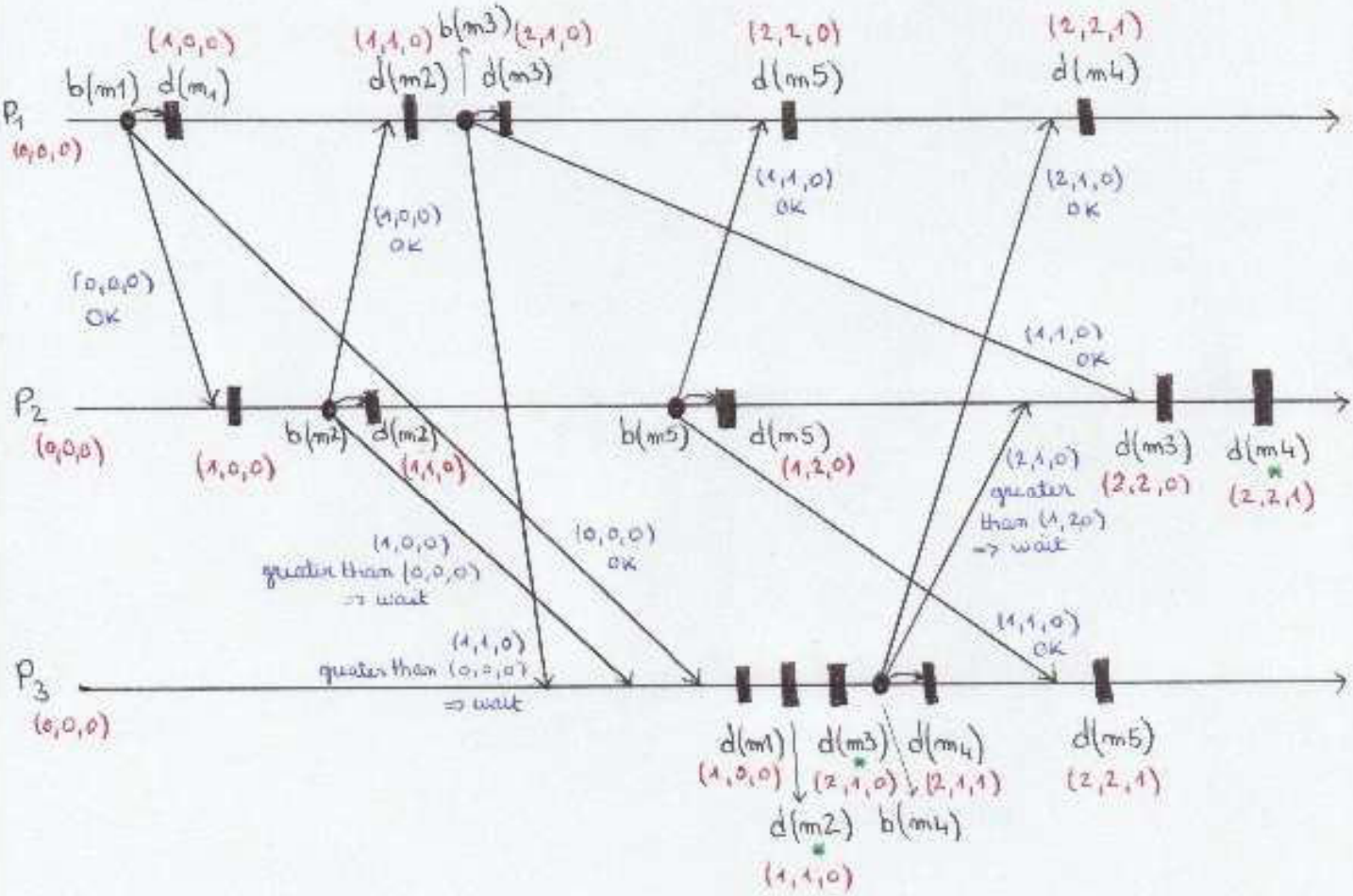
Distributed Algorithmics - homework #4

Justine Rochas

## Exercise 2

Here we are going to use the causal order broadcast algorithm in order to determine a consistent order in which each process have to deliver the message to the application layer. More precisely, a message received will have its delivery delayed if some messages that causally precede it haven't been received (and delivered) yet.




d(mX)* = delivery triggered by the delivery of the previous message

The final order for each process is:

$P_1$: m1, m2, m3, m5, m4

$P_2$: m1, m2, m5, m3, m4

$P_3$: m1, m2, m3, m4, m5

Here we can see that the messages are not delivered in the same order for each process because they are ordered according to a causality relation, which is not a total order relation. With a total order algorithm, we would have had the same order of messages for all the processes.

The merge operation is useless in this case due to the properties of the broadcast. Here are elements of proof:

- If the timestamp of a process $i$ is previously $(0,0,\dots 0)$, then obviously the receival of a message from a process $j$ require only a $+1$ on $V_i[j]$ to be correct
- Now suppose that the current timestamp of a process $i$ is $(x, y, \dots)$, then it means that $x$ events have been generated by $P_1$, $y$ events have been generated by $P_2$, etc... Since all events are broadcasts, all the processes will have eventually the same vector $(x, y, \dots)$. Therefore, the next message to be delivered (not received) on $P_i$ will have a timestamp either equal to the current timestamp on $P_i$ or smaller if the sender was not up to date yet. In this case, only the timestamp of $P_i$ has to be considered, and only the field of the sender in its vector has to be increased by 1.