Assignment 6

Alexandros Karakasidis

1.

It is undesirable to set the clock back because some processes might have already been executing (e.g. make creating an object file) and setting the clock back will cause inconsistencies. The correction for this fast clock can be achieved as following:

Seconds Elapsed	Clock Time Without Adjustment	Clock Time With Adjustment
0	10:27:54.0	10:27:54.0
1	10:27:55.0	10:27:54.5
2	10:27:56.0	10:27:55.0
3	10:27:57.0	10:27:55.5
4	10:27:58.0	10:27:56.0
5	10:27:59.0	10:27:56.5
6	10:28:00.0	10:27:57.0
7	10:28:01.0	10:27:57.5
8	10:28:02.0	10:27:58.0

The idea in this adjustment is that instead of increasing the timer, one second every time 1 second elapses, to increase to only 0.5 seconds, in order to achieve the desired value after 8 seconds.

2.

The client should choose the last entry of the table, since this exhibits the shortest delay. In specific, the average transfer time is 20/2 = 10 ms. so, the clock should be set to 10:54:28.342 + 0.10 = 10:54:28.352. Thus the accuracy is +/-10 ms.

The answer does not change in the case that the transfer time is 8ms. The thing that changes is the accuracy, which is now +/-2 ms.

3.

I will prove that the following relationship is true:

If e happened before e' and $k \geq 0$ messages exchanged between them, then L(e) < L(e').

- For k=0: e happened immediately before e' and no messages were exchanged. Obviously L(e) < L(e').
- I will suppose that if k=1, i.e. only 1 message exchanged, the relationship is valid, i.e. L(e) < L(k) <L(e').
- Given that k=n, (which means n messages exchanged), using the previous relationship, for each message let it be i of the n messages, the following relationship will be valid:

  $$L(e) < L(k_i) < L(e')$$

  Which proves the given relationship.

4.

Course slides (29) indicate the opposite!

5.

Acknowledgments have the sense that, upon reception, Lamport timestamps are increased by all receiving processes. Assuming that transmission is reliable, any messages can be multicasted without acknowledgements, given that the timestamp of each message is definitely greater than the preceding messages.

6.

In this case, the duplicate message can be removed using some kind of prioritization. In specific, when process having issued an *ELECTION* request, receives a request of the same kind, or of type *COORDINATOR*, originating from a process with lower id, it drops it. The process with the lower id which also issued the *ELECTION* request, will understand that its message was dropped, when it sees the same request from a process with a higher id. This algorithm is actually a merge between the ring and the bully algorithm.

7.

Let dt_s be the average synchronization delay, and dt_c the average time needed for the completion of a critical section. Then the total time required is:

$$T = dt_s + dt_c$$

The average throughput then will be :

$$1/T = 1/(dt_s + dt_c)$$

The maximum throughput is achieved when the synchronization delay is minimum and the critical section is completed in the minimum time, so we have respectively:

$$1/T_{min} = 1/(dt_{smin} + dt_{cmin})$$

8.

Extended definition: A host will never release a lock, until the operation for the specific lock is acknowledged. A host will never acquire a lock when it has released a lock after completing or aborting a transaction.

A host wishing to commit a transaction will not do it, unless all the locks (some of them located at other hosts) will have been acquired. This is the growing phase. When the transaction commits, or aborts, the locks are released by the hosts, which is the shrinking phase.