Secure and Dependable Systems
Jacobs University Bremen
Dragi Kamov

Course: CO21-320203
March 28th, 2019

# Homework 3

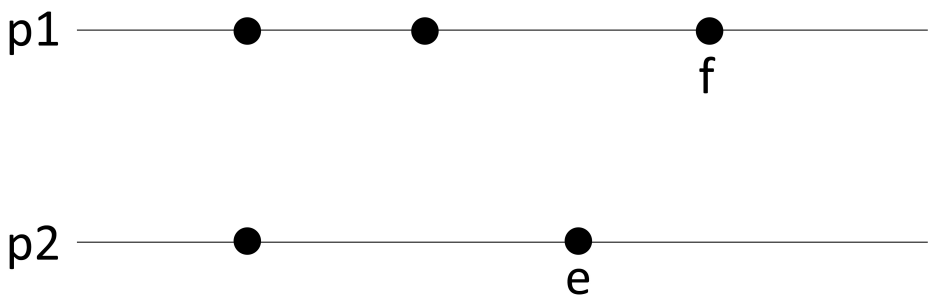
## Problem 3.1

**Solution:**

a) (i) $\Theta_L(e) < \Theta_L(f) \implies e \prec f$

(ii) $\Theta_L(e) < \Theta_L(f) \implies f \prec e$

Both of the statements are false, because it is impossible to determine which event happened first according to the clock value.
From the following picture we can see that although $\Theta_L(e) < \Theta_L(f)$, nothing can be inferred about which event happened first.




b) (i) $\Theta_V(e) < \Theta_V(f) \implies e \prec f$

(ii) $\Theta_V(e) < \Theta_V(f) \implies f \prec e$

The first statement is true, whereas the second one is false. In order to prove that, I am going to prove the right side of

$$e \prec f \iff \Theta_V(e) < \Theta_V(f)$$

This is equivalent to showing

$$!(e \prec f) \rightarrow !(\Theta_V(e) < \Theta_V(f))$$

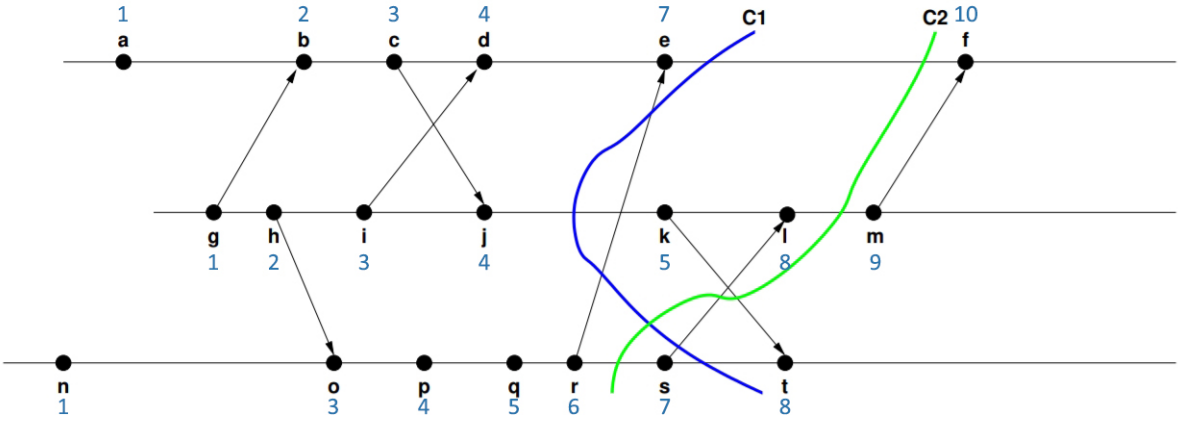
Let's suppose $e$ occurs at $p_i$ and $f$ at $p_j$, and $e$ doesn't happen before $b$. Let $\Theta_V(e)_i = k$. Since $e$ doesn't happen before $f$ there is no chain of messages from $p_i$ to $p_j$ originating at $p_i$'s kth step or later and ending at $p_j$ before $f$. Thus $\Theta_V(f)_i < k$. Thus $!(\Theta_V(e) < \Theta_v(f)$.

c) In order to find out whether $e$ and $f$ are concurrent we must do an element by element comparison of the corresponding timestamps; if some elements of $\Theta_V(e)$ are smaller than their $\Theta_V(f)$ counterparts and some are greater than their matches, then we know that the events are concurrent.

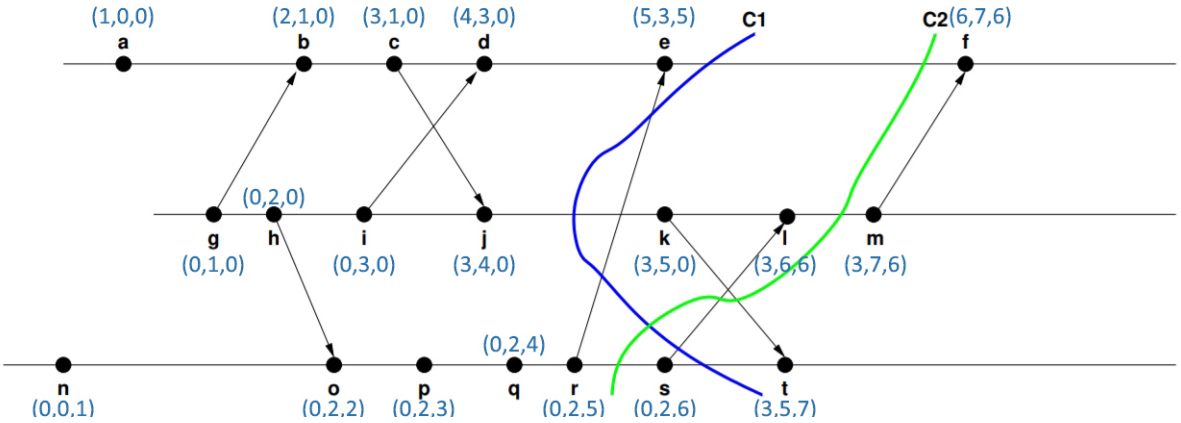
## Problem 3.2

**Solution:**

a) Lamport clock values for all events.




b) Vector clock values for all events.




c) The cut $C_1$ is consistent as there is no step outside the cut that happens before steps $e$, $j$ and $s$. On the other hand, cut C2 is not consistent as we can see that $s$ who happens outside the cut, happens before $l$ that is inside the cut.

## Problem 3.3

**Solution:**

Referenced from here

```
upon event < Init > do
    for all pi ∈ S: VC[pi] := 0;
    pending := ∅

upon event < rcoBroadcast, m> do
    trigger < rcoDeliver, self, m>;
    trigger < rbBroadcast, [Data,VC,m]>;
    VC[self] := VC[self] + 1;

upon event <rbDeliver, pj, [Data,VCm,m]> do
    if pj != self then
        pending := pending ∪ (pj, [Data,VCm,m]);
        deliver-pending.

procedure deliver-pending is
    While (s, [Data,VCm,m]) ∈ pending s.t.
        for all pk: (VC[pk] >=  VCm[pk]) do
            pending := pending - (s, [Data,VCm,m]);
            trigger < rcoDeliver, self, m>;
            VC[s] := VC[s] + 1.
```