Quiz 2 (75 Min.) Marks 75

General Instructions:

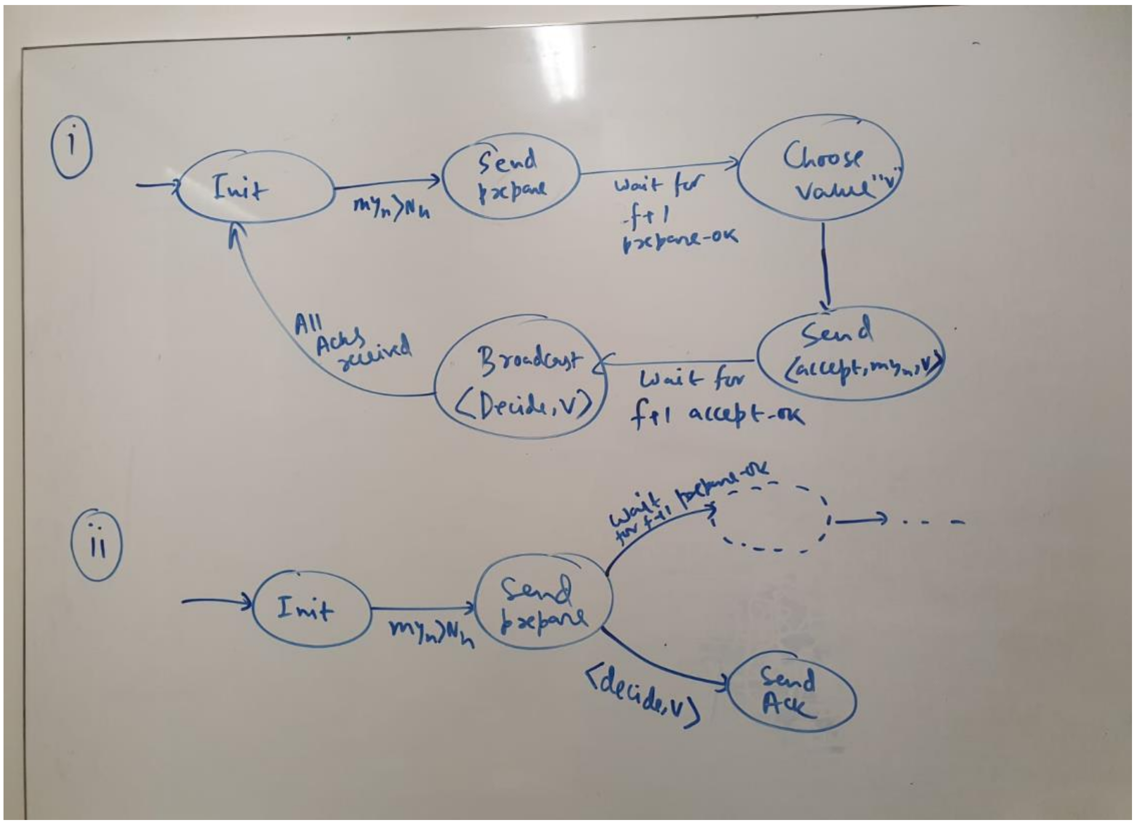
- ***This quiz is open book, open notes and you can use your electronic devices to refer to lecture notes, slides and likewise. However, usage of ChatGPT and other LLMs are strictly prohibited. If discovered violating this rule, the candidate will be disqualified from the quiz following SUTD regulations.***
- *If you are answering some of the questions (e.g., diagrams) by pen and paper, and not in the soft copy, then please put a note "**see hard copy**" at corresponding places in the soft copy.*
- *For each white paper used in the hard copy, write your name and student ID.*

(Paxos and Fault Tolerance) 40 Marks

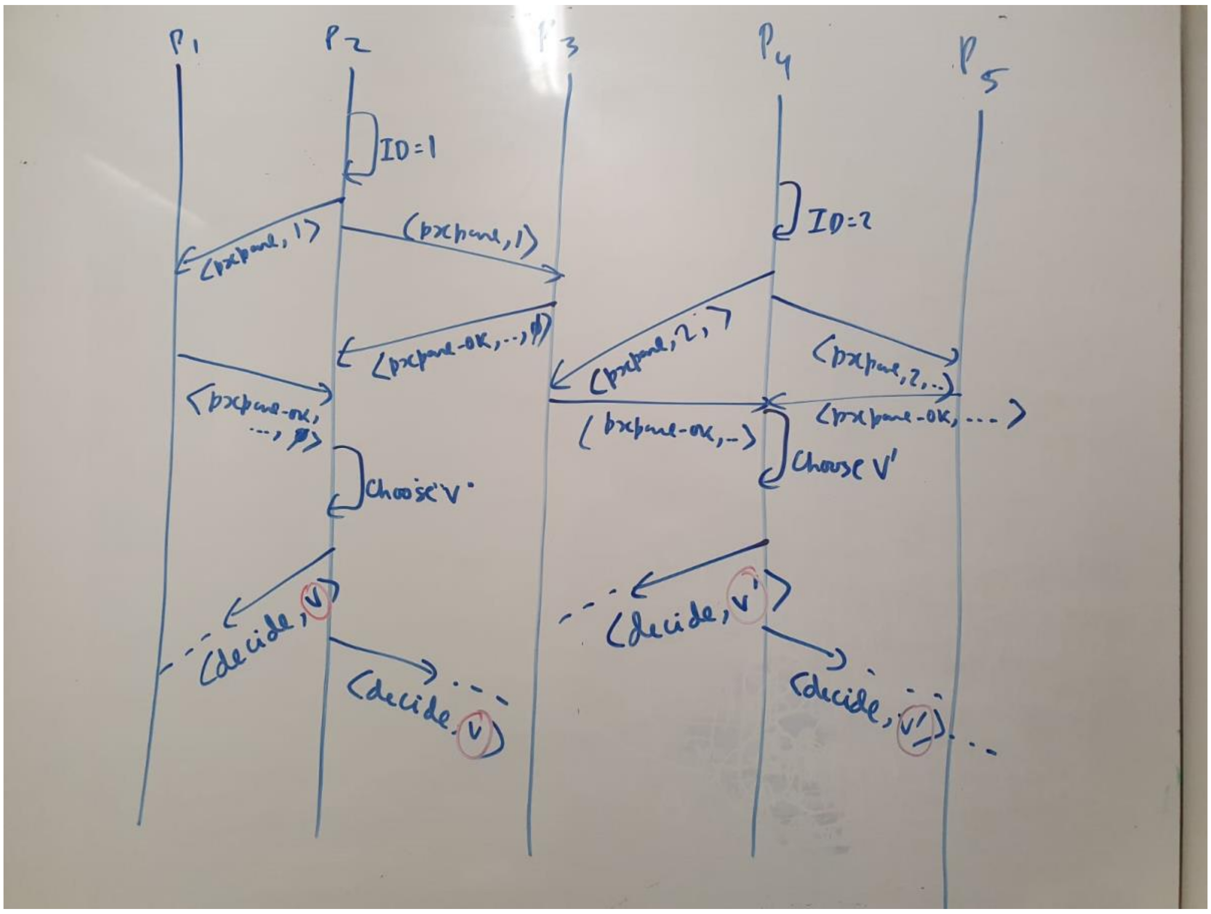
1. (a) Draw the state transition sequence for a Paxos node in the following scenarios: (i) the node becomes a proposer, eventually chooses its *own value*, gets majority acceptance and eventually gets acknowledgement from all learners that the value is received. (ii) the node becomes a proposer, sends <prepare> message, but receives a "decided" value (as part of the <decide, V> message) when waiting for the <prepare-OK>/<prepare-reject> messages.
(5 marks + 5 marks)

What's Important to Show:

➔ **State transition sequence is not sequence diagram. Nonetheless, I have not deducted marks even if a correct sequence diagram was provided.**
➔ **Need to show the value chosen.**
➔ **You may choose to show the "timeout" for a more complete state machine, however, that was not necessary for the above scenarios.**


i
Init
myn>Nh
Send prepare
wait for f+1 prepare-ok
Choose value "v"
Send <accept, myn, v>
wait for f+1 accept-ok
Broadcast <Decide, v>
All Acks received
ii
Init
myn>Nh
Send prepare
wait for f+1 prepare-ok
<decide, v>
Send Ack

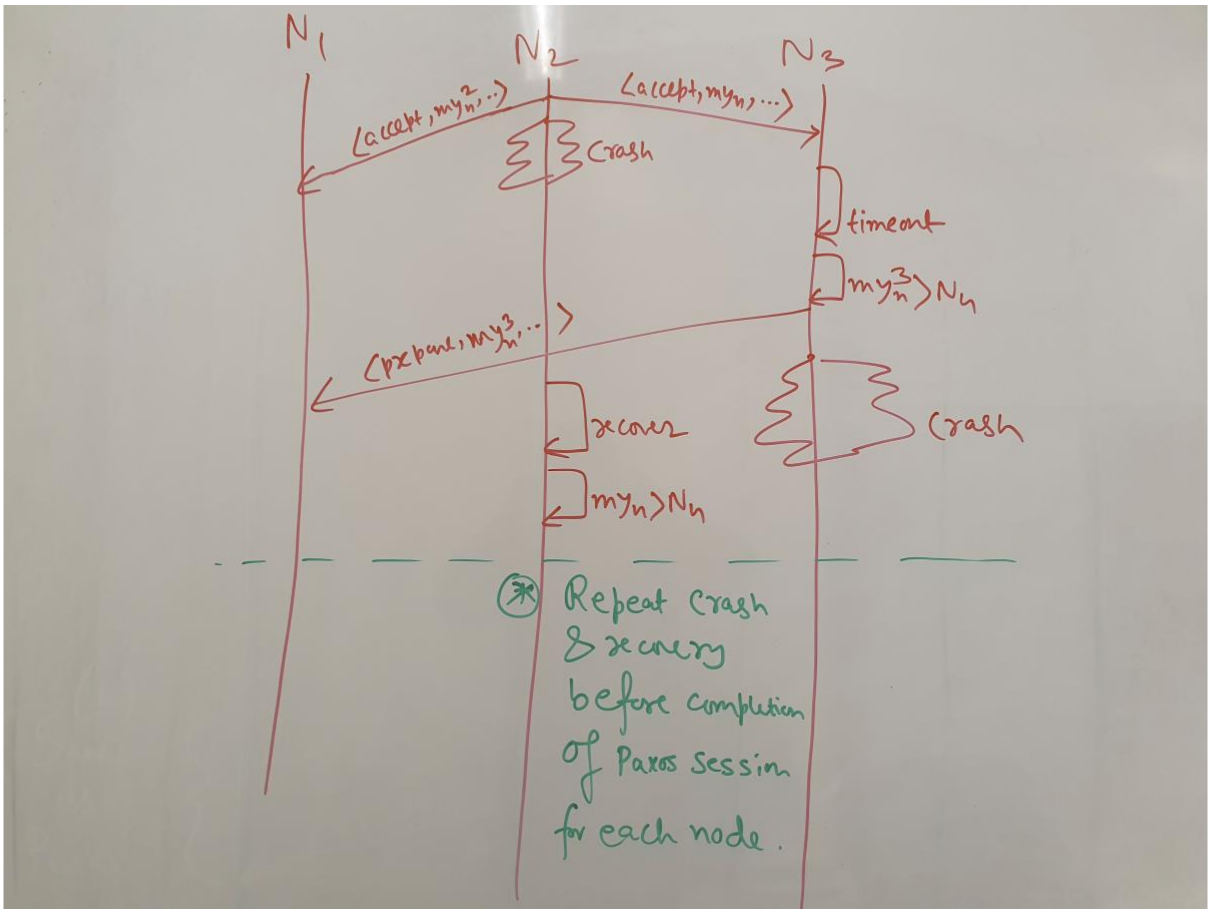
(b) Your friend says that there is no need for the three stages of Paxos as the value is already sent during the <prepare-OK> stage. Thus, the proposer can choose the value as per the original logic of Paxos and send <decide, “chosen value”> to all learners. Discuss whether your friend is correct. Concretely, show (e.g., via a sequence diagram or equivalent) whether the consensus property will be preserved by culling the acceptance stage altogether.
(10 marks)



Only some relevant messages are shown.

The friend is incorrect, as culling the acceptance stage may lead to violation of stable property and consensus.

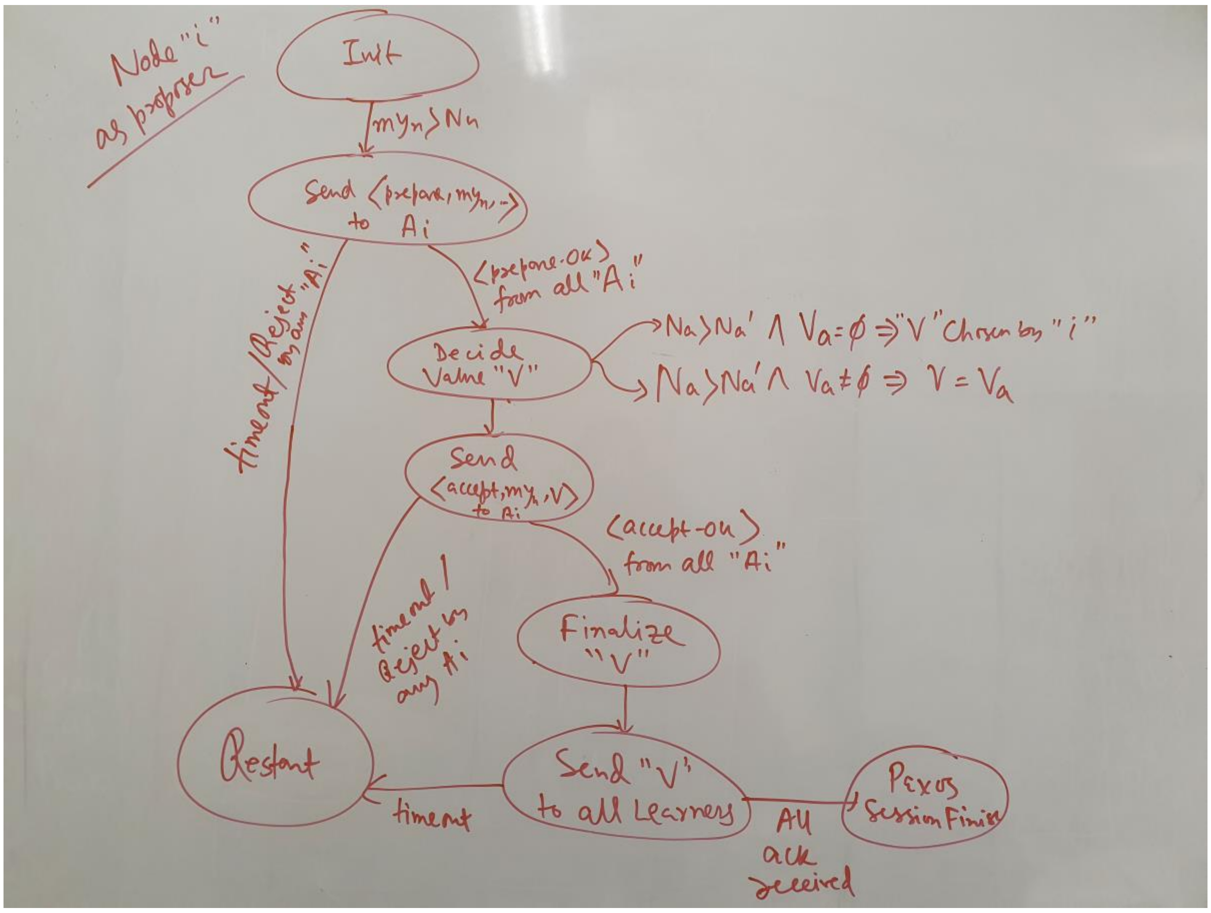
(c) Consider the design of Paxos fault tolerant protocol for N=3 nodes. Show a scenario where Paxos never terminates even though at most one node (i.e., less than half of total nodes) fails at a time. You may either draw a sequence diagram or discuss point by point the scenario that results in non-termination.
(5 marks)



(d) Reduce the coordinator election algorithm using Paxos. In other words, instantiate Paxos protocol to solve the coordinator election problem. Note: the generic answer on Paxos design will not award any marks. You need to clearly discuss how the coordinator election problem is solved with your approach (e.g., using a Paxos state machine customized to solve coordinator election).
(5 marks)

Coordinator Selection can simply be reduced to a Paxos session where the consensus needs to be reached on the Node ID representing coordinator. Thus, proposals can be sent by arbitrary nodes where Value can be set to the self-Node ID.

(e) Consider the design of Paxos fault tolerant protocol with 2N+1 nodes and usual assumptions (i.e., <=N failures, no byzantine failures). However, instead of assigning all nodes as acceptors, each node **Pi** maintains a Quorum of acceptors **Ai** where **Ai** is a strict subset of all nodes. Discuss how such Ai could be designed and draw the proposer state machine diagram such that the consensus is preserved.
(10 marks)



Node "i" as proposer
Init
my_n > N_a
Send <prepare, my_n, -> to Ai
<prepare-OK> from all "Ai"
timeout/Reject by any "Ai"
Decide Value "V"
N_a > N_a' ∧ V_a = ϕ ⇒ "V" chosen by "i"
N_a > N_a' ∧ V_a ≠ ϕ ⇒ V = V_a
Send <accept, my_n, V> to Ai
<accept-ok> from all "Ai"
timeout / Reject by any Ai
Finalize "V"
Restart
Send "V" to all Learners
timeout
All ack received
Paxos Session Finish

Ai and Aj need to be designed in a way such that they have at least one overlapping node.

The reasoning is similar to the general Paxos case. Here note that for two arbitrary nodes Pi and Pj, the acceptor nodes Ai and Aj must have an overlap. Therefore, if a proposal is accepted with ID “N” and a later proposal is accepted with ID “N+1” for acceptors Ai and Aj respectively, it guarantees that <prepare---OK...> messages sent by Aj and <accept> messages sent by Ai to have an overlap. Therefore, proposal ID “N+1” (and any subsequent proposal) will be accepted with the same value with which the proposal id “N” was accepted.

(Consistency Models) 35 Marks

2. Consider the following program executing in two different machines A and B:

Machine A:

```
x = 1;
y = 1;
if (y == 1)
    print(x)
if (x == 1)
    print(y)
```

Machine B:

```
y = 2;
x = 2;
if (x == 2)
    print(y);
if (y == 2)
    print(x)
```

Variables x and y are shared and initialized to zero. Furthermore, you can assume that each assignment, print and conditional checks are atomically performed.

With the aforementioned assumptions, answer the following questions:

a. Show what values of x and y would be printed under sequential consistency. List at least three possibilities and show the respective total order corresponding to the output.
(10 marks)

b. Show what values of x and y would be printed under total store order. List at least three possibilities (including at least two that do not appear in sequential consistency) and provide the rationale behind each output.
(10 marks)

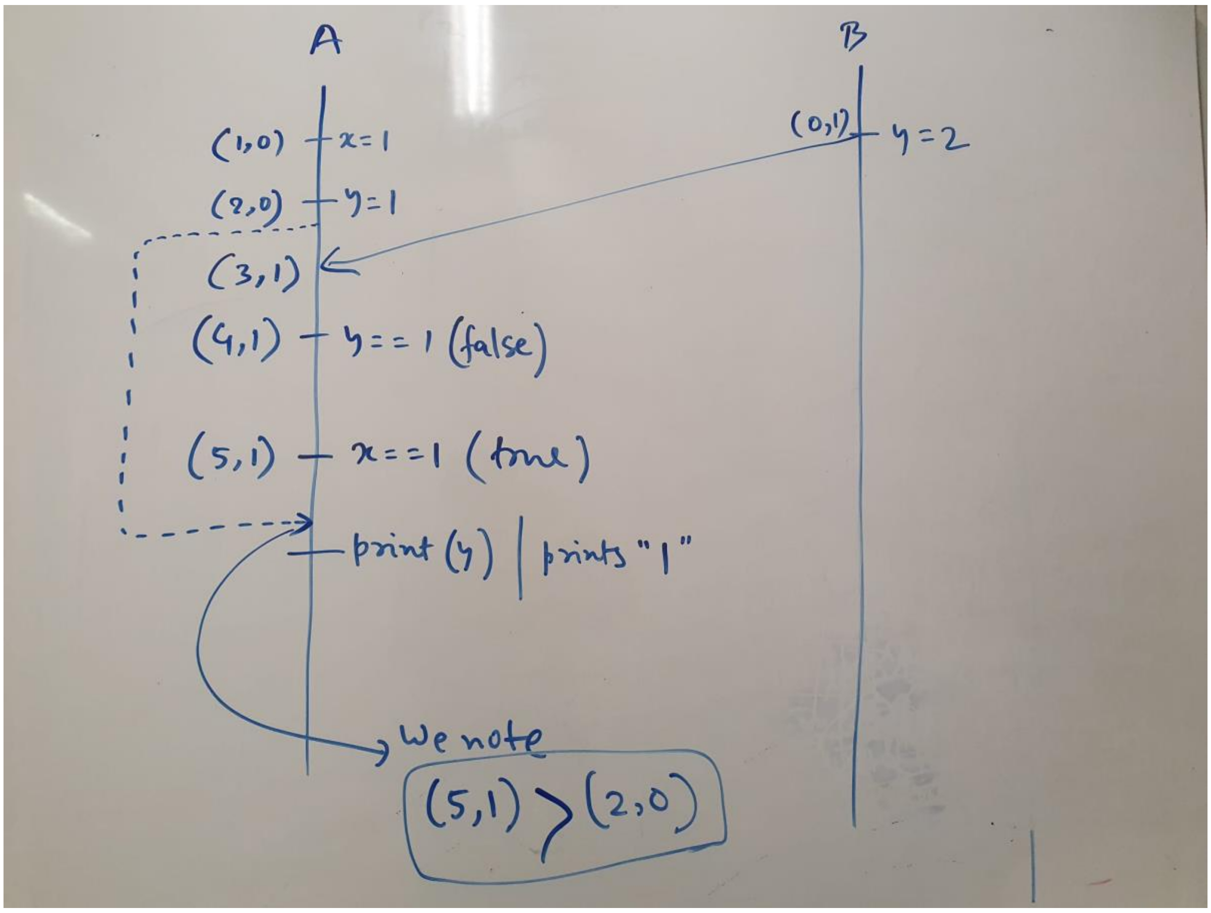
Sequential consistent:

x.A=1 -> y.A=1 -> y.B=2 -> x.B=2
x.A=1 -> y.B=2 -> y.A=1 -> x.A=2
y.B=2 -> x.B=2 -> x.A=1 -> y.A = 1

Total Store order:

y.A = 1 -> x.A = 1 -> y.B = 2 -> x.B =2 (not allowed in sequential)
y.A = 1 -> x.A = 1 -> x .B = 2 -> y.B = 2 (not allowed in sequential)
y.B = 2 -> x.B = 2 (nothing printed from machine A)

c. In the above example, show one possible print that will violate causal consistency. Use vector clocks to show that such a print indeed violates causal consistency.
(10 marks)



d. Assume that both Machine A and Machine B maintain a shared FIFO queue assigned to each shared variable. For example, Machine A and Machine B might maintain shared queues Qx and Qy attributed to shared variables x and y, respectively. All read and write requests to variables x and y are handled by these queues Qx and Qy, respectively. Argue whether such a design would preserve sequential consistency.
(5 marks)

Note: While writing the scenarios for (a) and (b), kindly show the order in which the statements in Machine A and Machine B are executed. No marks will be awarded otherwise.

Not Sequentially Consistent.

Write (X) -> Queue Qx
Write (Y) -> Queue Qy (this one is served before Qx, then you violate sequential consistency, as it violates program order).