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

May 30, 2013

Network Algorithms, Summer Term 2013
Problem Set 4 – Sample Solution

Exercise 1: Concurrent Ivy

1. The three nodes are served in the order v_2, v_3, v_1.
2. Figure 1 depicts the structure of the tree after the requests have been served. Since v_1 is served last, it is the holder of the token at the end.

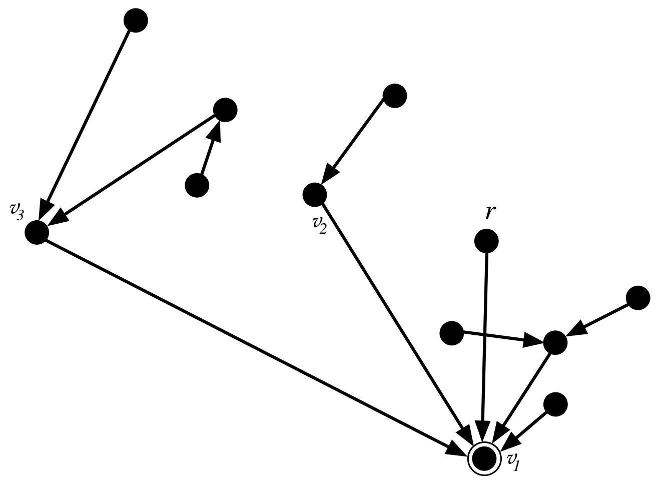

Figure 1: Tree after the requests have been served.

Exercise 2: Tight Ivy

In order to show that the bound of $\log n$ steps on average is tight, we construct a special tree, called *Binomial Tree*, which is defined recursively as follows. The tree $\mathcal{T}_0$ consists of a single node. The tree $\mathcal{T}_i$ consists of a root together with i subtrees, which are $\mathcal{T}_0, \ldots, \mathcal{T}_{i-1}$, rooted at the i children of the root, see Figure 2.
First, we will show that the number of nodes in the tree $\mathcal{T}_i$ is 2^i. This obviously holds for $\mathcal{T}_0$. The induction hypothesis is that it holds for all $\mathcal{T}_0, \ldots, \mathcal{T}_{i-1}$. It follows that the number of nodes of $\mathcal{T}_i$ is $n = 1 + \sum_{j=0}^{i-1} 2^j = 2^i$.
We will show now that the radius of the root of $\mathcal{T}_i$ is $\mathcal{R}(\mathcal{T}_i) = i$. Again, this is trivially true for $\mathcal{T}_0$. It is easy to see that $\mathcal{R}(\mathcal{T}_i) = 1 + \mathcal{R}(\mathcal{T}_{i-1})$, because $\mathcal{T}_{i-1}$ is the child with the largest radius. Inductively, it follows that $\mathcal{R}(\mathcal{T}_i) = i$.
By definition, when cutting of the subtree $\mathcal{T}_{i-1}$ from $\mathcal{T}_i$, the resulting tree is again $\mathcal{T}_{i-1}$. Let $\mathcal{C} : \mathcal{T}_i \mapsto \mathcal{T}_{i-1}$ denote this cutting operation. For all $i > 0$, we thus have that $\mathcal{C}(\mathcal{T}_i) = \mathcal{T}_{i-1}$. We will now start a request at the single node v with a distance of i from the root in $\mathcal{T}_i$. On its path to the root, the request passes nodes that are roots of the trees $\mathcal{T}_1, \ldots, \mathcal{T}_i$. All of those nodes become children of the

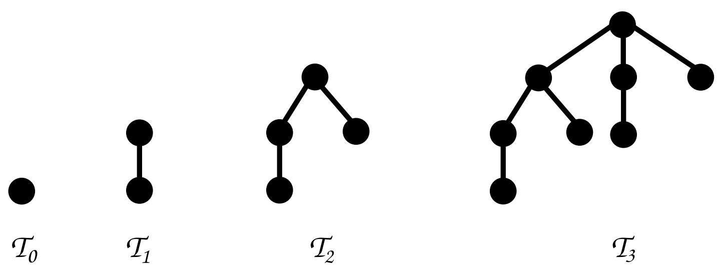

Figure 2: The trees $\mathcal{T}_0, \ldots, \mathcal{T}_3$.

new root v according to the Ivy protocol. The new children lose their largest "child" subtree in the process, thus the children of node v have the structures $\mathcal{C}(\mathcal{T}_1), \ldots, \mathcal{C}(\mathcal{T}_i) = \mathcal{T}_0, \ldots, \mathcal{T}_{i-1}$. Hence, the structure of the tree does not change due to the request and all subsequent requests can also cost i steps. Since $n = 2^i$, each request costs exactly $\log n$.

Exercise 3 from Exercise Sheet 2: 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.