HW1 Solutions: CS425 FA15

1. (Solution and Grading by: Ayush Jain)

x: minimum delay from network interface to server in microseconds
y: minimum delay from server to network interface in microseconds

Minimum Client->Server latency (min1) = 13.7+x
Minimum Server->Client latency (min2) = 19.3+y
RTT = 1290

Error \leq (RTT-min1-min2)/2
     = (1290-13.7-x-19.3-y)/2
     = (1290-33-x-y)/2
     \leq 1257/2
     = 0.6285 ms

2. (Solution and Grading by: Qi Wang)
4. (Solution and Grading by: Guangxiang)

Process state:
P0: S(b)
P1: Initial State
P2: S(c)

Channel state: (Cij mean channel from process i to process j)
C10: empty
C20: c
C01: b
C21: a
C02: empty
C12: empty

5. (Solution and Grading by: Guangxiang)
   a. reduce task
   b. NM
   c. reduce task
   d. RM
   e. AM
   f. map task
   g. RM
   h. RM
   i. map task

6. (Solution and Grading by: Ayush Jain)
The solution below chains two map-reduces.

**Map1:**

**Input:** \((a, b)\) tuples where \(a\) follows \(b\)
Output: \((a, ('follows', b)), (b, ('is followed by', a))\)

**Reduce1:**

**Input:** \((a, \text{ all value}(a))\) where \text{value}(a) is either of the form \( ('follows', b)\) or \( ('is followed by', b)\); \text{all value}(a) is the collection of all values associated with a

```python
numFollowers = 0  # # of followers of a
numFollowed = 0   # # of people followed by a
for val in all value(a):
    if val[0] == 'follows':
        numFollowed += 1
    else:
        numFollowers += 1
if numFollowers > 1 million:  # a has > 1 million followers,
    # satisfies condition (1)
    for val in all value(a):
        if val[0] == 'is followed by':
            # If 'a' has more than 1 million followers, everyone
            # who follows 'a' satisfies condition (3)
            output((val[1], 'follows someone with million followers'))
    if numFollowed >= 10:  # a also satisfies condition (2)
        output((a, 'has million followers and follows at least 10 people'))
```

**Map2:** (Identity)

**Input:** \((a, s)\) outputs from Reduce1 where \(s\) is either \( 'is followed by someone with million followers'\) or \( 'has million followers and follows at least 10 people'\)

**Output:** same as above

**Reduce2:**

**Input:** \((a, S)\) where \(S\) is the collection of all values associated with a

**Output:** \(b\) such that \(b\) is 'similar' to Hillary Clinton

```python
if 'follows someone with million followers' in S and 'has million followers and follows at least 10 people' in S:
    output(a)
```

7. (Solution and Grading by: Yi Zhang)

a. No, the algorithm is not complete. In the worst case, all \(N\) processes may end up selecting the same \(k\) processes to heartbeat from, say \(p_1\) through \(p_k\). This means \(p_{k+1}\) through \(p_N\) do not have anyone receiving heartbeats from them, and their failure(s) will not be detected. Concretely, suppose there are 5 processes in the system and \(k\) is 2.

b. No, the algorithm doesn't satisfy accuracy. No heartbeating protocol is accurate, because one cannot distinguish between a failed process and a process whose heartbeats are being lost.
8. (Solution and Grading by: Yi Zhang)

The underlying bug is likely that the protocol does not implement the $2^\ast T_{\text{fail}}$ wait policy, i.e., it deletes the entry pointing to the failed node right after $T_{\text{fail}}$. Suppose $P_j$ is dead and $P_i$ immediately deletes it after $T_{\text{fail}}$. $P_j$ may receive gossip about $P_j$ from other nodes. $P_i$ will reinstall $P_j$ to its membership list since it thinks of $P_j$ as a new node. Then, $P_i$ will gossip this information to other nodes. This situation may go back and forth so that $P_j$ will never get deleted from the membership lists.

9. (Solution and Grading by: Alex Zahdeh)

10. (Solution and Grading by: Alex Zahdeh)

Last timestamp should be $(5,10,11,7)$