HW2 Solutions: CS425 FA17

1. (Solution and Grading by: Faria)
   Formula: $h_i(x) = (i \cdot x^i - 2^x) \mod m$
   $m = 64$

<table>
<thead>
<tr>
<th>2</th>
<th>2</th>
<th>2</th>
<th>2</th>
<th>2</th>
<th>2</th>
<th>2</th>
<th>2</th>
<th>2</th>
<th>2</th>
<th>2</th>
<th>2</th>
<th>2</th>
<th>2</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

   h1:  31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13
   h2:  0  4 12 24 40 60 20 48 16 52 28  8 56 44 36 32 32 36 44
   h3: 33 20 43 56 13 60 23 48 25 36 35 40  5 12 15 32 17 52 27
   h4:  2 60 62 56 58 52 54 48 50 44 46 40 42 36 38 32 34 28 30

   Program terminates on 2035.

   Rubric:
   No calculation, no table of values, and incorrect answer but describes the condition of false positive correctly → -5
   No calculation but table of values, incorrect answer → -2
2. (Solution and Grading by: Faria)
   The probability \( p \) of a false positive is approximated by the following formula, where \( k \) is the number of hash functions, \( n \) is the size of the input set and \( m \) is the size of the Bloom filter in bits:
   \[
   p = \left( 1 - e^{-\frac{kn}{m}} \right)^k
   \]

   Case 1: 5 inputs
   Original bloom filter with 1024 bits and 4 hash functions. The formula can be applied directly.
   False positive rate: \( 1.39953299 \times 10^{-7} \)
   Orlando’s bloom filter:
   In this case, there is only a false positive if all 4 bloom filters report that the number is present. Thus,
   \[
   \text{False positive rate: } (1 - e^{(-15/256)})^4 = 1.39953299 \times 10^{-7}
   \]
   They both have a similar false positive rate.

   Case 2: 100 inputs
   Original bloom filter:
   False positive rate: 0.01093397922
   Orlando’s bloom filter: \( (1 - e^{(-100/256)})^4 = 0.01093397922 \)
   They both show the same false positive rate.

   Second formula is also acceptable:
   Input: 5
   Original: 1.40224356e-7
   Orlando’s: 1.4104201e-7
   Original is better.

   Input: 100
   Original: 0.0109514547
   Orlando’s: 0.01100412416
   Original is better.
**Rubric:**  
No conclusion → -1  
Per every incorrect value of k,n,m → -2  
Incorrectly copied formula → -2  
Answer with no calculations/formula → -5  
Did not account for 4 bloom filters in Orlando’s proposal → -2
3. (Solution and Grading by: Jayasi)

- **Sequential Consistency**: Requires that all the operations appear to be executed in some sequential order that is consistent with the order seen at individual processes.
- **Linearizability**: When the order above also preserves the global (i.e., wall-clock) ordering of operations.
- **Causal consistency**: Requires that the effects are observed only after their causes - reads will not see a write unless its dependencies are also seen.

In the above figure, the following values of X could be observed by \(R_{\text{final}}\) (and still satisfy the consistency model) based on the consistency model used:

- **Sequential Consistency**: 43 (set by \(W_3\)) (Since \(W_1\) happens before \(W_3\) in process A, \(R_{\text{final}}\) can observe the value 43 while respecting the per process local ordering)
- **Linearizability**: 94 (set by \(W_4\)) (Following the global clock, the most recent write)
- **Causal consistency**: 32 (set by \(W_2\)) (The causal ordering is preserved by reading the value set \(W_2\))
4. (Solution and Grading by: Rui)

We assume the latency of sending a message as $L$ in child clock.

We denote the time drift as $d = \frac{\text{parent time frequency}}{\text{child time frequency}} - 1$,

\[
\begin{align*}
\text{Parent process} & \\
\text{Child process} & \\
\end{align*}
\]

Split the $ts1 \sim tr2$ period into three parts as above diagram, where

\[
\begin{align*}
a &= L & (1) \\
b &= (ts2 - tr1)/(d + 1) & (2) \\
c &= L & (3) \\
\end{align*}
\]

On the other hand, $a + b + c = tr2 - ts1 \Rightarrow 2L + (ts2 - tr1)/(d + 1) = tr2 - ts1 \quad (4)$

For the first message: $tr1 = ts1 + L \times (d + 1) \Rightarrow L = (tr1 - ts1)/(d + 1) \quad (5)$

With (5) substituted into (4), we could get:

\[
\begin{align*}
d &= (tr1 + ts2 - 2ts1) / (tr2 - ts1) - 1 \\
\text{If we treat } ts1 = 0, \quad d &= (tr1 + ts2)/tr2 - 1
\end{align*}
\]
5. (Solution and Grading by: Xiaoyao)

From Cristian's algorithm, error <= (RTT - min1 - min2) / 2
where
Min1 = client’s APP to NIC time + fixed transmission time + server’s NIC to APP time
Min2 = server’s APP to NIC time + fixed transmission time + client’s NIC to APP time

If we denote:
- the fixed transmission time to be p,
- the unknown client’s APP to NIC time as x,
- the unknown client’s NIC to APP time as y,

then we can simplify the error bound to be:
error <= (1.93 - (x + p + 55.6/1000) - (0.21 + p + y)) / 2
error <= (1.93 - 55.6/1000 - 0.21 - (2p + x + y)) / 2 <= (1.93 - 55.6/1000 - 0.21) / 2 = 0.8322 ms

Therefore, the error is bounded by 0.8322ms (we assume min bound to be 0)
6. (Solution and Grading by: Ashwini)
   a. Size of the vectors grows proportionally to the numbers of clients and servers. This can take up a lot of space as well as take longer to compute timestamp comparisons.
   b. If the system runs long enough, vector clock values may become too large and overflow. Some vector clock entries may need to be reset or reduced without violating causality. (We will accept other reasonable pros/cons too, though we expect everyone to catch the first issue above.).

Solution to above problems is to prune the vector clocks as they grow.
   - Prune vector clocks: If the vector clock becomes too big, then some of the old entries could be removed by looking at their timestamp values. For more details on this, see Riak’s (past) use of vector clocks and their pruning techniques: [http://basho.com/posts/technical/why-vector-clocks-are-hard/](http://basho.com/posts/technical/why-vector-clocks-are-hard/)
   - Add a timestamp to each clock field and update it to current local time whenever the field is incremented.

7. (Solution and Grading by: Ashwini)
Rubric - If there's mistaken entry, deduct one point, and continue to check student's solution treating the mistaken entry as correct (to avoid double penalties) - multiple points deducted for causally unrelated mistakes
9. (Solution and Grading by: Rui)

We denote the quorum size as \( Q \), satisfying

\[
2Q - N \geq K \Rightarrow Q \geq \frac{N + K}{2} \Rightarrow Q_{\text{min}} = \lceil \frac{N + K}{2} \rceil
\]

a. 51
b. \( \lceil (N + 1)/2 \rceil \)
c. \( \lceil (N + 10)/2 \rceil \)
d. \( \lceil 2N/3 \rceil \)
e. \( \lceil 3N/4 \rceil \)
f. \( \lceil 5N/6 \rceil \)
g. \( N \)
The new timestamps still preserve causality.

Preserving causality means $E_1 \rightarrow E_2 \rightarrow \text{timestamp}(E_1) < \text{timestamp}(E_2)$. Since there’s either at least one message sent or instruction on the path from $E_1 \rightarrow E_2$, and timestamps of events on the path only monotonically increase, timestamp($E_1$) must be less than timestamp ($E_2$).

When we can draw a path of events that follows some order, their timestamp values will always be increasing as long as it follows the rules that:

1. A process increments its counter when a send or an instruction happens at it.
2. A send (message) event carries its timestamp
3. For a receive (message) event the counter is updated by $\text{max(} \text{local clock, message timestamp} \text{)} + (i+1)$.

You can also prove this by contradiction, similar to the proof in class.