UNIVERSITY OF LONDON

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1999

MEng Honours Degrees in Computing Part IV
MEng Honours Degree in Information Systems Engineering Part IV
MSci Honours Degree in Mathematics and Computer Science Part IV
MSc Degree in Advanced Computing
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the
Diploma of Membership of Imperial College
Associateship of the City and Guilds of London Institute
Associateship of the Royal College of Science

PAPER 4.37 / I 4.12

DISTRIBUTED ALGORITHMS

Friday, May 7th 1999, 10.00 – 12.00

Answer THREE questions

For admin. only:
paper contains 4 questions

1 In a distributed system of connected processes, a *logical ring* can be used to resolve conflicts and assign some privilege to one process.

a Describe the main principles involved in logical ring algorithms, indicating the particular *safety* and *liveness* properties which the algorithms should satisfy.

b The dining philosophers problem illustrates conflict between neighbouring philosophers in their use of a shared fork. A logical ring between each pair of neighbouring philosophers can be used to resolve each fork conflict. Thus a ring of N philosophers will actually consist of N logical rings.

Since a philosopher node may be involved in thinking when a fork is passed by its predecessor, it is suggested that each node is composed of two processes, one to perform Philosophising (**Phil**) and the other to perform the fork rings processing (**Servant**). When hungry, a Phil process would then be required to request forks from its servant process.

i) Outline the relevant configuration for the ring of philosophers system.

ii) Using message passing, outline the relevant code for the Phil and Servant processes at each philosopher node. Briefly describe how your processes work. You may assume no loss of messages.

iii) Does this solution satisfy the required safety and liveness properties?

The two parts carry, respectively, 20%, 80% of the marks.

2. A system consists of a number of distributed nodes each of which needs *exclusive* access to a particular resource at various times.

a. Lamport's algorithm for mutual exclusion uses three types of timestamped message: request, release and acknowledgement. Each process maintains Q, a local view of the state of all processes. In a system of N processes, outline the following information for process Pi:

i) the information stored in Q and its initial value,

ii) the condition required for exclusion,

iii) the actions of a process on receipt of a timestamped request message.

What is the communications overhead in order for a process to gain exclusive access?

b. Lamport's algorithm (part (a)) requires that there be *no overtaking* of messages between sites. Show a scenario where, if two processes P0 and P1 both send request messages, and overtaking of an acknowledgement message is possible, it can lead to a violation of mutual exclusion. Indicate the state of the local Q's in each process at each stage.

c. Ricart and Agrawala's algorithm does not require that there be no overtaking of messages between sites. Give an outline description for this algorithm, indicating the types of message used and the information that is stored at each site.

What is the communications overhead in order for a process to gain exclusive access?

The three parts carry, respectively, 35% , 30% , 35% of the marks.

Turn over ...

3 a i) What are the properties describing the failure-free behaviour of *processes* and *communication links* in point-to-point networks? State which of them is/are *safety* and which is/are *liveness* properties.

ii) In terms of these properties, describe briefly what we mean by "process failure" (crash) and "link failure" (message loss), in point-to-point networks.

b i) What are the three properties of distributed systems said to be asynchronous?

ii) What are the three dimensions of System Models to be considered in Distributed Computing?

c i) In the case of the *Three-Phase Commit Algorithm* (3PC), any process participating to the algorithm can be in exactly one state at any time. The possible states are: "Uncertain", "Ready", "Committed", "Aborted". Some pairs of states cannot co-exist in processes participating in the algorithm. Fill in the cells of the table below with "Yes" or "No" to indicate whether the combination corresponding to that cell is possible or not, respectively. A cell has been completed as an example: if a process is in state "Aborted", then another process cannot be in state "Ready". (Reproduce the table in your exam book.)

Uncertain				
Ready				
Aborted		No		
Committed				
	Uncertain	**Ready**	**Aborted**	**Committed**

ii) Assume that while the *Termination Protocol* of *3PC* is being executed with process *i* as the coordinator (epoch *i*), process *i* crashes. Describe informally (in plain english) the subsequent steps of the Termination Protocol. Assume that no other process failures occur, for long enough for the protocol to terminate.

d i) What is the number of *rounds* and the number of *messages* required for 3PC, if no process and no link failures occur (best-case scenario) and the final decision is "1" (commit)? What are these numbers (rounds and messages), when the final decision is "0" (abort)? Provide exact numbers and not "big-O" notations.

ii) Assume that f process failures occur while 3PC is executed in a group of n processes ($f < n$). What is the worst possible number of rounds and number of messages required for the termination of the algorithm? (Assume that no link failures occur). When does this worst-case scenario occur? Provide exact numbers and not "big-O" notations.

[*Hint*: the provided number of messages will be an upper bound, thus it should be preceded by "≤"]

The four parts carry, respectively, 20%, 20%, 30%, 30% of the marks.

4 a Fill in the table below, using short phrases, with the *unsolvability conditions* (i.e. conditions that if they apply, then the problems are not solvable) of two fundamental computational problems of Distributed Computing: Consensus and "Strong" Atomic Commitment. (Reproduce the table in your exam book.)

	Consensus	**Strong AC**
Failure types		
Synchrony model		
Caused by...		

b i) What are the properties of the *"Weak" Consensus Problem*? State which of them is/are *safety* and which is/are *liveness* properties.

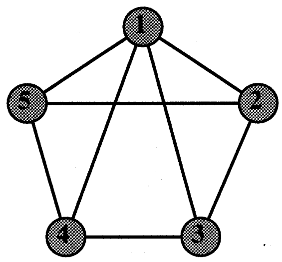
ii) What is the use of the "Weak" Consensus Problem in the proof of the FLP impossibility result? (Describe briefly).

c i) State the additional property that makes a Reliable Broadcast to be characterised as a FIFO Broadcast?

ii) State the additional property that makes a Reliable Broadcast to be characterised as a Causal Broadcast?

iii) Describe a simple algorithm (using pseudo-code description) which uses a FIFO broadcast to implement Causal broadcast (i.e. "transforms" FIFO to Causal broadcast). The presented algorithm does not have to be efficient. Does this transformation *preserve Total Order*? Why?

d Suppose you have a *synchronous, point-to-point* distributed system, with a topology as illustrated below (numbered circles depict processes, while edges depict bi-directional communication links). Assume the following system parameters:



The known upper bound on message transmission *delay* over a communication link that directly connects two processes is $\boldsymbol{\delta = 1}$ **ms**.

The maximum number of *faulty processes*, at any moment in time, in the system is $\boldsymbol{f = 1}$.

The maximum number of *faulty links*, at any moment in time, in the system is $\boldsymbol{k = 1}$.

i) Calculate the "worst shortest path" between any two processes for all possible combinations of 1 faulty process and 1 faulty link. d = ?

ii) Outline a simple algorithm (using pseudo-code description), which uses a Timed Reliable Broadcast to implement Atomic Broadcast, *for the above system.* Substitute Δ with its actual value (in *ms*), where appropriate in the algorithm.

The four parts carry, respectively, 20%, 20%, 30%, 30% of the marks

End of paper