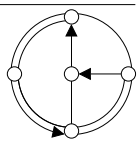
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



SS 2003 Prof. R. Wattenhofer / Regina Bischoff / Fabian Kuhn

Exam
Principles of Distributed Computing

Monday, October 6, 2003

Do not open or turn until told so by the supervisor!

Notes

There is a total of 90 points. The number of points is given before each individual question in parentheses. The total for each group of questions is indicated after the title.

Your answers can be in English or in German. Algorithms can be specified in high-level pseudo-code or as a verbal description. You do not need to give every last detail, but the main aspects need to be there. Big-O notation is acceptable when giving algorithmic complexities.

Points

Please fill in your name and student ID before the exam starts.

Name	Legi-Nr.

Question Nr.	Achieved Points	Max Points
1		15
2		22
3		24
4		14
5		5
6		10
Total		90

1 Complexity of Distributed Algorithms (15 Points)

a) (3) How is time complexity defined in the asynchronous model?

b) (3) Consider a flooding algorithm in a graph: A source transmits a message (with an ID) and each node forwards the message to all neighbors if it has not yet seen it. What is the time complexity of this algorithm in the synchronous and in the asynchronous model?

c) (3) How is the message complexity defined for synchronous algorithms? What is different for asynchronous algorithms?

d) (6) In an asynchronous setting, design an algorithm that synchronizes a given protocol. More precisely, assume you are given a synchronous algorithm S. You need to devise an algorithm A which guarantees that S will work as specified even if the network is asynchronous. What is the message complexity of your algorithm?

2 Restructuring the LSS (22 Points)

Recall the organizational structure of the LSS (the Liechtensteinian Secret Service from Exercise 3): each member of the LSS can communicate only with his direct superior and his direct subordinates over a secure phone line. On top of this tree hierarchy sits L, the "big boss." Members who do not have any subordinates are the *field agents*. All others (including L) are *office workers*. Let the total number of LSS members be $n > 1$.

a) (4) In an effort to improve the efficiency of the LSS, L suspects that there are too many office workers and not enough field agents to save the world. To that end, she needs to know exactly how many people in the company are office workers and how many are field agents. Devise an efficient asynchronous, distributed algorithm, started by L, to determine those numbers.

b) (2) What are the time and message complexities of your algorithm?

Because of political turbulences, Liechtenstein is now being split into two countries, Lichtstein and Lampenstein, who each want to have their own secret services, LiSS and LaSS, respectively. The politicians agree to create two groups of people out of the original LSS. The goal is that each new group collectively has the capacity to perform the same jobs as the LSS before. The jobs were such that, in the original LSS, every member knew how to execute his own task *and* all the tasks of his direct contacts (i.e. the direct superior's and subordinates' tasks). And since a person is only allowed single citizenship, he can only be part of either the LiSS or the LaSS. Note that we do not care about the internal structure of the future LiSS and LaSS at this point, only about membership.

c) (5) Devise an asynchronous algorithm that assigns each member of the LSS to either LiSS or LaSS. The algorithm is initiated by L and should terminate in time $O(depth(T))$, where T refers to the LSS structure.

d) (8) Same task as above. Now the algorithm may be synchronous, is started by all members simultaneously and should terminate in time $O(\log^* n)$, where n is the number of members in the original LSS. You may use the algorithms of the lecture as a black box. Show that your algorithm correctly solves the problem in the specified amount of time.

e) (3) If Liechtenstein had been split into several countries, how many such entities could have been created maximally and why?

3 Routing and Contention (24 Points)

In this problem, we consider the contention of routing on two different topologies. The contention at a node v is defined to be the number of routing messages passing through node v during the execution. The contention of a routing problem then is the contention of the worst-case node. Throughout Problem 3, use n for the number of nodes in the graph/topology. Give all results as a function of n.

a) (2) Give an oblivious routing algorithm for the 2-dimensional mesh.

b) (1) Neglecting contention, how long does it take to send a message from a source s to a destination t with your algorithm (in the worst case)?

c) (3) Construct a one-to-one (or a permutation) routing problem which has worst-case contention for your routing algorithm. Give a worst-case node. How bad is the contention?

We will now look at the same problem on hypercubes.

d) (4) Give a deterministic, oblivious routing algorithm for the hypercube which sends messages on shortest paths.

e) (1) Neglecting contention, how long does it take to send a message from a source s to a destination t with your algorithm (in the worst case)?

f) (8) Construct a one-to-one (or a permutation) routing problem which has bad (i.e. as large as possible) contention for your routing algorithm. Give a worst-case node. How bad is the contention?

g) (5) Show that the contention of your one-to-one routing problem for Question f) is (up to a constant factor) worst-case for your algorithm.

4 Byzantine Quorum Systems (14 Points)

Quorum systems have been defined in the lecture for load balancing and for tolerating faulty servers. The possible failures were limited to crashes.

Consider now a *Byzantine quorum system* (i.e., one that tolerates arbitrary (Byzantine) failures) and its application to a replicated read-write register. As with crash failures, the idea is that a client writes by sending a message to all servers in a quorum and reads by receiving a message from all servers in a quorum. No digital signatures must be used.

a) (4) Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of servers. For a given t, define a Byzantine quorum system on $\mathcal{P}$ that tolerates t failures. (Hint: We still require that some non-faulty server is in the intersection of every two quorums. But faulty servers may give wrong answers and the reader must still be able to determine the most recent value/timestamp pair from a correct server.)

b) (4) Describe algorithms for writing to and reading from a single-reader single-writer replicated read/write register, implemented by a Byzantine quorum system.

c) (6) Based on the 2-dimensional Grid quorum system, describe a Byzantine quorum system that tolerates t faulty servers. What is its load?

5 Arrow (5 Points)

Consider the tree for the Arrow shared variable protocol in Figure 1 below. The token is held by the circled node labeled r. Draw in the arrows for this initial state. Next, assume that there are six concurrent requests placed by the nodes v_1 through v_6. Assuming a synchronous execution of Arrow give the order of serviced requests.

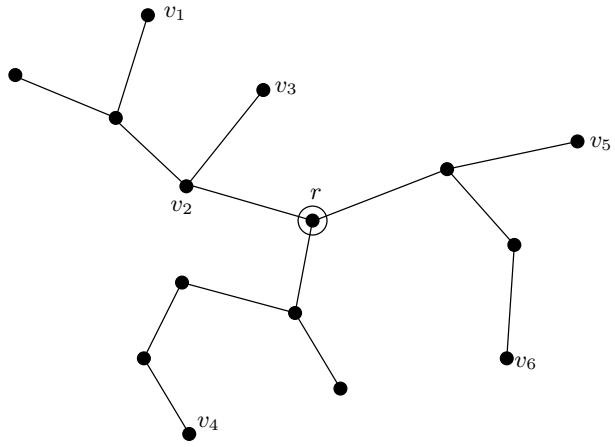


Figure 1: Tree for Question 5.

6 Independent Set (10 Points)

a) (4) Devise a synchronous uniform algorithm that computes a large independent set on a ring in $O(1)$ number of rounds. By large we mean that the computed set should be, in the expectation, a constant fraction of a maximum independent set. (Hint: Use randomization.)

b) (6) Give the value of your approximation ratio and prove it.