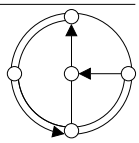
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



FS 2010 Prof. R. Wattenhofer / C. Lenzen

Principles of Distributed Computing
Exercise 8: Sample Solution

1 Counting with Asynchronous Wake-up

a) Every node that wakes up starts the algorithm as if it was the only node that woke up by itself, but includes its identifier and the current round number into all messages. Every node that is woken up by a message will append the received identifier and the (current) round number of the respective execution to its messages. If a node that is already awake receives a message with (*i*) a larger round number or (*ii*) the same round number and a smaller identifier, it "forgets" about everything that happend before and acts as if woken up by this message. If the round number is smaller or the round number equals its own, but the identifier is larger, it simply drops the message. If both are the same, it clearly belongs to the execution initiated by the same node causing this node to wake up and is processed normally.

In the end, only the execution started by the node with smallest identifier among the nodes that woke up first will "survive" and yield the same output as if executed in an environment where no other nodes wake up by themselves.[1]

b) The algorithm consists of all (awake) nodes broadcasting all identifiers they know for $2k$ rounds. Then u (and only u) decides that $k \geq n$ if it knows at most k identifiers and $k < n$ otherwise.

If $k \geq n$, the output is apparently correct, hence consider $k < n$. We claim that in round $2s$, $s \in \{0, \ldots, k\}$, any subset of $r + 1 - s$, $r \in \{s, \ldots, n - 1 + s\}$ awake nodes will know in total $\min\{r + 1, n\}$ identifiers. This is obviously true for $s = 0$ and all r as well as $r = n - 1 + s$, as any node knows its own identifier right from the start. Now assume the claim holds for some $s \in \{0, \ldots, k - 1\}$ and all $r \in \{s, \ldots, n - 1 + s\}$.

For $s + 1$ and $r = s + 1 \leq k < n$ we need to consider only a single node v. If this node wakes up in round $2s + 1$ or $2s + 2$, it will (by induction hypothesis) receive $s + 1 = r$ identifiers which must be different from its own (as it did not send any messages before). Thus it knows $r + 1$ values in round $2s + 2$. If it was awake already beforehand, the 2-interval connectivity implies that there is at least one link from v to some node w that exists both in round $2s + 1$ and $2s + 2$. If w is still asleep at the end of round $2s$, it will wake up in round $2s + 1$ and broadcast its identifier in round $2s+2$. Finally, if it is awake, we use the induction hypothesis for s and $r = s + 1 \leq k \leq n - 1$, implying that v and w together know at least $r + 1$ values, which they will exchange in round $2s + 1$. Thus, the claim holds for $s + 1$ and $r = s + 1$.

Now assume that the statement is also established for $s + 1$ and $r \in \{s + 1, \ldots, n - 2 + s\}$. Choose any set of nodes $S = \{v_1, \ldots, v_{r+1-s}\}$ that is awake in round $2s + 2$. If one of the nodes has been asleep at the end of round $2s$, it certainly contributes a new identifier and thus the claim follows from the fact that it is already known for $s+1$ and r, as the remaining nodes together have in total $r + 1$ different identifiers. If all the nodes are awake in round $2s$, there must be at least one edge from an (awake) node in S to some node w that exists

[1] Note that any protocol needs to obtain information from all nodes, thus nodes cannot terminate too early and "miss" the execution initiated by this node.

both in round $2s+1$ and round $2s+2$. Either w is asleep in round $2s$, implying that it wakes up and a node in S will learn its identifier which previously was unknown to all nodes in S, or all nodes from $S \cup \{w\}$ were awake in round $2s$. In the latter case, we can apply the hypothesis for s and $r+1 \leq k$ to see that in round $2s$, $S \cup \{w\}$ knew $r+2$ identifiers. In round $2s+1$, a node in S will receive all identifiers known to w, hence the same is true for the set S in round $2s+2$.

In summary, we can start from $s = r = 0$, go from s and all $r \in \{0, \dots, k\}$ to $s+1$ and $r = s+1$, and from there to all $r \in \{s+1, \dots, n-1+s\}$, i.e., the statement is proved by induction for all possible values of s and r. In particular, setting $s = r = k$, node u (that is, the set containing only the element u) will know $\min\{k+1, n\}$ elements. If $k < n$, this equals $k+1$ and thus u will correctly decide that $k < n$.

c) Starting from $k = 1$, u runs the algorithm from part **b)** repeatedly, in each step doubling k. Note that in the final run, u will learn about all $n \leq k$ identifiers, thus it can determine n exactly. Moreover, actually all nodes are awake and therefore aware of the fact that $k \geq n$ and they know all identifiers. Hence the algorithm may terminate and output n at all nodes without need for further intervention of u.

We conclude that the time complexity of the algorithm is the sum of the number of rounds the individual runs of the algorithm from part **b)** take, i.e.,

$$\sum_{i=0}^{\lceil \log n \rceil} 2 \cdot 2^i = 2\left(2^{\lceil \log n \rceil + 1} - 1\right) < 8n.$$

2 Token Dissemination

The schedule of u solely depends on the round number. Because there are n identifiers and only constantly many may be packed into one message, there must be one token t which is scheduled only $n-2$ times in the next $\Omega(n^2)$ rounds. The adversary separates the graph into two parts, one which already knows t, and one which does not know t. Initially, the one part consists of v only. Connectivity is maintained by a single link between the two parts whose one endpoint is v. It does not matter how the parts are connected internally.

Now, in any round where v does not send t, the number of nodes which do not know t remains the same. In the rounds in which t is sent, this number decreases by one. Thus, from the initially $n-1$ nodes not knowing t, after $\Omega(n^2)$ rounds at least one node remains, i.e., the algorithm must run for $\Omega(n^2)$ rounds in order to disseminate all information to all nodes.