Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. F. Kuhn

June 5, 2014

Network Algorithms, Summer Term 2014
Problem Set 2 – Sample Solution

Exercise 1: Almost Anonymous Leader Election

1. It is generally not possible to elect a leader if $n \equiv 0 \pmod 3$. The proof is analogous to the impossibility proof for anonymous deterministic leader election in rings. Assume that the identifiers are distributed perfectly symmetrically, i.e., between any two nodes with identifier 1, there are $(n-3)/3$ nodes with identifier 0. Every node has at least 2 perfectly symmetric nodes in the ring and therefore, for every node, there is always at least 2 other nodes that have the same state at all times. Formally, one can show by induction on the number of rounds that for every $r \geq 0$ and every $0 \leq d \leq \lceil (n-3)/6 \rceil$, all nodes at distance d from a node with ID 1 have the same state.

   Now, assume that $n \not\equiv 0 \pmod 3$. All 3 nodes with ID 1 start by sending a message around the ring. Whenever a node receives a message on one of its edges, it forwards it to the other edge. By counting how many ones a message has seen and how many zeroes there were in-between the ones, the ID 1 nodes can detect when they get their messages back. Further, they can also learn the structure of the ID assignment on the ring (how many 0-nodes there are on both sides and the number of 0-nodes between the other two ID 1 nodes). Based on that information, a leader can be uniquely determined, e.g., it can be the 1-node between the larger two stretches of 0-nodes, and if that is no unique, it is the 1-node between the smaller two stretches of 0-nodes.

2. Consider any ring with n nodes and some ID assignment. No node can distinguish this ring from a ring with kn nodes in which the same ID assignment is repeated k times (for every positive integer k). Further in the ring with kn nodes for $k > 1$, every node has at least $k-1$ perfectly symmetric nodes.

Exercise 2: Leader Election in Trees

We only sketch a solution. Each node that has heard from all but one neighbors, forwards a message to the remaining neighbor. That is, at the beginning all the leaves send a message to their single neighbor and then, messages propagate towards the "center" of the tree. While sending these messages "up" the tree, the nodes can count the number of visited nodes (i.e., the sizes of the sub-tree). This computation stops as soon as two nodes receive such a message from each other or if a node u receives messages from all neighbors before it can propagate any message (i.e., if the messages from the last two neighbors arrive at the same time). In the second case, u becomes the leader. In the first case, we can elect one of the two nodes as leader, if they are roots of subtrees of different sizes. This is guaranteed if n is odd. If n is even, the tree could e.g., be a path of length n such that in a synchronous execution, no leader can be elected.

Exercise 3: License to Match

1. We use a variant of the Echo algorithm (Algorithm 12). A node (i.e. an agent in the hierarchy) matches up all (except for at most one) of its children. If one participating child remains and

the node itself also participates, it matches itself with that child. If either the node or one of its children remain, then the node sends a request to "match" upwards in the hierarchy. Otherwise, it sends a "no match" and that subtree is done. We give an asynchronous, uniform matching algorithm below.

Algorithm 1 Edge-Disjoint Matching

```
1: wait until received message from all children
2: while at least 2 requests remain (including myself) do
3:    match any two requests
4: end while
5: if exists leftover request then
6:    send "match" to parent (= superior)
7: else
8:    send "no match" to parent
9: end if
```

When a node v sends a "match" request to its parent u, then the edge $\{u, v\}$ will be used only once since there will be only one request in the subtree rooted at v. Along with the messages of the algorithm, the required path information is sent; we left this out in the pseudocode to improve readability.

2. Let T be the tree with n nodes. Assuming each message takes at most 1 time unit, then the time complexity of Algorithm 1 is in $O(\text{depth}(T))$ since all the requests travel to the root (and back down if we inform the agents of their assigned partners). On each link, there are at most 2 messages: 1 that informs the parent whether a match is needed and optionally 1 more to be informed by the parent of the match partner. So there are a total of at most $2(n-1)$ messages.