HW2 Solutions: CS425 FA23

1. (Solution and Grading by: Maleeha+other TAs.)
    a. 14 other processes receive the message.
    b. 3
    c. 2

2. (Solution and Grading by: Lavanya+other TAs.)

	A	B	C	D	E	F	G
Mercury	1		1		1		1
Venus		1		1		1	
Earth	1	1	1	1	1	1	1
Mars	1	1	1	1	1	1	
Jupiter	1	1	1	1	1		
Saturn	1	1	1	1			
Uranus	1	1	1				
Neptune	1	1					
Total	7	7	6	5	4	3	2

A peer to peer system like BitTorrent follows the ***rarest shard first policy*** in order to prioritize downloading shards which are least replicated among neighbors. Each shard in the system is replicated following times: A: 7 times, B: 7 times, C: 6 times, D: 5 times, E: 4 times, F: 3 times, G: 2 times

a) G : Since G is the rarest shard among neighbors and thus will be fetched first.
b) G -> F -> E -> D -> C -> A -> B
c) G : The 10th planet should have a set of shards that contains shard G but does <u>not</u> contain shard F so that shards G and F have a tie break at 3 and shard F is fetched first.

3. (Solution and Grading by: Shubhi Jain+other TAs.)
    a. False positive rate of original bloom filter = (1 - e^(-kn/m))^k (k = 2, m = 4096)
    b. Since all hash functions are different, each of the bloom filter is independent of the other and hence we can find the false positive of Leo-any and Leo-all as follows:
        i. False positive rate of Leo-any: Sum of false positive rates of the 4 bloom filters = P(B1 U B2 U B3 U B4) = P(B1) + P(B2) + P(B3) +P(B4) - P(B1 ∩ B2) - P(B1 ∩ B3) - P(B1 ∩ B4) - P(B2 ∩ B3) - P(B2 ∩ B4) - P(B3 ∩ B4) + P(B1 ∩ B2 ∩ B3) + P(B1 ∩ B2 ∩ B4) + P(B1 ∩ B3 ∩ B4) + P(B2 ∩ B3 ∩ B4) - P(B1 ∩ B2 ∩ B3 ∩ B4) = 4[(1 - e^(-kn/m))^k] - 6[(1-e^(-kn/m))^2k] + 4[(1-e^(-kn/m))^3k] - (1 - e^(-kn/m))^4k (k = 2, m = 1024)
        ii. False positive rate of Leo-all: Product of false positive rates of the 4 bloom filters = (1 - e^(-kn/m))^4k (k = 2, m = 1024)
    c. Now plugging in the values of n in the above formulae we get the following values:
        i. N = 10:
            1. False positive rate of original bloom filter = (1 - e^(-2*10/4096))^2 = **2.37 * 10^-5**
            2. False positive rate of Leo-any = 4[(1-e^(-2*10/1024))^2] - 6[(1-e^(-2*10/1024))^2*2] + 4[(1-e^(-2*10/1024))^3*2] - (1 - e^(-2*10/1024))^(4*2) = 1.49 *10^-3 - 8.397 * 10^-7 + 2.094 * 10^-10 - 1.95 * 10 ^-14 = **1.49 * 10^-3**
            3. False positive rate of Leo-all = (1-e^(-2*10/1024))^(4*2) = **1.95 * 10 ^-14**

            The best bloom filter = **Leo-all**
        ii. N = 1000
            1. False positive rate of original bloom filter = (1 - e^(-2*1000/4096))^2 = **0.149**
            2. False positive rate of Leo-any = 4[(1-e^(-2*1000/1024))^2] - 6[(1-e^(-2*1000/1024))^2*2] + 4[(1-e^(-2*1000/1024))^3*2] - (1 - e^(-2*1000/1024))^(4*2) = 2.94 - 3.25 + 1.59 - 0.294 = **0.995**
            3. False positive rate of Leo-all = (1-e^(-2*1000/1024))^(4*2) = **0.294**

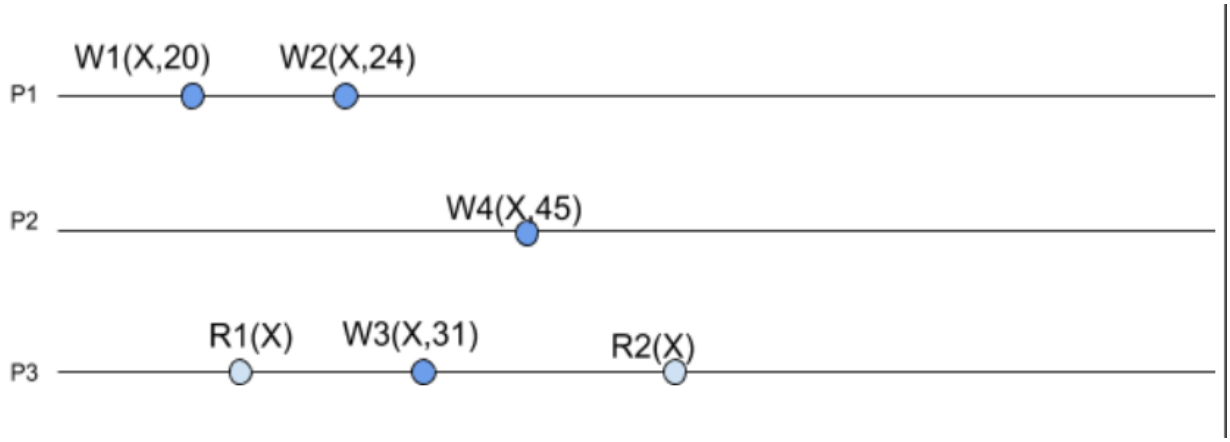
            The best bloom filter = **Original**

4. (Solution and Grading by: Lilia+other TAs.)
    a. Advantage: less memory usage (no need to store bloom filters in memory); writes to disk faster (no need to update the bloom filter on write to SSTables).
    Disadvantage: searches slower / reads slower (no quick way to check whether the search key exists in a SSTables for read).
    b. Advantage: faster reads (no need to merge data on disk (in SSTables) with data in RAM (in memtables)); no cache management needed; less memory usage.
    Disadvantage: slower writes (lots of disk accesses).
    c. Advantage: faster reads (no need to touch multiple SSTables).
    Disadvantage: extra bandwidth on flush (overhead to fetch and write multiple SSTables from/to disk); slower writes (need to modify SSTables every time)

5. (Solution and Grading by: Shulin Pan+other TAs.)
   The key realization is that given two sets of size N1 and N2, and a total of N nodes, the minimum overlap between the sets is N1+N2-N (try it with an example!).
   k=2: the intersection of two quorums is M + M - N
   Since we want the intersection to be non empty, this M + M - N is >= 1. Therefore, M >= (1 + N) / 2
   k=3: the intersection of three quorums is the intersection of 2 quorums (M + M - N) and one other quorum. Therefore, it's (M+M-N)+M - N, so M >= (1+2N)/3.
   k=4: the intersection of three quorums is the intersection of 3 quorums (M+M-N)+M - N and one other quorum. Therefore, it's ((M+M-N)+M - N)+M - N, so M >= (1+3N)/4.
   Inductively, M's minimum value is ⌈(1 + (k-1)N) / k⌉ for any k.

6. (Solution and Grading by: Kshitij+other TAs.)

- **Linearizability**: operations (reads and writes) appear to occur atomically and instantaneously, respecting a global total order consistent with real-time.
- **Sequential Consistency :** Implies that operations appear to take place in some sequential order which is consistent with the order of operations at each individual process
- **Causal Consistency:** Causally-related operations should appear in the same order on all processes but processes may disagree about the order of causally independent operations.
- **Eventual Consistency :** after a finite period without updates, all replicas will reach a consistent state, even though they may temporarily exhibit inconsistencies during concurrent updates.



Wi implies write Operation and Ri implies read operation in the above diagram.If we consider R2(X), it will return different values based on the consistency Model.

- **Linearizability -** R2(X) returns 45 as the last write as per the global clock is W4
- **Sequential Consistency -** R2(X) can return 24(Since W1 happens before W2 in P1, per process ordering is respected)
- **Causal Consistency -** R2(X) can return 31(Causal Ordering Preserved by reading W3**)**
- **Eventual Consistency -** R2(X) could return any of the values, like let's say 20, since in an eventually consistent system,factors like network delays or timing of updates can cause the read operation to return different values

Alternative solutions are also acceptable. Here is a simple one from Indy --

First let’s discuss Sequential vs. Linearizability. First, the similarity -- both of these create a total order on all operations across the system.

For the difference -- here's a simple example. Consider P1 does W(a=3). Some time later P2 does W(a=4). Some time later after that P3 does a R(a) and then R(a).

Linearizability must obey real global times of ops. In linearizability, P3's two reads must both return a=4 , the latest write in global time.

Sequential consistency does not need to obey real global times, but only each process' op order. In sequential consistency, it is ok if P3's first read gets a=3 and second read gets a=4 (or even a=3). This is because you can "shift" the relative order of operations across processes (as long as you don't change the relative linear order for any given process' instructions) for it to make sense. So in a world where P3's read came "just before" P2's write, this makes sense.

For causal consistency, if you add a fourth process P4 with two reads, and the two writes by P1 and P2 were concurrent, then P3's reads can return a=3 and a=4, while P4's reads can return a=4 and a=3 (note this is not allowed in either linearizability or sequential consistency). (If you don’t want to add a fourth process, you can add causally dependent writes by P1 and P2 and reorder their reads at P3, P1).

For eventual consistency, P3’s (and P4's) reads can both return stale value before any of P1's or P2's reads.

You can see other similar examples here:
https://www.cs.princeton.edu/courses/archive/fall18/cos418/docs/p8-consistency.pdf

7. (Solution and Grading by: Chaitanya Bhandari+other TAs.)

Knowns:

- RTT for one round of synchronization messages = 780μs
- Server-side delay from application to the network interface = 56μs
- Server-side delay from the network interface to the application = 120μs
- Client-side delay from the network interface to the application = 340μs
- Error = 0

Unknowns:

- Client-side delay from application to the network interface = Xμs

error <= (RTT-min2-min1)/2
RTT = 780μs
min1 = Minimum client → server latency = (X + 120)μs
min2 = Minimum server → client latency = (56 + 340)μs = 396μs

(780-(X + 120)-396)/2 = 0
(780-120-396-X)/2 = 0
(264-X)/2 = 0

Client-side delay from application to the network interface for 0 error: X ∈ [264μs, 264μs]

Alternative answer if the student has considered all delays apart from the RTT to be minimum: X ∈ [0μs, 264μs]

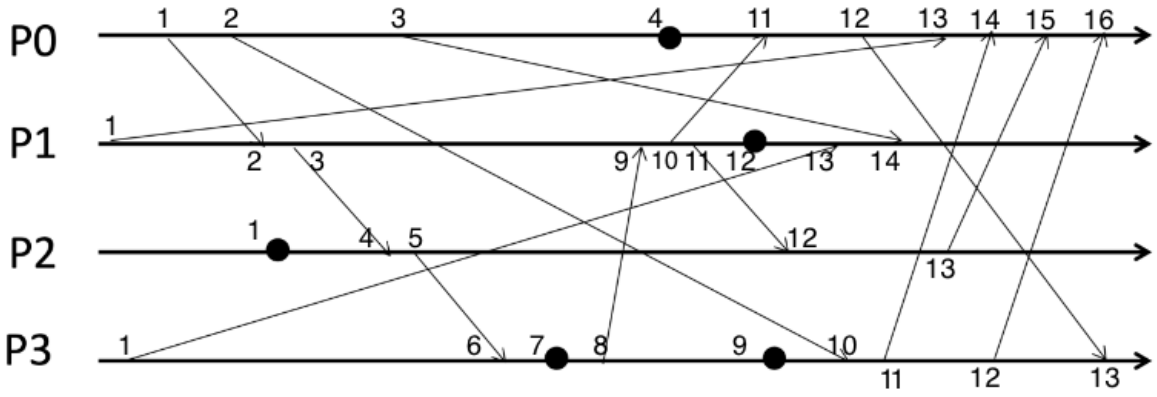
8. (Solution and Grading by: Aman+other TAs.)

The new error is 0.

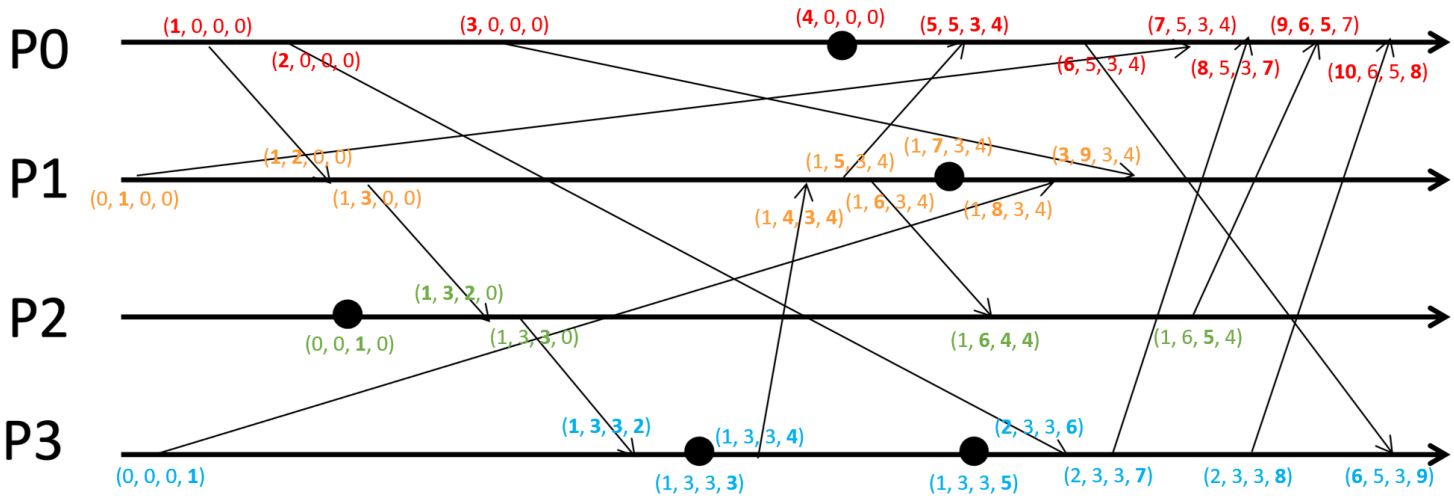
Let L be the 1-way latency, in both directions
tr1 = ts1 + L + oreal
tr2 = ts2 + L - oreal
Subtracting, we get:
tr1 - tr2 = ts1 + L + oreal - (ts2 + L - oreal) = ts1 - ts2 + 2oreal
Because o = (tr1 - tr2 + ts2 - ts1)/2, we plug in to see that o = oreal, so the error is 0.

Intuitively, the symmetry guarantee means the latencies in each direction cancel and we can isolate the skew.

9. (Solution and Grading by: Taksh+other TAs.)



10. (Solution and Grading by: Christina+other TAs.)



========== END of HOMEWORK 2 SOLUTION ==========