Lecture 10 · SENG 41283 · Distributed and Cloud Computing

Distributed Transactions

Flat and Nested Transactions · The Coordinator · One-Phase Commit · Two-Phase Commit · Nested 2PC

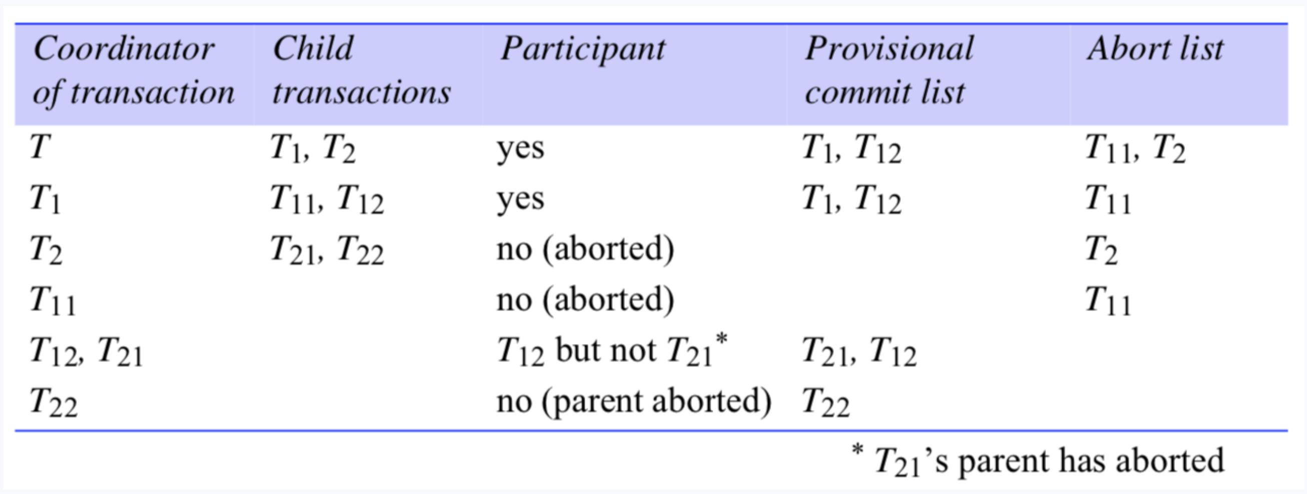
1. Flat and Nested Distributed Transactions

Flat Transaction

A client makes requests to more than one server, completing each request before going on to the next. Objects are accessed sequentially. When servers use locking, a transaction can only be waiting for one object at a time.

Nested Transaction

The top-level transaction can open subtransactions, each of which can open further subtransactions to any depth of nesting. Example: a client transfers £10 from A to C and £20 from B to D. Structured as four nested transactions (two deposits, two withdrawals), the four requests run in parallel, achieving better performance than a sequential flat transaction.



Coordinator of transaction	*Child transactions*	*Participant*	*Provisional commit list*	*Abort list*
T	T_1, T_2	yes	T_1, T_{12}	T_{11}, T_2
T_1	T_{11}, T_{12}	yes	T_1, T_{12}	T_{11}
T_2	T_{21}, T_{22}	no (aborted)		T_2
T_{11}		no (aborted)		T_{11}
T_{12}, T_{21}		T_{12} but not T_{21}*	T_{21}, T_{12}	
T_{22}		no (parent aborted)	T_{22}	

* T_{21}'s parent has aborted

Figure 1.1 Nested transactions: four subtransactions (T1, T2, T3, T4) running in parallel inside top-level transaction T.

2. The Coordinator

A client starts a transaction by sending an openTransaction request to a coordinator. The coordinator returns a unique Transaction Identifier (TID) comprising: (1) identifier of the creating server (e.g. IP address); (2) a number unique to that server. The coordinator is responsible for committing or aborting the transaction at the end.

- Each server managing an accessed object is a **participant** that tracks all recoverable objects involved in the transaction.
- The **join** operation is used whenever a new participant joins; the coordinator records it.
- A participant may call abortTransaction on the coordinator if it cannot continue.

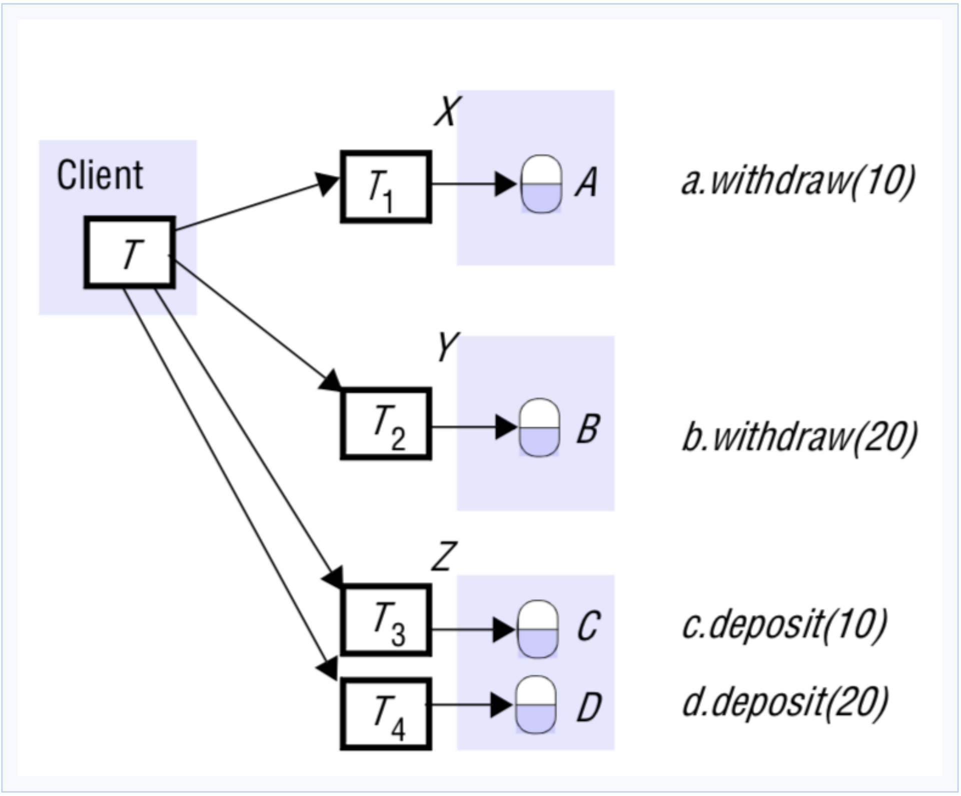


Figure 2.1 Coordinator of a distributed transaction: coordinator, participants, and the join protocol.

3. Atomic Commit Protocols

Atomicity requires: when the transaction ends, EITHER all operations are carried out OR none of them.

3.1 One-Phase Atomic Commit (1PC)

The coordinator communicates the commit or abort decision to all participants and repeats until all acknowledge. Does not allow a server to make a unilateral decision to abort when the client requests a commit. Fails when: deadlock resolution silently aborts a transaction; validation fails in optimistic concurrency control; or a server crashes and is replaced.

3.2 Two-Phase Commit Protocol (2PC)

Designed to allow any participant to abort its part. Before voting Yes, a participant saves all altered objects to permanent storage with status ‘prepared’ — it will eventually be able to commit even if it crashes and is replaced.

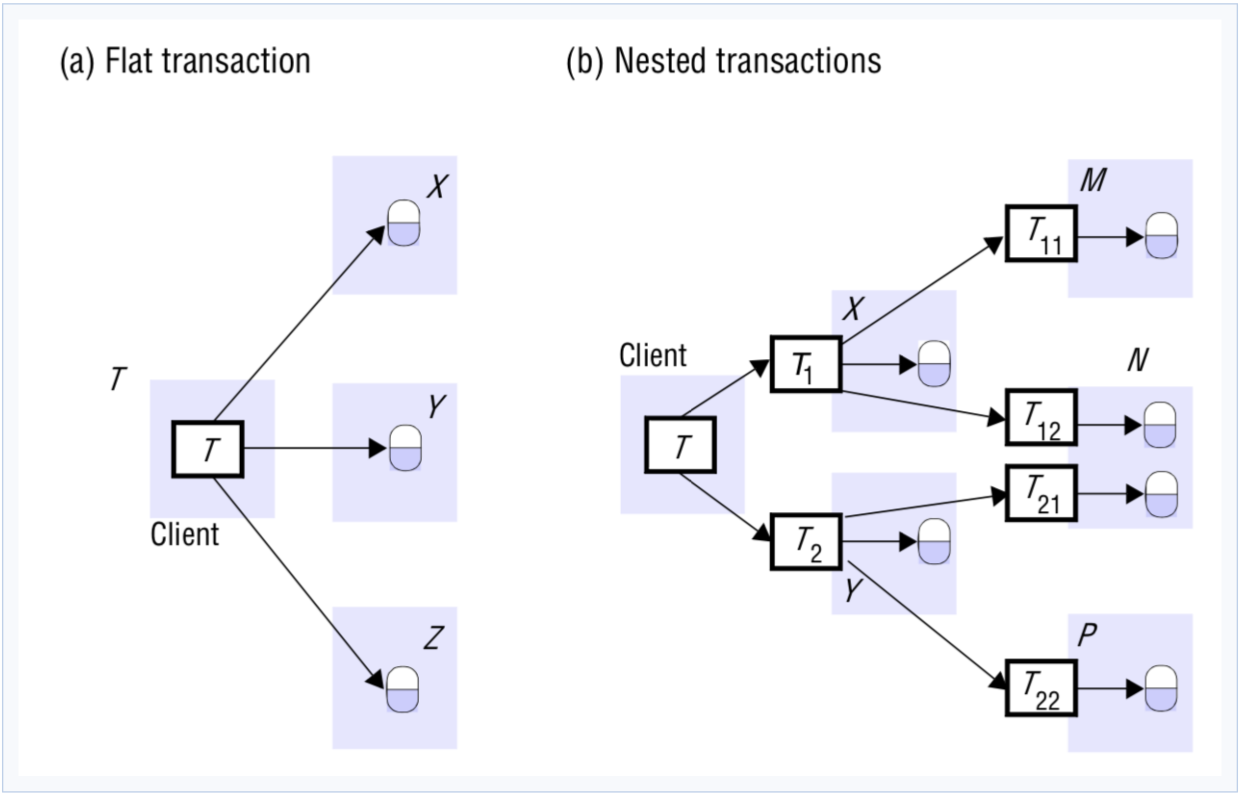


Figure 3.1 2PC: Phase 1 (voting) and Phase 2 (completion) message flow between coordinator and participants.

Phase	Step	Description
Phase 1 (Voting)	1	Coordinator sends canCommit?(trans) to each participant.
Phase 1 (Voting)	2	Each participant replies: Yes (saves objects to permanent storage) or No (aborts immediately).
Phase 2 (Completion)	3a	All Yes: coordinator sends doCommit(trans) to each participant.
Phase 2 (Completion)	3b	Any No: coordinator sends doAbort(trans) to all participants that voted Yes.
Phase 2 (Completion)	4	Participants that voted Yes wait for doCommit or doAbort; on commit, call haveCommitted.

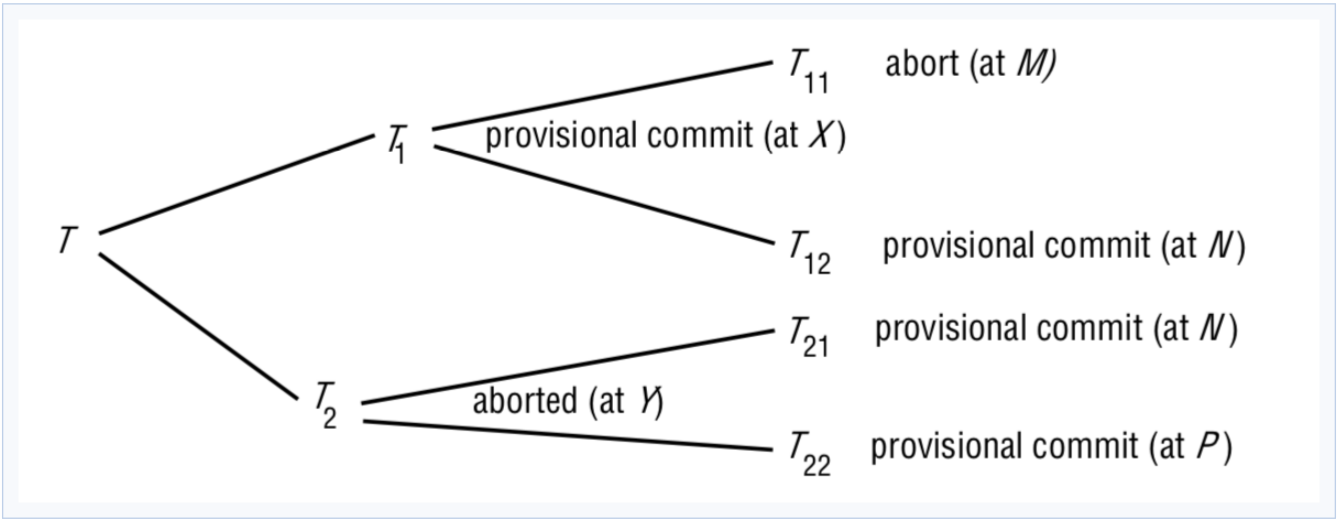

T
T1
provisional commit (at X)
T11
abort (at M)
T12
provisional commit (at N)
T2
aborted (at Y)
T21
provisional commit (at N)
T22
provisional commit (at P)

Figure 3.2 2PC protocol operations: canCommit?, doCommit, doAbort, haveCommitted, getDecision.

2PC Drawback — The Blocking Problem

A participant that voted Yes is in an uncertain state and cannot proceed until it hears from the coordinator. It cannot release the objects it holds. If the coordinator has failed, the participant cannot obtain the decision until the coordinator is replaced, causing **extensive delays**.

4. Two-Phase Commit for Nested Transactions

When a subtransaction completes, it either **provisionally commits** (nothing saved to permanent storage; commitment depends on ancestors) or **aborts**. After all subtransactions complete, the provisionally committed ones participate in a full 2PC. Subtransactions with an aborted ancestor must also abort.

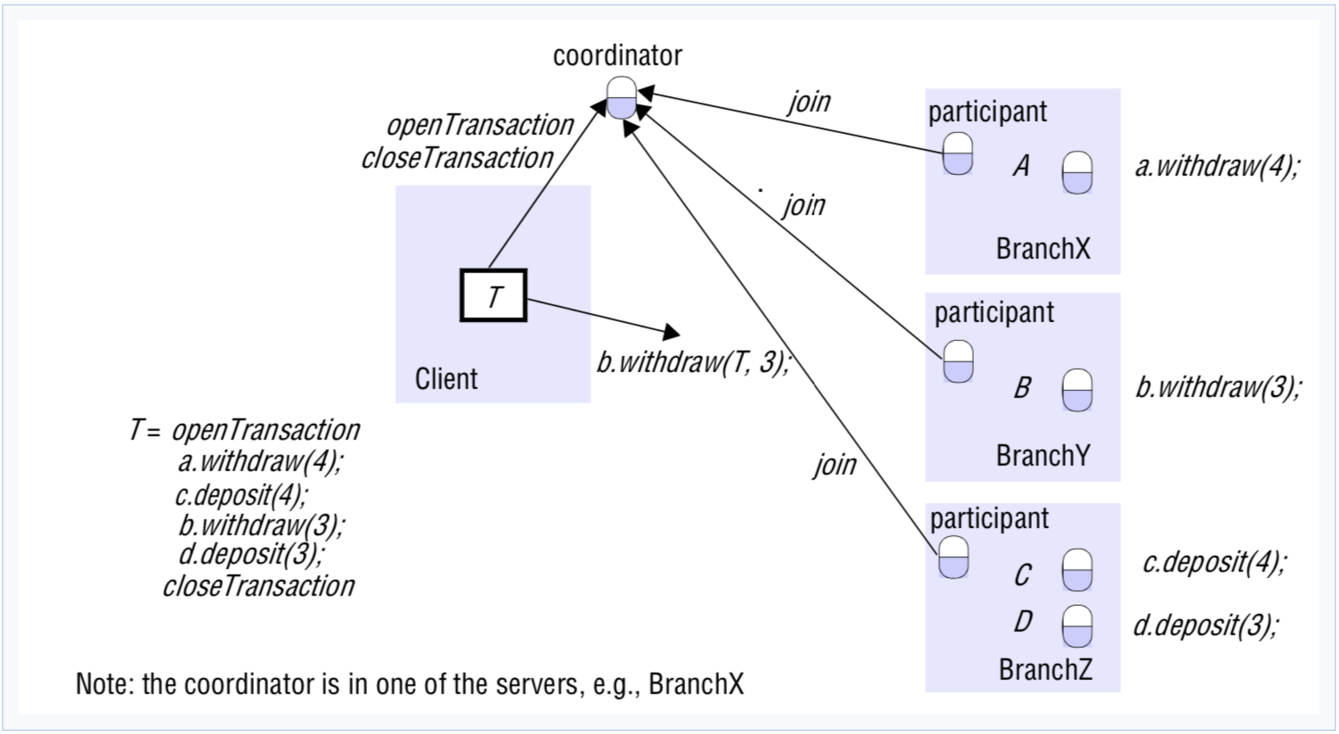


Figure 4.1 Nested 2PC: subtransaction tree showing provisional commits and the propagation of aborts.

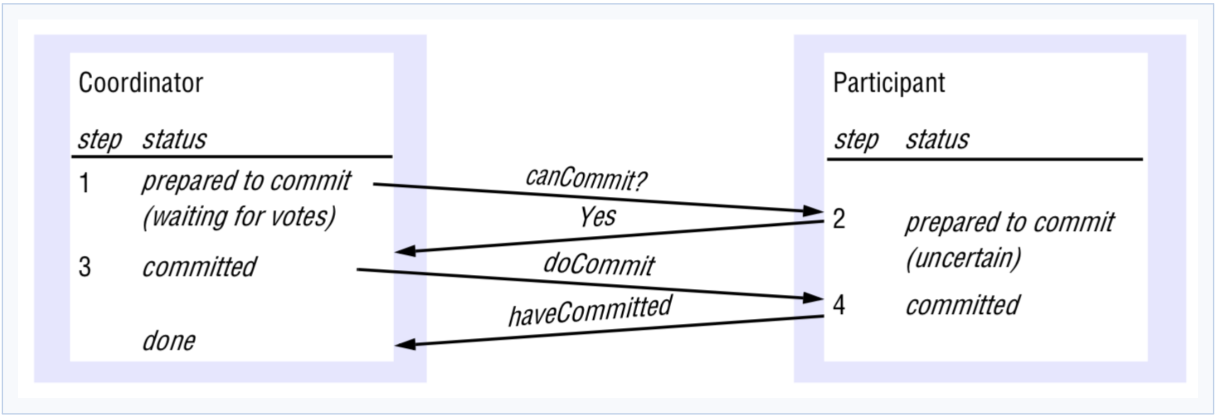


Figure 4.2 Subtransaction state transitions: open, provisionally committed, committed, and aborted.

A nested TID is an extension of its parent's TID, making the top-level TID determinable from any subtransaction TID. All subtransaction TIDs must be globally unique.