ECE 454 Assignment 6

Amir Benham 20393292, Andrew Svoboda 20369388

1. We are going to break this up in to three cases: alternating reads and writes, series of reads followed by a write and a series of writes followed by a read. We believe that any sequence can boiled down to some combination of these cases.

   Case 1: Alternating Reads and Writes

   $$W(X1) \rightarrow R(X1) \rightarrow W(X2) \rightarrow R(X2) \rightarrow W(X3) \rightarrow R(X3)$$

   Since we are reading the value after every write, we know that the changes must have propagated due to read your write consistency. Therefor we know each write will be write on the most recent therefor write follow read will be true in this case. Therefor this case would infer write follow read.

   Case 2: Series of Reads Followed by Write

   $$W(X1) \rightarrow R(X1) \rightarrow R(X1) \rightarrow R(X1) \rightarrow W(X2) \rightarrow R(X2) \rightarrow R(X2) \rightarrow R(X2)$$

   In this case the series of read are constantly reading the same value which causes no new propagation happens after the first read. Therefor it is trivial to reduce this case to case 1.

   Case 3: Series of Writes Followed by Read

   $$W(X1) \rightarrow W(X2) \rightarrow W(X3) \rightarrow R(X3) \rightarrow W(X4) \rightarrow W(X5) \rightarrow W(X6) \rightarrow R(X6)$$

   Write follow read is only concerned with the first write after a read. We know from read your write that after doing a read the most recent write will be propagated therefore the first write after a read will write the most recent value. The writes after that may not propagate in order by nether read your write or read follow read is concerned about that. Therefor this case would infer write follow read.

   Based on these cases we believe that read your write would infer write follow read.

2. Proof by counter example.

   $$W(X1) \rightarrow R(X1) \rightarrow W(X2) \rightarrow W(X3) \rightarrow W(X4) \rightarrow R(X2|X3|X4)$$

   In the example above we know that W(X2), W(X3), W(X4) will all at least write on the value of X1 but we have no guarantee that tells which one will propagate last. There for we cannot guarantee that the next read will read X4 therefor it cannot infer read your writes.

3. (a) To find a global minimum, we check by brute force:

   For every server of every ordering: open it and assign all clients to it

   pick another server from the ordered list, compute the cost for all clients to connect to this server compare the cost of the clients current cost to their server to this new server sum all the decreases in cost, and compare the sum to the cost to open this new server if the saved cost sum is greater than the cost to open the server, open the server

   thus we have a time complexity of c*(f!), where we must iterate over every client for every ordering of all servers, assuming summation and comparisons are insignificant in time complexity

   (b)

4. need to check every set of k servers, that is, if there is no cost to open a server, you minimize the cost by picking k servers. but you have i servers, so then its "i choose k" possibilities, and you need to check all those. that seems really high and is not efficient (factorial is non polynomial time... so eh?)

5. (a) Never drop. Because we aren't given quantitative specs for space requirements, we will assume that the file size is negligible compared to the full size of the data store, and thus the file should always be replicated. we also aren't given specs on synchronization or how often we would update the file (and thus incur a cost of e to replicate the file).

   (b) To find an expression for replicate threshold we need to look at the difference in cost between q and p with respect to cost of replicating. Which leads us to get $n_1 + n_2 > \frac{e}{q-p}$

6. (a) Each write must utilise at least $> \frac{N}{2}$ servers. We know thus that a majority of servers must have the most recent version of item x. To read, we need $N_r + \frac{N}{2} > N$, which implies we must have some overlapping set of servers with respect to those that have the most recent version. ie, 10 servers, 6 must write, and we need x + 6 ¿ 10, so we pick at least 5 servers, implying we have at least 1 with the new data.

   (b) We have causal consistency across all the servers with respect to the data item x read/written into the system. However, on an individual server level we do not have causal consistency. We have have the case where data item x is written once to a majority of servers, and then for a long time only overwrites one of the servers across many more writes. These servers that are ignored do not demonstrate causal consistency as the data writes are sometimes missing from each server, and so the ordering can be different for all servers.