Homework 2 Solutions: CS425 FA25

Note to Students: Recommended solutions for HW2 (FA25) are below. For many questions, alternate solutions are possible and reasonable correct solutions will be accepted during grading. So please refrain from asking questions about solutions until you receive your HW grades back.

1. (Solution and Grading by: Talha Waheed.)

    a. A flood with TTL *i* reaches exactly the nodes ≤ *i* hops away. To be able to reach all nodes from any sender must be the longest shortest-path between any two nodes. In the tree, this would be between two opposite leaf nodes. The number of hops between them would be twice the height of the tree. We're told that the tree is a perfect binary tree with 2^m-1 nodes. Therefore the height of the tree would be m-1. So the answer is **2*(m-1)**.

    b. The child of the root will send queries to its own subtree, the root and its sibling tree. For its own subtree, the height would be (m-1)-1=m-2. As TTL is m-2, it will send queries to all nodes in the subtree i.e. (2^(m-2+1)-1)=2^(m-1)-1. Excluding the sending node, 2^(m-1)-2.
    Next, for the sibling tree, already two hops would be taken so the remaining TTL is m-2-2=m-4. So the height of the sibling tree that is traversed is m-4, i.e. 2^(m-3)-1 nodes.
    Therefore, the answer is (2^(m-1)-2) + 1 + (2^(m-3)-1) = **2^(m-1) + 2^(m-3) - 2**

    c. TTL will be the hops to the farthest nodes, i.e. the leaves in the opposite subtree. Therefore hops=1 (up to root) + 1 (down to sibling node) + (m−2) (sibling node) = **m**.

2. (Solution and Grading by: Tianchen Wang.)

   a. Finger Table at N2020:

i	ft[i]
0 (2020 + 2^0 = 2021)	2024
1 (2020 + 2^1 = 2022)	2024
2 (2020 + 2^2 = 2024)	2024
3 (2020 + 2^3 = 2028)	2028
4 (2020 + 2^4 = 2036)	1996
5 (2020 + 2^5 = 2052)	1996
6 (2020 + 2^6 = 2084)	1996
7 (2020 + 2^7 = 2148)	1996
8 (2020 + 2^8 = 2276)	1996
9 (2020 + 2^9 = 2532)	1996
10 (2020 + 2^10 = 3044)	1996
11 (2020 + 2^11 = 4068)	1996

   b. Finger Table at N2016 (full finger table not needed for this question):

i	ft[i]
0 (2016 + 2^0 = 2017)	2020
1 (2016 + 2^1 = 2018)	2020
2 (2016 + 2^2 = 2020)	2020
3 (2016 + 2^3 = 2024)	2024
4 (2016 + 2^4 = 2032)	2032
5 (2016 + 2^5 = 2048)	1996
6 (2016 + 2^6 = 2080)	1996
7 (2016 + 2^7 = 2144)	1996
8 (2016 + 2^8 = 2272)	1996

9 (2016 + 2^9 = 2528)	1996
10 (2016 + 2^10 = 3040)	1996
11 (2016 + 2^11 = 4064)	1996

Path: **2016 > 1996 > 2000**

c. Nodes after 2016 would not know the existence of 2016, since for the largest node, 2032 + 2^11 = 4080, meaning the last entry in the finger table only leads to 1996.

   1996 has its 4th entry being 2012 and 5th entry being 2028, no need for update.
   2000 has its 4th entry being 2016, so needs to be updated.
   2004 has its 3rd entry being 2012 and 4th entry being 2020, no need for update.
   2008 has its 3rd entry being 2016, so needs to be updated.
   2012 has its 0th, 1st, and 2nd entry being 2016, so needs to be updated.

   Therefore, the finger tables for **2000, 2008, and 2012** need to be updated.

3. (Solution and Grading by: Gabriella Xue.)

   a. Worst-case lookup cost is **O(1)**. This part of Cinelips does not differ from Kelips because the lookup protocol works the same: send a query to the contact in the file's affinity group to retrieve the file location.

   b. The memory usage is **O(N)**. Each of the 10 affinity groups has N/10 nodes and thus an O(N/10) membership list, or O(N).

   c. **Advantage of Cinelips over Kelips:**.
   Any one of the following points is acceptable:
      - A larger fraction of files will be in the sender's affinity group in Cinelips (10% in CineLips vs. 1/sqrt(N) in Kelips). In the best case, the file location is present locally at the sender in Cinelips and therefore does not involve a hop to look up the file location.
      - No need to rehash and redistribute nodes and files in Cinelips when nodes join or leave, since the number of affinity groups is fixed to 10 in Cinelips. (But it is acceptable if students assume that the number of affinity groups in Kelips is also fixed to K based on an estimate of N in the beginning).
      - More replication of file metadata within a group in Cinelips.

   Advantage of Kelips over Cinelips:
      - Better memory usage (O(sqrt(N) in Kelips vs. O(N) in Cinelips)

   We will accept other reasonable answers.

   d. In theory, Cassandra does not have a lower latency than Kelips. Their lookup costs are **both O(1) on average.**

   However, we will also accept if you say that Cassandra is better because each node has the complete membership list and fetches the data directly, thus incurring one less hop. In contrast, Cinelips incurs one additional hop to retrieve the data.

4. (Solution and Grading by: Lilia Tang.)

   We can simulate the Bloom filter by marking the output of each h1, h2, h3, h4 evaluated on the inputs 2024… until all outputs have already been marked. The newly marked values are bolded.

input	h1	h2	h3	h4	marked	FP
2024	16	8	8	8	[**8, 16**]	0
2025	18	26	2	10	[**2**, 8, **10**, 16, **18**, **26**]	0
2026	20	14	18	26	[2, 8, 10, **14**, 16, 18, **20**, 26]	0
2027	22	4	30	28	[2, **4**, 8, 10, 14, 16, 18, 20, **22**, 26, **28**, **30**]	0
2028	24	28	12	12	[2, 4, 8, 10, **12**, 14, 16, 18, 20, 22, **24**, 26, 28, 30]	0
2029	26	22	2	30	[2, 4, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30]	1

5. (Solution and Grading by: Chirag Shetty.)

- First note that quorums are chosen **arbitrarily.** So we are trying to find an M such that no matter how you choose k quorums of size M, there will be atleast r common elements common to ALL the quorums.
- **Observation 1:** If we want two sets with size M1, M2 to intersect over a universal set of size N, then we must have M1 + M2 > N. Else you can create two disjoint sets.
- **Observation 2:** Further, if two sets are intersecting, then the minimum size of their intersection is M1 + M2 - N (i.e union of M1 and M2 is the whole set N. Else there will be extra points in N that is neither in M1 or M2 and you can reduce the size of the intersection by adding these extra points to M2, while removing points from the intersection)

  A. For the case of k=3 quorums Q1, Q2, Q3. First we want Q1 and Q2 to intersect and the minimum size of their intersection, say set I_1 is (M + M - N) = 2M-N
- Now, the minimum size of intersection of I_1 and Q3 is (2M-N) + M - N
- We want (2M-N) + M - N >= r i.e **M >= (2N + r)/3**

  B. We can extend this to any k.
- We observe Minimum size of intersection with k+1 quorums = minimum intersection size of quorum Q_(k+1) with the smallest intersection of all quorums until k.
- Repeating this for k = 4, 5, we see that minimum intersection size is of the form k*M-(k-1)*N
- We will prove this by induction.
- Let I_k be the minimum size of quorums with k quorums of size M. For induction, we need to prove, if I_k = k*M-(k-1)*N, then I_(k+1) = (k+1)*M- k*N
- We know, Minimum size of intersection with k+1 quorums is intersection of Q_(k+1) with I_(k) i.e I_(k+1) = I_k + M - N = k*M-(k-1)*N + M - N = (k+1)*M- k*N
- Thus proved, I_k = k*M-(k-1)*N
- Finally, we want this minimum intersection I_k to be atleast r. Thus k*M-(k-1)*N >= r
- M >=((k-1)*N + r) / k
- **Ans: minimum M that guarantees there is always an element common to all quorums is ceil[((k-1)*N + r) / k]**

6. (Solution and Grading by: Kartik Ramesh.)

a. Advantage: Deleting records immediately from Memtables and SSTables frees up storage without waiting for compaction. Without tombstones, we will also incur lower compaction overheads.
   Disadvantage: A delete operation takes longer to complete since it incurs disk reads and writes. Without tombstones, replicas that missed the delete might continue to serve stale data. Additionally, if a replica was down, it retains the deleted data which can lead to “zombie data” when the replica comes back up.
b. Advantage: Preventing duplicate inserts makes reads faster since you only need to find one key in an SSTable, as opposed to reconciling multiple occurrences to find the latest write. It also lowers storage overhead.
   Disadvantage: It slows down writes, since every write must check if the key already exists. This disables updating keys, even when users might want to. Additionally, it breaks Cassandra’s eventual consistency model since we can not guarantee writes under partitions if we have to ensure consistency across replicas.
c. Advantage: Skipping Bloom Filters reduces storage overheads.
   Disadvantage: Finding keys within SSTables in the absence of Bloom Filters would incur excessive disk read operations.
d. Advantage: Removing compactions reduces system complexity, and eliminates CPU and I/O overhead during the compaction process.
   Disadvantage: SSTables grow indefinitely, degrading read performance and increasing space overhead. Additionally, tombstones will never get removed.

   We will accept other reasonable answers, as well as those that focus on the distributed read and write protocols, rather than just the local storage.

7. (Solution and Grading by: Ashish Kashinath.)

Suppose parent is process A and child is process B.Referring to notation from NTP slides

Ts1	Send timestamp of parent for message 1
Tr1	Receive timestamp of child for message 1
Ts2	Send timestamp of child for message 2
Tr2	Receive timestamp of parent for message 2
L1	latency of message 1
L2	latency of message 2
Oreal	real offset between the parent and child clocks. Suppose the child is ahead of the parent by Oreal. Equivalently, the parent is ahead of the child by -Oreal.
O	Clock adjustment that the child will do
| Oreal – O |	NTP error bound

Then, the relation between the timestamps is:

$tr1 = ts1 + L1 + Oreal$,

$tr2 = ts2 + L2 – Oreal$

Since L1 and L2 are identical, lets us say L1 = L2 = l

Then,

$tr1 = ts1 + l + Oreal \rightarrow (1)$ and

$tr2 = ts2 + l – Oreal \rightarrow (2)$

Subtraction of Eqns. (1) – (2) gives $tr1 – tr2 = ts1 – ts2 + 2*Oreal$

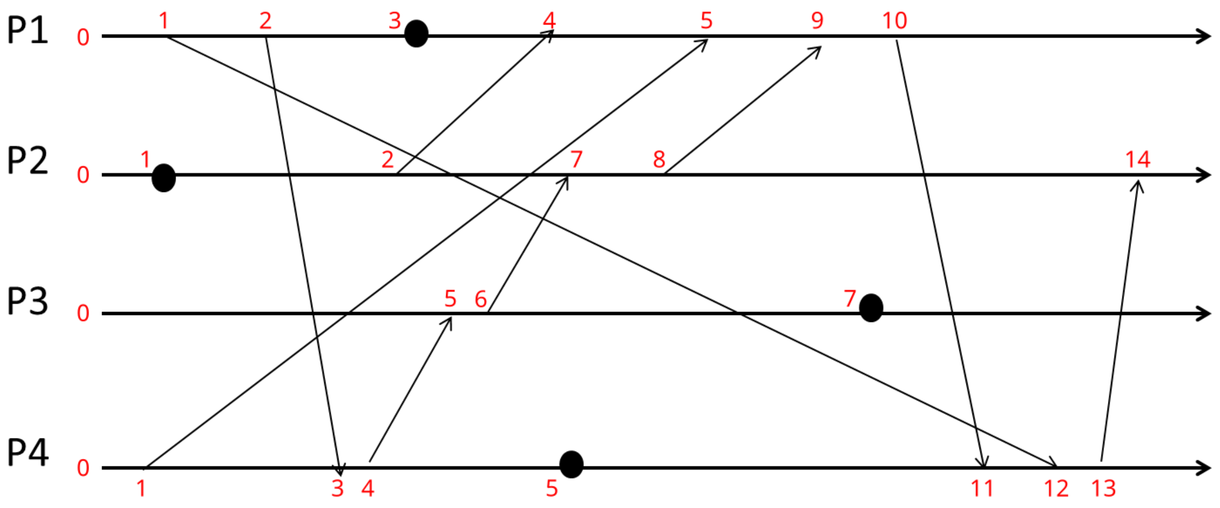
i.e., $Oreal = [(tr1 – tr2) + (ts2 – ts1)] / 2$ = offset adjustment of the child, O

NTP Error Bound = | Oreal – O | = 0

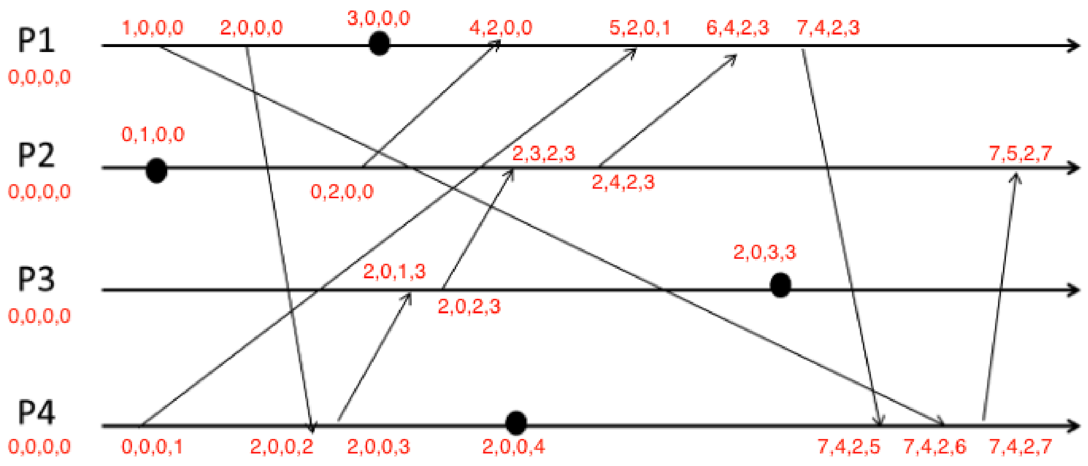
Thus, we make 2 observations : (a) First, in the case of symmetrical delays, the offset adjustment by the child is identical to the real clock offset between the parent and child, (b) Second, the NTP error bound becomes zero in case of symmetrical delays.

In contrast, in asymmetrical networks, NTP error bound is bounded by the round trip time.

8. (Solution and Grading by: Nishant Sheikh.)



9. (Solution and Grading by: Madhav Jivrajani.)



10. (Solution and Grading by: Hanbo Guo.)

a. **Yes**, obeys causality. Recall causality means A -> B => timestamp(A) < timestamp(B). If A and B on the same process, timestamp(B) = timestamp(A) + something, so causality remains. If A is the send event and B is the receive event, timestamp(B) = max(timestamp(A) + B's local timestamp) + positive integer > timestamp(A), so causality remains.

b. **Yes,** obeys causality. Note that ALL elements in the vector increments by 1 on an instruction or event, so it reduces to Lamport timestamps.

c. **No**. While it does obey causality, in (b)'s algorithm (and Lamport timestamps), we can no longer tell if two events are concurrent or causally related from just the timestamps. In (b)'s algorithm, we lose causality information that each element of the vector would typically track; thus, timestamp(A) < timestamp(B) does not imply A → B, which vector timestamp guarantees, where VT_A < VT_B implies A → B.