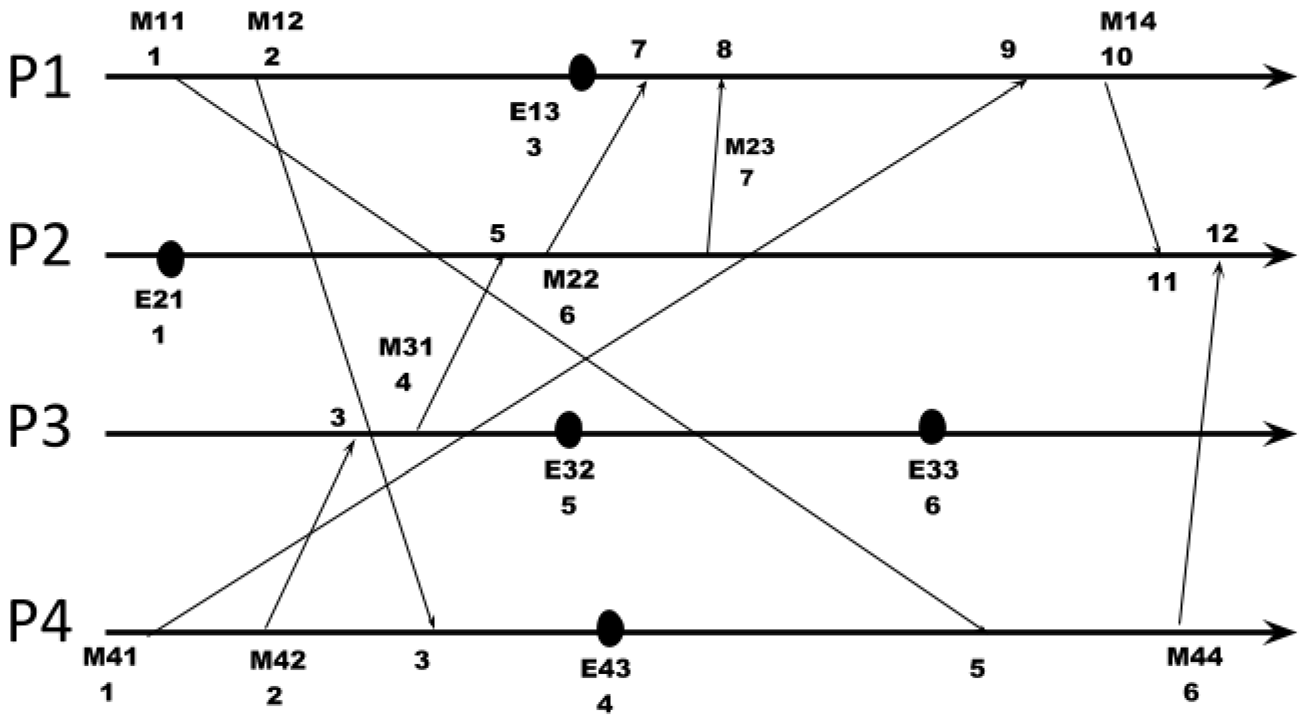
Midterm B (On-Campus) Solutions: CS425 FA23

1. (Solution and Grading by: Maleeha+TAs.)

    1. (c) PUE is determined by dividing the total amount of power entering a data center by the power used to run the IT equipment within it. PUE = 1000/800 = 1.25 (c)
    2. (b) Since p has a higher heartbeat counter value, it won't update it with the lower value that r sends.
    3. (e) Rarest block fetched first (e)
    4. (a) . Note that for (c) equal Vector timestamps imply the two events e1 and e2 are identical. For (d) equal Lamport TS indicate that either e1 = e2, or e1 is concurrent with e2, but definitely e1 and e2 are not causally related.
    5. (d) Both (a) and (b) will be false. Since C100 sets R=1, there is no guarantee that it will see its own writes. Nor can it see other clients' recent writes.
    6. **e1 happens before e2, or e1 → e2,**
       Rubric:
       +2 for true or happens before
       +1 for e1 < e2 or e1<=e2
    7. **5 (Full Credit for all since question out of syllabus)**
    8. **Query Hit, Pong**
       Rubric: 1 point per answer
    9. **Detection Timeout** (for gossip heartbeat), T_fail , timeout
       Rubric: T_timeout +1
    10. **Globus**
        Rubric: Inter-site protocol +1 or minor errors in spelling

2. (Solution and Grading by: Kshitij+TAs.)

a)



b) The Last element of the Vector Timestamp assigned to M44 would be 6, because there are 6 events at P4, and vector timestamps rules say that the last entry (its own entry) at P4 is incremented by +1, whenever any event occurs at P4 (regardless of whether the event is a send, receive, or instruction)

Rubric: Part a worth 14 points and part b worth 6 points

- -2 for every incorrect/missed timestamp (Check where the error is stemming from and don't double penalize for the same mistake)
- -8 (max penalty for part a (for a valid attempt))
- -14 (if part a blank)
- -4 for incorrect answer (part b) (with some explanation)
- -6 if left blank or wrong answer with no justification (part b)

3. (Solution and Grading by: Christina+TAs.)

a. (-1 for every incorrect entry) Finger table entries for machine with peer id 985:

i		ft[i]
0	985 + 2^0 (mod 1024) = 986	0
1	985 + 2^1 (mod 1024) = 987	0
2	985 + 2^2 (mod 1024) = 989	0
3	985 + 2^3 (mod 1024) = 993	0
4	985 + 2^4 (mod 1024) = 1001	0
5	985 + 2^5 (mod 1024) = 1017	0
6	985 + 2^6 (mod 1024) = 25	29
7	985 + 2^7 (mod 1024) = 89	169
8	985 + 2^8 (mod 1024) = 217	408
9	985 + 2^9 (mod 1024) = 473	0

b. (-2 for every incorrect hop) Here is the routing for file id 30 starting from peer id 169: **169 => 985 => 29 => 70**

The steps are shown in detail below:

First, look at the finger table for peer id 169 and hop to peer id 985:

i		ft[i]
0	169 + 2^0 (mod 1024) = 170	408
1	169 + 2^1 (mod 1024) = 171	408
2	169 + 2^2 (mod 1024) = 173	408
3	169 + 2^3 (mod 1024) = 177	408
4	169 + 2^4 (mod 1024) = 185	408
5	169 + 2^5 (mod 1024) = 201	408
6	169 + 2^6 (mod 1024) = 233	408
7	169 + 2^7 (mod 1024) = 297	408
8	169 + 2^8 (mod 1024) = 425	985

9	169 + 2^9 (mod 1024) = 681	985

Then, look at the finger table for peer id 985 (from part a) and hop to peer id 29. Finally, look at the finger table for peer id 29 and notice that there is no entry that is less than 30. Therefore, hop to the next machine, machine with peer id 70 to route the message

c. (-2 for every incorrect node) Just one node needs to change its finger table: **985**

4. (Solution and Grading by: Lilia+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=hour, 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
```

Rubric:

<u>Calculation-based deductions</u> (should only receive one of these)

-10 for incorrect interpretation/calculation of average that doesn't have a sensible interpretation, makes problem much easier. Common wrong solution calculates argmax_hr of #likes(hr)/#posts(hr)

-5 incorrect interpretation/calculation of average but with sensible interpretation / preserving similar difficulty. Common wrong solution returns the hour where posts that were liked in that

hour had the highest average likes during that hour. In other words, argmax_hr of \average_{posts liked in hr} {#likes(p | hr)/#posts(hr)}
-3 for careless mistake of finding argmax of an incorrect calculation of average, and getting the correct calculation would not involve any change to the structure of the MR program. For e.g. getting argmax of the sum but the # posts was available in the last MR
-5 for other calculation errors
<u>Other</u>
-5 for major errors revealing fundamental misunderstandings of MapReduce. This can be deducted multiple times, but only once for each type of mistake. Mistakes include but are not limited to

- Map not paired with a reduce, or reduce uses different values than the feeding map
- Map performs operations dependent on more than one input
- Reduce performs operations dependent on data from different keys
- Not reading both datasets, not reading them in separate MRs

-4 for lack of parallelism (using <=3 MRs)
-1 for other minor errors

5. (Solution and Grading by: Taksh+TAs.)

a. M = 2k + 1
   Completeness is violated when there is a possibility that M simultaneous failures may not all be detected. So, in this scenario, if M simultaneous failures occur, and all the selected targets of the process i (including predecessors, successors, and random processes) fail to receive heartbeats, then not all of these M simultaneous failures may be detected. For this scenario, p(i) itself fails, it's k predecessors fail and its k successors fail simultaneously.
   Hence M = 1 + k + k = 2k + 1

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

c. Worst case : N-1
   A process might have to send heartbeats to all other processes based on the predecessor, successor and randomly selected processes.

   Best case : 2k
   This scenario can happen when there are no processes to randomly select from after we have selected k predecessors and k successors.

   Average case : 3k
   On an average, a process will select k predecessors, k successors and k randomly selected processes to send heartbeats to.

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 B SOLUTION ===============