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:

a. **P1 and P2 are blocked.**
C sends "prepare-to-commit"
P1 & P2 send "yes-prepared"
C crashes, which leaves P1 and P2 blocked.

b. **Only P2 is blocked.**
C sends "prepare-to-commit"
P1 & P2 send "yes-prepared"
Communication failure at P2 makes P2 blocked while P1 and C terminate the protocol and crash, which leaves P2 blocked.

Or replace the last line by:
P2 crashes
C and P1 commit.
C and P1 crash
P2 recovers, but is blocked

c. **C is blocked.**
This cannot happen since coordinator can always unilaterally abort an undecided transaction.

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.

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. It prepares and then sends a Request-to-Prepare to Process 2. Then 2 prepares and sends a Request-to-Prepare to 3. And so forth. When Process n receives a request to prepare, it commits and sends a Commit message to *n*-1, which commits and sends a Commit message to *n*-2, and so forth.

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

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.

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*.