CS384 - Qualifying Exam

Problem 1 Suppose we have a distributed system with three independent processes given in Figure 1. The horizontal lines indicate the time-line associated with each process, and the dot indicates an event occurred.

- Initially, the system's vector clock is $\vec{0}$. Then what is the vector clock value for every event in the system?
- According to Figure 1, is $c \prec b$? Are b and g concurrent? Justify your answers.

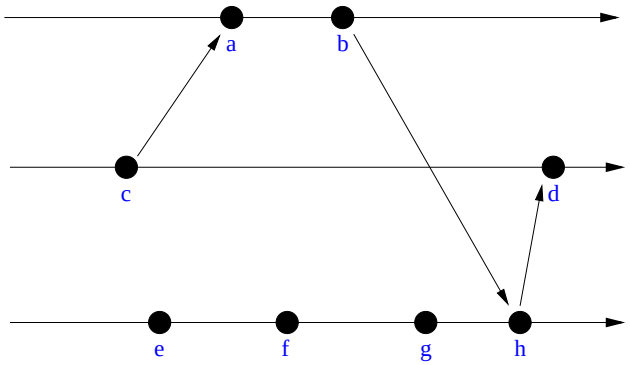

Figure 1: A distributed system with three processes

Problem 2 A quorum-based replica management system consisting of N servers engages only a designated fraction of the replicas for every read and write operation. These are known as read quorum (R) and write quorum (W). Assume that N is an even number.

- Will sequential consistency be satisfied if $W = \frac{N}{2}$ and $R = \frac{N}{2}$?
- Will sequential consistency be satisfied if $W = \frac{N}{2} + 1$ and $R = \frac{N}{2} + 1$?
- If $N = 20$ and $W = 15$, then what should be the minimum value of R so that sequential consistency is satisfied?

Problem 3 Suppose we have the following algorithm to reach an agreement in the Byzantine general's problem: (1) the commander sends a value to very lieutenant, (2) each lieutenant sends the received value to the other lieutenants, and (3) the final outcome of each lieutenant is the majority value of his or her received values.

- Show a situation where the proposed algorithm does not work when there are 7 generals and 2 of them are traitors. You can assume that the only faulty behavior is that the traitors can alter the messages.
- Using the same example, show, step by step, how Lamport's algorithm allows the generals to reach an agreement.

Problem 4 Mutual exclusion is important when a process in a distributed system enters its critical section. Under this context, answer the following questions:

- What is the safety property? What is the liveness property?
- Suppose we have the following distributed algorithm to guarantee mutual exclusion (where CS denotes critical section):

  1. Broadcast a time-stamped ***request*** to all
  2. When ***request*** received, enqueue it in a local Q. If not in CS, send ***ack.***, else postpone sending ***ack.*** until exit from CS
  3. Enter CS, when (a) you are at the head of your Q and (b) you have received ***ack.*** from all processes in the system
  4. To exit from CS, (a) delete the ***request*** from your Q, and (b) broadcast a time-stamped ***release***
  5. When a process receives a ***release*** message, it removes the sender from its Q

  Prove that at most one process can be in its CS at any time. Also prove the safety property and the liveness property.