ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

FS 2008

Prof. R. Wattenhofer, Dr. F. Kuhn
T. Locher, Y. A. Oswald, C. Lenzen

Exam
Principles of Distributed Computing

Monday, August 18th, 2008

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. You have two hours for the exam.

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		19
2		33
3		34
4		34
Total		120

1 Distributed Computation Models (19 Points)

a) (3) Is asynchronous or synchronous communication a better model for the Internet and why?

b) (3) A researcher develops a deterministic distributed algorithm for a certain problem. Another researcher finds a randomized algorithm with the same (expected time and message) complexities for the same problem. Which result is stronger and why?

c) (4) Assume you want to prove a lower bound on the *time complexity* of a problem. Would you rather have a lower bound for randomized algorithms in an asynchronous model or a lower bound for deterministic algorithms in a synchronous model? Explain your reasoning!

d) (2) In the all-to-all model, why do we restrict the maximum message size, e.g. to $O(\log n)$ bits?

e) (3) A guy at a company called "Gogol" tells you that all their data is stored on three different machines as a precautionary measure. Assuming that a "failed" machine may exhibit arbitrary behavior, is it a good idea to have exactly three copies? Would it be enough to store only two copies or would it be better to have more than three? Explain your reasoning!

f) (4) Peer-to-peer networks often exhibit the structure of hypercubes. Why is a hypercubic structure better than using a single server (star graph) that answers all queries directly? Why don't peers simply connect to all other peers (complete graph)?

2 Coloring Trees (33 Points)

Assume that we are given an arbitrary tree consisting of n nodes. Initially, there is exactly one node with color i for each color i, $i \in \{0, \ldots, n-1\}$. The assignment of colors to nodes is arbitrary. We further assume that messages are exchanged *asynchronously.*

We study the following distributed coloring algorithm:

```
 1: Each node $v$ executes the following code:
 2: Send $c_v$ to all neighbors and receive their colors
 3: (Now $v$ knows the initial colors of all neighbors!)
 4: while there is an undecided neighbor with a larger color do
 5:     wait until message received from some neighbor $w$
 6:     Update the color of $w$
 7: end while
 8: (Now $v$ has the largest color among all undecided nodes in $v$'s neighboorhood!)
 9: Decide on the lowest color not taken by any neighbor
10: Send a message containing the final color to all neighbors
```

a) (5) Argue why this algorithm guarantees that the resulting coloring is legal!

b) (3) Prove that any node with degree k that enters the while-loop in Line 3 (i.e., that initially has a neighbor with a larger color) always has a color in the range $\{0, \ldots, k\}$ after termination!

c) (5) Since we consider asynchronous message passing, there are many possible executions on the same tree with the same initial coloring. Argue whether or not the resulting coloring depends on this asynchrony!

d) (4) Consider a list of four nodes $u - v - w - x$. Give an initial coloring, using all the colors $0, 1, 2$, and 3, such that the number of different colors in the end is minimized and another initial coloring for which it is maximized!

e) (6) Find a tree that uses 4 different colors after termination! Find one that uses 5 colors! In both cases, give the initial coloring and the coloring after termination!

f) (10) Prove an upper bound and a lower bound for this algorithm on the number of used colors in any tree consisting of n nodes!

3 Vertex Cover (34 Points)

A *vertex cover* of a graph $G = (V, E)$ is a subset $S \subseteq V$ of the nodes of G such that for every edge $\{u, v\} \in E$, $\{u, v\} \cap S \neq \emptyset$. In other words, a vertex cover is a set of nodes that "covers" all edges.[1] Given a graph G, the minimum vertex cover problem asks for a vertex cover S of G that is as small as possible.

Let Δ denote the largest degree of G. Assume that all nodes know Δ. Consider the following distributed algorithm that starts with an empty set S and adds nodes to S until S is a vertex cover. The algorithm runs in phases (one phase is one iteration of the while loop). We call an edge $e = \{u, v\}$ *covered* as soon as u or v is in S. Otherwise, e is *uncovered*. For a node v, $d(v)$ denotes the number of uncovered edges adjacent to v.

```
$S := \emptyset$; $\Delta' := \Delta$;
while uncovered edges do
    add nodes $v$ with $d(v) \geq \Delta'/2$ to $S$
    $\Delta' := \Delta'/2$
end while
set $S$ is vertex cover
```

a) (5) Argue why the above algorithm computes a valid vertex cover!

b) (5) The algorithm is given in an implicit form. In particular, it is not specified what messages the nodes need to exchange. Give more explicit pseudo-code of the algorithm describing the actions of every node! Your pseudo-code should contain all necessary messages and local computations.

c) (3) What is the time complexity of a single phase of the algorithm? What is the time complexity of the algorithm?

d) (10) Argue why the number of nodes added to S in a single phase is at most 4 times the size of an optimal (i.e., smallest possible) vertex cover!

e) (3) Give an upper bound on the approximation ratio of the above algorithm!

Consider the following simple non-distributed algorithm to compute a vertex cover: Start with an empty set $S = \emptyset$. As long as not all edges are covered, pick some uncovered edge $\{u, v\}$ and add u and v to the vertex cover S. This simple algorithm computes a 2-approximation for the minimum vertex cover problem.

f) (8) Give a time-efficient (randomized) distributed algorithm for this 2-approximation algorithm! What is the time complexity of your distributed algorithm?

[1] Note that a vertex cover is similar to a dominating set, which covers all *nodes*.

4 Byzantine Behavior (34 Points)

Assume there are n nodes and $m \leq n$ of these nodes are Byzantine nodes. All nodes know exactly which nodes are Byzantine. You can assume that all n nodes have unique identifiers.

Your goal is now to find a non-Byzantine node among these n nodes! For this purpose, you can ask them questions. You can ask one question in each round of communication and one response also costs one round. A question or a response may consist of arbitrary constant terms and contain at most one identifier. A possible question/answer is, e.g., "Is X lying?"/"X is Byzantine".

Consider the following algorithm for this problem:

```
Step 1: Ask any node $v$ to give another node $w$ that is non-Byzantine.
Step 2: Ask $w$ to give a node $u$ that is non-Byzantine.
Step 3: If $u \neq v$ then set $w := u$ and go to Step 2 else choose node $v$.
```

a) (4) Show that this algorithm might fail for any number $m \geq 2$ of Byzantine nodes!

b) (6) Give an exact bound $\mathcal{B}(n)$ as a function of n, on how many nodes can be Byzantine, i.e., show that $m > \mathcal{B}(n)$ implies that it is not possible to find a non-Byzantine node. Moreover, give a simple algorithm that shows that the problem can be solved if $m \leq \mathcal{B}(n)$! Using your simple algorithm, what is (asymptotically) the maximum number of required rounds?

In the following, we assume that there are only $m \leq \mathcal{B}(n)$ Byzantine nodes, i.e., the problem can be solved.

c) (5) Show how a non-Byzantine node can be found in expected $O(n)$ rounds using randomization!

d) (8) Give a deterministic algorithm that finds a non-Byzantine node in $O(n)$ rounds!
Hint: Try to find an algorithm for a simplified model where only the size of the response is bounded but the questions may be arbitrarily large (in particular, they may contain more than one identifier!). You will already get points for such an algorithm. Then, try to find an algorithm that works without this simplification.

e) (3) Give some arguments why it is not possible to find a non-Byzantine node in less than $O(n)$ rounds![2]

Since it is not possible to find a non-Byzantine node in less than $O(n)$ rounds, we might settle for finding one with high probability, i.e. with probability $1 - 1/n$. Assume that $m < \frac{1}{3}n$.

f) (8) Devise an algorithm that finds a good node with high probability in less than $O(n)$ rounds! How many rounds does it take (asymptotically) using your algorithm with high probability?
Hint: Use that fact that the majority of nodes in a sufficently large random subset of all nodes are non-Byzantine! For this purpose, you can use the following Chernoff bound: Let $X_1, \ldots, X_n$ be independent Bernoulli variables with $P[X_i = 1] = p_i$. Let $X = \sum_{i=1}^{n} X_i$ denote the sum of the X_i, then for any $\delta \in (0, 1]$: $P[X < (1-\delta)E[X]] < e^{-E[X]\delta^2/2}$.

[2] The slightly informal "less than $O(n)$" means that the number of rounds increases sublinearly with n.