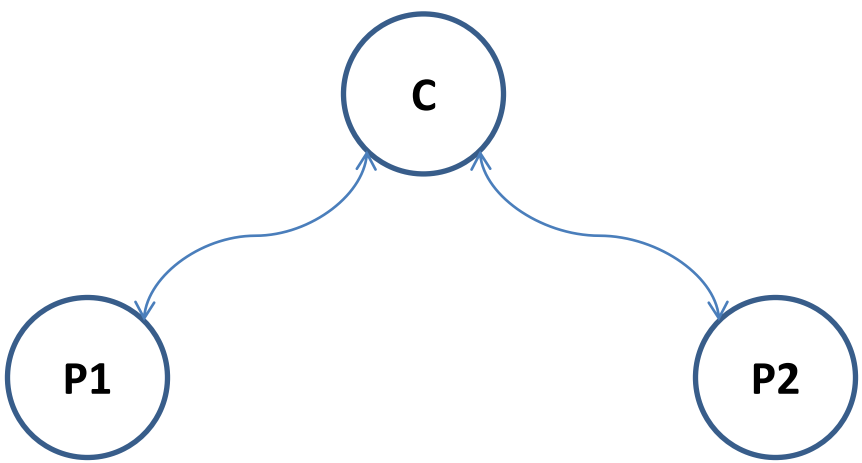
Assignment 5 - Solution

Problem 1

Consider a system that uses the two-phase commit protocol with the cooperative termination protocol and no other optimizations. Assuming there are two participants (P1 and P2) and a coordinator (C), for each of the following either describe an execution scenario or explain why it cannot happen:

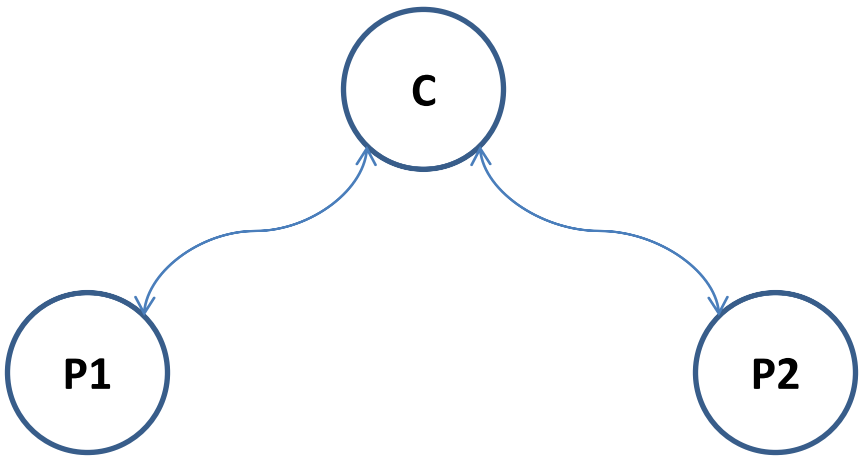


1.A P1 and P2 are blocked.

C sends “prepare-to-commit”

P1 & P2 send “yes-prepared”

C crashes, which leaves P1 and P2 blocked.



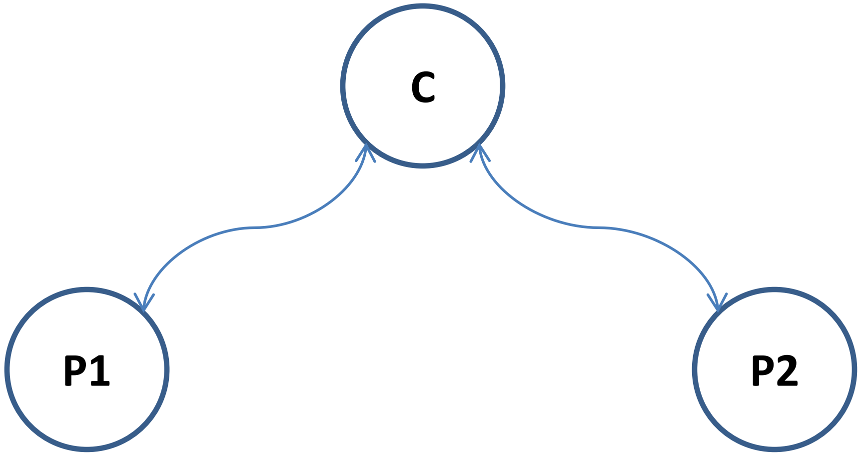
1.B. Only P2 is blocked.

1. C sends “prepare-to-commit”
2. P1 & P2 send “yes-prepared”
3. **Communication failure at P2**
   - P1 and C terminate the protocol
   - P1 and C crash
   - P2 is blocked

1. C sends “prepare-to-commit”
2. P1 & P2 send “yes-prepared”
3. **P2 crashes**
   - P1 and C terminate the protocol
   - P1 and C crash
   - P2 is recovers and is blocked

1.C C is blocked.

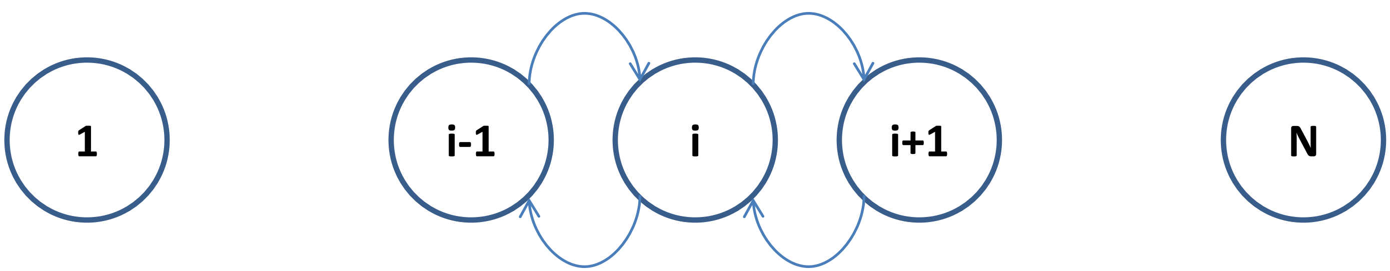
This cannot happen.
The coordinator can always unilaterally abort an undecided tx.



Problem 2

Suppose there are n processes involved in 2PC, where process 1 is the transaction's home. Suppose the processes are arranged in a chain (NOT a ring), so that each process can only communicate with adjacent processes in the chain.

That is, process 1 can communicate only with process 2, process n-1 can communicate only with process n, and for each i where $1<i<n$, process i can communicate only with processes i-1 and i+1.



2.A Devise a version of the 2PC protocol for this arrangement of processes that uses 2n – 2 messages to commit a transaction.

Process 1

Starts the commit activity.

Prepares, then sends a Request-to-Prepare to Process 2.

Process 2

Prepares, then sends a Request-to-Prepare to 3.

…

Process n

Receives a request to prepare

It commits

Sends a Commit message to *n*-1.

2.b. In the protocol you devised in (a), is there any process that is never in an uncertainty period?

Yes, Process *n*.

2.C In the protocol you devised in (a), what action commits the transaction?

The log write of a commit record at process *n* effectively commits the transaction.

2.D Explain how to modify the protocol to speed up the protocol in the event that a process votes No.

A process that votes No
should send Abort to its two neighbors (if present), including lower and higher numbered processes.

The lower-numbered neighbor should propagate the abort back toward process 1.

The higher-numbered neighbor should propagate the abort up toward process *n*.