University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
S. Faour, A. Mályusz

Theory of Distributed Systems
Sample Solution Exercise Sheet 11

Due: Wednesday, 17th of July 2024, 12:00 noon

Exercise 1: Aggregation in the MPC Model *(15 Points)*

Assume you are given a number of $M \in O\big(\frac{N}{S} \log_S N\big)$ machines, where N is the number of *aggregation messages* that are collectively stored by the machines M_i, $i \in \{1, \ldots, M\}$. Each machine M_i has a memory large enough to store S such messages. By definition of the MPC model every machine can send and receive at most S aggregation messages per round.

Each aggregation message m has an aggregation value v_m, a target machine t_m and an aggregation group g_m. All messages in the same group go to the same target and each machine is the target of not more than one aggregation group. The *aggregation problem* is solved when every target machine t_m learns an aggregation message m that has *minimal* value among all aggregation messages of its aggregation group g_m. Formulate an algorithm that solves said aggregation problem in $O(\log_S N)$ rounds such that no machine sends or receives more than $S/2$ messages per round *in expectation.*

Simplifications: You may assume that the intial aggregation messages are stored on $\lceil \frac{N}{S} \rceil$ machines and none of those machines is a target of an aggregation message. This means that machines can be partitioned into $O(\log_S N)$ levels with a separate level for sources and targets of aggregation messages, respectively. You may further assume that we have sufficient long string of "public random bits", which can be used to make random decisions that are the same for all machines, (since all machines utilize the same random bit string).

Sample Solution

We arrange the $O\big(\frac{N}{S} \log_S N\big)$ machines into $\ell := 1 + \lceil \log_{S/2} N \rceil \in O(\log_S N)$ levels $L_1, \ldots, L_\ell$ of $2\lceil \frac{N}{S} \rceil + 2$ machines each.Furthermore, we arrange the levels such that initially all messages are held only by machines in L_1 ("message sources"), and the targets of the aggregation messages are in level L_ℓ.

Outline: The idea of solving the above aggregation problem is to establish aggregation trees between machines of successive levels. There will be one aggregation tree for each aggregation group with leaves in L_1 and roots in L_ℓ. The messages are then send up the trees like in a converge cast. Machines will choose their according tree parents in the next level randomly, which ensures that no machine obtains too many messages *in expectation.*

Aggregation trees on first level: For level 1, we say that a machine in L_1 that has a message m with the aggregation group g_m, participates in the aggregation tree of that group g_m (a given machine can participate in multiple aggregation groups, as it holds multiple messages).

For each aggregation group g the machines in L_1 choose a random subset L_g^2 from the next level L_2 of size $|L_g^2| = N \cdot (\frac{2}{S})^2$. Note that all machines participating in the aggregation tree of group g can all agree on the same random set L_g^2 using the public randomness.

Then each machine that takes part in group g picks a random parent node from L_g^2. Note that this random decision is now independent from the parent choice of other machines! By doing this for all

machines in L_1 and for aggregation groups, each machine in L_1 will now have a parent node for each aggregation group it participates in, see Figure 1.

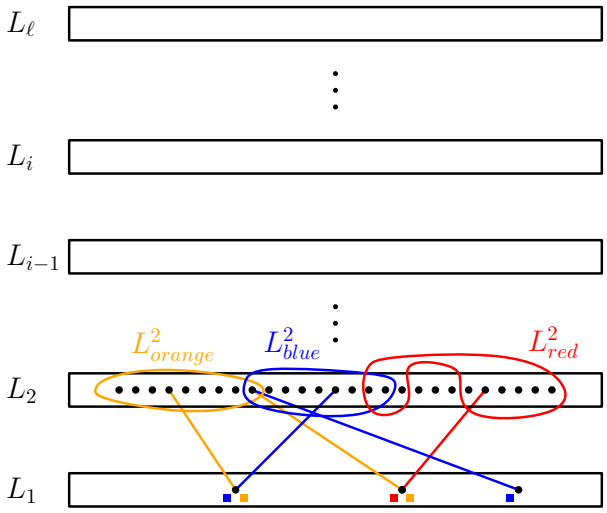

Figure 1: Three example machines in L_1 that have messages (little colored boxes) from three aggregation groups (orange, blue, red). We determine random sets of machines from $L^2_{orange}, L^2_{blue}, L^2_{red}$. Each machine picks a random parent from the according random set for each group it participates in.

Aggregation trees for subsequent levels: The description above forms the base case above and now we describe how to connect L_{i-1} to level L_i for $2 < i \leq \ell - 1$. We say that a machine $\mu \in L_{i-1}$ participates in aggregation group g if it is in the according set $\mu \in L^{i-1}_g$.

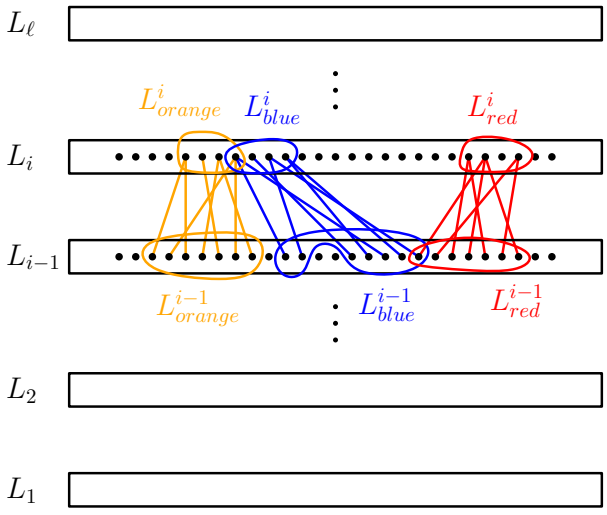

Figure 2: Nodes in L^{i-1}_g choose a parent uniformly at random from L^i_g (g = orange, blue, red).

Similar as before, for each aggregation group g the machines in L_{i-1} choose a random subset L^i_g from the next level L_i of size $|L^i_g| = N \cdot (\frac{2}{S})^i$. Again, all machines agree on the same random sets L^i_g using public randomness. Let μ be a machine that participates in aggregation group g. As before, it chooses a parent in L^i_g uniformly at random (and also independent from other machines), c.f. Figure 2.

We have to make a distinction for the last level ℓ. There we simply connect all nodes in aggregation group $L^{\ell-1}_g$ to the respective target $t \in L_\ell$ (c.f., Figure 3).

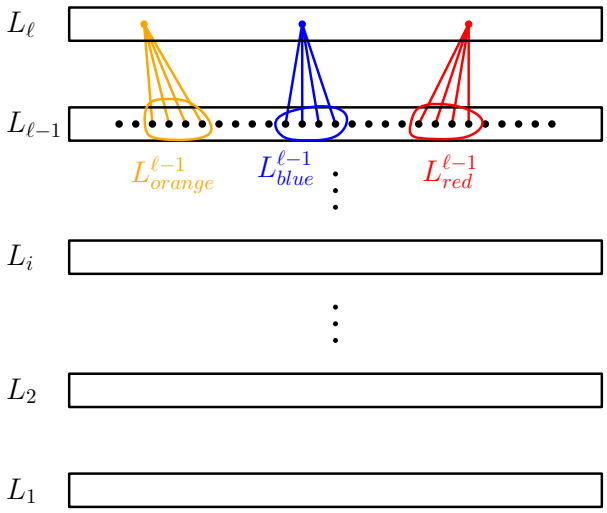

Figure 3: Nodes in $L_g^{\ell-1}$ create an edge to the target machine of group g (g = orange, blue, red).

Aggregation Algorithm: In round i, every machine in level i sends for every aggregation group g_m it has a message of, *one* message m with smallest value v_m among all messages that this machine has from that group g_m to its parent in $L_{g_m}^{i+1}$.

Analysis: Since one minimum value message of each group will always be forwarded to the target (root) of the aggregation tree, the target machine will eventually learn a minimum value message of its group, thus the algorithm is correct.

The runtime is also not hard to see. We have to iterate through the layers of the above structure and in each layer first determine parent machines and then send the messages to the appropriate parent machines. This takes $O(1)$ per layer and we have $\ell \in O(\log_S N)$ layers.

Now we argue why the property of these aggregation trees that each node obtains at most S messages, is true *in expectation*. Let $\mu \in L_i$. Since we pick $N(\frac{2}{S})^i$ machines for L_g^i out of the level L_i of size $2\lceil N/S \rceil + 2$, we have that

$$\mathbb{P}\big(\mu \in L_g^i\big) = \frac{|L_g^i|}{|L_i|} = \frac{N(\frac{2}{S})^i}{2\lceil N/S \rceil + 2} \leq \frac{2^{i-1}}{S^{i-1}}.$$

The probability that μ is chosen as parent by some $\mu' \in L_g^{i-1}$, conditioned on the above event is

$$\mathbb{P}\big(\mu \text{ parent of } \mu' \mid \mu \in L_g^i\big) = \frac{1}{|L_g^i|} = \frac{S^i}{N \cdot 2^i}.$$

The general probability that μ is parent of some fixed $\mu' \in L_{i-1}$ is

$$\mathbb{P}\big(\mu \text{ parent of } \mu'\big) = \mathbb{P}\big(\mu \text{ parent of } \mu' \mid \mu \in L_g^i\big) \cdot \mathbb{P}\big(\mu \in L_g^i\big) \leq \frac{S}{2N}.$$

Let $i < \ell$. Consider a message that some machine $\mu' \in L_{i-1}$ has. The probability that $\mu \in L_i$ is the recipient of this message is then $\mathbb{P}\big(\mu \text{ parent of } \mu'\big)$. Since on each level the maximum number of messages is at most N, we have that μ gets at most $N \cdot \mathbb{P}\big(\mu \text{ parent of } \mu'\big) \leq \frac{S}{2}$ messages in expectation.

Let $i = \ell$ (last level). We have to argue that the targets do not have a large in degree. Since we chose $\ell = 1 + \lceil \log_{S/2} N \rceil$, we have that

$$|L_g^{\ell-1}| = N \cdot \big(\frac{2}{S}\big)^{\ell-1} \leq N \cdot \big(\frac{2}{S}\big)^{(\log_{S/2} N)-1} = N \cdot \big(\frac{2}{S}\big)^{\log_{S/2} N} \cdot \frac{S}{2} = N \cdot \frac{1}{N} \cdot \frac{S}{2} = \frac{S}{2},$$

therefore, machine μ in level ℓ will be parent of at most $\frac{S}{2}$ many machines in level $L_g^{\ell-1}$, and since each machine is target of at most one aggregation group will get at most $\frac{S}{2}$ messages.

Exercise 2: Implement a Phase of Borůvka's Algorithm *(5 Points)*

In class, we sketched how to implement one phase of Borůvka's MST algorithm in the strongly sublinear regime $S = n^\alpha$ for some constant $0 < \alpha < 1$. Argue in more detail how this can be done in $O(1)$ rounds, given that we can solve the above aggregation problem.

Sample Solution

Recall that Borůvka's algorithm iteratively constructs MSTs on ever larger subgraphs, by connecting pairs of trees with a minimum weight edge between them in each phase. Each machine was responsible for such a tree, or rather the set of nodes connected by that tree, which we called a *fragment*.

Each fragment has in ID (usually the smallest node ID in the fragment). During the algorithm we had to maintain the invariant that for each edge the machine that stores that edge has to know which fragments its endpoints are in. The machine responsible for a given fragment then has to learn the minimum weight edge that is ougoing from its fragment, and have to do this for all fragments. Clearly this is an aggregation problem.

For each fragment ID x we generate an aggregation group and the target of this group is the machine responsible for fragment x. Then each machine that has an edge with exactly one endpoint in fragment x makes a message containing that edge and associated information. We then solve the aggregation problem above. Afterwards the machine responsible for x knows the minimum weight outgoing edge for fragment x and can merge the corresponding fragments.

We can then broadcast the information of merged fragments down the aggregation trees again (which is in some sense the "reverse" of the prior converge cast and does not take any longer). After that broadcast all machines know the *new* fragments of their edges and the next Borůvka phase can start.

We have that $S = n^\alpha$ for a constant α. We have at most $N = n^2$ messages. The runtime to solve the aggregation problem and to do the broadcast down the aggregation trees is

$$\mathcal{O}(\log_S N) = \mathcal{O}(\log_{n^\alpha}(n^2)) = \mathcal{O}(2\log_{n^\alpha} n) = \mathcal{O}(\frac{1}{\alpha}).$$