CS 603 (Park) **Midterm** **March 19 (Wed.), 1997**

PROBLEM 1 (20 pts)

(a) What are the axioms that define the *happended before* relation "$\rightarrow$"? How can they be used to extend $\rightarrow$ to messages in a meaningful way (i.e., $m \rightarrow m'$)?

(b) What is the difference between a logical clock and a physical clock? Give two examples, one for each, where their unique traits cannot be easily replaced by the other.

(c) Explain the difference between total ordering and causal ordering of messages. How can one define an even weaker relation than causal ordering, albeit an impractical one, which captures the notion of "causality" more precisely?

(d) Define *serializability* in the context of transactions. In the case of complex transactions involving a large collection of variables/objects, what technique can be used to enhance concurrency (via interleaving) while still maintaining serializability?

PROBLEM 2 (20 pts)

(a) What is optimistic concurrency control and when (i.e., under what conditions) should it be used in place of conservative/pessimistic schemes? In what sense is the timestamp-based scheme (a form of optimistic concurrency control) superior than Kung & Robinson's original formulation (i.e., "traditional" optimistic concurrency control)?

(b) Define the notion of *resource graph* as used in deadline detection and explain its utility in completely characterizing deadlocks in terms of a graph property. What are *false deadlocks* and how can they arise in distributed systems? Give a solution which solves the problem.

(c) Explain the impossibility of the two-party consensus problem when the agents are assumed to be mistrustful of each other although each one is known not to cheat. How is it possible to use two-way handshaking (i.e., positive acknowledgement scheme) to achieve reliable message transmission in spite of the impossibility of consensus problem?

(d) What are the three properties a consensus protocol needs to satisfy and explain their meaning. Why is *validity* needed?

PROBLEM 3 (30 pts)

(a) Assume you're asked to design a reliable distributed information storage/retrieval system using straightforward replication (i.e., without use of error-correcting code to achieve higher efficiency). What is the trivial solution to achieve k-fault-tolerance in an n-node system? What implicit assumption about the connectivity property of the system is being used?

What *minimal* assumption about the degree of each node $deg(v)$ is needed to achieve k-fault-tolerance? Is this also a sufficient condition? Explain.

(b) Give a sketch of the distributed total-order multicast algorithm discussed in class. In what sense is it inferior to the centralized algorithm ignoring issues of efficiency and fault-tolerance? (*Hint: Think of the holdback queue and when messages can be dequeued/delivered.*)

Give an outline of the proof of the safety property; i.e., make clear what the structure of the proof is and its principal components. Why is liveness a direct consequence of safety?

PROBLEM 4 (30 pts)

(a) Consider the centralized mutual exclusion algorithm discussed in class. Extend the algorithm to tolerate Byzantine (i.e., malicious) faults in the clients excluding the coordinator itself. In part, this means that regular node-failures must not indefinitely stall the correct operation of the remaining system and adversarial behavior such as impersonation must be suitably handled. Argue why your algorithm is correct and indicate any assumptions (if any) you make to show correctness.

(b) Sketch the design of both a centralized and decentralized load balancing algorithm. State what objective function you're trying to optimize and indicate how in both cases this is to be achieved. Assume process migration is not allowed.