Distributed Database Systems and Data Warehouses

Dr. Volodymyr Sokol
(vlad.sokol@gmail.com)

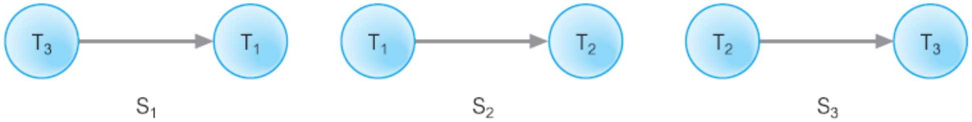
LECTION 5

Distributed Deadlock Management

- Consider three transactions T1, T2, and T3 with:
  - T1 initiated at site S1 and creating an agent at site S2
  - T2 initiated at site S2 and creating an agent at site S3
  - T3 initiated at site S3 and creating an agent at site S1
- The transactions set shared (read) and exclusive (write) locks as illustrated below, where read_lock(T_i, x_j) denotes a shared lock by transaction T_i on data item x_j and write_lock(T_i, x_j) denotes an exclusive lock by transaction T_i on data item x_j.

Time	S_1	S_2	S_3
t_1	read_lock(T_1, x_1)	write_lock(T_2, y_2)	read_lock(T_3, z_3)
t_2	write_lock(T_1, y_1)	write_lock(T_2, z_2)	
t_3	write_lock(T_3, x_1)	write_lock(T_1, y_2)	write_lock(T_2, z_3)

- We can construct the wait-for graphs (WFGs) for each site, and there are no cycles in the individual WFGs, which might lead us to believe that deadlock does not exist
- However, if we combine the WFGs, we can see that deadlock does exist due to the cycle: T1 → T2 → T3 → T1



Distributed Deadlock Management

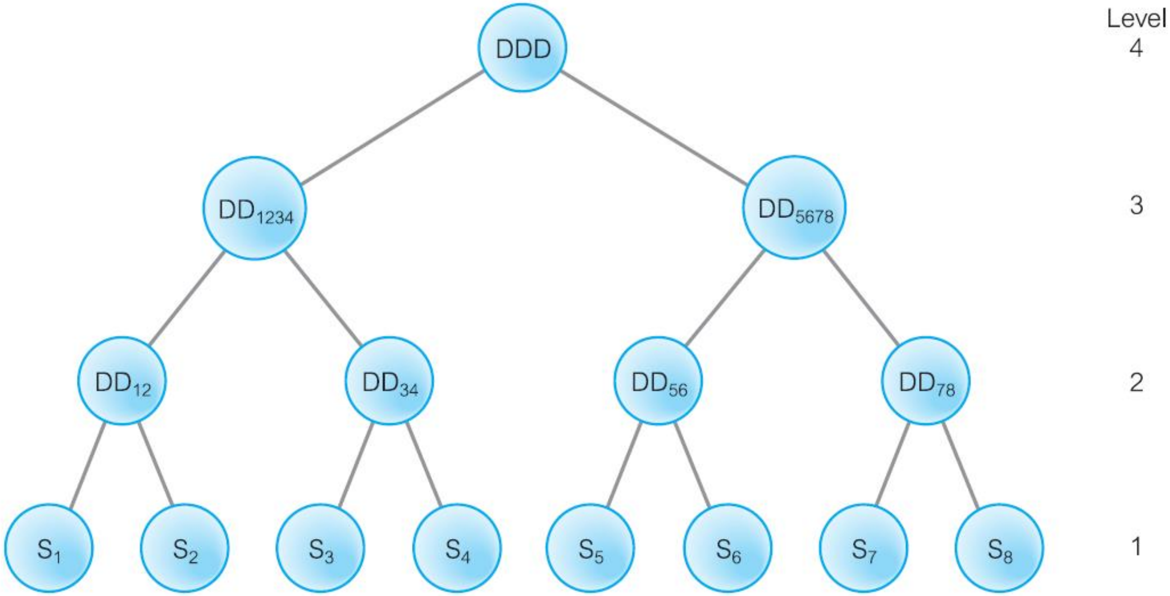
- There are three common methods for handling deadlock detection in DDBMSs
  - **centralized**
  - **hierarchical**
  - **distributed** deadlock detection.

Centralized deadlock detection

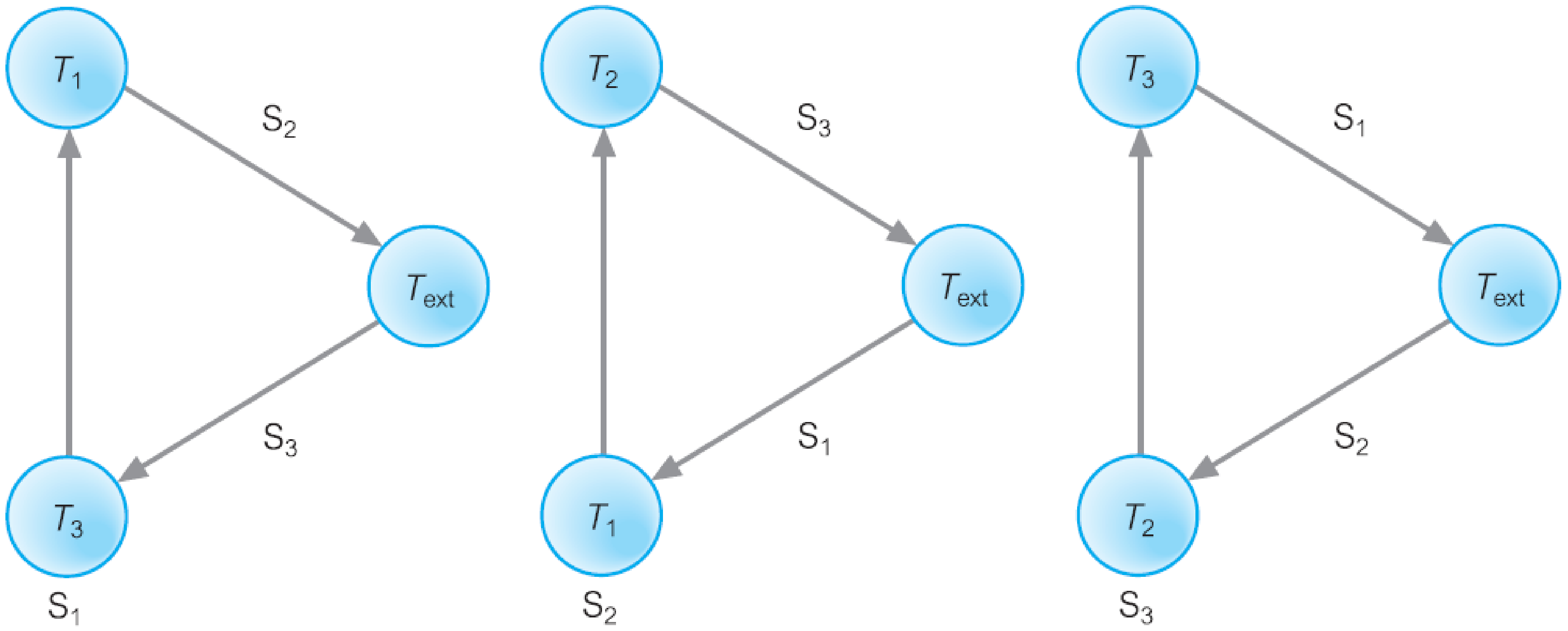
- Single site is appointed as the Deadlock Detection Coordinator (DDC)
- The DDC has the responsibility of constructing and maintaining the global WFG
- Periodically, each lock manager transmits its local WFG to the DDC. The DDC builds the global WFG and checks for cycles in it
- If one or more cycles exist, the DDC must break each cycle by selecting the transactions to be rolled back and restarted. The DDC must inform all sites that are involved in the processing of these transactions that they are to be rolled back and restarted.

Hierarchical deadlock detection

- With hierarchical deadlock detection, the sites in the network are organized into a hierarchy.
- Each site sends its local WFG to the deadlock detection site above it in the hierarchy



Distributed deadlock detection



Distributed Deadlock Detection

$S_1: T_{ext} \rightarrow T_3 \rightarrow T_1 \rightarrow T_{ext}$

$S_2: T_{ext} \rightarrow T_1 \rightarrow T_2 \rightarrow T_{ext}$

$S_3: T_{ext} \rightarrow T_2 \rightarrow T_3 \rightarrow T_{ext}$

- Transmit LWFG for S_1 to the site for which transaction T_1 is waiting, site S_2.
- LWFG at S_2 is extended and becomes:

$S_2: T_{ext} \rightarrow T_3 \rightarrow T_1 \rightarrow T_2 \rightarrow T_{ext}$

Distributed Deadlock Detection

- Still contains potential deadlock, so transmit this WFG to S_3:

$$S_3: T_{ext} \rightarrow T_3 \rightarrow T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_{ext}$$

- GWFG contains cycle not involving T_{ext}, so deadlock exists.

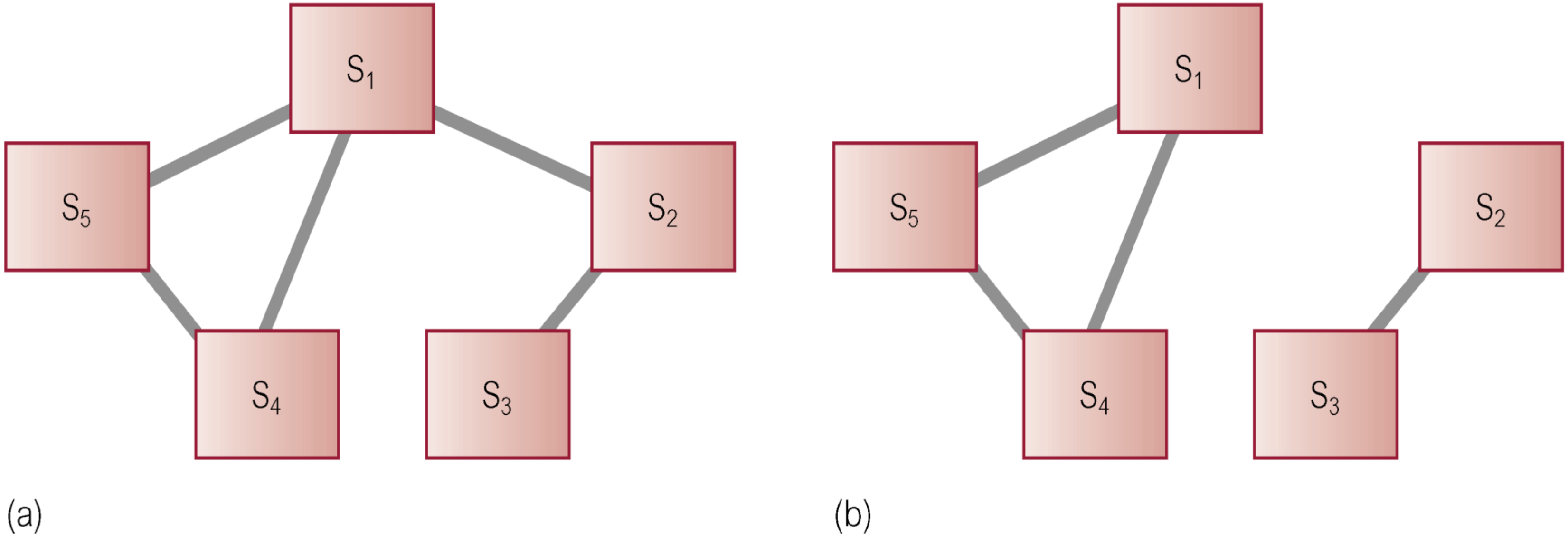
Distributed Deadlock Detection

- Four types of failure particular to distributed systems:
  - Loss of a message.
  - Failure of a communication link.
  - Failure of a site.
  - Network partitioning.
- Assume first are handled transparently by DC component.

Distributed Recovery Control

- DDBMS is highly dependent on ability of all sites to be able to communicate reliably with one another.
- Communication failures can result in network becoming split into two or more partitions.
- May be difficult to distinguish whether communication link or site has failed.

Partitioning of a network



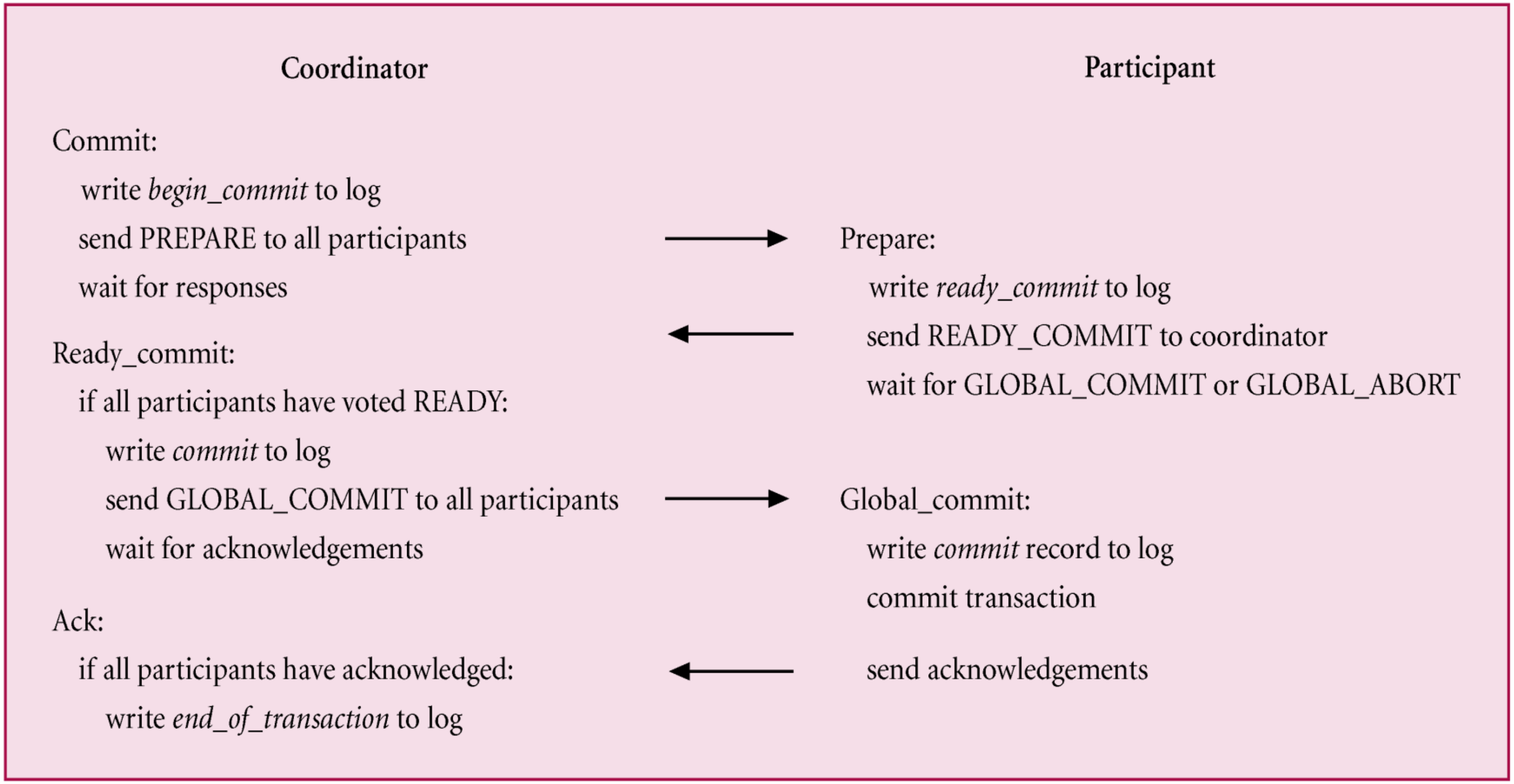
Two-Phase Commit (2PC)

- Two phases: a *voting phase* and a *decision phase*.
- Coordinator asks all participants whether they are prepared to commit transaction.
  - If one participant votes abort, or fails to respond within a timeout period, coordinator instructs all participants to abort transaction.
  - If all vote commit, coordinator instructs all participants to commit.
- All participants must adopt global decision.

Two-Phase Commit (2PC)

- If participant votes abort, free to abort transaction immediately
- If participant votes commit, must wait for coordinator to broadcast global-commit or global-abort message.
- Protocol assumes each site has its own local log and can rollback or commit transaction reliably.
- If participant fails to vote, abort is assumed.
- If participant gets no vote instruction from coordinator, can abort.

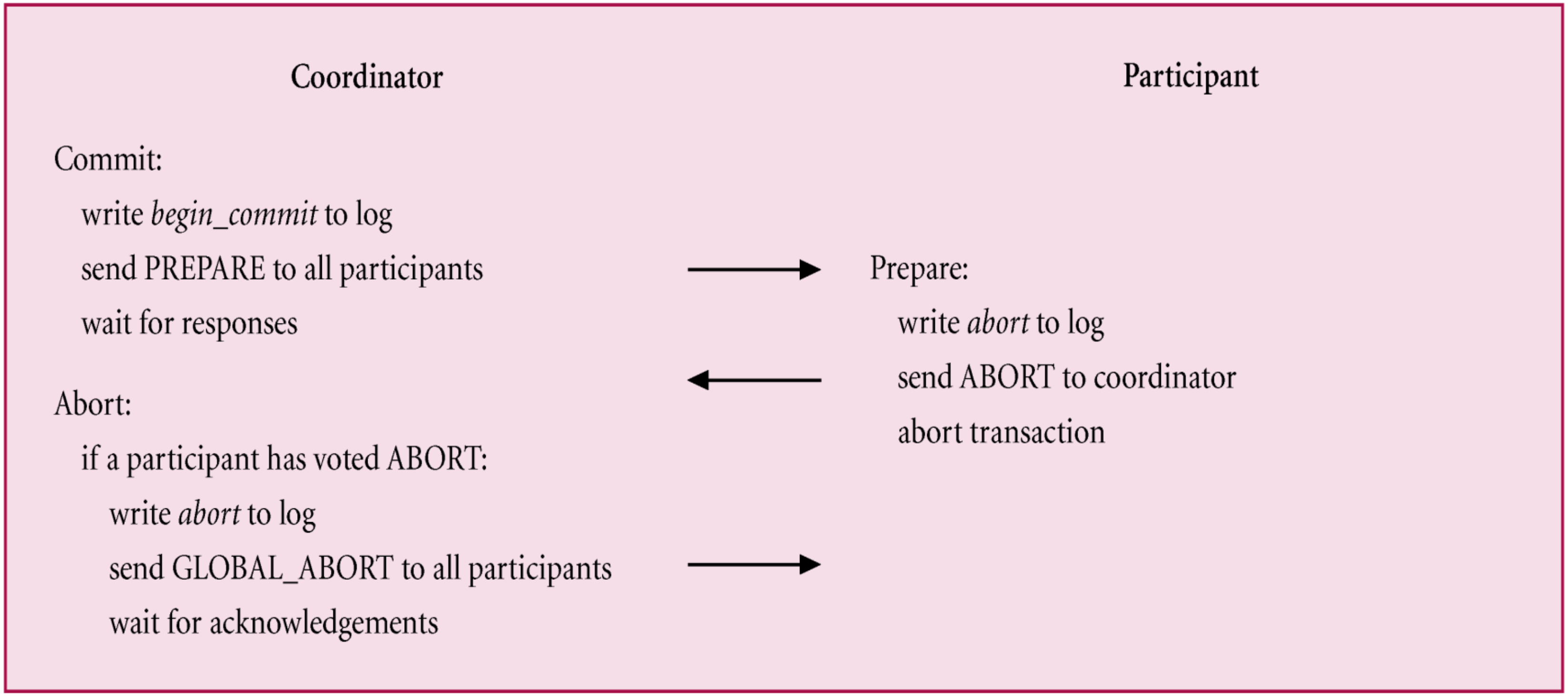
2PC Protocol for Participant Voting Commit



Coordinator		Participant
Commit:		
write *begin_commit* to log		
send PREPARE to all participants	→	Prepare:
wait for responses		write *ready_commit* to log
	←	send READY_COMMIT to coordinator
Ready_commit:		wait for GLOBAL_COMMIT or GLOBAL_ABORT
if all participants have voted READY:		
write *commit* to log		
send GLOBAL_COMMIT to all participants	→	Global_commit:
wait for acknowledgements		write *commit* record to log
		commit transaction
Ack:		
if all participants have acknowledged:	←	send acknowledgements
write *end_of_transaction* to log		

(a)

2PC Protocol for Participant Voting Abort



```
Coordinator                                              Participant

Commit:
  write *begin_commit* to log
  send PREPARE to all participants          ——→    Prepare:
  wait for responses                                  write *abort* to log
                                            ←——       send ABORT to coordinator
Abort:                                                abort transaction
  if a participant has voted ABORT:
    write *abort* to log
    send GLOBAL_ABORT to all participants   ——→
    wait for acknowledgements
```

(b)

2PC Termination Protocols

- Invoked whenever a coordinator or participant fails to receive an expected message and times out.

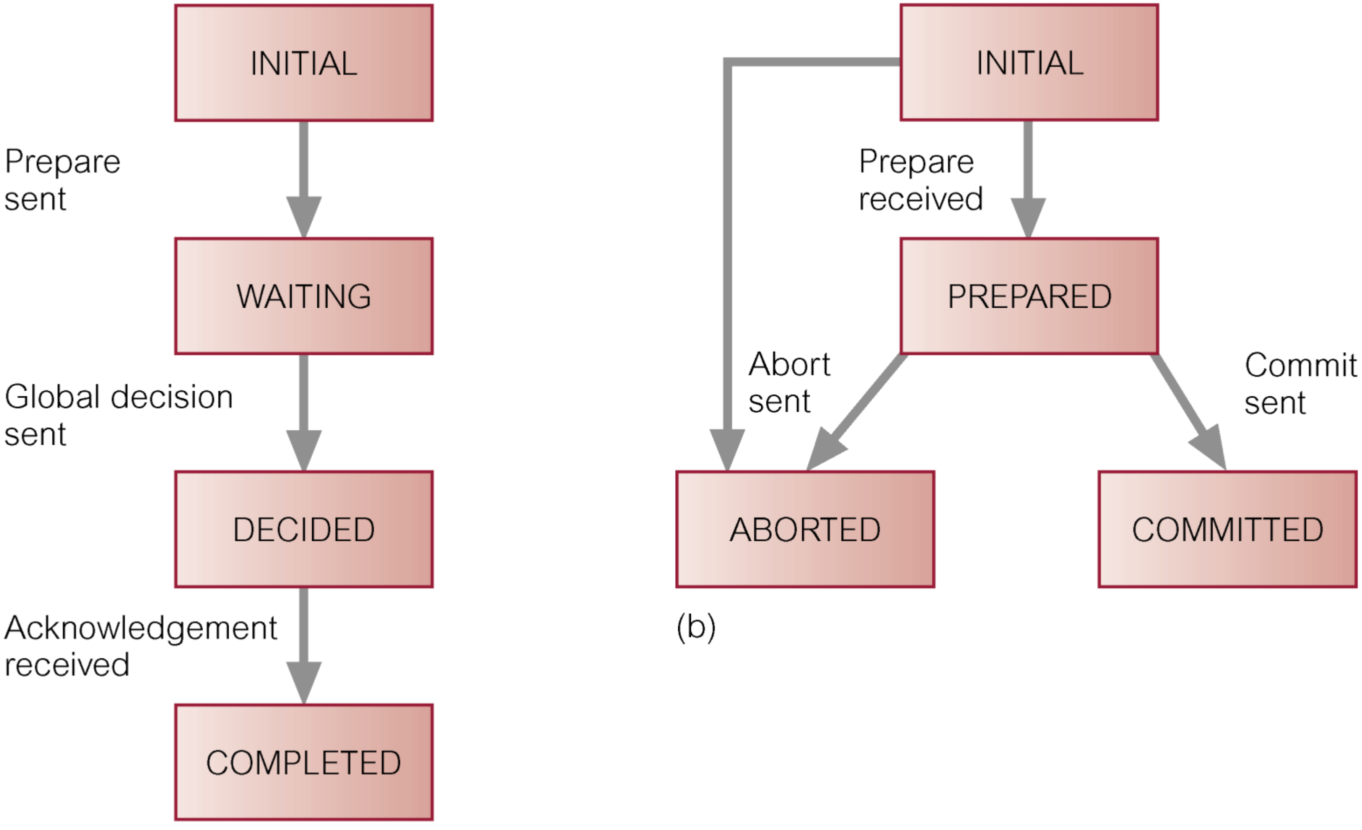
Coordinator

- Timeout in WAITING state
  - Globally abort transaction.
- Timeout in DECIDED state
  - Send global decision again to sites that have not acknowledged.

2PC - Termination Protocols (Participant)

- Simplest termination protocol is to leave participant blocked until communication with the coordinator is re-established. Alternatively:
- Timeout in INITIAL state
  - Unilaterally abort transaction.
- Timeout in the PREPARED state
  - Without more information, participant blocked.
  - Could get decision from another participant .

State Transition Diagram for 2PC



(b)

(a)

2PC Recovery Protocols

- Action to be taken by operational site in event of failure. Depends on what stage coordinator or participant had reached.

Coordinator Failure

- Failure in INITIAL state
  - Recovery starts commit procedure.
- Failure in WAITING state
  - Recovery restarts commit procedure.

2PC Recovery Protocols (Coordinator Failure)

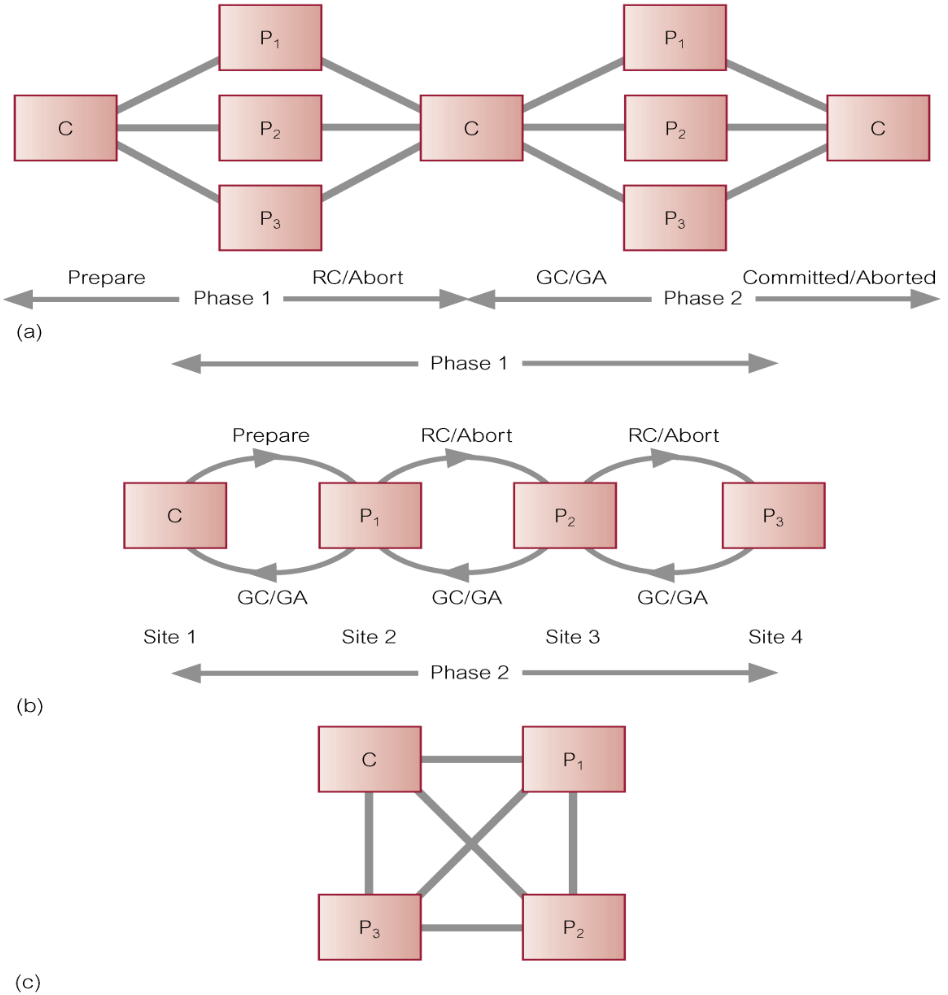
- Failure in DECIDED state
  - On restart, if coordinator has received all acknowledgements, it can complete successfully. Otherwise, has to initiate termination protocol discussed above.

2PC Recovery Protocols (Participant Failure)

- Objective to ensure that participant on restart performs same action as all other participants and that this restart can be performed independently.

- Failure in INITIAL state
  - Unilaterally abort transaction.
- Failure in PREPARED state
  - Recovery via termination protocol above.
- Failure in ABORTED/COMMITTED states
  - On restart, no further action is necessary.

2PC Topologies


P1
P2
P3
C
C
C
P1
P2
P3
Prepare
Phase 1
RC/Abort
GC/GA
Phase 2
Committed/Aborted
(a)
Phase 1
Prepare
RC/Abort
RC/Abort
C
P1
P2
P3
GC/GA
GC/GA
GC/GA
Site 1
Site 2
Site 3
Site 4
Phase 2
(b)
C
P1
P3
P2
(c)