Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn

Distributed Systems, Summer Term 2020
Exercise Sheet 11

Exercise 1: Distributed Network Partitioning

In this exercise, we will derive an asynchronous distributed version of the cluster construction algorithm presented in the lecture.
Assume that (in $O(n)$ time and using $O(m + n \log n)$ messages) a spanning tree has already been computed. You can further assume that the constant ρ in the algorithm is 2. Moreover, you can ignore the intercluster edges in this exercise, i.e., the number of intercluster edges does not have to be reduced once the clusters are built.
Just like the centralized algorithm, the distributed algorithm repeatedly applies the following two steps to construct a partitioning:

1. Find a (cluster) leader.
2. Construct a cluster C, remove all the nodes $v \in C$ and remove all the edges $\{u, v\}$ for which $u \in C$ or $v \in C$ (or both).

Let us now construct the clusters:

a) We need a first leader to start the algorithm. Describe how a leader can be determined in $O(n)$ time on the spanning tree using $O(n)$ messages!

b) Given a leader, we need to build the cluster by adding more and more nodes to the cluster. Describe how the leader constructs the cluster! Let C denote the constructed cluster. Show that the time complexity to construct the cluster is bounded by $O(|C|)$!

Hint: If the radius of C is r, show that the time complexity is $O(r^2) \subseteq O(\log^2 |C|) \subset O(|C|)$!

c) Let $E' \subset E$ be the set of edges that can be removed in Step 2. Show that the construction of cluster C requires $O(|E'|)$ messages!

Hint: Use the observation that edges closer to the leader have to be traversed more often, but there are more edges with a greater distance to the leader!

d) Once a cluster has been constructed, we need to find the next leader in the remaining graph. Show how each node in the graph can be visited in $O(n)$ time using $O(n)$ messages! How can this tree-traversal scheme be used to find the cluster leaders every time a new cluster has been built?

e) Putting everything together, show that the entire partitioning requires $O(n)$ time and uses $O(m)$ messages![1]

[1] The construction phase of the spanning tree is (clearly) not considered in these bounds.

Solutions

We use the following definition: For any node v let $N_i(v) := \{w \in V \mid d(v,w) \leq i\}$, i.e., $N_i(v)$ 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 ID amongst these and their own 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 an 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 has $n-1$ edges.

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 $i-1$ (an $(i-1)$-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 $|N_i(l)|$ 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 $2|N_{i-1}(l)| \geq |N_i(l)|$. Then l informs the nodes in $N_r(l)$ 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 $O(r^2) \subseteq O(\log^2|C|) \subset O(|C|)$

c) The i^{th} step of the construction from b) will require two messages to be sent over all edges 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 $|N_i(l)|-1$. We have the condition $|N_i(l)| > 2|N_{i-1}(l)|$ for $i \in \{2,\dots,r\}$, hence we can estimate $|N_i(l)| \leq 2^{-r+i}|N_r(l)|$ for $i \in \{1,\dots,r\}$. Summing over i, we may estimate the total number of messages by

$$\sum_{i=1}^{r} 2(|N_i(l)|-1) < 2|N_r(l)| \sum_{i=1}^{r} 2^{-r+i} = 2|N_r(l)| \sum_{i=0}^{r-1} 2^{-i} < 4|N_r(l)| < 4|E'|.$$

Thus, in total at most $(4+2)|E'| \leq O(|E'|)$ 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 complexityis $O(n + m) = O(m)$.