Brian Bauman

CSC 536 Homework 1

1.

1. The client should use the time from the third request to set its clock. Because no other information about the request and response times is known, the client should set its clock to 2:32:12.135 — the time reported by the timeserver, plus half the round-trip time. This time is, at most, 13 ms off compared to the time on the timeserver.
2. The time to which the client should set their clock remains 2:32:12.135, but the accuracy has improved. At most, the clock is 10 ms off compared to the time on the timeserver.
3. The time to which the client should set their clock is 2:32:12.136 and the accuracy of the measurement is slightly better — the time is, at most, 9 ms off compared to the time on the timeserver.

2.

1. Because the one clock moves 1% slower, the maximum clock skew over 1 minute is .01 minutes or 600 seconds.
2. If you will allow a skew of 200 milliseconds, you should resynchronize every 20 seconds.

3.

The Lamport timestamps are —

A: 1

B: 4

C: 5

D: 6

E: 7

F: 2

G: 4

H: 8

I: 1

J: 3

K: 5

L: 2

M: 3

N: 4

O: 5

P: 6

4.

I’m assuming that the vector timestamps increment with the send and receipt of a message. The vector timestamps are —  
  
A: [1, 0, 0]

B: [2, 1, 2]

C: [3, 4, 2]

D: [4, 4, 4]

E: [5, 4, 4]

F: [1, 2, 0]  
G: [1, 4, 0]  
H: [5, 6, 4]   
I: [0, 1, 0]  
J: [1, 3, 0]  
K: [1, 5, 0]  
L: [0, 1, 1]  
M: [0, 1, 2]  
N: [1, 3, 3]  
O: [1, 3, 4]  
P: [1, 5, 5]

5.

Because B chimed when its clock was [3, 2, 1], we know that A chimed afterwards — each entry in its vector timestamp ([6, 2, 5]) is greater than or equal to that of B. Another way to think of it is that B’s chime is it’s second “event” and A has heard of two events from B when A chimed. Similarly, A must have chimed after C because A has heard of 5 events from C when A chimed, and C’s chime was its 3rd event.

6.

1. P records its state as (7, 3), sends a marker to Q and R, and begins listening for in-flight messages.
2. Q receives the L message and increments its state to (7, 2).
3. Q sends the L message to R.
4. Q receives the marker from P, records its state as (7, 2), sends a marker to P and R, and begins listening for in-flight messages (from R only, as it received a marker from P).
5. P receives the marker from Q and stops listening on that channel.
6. R sends the K messages to Q.
7. R receives the marker from P, records its state as (6, 3), sends a marker to P and Q, and begins listening for in-flight messages (from Q only, as it received a marker from P).
8. P receives the marker from R and stops listening on that channel.
9. R receives the L message from Q and records it as a message in transit.
10. R receives the marker from Q and stops listening on that channel.
11. Q receives the K message from R and records it as a message in transit.
12. Q receives the marker from R and stops listening on that channel.

At this point, no more messages are being recorded and the following state was saved:

P: (7, 3), no messages in transit

Q: (7, 2), K message in transit from R

R: (6, 3), L message in transit from Q