**NEWCASTLE UNIVERSITY**

**SEMESTER 1 2018/2019**

# DISTRIBUTED ALGORITHMS

Time allowed - 1½ Hours

**Instructions to candidates:**

Answer ALL questions

The total marks available for this exam are 100

Marks shown for sub-sections are indicative only

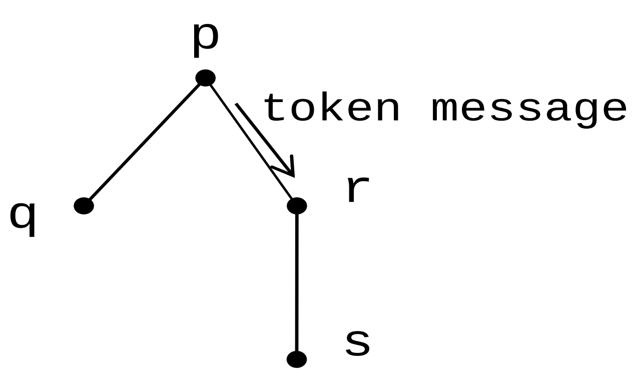
*[Turn Over]*

## Question 1

a) Define a *precedence* relation by which two events that occur during a distributed computation may be ordered. [6 marks]

b) Describe the wave algorithm for trees. Indicate the diffusion part of the algorithm. [10 marks]

c) Suppose that four processes, namely p, q, r, and s, form a tree with p as the root and execute the wave algorithm. The figure below depicts the tree. At some time, say, T during an execution of the algorithm, the only message that is in transit in the entire system is the token sent by p to r. (The figure also portrays this situation at time T.)



i) Assume that no process has decided at time T. Describe an execution that could have taken place until time T and, ignoring the diffusion part of the algorithm, show how the execution continues until two processes decide. [14 marks]

ii) Investigate whether any process could have decided before time T, given that the token from p to r is the only message in transit at time T. Ignore the diffusion part in your investigation. [14 marks]

Hint: A possible line of investigation is to consider that the token from p to r takes a long transmission time, much longer than the time the token from r may take to reach p.

d) Your colleague, Mr Paul Errorprone, claims that processes p and q decide in his answer to part i) of c). Explain to Paul that his answer cannot be correct. [6 marks]

## Question 2

a) What are the assumptions normally made on the *Trusted Third Party* (TTP) of a fair-exchange protocol? How are the fair-exchange protocols classified based on the manner they use the TTP? [10 marks]

b) Presented below are the five steps of a protocol P which a player A uses to send a digital item M to another player B and to obtain a non-repudiation of receipt (NRR) from B. Show that P *cannot* be fair and fix all its flaws so that P becomes fair. [18 marks]

A $\rightarrow$ TTP: (A, B);
(* A informs TTP of its fair-exchange partner*)
TTP $\rightarrow$ A: (A, B, TTP, K, EOO);
A $\rightarrow$ B: $M_A$ = {$e_K$(M), $Sig_A(H(e_K(M)))$};
B $\rightarrow$ TTP: $Sig_B(H(M_A))$;
TTP $\rightarrow$ B: (A, B, TTP, K, EOO);
TTP $\rightarrow$ A: $Sig_B(H(M_A))$;

Notations used bear their conventional meaning:

- $Sig_A(\Phi)$ is the unique and verifiable signature of A for message $\Phi$;
- $e_K(\Phi)$ is the encryption of $\Phi$ with key K;
- H is a one-way and collision-resistant hash function; and,
- EOO (Evidence Of Origin) is $Sig_{TTP}$(A, B, TTP, K).

c) Consider a transaction involving objects hosted in two servers, S1 and S2. At the end of the transaction, two-phase commit protocol is executed with the client C acting as the coordinator. Suppose that C crashes before it completes its execution of the protocol and that S1 and S2 do not crash.

   i) Describe an execution in which S2 decides on commit and S1 remains blocked (i.e., remains undecided) until C recovers from its crash. [8 marks]

   ii) Describe an execution in which S2 decides without entering the second phase and S1 remains blocked until C recovers from its crash. [6 marks]

d) It was discussed in the lectures that an execution of the two-phase commit protocol is *guaranteed* to terminate only if there exists a moment when all crashed nodes have recovered and nodes do not crash thereafter. Justify this termination requirement by producing an execution in which the coordinator and one server never stop crashing, i.e., keep crashing after each recovery, and the server never decides. [8 marks]

**END**