Midterm Review Problems

Notes: The midterm will be closed book, closed note. It will last 75 minutes. The midterm paper will contain the following two types of questions - multiple choice (one correct answer), and longer questions (either fill in the blanks or design questions, with the former dominating).

You can expect questions of the format asked in the homeworks, or in the format below. Most questions will be about running a protocol discussed in class; there might be some design questions too.

Numbers and length of actual questions may be different on the actual midterm. Solutions to this review will not be available, but you are free to discuss solutions with other students.

Study Material: Everything covered during Lectures 1-12, Homeworks 1-2.

1. [Many such questions – Choose the one correct answer] In synchronous systems,
   (a) message delays are unbounded
   (b) the time for a process to execute an instruction is unbounded
   (c) processes may crash
   (d) none of the above.

2. [Several such questions – Larger Problem: Application] Given a Chord overlay with m=7 and node IDs of 1, 2, 3, 4, 11, 22, 33, 44, 100, 111, then
   1. The fifth finger table entry of node 3 is ___________
   2. The number of nodes with 3 in their finger tables is ___________

3. [Few such questions – Larger Problem: Design] Consider a mesh-like overlay with m rows and k columns. Each of the mk points in the mesh contains exactly one process, so N = mk. Each processID consists of two integers - (column number,row number)=(c,r). Unicast communication from each process (c,r) is possible only to its directly adjacent processes, called “neighbors”; these are (c-1,r),(c+1,r),(c,r-1),(c,r+1). If a neighbor process does not exist (as when the index is out of bounds) or has crashed, no communication is possible with it.
   This is depicted in the figure (where processIDs appear below and to the left of the mesh point).
   Each link in the system may arbitrarily delay a message (but the delay is finite). Links may not preserve FIFO ordering of messages. Unless otherwise specified, assume that processes do not crash.
   Answer the following questions:
   1. If m is an even integer, give pseudocode for a token-based mutual exclusion protocol. What are the bandwidth, client delay, and synchronization delay for your protocol?
   2. The structure of the mesh allows us to design a more efficient protocol. Give pseudocode for an efficient non-token-based mutual exclusion protocol. What are the bandwidth, client delay, and synchronization delay for your protocol? (Clue: We’ve discussed a similar protocol in class.)
   3. Give a protocol for reliable multicast in this mesh overlay. If there are process crashes in the system, under what conditions will a multicast initiated by non-faulty process (0,0) reach all non-faulty processes?
4. [Few such questions – Larger Problem: Application + Design] For the same conditions and assumptions described in Problem 2, answer the following questions:

1. Suppose files were stored in this system by first using two different hash functions SHA-1 and MD-5 to hash the filename to a row number and column number (within the above limits) respectively, and then by storing the file at the process residing at that mesh point. Give a protocol to route a query for a filename \( f \). In your protocol, the worst case time to route a query if \( m = k = 20 \), and each link has a maximum delay of 100ms, is ____________

2. Can your protocol from the previous question be used to route from any process to any other process when there are no process crashes? If no, say why. If yes, how would you modify the protocol so that it works in spite of process crashes (pseudocode not necessary)? How efficient is your protocol (worst case latency and worst case message complexity)?

3. Assume that process \((0,0)\) never crashes. If you are given that at most one other process crashes, design a protocol to run at all processes so that when the first process crash occurs, process \((0,0)\) eventually knows about this crash. 
(Optional) How would you modify your protocol if at most 2 processes crash (process \((0,0)\) never crashes).

Figure 1: Figure for Questions 2,3