ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

FS 2017

Prof. Dr. Mohsen Ghaffari,
Prof. Dr. Roger Wattenhofer

Exam
Principles of Distributed Computing

Monday, August 14, 2017
14:00 – 16:00

Do not open or turn until told to by the supervisor!

The exam lasts 120 minutes, and there is a total of 120 points. The maximal number of points for each question is indicated in parentheses. Your answers must be in English. Be sure to always justify (prove) your answers. Algorithms can be specified in high-level pseudocode or as a verbal description. You do not need to give every last detail, but the main aspects need to be there. Big-O notation is acceptable when giving algorithmic complexities. Please write legibly. If we cannot read your answers, we cannot grade them.

Please write down your name and Legi number (your student ID) in the following fields.

Name	Legi-Nr.

Exercise	**Achieved Points**	**Maximal Points**
1 - Multiple Choice		12
2 - Arrow and Ivy		12
3 - Sorting Networks		14
4 - Very Small Radius		26
5 - Global Computations		16
6 - Special Coloring		20
7 - Combinatorics of Radio Transmissions		20
Total		**120**

1 Multiple Choice (12 points)

Evaluate each of the following statements in terms of correctness. Indicate whether a statement is true or false by ticking the corresponding box. Each correct answer gives one point. Each wrong answer and each unanswered question gives 0 points.

Statement	**true**	**false**
Any rooted binary tree with n nodes can be 1-colored in time $O(n)$.	☐	☐
Any rooted binary tree with n nodes can be 2-colored in time $O(\log n)$.	☐	☐
Any rooted binary tree with n nodes can be 3-colored in time $O(\log^* n)$.	☐	☐
Any rooted binary tree with n nodes can be 4-colored in time $O(1)$.	☐	☐
The flooding algorithm can be used to determine if a graph is a tree.	☐	☐
The Gallager-Humblet-Spira algorithm can be modified to compute a spanning tree of maximum weight by defining a blue edge to be the maximum weight outgoing edge.	☐	☐
In Luby's Maximal Independent Set (MIS) algorithm, in each round each node is removed with probability at least 1/10.	☐	☐
Any planar graph with n nodes can be 7-colored in time $O(\log^* n)$.	☐	☐
In any graph with maximum degree at most 5, any maximal independent set has size at least 1/5 of the maximum independent set.	☐	☐
Suppose that Alice knows 5 message each with 100-bits, Bob knows 4 of these messages, and Alice doesn't know which ones are known to Bob. Alice needs to send at least 200 bits to Bob so that he also knows all the messages.	☐	☐
Any labeling scheme for distance in cycles needs to use labels of size at least $\Omega(\log^2 n)$.	☐	☐
There exists at least one task on unweighted simple graphs such that any labeling scheme needs to use labels of size at least $\Omega(n^3)$.	☐	☐

Solutions

Statement	true	false
Any rooted binary tree with n nodes can be 1-colored in time $O(n)$. *Reason: There is no 1-coloring for graphs with more than one vertex.*		✓
Any rooted binary tree with n nodes can be 2-colored in time $O(\log n)$. *Reason: Far away nodes tell you you need linear time.*		✓
Any rooted binary tree with n nodes can be 3-colored in time $O(\log^* n)$. *Reason: Theorem 1.23.*	✓	
Any rooted binary tree with n nodes can be 4-colored in time $O(1)$. *Reason: If that was possible, we could use Algorithm 1.21 to get down to three colors in time $O(1)$.*		✓
The flooding algorithm can be used to determine if a graph is a tree. *Reason: A node that gets more than one message to join the tree knows that it is in a cycle.*	✓	
The Gallager-Humblet-Spira algorithm can be modified to compute a spanning tree of maximum weight by defining a blue edge to be the maximum weight outgoing edge. *Reason: Just replace the minimum with the maximum in the proofs in the lecture notes.*	✓	
In Luby's Maximal Independent Set (MIS) algorithm, in each round each node is removed with probability at least 1/10. *Reason: Low-degree nodes whose neighbors all have high degrees have a tiny chance of being removed.*		✓
Any planar graph with n nodes can be 7-colored in time $O(\log^* n)$. *Reason: Linial's lower bound (Theorem 7, Lecture 7). Any tree is planar and coloring un-oriented trees with less than $o(\Delta/\log\Delta)$ colors needs at least $\Omega(\log_\Delta n)$ rounds.*		✓
In any graph with maximum degree at most 5, any maximal independent set has size at least 1/5 of the maximum independent set. *Reason: A simple blaming argument a la Lemma 1 of lecture 8.*	✓	
Suppose that Alice knows 5 message each with 100-bits, Bob knows 4 of these messages, and Alice doesn't know which ones are known to Bob. Alice needs to send at least 200 bits to Bob so that he also knows all the messages. *Reason: Sending the XOR would allow to complete Bob's knowledge with 100 bits.*		✓
Any labeling scheme for distance in cycles needs to use labels of size at least $\Omega(\log^2 n)$. *Reason: For cycles, it is sufficient to give the nodes numbers from 1 to n in clockwise order, which leads to label size $O(\log n)$.*		✓
There exists at least one task on unweighted simple graphs such that any labeling scheme needs to use labels of size at least $\Omega(n^3)$. *Reason: One can store the whole graph on every node using $O(n^2)$ bits.*		✓

2 Arrow and Ivy (12 points)

In this task we want to compare the two main algorithms for shared objects, Arrow and Ivy. Figure 1 depicts a precomputed spanning tree of a complete graph G. The nodes v_1, v_2 and v_3 want to access the object which is stored at the node w. Assume that the requests are filed sequentially, i.e. a node files a request after the previous node has already received the object.

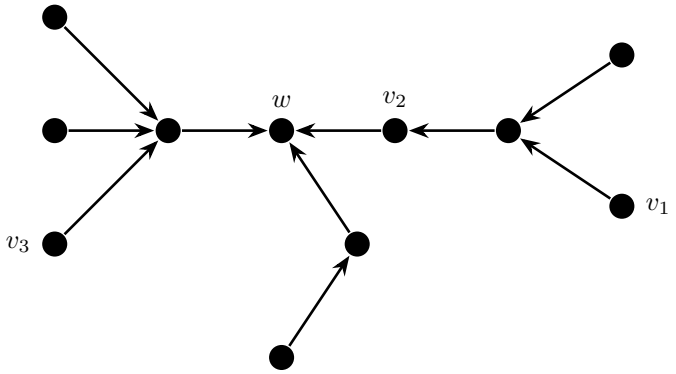

Figure 1: A precomputed spanning tree of G

A) [4] Give the worst-case ordering of the requests v_1, v_2, v_3 if you apply Arrow.

B) [4] Give the best-case ordering of the requests if you apply Ivy.

C) [4] Your friend claims that on average the Arrow algorithm is strictly better than the Ivy algorithm for G with this precomputed spanning tree and any sequential request sequence. Is your friend correct?

Solutions

A) v_1, v_3, v_2 is the worst-case sequence. Note that v_1 and v_3 should be two consecutive requests, since placing v_2 in the middle would minimize the distance to both v_1 and v_2. v_2 should be the last request becaue it is closest to the object. Now we need to observe that the sequence v_1, v_3, v_2 gives a larger sum of all paths than v_3, v_1, v_2.

B) v_3, v_2, v_1 is the best-case sequence. In this case, v_1 and v_2 should be consecutive requests because they lie on the same branch of the tree with the object as the root. Starting with v_1 maximizes the distance for v_2 to get the object. The sequence v_2, v_1, v_3 gives a larger distance than the sequence v_3, v_2, v_1.

C) No. Worst-case average running time for Ivy is $\log(n) = \log(11)$, while the worst-case sequence for ARROW is diameter of the tree ($D = 5$), which is larger.

3 Sorting Networks (14 points)

A) [4] Consider a correct sorting network. Show that for each pair of adjacent wires i, $i+1$, the network contains a comparator that compares wires i and $i+1$.

B) [10] For each integer $n \geq 2$ determine whether there exists a correct sorting network of width n that contains exactly one comparator for each pair of adjacent wires. (There is no restriction on the number of comparators for non-adjacent wires.) If the answer is yes, provide a construction of a sorting network as described. If the answer is no, show that a sorting network as described cannot exist.

Solutions

A) Let S be a sorting network such that there is no comparator that compares wires i and $i+1$. Consider the input sequence I that consists of $i-1$ zeros, a one at position i, a zero at position $i+1$, and the rest ones.

None of the wires will change I, because I is already sorted except for positions i and $i+1$. Hence, the output will equal I and therefore the sorting network is not correct.

B) Such a network exists. There are different possibilities on how to construct such a network. The following one is basically a distributed version of the well-known *selection sort* algorithm.

Compare the first wire to wires $2, 3, 4, \ldots, n$. After that, the smallest value of the sequence will be in the first wire.

After that, compare the second wire to wires $3, 4, 5, \ldots, n$ to get the second-smallest value into wire 2.

Similarly, add a comparator between wire i and $i+1$, then between i and $i+2$, etc. and one comparator between i and n to get the i-th smallest value into the i-th wire.

In this network, there is exactly one comparator between each pair of adjacent wires i and $i+1$.

4 Very Small Radius (26 points)

In the lecture, we studied the diameter of a graph. A very related notion is the radius of a graph. Throughout this exam question, we assume that all considered graphs are connected.

The radius $r(u)$ of a node u in a graph G is the maximum of the distances from u to all other nodes, i.e., $r(u) = \max_{v \in G}(\text{dist}(u, v))$. The radius $r(G)$ of a graph G is the minimum of the radii of all nodes in G, i.e., $r(G) = \min_{u \in G}(r(u))$.

We want to study the problem of determining whether a graph has radius 1 or not.

A) [4] Radoslav claims that if the radius of a graph is 1, then its diameter is always 2. Diana claims that if the diameter of a graph is 2, then its radius is always 1. Who is right, who is wrong?

B) [22] Assume that the nodes do not know n, i.e. the total number of nodes in the graph. Design a synchronous deterministic distributed algorithm that determines if the input graph has radius 1 or not. Each node has a unique $O(\log n)$-bit identifier. In each round each node can send a message that has $O(\log n)$ bits over each incident edge.

You can obtain up to 5 points if your algorithm has the following properties:

1) If the radius is 1, then all nodes output "YES".

2) If the radius is not 1, then at least one node outputs "NO".

You can obtain up to 5 **additional** points if the runtime of your algorithm is **constant**.

You can obtain up to 12 **additional** points if the runtime of your algorithm is **constant** and the algorithm has the following properties:

1) If the radius is 1, then all nodes have to output "YES".

2) If the radius is not 1, then all nodes have to output "NO".

Solutions

A) Both are wrong.

Consider the graph which consists of only two vertices which are connected by an edge. Clearly, this graph has both radius 1 and diameter 1.

Furthermore, consider the cycle on four vertices. It has radius 2 and diameter 2.

B) Sample solution for 10 points

In round 1, each node v sends the pair $(d(v), \mathrm{ID}(v))$, consisting of its own degree and ID to all its neighbors. In round 2, each node sends the "largest" pair it has received or sent in round 1 to all neighbors, where a pair is considered as larger than another pair if and only if it contains either the larger degree or the same degree, but the larger ID. If a node receives a pair in round 2 that it has neither received nor sent in round 1, then it outputs "NO", otherwise it outputs "YES".

It follows directly from the algorithm description that the runtime of the algorithm is 2, i.e., constant. We show correctness as follows:

If the radius is 1, then there must be a node that is connected to all other nodes. In particular, the node u with largest $(d(u), \mathrm{ID}(u))$ is connected to all other nodes. Each node receives or sends $(d(u), \mathrm{ID}(u))$ in round 1, and receives nothing else than $(d(u), \mathrm{ID}(u))$ in round 2. Hence, each node outputs "YES".

If the radius is not 1, then there is a node in distance 2 to the node u with largest $(d(u), \mathrm{ID}(u))$. This node does not receive or send $(d(u), \mathrm{ID}(u))$ in round 1, but it receives $(d(u), \mathrm{ID}(u))$ in round 2. Hence, it outputs "NO".

Sample solution for 22 points

A possible algorithm for determining in constant runtime whether the radius is 1 (and also adhering to the stronger specification that all nodes have to output "NO" if the radius is not 1) is the following:

In round 1, each node v sends the pair $(d(v), \mathrm{ID}(v))$, consisting of its own degree and ID to all its neighbors. Then each node v determines the "largest" pair (let's call it $P(v)$) of all pairs it has received in round 1 and the pair v sent itself. For this, a pair $(d(u), \mathrm{ID}(u))$ is considered as larger than a pair $(d(w), \mathrm{ID}(w))$ if and only if either $d(u) > d(w)$ holds or both $d(u) = d(w)$ and $\mathrm{ID}(u) > \mathrm{ID}(w)$ hold. Note that $P(v)$ may be different from $(d(v), \mathrm{ID}(v))$.

In round 2, each node v sends $P(v)$ to all neighbors. In round 3, each node v sends "no" to all neighbors if v received at least one pair different from $P(v)$ in round 2 (otherwise v does not send a message). In round 4, each node sends "no" to all neighbors if it sent or received a "no" in round 3 (otherwise the node does not send a message).

Now, if a node sent or received a "no" in round 4, it outputs "NO", otherwise it outputs "YES".

It follows directly from the algorithm description that the runtime of the algorithm is 4, i.e., constant. In the following we show that the algorithm is correct, i.e., that it satisfies Properties 1) and 2) (of the latter specification).

First, we consider the case that the radius is 1. Then, by definition, there must be a node, say v, that has radius 1. Hence, v is directly connected to all other nodes and has degree $n-1$. Let u denote the node with degree $n-1$ that has the largest ID. (Since v has degree $n-1$, we know that u exists.) Observe that u is directly connected to all other nodes.

Hence, in round 1, each node receives or sends $(d(u), \mathrm{ID}(u))$. By the definition of w, we have $(d(u), \mathrm{ID}(u)) > (d(w), \mathrm{ID}(w))$ for all $w \neq u$. Thus, $P(w) = (d(u), \mathrm{ID}(u))$ for all nodes w. In particular, each node w only receives $P(w)$ in round 2. Therefore, no node sends "no" in round 3, hence also no "no" is sent in round 4. It follows that all nodes output "YES", which shows Property 1).

Second, consider the case that the radius is not 1. We claim that each node v has a node w in its (inclusive) 2-hop-neighborhood that receives a message different from $P(w)$ in round 2:

Let v be an arbitrary node. Let u be the node from v's (inclusive) 2-hop-neighborhood that maximizes $(d(u), \mathrm{ID}(u))$. If u is not contained in v's (inclusive) 1-hop-neighborhood, then v receives $(d(u), \mathrm{ID}(u))$ in round 2, which is different from $P(v)$. Hence, setting $w := v$ proves the claim. If u is contained in v's (inclusive) 1-hop-neighborhood, then consider a node x that has distance 2 from u. Since we are in the case that the radius is not 1, such a node x must exist. From the definition of x, it follows that $P(x) \neq (d(u), \mathrm{ID}(u))$, whereas $P(u) = (d(u), \mathrm{ID}(u))$, by the definition of u. Let y be a neighbor of both u and x. (Such a node y must exist because $\mathrm{dist}(u, x) = 2$.) Since y receives both $P(u)$ and $P(x)$ in round 2 and $P(u) \neq P(x)$, it receives at least one message different from $P(y)$ in round 2. Thus, setting $w := y$ proves the claim. Note that, by the triangle inequality, y is in contained in the (inclusive) 2-hop-neighborhood of v since $\mathrm{dist}(v, u) \leq 1$ and $\mathrm{dist}(u, y) = 1$.

Hence, the claim is true and it follows that each node v has a node w in its (inclusive) 2-hop-neighborhood that sends "no" in round 3. This, in turn, implies that there is a node in v's (inclusive) 1-hop-neighborhood that sends "no" in round 4. Therefore, v sends or receives a "no" in round 4, and v outputs "NO". It follows that all nodes output "NO", which shows Property 2).

5 Global Computations (16 points)

Model: The network is abstracted as an n-node undirected graph $G = (V, E)$, with diameter D. Each node has a unique $\Theta(\log n)$-bit ID. Initially, each node only knows its own edges, as well as the value of n. We consider the CONGEST model where per round each node can send $O(\log n)$ bits to each of its neighbors.

Problem: Suppose that each node $v \in V$ receives an input value $x_v \in \{1, 2, \ldots, n^{10}\}$. The objective is for all nodes to learn the average of the values of the $k = \lceil \sqrt{n^{0.4}} \rceil$ largest inputs. Suppose that the diameter is $D = \lceil \sqrt{n^{0.6}} \rceil$. Devise a fast algorithm for the problem. Provide a correctness argument for your algorithm and analyze its round complexity.

A correct algorithm receives **6 points**. Proving the algorithm's correctness receives another **5 points**, and obtaining a good round complexity (with analysis) receives the last **5 points**.

Solutions

Algorithm & Round Complexity

- First, we elect a leader, and then we build a BFS rooted in this leader, all in $O(D)$ rounds (see, e.g., Lecture 2).
- Then, we make nodes upcast the largest k-values to the root, in a pipelined fashion, in $O(D+k)$ rounds, as follows: The upcast start from the deepest level nodes—those at depth D of the BFS—and proceeds at a speed of one hop per round, towards the root. That is, each node v at depth d_v starts in round $D-d_v$. Notice that at this time, node v has received one message from each of its children (if it has any). Once a node node v starts, per round, it forwards to its BFS-parent the largest value that v has seen so far, conditioned on that it has not seen k values larger than this one. We prove that after $O(D+k)$ rounds, the root learns the largest k values.
- Then, the root computes the average of these values locally and then broadcasts it to all nodes in $O(D)$ additional rounds (see, e.g., Lecture 2).

Overall, the round complexity is $O(D+k) = O(n^{0.3} + n^{0.2}) = O(n^{0.3})$.

Correctness Argument The correctness of leader election, BFS construction, and then finally broadcast from leader is standard. Thus, the only part that needs to be proven is the second step, i.e., that after $O(D+k)$ rounds of the upcast step, the BFS-root learns the k largest values.

By induction on r, we prove that at the end of round $r = h+i$, each node v at height $h = D-d_v$ has sent the largest i values in its subtree to its parent. A small side-note is that the statement is inaccurate for the root, as the root has no parent to send to. But in that case, we use it to infer that the root has these messages (and it would have delivered them to its parent, if it had one).

The base case of $r = 1$ is trivial, as in that case the statement is only meaningful for nodes at depth D—i.e., height 0—, who are necessarily leaves, and they clearly have sent their largest value to their parent by the end of round 1.

For the inductive step, suppose that we know that at the end of round $r = h + i$, each node v' at height $h = D - d_{v'}$ has sent the largest i values in its subtree to its parent. We want to prove that at the end of round $r + 1 = h + i + 1$, each node v at height $h = D - d_v$ has sent the largest $i + 1$ values in its subtree to its parent. By the inductive assumption, at the end of round $r = h + i$, node v has received the largest $i + 1$ values of the subtrees of each of its children, as they are in height $h - 1$. Hence, at the end of round r, node v has seen all the $i + 1$ largest values in its subtree. Moreover, node v has already sent the largest i values in its subtree by the end of round r. Therefore, in round $r + 1$, it will send the $(i + 1)^{th}$ largest value in its subtree, unless it has already done so in the past. That is, by the end of round $r + 1$, node v has sent the $i + 1$ largest values in its subtree.

Now applying the claim to the root, and at the end of round $r = D + k$, we get that the root, who sits at height at most D, has received all the k largest messages in its subtree.

6 Special Coloring (20 points)

Model: The network is abstracted as an n-node undirected graph $G = (V, E)$. Initially, each node only knows its own edges, as well as the values of n and the maximum degree Δ. We consider the CONGEST model where per round each node can send $O(\log n)$ bits to each of its neighbors.

Problem: Each node v must choose a color value $c_v \in \{\Delta^{3/2}, \Delta^{3/2}+1, \ldots, 2\Delta^{3/2}\}$. Each edge $e = \{v, u\}$ is annotated with a *conflict description* (a_e, b_e) for some values $a_e \in \{10\Delta, 10\Delta + 1, \ldots, 20\Delta\}$ and $b_e \in \{0, 1, \ldots, a_e - 1\}$; we say that the coloring is invalid and has a conflict on edge e if we have $c_v \equiv b_e + c_u \pmod{a_e}$. The conflict description values (a_e, b_e) are known to both endpoints v and u of the edge e. Notice that the coloring is invalid if there is a conflict on any edge.

Devise a fast randomized distributed algorithm that computes a valid coloring, with high probability. The algorithm should terminate within your claimed round complexity and the computed coloring must be valid with probability at least $1-1/n$. Analyze your algorithm's round complexity and prove its correctness.

A correct algorithm receives **8 points**. Proving the algorithm's correctness receives another **4 points**, and obtaining a good round complexity (with analysis) receives the last **8 points**.

Solutions

Algorithm Per round, each node v picks a tentative color q_v uniformly at random from the range of all possible colors $\{\Delta^{3/2}, \Delta^{3/2}+1, \ldots, 2\Delta^{3/2}\}$. Then, if this tentative color is not in conflict with any of the tentative colors that neighbors picked, or the permanent colors that neighbors have already fixed in the past, this tentative color q_v becomes the permanent color of node v. All nodes do this in parallel. We prove that after $O(\log n)$ rounds, w.h.p., all nodes are colored.

Correctness Argument For each node v, each neighbor u blocks at most $\frac{2\Delta^{3/2}}{10\Delta} = \frac{\Delta^{1/2}}{5}$ of the colors of v. This is because, there is one color that u chose in this round as a tentative color, or had fixed in the past as a permanent color. Every two consecutive possible colors of v that are in conflict with this color c_u of u are a_e apart, where $e = \{u, v\}$. Hence, there are at most $\frac{2\Delta^{3/2}}{a_e} \leq \frac{2\Delta^{3/2}}{10\Delta} = \frac{\Delta^{1/2}}{5}$ such colors of v blocked by neighbor u. Since v has at most Δ neighbors, in total, at most $\Delta \cdot \frac{\Delta^{1/2}}{5} = \Delta^{3/2}/5$ of its color choices are blocked. Since this is less than the $\Delta^{3/2}$ color options that v has, there are always many colors available for v. Thus, the algorithm will eventually find a permanent color for v. Once v gets colored permanently, no neighbor can pick a color conflicting with that of v.

Round Complexity We prove that after $\log n$ rounds, all nodes are colored, with high probability. As argued above, per round, at most a $1/5$ fraction of the color choices of node v are blocked by neighbors. Hence, per round, the probability that node v receives a permanent color is at least $4/5$. Since node v picks independent random colors in different rounds, the probability that after $\log n$ rounds, node v still has no permanent color is at most $(1/5)^{\log n} \leq 1/n^2$. Using a union bound over all nodes, we get that after $\log n$ rounds, the probability that there is at least one node that remains uncolored is at most $n \cdot 1/n^2 = 1/n$. That is, with probability $1 - 1/n$, all nodes are colored within the first $\log n$ rounds.

7 Combinatorics of Radio Transmissions (20 points)

Model: We assume the *radio networks* model, which works as follows: The network is abstracted as an n-node undirected graph $G = (V, E)$. Each node has a unique $\Theta(\log n)$-bit ID. Initially, each node knows only the values of n and Δ, and its own identifier. Per round, each node either transmits a message or listens. A node receives a message only if it is listening and exactly one of its neighbors is transmitting a message, and in that case, the node receives the message of that single transmitting neighbor. If a node is not listening, or two or more of its neighbors are transmitting, then the node does not receive anything.

Problem: Suppose that each node v has a message m_v, which should be delivered to all of its neighbors. Devise a short and deterministic schedule for transmissions so that at the end, each node u receives the messages of all of its neighbors. The schedule describes for each node v what it should do in each round — i.e., whether it should transmit its message m_v or it should listen — as a function of n, Δ, and the node's identifier ID_v. The schedule must be independent of G and should work for any n-node graph.You should also argue about the schedule's correctness, and analyze its length as a function of n and Δ.

A correct algorithm receives **8 points**. Arguing about the algorithm's correctness receives another **4 points**, and obtaining a good schedule length receives the last **8 points**.

Hint: Think about *cover-free families*, from lecture 5.

Solutions

We want a schedule such that each node u and every neighbor v of u, there exists at least one round t in which u successfully receives message m_v from v, that is, v sends m_v (to u), and u as well as all neighbors of u except v listen in that round t. This is achieved using a Δ-cover-free family.

Schedule Let $k = 2^{\Theta(\log n)} = \text{poly } n$ and $S_1, \ldots, S_k \subseteq \{1, \ldots, k'\}$ be a Δ-cover-free family of size $k' = O(\Delta^2 \log k) = O(\Delta^2 \log n)$, which exists by Lemma 3 in Chapter 5. Node v uses the set S_{ID_v} as schedule of length k' as follows. In round $t \in [k']$, if $t \in S_{ID_v}$, v transmits message m_v to all its neighbors, and listens if $t \notin S_{ID_v}$. This schedule has length $k' = O(\Delta^2 \log n)$.

Correctness Argument We argue that each node u and every neighbor v of u, there exists a round $t \in [k']$ in which u receives message m_v from v. Let $I \subseteq [k]$ of size $|I| \leq \Delta$ be the set of IDs of nodes $(N(u) \cup \{u\}) \setminus \{v\}$ that have to listen. By definition of a Δ-cover-free family, we have $S_{ID_v} \not\subseteq \bigcup_{i \in I} S_i$, and thus there exists a $t \in [k']$ such that $t \in S_{ID_v}$ and $t \notin S_i$ for all $i \in I$. This means that in round t, v is sending message m_v and all the nodes $(N(u) \cup \{u\}) \setminus \{v\}$ are listening, thus u successfully receives message m_v.