CSE224 HW3

A53280596 Yawen Zhao

6.1 (a) 24h = 24 * 60 min = 24 * 60 * 60s = 24 * 60 * 60 * 1000ms. As the clock will drift 1 in 10^6, then for one clock it will drift approximately 24 * 60 * 60 * 1000 / 10^6 = 86.4ms. Then the max difference of these two clock can be 2 * 86.4 = 172.8ms

(b) For the skew to not exceed 20ms, the resynchronization interval should be 20ms / 2 * 10^6 = 10^4 s = 2.77h approximately.

6.2 Assume P to be process 1 and Q to be process 2,then we can get the following logical clock for each event(assume the event counter on each process is initially 0):

1) When P send a message to Q at event **a**, the logical clock of event **a** is 1.1;
2) When Q received the message event **a** send at event f, the logical clock of event **f** is 2.2;
3) When Q send a message to P at event **g**, the logical clock of event **g** is 2.2;
4) When P send a message to Q at event **b**, the logical clock of event **b** is 2.1;
5) When P send a message to Q at event **c**, the logical clock of event **c** is 3.1;
6) When Q received the message event **b** send at event **h**, the logical clock of event **h** is 4.2;
7) When P received the message event **g** send at event **d**, the logical clock of event **d** is 4.1;
8) When Q send a message to P at event **i,** the logical clock of event **i** is 5.2;
9) When Q received the message event **c** send at event **j**, the logical clock of event **j** is 6.2;
10) When P received the message event **i** send at event **e**, the logical clock of event **e** is 6.1;

6.3 Both process will initialize their vector clock as [0, 0]. Then we can get the following vector clock for each event:

1) When P send a message to Q at event **a**, the vector clock of event **a** is [1, 0];
2) When Q received the message event **a** send at event **f**, the vector clock of event **f** is [1, 1];
3) When Q send a message to P at event **g**, the vector clock of event **g** is [1, 2];
4) When P send a message to Q at event **b**, the vector clock of event **b** is [2, 0];
5) When P send a message to Q at event **c**, the vector clock of event **c** is [3, 0];
6) When Q received the message event **b** send at event **h**, the vector clock of event **h** is [2, 3];
7) When P received the message event **g** send at event **d**, the vector clock of event **d** is [4, 2];
8) When Q send a message to P at event **i**, the vector clock of event **i** is [2, 4];
9) When Q received the message event **c** send at event **j**, the vector clock of event **j** is [3, 5];
10) When P received the message event **i** send at event **e**, the vector clock of event **e** is [5, 4];

As for d and h, V(d) = [4, 2] and V(h) = [2, 3]. Then d(1) < h(1) and d(2) > h(2). Then d and h are concurrent. For f and e, V(f) = [1, 1] and V(e) = [5, 4]. Then f(1) < e(1) and f(2) < e(2). Then f is causally ordered before e.

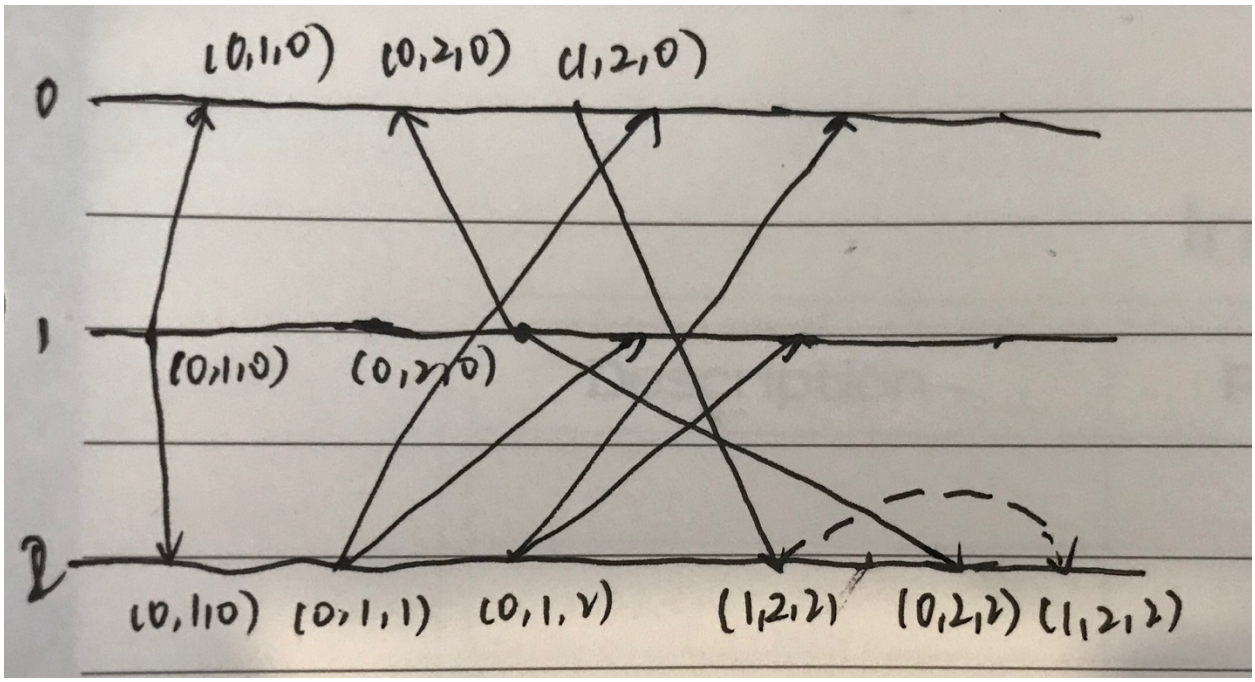
6.4 (a) is false. We can get it false by giving example from 6.3. Assume a is event d and b is event h and c is event i. From 6.3 we can know that event d is concurrent to event h and event h is causally

ordered before event i. However, event d and event i are concurrent instead of having causally order.
(b) is also false. We can also get it false by giving example from 6.3. Assume a is event h and b is event d and c is event i. From 6.3 we can know that event h is concurrent to event d and event d is concurrent to event i. However, event h is causally ordered before event I instead of concurrent to it.

6.12 The traffic light system which controls the traffic according to the current situation. Lack of physical time synchronization will cause traffic jam or accident.

15.2 Process 1 will take step e. As when the local clock of process 1 is (0, 2) and it received (2, 1) from process 0, it has nothing knowledge about the first event happened one process 0. So it will only accept m = (2, 1) after it get some knowledge of the first event which will be the message like (1, 0) or (1, 1) from process 0. When it receives one message like that it will then accept m. There is no need to requiring receive 2 of them. Hence, we can get that message will be accepted after receiving message (1, 0) or (1, 1).

15.3 a. Draw of one certain situation is shown as below:



b. The process will not accepted by process 2. As it has nothing knowledge about what happened on process 1 that make the event number on process 1 turn from 1 to 2. So it need to get the message from process 1 that contains the event message of event 2 on process 1 and then accept the message.