# ECE428 Homework 4

Due: 11:59 p.m. on Wednesday 27th, 2022

This assignment has 4 questions with 60 points in total. The solutions must be typed, and submitted via Blackboard. However, the diagrams can be hand-drawn. You must acknowledge any sources used to arrive at your solutions, other than the course materials and textbook. All homework assignments are expected to be an individual work, so no collaborations are allowed.

**Question 1: Transactions in Distributed System** *[25 points]*

Consider the following interleaving of three transactions T1, T2 and T3, where the reads and writes to variables are explicitly labeled. The lowercase variables are local to each transaction.

| | **T1** | **T2** | **T3** |
|---|---|---|---|
| 1: | x = read(A) | | |
| 2: | | z = read(D) | |
| 3: | | | w = read(C) |
| 4: | | | write(B, w+2) |
| 5: | y = read(B) | | |
| 6: | write(C, x+y) | | |
| 7: | | write(A, z+1) | |
| 8: | | write(E, z-1) | |
| 9: | | | write(D, w-2) |

(a) (3 points) Identify all conflicts among the interleaved transactions. For instance, you can specify the pairs of operation numbers.

(b) (2 points) Consider the following interleaving of just two transactions T1 and T2 that follows the same ordering as the full interleaving shown above, i.e.:

| **T1** | **T2** |
|---|---|
| x = read(A) | z = read(D) |
| y = read(B) | |
| write(C, x+y) | |
| | write(A, z+1) |
| | write(E, z+1) |

Is this interleaving serially equivalent, and why or why not?

(c) (3 points) Consider now part (b) with interleaving just T1 and T3, and then also with interleaving just T2 and T3. Are these other interleavings serially equivalent, and again, why or why not?

(d) (3 points) Let the local variables be initialized to A = 1, B = 2, C = 3, D = 4, and E = 5. What are the final values of these variables after running T1, T2, T3 after the full 3-transaction interleaving given in part (a)?

(e) (3 points) Now consider all six possible serial interleavings of T1, T2, and T3,i.e., T1-T2-T3, T2-T3-T1, T1-T3-T2 and so forth. For each such interleaving, compute the final value of variables A, B, C, D, and E with the same initial values as in part (d).

(f) (3 points) Explain how you can tell that a given interleaving is not serially equivalent without doing the computations as you did in the previous parts above.

(g) (2 points) How could the serially non-equivalent interleaving be prevented using a strict two-phase locking with reader/writer locks?

(h) (2 points) How could the serially non-equivalent interleaving be prevented using timestamps concurrency, if T1, T2, T3 have timestamps 1, 2, and 3, respectively?

(i) (4 points) Write down a serially equivalent execution of T1, T2, T3 where all three transactions overlap, i.e., there is a point in time when each of T1, T2, and T3 have executed at least one operation, but none of the transactions have yet completed. Explain why it guarantees to be serially equivalent.

**Question 2: Bitcoin** *[10 points]*

Consider a Bitcoin network with N = 100 nodes. Let us model the propagation of a newly mined block assuming a simplified model that disregards several complexities of the actual protocol. Thus, at time t, there are $N_t$ nodes that have a copy of the block, and $N_0 = 1$. Each node picks at random another node to send the block to (in a real Bitcoin network, the nodes only send blocks to random neighbors). The node already has the block with the probability $(N_t - 1)/(N - 1)$, so it does not have the block with the probability $(N - N_t)/(N - 1)$. Therefore, the expected number of nodes that receive the block are $N_t(N - N_t)/(N - 1)$. This can be described using the recurrence:

$$N_{t+1} = \lfloor N_t + N_t(N - N_t)/(N - 1) \rfloor$$

(a) (3 points) Starting with $N_0 = 1$, how many rounds are required until all nodes receive the block?

(b) (5 points) Calculate the probability that a chain split occurs. In each round, each node which has not yet received the block will mine a conflicting block with the probability $1/(600 \times N)$. You can use a simulation to calculate this probability, in which case make sure to include the simulation code in your answer, and use enough trials to get the accurate estimate.

(c) (2 points) Unrelated to above, find a number n such that **`echo NETID n | sha256sum`** results in string with at least 5 leading zeros assuming your own NETID. For example,

```
$ echo nikita 90242 | sha256sum
00000b8556ab757a1a7a6a3ab4b43ff0045975e439593b98a8281f244ab4a772  -
```

**Question 3: Banking Transactions** *[7 points]*

Consider a bank processing financial transactions. There are two types of transactions: (i) DEPOSIT account amount, which adds the amount to the account balance, and (ii) WITHDRAW account amount, which subtracts the amount from the account balance. Each transaction also has a consistency check where the transaction is aborted, provided that at the end of the transaction any account balance would become negative.

Consider the following transactions:

T1: DEPOSIT A 30; DEPOSIT B 30; DEPOSIT C 50
T2: WITHDRAW A 10; DEPOSIT B 10; WITHDRAW C 60
T3: WITHDRAW A 30; WITHDRAW C 10
T4: WITHDRAW B 40; WITHDRAW C 10

(a) (4 points) If the transactions are executed serially in the order T1, T2, T3, then T4, which of them will be committed and which will be aborted?

(b) (3 points) What are the final balances of accounts A, B, and C?

**Question 4: More Transactions in Distributed Systems** *[18 points]*

Consider these two transactions:

T1: read A; write B; write A; read C; write E
T2: read C; write D; read A; read E; write B

(a) (4 points) Write a non-serial interleaving of T1 and T2 that would be feasible using the strict two-phase locking with reader/writer locks.

(b) (4 points) Write down a partial interleaving of T1 and T2 that would lead to a deadlock, if the strict two-phase locking with reader/writer locks are used. State what lock and in which mode is being requested by each transaction.

(c) (2 points) Write down interleaving of T1 and T2 that is serially equivalent, but otherwise impossible with the strict two-phase locking (assuming reader/writer locks). Explain, why it is impossible with the strict two-phase locking.

(d) (4 points) Write down a (potentially partial) interleaving of T1 and T2 that would cause T1 to be aborted, if time-stamped ordering were used. Assume that T1 and T2 have the transaction timestamps 1 and 2, respectively, show how the timestamps are updated. Explain, why the abort happens.

(e) (4 points) Write down a non-serial interleaving of T1 and T2 that could happen if the timestamped ordering were used, where both T1 and T2 successfully commit. T1 and T2 should have the transaction timestamps 1 and 2, respectively. Show how the timestamps are updated during the transaction execution.