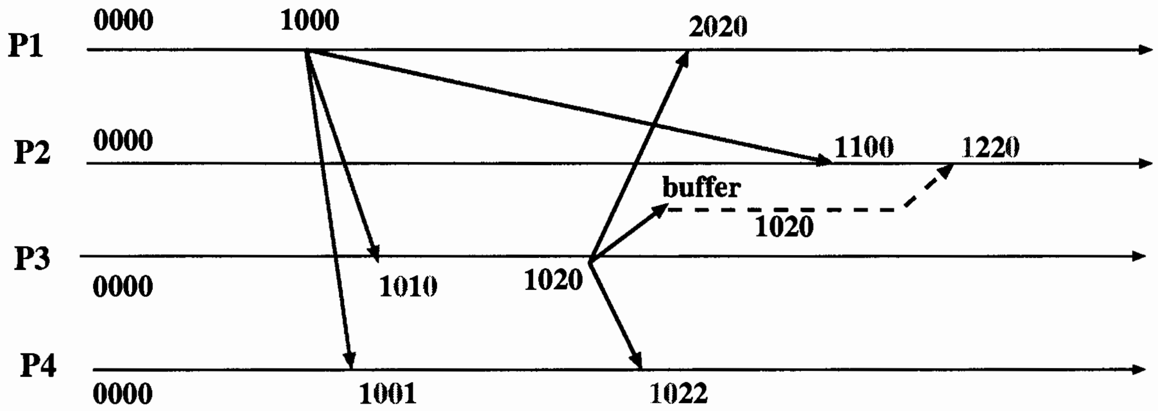
# Distributed Systems - solution

**CAUSAL ORDER using vector clocks - example**



```
algorithm: check for causal delivery order,
        i.e. receiver is not missing any message sender already has
        (system state at receiver ≥ system state at sender
          for processes other than sender and receiver
          recall we are assuming FIFO delivery from a given source)
        if not causal order, pend message - hold in buffer
        if causal order, deliver message to application and update local vector
            i.e. increment receiver's count and set all other counts
                to greater of sender's and receiver's values,
                thus updating receiver's record of system state
```

P3 (1020) to P1 (1000)
both have state of P2 as 0 and P4 as 0 -> deliver, P1->(2020)

P3 (1020) to P4 (1001)
both have state of P1 as 1 and P2 as 0 -> deliver, P4->(1022)

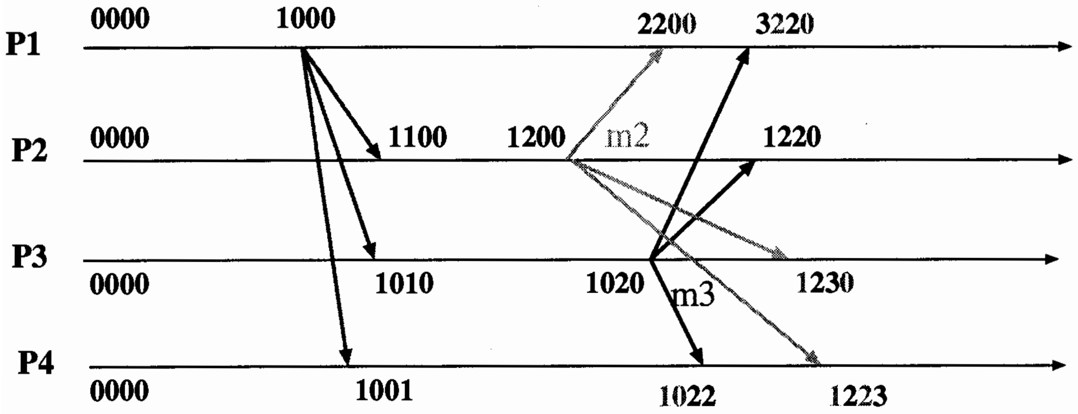
P3 (1020) to P2 (0000)
both have state of P4 as 0 - OK
P3 has state of P1 as 1, P2 has state of P1 as 0 -> buffer

P1 (1000) to P2 (0000)
both have state of P3 as 0 and P4 as 0 -> deliver, P2->(1100)

buffered message:

P3 (1020) to P2 (1100)
both have state of P1 as 1 and P4 as 0 -> deliver, P2-> (1220)

**TOTAL ORDER is not enforced by this algorithm - example:**



m2 and m3 are not causally related

P1 receives m1, m2, m3
P2 receives m1, m2, m3
P3 receives m1, m3, m2
P4 receives m1, m3, m2