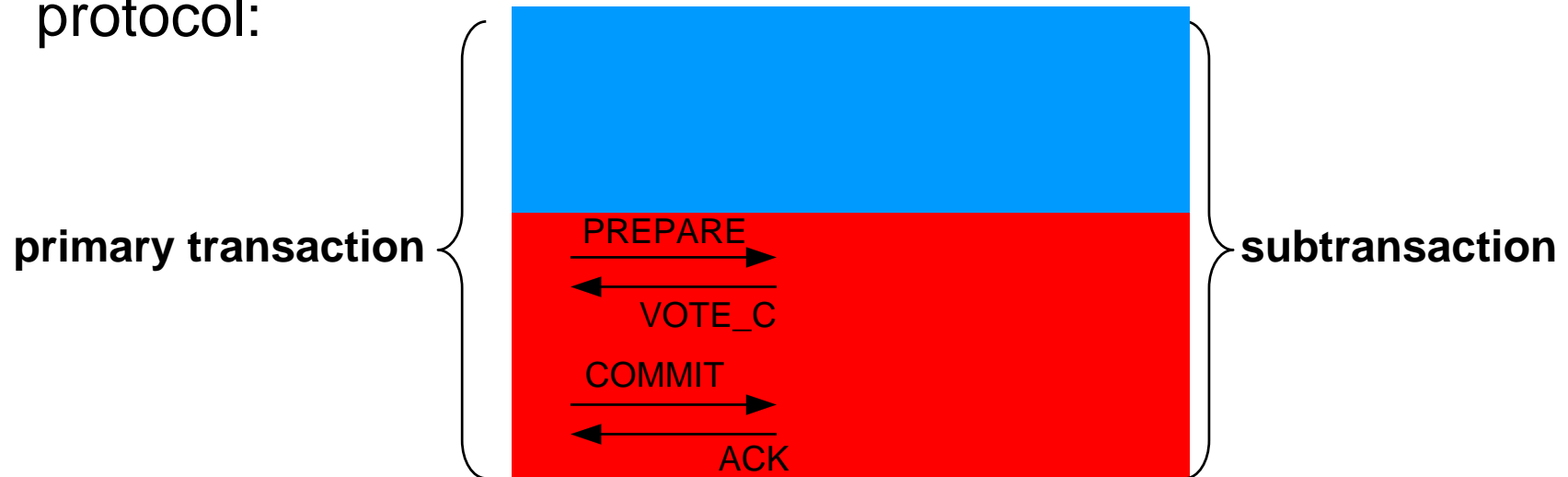
Read-only transactions

- Read-only site:
 - ◆ Neither recovery nor logging needed.
 - ◆ Only release of locks: Already possible in Phase 1 of 2PC, irrespective of success of global transaction → save entire second commit phase.
- If m subtransactions are read-only (of $n-1$),
 - ◆ number of messages reduced by $2m$ to $4 \cdot (n-1) - 2m$,
 - ◆ number of log writes reduced to $2n - m$.
 - ◆ If global transaction is read-only ($m = n$), only $2 \cdot (n-1)$ messages and no log writes.

One-Phase Commit (1)

87

- So far: work in subtransactions separated from commit protocol:

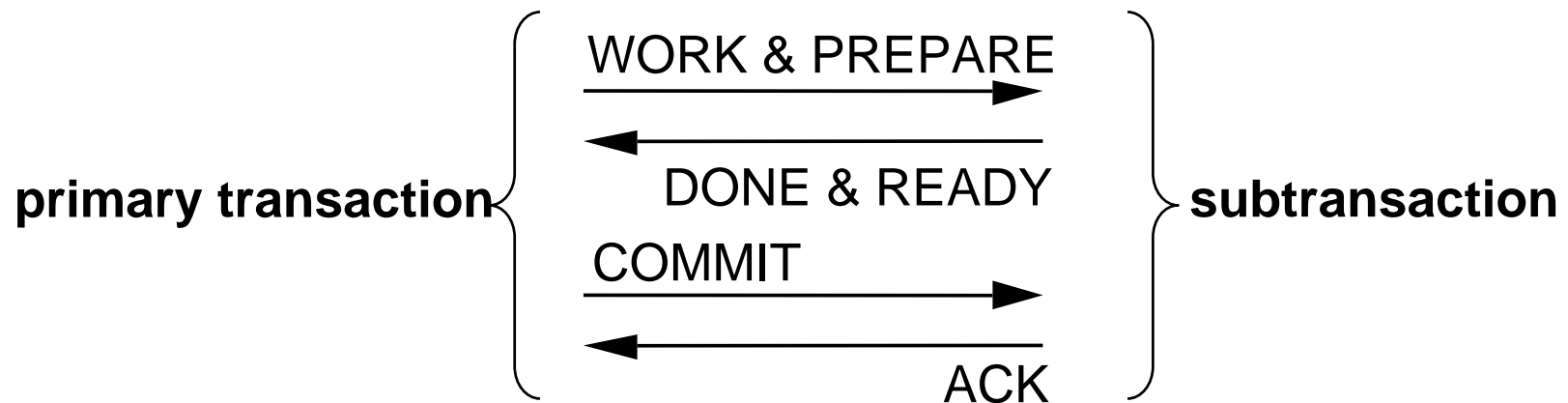


- Short distributed transactions with only one external database operation (e.g., money transfer to bank): commit processing is more expensive than transaction itself.
- ➔ Combine PREPARE message with WORK message.

One-Phase Commit (2)

88

- Subtransaction enters PREPARE state immediately after having executed operation and before replying to primary transaction.



- We save first commit phase, commit processing consists of only one phase to communicate global result.
- ➔ Only two messages per agent.

Why does it only work with short transactions?

Three-phase commit protocol (3PC)

Atomic Commitment Protocol - Summary

90

Atomic Commitment Protocol (ACP) is an algorithm that takes a (distributed) decision *Commit* or *Abort* such that:

- All sites that come to a decision come to the same decision.
- A site cannot change its decision once it has been taken.
- Decision for commit can only be taken if all sites have voted commit.
- If no failures occur and all sites vote commit the decision will be commit.
- Failures may occur but the algorithm can handle them. After all failures have been fixed and no new failures occur for a certain period of time a decision will be taken.

Atomic Commitment Protocol - Theorems

91

- **Theorem 1:**

If communication failures and total site failures (all sites down) are possible, each ACP may lead to blocking.

- **Theorem 2:**

No ACP can guarantee independent recovery of failed sites.

(Independent recovery: Global state can be recovered without communication.)

Atomic Commitment Protocol – 2PC

92

■ Theorem 1:

If communication failures and total site failures (all sites down) are possible, each ACP may lead to blocking.

■ Theorem 2:

No ACP can guarantee independent recovery of failed sites.

2PC may even be blocking if no communication failures occur!

- All sites fail.
- Failing site splits network.

Long waits if agents are in *prepared* state and coordinator is down.

Site failures only

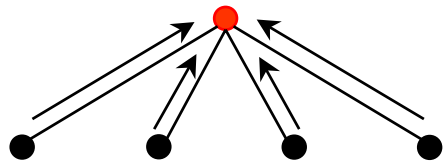
Invest more effort into reduction of blocking situations!

Solution 1: Tolerate site failures only:

- Non-blocking, except for total failures (all nodes down).
- Since only the sites may fail, **we can safely assume** that
 - ◆ all **up**-sites can communicate, and
 - ◆ the only reason a site may **time out** is that a partner site is **down**.

Introduce Non-Blocking Characteristic (1)

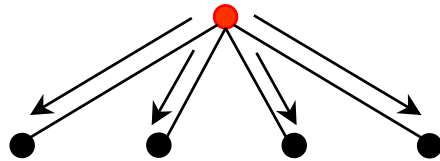
If an operational site is uncertain, no other site has yet committed.



VOTE_C-
Messages

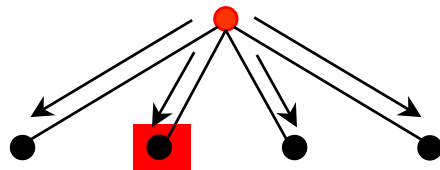
Non-blocking characteristic is not violated.

But 2PC may block.



COMMIT-
Messages

Non-blocking characteristic is typically violated, since COMMIT messages do not arrive at same time.



COMMIT-
Messages

Non-blocking characteristic is violated.

Introduce Non-Blocking Characteristic (2)

Conclusions:

- Non-blocking characteristic is not violated as long as all sites are uncertain.
- Sites that aborted cannot contribute to blocking because they did not decide on commit.

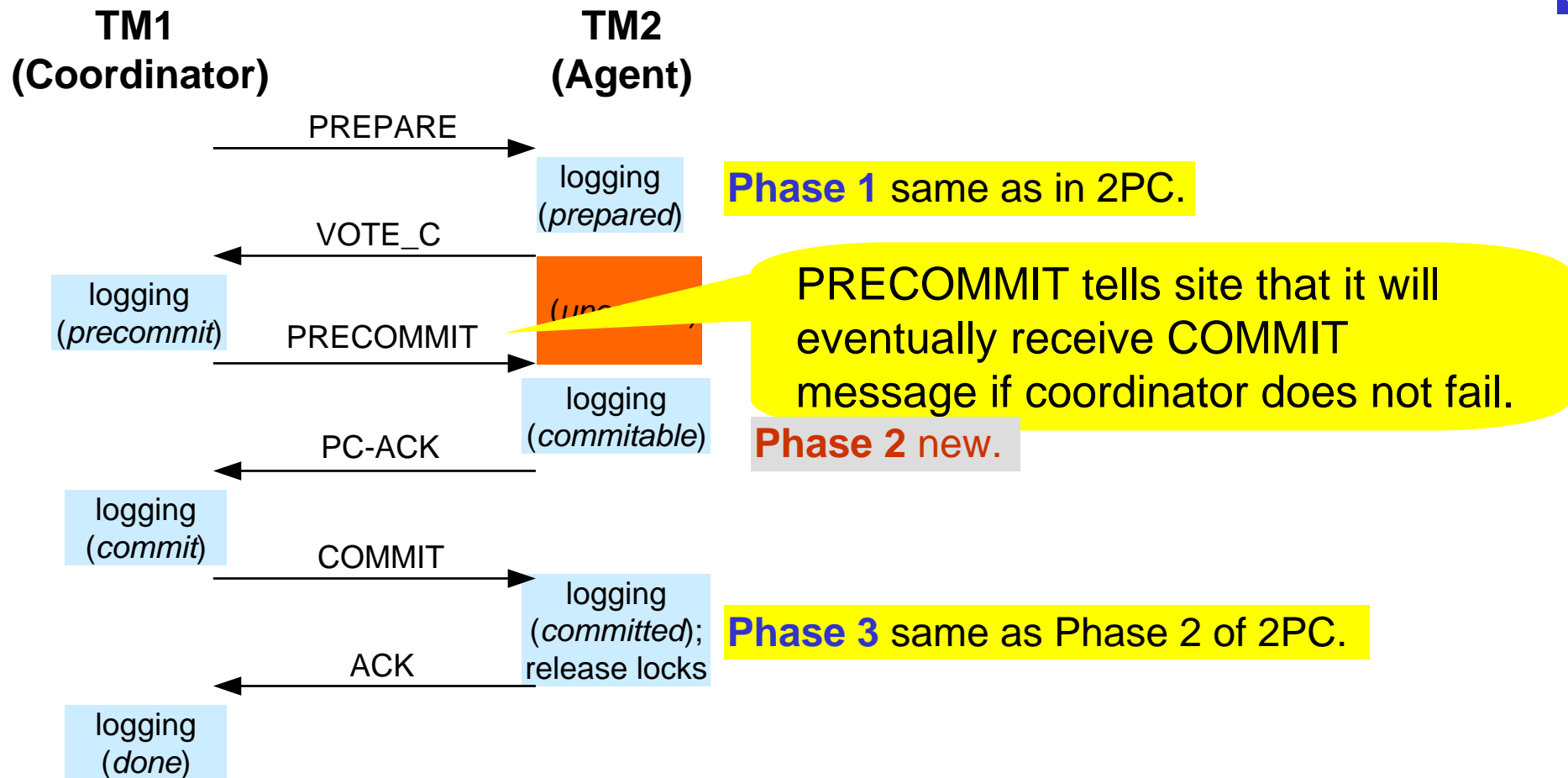
Desirable characteristic:

- Uncertain sites are allowed to abort.

Approach:

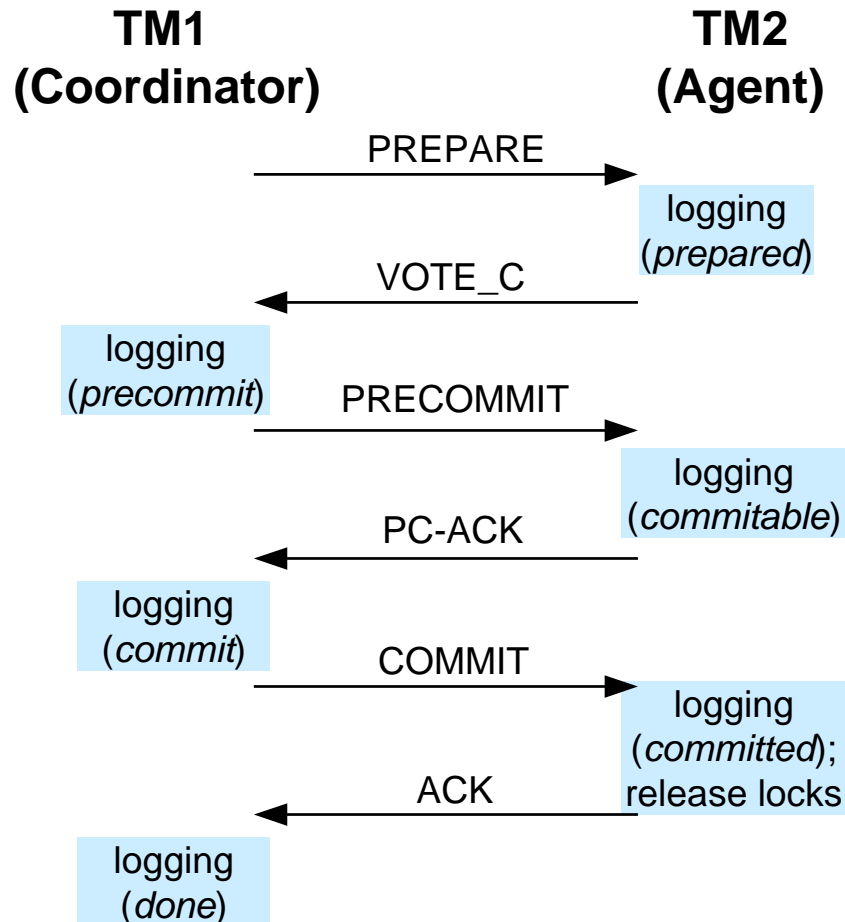
- End uncertainty, but do not yet commit \Rightarrow for a while sites that are uncertain and those that are certain but did not commit may co-exist.

3PC General scheme (1)



3PC General scheme (2)

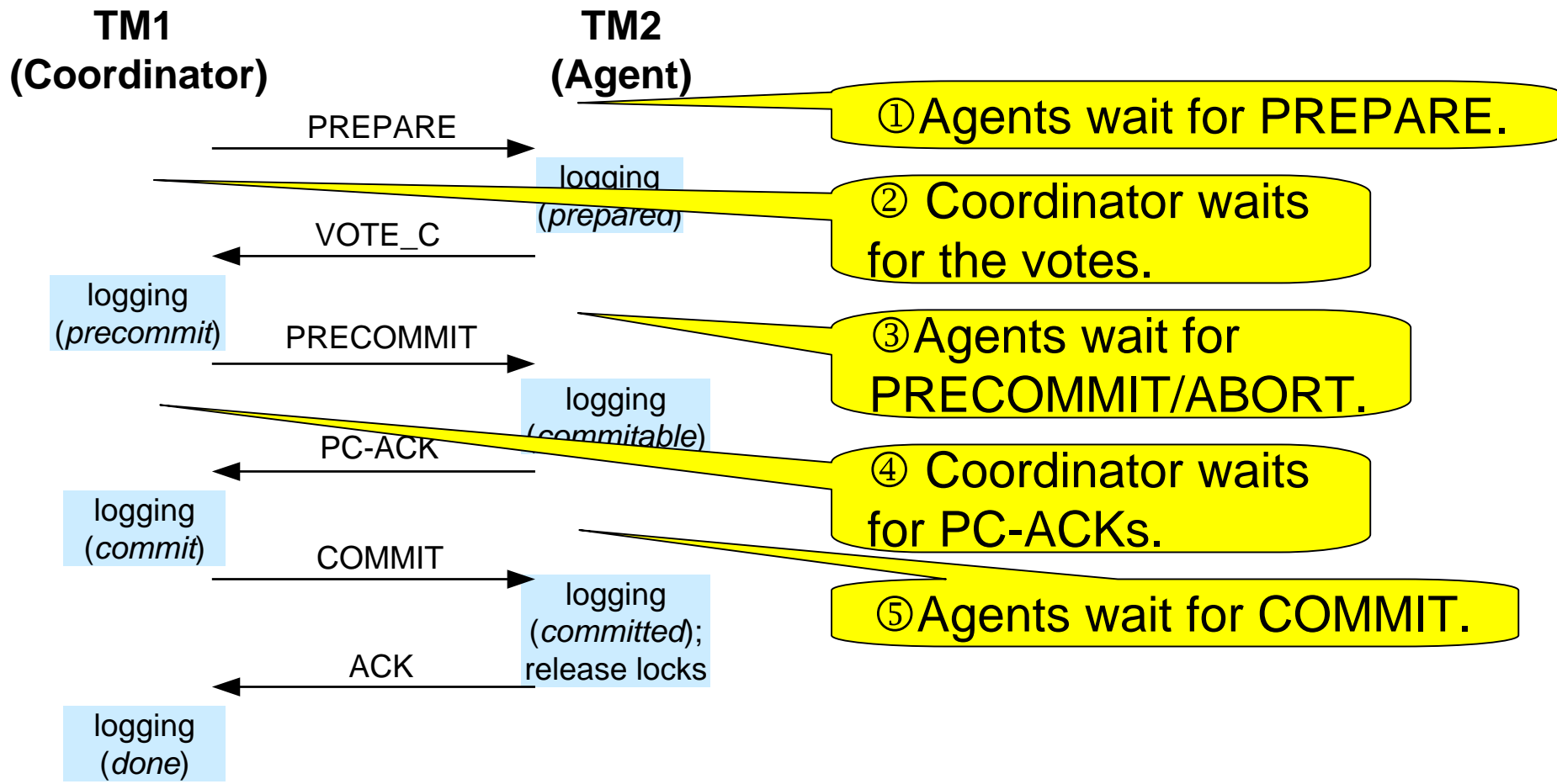
97



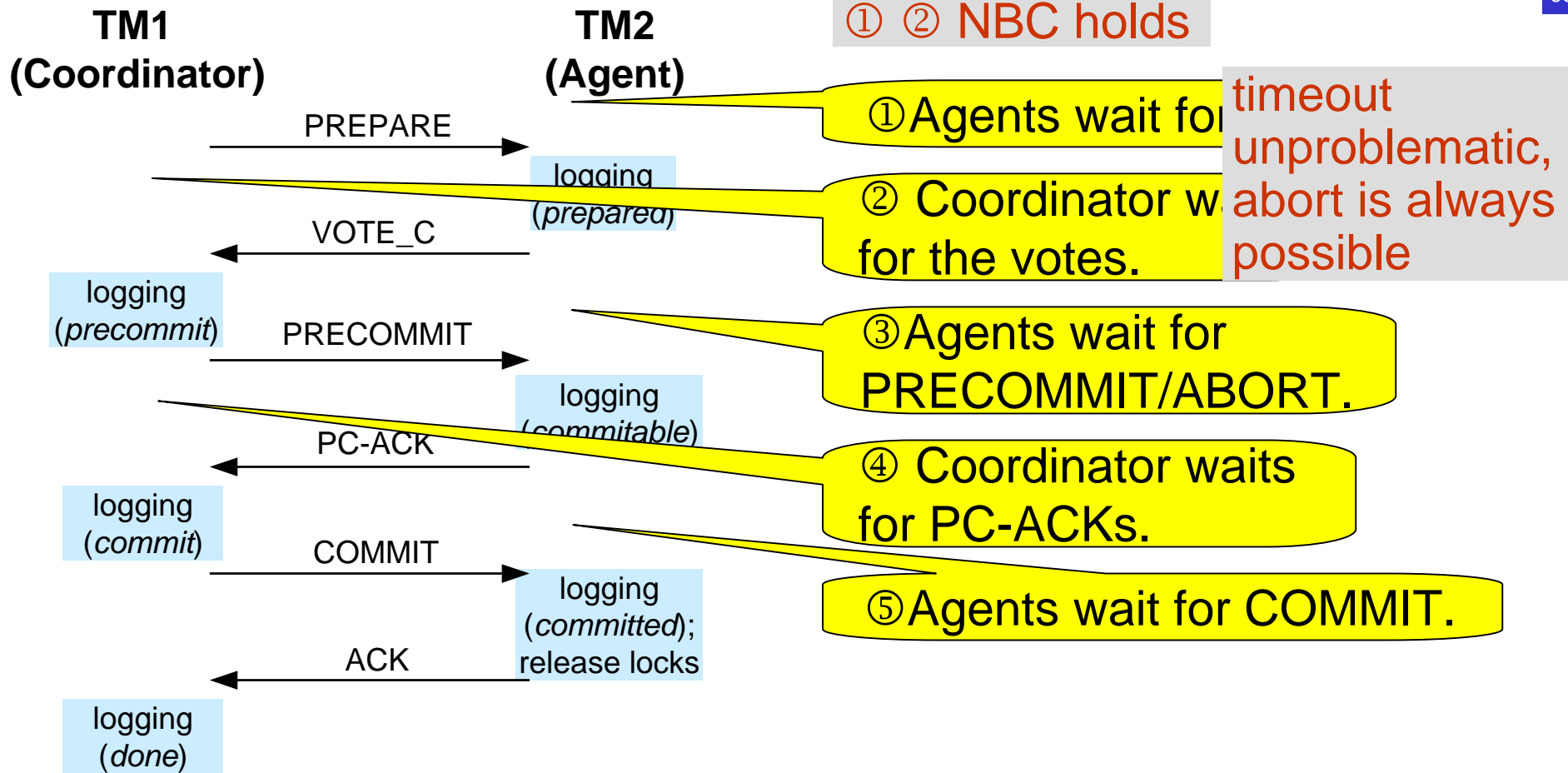
Phase 2:

- PRECOMMIT: Coordinator asserts that it will not abort transaction in future.
 - Site is no longer uncertain.
 - Site knows that it will eventually receive COMMIT message if coordinator does not fail.
 - TA may still abort if this coordinator fails.
- Agents write log entry and confirm.
- After PC-ACK messages have arrived, coordinator decides on commit and writes respective log entry.

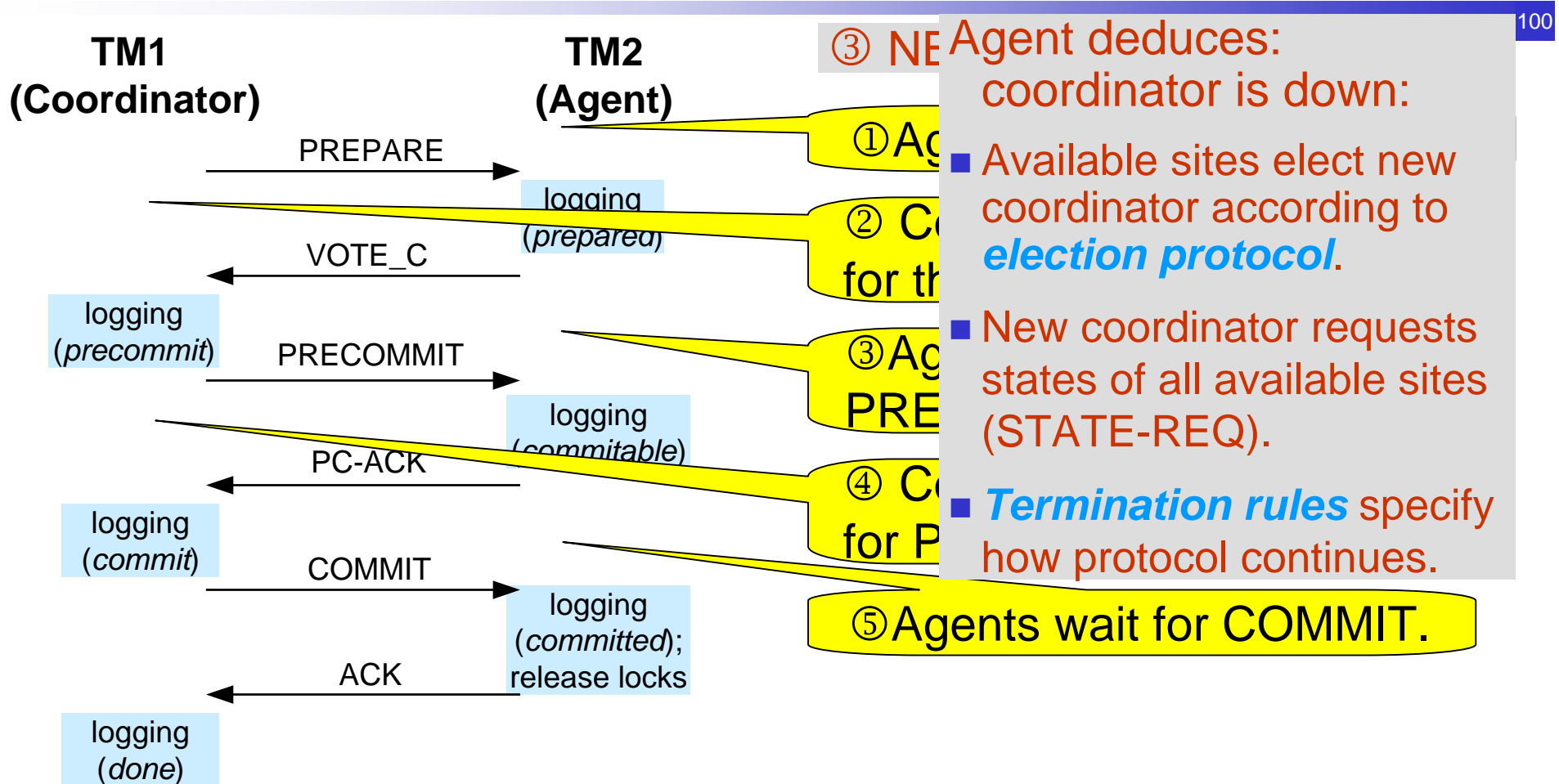
3PC Failure model (1)



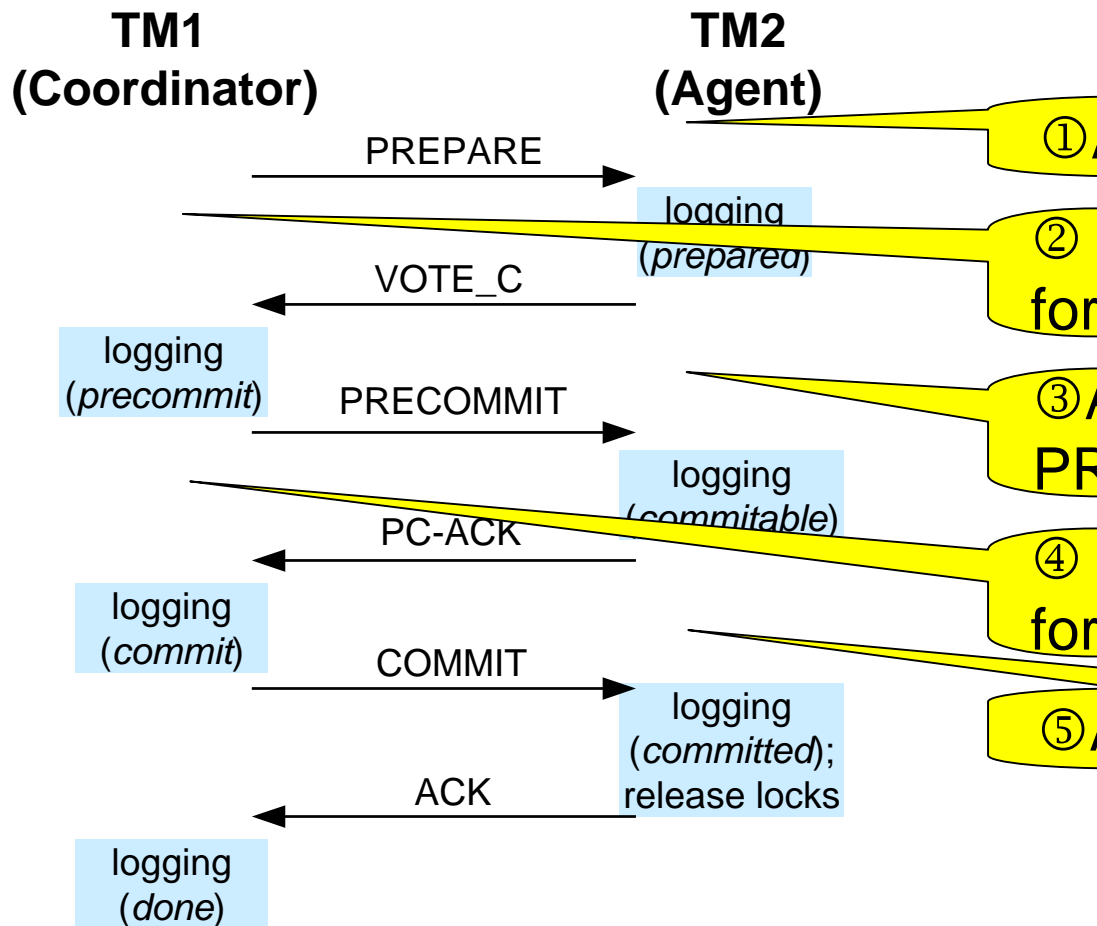
3PC Failure model (2)



3PC Failure model (3)



3PC Failure model (4)



④ NBC satisfied: Failed agent may still be uncertain, but no site committed yet.

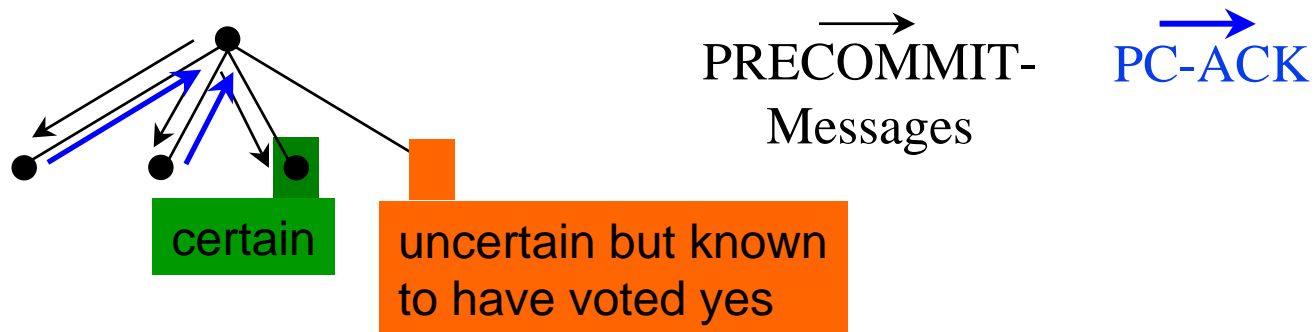
- Coordinator ignores missing PC-ACK messages; goes on after timeout period. ⇒ NBC remains satisfied.

- Failed agent must find out state of protocol after recovery.

for PC-ACKs.

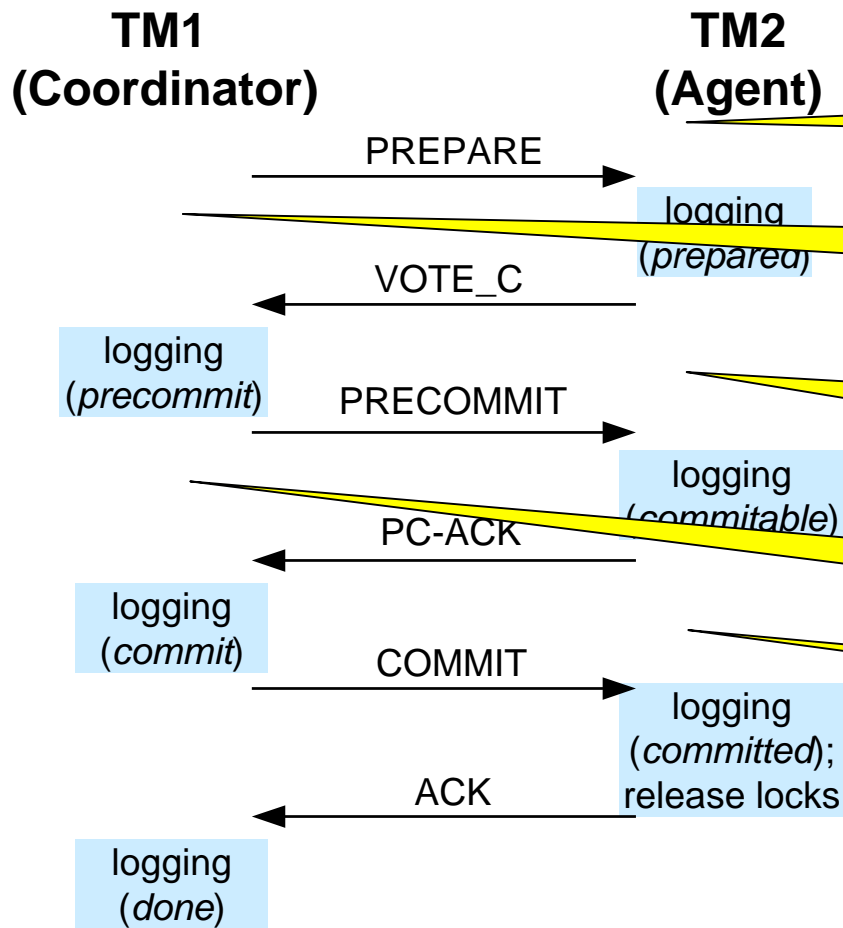
⑤ Agents wait for COMMIT.

3PC Failure model (4)



(message delays are always possible)

3PC Failure model (5)



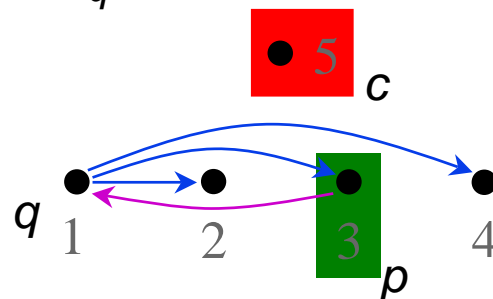
- ⑤ NB Agent deduces: Coordinator is down:
 - Available sites elect new coordinator according to *election protocol*.
 - New coordinator requests states of all available sites (STATE-REQ).
 - *Termination rules* specify how protocol continues.
- ① Agent deduces: Coordinator is down.
- ② Coordinator deduces: Agent is down.
- ③ Agent deduces: Coordinator is down.
- ④ Coordinator deduces: Agent is down.
- ⑤ Agents wait for COMMIT.

Election Protocol (1)

104

- Prerequisites:
 - ◆ Linear ordering of sites ($, <'$).
 - ◆ Each site p maintains set UP_p of sites which p believes to be up.
- Agent p times out due to failed present coordinator c .
 - ◆ Remove c from UP_p and select smallest site q .
 - ◆ If $p \neq q$ send message UR-ELECTED to q .
- Agent q considers itself the new coordinator unless it can communicate with processes with smaller ID
 - ◆ Removes c from UP_q .
 - ◆ Send messages STATE-REQ (in lieu of PC-ACK) to agents in its UP_q .

$UP_p \neq UP_q$ possible!



UR-ELECTED

STATE-REQ

Election Protocol (2)

105

- Let p' be such an agent receiving STATE-REQ from q .
- If p' did not timeout, it deduces the failure of c and recognizes q as the new coordinator.
 - ◆ Remove c and all $q' < q$ from $UP_{q'}$.
- Due to message delay p' may have received STATE-REQ from a new coordinator q' before that one failed, too.
 - ◆ We know $q' < q \Rightarrow$ ignore STATE-REQ messages from all sites $q'' < q$.

Termination rules (1)

106

After new coordinator has collected states of available sites:

- TR1: a site has aborted ③
⇒ coordinator decides abort and sends out ABORT.
- TR2: a site has committed ⑤ **some sites received COMMIT**
⇒ coordinator decides commit and sends out COMMIT.
- TR3: all sites that have reported their state are uncertain (PREPARED state) ③
⇒ coordinator decides abort and sends out ABORT.
- TR4: a process is committable, none is committed ⑤
⇒ PRE-COMMIT messages to sites in uncertain state and wait for ACKs.
Then commit decision, and COMMIT messages are sent out.

Termination rules (2)

107

How to deal with failures during termination:

- Ignore failures of agents.
- Failures of coordinator: Termination algorithm is repeated, but with fewer nodes \Rightarrow nodes do not need to remain available.

Lemma: If NBC holds before the termination protocol starts, it will hold after even a partial execution of that protocol.

Site recovery after failure

108

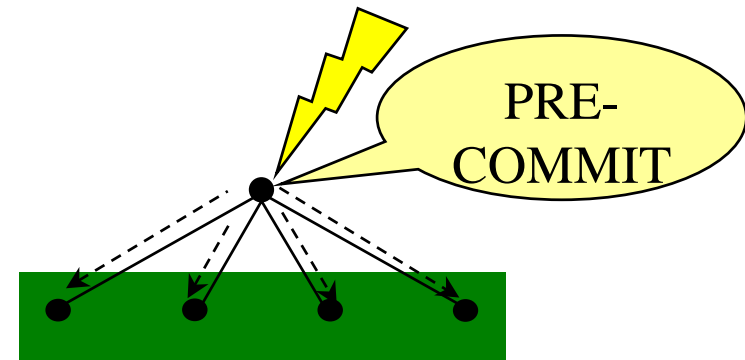
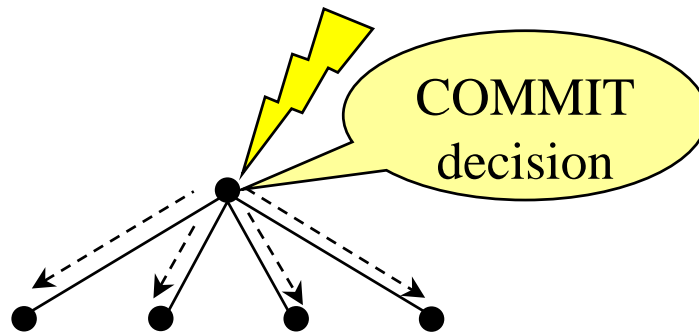
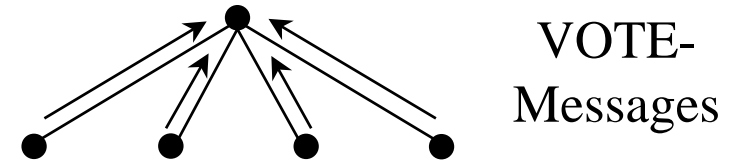
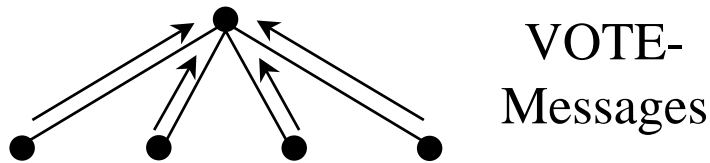
- Failure before VOTE_C: Unilateral abort and independent recovery.
- Failure after COMMIT or ABORT: Decision was taken, independent recovery.
- Failure while *uncertain* or *committable*: Contact other sites.
 - ◆ NBC guarantees that the site will eventually receive and adopt a decision.

Total failure

Protocol blocks in case of total failures. Unblocking proceeds as follows:

- Suppose that site p that has just recovered.
- As before, independent recovery before VOTE_C or after COMMIT or ABORT \Rightarrow Communicate decision to the other sites.
- As before, autonomous decision is not possible if the failure occurred while *uncertain* or *committable*.
 - ◆ Decision for commit or abort could have been taken by a site failing after p .
 - ◆ However, it can take a decision if it was the last one to fail \Rightarrow invoke the termination protocol.

2PC vs. 3PC



Blocked if coordinator takes decision but fails to send messages.

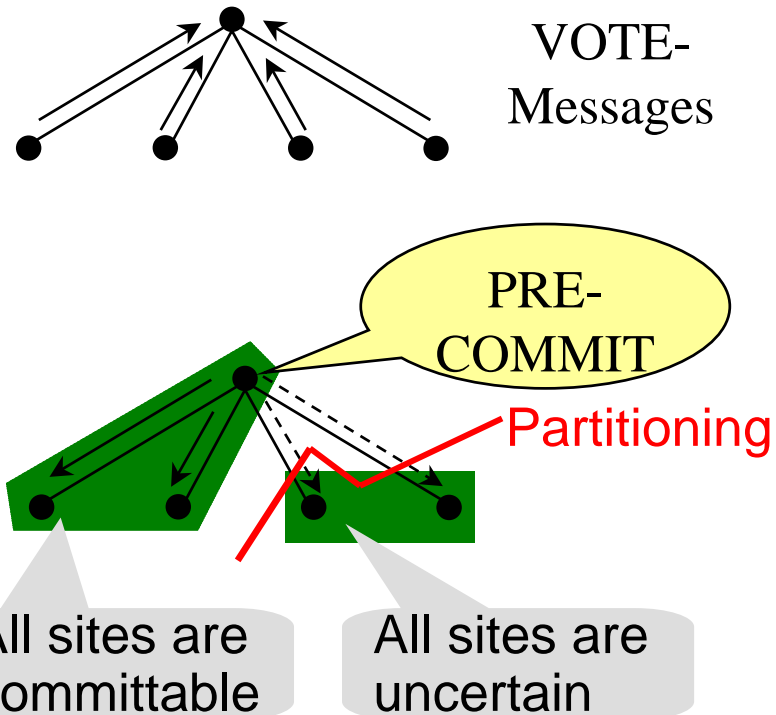
If coordinator initiates decision but fails to send messages:

- No wait, instead agents elect new coordinator.
- We know votes of all other nodes and that no decision has yet been taken.

Communication failures (1)

111

Present solution may result in inconsistent global state:



Sites ultimately commit (TR4).

Sites ultimately abort (TR3).

Problem: Election of multiple coordinators!

Communication failures (2)

112

Invest more effort into reduction of blocking situations!

Solution 1: Tolerate site failures only:

- Non-blocking, except for total failures (all nodes down).
- Since only the sites may fail, **we can safely assume** that
 - ◆ all **up**-sites can communicate, and
 - ◆ the only reason a site may **time out** is that a partner site is **down**.

None is true any longer!



Solution 2: Tolerate both, site and communication failures.

3PC extended (1)

113

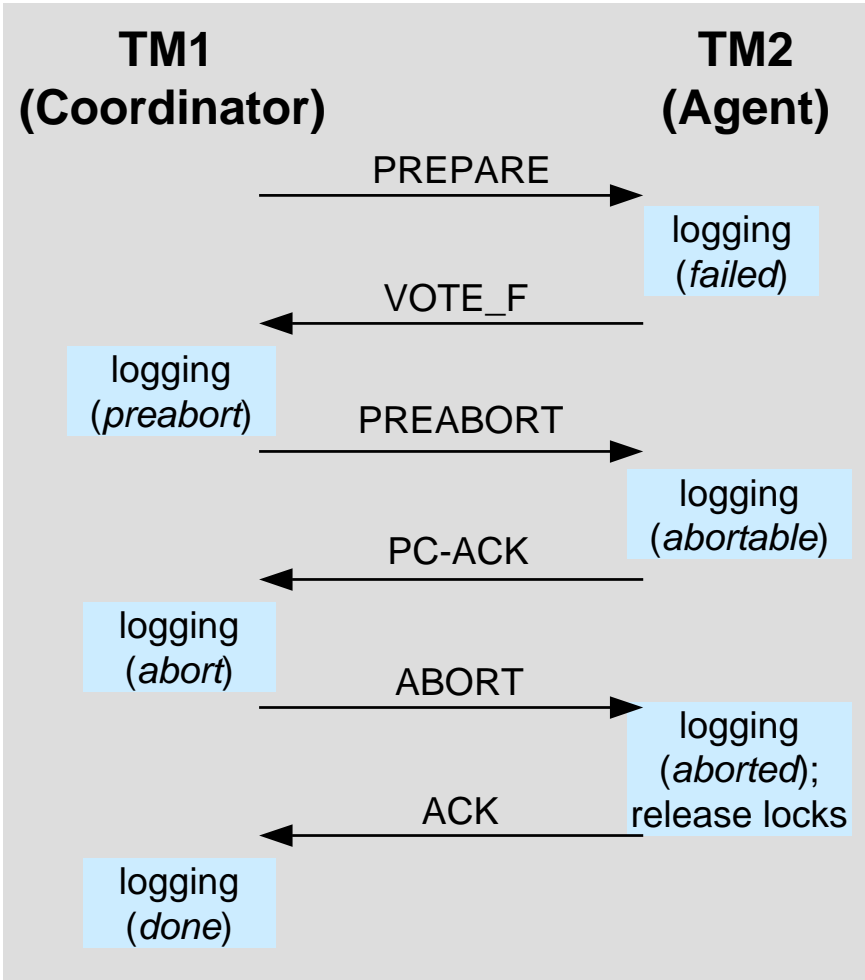
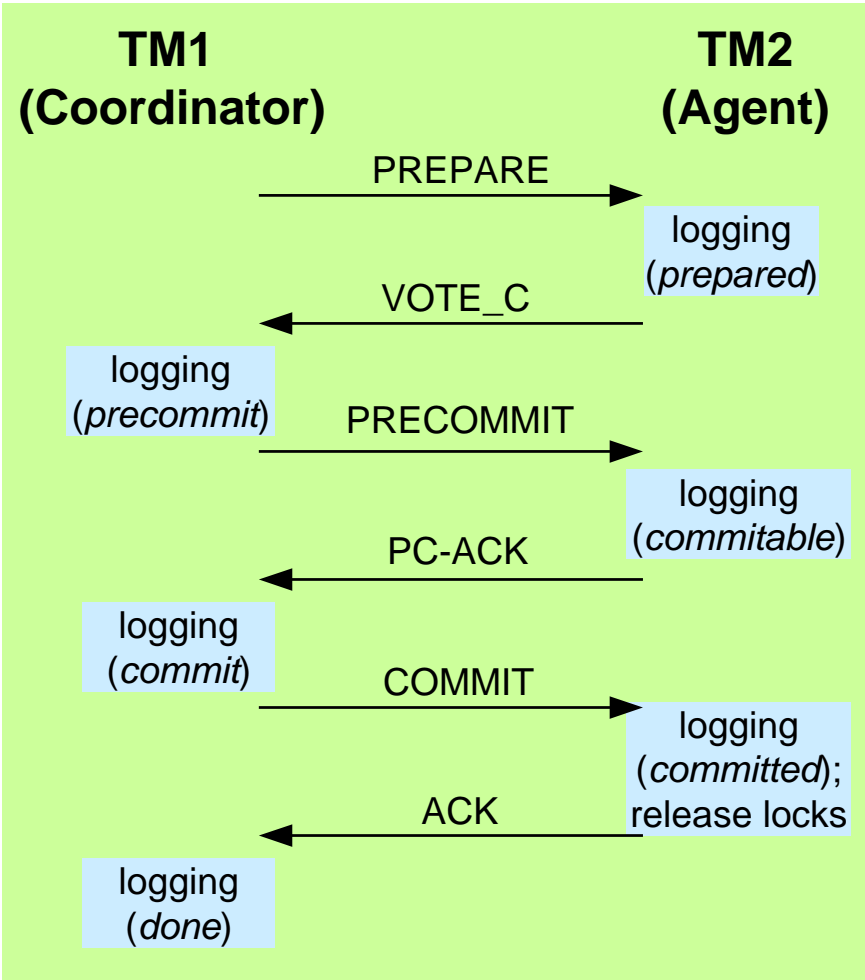
Problem: Election of multiple coordinators!

Remedy: A coordinator is only allowed to reach a decision if it can communicate with a majority of the sites.
⇒ Ensures that decision reached by two different coordinators is based on at least one site in common.

Conflict: A common site detects a conflict if its one coordinator intends to decide on commit and another on abort.

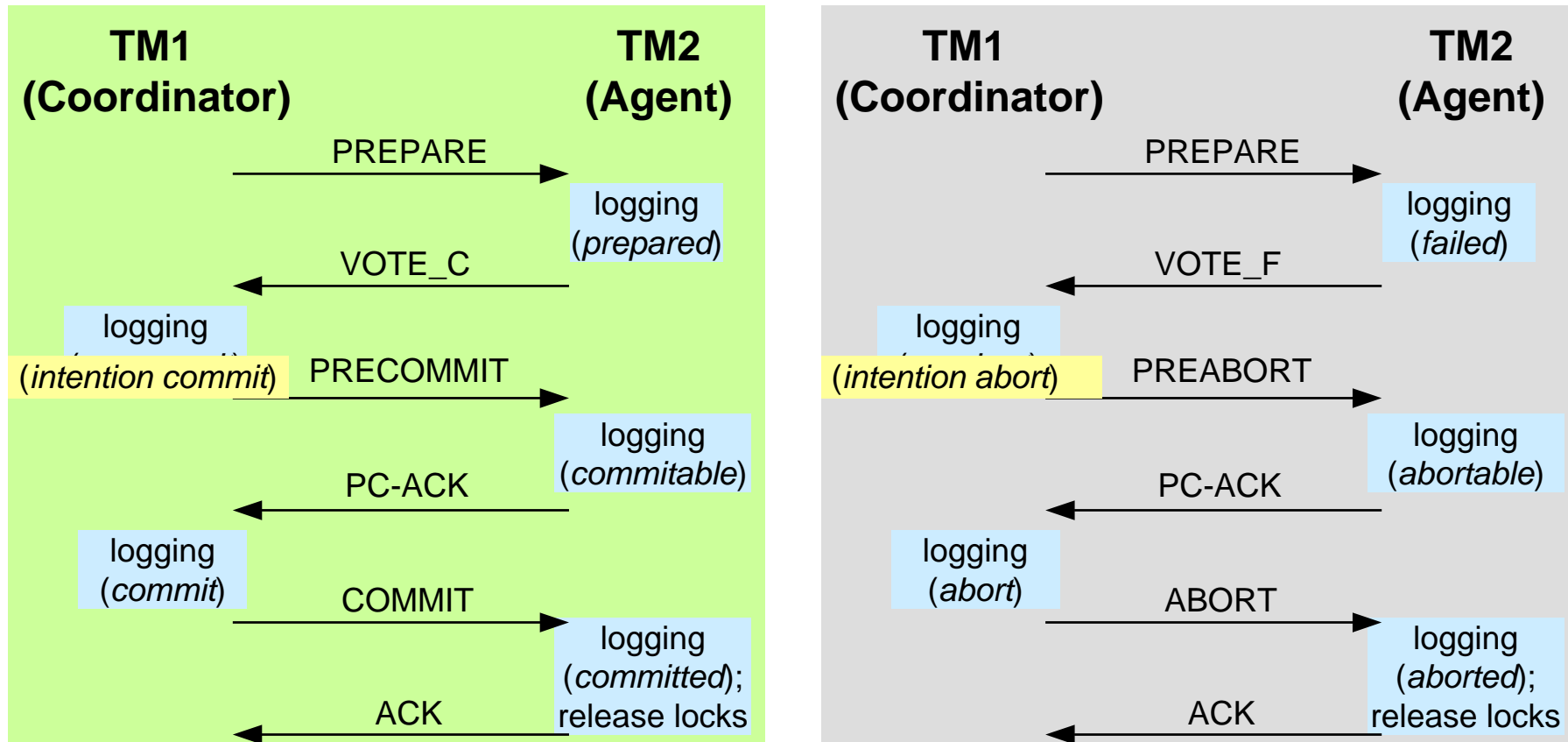
Required: Both, PRE-COMMIT and PRE-ABORT.

3PC extended (2)



3PC extended failure model

115



Coordinator no longer reachable by all:

- Sites in subnets elect new coordinator according to *election protocol*.
- New coordinators request states of all available sites (STATE-REQ).
- *Termination rules* specify how protocol continues.

Majority termination rules (1)

116

After new coordinator has collected states of available sites:

- TR1: a site has aborted
⇒ coordinator decides abort and sends out ABORT.
- TR2: a site has committed
⇒ coordinator decides commit and sends out COMMIT.
- TR3:
 1. Coordinator: One committable and a majority non-abortable states (prepared or committable) ⇒ *PRE-COMMIT messages* to all agents that have not sent committable.
 2. Agent: *PRE-COMMIT* ⇒ new state committable
⇒ *PRE-COMMIT-ACK*
 3. Coordinator: If agents in state committable form a majority: commit; otherwise blocking.

Majority termination rules (2)

117

After new coordinator has collected states of available sites:

- TR4:
 1. Coordinator: Majority of non-committable states (prepared or abortable) \Rightarrow *PRE-ABORT* messages to all agents that have not replied abortable.
 2. Agent: *PRE-ABORT* \Rightarrow new state abortable \Rightarrow *PRE-ABORT-ACK*.
 3. Coordinator: if agents in state abortable form a majority: abort; otherwise blocking.
- TR5: Otherwise: blocking.

Illustration (1)

- After timeout, green component elects new coordinator.
- New coordinator collects states from nodes in component.
- New coordinator decides on 'intention abort' (TR4).

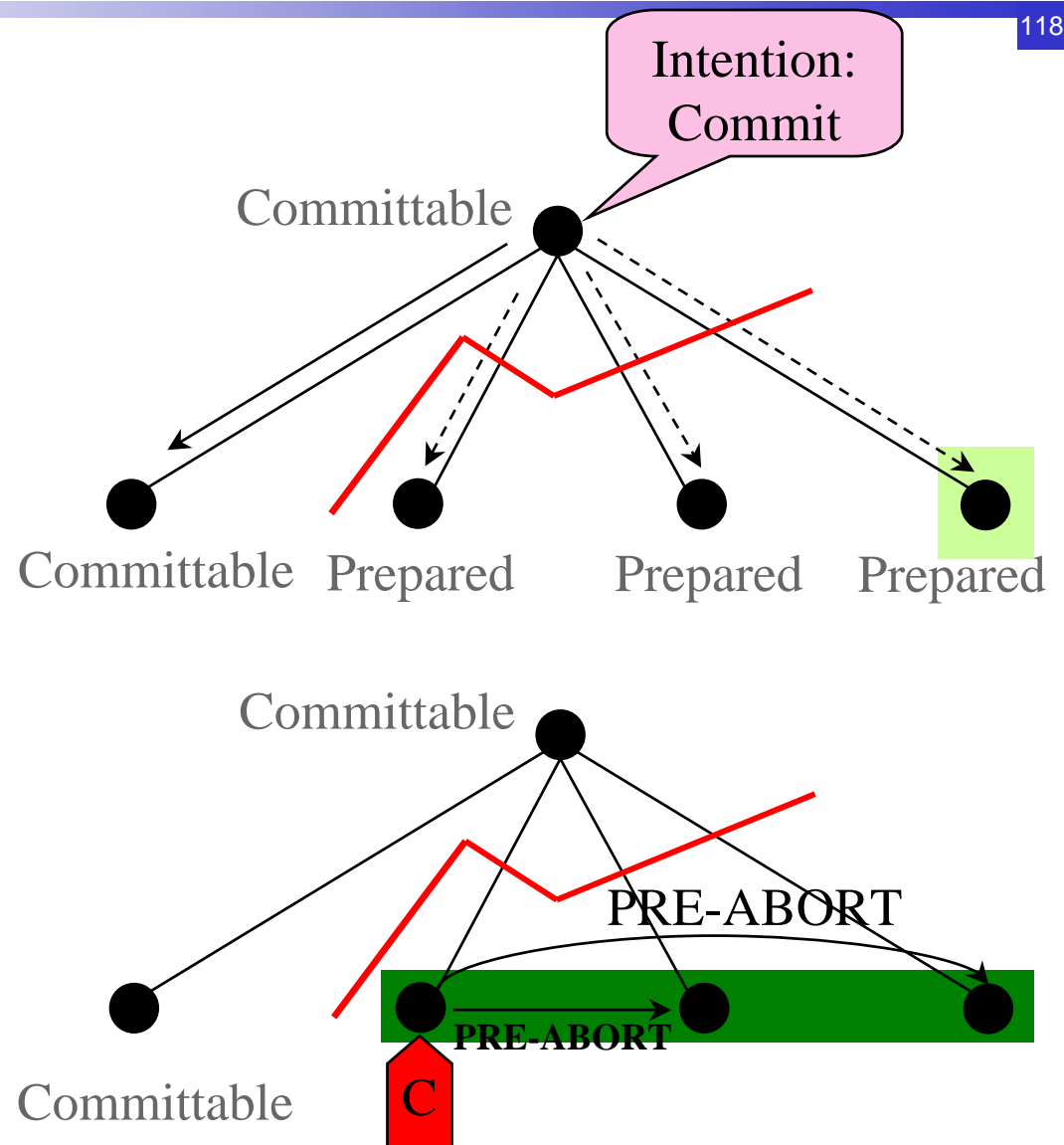


Illustration (2)

119

- What happens after communication has been re-established?

⇒ Agents in 'abortable' state will not react to PRE-COMMIT message, coordinator will not be able to form a commit majority ⇒ abort (TR4).

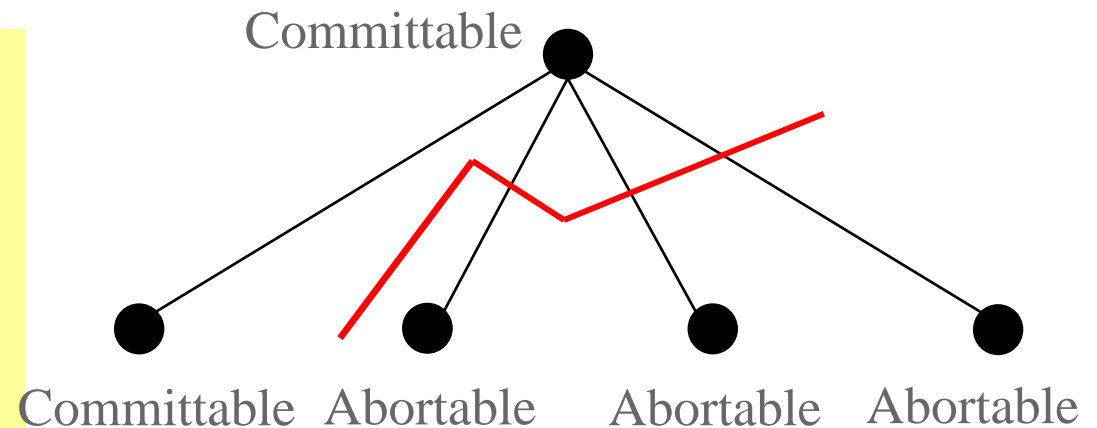
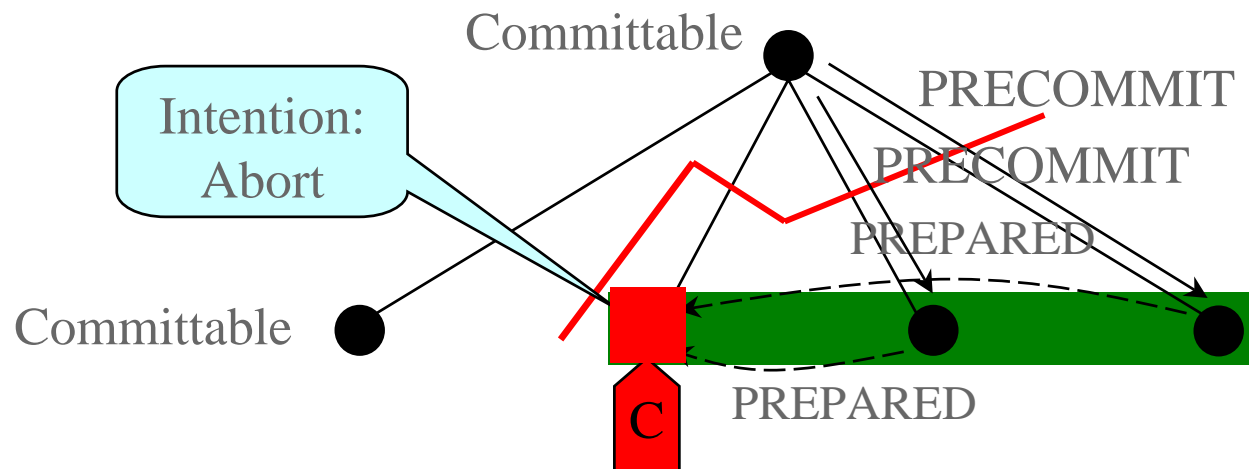


Illustration (3)

- What if there were no PRE-ABORT messages?



Performance comparison

■ Number of messages

(including optimized treatment of read-only subtransactions)

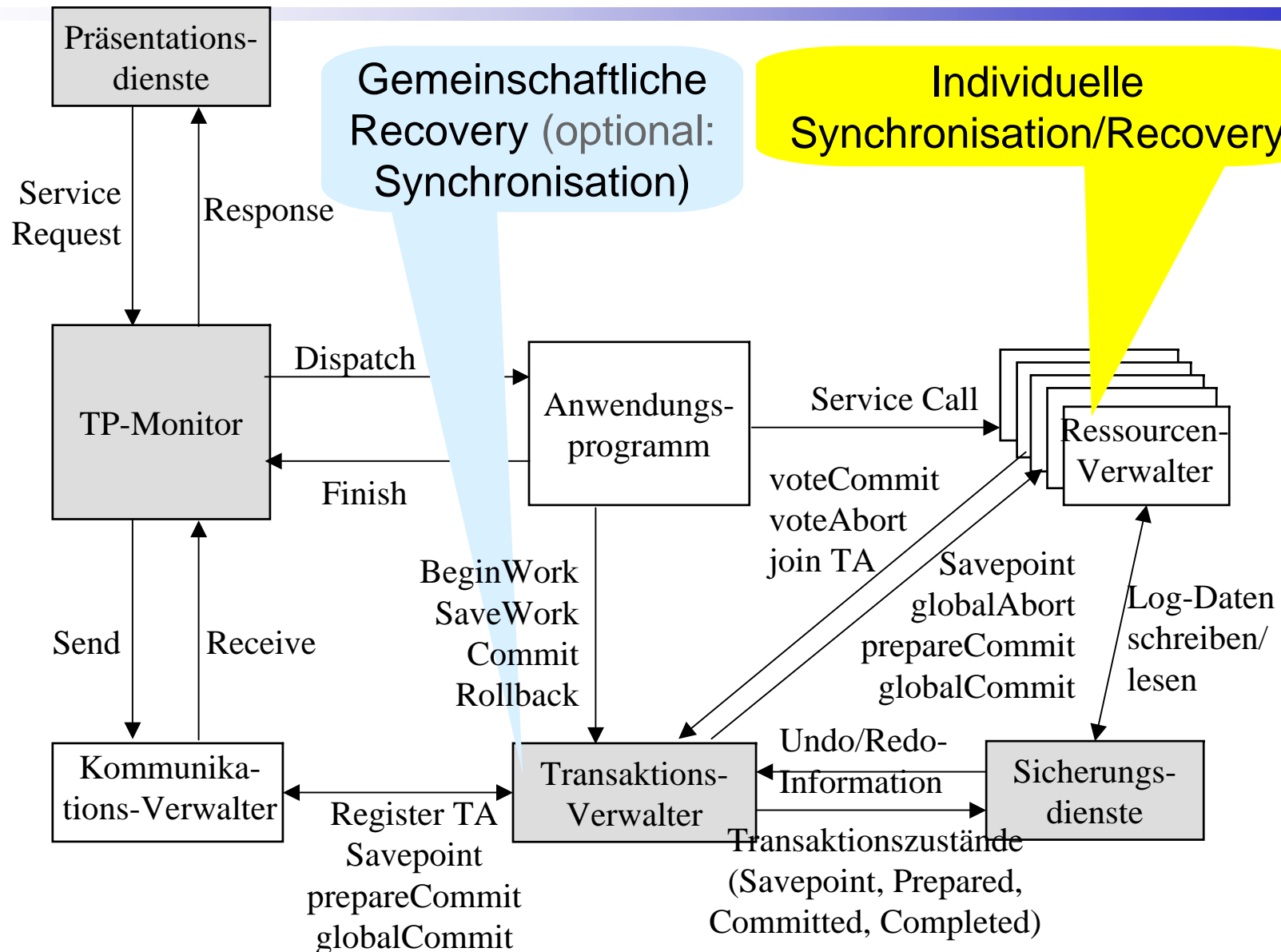
- ◆ transaction has executed at n nodes ($n > 1$),
 m nodes with read-only subtransactions ($m < n$)

	general	Example 1 ($n=2, m=0$)	Example 2 ($n=10, m=5$)
1PC	$2(n-1)$	2	18
Linear 2PC	$2n-1$	3	19
Centralized/ hierarchical 2PC	$4(n-1)-2m$	4	26
3PC	$6(n-1)-4m$	6	34

Also more log
writes ($3n$)

Architectures

Transaktionsverwaltungs-Monitore



X/Open-Modell eines verteilten Transaktionssystems

124

