ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

FS 2026 Dr. G. Zuzic

Principles of Distributed Computing
Sample Solution to Exercise 13

1 Pipelining

The goal of this Section is to prove the following Lemma.

Lemma 1 (Pipelining in a Tree)**.** *Let G be a network graph and T a spanning tree of depth D in G. Given k subsets of nodes (fragments) $S_1, S_2, \ldots, S_k$ (not necessarily connected or disjoint), Algorithm 1 performs echo operation within each fragment simultaneously in $O(D + k)$ rounds.*

Concretely, each node x initially knows some unique $O(\log n)$-bit fragment $ID(x)$ and some private $O(\log n)$-bit integer x_v. Upon completion of Algorithm 1, each node v learns the min, max, and sum of the inputs in its fragment $\{x_w \mid ID(w) = ID(v)\}$.

Algorithm 1 Fragment-wise Pipelined Flood/Echo

```
1:  Input: Rooted spanning tree T of depth D, each node v has fragment ID ID(v) and value x_v
2:  Phase 1: Preprocessing
3:  Each node computes its height h(v) via flood/echo on T
4:  Each node learns the values of D and k (number of distinct fragments)
5:
6:  Phase 2: Convergecast
7:  for t = 1 to D + k do
8:    for all non-root nodes v in parallel do
9:      Let M be the set of messages not yet forwarded to parent
10:     Group messages in M by fragment ID and aggregate (min, max, sum)
11:     if M ≠ ∅ then
12:       Forward the message with the smallest fragment ID to parent
13:
14: Phase 3: Broadcast
15: (to be filled in)
```

Question 1 Prove that after Phases 1 and 2, the root of T learns the min, max, and sum of all inputs $\{x_w \mid ID(w) = f\}$ in each fragment f.

Solution:

Proof. Let T be a spanning tree of G, rooted at node r, with depth D. Define the height $h(v)$ of a node v as the maximum distance from v to any leaf node in its subtree.

Correctness Analysis (Inductive Proof) Order fragment identifiers as $ID_1 < ID_2 < \cdots < ID_k$. We prove the following inductively on $h(v) + i$ where $i \in \{0, 1, \ldots, k-1\}$:

Claim: After round $h(v)+i$, all information regarding fragment ID_i from node v's subtree has been forwarded to v's parent.

Base Case ($h(v) = 0$): For leaf nodes, after i rounds, messages corresponding to the first i smallest fragment identifiers have certainly been sent to their parents. Thus, the base case holds trivially.

Inductive Step: Suppose the claim holds for all nodes with height at most $h(v) - 1$. Consider a node v with height $h(v)$ at round $h(v) + i$:

- All children c of node v have height $h(c) \leq h(v)-1$. Thus, by induction, at round $h(v)+i-1$, each child c has already forwarded all messages for fragment ID_i upwards to v.
- Additionally, node v has already forwarded messages corresponding to all fragments smaller than ID_i to its parent, ensuring no interference.
- Therefore, at round $h(v) + i$, node v forwards the aggregated message corresponding to fragment ID_i to its parent (if it hasn't already done so). If it already forwarded this message earlier, the claim still holds.

Hence, by induction, after $O(D + k)$ rounds, all fragment information is guaranteed to have reached the root r.

□

Question 2. Consider the message forwarding rule in line 12 of Algorithm 1: "Forward the message with the smallest fragment ID to parent." Suppose this rule is changed such that each node v with $M \neq \emptyset$ forwards an **arbitrarily chosen** aggregated message from M to its parent in each round where it is active. Would the algorithm still guarantee that the root learns the correct aggregates for all k fragments after $O(D + k)$ rounds? If yes, provide a proof. If not, provide a counter-example.

Solution: No, we will show that the root needs $\Theta(D \cdot k)$ rounds to learn the aggregate (e.g., the max) of all the fragments.

It is relatively straightforward to prove that this upper bound holds: consider each individual fragment i. After k rounds, every node v at height 1 will know the aggregate (e.g., min) corresponding to fragment i in its own subtree. Hence, after $2k$ rounds, every node at height 2 will know the same. Continuing, after $D \cdot k$ rounds, the root will know the aggregate of (each) fragment, hence $O(D \cdot k)$ rounds is sufficient.

We now focus on the lower bound.

Lemma 2 (Lower bound)**.** *There exists a recursively constructed tree of depth D and a forward schedule such that the root learns all aggregates at round $\Omega(D \cdot k)$.*

Proof. We will make a recursive tree construction of depth D that shows this.

Let $f(1, i)$ be a singleton leaf node which we will later connect in a larger tree. It has information about all k fragments that it sends in arbitrary order along its parent edge. However, we ensure that at round k it must send the information about its i'th fragment up the tree. E.g. if $i = 3$, then the order in which it sends information to its parent edge might be $1, 2, 4, 5, \ldots, k, 3$ at rounds $1, 2, 3, \ldots, k$, respectively (ensuring that at round k it sends the fragment 3 info up the tree).

Let $f(d, i)$ be a tree with depth d with the guarantee that, in round $d \cdot k$, it is forced to send a message along its parent edge and that message contains information corresponding to fragment i. We construct it as follows:

- Create the root node r.
- Create trees $f(d-1, 1), f(d-1, 2), \ldots, f(d-1, k)$ and connect r to the roots of all of them (where they are the children of r. It is clear that the depth of our new tree is d.
- Note that, by assumption, during round $(d-1)k$ we will receive information about all k fragments. Therefore, r is forced to forward information about all k fragments along r's

parent edge in some order during rounds $(d-1)k+1, (d-1)k+2, \ldots, (d-1)\cdot k + k = d \cdot k$. We can send them in arbitrary order but we need to make sure that at round $d \cdot k$ we send information about fragment i.

Finally, consider the tree $f(D, 1)$ (note that 1 is arbitrary since there is no parent edge). By assumption, it will learn the information about fragments at round $(D-1)k = \Omega(D \cdot k)$. □

Question 3. Fill in the details in Phase 3 to ensure each node v learns the min, max, and sum of the inputs in its fragment. Argue why it works.

Solution: Let $T = D + k$. For each round $t = 1, 2, \ldots, T$ and each tree edge $e = (v \to p(v))$, define

$$m_{t,e} = \begin{cases} \text{ID corresponding to the message sent on } e \text{ at time } t \text{ in Phase 2,} & \text{if any,} \\ \perp, & \text{otherwise.} \end{cases}$$

Phase 3 then runs for $t = 1$ to T as follows:

```
1: Phase 3: Broadcast
2: for t = 1 to T do
3:   for all edges e = (v → p(v)) in parallel do
4:     let m ← m_{T−t+1, e}
5:     if m ≠ ⊥ then
6:       send message m from p(v) down to v
```

Proof. In Phase 2 each fragment-aggregate traverses each edge e in some round $t \le T$. By construction, in Phase 3 that same message is sent back down e in round $T - t + 1$. Along any root-v path of length $h(v)$, if the fragment's message first reached the root no later than round $t + h(v) \le T$, then it returns to v by round

$$(T - t + 1) + h(v) \;\le\; (T - (h(v) + 1) + 1) + h(v) \;=\; T.$$

Thus every node v receives its fragment's (min,max,sum) by round $T = D + k$. □

2 Merging Fragments

Consider an undirected connected graph $G = (V, E)$ where $n = |V|$. Suppose that each node $v \in V$ has selected one of its incident edges (v, u) as the *proposal edge* of v; denote it $e_v = (v, u)$. For instance, in the MST algorithm of GHS/Boruvka, this would be the minimum-weight edge incident on v. Notice that the two endpoints of an edge might propose the same edge simultaneously. Consider the random process where each node flips a fair coin for itself. Then, we mark the proposed edge $e_v = (v, u)$ of node v only if v draws tail and u draws head.

Question 1. Prove that, in expectation, we mark at least $n/8$ edges, provided that $n \ge 2$.

Solution: Since every node selects a proposal edge and every proposal edge can be selected by at most two nodes, we get that at least $n/2$ distinct proposal edges are selected. Since the coin tosses are independent, we get that any proposal edge is marked with probability $1/4$. Let P be the set of proposal edges, so $|P| \ge n/2$, and let $X = \sum_{e \in P} X_e$, where X_e is an indicator random variable that has value 1 if e is marked and 0 otherwise. Given the above observations we can bound $\mathbb{E}[X] \ge |P| \cdot (1/4) \ge (n/2)/4 = n/8$.

Question 2. Prove that, if we contract all the marked edges, the resulting graph has at most $7n/8$ nodes, in expectation, provided that $n \ge 2$.

Solution: For every contraction, at least one node is "removed" from the graph as a result of the contraction. Thus, the expected number of remaining nodes is at most $n - \mathbb{E}[X] \le (7/8)n$.

Question 3. Consider repeating the above process for $20 \log n$ iterations: In each iteration, we contract all the marked edges, and remove self-loops. Then, select one (e.g., min-weight) incident edge per new node, and repeat the marking process as above using one coin toss per each new node. Use Question 2 to prove that, after $20 \log n$ iterations, with high probability, we have contracted everything to a single node.

Solution: Let X_i be the random variable whose value is the number of nodes after i iterations, minus 1 . Since, for any number $k \geq 2$ of nodes at the beginning of an iteration, the number of nodes drops, in expectation, to a value that is at most $(7/8)k$ at the end of the iteration (by (2 b)), the expected value of "the number of nodes -1 " at the end of the iteration is at most $(7/8)k - 1 < (7/8) \cdot (k-1)$. Also if we start with $k = 1$ nodes at the beginning of an iteration, "the number of nodes -1 " drops by at least a factor of $7/8$ in expectation since it is 0 at the beginning and the end of the iteration. Hence, we have $\mathbb{E}[X_i] \leq (7/8) \cdot \mathbb{E}[X_{i-1}]$, for any $i \geq 1$. We obtain

$$\mathbb{E}[X_{20 \log n}] \leq n \cdot (7/8)^{20 \log n} \leq n \cdot (1/2)^{3 \log n} \leq n \cdot n^{-3} = 1/n^2$$

By Markov's inequality, it follows that $\Pr(X_{20 \log n} \geq 1) \leq 1/n^2$. This implies, by the definition of our random variables, that the probability that more than one node remains after $20 \log n$ iterations is bounded from above by $1/n^2$.