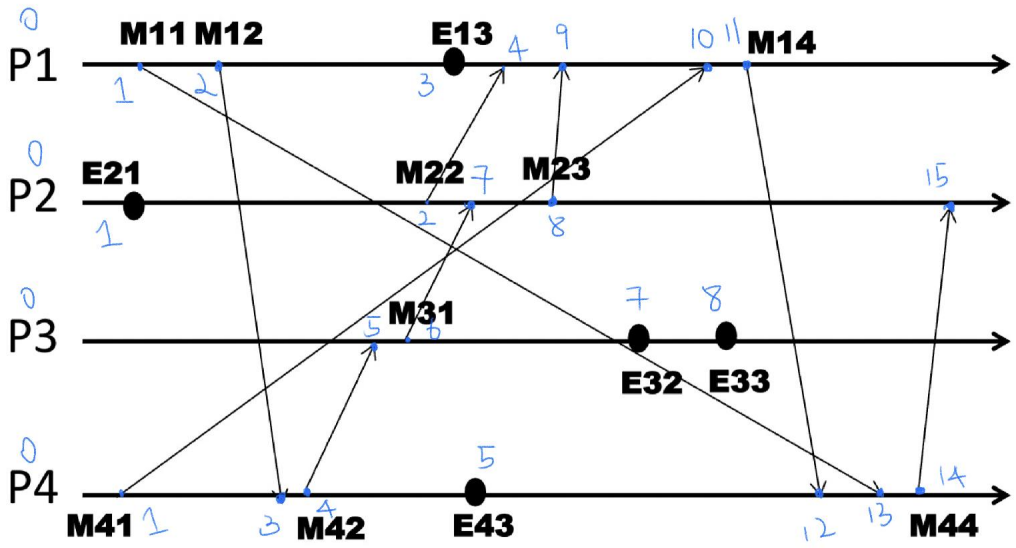
Midterm A (Coursera) Solutions: CS425 FA23

(Grading Rubrics are on Gradescope LTI)

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

    1. (f) (None of the above statements are true)
    2. (d) (Since C100 sets R=1, there is no guarantee that it will see its own writes. Nor can it see other clients' recent writes.)
    3. (b)
    4. (b) (Detection time does not change since it depends on detection timeout)
    5. (e) (Map outputs are quicksorted, Reduce inputs are mergesorted)
    6. **Consistency**
    7. **8 months.** $200K/(500TB*1000TB/GB*$0.05/GB-month) = 8 months.
    8. **(q, 33, 22)** – No change, since the heartbeat count for q at p is more up-to-date than the received heartbeat count for q from r.
    9. **S43**. Since replica nodes S22, S42, S41 are not available, Hadoop prefers rack-local: either rack 2 or rack 4. Since rack 2 has no free containers, the only rack-local container is at S43.
    10. **False** – a counterexample is an even pair that is concurrent, with LT1 < LT2 (which is possible), but VT1 and VT2 are (guaranteed to be) incomparable.

2. (Solution and Grading by: Shulin Pan+TAs.)



a. Send(M11): 1, Recv(M11): 13
   Send(M12): 2, Recv(M12): 3
   Send(M14): 11, Recv(M14): 12
   Send(M22): 2, Recv(M22): 4
   Send(M23): 8, Recv(M23): 9
   Send(M31): 6, Recv(M31): 7
   Send(M41): 1, Recv(M41): 10
   Send(M42): 4, Recv(M42): 5
   Send(M44): 14, Recv(M44): 15
   E(13): 3
   E(21): 1
   E(32): 7
   E(33): 8
   E(43): 5

b. The last element of M44 would be 7. That is because there are 7 total receiving/sending/events on P4 and each time we would increment the last element.

3. (Solution and Grading by: Lavanya Puri+TAs.)
    a. In a chord P2P system with 1024 points, m = 10 since 2^10 = 1024.
    The finger table entries of machine with peer ID 927 should therefore include the first peer whose id is greater than or equal to (927+2^i), where i ranges from 0 to 9:

i	(n + 2^i) mod 2^m	Finger Table Entry
0	>= 927 + 2^0 = 928	0
1	>= 927 + 2^1 = 929	0
2	>= 927 + 2^2 = 931	0
3	>= 927 + 2^3 = 935	0
4	>= 927 + 2^4 = 943	0
5	>= 927 + 2^5 = 959	0
6	>= 927 + 2^6 = 991	0
7	>= 927 + 2^7 = 1054 mod 1024 = 30	44
8	>= 927 + 2^8 = 1183 mod 1024 = 159	274
9	>= 927 + 2^9 = 1439 mod 1024 = 415	504

    b. To route a message for key (file id) 30 starting from peer id 149, the following hops will be taken:
    149 => 927 => 0 => 24 => 44

        1. To begin with we will look at peer id **149**'s finger table and look for the largest successor/finger entry <= 30. Thus, **hop to peer id 927.**

i	(n + 2^i) mod 2^m	Finger Table Entry
0	>= 149 + 2^0 = 150	274
1	>= 149 + 2^1 = 151	274
2	>= 149 + 2^2 = 153	274
3	>= 149 + 2^3 = 157	274
4	>= 149 + 2^4 = 165	274

5	>= 149 + 2^5 = 181	274
6	>= 149 + 2^6 = 213	274
7	>= 149 + 2^7 = 277	504
8	>= 149 + 2^8 = 405	504
9	>= 149 + 2^9 = 661	927

2. Now we refer to the finger table of peer id 927 from part a and look for the largest successor/finger entry <= 30. Thus, **hop to peer id 0.**
3. Now we refer to the finger table of peer id 0 and look for the largest successor/finger entry <= 30. Thus, **hop to peer id 24.**

i	(n + 2^i) mod 2^m	Finger Table Entry
0	>= 0 + 2^0 = 1	1
1	>= 0 + 2^1 = 2	2
2	>= 0 + 2^2 = 4	4
3	>= 0 + 2^3 = 8	13
4	>= 0 + 2^4 = 16	24
5	>= 0 + 2^5 = 32	44
6	>= 0 + 2^6 = 64	81
7	>= 0 + 2^7 = 128	149
8	>= 0 + 2^8 = 256	274
9	>= 0 + 2^9 = 512	927

4. Lastly at peer id 24, the first finger table entry will be peer id 44. Thus The largest successor/finger entry <= 30 does not exist. Hop to successor(24). That is, **hop to peer id 44**. **File with key=30 is stored here.**

c. The nodes needing to change finger table entries must at least have some finger table entry that is larger than 927. They must therefore at least satisfy the requirement that node_id > 415 (since, node_id + 2^9 > 927).

We can see that two peers 927 and 504 qualify for this. However, for peer 504 the last finger table entry is 504 + 2^9 = 1016 which is > 1000, thus its finger table entry will not change. **The only node whose finger table entry will change is peer id 927. (from i = 0 to i = 6)**

4. (Solution and Grading by: Aman+TAs.)

MR1 reads from D1 to get number of posts in an hour
```
M1(key=_, value=(a, p, time)):
    output (key=hour(time), value=(1, p))
R1(key=hour, value=V):
    Ps = set of ps in V
    for p in Ps
      output (key=p, value=(HOUR, len(V), hour))
```

MR2 reads from D2 to get the total likes for a post:
```
M2(key=_, value=(a, p, time)):
    output (key=p, value=1)
R2(key=p, V):
    output (key=p, value=(LIKES, len(V)))
```

MR3 reads from MR1 and MR2, output the contribution of that post to the average (# likes for that post/posts in the same hour):
```
M3(key=p, value=V):
  identity
R3(key=p, value=HOUR/LIKES tuples)
     (posts, hour) = (x, y) for (HOUR, x, y) in values
     likes = z for (LIKES, z) in values
     output (key=hour, value=sum(likes)/sum(posts))
```

MR4 reads from MR3's output and collects the average per hour:
```
M4(key=hour, value=v):
     identity
R4(key=1, value=Vs):
     output(hour, sum(Vs))
```

MR5 collects MR4's output into one key and outputs the answer
```
M5(key=hour, value=avg):
     output(1, (hour, avg))
R5(key=1, value=Vs):
     output the hour with the highest avg, and if there are ties then output all highest hours
```

5. (Solution and Grading by: Shubhi Jain+TAs.)
    a. M = 2k + 1
    This is because there can be a process p_i which has not been chosen as the random target by any other process in the system given that k << N. Process p_i will be selected as the target by its k successors and k predecessors and p_i itself will select these k successors and k predecessors as targets. If all the k successors and k predecessors of p_i fail simultaneously, p_i will be able to detect their failure. But after the failures, there are no processes that have p_i as their SWIM target. Hence, p_i's failure goes undetected.

    b. No, the algorithm is not 100% accurate. If the ack to any ping is dropped, then the process responsible for receiving the ack will not be able to distinguish between the dropped ack or an actual process failure.

    c. Worst case: N-1
    A process might be selected as the random target by every other process in the system and would need to send acks to all the other processes.

    Best case: 2k
    A process might not be selected as the random target by any other process in the system and thus would need to send acks to the pings of only its k successors and k predecessors.

    Average case: 3k
    Each process selects 3k SWIM targets in the system. Hence, there are a total of (3k*N) acks being sent out every second. Thus, on an average (3k*N)/N = 3k acks are sent out every second.

Rubric:

5. A.
- -10 if not attempted/completely incorrect
- -7 is mostly incorrect but attempted
- -3 if partially incorrect

5. B.
- -2 if correct answer but explanation not there/incorrect
- -5 if completely incorrect

5. C.
- -1 for each incorrect case/ unattempted
- -5 if completely incorrect

========== END of MIDTERM A SOLUTION ==========