**Assignment 3**

CS5352 Advanced Operating Systems Design Spring 2016

**Due Date: 4/13, 9 a.m., soft copy via Blackboard.**

**Late submissions are accepted till 4/18, 9 a.m., with 10% penalty each day. No submissions accepted after 4/18, 11 a.m.**

**Q1. (Chapter 6, Problem 2)** Consider the behavior of two machines in a distributed system. Both have clocks that are supposed to tick 1000 times per millisecond. One of them actually does, but the other ticks only 990 times per millisecond. If UTC updates come in once a minute, what is the maximum clock skew that will occur?

The second clock ticks 990,000 times per second. This gives an error of 10 msec/sec. In a minute this will grow to 600 msec. Technically it is 1% slower .01 x 60 = 600.

**Q2. (Chapter 6, Problem 5)** Add a new message to Fig. 6-9 that is concurrent with message m1, that is, it neither happens before m1 nor happens after m1.

A message from 1 to 2 or from 2 to 1 will satisfy the requirements. If it departs or arrives from 1 after 16 it will be ordered with respect to m1 ­so it must depart or arrive before 16. That leaves the possibilities as leaving process 2 at 0 and arriving at process 1 at 8 or leaving process 1 at 0 and arriving at process 2 at 10.

**Q3. (Chapter 6, Problem 10)** Consider Fig. 6-14. Suppose that the coordinator crashes. Does thisalways bring the system down? If not, under what circumstances does this happen?

If the algorithm is such that every request is answered immediately, either with permission or denial then so long as there is no process accessing resources and no processes queued a crash will not be fatal. The next process to request permission will fail to get any reply at all, and can initiate the election of a new coordinator. To make the system more robust we can store every message the coordinator receives on disk prior to sending a reply. This allows any reconstructed coordinator a list of accessed resources and queue by reading a file from the disk.

**Q4.** Please explain why the fields of “messages per entry/exit” and “delay before entry” are “1 to ∞” and “0 to n-1”, respectively, for the token ring algorithm.

With the token ring algorithm, the number is variable. When considering the messages per entry/exit, if every process constantly wants to enter a critical region, then each token pass will result in one entry and exit, for an average of one message per critical region entered. At the other extreme, the token may sometimes circulate for hours without anyone being interested in it. In this case, the number of messages per entry into a critical region is unbounded. In the case of the delay before entry, for the token ring, the time varies from 0 (token just arrived) to *n–*1 (token just departed) if it just departed it must circle the entire ring again.

**Q5. (Chapter 6, Problem 15)** In Fig. 6-21 we have two ELECTION messages circulating simultaneously. While it does no harm to have two of them, it would be more elegant if one could be killed off. Devise an algorithm for doing this without affecting the operation of the basic election algorithm.

When a process receives an election message, it checks to see who started it. If it itself started it, it turns the message into a coordinator message. If it did not start any election message, it adds its process number and forwards it along the ring. If it did send its own election message earlier and it has just discovered a competitor, it compares the originator’s process number with its own. If the other process has a lower number, it discards the message instead of passing it on. If the competitor is higher, the message is for- warded in the usual way. In this way, if multiple election messages are started, the one whose first entry is highest survives. The rest are killed off along the route.

**Q6. (Chapter 7, problem 2)** Explain in your own words what the main reason is for actually considering weak consistency models.

Weak consistency models come from the need to replicate performance. However, efficient replication can be done only if we can avoid global synchronizations, which, in turn, can be achieved by loosening consistency constraints.

**Q7. (Chapter 7, problem 6)** In Fig. 7-7, is 001110 a legal output for a sequentially consistent memory? Explain your answer.

Yes, if the processes run in order a, c, b, this result is obtained.

**Q8.** Please briefly describe the difference between the causal consistency model and sequential consistency model.

Sequential consistency means all processes will see all the events the same order. Causal consistency only guarantees that if event A causes event B, A will be seen before B in all processes but the total order of all events together may not be same.

**Q9. (Chapter 7, problem 17)** Consider a nonblocking primary-backup protocol used to guarantee sequential consistency in a distributed data store. Does such a data store always provide read-your-writes consistency?

No, as soon as the updating process receives an acknowledgment that its update is being processed, it may disconnect from the data store and reconnect to another replica. No guarantees are given that the update has already reached that replica. In contrast, with a blocking protocol, the updating process can disconnect only after its update has been fully propagated to the other replicas as well.

**Q10. (Chapter 7, problem 20)** A file is replicated on 10 servers. List all the combinations of read quorum and write quorum that are permitted by the voting algorithm.

The following possibilities of (read quorum, write quorum) are legal. (1,

10), (2, 9), (3, 8), (4, 7), (5, 6), (6, 5), (7, 4), (8, 3), (9, 2), and (10, 1).