UNIVERSITY OF EDINBURGH

FACULTY OF SCIENCE AND ENGINEERING

MASTER OF SCIENCE IN COMPUTER SCIENCE

DISTRIBUTED SYSTEMS

Tuesday 24 April 2001

09:30 to 11:00

Examiners:

I. Watson (External)

R.N. Ibbett (Chairman)

INSTRUCTIONS TO CANDIDATES

Answer any TWO questions.

All questions carry equal weight.

1. Consider a distributed system with several time servers which are able to receive coordinated universal time (UTC). These time servers exchange their UTCs periodically and calibrate the discrepancy between them. Clients of the time service can contact one or more time servers to get up-to-date clock information.

   (a) Explain briefly why it is not possible to keep all of the clients in perfect synchronisation even if all of the time servers are assumed to be consistent. [*5 marks*]

   (b) Consider the case where a client sends a request to a time server and the time server replies by sending its current time. Present a simple method for the client to adjust its clock. You can assume that the network traffic is symmetric and the time for a time server to process the request is estimated to be t_p. [*5 marks*]

   (c) Give two examples of problems which may arise if a clock is adjusted abruptly. One of the examples should address the clock being moved forwards and the other one should address the clock being moved backwards. [*6 marks*]

   (d) One of the algorithms for enforcing at-most-once message delivery to a server makes use of synchronised clocks. Processes include their local clock value (T_{local}) as a timestamp in the messages they send. For each sending process, the server stores in a table the largest timestamp ($T_{largest}$) which it has received from that process. Let T_{max} be the maximum life time for a message and T_{drift} be the maximum clock drift between clocks.

      (i) How can duplicate messages be detected by the server? [*4 marks*]

      (ii) When can a timestamp T be safely removed from the table? [*5 marks*]

2. In distributed systems the correct behaviour of the Bully algorithm depends on the following assumptions about failure and delay.

   1. All process identifiers are unique and known to every process.
   2. Messages are never lost, corrupted, or duplicated and are always delivered in order in a known finite time.
   3. A process takes a finite amount of time to handle a message.
   4. Processes may fail. A failure causes the computation to stop and failed processes do not generate erroneous messages.
   5. When a process recovers, it knows that it has experienced a failure.

   (a) Two of the assumptions above allow you to build a reliable failure detector. Which are these two assumptions and how would you build the failure detector? [*5 marks*]

   (b) Suppose that there are four processes in the system with process identifiers p_1, p_2, p_3, p_4 where p_4 is the leader. At some point in the computation, p_2 detects the failure of p_4 and initiates the election of a new leader. Describe the Bully algorithm by referring to this example. You can assume that no other process fails during the election. [*5 marks*]

   (c) Suppose that there are N processes in the system.

   (i) What is the worst case for the number of messages required for the algorithm to complete? [*3 marks*]

   (ii) What is the best case for the number of messages required for the algorithm to complete? [*3 marks*]

   (d) If assumptions 2 and 3 do not hold, what can go wrong with the Bully algorithm? How can you adapt the Bully algorithm to this case? [*9 marks*]

3. Multicast operations provide a natural mechanism for implementing group communication in distributed systems. A message can be sent to all of the processes in a group by issuing a single multicast operation.

   (a) It would be ideal to have all of the multicast messages delivered instantaneously and in the same order that they were sent. However, this is not realistic in practice. Why? *[5 marks]*

   (b) It is possible to define the semantics of a multicast operation so that the requirements for message delivery order are less strict.

   - FIFO order: Multicast messages from a single source are delivered in the order that they were sent.
   - Causal order: Causally related messages from multiple sources are delivered in their causal order.
   - Total order: All messages multicast to a group are delivered to all members of the group in the same order.

   Which one is the least difficult to implement? Which one is the most difficult? *[2 marks]*

   (c) Sketch an algorithm for implementing message delivery with causal ordering. You can assume that the source of the multicast is a member of the group and that groups are non-overlapping. *[8 marks]*

   (d) How can you implement FIFO order for overlapping groups? Consider two messages sent from the same source to a process in the intersection of these groups. *[10 marks]*