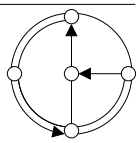
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



FS 2009 Prof. R. Wattenhofer / C. Lenzen / T. Locher

Principles of Distributed Computing
Solution 8

1 Distributed Network Partitioning

Throughout this exercise, we use the following definition: For any node v let $\mathcal{N}_i(v) := \{w \in V \mid d(v,w) \leq i\}$, i.e., $\mathcal{N}_i(v)$ denotes the set of nodes within distance i to v.

a) The problem is that without root, we have no parents or children. We use an echo/flooding algorithm on the tree to determine the node with the lowest ID and appoint it to be the leader. We assume w.l.o.g. that no two messages are received by a node at the same time. Each leaf sends its ID to its neighbor. All other nodes wait until they have received the IDs from all but one neighbor. Then they choose the smallest amongst these and their own IDs and send it to the remaining neighbor. If a node receives a message after sending its own, it compares the received ID to the one it sent. The smaller of the two is the ID of the leader, which is then flooded to the remaining $n-2$ nodes.

Correctness: Observe that each node determines the smallest ID in its subtree when the egde to the neighbor that has not sent a message yet is deleted. The last two nodes will send each other a message, after the subtrees corresponding to the two neighboring nodes have been processed (recall that messages do not arrive at identical times). Subsequently, they will broadcast the minimum of the lowest IDs of the two subtrees as the ID of the leader. Thus all nodes will be informed about a unique ID specifying the leader.

Complexity: The running time is $O(D) \leq O(n)$, where D is the diameter. The message complexity is $O(n)$ since a tree hat $n-1$ edges.[1]

b) The leader starts modified flooding/echo rounds with increasing diameter until the cluster criterion is met. Simultaneously, a BFS tree around the leader is constructed that will be needed to reduce the number of messages sent. Initially, the tree consists of the cluster leader l only, which is considered the root.

In round i the cluster leader initiates a flooding/echo on the BFS tree. A node that joined the BFS tree in round $i-1$ (a $(i-1)$-hop neighbor of l, or l itself if $i=1$) will notify all its direct neighbors within the whole graph. Once a node v not already in the BFS tree receives such a notification from a node p, it joins the BFS tree with p as parent and informs p about this. If a node already in the BFS tree receives such a notification, it informs the node sending it that it will not join the BFS tree. All nodes in the tree accumulate the number of nodes in their subtree and return them to their parent. Thus l will be informed about the number $|\mathcal{N}_i(l)|$ of nodes within range i when round i finishes after at most $2i$ time units.

This procedure is started with $i=1$ and stops when $2|\mathcal{N}_{i-1}(l)| \geq |\mathcal{N}_i(l)|$. Then l informs the nodes in $\mathcal{N}_r(l)$ that they participate in its cluster by a broadcast in the constructed BFS tree. The correctness of this procedure is obvious from the construction. The time complexity is $\sum_{i=1}^{r+1} 2i = (r+1)(r+2)$ for the flooding/echo steps, and additional $r+1$ time units for the final broadcast. In total we get a time complexity of $O(r^2)$. Due to the stop criterion we have $|C| \geq 2^r$. Thus we have $r \leq \log|C|$, implying $O(r^2) \leq O(\log^2|C|) \leq O(|C|)$.

[1] For more details, we refer to Algorithms 11 and 12 from the script.

c) The i^{th} step of the construction from b) will require two messages to be sent over all egdes from nodes that joined the BFS tree in the $(i-1)^{th}$ step, plus two messages per edge in the already constructed BFS tree. For the former, observe that during the (stepwise) construction of the BFS tree we send messages over each edge in E' at most twice, as E' is the set of edges with at least one endpoint in the final cluster. For the latter, note that the number of edges in the tree after the i^{th} step is $|\mathcal{N}_i(l)| - 1$. We have the condition $|\mathcal{N}_i(l)| > 2|\mathcal{N}_{i-1}(l)|$ for $i \in \{2, \dots, r\}$, hence we can estimate $|\mathcal{N}_i(l)| \leq 2^{-r+i}|\mathcal{N}_r(l)|$ for $i \in \{1, \dots, r\}$. Summing over i, we may estimate the total number of messages by

$$\sum_{i=1}^{r} 2(|\mathcal{N}_i(l)| - 1) < 2|\mathcal{N}_r(l)| \sum_{i=1}^{r} 2^{-r+i} = 4|\mathcal{N}_r(l)| \sum_{i=0}^{r-1} 2^{-i} < 8|\mathcal{N}_r(l)| < 8|E'|.$$

Thus, in total at most $(8+2)|E'| \leq O(|E'|)$ messages are sent during the construction phase of the cluster in the worst case.

d) Consider the tree to be rooted at the first leader. We apply a distributed depth-first search strategy on the tree. After each cluster construction step, the current leader continues the depth-first search for the next leader, which is the next node on the search path that is not assigned to a cluster yet. In the course of the algorithm execution, each edge of the tree is hence traversed twice, i.e., the time and message complexity is $O(n)$.

e) Let $C_i,\ i \in I$, denote the vertex sets of the constructed clusters and E'_i the edges "removed" after the cluster construction of the i^{th} cluster. By c) the total number of messages due to the cluster construction is bounded by $\sum_i O(|E'_i|) = O(m)$, since the sets E'_i are disjoint. By part b) of the exercise we have a worst-case estimate for the (total) time complexity of cluster construction of $\sum_i O(|C_i|) = O(n)$, using that the vertex sets of the clusters are disjoint. Finally we have to add $O(n)$ messages and $O(n)$ time due to leader election, which was shown in d). Putting everything together, we conclude that the time complexity is $O(n)$ and the message complexity is $O(n+m) = O(m)$.