NEWCASTLE UNIVERSITY

SEMESTER 1 2009/10

DISTRIBUTED SYSTEMS

Time allowed - 1½ Hours

Instructions to candidates:

Answer any TWO questions

Marks shown for subsections are indicative only

Question 1.

a) Define the possible RPC semantics that can be obtained in the presence of server crashes and lost messages. How can the *at most once* semantics be obtained using *call sequence numbers*? [8 marks]

b) Define the *happened before* relation, denoted usually as $\rightarrow$, on a set of events in a distributed system. Say a, b, c and d are events in a distributed system. Based on your definition of $\rightarrow$, explain whether it is possible to have the following: $a \rightarrow b$, $b \rightarrow c$, $c \rightarrow d$, and $d \rightarrow a$. [7 marks]

c) Let C be some clock system implemented in a distributed system which satisfies: if $a \rightarrow b$ then $C(a) < C(b)$ where $C(a)$ and $C(b)$ are the times according to C when a and b occurred respectively. Show that the clock system C cannot always ensure the following inverse condition: if $C(a) < C(b)$ then $a \rightarrow b$. [10 marks]

Question 2.

a) Describe briefly two important activities carried out by stubs in enabling client-server interaction. [7 marks]

b) State the ACID properties attributed to a transaction. State the rules to be enforced for achieving (i) two-phase locking, and (ii) strict two-phase locking in transaction systems. Identify the ACID properties met by enforcing these sets of rules. [8 marks]

c) Let transactions T1 and T2 be defined as follows.

T1:	T2:
T1.1: x=x+y;	T2.1: y=x-y;
T1.2: z=z+10;	T2.2: q=z+x;
T1.3: z=z*5;	

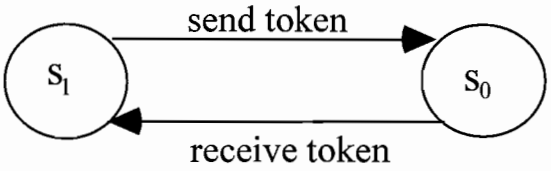
Let S1 and S2 be two schedules such that S1 = {T1.1; T1.2; (T1.3 || T2.1); T2.2} and S2 = {T2.1; T1.1; T1.2; T1.3; T2.2}, where T1.3 || T2.1 denote that T1.3 and T2.1 execute concurrently. For each schedule, show whether it can occur (i) under two phase locking, and (ii) under strict two phase locking. [10 marks]

Question 3.

a) What is a stable property? Give an example of a stable property in the context of a distributed computation. [4 marks]

b) Describe the algorithm by which each process in a distributed system should record information about its local state so that the global state of the system can be composed. Point out the requirements necessary for the termination of this algorithm. [8 marks]

c) Consider a single-token system that is made up of two processes p and q. A process is said to be in state s_1 or state s_0, if it has the token or does not have the token, respectively. The state transition diagram for a process is given in the figure below.



Let the token be initially with process p and the sequence of events that occur during a computation be denoted as: e_0, e_1, e_2, ... Thus, the event e_0, the first event of any computation, has to be "p sending the token to q". Let q record its state after e_0 and before e_1, and p record its state only after being prompted by q. Assuming that the execution of the algorithm terminates, determine the global state according to the information recorded by processes p and q. Did this global state actually occur during the execution of the algorithm? If not, determine the equivalent computation in which the recorded state occurs.

[13 marks]