Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. F. Kuhn
Mohamad Ahmadi, Hamid Ghodselahi

May 26, 2016

Network Algorithms, Summer Term 2016
Problem Set 3 – Sample Solution

Exercise 1: Leader Election (Message Complexity Improvement)

(a) Consider the following flooding/echo algorithm. Each node v keeps a variable $maxid_v$ which is the largest ID seen so far by v and a pointer p_v which determines its parent in the spanning tree being constructed during the execution. Initially for each node v, $maxid_v$ is its identifier and p_v is null. In the first round each node sends its own ID to all its neighbors. Thereafter, each node v, upon receiving an ID strictly larger than $maxid_v$, updates its $maxid_v$ to the new larger ID and p_v to one of the senders of this ID. After updating $maxid_v$ it sends the $maxid_v$ to all its neighbors except its parent. When a node receives either X or ACK_X from each of its neighbors, then it sends ACK_X to its parent. When a node with ID equals X receives ACK_X from all its neighbors, then the node realizes that it has the largest ID and it broadcasts through the constructed tree to inform all the nodes (i.e., upon receiving this information, termination is detected by all the nodes).

Clearly it takes D rounds until the largest ID in the network reaches all the nodes in the network and D rounds for doing the echo back to the node with the largest ID and D rounds more for informing all the nodes about the leader's ID. After $3D$ rounds all the nodes receive an ID larger than their own IDs. Hence all the nodes except the node with maximum ID decide to non-leader. Therefore, the whole running time of the algorithm is $O(D)$. Each node updates its $maxid$ variable at most n times and hence sends a message on each of its adjacent edges at most n times. Thus the message complexity of the algorithm is at most $O(mn)$.

(b) Each node v, in the first round receiving a message (or messages), stores it (or one of them) in the output buffer. In the next round it transmits its buffer content to all its neighbors that it has not received any message from so far.

Since any node in the network is at the distance of at most D of some source message, it will be informed about at least one message in at most D rounds of execution.

(c) The idea of solving this exercise is from the FindMax algorithm introduced by Chrobak[1]. The algorithm runs for $\log(N)$ phases. Each phase consists of D rounds. In each phase all nodes know that the maximum ID is in $[a, b]$, where a and b are positive integers and $a \leq b$. Initially $a = 1$ and $b = N$.

In each phase let $c = \lceil (a + b)/2 \rceil$. Assume that each node v (if any!), where $c \leq ID_v \leq b$, is a source node with source message $[c, b]$. Then by running the flooding algorithm of part (b), all the nodes receive $[c, b]$ in D rounds. Therefore, all the nodes will learn in this phase whether there exists a node with ID in $[c, b]$ or not. If there exists at least one such a node, then all the nodes update a to c. Otherwise, they update b to c. Therefore, in each phase the interval $[a, b]$ halves and after $\log(N)$ phases the interval closes and the maximum ID is known to all.

Each phase takes D rounds and over each edge at most 2 messages are sent. Hence, the total message complexity is $m \log(N)$. The time complexity of each phase is D, and hence the whole time complexity is $D \log(N)$.

[1] M. Chrobak, L. Gasieniec, and W. Rytter. Fast broadcasting and gossiping in radio networks. Journal of Algorithms 2002.

Exercise 2: Distributed Computation of the AND

(a) For the sake of contradiction let us assume that there exits an algorithm A which solves the problem. Examine what happens on a ring of $n \geq 3$ nodes, where all inputs are 1. In any round, the states of all nodes are identical (by induction, as in the proof of Lemma 3.4). Observe that these states do not depend on the size n of the ring, as the algorithm is uniform and the local topology is independent of n. Thus, there must be some constant round number t that does not depend on n such that all nodes terminate and output 1 in round t (as we assumed the algorithm to be correct).

Now run A on a ring of size $2(t+1)$, where exactly one node has 0 as input bit. Up to distance t from the node on the opposite side of the ring, all nodes have input 1. Again, analogously to Lemma 3.4, until round t, this node will have the same state as if in a ring where all nodes have input 1. Hence, in round t it will terminate and output 1, contradicting the assumption that A is correct and should output 0 at all nodes.

(b) All input values have to be sent all around the ring. Each node, to detect the return of its own message, adds a hop counter to its message. If the message has made n hops, it has arrived where it started.

(c) The following algorithm calculates the AND in a synchronous, non-uniform ring:

Algorithm 1 AND in the Ring: synchronous, non-uniform (n is the number of nodes)

```
1: if input bit = 0 then
2:    send 0 to the neighbor in the ring
3: end if;
4: for i := 2 to n do
5:    if received a 0 and have not already sent a 0 then
6:       forward the 0 to the other neighbor in the ring
7:    end if
8: end for;
9: if received at least one 0 then
10:   result := 0
11: else
12:   result := 1
13: end if;
```

If the result is 1, no message is sent, otherwise there is exactly one message over each link. Thus, the time and message complexity are both n.