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

Distributed Systems, Summer Term 2019
Exercise Sheet 2

In the following exercises we consider the CONGEST model. This is a **synchronous** message passing model with the additional property that the **size** of each message is bounded. If we assume that the nodes have IDs in $\{1, \dots, n\}$ and communicate by exchanging bitstrings, then each message is only allowed to contain $O(\log n)$ bits. This means that each message may contain for example (the binary representation of) a constant number of integers $\leq n^c$ for some constant c. However, it is not possible that a node sends another node the IDs of all its neighbors in a single message, as the degree of the network may not be bounded.

Remark: Do not confuse the message size and the message complexity.

1. k-Selection Problem in Graphs

Given a graph G with n nodes that have pairwise distinct input values $\leq n^c$ for some constant c, the k-selection problem for a $k \leq n$ is the problem of finding the k^{th}-smallest value in the graph.
Our goal is to describe a randomized distributed algorithm in the CONGEST model that solves the k-selection problem with an expected runtime of $O(D \cdot \log n)$.

a) Assume a tree T of depth D. Describe an algorithm that computes in $O(D)$ rounds for every node v a value s_v which equals the size (number of nodes) of the subtree with root v.

b) Assume a tree T of depth D and root r in which each node is able to flip coins. Describe a method to choose a node from the tree uniformly at random (i.e., each node has the same probability to be chosen) in time $O(D)$.

   Hint: Use the algorithm from a).

c) Describe a randomized algorithm that solves the k-selection problem with an expected runtime of $O(D \cdot \log n)$.

   Hint: Use the algorithm from b).

Sample Solution

a) Each node v learns the size of the subtree with root v in $O(D)$ rounds via the following convergecast algorithm:

```
Each node $v$ executes in parallel the following code:
if $v$.children == $\emptyset$ then
    $s_v = 1$
    send $s_v$ to parents
else if $v$.children == $\{u_1, \dots, u_{\deg(v)}\}$ then
    Wait until values $s_{u_1}, \dots, s_{u_{\deg(v)}}$ are received.
    $s_v = \sum_{i=1}^{\deg(v)} s_{u_i} + 1$
    send $s_v$ to parents ($s_v \leq n$)
```

b) We first compute in $O(D)$ rounds s_v for every node v. We introduce a new attribute v.marked for every node which may be true or false. At the beginning the root r is marked and all other nodes are unmarked.

```
Each node $v$ executes in parallel the following code:
**if** $v$ receives a notification **then**
    $v$ marks itself
**if** $v$ is marked **then**
    $v$ chooses itself with probability $1/s_v$.
    **if** $v$ is chosen **then**
        $v$ sends its ID to the root.
    **else**
        $v$ picks one of its children $u_1, \dots, u_{\deg(v)}$, each with probability $\frac{s_{u_i}}{s_v - 1}$, and sends a
        notification to it.
    $v$ unmarks itself.
```

c) At the beginning, all nodes are marked. The algorithm operates in phases. With the algorithm from b), the value x of some random node is chosen. With flooding/echo every node learns this value. With a similar algorithm as in a), the root learns how many nodes have a smaller and how many have a larger value, i.e., it computes the rank of x (its position in the sorted list of all values). If rank$(x) = k$, then x is the k^{th}-smallest value. If rank$(x) < k$, this information is broadcasted and each node with a value $\leq x$ unmarks itself. If rank$(x) > k$, each node with a value $\geq x$ unmarks itself. Next, the value y of some leftover marked node is chosen randomly and the process is repeated.

Runtime Analysis: Let u be the random node that is chosen in some phase. At the end of the phase, either all nodes with smaller value or all nodes with larger value get unmarked (or the algorithm terminates). We call a phase *good* if at least one third of the unmarked nodes have a smaller value than u and at least $1/3$ have a larger value than u, which means that at least $1/3$ of the remaining marked nodes become unmarked. After $\log_{3/2} n$ good phases, no node is unmarked and the algorithm terminates. The probability that a round is good is $1/3$. It follows that after $3 \log_{3/2} n$ phases, the expected number of good phases is at least $\log_{3/2} n$. So the algorithm needs in expectation at most $3 \log_{3/2} n$ phases, each taking $O(D)$ rounds.

2. Leader Election

Given a graph G, describe a deterministic algorithm in the CONGEST model such that every node learns the smallest ID in the graph and terminates after $O(D)$ rounds. Analyse the message complexity of the algorithm.

Sample Solution

Every node u stores the smallest ID it has seen in variable x_u.

1. Initially, u sets $x_u = \text{ID}_u$ and u.parent=None and sends x_u to its neighbors.

2. If $x_v < x_u$ for the smallest value x_v that u receives from a neighbor (there might be several neighbors with this value), set $x_u = x_v$, u.parent= v and send x_u to all neighbors.

This way, only the node with the smallest ID successfully builds a spanning tree. After receiving an echo it appoints itself as leader and broadcasts this decision. The message complexity is $O(m \cdot n)$.