CS425 Fall 2022 – Homework 2

(a.k.a “Once upon a time in Distributed Hollywood”)

Due: October 3, 2022 2:00PM

Mohan Amritesh Dasari

mdasari2

**Solution to Q1)**

Regular Bloom Filter:

*Case 1:*

K = 4; m = 2048; n = 4

False Positive = (1-e^(-k\*n/m))^k

= **3.667\*10^(-9)**

*Case 2:*

K = 4; m = 2048; n = 100

False Positive = (1-e^(-k\*n/m))^k

= **9.909\*10^(-4)**

Leo Bloom Filter:

To find the false positive rate we find the FP rate of each of the 4 bloom filters and multiply them. False positive means that the data doesn’t exist but we return that it exists. And because we are using 4 bloom filters instead of one traditional one, the chances of getting false positives on all 4 are significantly lower than getting false positive on one bloom filter.

*Case 1:*

K = 4; m = 512; n = 4

False Positive = ((1-e^(-k\*n/m))^k)^4

= **6.446\*10^(-25)**

*Case 2:*

K = 4; m = 512; n = 100

False Positive = ((1-e^(-k\*n/m))^k)^4

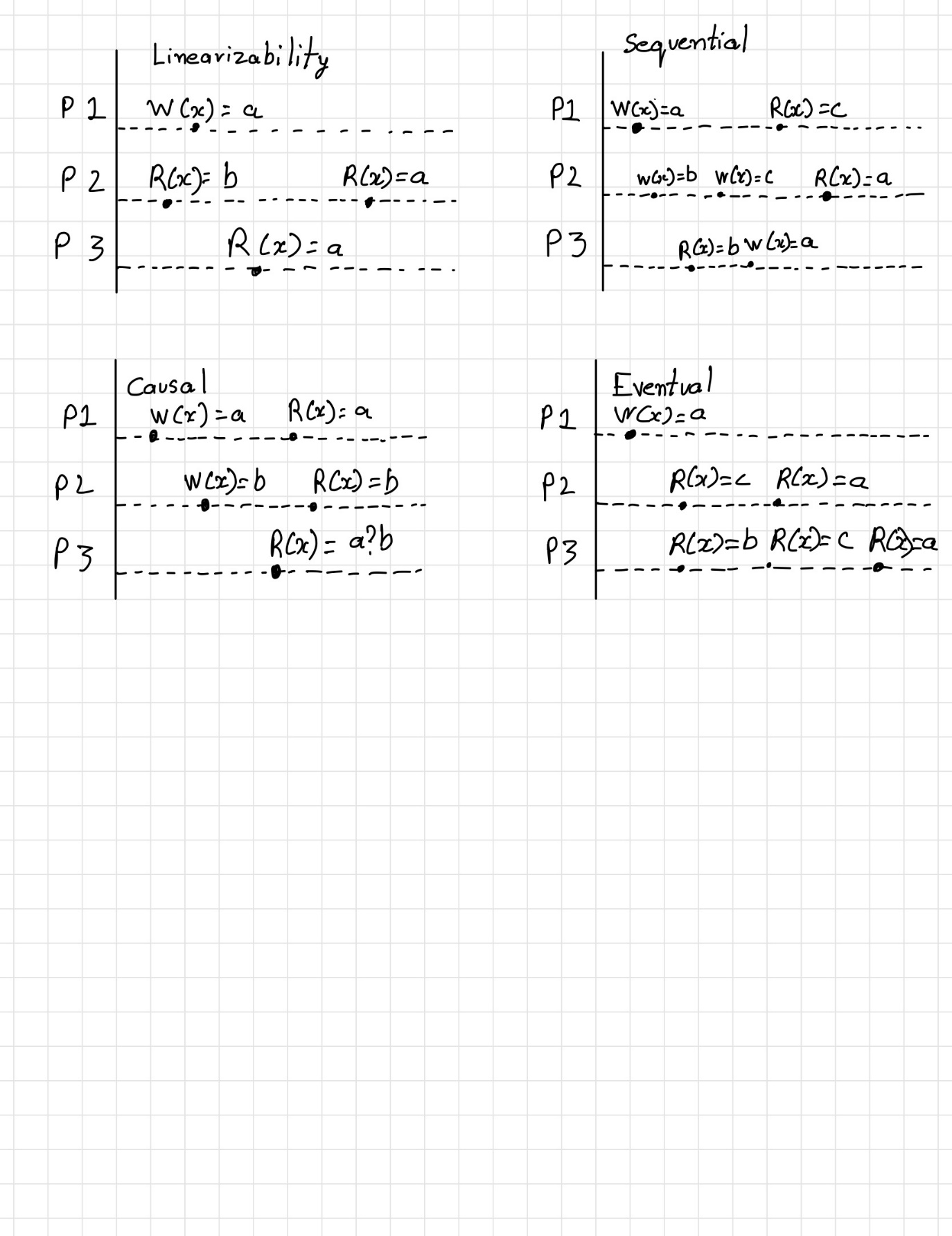
= **5.573\*10^(-5)**

**Hence, from the calculation above, we can see that Leo’s Bloom filter gives better False Positive rates than a Regular Bloom Filter in both the cases.**

**Solution to Q3)**

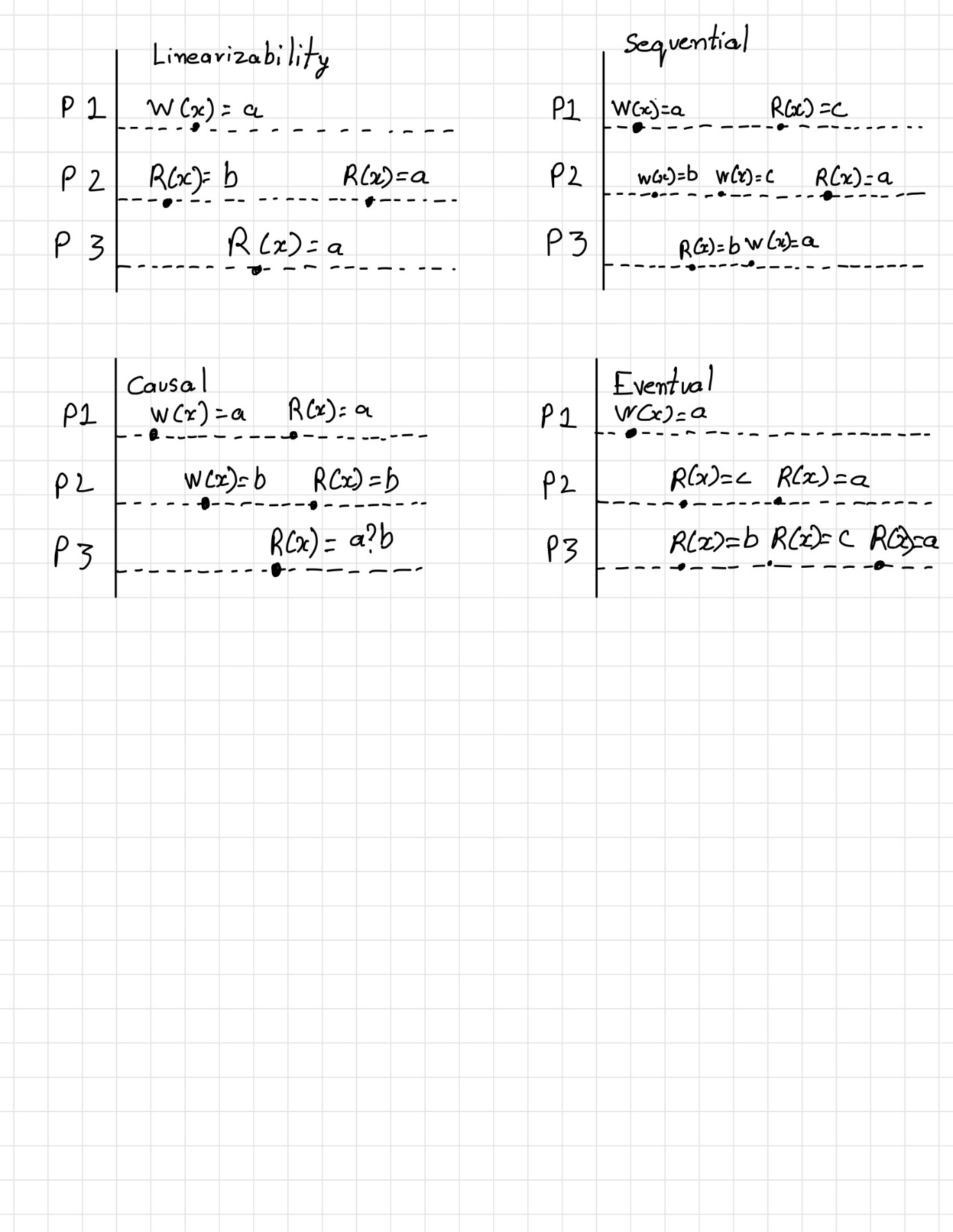
*Linearizability:* Linearizability states that every operation happens atomically, meaning an operation a occurs before operation b is executed, then the operation a should take place before operation b. Each operation is instantly available to all clients. Strongest consistency model.

Example:



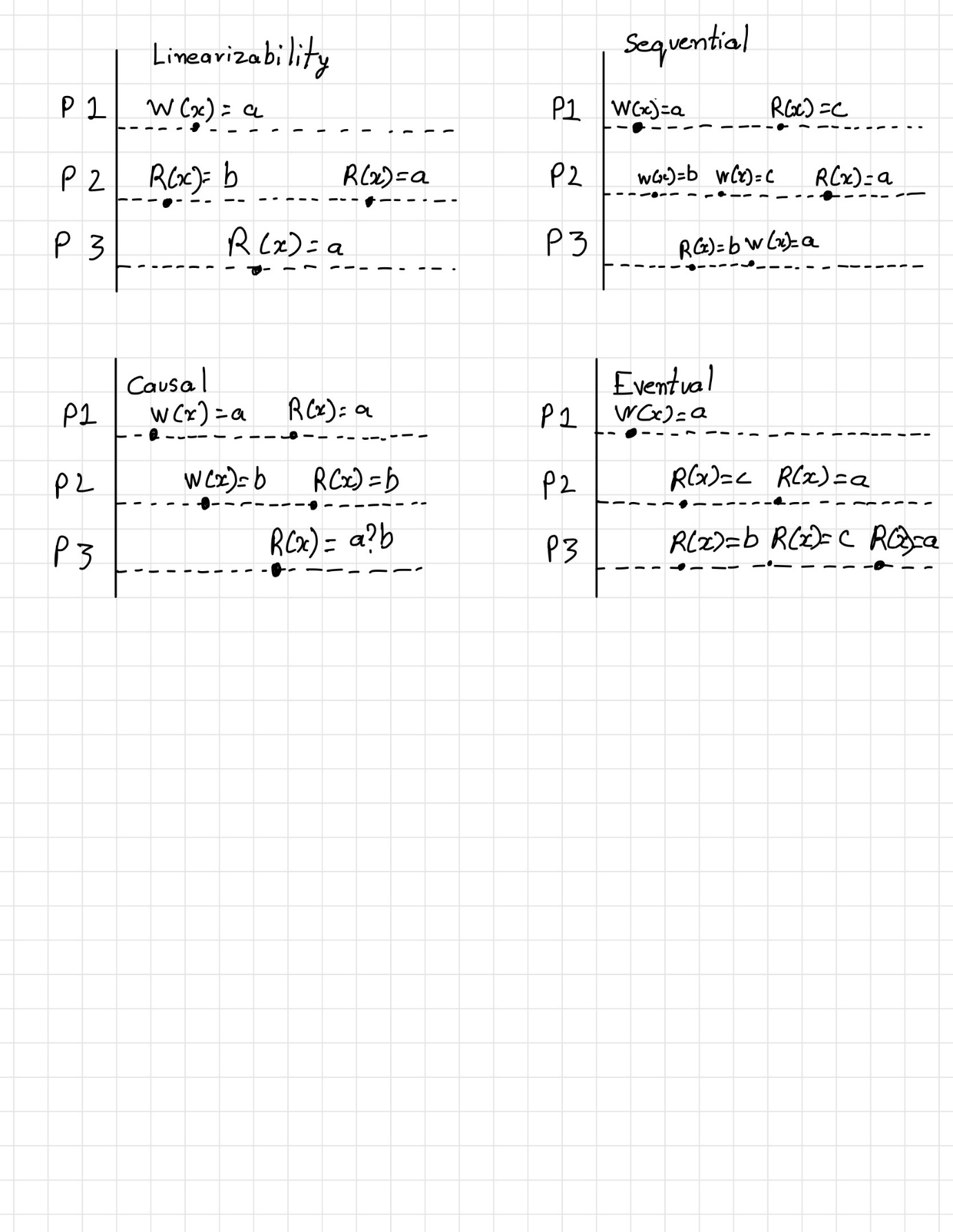
*Sequential Consistency:* In this type of consistency model, we can re-order events such that the order in which the operations are provided by one processor stays the same. Essentially what it means is that since each process is in its own state which is not similar to other processes, each process will have its own stale state and once it observes another process’s operation on it, it cannot go back to its current stale state. Key difference between sequential and Linearizability is that linearizability maintains atomic ordering but not necessarily program order like sequential.

Example:



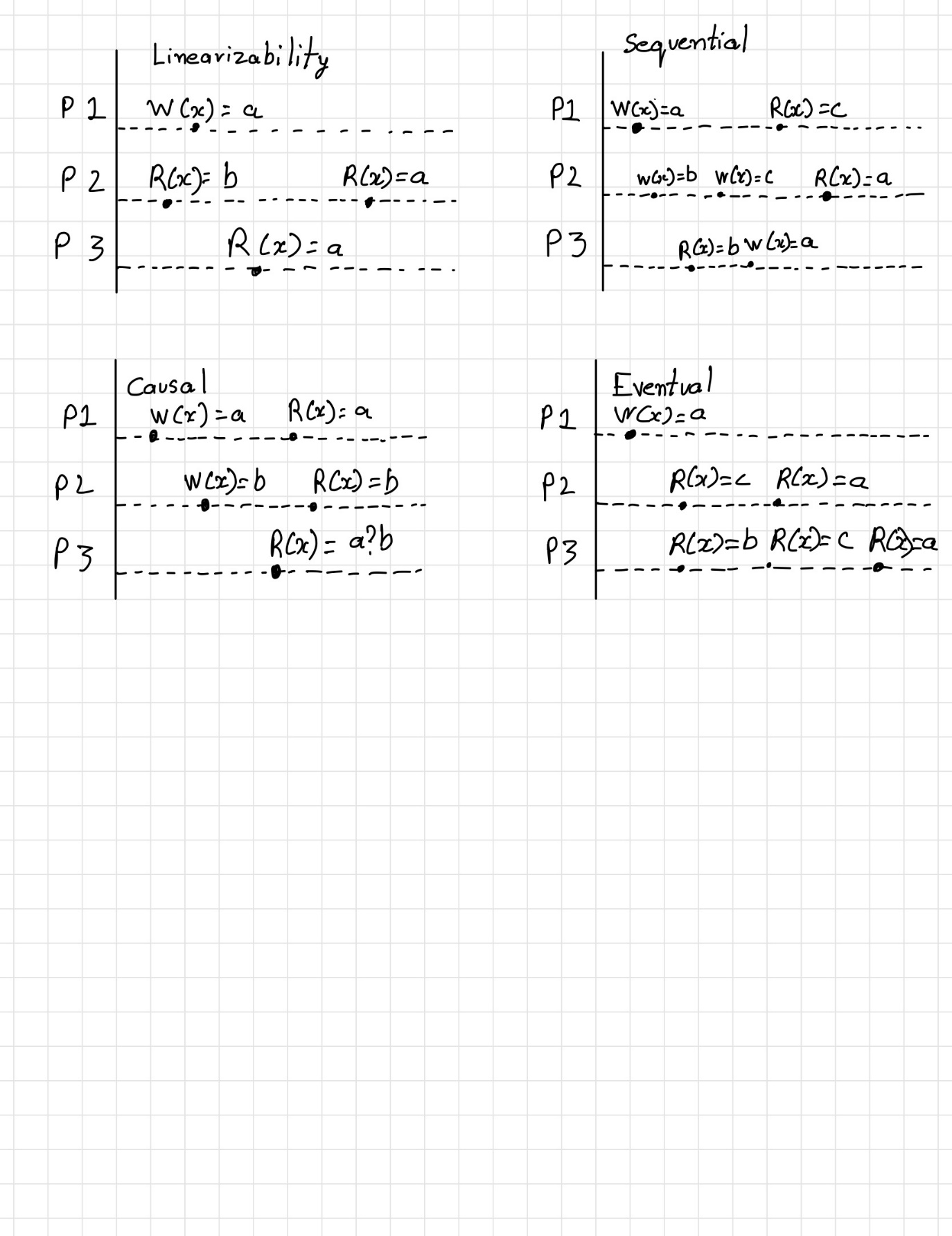
*Causal Consistency:* This form of consistency, causally related operations should appear in the same order in all processes. Very weak consistency model. This model states that operations that are causally related must be seen in the same order by all processors.

Example:



*Eventual Consistency:* Eventual consistency provides a distributed system with very high availability but very low consistency. This model states that given enough time without updates, the values of individual keys will be consistent across all nodes.

Example:



**Solution to Q4)**

RTT = 11.11ms

Min1 = 0.66ms

Min2 = 0.066ms

The formula to calculate error is given by (RTT-min1-min2)/2

So plugging the known values into the given formula we get:

= (11.11-0.66-0.066)/2

= 10.384/2

= **5.192ms**

That is the error in Thirteen’s calculation

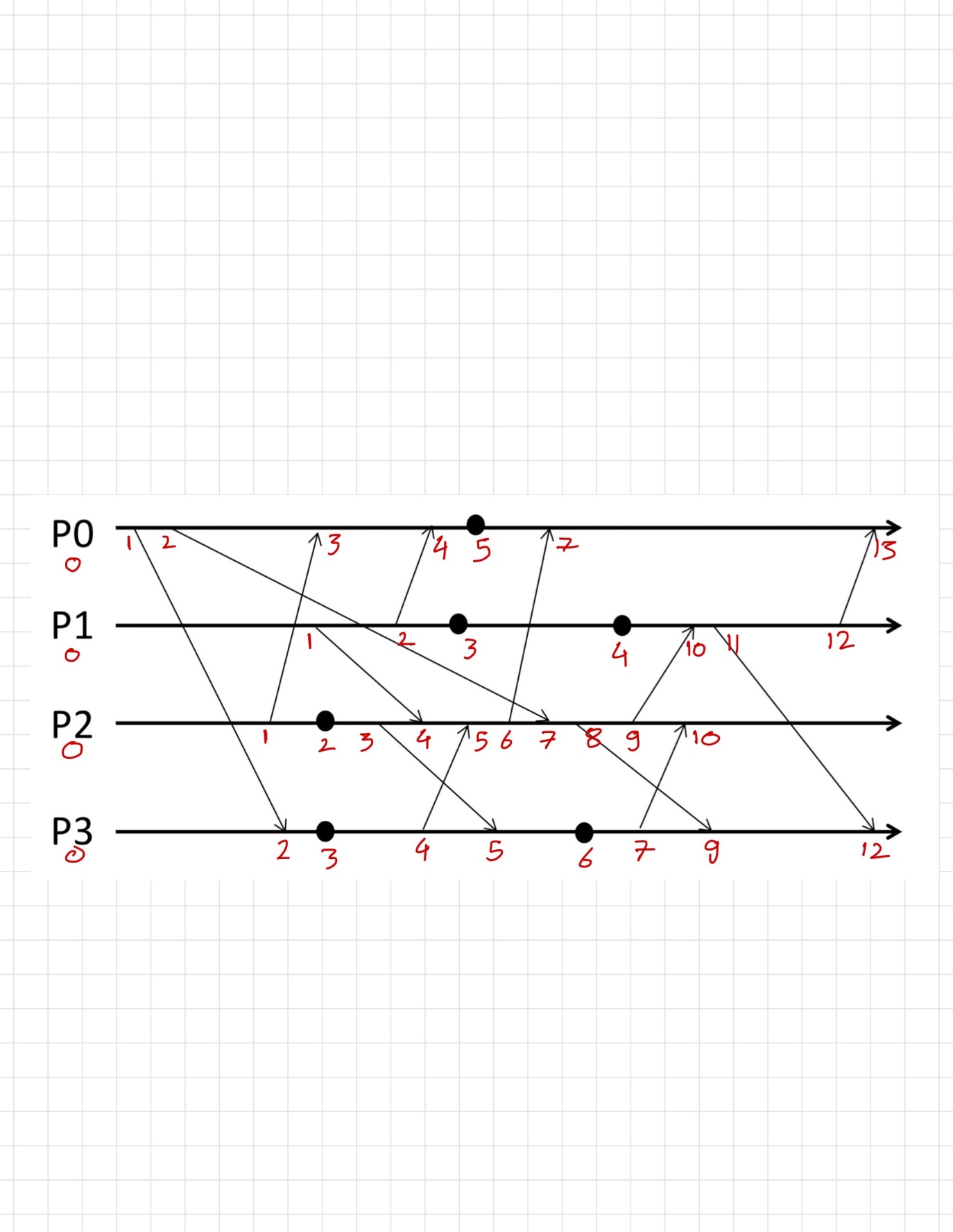
**Solution to Q5)**

Given timeline of messages:

A picture containing wire, electronic, day

Description automatically generated

The starting timestamp of all processes will be 0 and at every message receive, we do max(local,message)+1.



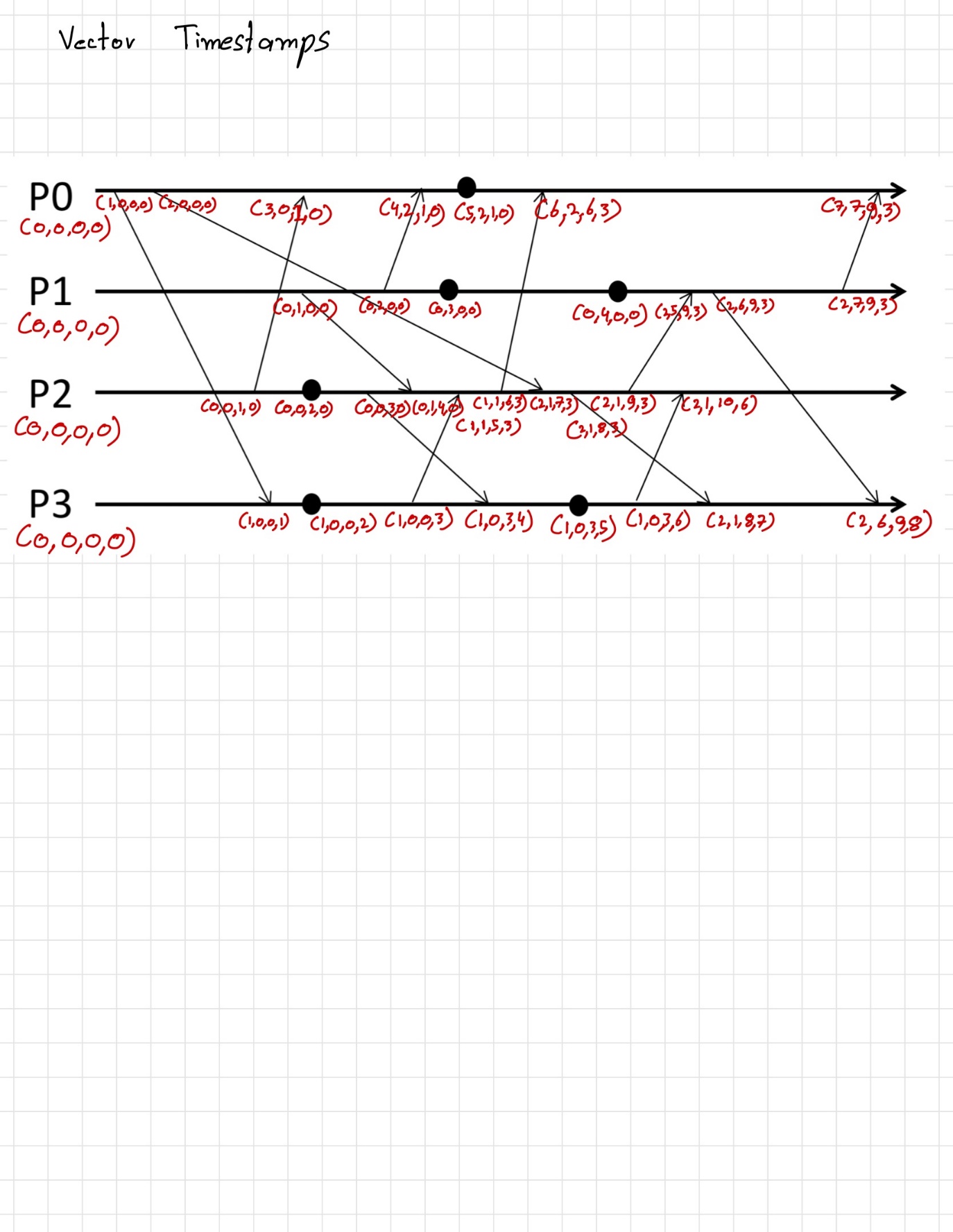
**Solution to Q6)**

Initial Timeline of messages from Q5:

A picture containing wire, electronic, day

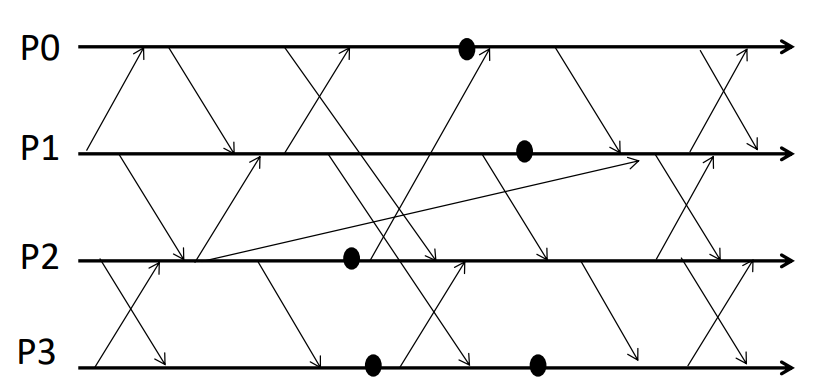
Description automatically generated

Vector Timestamp of the above messages:

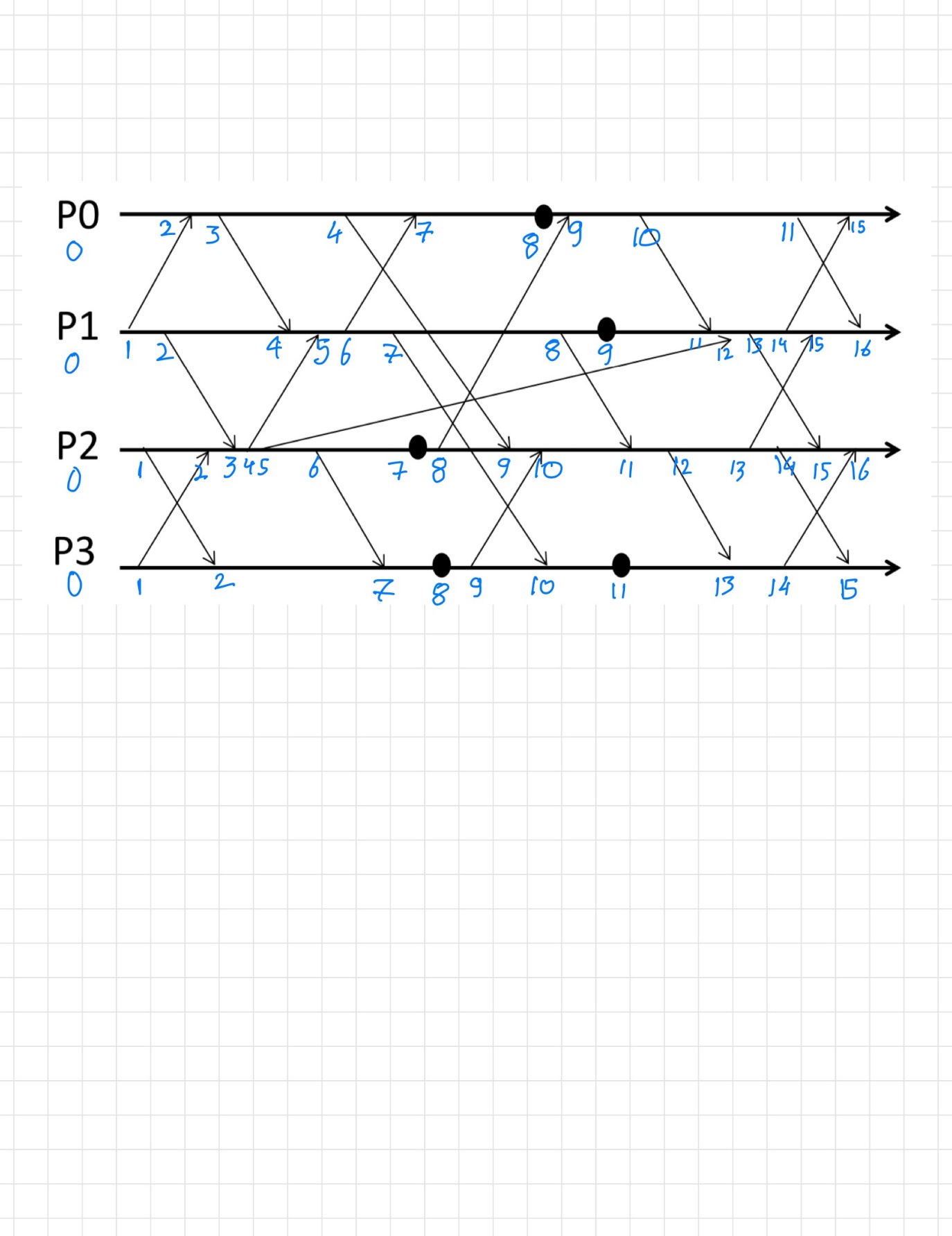


**Solution to Q7)**

Initial timeline of messages:



Lamport Timestamps of the above messages are as follows:



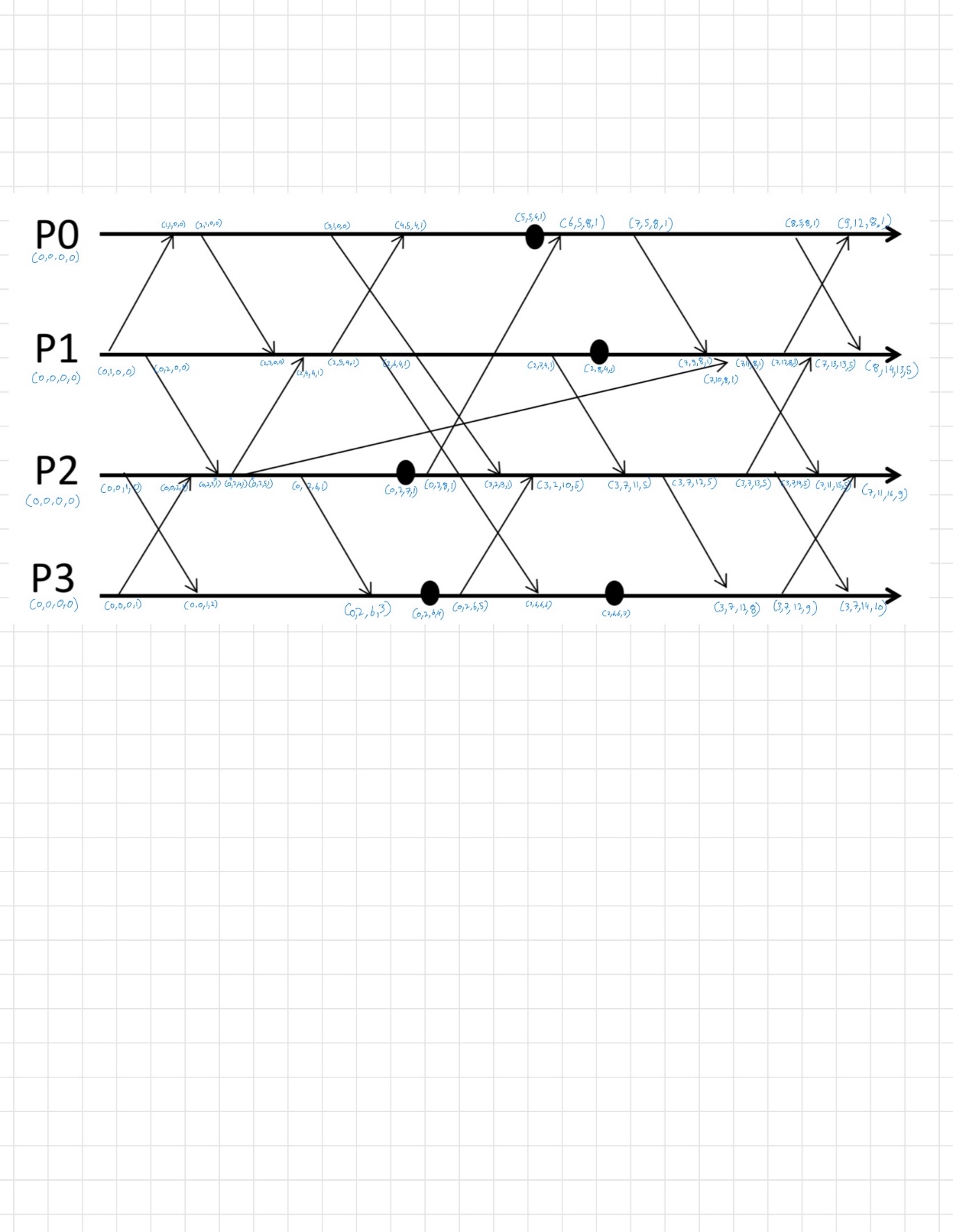
**Solution to Q8)**

Initial timeline of messages:

A picture containing indoor, wire

Description automatically generated

Vector Timestamps of above messages are as follows:



Sorry for such small font, since that is the only way, I could fit all those vectors in such a small space.

**Solution to Q10)**

1. *Write latency at a single replica node:* No changes. Compactor is not used in the process of writing to a replica node.
2. *Read latency at a single replica node:* Since the compactor is down, the number of sstables increases substantially and hence reading from those tables will lead to higher latency than if the compactor was still functioning.
3. *Correctness of writes at a single replica node:* Compactor plays no role in writing to a replica and hence there won’t be any effect in writing to a replica node.
4. *Correctness of reads at a single replica node:* Accuracy of a read request made to a single replica node reduces as the key in the sstable may contain older value depending on the consistency level of the entire Cassandra system and since the compactor is down, the sstable won’t be compacted and hence correctness of reads at the replica node will reduce when compared to when the compactor was running.
5. *Correctness of writes across the cluster (for a given consistency level):* Compactor is not used in the write process and hence there is no change in the correctness of writes across the cluster.
6. *Correctness of reads across the cluster (for a given consistency level):* If the consistency level was ‘ALL’, each operation will be executed on each and every replica node. Meaning every replica will have the same latest values and hence will maintain same correctness. If the consistency level was ‘QUORUM’, it ensures strong consistency therefore our correctness of reads will remain the same. From the above examples we can concur that for a given consistency level, correctness of reads across the cluster remains the same and does not depend on whether the compactor is up or not.