Information Gathering in Ad-Hoc Radio Networks

Marek Chrobak
Department of Computer Science
University of California at Riverside

Kevin P. Costello
Department of Mathematics
University of California at Riverside

Leszek Gąsieniec
Department of Computer Science
University of Liverpool

Abstract

In the ad-hoc radio network model, nodes communicate with their neighbors via radio signals, without knowing the topology of the underlying digraph. We study the information gathering problem, where each node has a piece of information called a *rumor*, and the objective is to transmit all rumors to the designated target node. For the model without any collision detection we provide an $\tilde{O}(n^{1.5})$ deterministic protocol, significantly improving the trivial bound of $O(n^2)$. We also consider a model with a mild form of collision detection, where a node receives a 1-bit acknowledgement if its transmission was received by at least one out-neighbor. For this model we give an $\tilde{O}(n)$ deterministic protocol for information gathering in acyclic graphs.

2012 ACM Subject Classification Discrete Mathematics → Combinatorics • Combinatorial Optimization • Theory of Computation → Design and Analysis of Algorithms → Distributed Algorithms • Networks → Ad-Hoc Networks

Keywords and phrases algorithms, radio networks, information dissemination

Digital Object Identifier 10.4230/LIPIcs...

Funding M. Chrobak's research supported by NSF grant CCF-1536026.

1 Introduction

We address the problem of information gathering in ad-hoc radio networks. A *radio network* is represented by a directed graph (digraph) G, whose nodes represent radio transmitters/receivers and directed edges represent their transmission ranges; that is, an edge (u, v) is present in the digraph if and only if node v is within the range of node u. When a node u transmits a message, this message is immediately sent out to all its out-neighbors. However, a message may be prevented from reaching some out-neighbors of u if it collides with messages from other nodes. A collision occurs at a node v if two or more in-neighbors of v transmit at the same time, in which case v will not receive any of their messages, and it will not even know that they transmitted.

Radio networks, as defined above, constitute a useful abstract model for studying protocols for information dissemination in networks where communication is achieved via broadcast channels, as opposed to one-to-one links. Such networks do not need to necessarily utilize radio technology; for example, in local area networks based on the ethernet protocol all nodes communicate by broadcasting information through a shared carrier. Different variants of this model have been considered in the literature, depending on the assumptions about the node labels (that is, identifiers), on the knowledge of the underlying topology, and on allowed message size. In this work we assume that nodes are labelled $0, 1, ..., n-1$, where n is the network size. (All our results remain valid if the labels are selected from the range $0, 1, ..., O(n)$). We focus on the *ad-hoc model*, where the digraph's topology is unknown when the computation starts, and a protocol needs to complete its task within a desired time bound, no matter what the topology is. At the beginning of the computation each node v is in possession of a unique piece of information, that we refer to as a *rumor*. Different communication primitives are defined by specifying how these rumors need to be disseminated across the network. In this paper we do not make any assumptions about the size of transmitted messages; thus a node can aggregate multiple rumors and transmit them in one message. In fact, it could as well transmit as one message the complete history of its past computation.

The two most studied information dissemination primitives for this model are broadcasting and gossiping. In *broadcasting* (or *one-to-all communication*), a single source node s attempts to deliver its rumor to all nodes in the network. For broadcasting to be meaningful, we need to assume that all nodes in G are reachable from s. In *gossiping* (or *all-to-all communication*), the objective is to distribute all rumors to all nodes in the network, under the assumption that G is strongly connected. Both these primitives can be solved in time $O(n^2)$ by a simple protocol called RoundRobin, where all nodes transmit cyclically one at a time (see Section 2). Past research on ad-hoc radio networks focussed on designing protocols that improve this trivial bound.

For broadcasting, gradual improvements in the running time have been reported since early 2000's [6, 21, 2, 3, 12, 13, 11], culminating in the upper bound of $O(n \log D \log\log(D\Delta/n))$ in [10], where D denotes the diameter of G and Δ its maximum in-degree. This is already almost tight, as the lower bound of $\Omega(n \log D)$ is known [9]. For randomized algorithms, the gap between lower and upper bounds is also almost completely closed, see [1, 22, 11].

In case of gossiping, major open problems remain. The upper bound of $O(n^2)$ was improved to $\tilde{O}(n^{1.5})$ in [6, 29] and then later to $\tilde{O}(n^{4/3})$ in [18], and no better bound is currently known[1]. No lower bound better than $\Omega(n \log n)$ (that follows from [9]) is known. In contrast, in the randomized case it is possible to achieve gossiping in time $\tilde{O}(n)$ [11, 23, 7].

The reader is referred to survey papers [15, 20, 16, 27, 19] that contain more information about information dissemination protocols in different variants of radio networks.

In this paper we address the problem of *information gathering* (that is, *all-to-one communication*). In this problem, similar to gossiping, each node v has its own rumor, and the objective is to deliver these rumors to a designated target node t. (We assume that t is reachable from all nodes in G.)

The problem of information gathering for trees was introduced in [5], where an $O(n)$-time algorithm was presented. Other results in [5] include algorithms for the model without rumor aggregation or the model with transmission acknowledgements.

Our results. Our main result, in Section 4, is a deterministic protocol that solves the information gathering problem in arbitrary ad-hoc networks in time $\tilde{O}(n^{1.5})$. To our knowledge this is the first protocol for this problem that achieves running time faster than the trivial $O(n^2)$ bound. One of our key technical contributions is in solving this problem in time $\tilde{O}(n^{1.5})$ for acyclic graphs (Section 3). Previous protocols developed for gossiping on strongly connected graphs rely on feedback (see the discussion below), and are not applicable to this problem. This algorithm for acyclic graphs is based on careful application of

[1] We use notation $\tilde{O}(f(n))$ to conceal poly-logarithmic factors; that is, $g(n) = \tilde{O}(f(n))$ iff $g(n) = O(f(n) \log^c n)$ for some constant c. Also, we write $f(n) = \tilde{\Omega}(g(n))$ if and only if $g(n) = \tilde{O}(f(n))$.

combinatorial structures called strong selectors, combined with a novel amortization technique to measure progress of the algorithm. To extend this protocol to arbitrary graphs, we integrate it with a gossiping protocol. Roughly, the two sub-protocols run intertwined in parallel, with the sub-protocol for acyclic graphs transferring rumors between strongly connected components and the gossiping sub-protocol disseminating them within each strongly connected component. This requires overcoming two challenges. One is that the partition of G into strongly connected components is not actually known, so the combined protocol needs to gradually "learn" the connectivity structure of G while it executes. The second challenge is in synchronizing the computation of the two sub-protocols, since they are based on entirely different principles.

In the second part of the paper, in Section 5, we take the "dual" approach to investigate the time complexity of information gathering: rather than optimizing the running time in the general case, we examine what assumptions on the model would permit achieving running time $\tilde{O}(n)$. To this end, we consider a relaxation of our model by allowing a mild form of collision detection. In this new model each node v, after each transmission, receives a 1-bit acknowledgement indicating whether its transmission was received by at least one out-neighbor. With this assumption, we provide an $\tilde{O}(n)$-time algorithm for information gathering in acyclic radio networks.

Additional context and motivations. If G is strongly connected then information gathering and gossiping are equivalent. (This also naturally applies to connected *undirected* graphs.) Trivially, a gossiping algorithm gathers all rumors in t, solving the information gathering problem. On the other hand, one can solve the gossiping problem by running an information gathering protocol followed by any $\tilde{O}(n)$-time broadcasting protocol with source node t. However, unlike gossiping, the information gathering problem applies to a wider class of digraphs, namely all digraphs where the target node is reachable from all nodes.

This weakening of connectivity assumptions introduces new challenges for information gathering. The crucial one is *lack of feedback*, namely that the nodes in the network do not receive any information about the fate of their transmissions. This should be contrasted with the gossiping problem where the nodes can take advantage of strong connectivity to eventually learn whether their earlier transmissions were successful. In fact, the existing protocols for gossiping critically rely on this feature, as they use it to identify nodes that have collected a large number of rumors, and subsequently broadcast these rumors to the whole network, thus removing them from consideration and reducing congestion.

Some evidence that feedback might help to speed up information gathering can be found in [4], where the authors studied the model in which rumor aggregation is not allowed. In this model they developed an $O(n)$-time protocol for trees, under the assumption that nodes receive immediate acknowledgements of successful transmissions. In contrast, without feedback the best known upper bound (also from [4]) for this problem is $O(n \log\log n)$.

Various forms of feedback have been studied in the past in the context of *contention resolution* for multiple-access channels (MAC), where nodes communicate via a single shared channel. (Ethernet is one example.) Depending on more specific characteristics of this shared channel, one can model this problem as the information gathering problem either on a complete graph or a star graph, which is a collection of n nodes connected by directed edges to the target node t. (See [24, 26, 25, 14] for information about contention resolution protocols.) For instance, in [5] a tight bound of $\Theta(n \log n)$ was given for randomized information gathering on star graphs (or MACs) even if the nodes have no labels (are indistinguishable) and receive no feedback.

As explained earlier, in our model rumor aggregation is allowed. This capability is needed to beat the $O(n^2)$ upper bound, as without rumor aggregation it is quite easy to show a lower bound of $\Omega(n^2)$ for both gossiping and information gathering, even for randomized algorithms and with the topology known [17].

Interestingly, the randomized gossiping algorithms in [7, 23] can be adapted to information gathering, retaining their $\tilde{O}(n)$ expected running time. Thus randomization can help not only to overcome collisions, but also lack of feedback[2].

2 Preliminaries

Graph terminology. Throughout the paper, we assume that the radio network is represented by a digraph (directed graph) $G = (V, E)$ with a distinguished target node t that is reachable from all other nodes. By $n = |V|$ we denote the number of nodes in G. If $(u, v) \in E$ then we refer to u as the *in-neighbor* of v and to v as the *out-neighbor* of u. For any node v, by $N^-(v) = \{u \in V : (u, v) \in E\}$ we denote the set of its in-neighbors.

For brevity, we will refer to strongly connected components of G as *sc-components*. For each node v, the sc-component containing v will be denoted by $C(v)$. We partition the set of in-neighbors of v into those that belong to $C(v)$ and those that do not: $N^-_{\text{SCC}}(v) = N^-(v) \cap C(v)$ and $N^-_{\text{ACY}}(v) = N^-(v) \setminus C(v)$.

The *ancestor set* of v in G, denoted $Anc[v]$, is the set of all nodes of G from which v is reachable (via a directed path). Note that $C(v) \subseteq Anc[v]$. In fact, if A is an sc-component then all vertices in A have the same ancestor set. It will be thus convenient to extend this definition to sc-components of G; if A is an sc-component then its ancestor set is defined as $Anc[A] = Anc[v]$, for some arbitrary $v \in A$. The *proper ancestor set* of A is $Anc(A) = Anc[A] \setminus A$.

Radio networks. We now give the description of the radio network model that our results apply to. We first provide the standard definition, as used in the earlier literature. Later in this section we will show that some restrictions of this model can be relaxed, in order to simplify the algorithms and proofs.

As mentioned in the introduction we assume that each node of G has a unique label from the set $[n] = \{0, 1, ..., n-1\}$. For convenience, we will identify nodes with their labels, so a "node u" really means the node with label u. We assume that n is known. All our protocols still work correctly within the claimed time bounds if the label set is $[N]$ for $N = O(n)$, provided that N is known upfront (but not necessarily n).

The time is divided into discrete *time steps* numbered with non-negative integers. We assume that all nodes start to execute the protocol simultaneously at time step 0. At each step each node can be either in the *transmitting state*, when it can only transmit, or in the *receiving state*, when it can only listen to transmissions from other nodes. Only one message can be transmitted at each step. This is not an essential restriction because, as already mentioned, we are not imposing any restrictions on the size or format of messages transmitted by nodes.

If a node u transmits a message at a time τ, this message is sent to all out-neighbors of u in the same step. If v is one of these out-neighbors then v will receive this message

[2] We should point out, however, a subtle difference. In the randomized case of information gathering, while gathering all rumors in the target node will indeed complete in expected time $\tilde{O}(n)$, the nodes in the network have no way to determine whether the process has completed, except for simply just waiting for $O(n^2)$ steps. In case of gossiping, the expected running time of $\tilde{O}(n)$ includes broadcasting the confirmation of the process' completion to the whole network.

at time τ provided that v is in the receiving state and that u is the only in-neighbor of v that transmits at time τ. If there are two or more in-neighbors of v that transmit at time τ then a *collision occurs*, and v does not receive any information (independently of its current state). In other words, collisions are indistinguishable from absence of transmissions. There is no feedback mechanism available in this model, that is a sender of a message does not receive any information as to whether its transmission was successful or not. (We will relax this restriction later in Section 5.)

Selectors. A *strong* (n,k)*-selector* is a sequence of label sets $S = (S_0, S_1, ..., S_{\ell-1})$ (that is, $S_i \subseteq [n]$ for each i) that "singles out" each label from each subset with at most k labels, in the following sense: for each $X \subseteq [n]$ with $|X| \leq k$ and each $x \in X$ there is an index i such that $S_i \cap X = \{x\}$. It is known [8] that there exist strong (n,k)-selectors of size $\ell = O(k^2 \log n)$.

Such selectors are often used in protocols for ad-hoc radio networks for reducing collisions. In a protocol based on a strong (n,k)-selector S each time step is associated with some set S_i and only the nodes in S_i are allowed to transmit at this time step. To illustrate more concretely how such selectors are used in our work, consider a node v of in-degree at most k, and suppose that its in-neighbors are initially dormant, staying in the receiving state, and they activate at certain (possibly different) times. Each in-neighbor w of v, after it activates, transmits its message at a time step τ if and only if $w \in S_{\tau \bmod \ell}$. (Observe that, importantly, at any time τ all nodes use the *same* transmission set $S_{\tau \bmod \ell}$.) Let X be the set of in-neighbors of v and suppose that some $w \in X$ activates at time η. Then the definition of strong (n,k)-selectors implies that there will be $\tau \in [\eta, \eta + \ell)$ such that $S_{\tau \bmod \ell} \cap X = \{w\}$. In other words, w will be the only in-neighbor of v that transmits at time step τ. So if v stays dormant until time $\eta + \ell$ then v will receive w's message after at most $\ell = O(k^2 \log n)$ time steps since the activation time of w. (In fact, it is sufficient if v happens to be in the receiving state at time τ.) We stress that this is true independently of the label assignment and of activation times of the nodes other than w.

For all $j = 0, 1, ..., \frac{1}{2}\log n$, by 2^j-Select $= (S_0^j, S_1^j, ..., S_{\ell_j - 1}^j)$ we will denote a strong $(n, 2^j)$-selector of size $\ell_j = O(4^j \log n)$. Without loss of generality we can assume that $\ell_{j+1} = 4\ell_j$ for all $j \leq \frac{1}{2}\log n - 1$. We remark that in Section 5, where we consider transmissions with acknowledgements, it will be desirable to have many (but not necessarily all) of a collection of competing in-neighbors of a node transmit successfully. For this purpose we will there introduce a different type of selectors.

Another basic protocol that is often used is called RoundRobin. In this protocol all nodes transmit cyclically one by one; that is each node w transmits in a step τ if and only if $w = \tau \bmod n$. In RoundRobin there are no collisions, so, in the setting above, node w will successfully transmit its message to v in at most n time steps. Observe that a protocol based on a strong (n,k)-selector can be faster than RoundRobin only when $k = O(\sqrt{n/\log n})$.

Note: To avoid clutter, in the definitions above, as well as later throughout the paper, we omit the notation for rounding and assume that in all formulas representing integer quantities (the number of nodes, steps, etc.) their values are appropriately rounded. This will not affect asymptotic running time estimates.

Simplifying assumptions. To streamline the presentation of our protocols, in the paper we use a relaxed communication model with two additional features:

(MFC) We assume that some number κ of radio *frequency channels*, numbered $0, 1, ..., \kappa - 1$, is available for communication. So a node may receive and transmit up to κ messages at each step, one per channel. There is no interference between channels; that is, a message sent on one channel cannot collide with a message sent on a different channel. For each individual frequency the standard collision rule applies: if two in-neighbors of a node transmit on this frequency channel at the same time step, then a collision occurs and this node does not receive any information on this channel at this step.

(SRT) Further, for each frequency f, a node can receive and transmit at frequency f in a single step. The restriction is that the messages transmitted at all frequencies in any step do not depend on the messages received in this step.

All protocols in this paper are presented in terms of this relaxed model. Below we explain, however, how to convert these protocols to the standard radio network model (as defined earlier), increasing the running time only by factor $O(\kappa)$; that is, a protocol that uses features (MFC) and (SRT), and for which we give an upper bound $O(T)$ on the running time, can be converted into a protocol in the standard model whose running time is $O(\kappa T)$. Since $\kappa = O(\log n)$ in our protocols, their $\tilde{O}(\cdot)$-complexity is not affected.

Simulating multiple frequencies. We first explain how we can convert *any* protocol $\mathcal{A}$ that uses κ frequencies and runs in time $O(T)$ into a protocol $\mathcal{A}'$ that uses only one frequency and runs in time $O(\kappa T)$. This can be done by straightforward time multiplexing. In more detail: $\mathcal{A}'$ organizes all time steps $0, 1, 2, \ldots$ into *rounds*. Each round $r = 0, 1, 2, \ldots$ consists of κ consecutive steps $r\kappa, r\kappa + 1, \ldots, r\kappa + \kappa - 1$. Each step s of $\mathcal{A}$ is simulated by round s of $\mathcal{A}'$. For each frequency f, the message transmitted at frequency f by $\mathcal{A}$ is transmitted by $\mathcal{A}'$ in step $s\kappa + f$, that is the fth step of round s. At the end of round s, $\mathcal{A}'$ will know all messages received in this round, so it will know what messages would $\mathcal{A}$ receive in step s, and therefore it knows the state of $\mathcal{A}$ and can determine the transmissions of $\mathcal{A}$ in the next step.

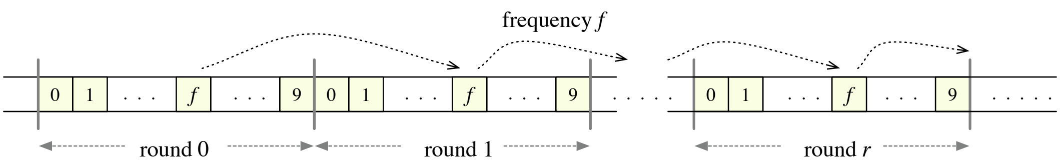

Figure 1 Partition of $\mathcal{A}'$'s time steps into rounds, for $\kappa = 10$ frequencies.

Simulating simultaneous receiving/transmitting. By the argument above, we can assume that we have only one frequency channel. We first give a generic argument that applies to arbitrary protocols. (The construction in this paragraph is not strictly needed for our results, but we include it, as it has some independent interest and provides useful context.) For an arbitrary protocol we claim that we can disallow simultaneous receiving and transmitting at the cost of only adding a logarithmic factor to the running time. To see this, suppose that $\mathcal{B}$ is some transmission protocol where nodes can transmit and listen at the same time. (Recall that the transmission of $\mathcal{B}$ at any step does not depend on the information it receives in the same step.) We use a strong $(n, 2)$-selector 2-SELECT $= (S_i^1)_i$ of size $\ell_1 = O(\log n)$. We replace each step τ of $\mathcal{B}$ by a time segment I_τ of length ℓ_1. For any node w and any $i = 0, 1, \ldots, \ell_1 - 1$, if $w \in S_i^1$ then at the ith step of segment I_τ node w transmits whatever message it would transmit in $\mathcal{B}$ at time τ; otherwise w is in the receiving state. By definition, in this new protocol $\mathcal{B}'$ nodes do not transmit and receive at the same time. Further, for any edge (u, v), let i be such that $S_i^1 \cap \{u, v\} = \{u\}$. If u transmitted successfully to v in step τ of $\mathcal{B}$, then in $\mathcal{B}'$ in the i-th step of segment I_τ u will be in the transmitting state and v will be in the receiving state, guaranteeing that u's message will reach v.

We now claim that for the type of protocols presented in the paper this additional factor of $\log n$ is not necessary; that is, allowing simultaneous reception and transmission does not affect their asymptotic running time at all. Our protocols ArbGather in Section 4 and AcyGatherAck in Section 5 are *selector-based*, namely the computation of each node is divided into time intervals, where in each interval the node transmits either according to RoundRobin or according to some strong selector, in the manner described earlier in this section. In case of RoundRobin, the simultaneous reception and transmission capability is (trivially) not needed. For intervals where a strong selector is used, the argument how this capability can be removed was given in [4]. Roughly, the idea is that whenever a protocol uses a strong (n, k)-selector, this selector can be replaced by a strong $(n, k + 1)$-selector. The size of this $(n, k + 1)$-selector is $O(k^2 \log n)$, so for $k > 1$ it is asymptotically the same as for a strong (n, k)-selector. (And the contribution to our running time bounds from the strong $(n, 1)$-selectors is comparatively negligible.) This guarantees that, during each complete cycle (of length $O(k^2 \log n)$) of this selector, for any node v with k in-neighbors and any v's in-neighbor u there will be a step when v is in the receiving state and u is the only in-neighbor in the transmitting state.

We will make yet another assumption in the paper, this one concerning the initial knowledge that the nodes possess about the digraph. In the standard definition given earlier in this section, the nodes know the size n of the digraph but do not know its topology. Without any loss of generality, we will relax the latter restriction as follows:

(INN) We assume that when the computation starts each node v knows the labels of its in-neighbors, that is the set $N^-(v)$.

The reason that we can assume (INN) is that any protocol in our paper, prior to starting its execution, can execute one cycle of RoundRobin, where each node transmits only its own label. This costs only time $O(n)$, so the asymptotic running time of the protocol is not affected. (In the paper we only consider protocols whose worst-case running time is $\Omega(n)$.)

3 $\tilde{O}(n^{1.5})$-Time Protocol for Acyclic Digraphs

We first consider ad-hoc radio networks whose underlying digraph G is acyclic and has one designated target node t that is reachable from all other nodes in G. We give a deterministic information gathering protocol that gathers all rumors in the target node t in time $O(n^{1.5} \log^3 n)$, independently of the topology of G.

As explained in Section 2, we make Assumptions (MFC), (SRT) and (INN), namely that the protocol has multiple frequency channels available, that on each frequency it can simultaneously receive and transmit messages at each step, and that each node knows its in-neighbors.

The protocol is based on strong selectors 2^j-Select and on RoundRobin, following the principles outlined in Section 2. Thus, whenever we say that a node w *transmits according to* 2^j-Select during some time interval $[\eta, \eta']$, we mean that for any time step $\tau \in [\eta, \eta']$ node w transmits if and only if $w \in S^j_{\tau \bmod \ell_j}$. (Recall that ℓ_j is the size of 2^j-Select.) The concept of *transmitting according to* RoundRobin is defined analogously.

Let $\theta = \frac{1}{2}(\log n - \log\log n) + 2$. In the algorithm below we use a sequence of $\theta + 1$ values $\beta_0, \beta_1, ..., \beta_\theta$, defined as follows: $\beta_0 = 0$, $\beta_j = \sum_{g<j} \ell_g$ for $j = 1, ..., \theta - 1$, and $\beta_\theta = \sum_{g<\theta} \ell_g + n$.

Protocol AcyGather. We start with an overview of the algorithm. The algorithm transmits on θ frequencies numbered $0, 1, ..., \theta - 1$. Each node v uses information received from its in-neighbors to determine its *activation time*, denoted by $\alpha(v)$. Node v will be *active* during its *activity period* $[\alpha(v), \alpha(v)+\beta_\theta)$; before step $\alpha(v)$ it is called *dormant* and after step $\alpha(v)+\beta_\theta-1$ it is called *expired.* Dormant and expired nodes do not transmit; active nodes may or may not transmit at any given step. While active, v will also periodically compute and transmit a value called its *recommended wake-up time*, denoted rws_v. (The out-neighbors of v use these values to determine their own activation times.) Each message transmitted by v will contain the following information: (i) all rumors collected by v, including its own, (ii) the label of v, and (iii) the current value of rws_v.

We are now ready to detail the steps of the algorithm, and in particular to describe how exactly the values of $\alpha(v)$ and rws_v are computed by a node v.

First, we explain how v determines its activation time $\alpha(v)$. If v is a source node (that is, its in-degree is 0), then $\alpha(v)=0$. Otherwise $\alpha(v)$ is determined by the messages received by v, as follows. For each in-neighbor u of v, we denote by $rws^1_{u\to v}$ the *first* rws_u value received by v from u. (This may not be the first rws_u value *transmitted* by u, since earlier transmissions of u might have collided at v.) Node v waits until it receives messages from all its in-neighbors, and, as soon as this happens, if u is the last in-neighbor of v that successfully transmitted to v, then v sets $\alpha(v)=rws^1_{u\to v}$.

Next, we define the transmission sequence of v. The activity period $[\alpha(v), \alpha(v)+\beta_\theta)$ of v is divided into θ *activity stages*, where, for $j=0,1,...,\theta-1$, the jth activity stage consists of the time interval $[\alpha(v)+\beta_j, \alpha(v)+\beta_{j+1})$. (See Figure 2.) During its jth activity stage, for $j\le\theta-2$, node v transmits according to selector 2^j-Select using frequency j. During the $(\theta-1)$th activity stage, the protocol transmits using RoundRobin on frequency $\theta-1$. At all other times v does not transmit. The recommended wake-up time value included in v's messages during its jth activity stage is $rws_v=\alpha(v)+\beta_{j+1}$. (Note that it changes from one activity stage to next.)

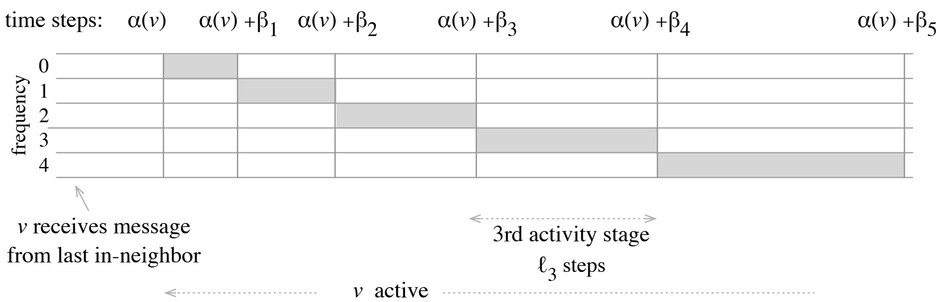

Figure 2 Illustration of activity stages. (The picture is not up to scale. In reality the length of activity stages increases at rate 4.) Shaded regions show frequencies used in different activity stages.

Correctness. We first show that the algorithm is correct, in the sense that each rumor will eventually reach the target node t. This should be intuitively clear, because once a node becomes active, it is guaranteed to successfully transmit its message to its all out-neighbors using the RoundRobin protocol during its last activity stage.

Formally, we can prove correctness by induction. For a given node v, let $\delta(v)$ denote the length of the *longest* directed path from some source node to v. (This path cannot repeat vertices due to our assumption that G is acyclic.) Let $\delta^*=\max_{v\in G}\delta(v)$. The definition of $\delta()$ implies directly the following observation:

▶ Observation 1. The values of $\delta()$ satisfy the following properties:
(i) $\delta(v) = 0$ if and only if v is a source node of G.
(ii) $\delta(v) = \delta^*$ if and only if $v = t$.
(iii) If $u \in N^-(v)$ then $\delta(u) < \delta(v)$.
(iv) If $\delta(v) > 0$ then there is $u \in N^-(v)$ with $\delta(u) = \delta(v) - 1$.

It is convenient to visualize $\delta()$ in terms of partitioning of G into layers $B_0, B_1, ..., B_{\delta^*}$, where B_i denotes the set of nodes with $\delta(v) = i$. This partitioning is illustrated in Figure 3. The key properties are that B_0 consists of the source nodes, $B_{\delta^*} = \{t\}$, and all edges go from lower- to higher-indexed layers. (This representation will be useful in Section 5.)

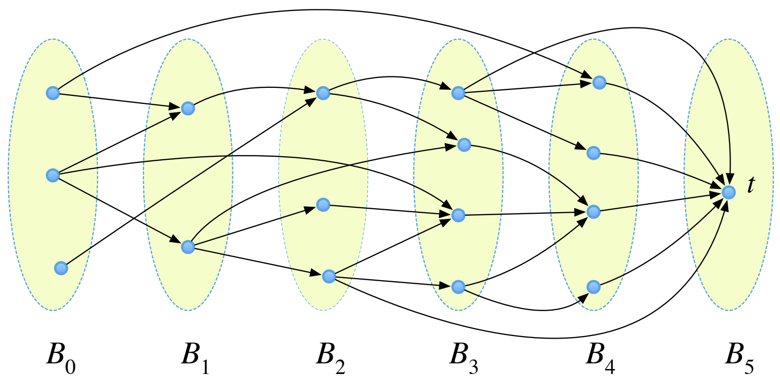

Figure 3 Partiton of G into layers $B_0, B_1, ..., B_{\delta^*}$. In this example $\delta^* = 5$.

We claim that for each $i = 0, 1, ..., \delta^*$, and for each node $v \in B_i$, we have $\alpha(v) \le i\beta_\theta$, and that at time $\alpha(v)$ all rumors from $Anc[v]$ are already in v. This is sufficient, because this guarantees that after $\delta^*\beta_\theta$ steps all rumors will be collected in the target node t. The claim is trivially true for $i = 0$, because all nodes in B_0 are source nodes. For $i > 0$, suppose that the claim holds for indices $i' = 0, 1, ..., i-1$. This means that for each $w \in N^-(v)$, as $w \in \bigcup_{i'=0}^{i-1} B_{i'}$, we have $\alpha(w) \le (i-1)\beta_\theta$ and at time $\alpha(w)$ all rumors from $Anc[w]$ are in w. In its last activity stage $[\alpha(w) + \beta_{\theta-1}, \alpha(w) + \beta_\theta)$ node w will transmit using RoundRobin, so v will receive a message from w no later than at time $\alpha(w) + \beta_\theta - 1$. This messages includes all rumors in $Anc[w]$ and w's current recommended activation time value rws_w, whose maximum value is $\alpha(w) + \beta_\theta \le (i-1)\beta_\theta + \beta_\theta \le i\beta_\theta$. Then the definition of $\alpha(v)$ implies that $\alpha(v) \le i\beta_\theta$. At time $\alpha(v)$ node v will have received messages from all its in-neighbors, so at that time it will collect rumors from $\bigcup_{w \in N^-(v)} Anc[w] \cup \{v\} = Anc[v]$, completing the proof of the inductive step, and the claim.

Running time. Next, we analyze the running time of Protocol AcyGather. Figure 4 provides a snapshot of the computation of Protocol AcyGather that should be helpful in understanding our analysis. It shows three types of nodes: expired, active and dormant. The in-neighbors of dormant nodes can be of any three types, but each dormant node has at least one in-neighbor that is either dormant or active. The in-neighbors of active nodes are either expired or active. All in-neighbors of expired nodes are expired. These properties follow from the algorithm, because the activation times of all non-source nodes v satisfy

$$\max_{u \in N^-(v)} \alpha(u) \; < \; \alpha(v) \; \le \; \max_{u \in N^-(v)} \alpha(u) + \beta_\theta \,,$$

and because the activation periods of all nodes have the same length (which implies that an expiration time of a node is strictly after the expiration times of all its in-neighbors).

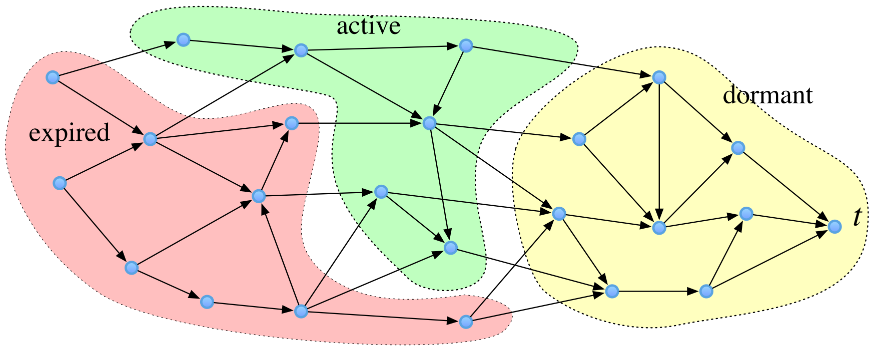

Figure 4 Bird's-eye view of the computation of Protocol AcyGather.

To establish our upper bound, we choose in the graph G a *critical path*

$$P = (v_0, v_1, ..., v_p = t)\,,$$

defined as follows: for each $a = p-1, p-2, ..., 0$, v_a is the in-neighbor of v_{a+1} who was last in succeeding to transmit its message to v_{a+1} (that is, $\alpha(v_{a+1}) = rws^1_{v_a \to v_{a+1}}$), and v_0 is a source node. The argument about correctness of protocol AcyGather, presented above, implies that P is well defined. (Note that, since we define this path in the backward order, the indexing of the nodes v_a can be determined only after we determine the whole path.) The definition of P implies that the overall running time is upper-bounded by the time for the rumor of v_0 to reach t along P.

If at a step τ a node v is in its j-th activity stage (that is, $\tau \in [\alpha(v) + \beta_j, \alpha(v) + \beta_{j+1})$) then we refer to j as *v's stage index in step τ*. We extend this (artificially) to dormant and expired nodes as follows: if v is dormant then its stage index is -1, and if v is expired then its stage index is θ. As time progresses, an active node will move from one activity stage to next, which results in an increment of its stage index. Within any time interval multiple nodes may have their stage indices incremented.

Note that the stage index of each individual node is incremented $\theta + 1 = O(\log n)$ times, so the total number of these increments in the whole computation is $O(n \log n)$. Our estimate on the running time is obtained by "charging" the delay between activation times of consecutive nodes on P to stage-index increments in the graph. The intuition is that if collisions cause a long delay when v_a attempts to send its message to v_{a+1}, then v_{a+1} must have many in-neighbors that transmit in this time period. But then these in-neighbors will have their stage indices incremented during this time, and since the overall number of stage-index increments is $O(n \log n)$, we cannot have too many such long delays.

We now formalize this intuition. Consider some node $v_a \neq t$ on P. (See Figure 5.) Our argument is based on the following key lemma.

▶ **Lemma 1.** *There are the total of $\Omega(\frac{1}{\sqrt{n}\log n}(\alpha(v_{a+1}) - \alpha(v_a)))$ stage-index increments in the time interval $[\alpha(v_a), \alpha(v_{a+1}))$.*

Proof. Suppose that the first transmission of v_a that is successfully received by v_{a+1} occurs during v_a's h-th activity stage.

▶ Claim 1. For $a < p$ and $h < \theta - 1$ we have $\alpha(v_{a+1}) - \alpha(v_a) = O(4^h \log n)$.

This claim follows from the definition of P, as $\alpha(v_{a+1}) = rws^1_{v_a \to v_{a+1}} = \alpha(v_a) + \beta_{h+1}$, and $\beta_{h+1} = \sum_{g<h+1} \ell_g = O(4^h \log n)$.

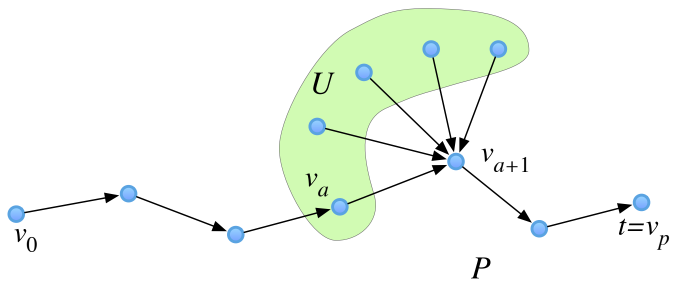

Figure 5 Illustration of the time analysis for acyclic graphs.

We now continue the proof of the lemma. If $h = 0$ then there is at least one stage increment in $[\alpha(v_a), \alpha(v_{a+1}))$ (namely the increment of the stage index of v_a from -1 to 0 at time $\alpha(v_a)$) and $\alpha(v_{a+1}) - \alpha(v_a) = \ell_0 = O(\log n)$, so the lemma holds trivially.

Thus for the rest of the proof we can assume that $1 \le h \le \theta - 1$. By the choice of h, v_a has not succeeded in its $(h-1)$th activity stage $[\alpha(v_a) + \beta_{h-1}, \alpha(v_a) + \beta_h)$. Let U be the set of in-neighbors of v_{a+1} (including v_a) whose $(h-1)$th activity stage overlapped that of v_a.

▶ Claim 2. $|U| > 2^{h-1}$.

To justify Claim 2, we argue by contradiction. Suppose that $|U| \le 2^{h-1}$. During this activity stage v_a transmitted according to 2^{h-1}-Select using only frequency $h-1$. Further, by the definition of the protocol, at each step of this stage the in-neighbors of v_{a+1} with stage index other than $h-1$ did not use frequency $h-1$ for transmissions. So the transmissions from v_a to v_{a+1} in this stage can only conflict with transmissions from $U \setminus \{v_a\}$ to v_{a+1}. The definition of strong selectors and the assumption that $|U| \le 2^{h-1}$ imply that then v_a would have successfully transmitted to v_{a+1} during its $(h-1)$th activity stage, contradicting the definition of h. Thus Claim 2 is indeed true.

▶ Claim 3. The total number of stage-index increments in time interval $[\alpha(v_a), \alpha(v_{a+1}))$ is at least 2^{h-1}.

The proof of Claim 3 is quite simple: The $(h-1)$th activity stage lasts ℓ_{h-1} steps, so for each node in U its $(h-1)$th activity stage ends before time $\alpha(v_a) + \beta_h + \ell_{h-1} < \alpha(v_a) + \beta_{h+1} = \alpha(v_{a+1})$. By the definition of U, it also cannot end before $\alpha(v_a)$. So, each time the $(h-1)$th activity stage of a node in U ends, it contributes 1 to the total number of stage-index increments in time interval $[\alpha(v_a), \alpha(v_{a+1}))$. This implies Claim 3.

Now, to complete the proof of the lemma we have two cases. If $h < \theta - 1$ then, by Claim 3, the number of stage-index increments in time interval $[\alpha(v_a), \alpha(v_{a+1}))$ is at least

$$2^{h-1} = \tfrac{1}{2} \cdot 2^{-h} \cdot 4^h = \Omega\big(\tfrac{1}{\sqrt{n}\log n} (\alpha(v_{a+1}) - \alpha(v_a)) \big),$$

because $2^{-h} \ge 2^{-\frac{1}{2}\log n} = 1/\sqrt{n}$ (as $h \le \frac{1}{2}\log n$) and $4^h = \Omega(\, (\alpha(v_{a+1}) - \alpha(v_a))/\log n\,)$, by Claim 1.

On the other hand, if $h = \theta - 1$ then $\alpha(v_{a+1}) - \alpha(v_a) = n$. From the definition of θ we have $h \ge \frac{1}{2}\log n - \log\log n + 1$ so, applying Claim 3, we obtain that the number of stage-index increments in time interval $[\alpha(v_a), \alpha(v_{a+1}))$ is at least $2^{h-1} \ge 2^{\frac{1}{2}\log n - \log\log n} = \sqrt{n}/\log n = \frac{1}{\sqrt{n}\log n}(\alpha(v_{a+1}) - \alpha(v_a))$. ◀

We now show how we can use Lemma 1 to establish an $O(n^{1.5}\log^3 n)$ upper bound on the running time of Protocol AcyGather.

▶ **Theorem 2.** *Let G be an acyclic directed graph with n vertices and a designated target node reachable from all other nodes. Algorithm* AcyGather *completes information gathering on G in time $O(n^{1.5} \log^3 n)$.*

Proof. Let T be the running time of Protocol AcyGather in our relaxed model of communication, namely with assumptions (MFC), (SRT), and (INN). Since $\alpha(v_0) = 0$ and $T \leq \alpha(v_p)$, we can bound this running time as $T \leq \sum_{a=0}^{p-1}(\alpha(v_{a+1}) - \alpha(v_a))$. Then Lemma 1 implies that the total number of stage index increments during the computation is $\Omega(T/\sqrt{n}\log n)$. Since this number is also $O(n \log n)$, it gives us that $T = O(n^{1.5} \log^2 n)$.

The number of frequencies is $\kappa = O(\log n)$. Thus, as explained in Section 2, we can eliminate all three assumptions (MFC), (SRT), and (INN), increasing the running time only by a factor $O(\log n)$. Such modified Protocol AcyGather will run in time $O(n^{1.5} \log^3 n)$ in the standard model of ad-hoc radio networks. ◀

Note: With more careful analysis the logarithmic factors in Theorem 2 can be slightly improved (at least to $O(n^{1.5} \log^{2.5} n)$). We leave this as an exercise for the reader.

4 $\tilde{O}(n^{1.5})$-Time Protocol for Arbitrary Digraphs

We now extend our information gathering protocol AcyGather from Section 3 to arbitrary digraphs, retaining running time $O(n^{1.5} \log^3 n)$. Throughout this section G will denote an n-vertex digraph with a designated target node t that is reachable from all other nodes in G.

The main obstacle we need to overcome is that protocol AcyGather critically depends on G being acyclic. For instance, in that protocol each node waits until it receives messages from all its in-neighbors. If cycles are present in G, this leads to a deadlock, where each node in a cycle waits for its predecessor. On the other hand, the known gossiping protocols [6, 29, 18] do not work correctly if the graph is not strongly connected, because they rely on broadcasting to periodically flush out some rumors from the system, and on leader election to synchronize computation.

The idea behind our solution is to integrate protocol AcyGather with the gossiping protocol from [18], using AcyGather to transmit information between different sc-components of G and using gossiping to disseminate information within sc-components. The idea is natural but it faces some technical challenges. One challenge is that the sc-components are actually not known. In fact, a node v doesn't even know the size of $C(v)$, but it needs to provide this size to the gossiping protocol. To get around this issue, the algorithm runs in parallel $\log n$ copies of a gossiping protocol for sizes that are powers of 2. Another challenge is that transmissions from outside of an sc-component may interfere with the execution of the gossiping protocol in this sc-component. This will be addressed by executing the gossiping algorithm repeatedly until it succeeds. In order to verify whether the gossiping succeeded, its nodes will distribute and collect additional information, not just rumors.

Protocol SccGossip for gossiping. We will refer to the gossiping algorithm from [18] as SccGossip. The following property of SccGossip is crucial for our algorithm:

(scc) If the input digraph is strongly connected and has at most $\bar{n}$ vertices, with the node labels from the set $[N] = \{0, 1, ..., N-1\}$, then algorithm SccGossip completes gossiping in time $O(\bar{n}^{4/3} \log^{10/3} N)$.

The bound on the running time of SccGossip given in [18] (see their Theorem 2) assumes that N is bounded polynomially in $\bar{n}$ and it does not explicitly separate the dependence on $\bar{n}$ and N. In Appendix A we explain how the bound in Property (scc) follows from the analysis in [18].

As explained earlier, one idea of our algorithm is to execute SccGossip on its sc-components. The details of this will be provided shortly. For now, we only make an observation that captures one basic principle of this process. Let A be an sc-component of size n_A and let $j = \lceil \log n_A \rceil$, so that $2^{j-1} < n_A \le 2^j$. Let $\textsc{SccGossip}_j$ denote SccGossip specialized for strongly connected digraphs of size $\bar{n} = 2^j$ and label set $[N] = [n]$, and let $T_{\textrm{scc}}(j)$ be the running time of $\textsc{SccGossip}_j$ on such digraphs. Suppose also that all nodes in $Anc(A)$ are idle and that the nodes in A execute $\textsc{SccGossip}_j$, all starting at the same time and ignoring any information collected so far in A. Since the nodes in $Anc(A)$ are idle, there will be no interference from outside A. Then the execution of $\textsc{SccGossip}_j$ on A will be identical to its execution on the sub-digraph of G induced by A; that is, as if the rest of G did not exist. Therefore, using Property (scc), this execution of $\textsc{SccGossip}_j$ will complete correctly in time $T_{\textrm{scc}}(j) = O(n_A^{4/3} \log^{10/3} n)$. In our analysis we will use the estimate $T_{\textrm{scc}}(j) = \tilde{O}(n_A^{4/3})$, hiding the polylogarithmic factor, as the degree of this factor will not affect the overall asymptotic running time of our algorithm.

Algorithm ArbGather. Our protocol can be thought of as running two parallel subroutines, the SCC-subroutine and the ACY-subroutine, that use two disjoint sets of frequencies. There will be θ *ACY-frequencies* indexed $0, 1, ..., \theta - 1$, where $\theta = \frac{1}{2}(\log n - \log\log n) + 2$, as in Section 3. These will be used by the ACY-subroutine to simulate protocol AcyGather. We will also have $\theta' = \log n$ *SCC-frequencies* indexed $0, 1, ..., \theta' - 1$, used by the SCC-subroutine to simulate protocol SccGossip. Due to using different frequencies, there will be no signal interference between these two subroutines. In the SCC-subroutine, each SCC-frequency j will be used to simulate $\textsc{SccGossip}_j$. Thus these different instances of $\textsc{SccGossip}_j$ will also do not interfere with each other.

Before providing the detailed descriptions of these subroutines, we give an overview of the algorithm, expanding on the intuitions described earlier in this section. A good way to visualize the computation of Algorithm ArbGather is to think of the ACY-subroutine as a "master process" that transmits rumors between sc-components (that form an acyclic graph), while the SCC-subroutine is run as a "background" iterative process in all nodes. Consider some sc-component A. The nodes in A will repeatedly run in parallel θ' copies of protocol SccGossip, one for each SCC-frequency, all attempting to determine A and to collect rumors from $Anc[A]$. While a node v in A executes these protocols SccGossip, its in-neighbors in $Anc(A)$ (that is, outside of A) may be transmitting to v. Messages sent from $Anc(A)$ to v on ACY-frequencies contribute to progress, as they contain rumors from $Anc(A)$, that v needs to collect, and do not interfere with v's executions of SccGossip. Messages sent from $Anc(A)$ to v on SCC-frequencies are problematic, because they can collide with transmissions of SccGossip's coming from in-neighbors of v in A. (And also, they do not provide useful information.)

We focus on one particular value of j, namely $j = \lceil \log n_A \rceil$, as for this j $\textsc{SccGossip}_j$ is most likely to succeed quickly on A. Some number of repetitions of $\textsc{SccGossip}_j$ in A may fail, either because of collisions with transmissions on SCC-frequencies from $Anc(A)$, or because of some nodes in A not being yet ready to participate in the gossiping in A, if they have not yet received messages from all of their in-neighbors in $Anc(A)$. Eventually though, all the nodes in $Anc(A)$ will complete their own executions of $\textsc{SccGossip}_j$, stop the simulation of SccGossip altogether, and switch to the ACY-subroutine. As a result, these nodes will no longer interfere with the executions of $\textsc{SccGossip}_j$ in A, and their execution of the ACY-subroutine will successfully transmit their messages to the nodes in A. At this point a repetition of $\textsc{SccGossip}_j$ in A will succeed, in the sense that each node $v \in A$ will determine A, it will receive all rumors from $Anc[A]$, and it will also learn that *other nodes* in A have successfully completed this process. This will happen for all nodes in A during the same repetition of $\textsc{SccGossip}_j$, allowing the algorithm to "synchronize" the computation of all nodes in A so that they all can simultaneously stop executing the SCC-subroutine and switch to executing the ACY-subroutine.

Next, we give formal descriptions of both subroutines.

The SCC-subroutine. For each $j = 0, 1, ..., \theta' - 1$, SCC-frequency j will be used to simulate protocol $\textsc{SccGossip}_j$. For $s = 0, 1, ...$ let $\psi_{j,s} = 2sT_{\text{SCC}}(j)$. For each SCC-frequency j, the computation of v on frequency j is partitioned into *j-frames*, where the s-th j-frame, for any $s \geq 0$, is $[\psi_{j,s}, \psi_{j,s+1})$ — a time interval sufficient for two complete simulations (described below) of $\textsc{SccGossip}_j$ on a digraph with at most 2^j nodes. For each j, these simulations start at time 0 and stop as soon as v determines that for at least one SCC-frequency j' the simulation of $\textsc{SccGossip}_{j'}$ successfully completed in $C(v)$. (More precisely, they stop at time $\alpha_{\text{ACY}}(v)$ that will be defined shortly.)

Consider a j-frame r, for some $r \geq 0$. In this j-frame v executes two runs of $\textsc{SccGossip}_j$. The first one is called the *exploration run* and it is used to distribute node labels; that is, any node v uses its own label v as the "rumor" for the purpose of gossiping. The second run of $\textsc{SccGossip}_j$ is called the *dissemination run*. In this run v's "rumor" is the 5-tuple

$$[\, v\,,\, \tilde{C}(v)\,,\, N^-(v)\,,\, \tilde{N}^-_{\text{ACY}}(v)\,,\, R(v)\,]\,,$$

where

- $\tilde{C}(v)$ is the set of labels received by v during the exploration run of j-frame r, including v itself.
- $\tilde{N}^-_{\text{ACY}}(v) \subseteq N^-(v)$ is the set of in-neighbors of v that have successfully transmitted a message to v on some ACY-frequency before time $\psi_{j,r}$ (the beginning of r-th j-frame), and
- $R(v)$ is the set of all (original) rumors received on ACY-frequencies before time $\psi_{j,r}$, plus the rumor of v.

Let $\tilde{C}'(v)$ be the set of node labels received by v in this dissemination run of $\textsc{SccGossip}_j$, including v itself. Then, immediately after the dissemination run, v performs three tests:

Test 1*:* Is it true that $\tilde{C}(v) = \tilde{C}'(v)$?
Test 2*:* Is it true that $\tilde{C}(v) = \tilde{C}(u)$ for all $u \in \tilde{C}'(v)$?
Test 3*:* Is it true that $N^-(u) \setminus \tilde{N}^-_{\text{ACY}}(u) \subseteq \tilde{C}(v)$ for all $u \in \tilde{C}'(v)$?

If at least one of these tests fails, v continues the execution of the SCC-subroutine on frequency j, proceeding to j-frame $r + 1$. If all tests pass, v aborts its SCC-subroutine altogether (thus aborting the simulations of $\textsc{SccGossip}_{j'}$ for all frequencies j') and switches to the ACY-subroutine, with its set of collected rumors being $\bigcup_{u \in \tilde{C}(v)} R(u)$. Let $\alpha_{\text{ACY}}(v) = \psi_{j,r+1}$ denote this time step. We call it the *ACY-activation time* of v.

The ACY-subroutine. The ACY-activation time $\alpha_{\text{ACY}}(v)$ of v, defined above, plays the role of v's activation time in protocol $\textsc{AcyGather}$. In this subroutine v will transmit at the ACY-frequencies and it simply executes $\textsc{AcyGather}$ in its *ACY-activity* period

$[\alpha_{\text{ACY}}(v), \alpha_{\text{ACY}}(v) + \beta_\theta)$. The activity stages and the transmissions of each node are defined in exactly the same way as in protocol AcyGather (except that we use $\alpha_{\text{ACY}}(v)$ instead of $\alpha(v)$). At each step any node is in one of three possible states: ACY-dormant, ACY-active, or ACY-expired. These concepts are natural adaptations of the corresponding concepts for protocol AcyGather. (The only difference is that now ACY-dormant nodes are not truly "dormant", as they execute the SCC-subroutine.)

Correctness. To show correctness, we need to show that all rumors will eventually reach t, the target node of G. The basic structure of the proof is similar to the proof for Protocol AcyGather in Section 3, namely we proceed by induction. The difference is that now in one step of the inductive argument we analyze progress in a whole sc-component, rather than a single vertex; that is we show that all nodes in any given sc-component A will collect all rumors from $Anc[A]$. This can be captured formally by considering an auxiliary acyclic digraph $\mathsf{SccDag}(G)$ that represents the structure of sc-components of G. The vertices of $\mathsf{SccDag}(G)$ are the sc-components of G, and for any two sc-components A and A', we include edge (A, A') in $\mathsf{SccDag}(G)$ iff there are vertices $u \in A$ and $v \in A'$ with edge $(u, v) \in E$. The induction is then with respect to the values of $\delta(A)$, the maximum path length from a source sc-component to A in $\mathsf{SccDag}(G)$. In the argument below we focus on analyzing the algorithm's progress within one sc-component A (see Lemma 3 below), as the details of the argument showing that the nodes in A will receive the rumors from $Anc(A)$ are essentially the same as in Section 3, that is they are based on the usage of RoundRobin in Protocol AcyGather.

With each node v of G we will associate a time step called the *SCC-ready time*, denoted $\rho_{\text{SCC}}(v)$. Its definition is identical to the activation time in AcyGather: If $N^-_{\text{ACY}}(v) = \emptyset$ then $\rho_{\text{SCC}}(v) = 0$. Otherwise, $\rho_{\text{SCC}}(v)$ is the last received value $rws^1_{u \to v}$ for $u \in N^-_{\text{ACY}}(v)$, where $rws^1_{u \to v}$ denotes the first rws_u value received by v from u. As explained earlier, these rws_u values will be received on ACY-frequencies. (We stress that the SCC-ready times are used only for the analysis. In fact, the value of $\rho_{\text{SCC}}(v)$ depends on $C(v)$, so it cannot even be computed before v determines $C(v)$.) We extend this definition naturally to sc-components; if A is an sc-component, we let $\rho_{\text{SCC}}(A) = \max_{u \in A} \rho_{\text{SCC}}(u)$.

Any node v that ever becomes ACY-active is guaranteed to successfully transmit during the ACY-subroutine, because this subroutine involves a round of RoundRobin. Thus the key difficulty in proving correctness is to show that, for any sc-component A, each node $v \in A$ will correctly complete the SCC-subroutine, meaning that it will collect all rumors from $Anc[v]$ before time $\alpha_{\text{ACY}}(v)$, when it switches to executing the ACY-subroutine. We make this more precise in the lemma below.

▶ **Lemma 3.** *Let A be an sc-component of G. Let n_A be its size and $j = \lceil \log n_A \rceil$. Then*
(i) $\rho_{\text{SCC}}(A)$ is well-defined (that is, finite).
(ii) All nodes in A complete subroutine SccGossip *at the same time. In other words, all values $\alpha_{\text{ACY}}(v)$, for $v \in A$, are equal. (Below we use notation $\alpha_{\text{ACY}}(A)$ for this common value of $\alpha_{\text{ACY}}(v)$ for $v \in A$.)*
(iii) At time $\alpha_{\text{ACY}}(A)$ each node in A has all rumors from $Anc[A]$.
(iv) $\alpha_{\text{ACY}}(A) \le \rho_{\text{SCC}}(A) + 4 \cdot T_{\text{SCC}}(j)$.

Proof. As indicated earlier, the proof of Lemma 3 is by induction with respect to $\delta(A)$. That is, assuming that all nodes in $Anc(A)$ satisfy the claims (i)-(iv) of the lemma, we argue that they also hold for A.

We start with part (i). By the inductive assumption, at some time step all nodes in $Anc(A)$ will complete their execution of the SCC-subroutine and start the ACY-subroutine.

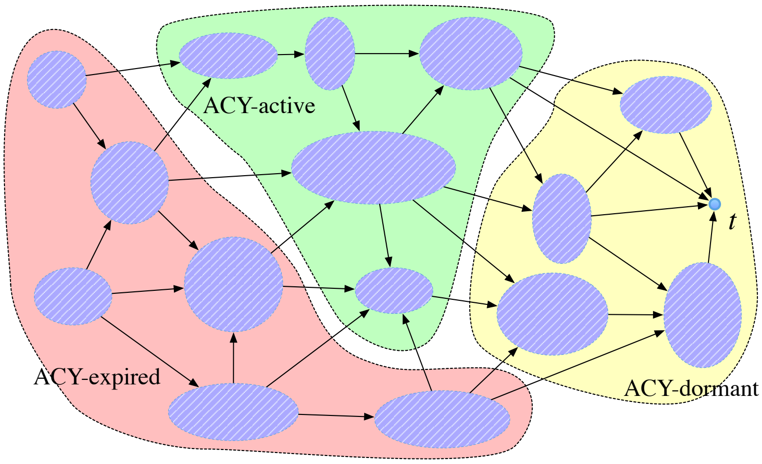

Figure 6 Bird's-eye view of the snapshot of a computation of Protocol ArbGather. The striped ovals represent sc-components of G or, equivalently, vertices of $\mathsf{SccDag}(G)$. The arrows are the edges of $\mathsf{SccDag}(G)$, with two sc-components connected by an edge if they contain vertices connected by an edge of G. Note that in each sc-component all nodes are of the same type. This follows from Lemma 3 and from all ACY-activity intervals having the same length.

Since the ACY-subroutine involves RoundRobin, eventually all nodes in A will receive messages of Protocol AcyGather from their in-neighbors outside A. (For details, see the argument in Section 3). This implies that $\rho_{\text{SCC}}(v)$ is well-defined for each $v \in A$, implying part (i).

In the proof of the remaining parts (ii)-(iv), the main idea is that once the nodes in $Anc(A)$ stop executing their SCC-subroutine, and thus do not interfere with A, the computation of the SCC-subroutine in frequency j is guaranteed to succeed in A in time $T_{\text{SCC}}(j)$. One minor but complicating caveat is that this may not actually be true as stated, because a node in A can conceivably get "lucky" by having passed Tests 1-3 on some other frequency j', in which case this node will terminate its execution of the whole SCC-subroutine in the middle of the current j-frame. In this case we need to argue that then in fact *all* nodes in A will succeed on frequency j' at that time.

To formalize this idea, we first show that at least one $v \in A$ will terminate its SCC-subroutine and become ACY-active at time step $\alpha_{\text{ACY}}(v) \leq \rho_{\text{SCC}}(A) + 4 \cdot T_{\text{SCC}}(j)$. Let r be the index such that $\psi_{j,r-1} \leq \rho_{\text{SCC}}(A) < \psi_{j,r}$. Since $\psi_{j,r+1} = \psi_{j,r-1} + 4 \cdot T_{\text{SCC}}(j) \leq \rho_{\text{SCC}}(A) + 4 \cdot T_{\text{SCC}}(j)$, it is sufficient to show that at least one $v \in A$ will have $\alpha_{\text{ACY}}(v) \leq \psi_{j,r+1}$. If some node $v \in A$ passes Tests 1-3 on some frequency $j' \neq j$ before time $\psi_{j,r+1}$ then $\alpha_{\text{ACY}}(v) \leq \psi_{j,r+1}$, and we are done. So suppose that this is not the case, that is all nodes in A remain ACY-dormant and execute SccGossip$_j$ until time $\psi_{j,r+1}$. By the choice of r, at times $\tau \geq \psi_{j,r}$ the nodes in $Anc(A)$ are either ACY-active or ACY-expired, so they do not transmit at SCC-frequencies anymore. Then the two runs of SccGossip$_j$ in j-frame r will not experience any interference from outside A. Therefore, by the paragraph before the description of the algorithm, both these runs will complete correctly, namely for each node $v \in A$ all Tests 1-3 will pass and v will become ACY-active at time $\alpha_{\text{ACY}}(v) = \psi_{j,r+1}$, as needed. (Note that in this case this holds in fact for *all* nodes in A.)

So now we know that at least one $v \in A$ became ACY-active at time $\psi_{j,r+1}$ or earlier. Let v be the *first* node in A that became ACY-active, and let j' be the frequency for which v passed Tests 1-3, say after executing two runs of SccGossip$_{j'}$ in some j'-frame s. Thus

$\alpha_{\text{ACY}}(v) = \psi_{j',s+1} \le \psi_{j,r+1}$. We need to show that then all other nodes in A pass these tests in the same j'-frame s.

To this end, we claim first that at time $\psi_{j',s+1}$ we have $\tilde{C}(v) = A$. Indeed, Tests 1-2 imply that each $u \in \tilde{C}(v)$ and v are reachable from each other, and therefore $\tilde{C}(v) \subseteq A$. And if we had $A \setminus \tilde{C}(v) \neq \emptyset$ then, by strong connectivity of A, there would be a vertex $z \in A \setminus \tilde{C}(v)$ with an out-neighbor u in $\tilde{C}(v)$. By the choice of v, this z is still ACY-dormant, so it has not yet done any transmissions on ACY-frequencies. This means that $z \in N^-(u) \setminus \tilde{N}^-_{\text{ACY}}(u)$ and $z \notin \tilde{C}(u)$, so Test 3 would fail for this u, and v could not terminate its execution of the SCC-subroutine — contradiction. We thus conclude that $\tilde{C}(v) = A$, as claimed.

Next, we claim that the two runs of $\textsc{SccGossip}_{j'}$ in A in j'-frame s did not experience any interference from nodes outside A. Consider any $u \in A$. By the choice of v, the nodes in A are not ACY-active before j'-frame s, and therefore $\tilde{N}^-_{\text{ACY}}(u) \cap A = \emptyset$. Also, since u passed v's Test 3, we have $N^-(u) \setminus \tilde{N}^-_{\text{ACY}}(u) \subseteq A$. This implies that $\tilde{N}^-_{\text{ACY}}(u) = N^-_{\text{ACY}}(u)$. Therefore all nodes $z \in N^-_{\text{ACY}}(u)$, since they transmitted on ACY-frequencies before j'-frame s, must be either ACY-active or ACY-expired throughout j'-frame s, so they do not transmit at SCC-frequencies. This shows that there is no interference from outside of A during j'-frame s, as claimed.

With no outside interference, both the exploration and dissemination runs of $\textsc{SccGossip}_{j'}$ in A will complete successfully for *all* nodes $v \in A$. This implies (ii) and (iii). As for (iv), it now follows from the bounds we established earlier: $\alpha_{\text{ACY}}(A) = \alpha_{\text{ACY}}(v) \le \psi_{j,r+1} \le \rho_{\text{SCC}}(A) + 4 \cdot T_{\text{SCC}}(j)$. ◀

Running time. Next, we estimate the running time. The argument follows the reasoning in Section 3, but now we need to account for the contribution of the SCC-subroutine. To this end, we apply Lemma 3, which gives us that for an sc-component A of size n_A and for $j = \lceil \log n_A \rceil$ we have $\alpha_{\text{ACY}}(A) - \rho_{\text{SCC}}(A) \le 4 \cdot T_{\text{SCC}}(j) = \tilde{O}(n_A^{4/3})$. The quantity $\alpha_{\text{ACY}}(A) - \rho_{\text{SCC}}(A)$ represents the contribution of A to slowing down the ACY-subroutine. Accumulated over all sc-components, this slowdown works out to be $\tilde{O}(n^{4/3})$. On the other hand, the ACY-subroutine itself, which simply executes $\textsc{AcyGather}$, contributes the total of $O(n^{1.5} \log^3 n)$ to the running time. Putting it all together, we obtain the $O(n^{1.5} \log^3 n)$ upper bound on the running time of Algorithm $\textsc{ArbGather}$.

To make this argument more precise, we extend the definition of a critical path that was introduced in Section 3. In this section, the *critical path* (see Figure 7) will be defined as a sequence of nodes $P = (v_0, w_0, v_1, w_1, ..., v_p, w_p = t)$ determined as follows:

- For any $a = p, p-1, ..., 0$, suppose that w_a has already been defined, and let $C_a = C(w_a)$. If $Anc(C_a) \neq \emptyset$, then let $v_a \in C_a$ be the node for which $\rho_{\text{SCC}}(v_a) = \rho_{\text{SCC}}(C_a)$. In other words, v_a is the node in C_a for which $\rho_{\text{SCC}}(v_a)$ is maximum. (It could happen that $v_a = w_a$.) On the other hand, if $Anc(C_a) = \emptyset$ (that is, C_a is a source sc-component), then $a = 0$ and $v_0 \in C_0$ is arbitrary; for example we can take $v_0 = w_0$.
- For any $a = p-1, p-2, ..., 0$, suppose that v_{a+1} has already been defined. Then w_a is the node in $N^-_{\text{ACY}}(v_{a+1})$ for which $\rho_{\text{SCC}}(v_{a+1}) = rws^1_{w_a \to v_{a+1}}$. In other words, w_a is the in-neighbor of v_{a+1} outside of C_{a+1} whose message was received last by v_{a+1}.

Denote by T the running time of protocol $\textsc{ArbGather}$ on G. For $a = 0, 1, ..., p$, let also $n_a = |C_a|$ and $j_a = \lceil \log n_a \rceil$. By Lemma 3(iii), at time $\alpha_{\text{ACY}}(C_p)$ all rumors from G will be collected by t; thus $T \le \alpha_{\text{ACY}}(C_p)$. By definition, $\rho_{\text{SCC}}(C_0) = 0$. So we can bound T as

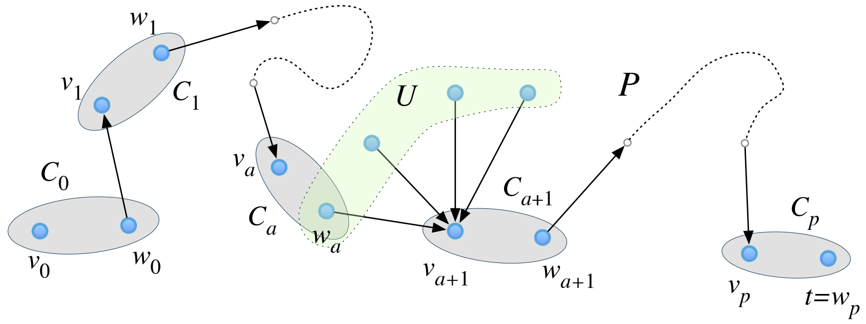

Figure 7 Illustration of the time analysis for arbitrary digraphs.

follows

$$
\begin{aligned}
T \;&\le\; \alpha_{\text{ACY}}(C_p) - \rho_{\text{SCC}}(C_0)\\
&= \sum_{a=0}^{p} [\, \alpha_{\text{ACY}}(C_a) - \rho_{\text{SCC}}(C_a)\,] + \sum_{a=0}^{p-1} [\, \rho_{\text{SCC}}(C_{a+1}) - \alpha_{\text{ACY}}(C_a)\,]\,.
\end{aligned}
$$

We estimate the two sums separately. From Lemma 3(iv) we have $\alpha_{\text{ACY}}(C_a) \le \rho_{\text{SCC}}(C_a) + 4 \cdot T_{\text{SCC}}(j_a)$, so the first sum is

$$
\begin{aligned}
\sum_{a=0}^{p} [\, \alpha_{\text{ACY}}(C_a) - \rho_{\text{SCC}}(C_a)\,] \;&\le\; 4 \cdot \sum_{a=0}^{p} T_{\text{SCC}}(j_a)\\
&=\; 4 \cdot \sum_{a=0}^{p} \tilde{O}(n_a^{4/3}) \;=\; \tilde{O}(n^{4/3}),
\end{aligned}
$$

because $\sum_{a=0}^{p} n_a \le n$. To estimate the second sum, by the definition of v_{a+1} and w_a and by Lemma 3(ii) we have

$$
\rho_{\text{SCC}}(C_{a+1}) - \alpha_{\text{ACY}}(C_a) \;=\; \rho_{\text{SCC}}(v_{a+1}) - \alpha_{\text{ACY}}(w_a) \;=\; rws^1_{w_a \to v_{a+1}} - \alpha_{\text{ACY}}(w_a)\,.
$$

We can now apply the charging argument identical to that in Section 3, namely we charge the delay $rws^1_{w_a \to v_{a+1}} - \alpha_{\text{ACY}}(w_a)$ to stage index increments. (Specifically, these will be charged to the stage-index increments in the set U of in-neighbors of v_{a+1}, defined analogously as in the proof of Lemma 1.) This will give us a bound of $O(n^{1.5}\log^3 n)$ on the second sum. We thus obtain the main result of this paper:

▶ **Theorem 4.** *Let G be an arbitrary digraph with n vertices and a designated target node reachable from all other nodes. Algorithm* ArbGather *completes information gathering in G in time $O(n^{1.5}\log^3 n)$.*

5 $\tilde{O}(n)$-Time Protocol With Acknowledgements for Acyclic Graphs

We now consider the problem of gathering in acyclic graphs with a weak form of acknowledgment of transmission success. In this model, following each transmission from a node v, v receives a single bit of information indicating whether at least one node successfully received that transmission. Thus v does not learn which specific node, or how many nodes in total, received its transmission. Our main goal in this section is to show that this single bit is enough to allow for gathering to be performed in time $O(n \log^2 n)$ on acyclic graphs with n vertices. The key idea here will be that nodes which have successfully transmitted can at least temporarily stop transmitting, making it easier for other nodes to succeed. In order for this to work, though, we need to guarantee that successful transmissions are occurring at a reasonable rate. The following combinatorial object will be our main tool for this.

We say that a collection $(S_0, S_1, \ldots, S_{b-1})$ of label sets forms a (n,k)-*half-selector* if for every $X \subseteq [n]$ with $|X| \le k$ there are at least $|X|/2$ choices of $x \in X$ for which there is an index i with $S_i \cap X = \{x\}$. (This is in contrast to strong selectors where we want this property to hold for *every* choice of x). It is a consequence of Lemma 1 in [6] that for every n and $k \le n$ there exists an (n,k)-half-selector of size $O(k \log n)$.

For all $j = 0, 1, \ldots, \log n$, by 2^j-HalfSelect $= (S^j_0, S^j_1, \ldots, S^j_{b_j-1})$ we will denote an $(n, 2^j)$-half-selector of size $b_j = O(2^j \log n)$. Without any loss of generality we can also assume that $b_{j+1} = 2b_j$ for all $j \le \log n - 1$, implying that $b_j = \gamma 2^j \log n$ for some absolute constant γ.

As in the previous sections, our algorithm is presented in the relaxed radio-network model that satisfies Assumptions (MFC), (SRT) and (INN) from Section 2. The algorithm will use $\kappa = \log n + 2$ frequencies. The intuition here is that for $0 \le j \le \kappa - 2$ frequency j will be used to handle potential interferences involving at most 2^j vertices.

Algorithm AcyGatherAck. At any given time step, a node can be either *dormant* or *active*. Initially the source nodes (with no in-neighbors) will be active and the remaining nodes will be dormant. Any active node transmits according to 2^j-HalfSelect on each frequency $j = 0, 1, ..., \kappa - 2$, and according to RoundRobin on frequency $\kappa - 1$. An active node which receives an acknowledgement of a successful transmission moves to the dormant state, and a dormant node which receives a transmission becomes active.

We remind the reader that when we write that "an active node transmits according to 2^j-HalfSelect", we mean that at a step τ this node, say v, will transmit if and only if $v \in S^j_{\tau \bmod b_j}$. (Thus, on any given frequency, at any given time step all nodes use the same transmission set to determine if they are supposed to transmit or not.) The meaning of transmitting according to RoundRobin is analogous.

Observe that, unlike in the previous algorithms, it is now possible for a node to become active multiple times during the process as it continually receives new messages. Also, the target node, once it receives the first message, remains in the active state forever.

Analysis. We now prove correctness of Algorithm AcyGatherAck and establish an upper bound of $O(n \log n)$ on its running time. To this end, we show that after at most $O(n \log n)$ steps each rumor will reach the target node t.

Our proof uses again the layered structure of G introduced in Section 3. Recall that, for a node v, $\delta(v)$ denotes the length of the *longest* directed path from some source node to v, and that $\delta^* = \max_{v \in G} \delta(v) = \delta(t)$. For $i = 0, 1, ..., \delta^*$, B_i denotes the set of nodes with $\delta(v) = i$. (See Figure 3 for illustration.)

Let $\tau_i = 4\gamma \sum_{p<i} |B_p| \log n$ for all $i = 0, 1, ..., \delta^*$. (In particular, $\tau_0 = 0$.) Our argument is based on the lemma below:

▶ **Lemma 5.** *The following two properties hold for every $i = 0, 1, ..., \delta^*$:*
(i) All nodes in $\bigcup_{p<i} B_p$ remain dormant at all times after τ_i (inclusive).
(ii) At time τ_i each rumor is in some active node in $\bigcup_{p \ge i} B_p$.

Proof. We establish Lemma 5 inductively. Both parts (i) and (ii) of the claim hold vacuously for $i = 0$. Now we assume that parts (i) and (ii) hold for some i and we consider the computation of the nodes in $\bigcup_{p\geq i} B_i$, beginning at time τ_i. The proof involves three claims below that capture fundamental properties of Algorithm AcyGatherAck.

▶ Claim 4. If some rumor is in an active node in $\bigcup_{p\geq i} B_p$ at some time step τ, then it will be in some active node in $\bigcup_{p\geq i} B_p$ at any time step $\tau' \geq \tau$.

To justify Claim 4 note that if some rumor is in an active node $w \in \bigcup_{p\geq i} B_p$ then w remains active until it successfully transmits its message, which includes this rumor, to at least one of its out-neighbors. And once an out-neighbor of w, say u, receives this message from w, it gets immediately activated, if it's not already active. Since $u \in \bigcup_{p\geq i} B_p$ as well, Claim 4 follows.

▶ Claim 5. Let A be the set of nodes in B_i that are active at time τ_i. Then (i) Any node in $B_i \setminus A$ remains dormant forever, and (ii) any node in A becomes permanently dormant right after its first successful transmission.

The proof of Claim 5 is quite simple: Each node in A that successfully transmits is immediately made dormant by the algorithm. The nodes in B_i will not receive any rumors after time τ_i since, by the inductive hypothesis (i) and Observation 1, none of their in-neighbors will be active after time τ_i. So any node in B_i that is dormant at some time $\tau \geq \tau_i$ will remain dormant forever.

▶ Claim 6. Each node in A will succeed in transmitting its message and become dormant before time τ_{i+1}.

To prove Claim 6, choose j such that $2^{j-1} < |B_i| \leq 2^j$. Trivially, $|A| \leq |B_i| \leq 2^j$. Since the algorithm runs 2^j-HalfSelect on frequency j, at least $|A|/2$ nodes in A will have a time step in the interval $[\tau_i, \tau_i + b_j)$ when they will successfully transmit, at which point they become permanently dormant, by Claim 5. Thus, if A' is the set of nodes in B_i that are active at time $\tau_i + b_j$, then $|A'| \leq |A|/2 \leq 2^{j-1}$. Next, we look at time interval $[\tau_i + b_j, \tau_i + b_j + b_{j-1})$. Since the algorithm runs 2^{j-1}-HalfSelect on frequency $j-1$, using the same argument, if A'' is the set of nodes in B_i active at time $\tau_i + b_j + b_{j-1}$ then $|A''| \leq |A'|/2 \leq 2^{j-2}$. Continuing inductively, all the nodes in A will succeed (and become dormant) no later than at time

$$\begin{aligned}
\tau_i + \textstyle\sum_{q=0}^{j} b_q \;&=\; \tau_i + \gamma(\textstyle\sum_{q=0}^{j} 2^q) \log n \\
&<\; \tau_i + \gamma 2^{j+1} \log n \\
&<\; \tau_i + 4\gamma |B_i| \log n \;=\; \tau_{i+1},
\end{aligned}$$

completing the proof of Claim 6.

By Claims 5 and 6, all nodes in $\bigcup_{p<i+1} B_p$ will be permanently dormant starting no later than at time τ_{i+1}. Each successful transmission from B_i arrives at a node in $\bigcup_{p\geq i+1} B_p$, so at time τ_{i+1} all rumors are in some active nodes in $\bigcup_{p\geq i+1} B_p$, by Claims 6 and 4. This completes the inductive step and the proof of Lemma 5. ◀

Taking $i = \delta^*$ in Lemma 5, we obtain that the algorithm delivers all rumors to t in at most $\tau_{\delta^*} = 4\gamma \sum_{p=0}^{\delta^*-1} |B_p| \log n \leq 4\gamma n \log n$ time steps. This proves correctness and establishes an $O(n \log n)$ bound on the running time in our relaxed communication model with κ frequencies and assumptions (MFC), (SRT), and (INN). Since $\kappa = O(\log n)$, as explained in Section 2, this implies an $O(n \log^2 n)$-time bound on the running time of Algorithm AcyGatherAck in the standard single-frequency model. This is summarized in the theorem below.

▶ **Theorem 6.** *Let G be an acyclic directed graph with n vertices and a designated target node reachable from all other nodes. Using acknowledgements of successful transmissions, Algorithm* AcyGatherAck *completes information gathering in G in time $O(n \log^2 n)$.*

6 Final Comments

In this paper we provided an $\tilde{O}(n^{1.5})$-time protocol for information gathering in ad-hoc radio networks, improving the trivial upper bound of $O(n^2)$. For the model with transmissions acknowledgments we gave a $\tilde{O}(n)$-time protocol for acyclic digraphs.

We hope that some ideas behind our algorithms will lead to further improvements, and perhaps find applications to other communication dissemination problems in ad-hoc radio networks. One idea that is particularly promising is the amortization technique in Section 3, where a failure of a node in transmitting its message is charged to stage-index increments of the interfering nodes. Another idea is the technique for integrating a gossiping protocol (applicable only to strongly connected digraphs) with an information gathering protocol for acyclic digraphs, to obtain an information gathering protocol for arbitrary digraphs. Using this technique, improving the upper bound to below $\tilde{O}(n^{1.5})$ should be possible by designing an appropriate protocol that beats the $\tilde{O}(n^{1.5})$ bound for acyclic graphs.

Several open problems remain. The two most intriguing problems are about the time complexity of gossiping and information gathering, as for both problems the best known lower bounds are only $\Omega(n \log n)$, the same as for broadcasting.

There are a number of other natural questions about information gathering protocols for radio networks that deserve study. For example, how does the complexity of information gathering depend on the graph diameter D and maximum degree Δ? Some initial work in this direction was done in [28], where refined bounds for the case of trees were given. Another natural direction of research would be to analyze the complexity of information gathering when the graph topology is known. This has been well studied for broadcasting and gossiping – see, for example, the survey in [15].

Acknowledgements. We are very grateful to the anonymous reviewers for pointing out several deficiencies in the original submission and for insightful comments that helped us improve the quality of presentation.

References

1 Noga Alon, Amotz Bar-Noy, Nathan Linial, and David Peleg. A lower bound for radio broadcast. *J. Comput. Syst. Sci.*, 43(2):290–298, 1991.

2 Danilo Bruschi and Massimiliano Del Pinto. Lower bounds for the broadcast problem in mobile radio networks. *Distributed Computing*, 10(3):129–135, 1997.

3 Bogdan S. Chlebus, Leszek Gasieniec, Alan Gibbons, Andrzej Pelc, and Wojciech Rytter. Deterministic broadcasting in ad hoc radio networks. *Distributed Computing*, 15(1):27–38, 2002.

4 Marek Chrobak and Kevin P. Costello. Faster information gathering in ad-hoc radio tree networks. *Algorithmica*, 80(3):1013–1040, 2018.

5 Marek Chrobak, Kevin P. Costello, Leszek Gasieniec, and Dariusz R. Kowalski. Information gathering in ad-hoc radio networks with tree topology. *Information and Computation*, 258:1–27, 2018.

6 Marek Chrobak, Leszek Gasieniec, and Wojciech Rytter. Fast broadcasting and gossiping in radio networks. *Journal of Algorithms*, 43(2):177–189, 2002.

7 Marek Chrobak, Leszek Gasieniec, and Wojciech Rytter. A randomized algorithm for gossiping in radio networks. *Networks*, 43(2):119–124, 2004.

8 Andrea E. F. Clementi, Angelo Monti, and Riccardo Silvestri. Selective families, superimposed codes, and broadcasting on unknown radio networks. In *Proc. 12th Annual Symposium on Discrete Algorithms (SODA'01)*, pages 709–718, 2001.

9 Andrea E. F. Clementi, Angelo Monti, and Riccardo Silvestri. Distributed broadcast in radio networks of unknown topology. *Theor. Comput. Sci.*, 302(1-3):337–364, 2003.

10 Artur Czumaj and Peter Davies. Faster deterministic communication in radio networks. In *Proc. 43rd International Colloquium on Automata, Languages, and Programming (ICALP'16)*, pages 139:1–139:14, 2016.

11 Artur Czumaj and Wojciech Rytter. Broadcasting algorithms in radio networks with unknown topology. *Journal of Algorithms*, 60(2):115 – 143, 2006.

12 Gianluca De Marco. Distributed broadcast in unknown radio networks. In *Proc. 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'08)*, pages 208–217, 2008.

13 Gianluca De Marco. Distributed broadcast in unknown radio networks. *SIAM J. Comput.*, 39(6):2162–2175, 2010.

14 Antonio Fernández Anta, Miguel A. Mosteiro, and Jorge Ramón Muñoz. Unbounded contention resolution in multiple-access channels. *Algorithmica*, 67(3):295–314, 2013.

15 Leszek Gasieniec. On efficient gossiping in radio networks. In *Proc. 16th Int. Colloquium on Structural Information and Communication Complexity (SIROCCO'09)*, pages 2–14, 2009.

16 Leszek Gasieniec. Deterministic broadcasting in radio networks. In *Encyclopedia of Algorithms*, pages 529–530. Springer US, 2016.

17 Leszek Gasieniec and Igor Potapov. Gossiping with unit messages in known radio networks. In *Proc. 2^{nd} IFIP International Conference on Theoretical Computer Science (TCS'02)*, pages 193–205, 2002.

18 Leszek Gasieniec, Tomasz Radzik, and Qin Xin. Faster deterministic gossiping in directed ad hoc radio networks. In *Proc. Scandinavian Workshop on Algorithm Theory (SWAT'04)*, pages 397–407, 2004.

19 Alon Itai. Randomized broadcasting in radio networks. In *Encyclopedia of Algorithms*, pages 1734–1738. Springer US, 2016.

20 Tomasz Jurdzinski and Dariusz R. Kowalski. Wake-up problem in multi-hop radio networks. In *Encyclopedia of Algorithms*, pages 2352–2354. Springer, 2016.

21 Dariusz R. Kowalski and Andrzej Pelc. Faster deterministic broadcasting in ad hoc radio networks. *SIAM J. Discrete Math.*, 18(2):332–346, 2004.

22 Eyal Kushilevitz and Yishay Mansour. An $\Omega(D \log(N/D)$ lower bound for broadcast in radio networks. *SIAM J. Comput.*, 27(3):702–712, 1998.

23 Ding Liu and Manoj Prabhakaran. On randomized broadcasting and gossiping in radio networks. In *Proc. 8th Annual Int. Conference on Computing and Combinatorics (COCOON'02)*, pages 340–349, 2002.

24 Gianluca De Marco and Dariusz R. Kowalski. Contention resolution in a non-synchronized multiple access channel. In *Proc. of the 27th Int. Symposium on Parallel Distributed Processing (IPDPS)*, pages 525–533, 2013.

25 Gianluca De Marco and Dariusz R. Kowalski. Fast nonadaptive deterministic algorithm for conflict resolution in a dynamic multiple-access channel. *SIAM Journal on Computing*, 44(3):868–888, 2015.

26 Gianluca De Marco and Dariusz R. Kowalski. Contention resolution in a non-synchronized multiple access channel. *Theor. Comput. Sci.*, 689:1–13, 2017.

27 David Peleg. Time-efficient broadcasting in radio networks: A review. In *Proc. Distributed Computing and Internet Technology, 4th International Conference, ICDCIT'07*, pages 1–18, 2007.

28 Parker Williams. *Information Gathering on Bounded Degree Trees and Properties of Random Matrices*. PhD thesis, University of California, Riverside, 2017.

29 Ying Xu. An $O(n^{1.5})$ deterministic gossiping algorithm for radio networks. *Algorithmica*, 36(1):93–96, 2003.

A About Protocol SccGossip

The analysis of Algorithm SccGossip in [18] (called Gossip2 in that paper) assumes that N is bounded polynomially in $\bar{n}$. Since in this case $\log N = O(\log \bar{n})$, the bound on the running time of their algorithm (see Theorem 2 of that paper) does not explicitly separate the dependence on $\bar{n}$ and N. We now explain how the bound given in our Property (scc) follows from the analysis in [18]. In essence, the running time depends on N in three ways: as the label range of selectors, as the range of binary search, or as the range of a doubling (or halving) process. All these three contributions are only logarithmic in N. (In fact, in some cases the range of binary search or doubling can be reduced to $\bar{n}$, but this is not relevant to our application.) A more detailed explanation follows.

The algorithm in [18] uses a broadcasting algorithm from [6] in their procedure Disperse() (Section 3). In [6] it is assumed that the label set is $[\bar{n}]$ and the running time is given as $O(\bar{n} \log^2 \bar{n})$. However, all this algorithm does is to repeatedly run selectors, so for the label range $[N]$ its running time can be expressed as $O(\bar{n} \log^2 N)$. Procedure Disperse() in [18] also involves a binary search, introducing another factor of $O(\log N)$. Thus the running time of Disperse() (Lemma 2 in [18]) can be restated as $O((\bar{n}/x) + r)\bar{n} \log^3 N)$. Phase I of Algorithm Gossip2 in Section 3.2 in [18] involves a halving process that executes a selector in each iteration. Phase II involves $O(\log \bar{n})$ iterations, each executing a selector. For this reason, only logarithmic factors in the running time will depend on N.