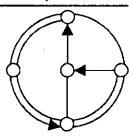
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



FS 2009

Prof. R. Wattenhofer
C. Lenzen, T. Locher

Exam
Principles of Distributed Computing

03.08.2009

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

Notes

There is a total of 120 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 may be in English or in German. Algorithms can be specified in high-level pseudocode or as a verbal description, unless otherwise mentioned. 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. However, try to provide exact bounds whenever possible.

Points

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

Name	Legi-Nr.

Question Nr.	Achieved Points	Max Points
1		24
2		32
3		33
4		31
Total		120

1 Distributed Computation Models (24 Points)

a) (3) When is it reasonable to assume that local computations take zero time?

b) (3) Give one example each for distributed systems whose communication can be modeled by means of message passing, shared memory, or all-to-all communication. Add short explanations for your choices.

c) (2) Do you consider a running time in the order of the graph diameter fast or slow in the message passing model? Explain your reasoning!

d) (2) In the message passing model, when does it *not* matter whether communication is synchronous or asynchronous?

e) (3) In the shared memory model, why does it affect the (potential) fault tolerance of the system whether communication is synchronous or asynchronous?

f) (2) In the all-to-all communication model, what prohibits the system from solving virtually everything in a single round?

g) (4) Explain the meaning of the notions of *online* and *offline* problems. Why might it be useful to analyze the offline complexity of a problem encountered in an online setting?

h) (2) What does "termination" mean for distributed algorithms?

i) (3) What is the relation between termination and self-stabilization?

2 Graph Problems (32 points)

Give distributed, synchronous, deterministic algorithms solving the following problems. You may employ any algorithm from the lecture fulfilling these criteria. Furthermore, the nodes have unique identifiers $1, \ldots, n$, n is known, and all graphs are connected.[1] No proofs are required in this exercise. (Note that you can still earn points if your solution is slightly worse than required!)

a) (3) A *dominating set* is a subset of the nodes such that any node is either in the set or has a neighbor in the set. On a ring, find in the least possible number of rounds a dominating set which is at most 3 times larger than the minimum size!

b) (5) Compute the diameter of an arbitrary graph in D time!

c) (5) Compute a 2-approximation on the diameter D of an arbitrary graph in $O(D)$ rounds, where messages contain at most $O(\log n)$ bits!

d) (3) Given a *bipartite*, i.e., 2-colorable graph, obtain a 2-coloring in at most D time using empty[2] messages!
Hint: Exploit that leader election is trivial due to the identifiers!

e) (8) Decide whether a graph is a tree in $O(n)$ rounds using at most $O(n)$ empty messages.

f) (8) The *girth* g of a graph is the size of a smallest cycle. Assume that g and all node degrees upper bounded by a constant. Provide an algorithm using messages of size $O(\log n)$ that terminates after at most $g + D$ rounds. Note that in the end all nodes must know g!

[1] Note, however, that the diameter D is unknown!
[2] That means a node either sends a message, or it does not, but does not contain any further information.

3 Job Distribution (33 Points)

Assume we are given a distributed system, modeled by a connected graph $G = (V, E)$ where $|V| = n$, in which a particular root node v_0 holds n jobs that need to be processed. Since it takes a long time to process a job, it is best to distribute the jobs among the nodes.

We assume that messages are exchanged *synchronously*, i.e., the system operates in rounds. Furthermore, we assume that at most one job/result can be sent over each edge per round. If a node receives a job in round r, it can start processing it in round $r + 1$. Accordingly, if a job is completed in round r, the result can be forwarded towards v_0 in round $r + 1$. It takes $c := 10$ rounds to process a job. We further assume that a job must be processed atomically, i.e., a node cannot partially process the job and then hand it over to another node. Once the job is completed, the result must be sent back to v_0. The *time complexity* is defined as the number of rounds it takes to distribute the jobs, process them, and accumulate the results at node v_0.

Note that a job can be completed in c rounds even if the node must also handle incoming and outgoing messages during this time. Of course, v_0 can also process some jobs itself.

a) (4) For any $n := |V|$, show that a graph exists for which the time complexity is $c + O(1)$, i.e., it takes $c + O(1)$ rounds to distribute and process the jobs and accumulate the results!

b) (6) Consider the graph consisting of 7 nodes in Figure 1. Assume that all nodes know the topology. Show how the jobs must be forwarded such that the time complexity is minimized! What is the time complexity?

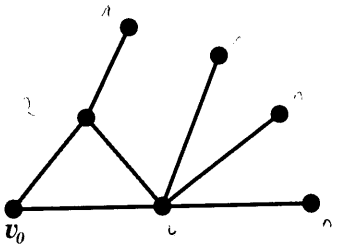


Figure 1: Example graph consisting of 7 nodes. Node v_0 has 7 jobs that need to be processed.

In order to avoid the situation that any two neighboring nodes forward jobs to each other in the same round, we first construct a *spanning tree* on which the jobs are forwarded:

- The root node v_0 sends a request to all neighbors in the first round.
- If a node v receives requests from nodes $v_1, \ldots, v_t$ in round r, it chooses any node $v_i \in \{v_1, \ldots, v_t\}$ as its parent. It then sends a message to v_i that v_i is its parent and a message indicating that v already has a parent to all other nodes $v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_t$. Moreover, it sends a request to all other neighbors.

c) (3) What is the time complexity of this algorithm (on arbitrary graphs)?

d) (3) How much larger can the diameter of the constructed spanning tree be than the diameter of the original graph G and why?

e) (3) In the given algorithm, node v_0 is not informed when the construction of the spanning tree is complete. Modify the algorithm in such a way that v_0 is informed quickly once the construction is complete! What is the (asymptotic) time complexity of your algorithm?

f) (6) Now that we have a complete spanning tree, we wonder if we might have worked too hard. Give a graph for which it takes a long time to build a spanning tree, but all n jobs can in fact be processed quickly! For arbitrary n and c, what is the worst-case ratio between the time to construct the spanning tree and the time required to process all jobs and report the results?

g) (8) For arbitrary n and c, give matching lower and upper bounds on the time complexity of the problem! Explain how you derive your bounds!

4 Minimum Dominating Sets (31 Points)

A *dominating set* is a subset of the nodes such that each node is either in the set or has a neighbor in the set. A *minimum dominating set* is a dominating set containing the least possible number of nodes. In the following, we denote by $\mathcal{N}_v$ the neighbors of a node v (not including v!). For a set of nodes $A \subseteq V$, $\mathcal{N}_A = \bigcup_{v \in A} \mathcal{N}_v$, i.e., the union of all neighborhoods of nodes in A. Have a look at the following (centralized) algorithm:

Input: a graph $G = (V, E)$
Output: a dominating set $S \subseteq V$

1: $S := \emptyset$.
2: For each $v \in V$ add a $w \in \mathcal{N}_v$ of maximum degree to S. If several neighbors have this degree, choose the one with smallest identifier.
3: $A := \{v \in V \text{ with } |S \cap \mathcal{N}_v| \geq 10\}$. $S := S \setminus \mathcal{N}_A$.
4: $S := S \cup A$.
5: $S := S \cup (V \setminus \mathcal{N}_S)$.
6: Return S.

---

a) (6) Describe verbally what the algorithm does!

b) (3) Prove that the algorithm indeed returns a dominating set!

c) (5) Explain how the algorithm can be translated into a synchronous distributed algorithm. How many rounds are at least required?

d) (4) Let M be a minimum dominating set of G. Prove that the nodes entering S in Line 2 that are not removed in Line 3 are at most $10|M|$ many.

e) (5) Show that the number of nodes chosen in Line 4 can be larger than $|M|$ by a factor of $\Omega(n)$, i.e., $|A| \in \Omega(n \cdot |M|)$!

f) (8) Assume that G is *regular*, i.e., all nodes have the same degree Δ. Prove that the worst-case approximation ratio of the algorithm is $\Theta(\Delta)$!