The Distributed Wireless Gathering Problem[*]

Vincenzo Bonifaci[†] Peter Korteweg[‡] Alberto Marchetti-Spaccamela[§]
Leen Stougie[¶]

September 15, 2010

Abstract

We address the problem of data gathering in a wireless network using multi-hop communication; our main goal is the analysis of simple algorithms suitable for implementation in realistic scenarios. We study the performance of distributed algorithms, which do not use any form of local coordination, and we focus on the objective of minimizing average flow times of data packets. We prove a lower bound of $\Omega(n)$ on the expected competitive ratio of any acknowledgment-based distributed algorithm minimizing the maximum flow time, where n is the number of nodes of the network. Next, we consider a distributed algorithm which sends packets over shortest paths, and we use resource augmentation to analyze its performance when the objective is to minimize the average flow time. If interferences are modeled as in Bar-Yehuda et al. (J. of Computer and Systems Science, 45(1):104–126, 1992) we prove that the algorithm is $(1+\epsilon)$-competitive, when the algorithm sends packets a factor $O(\log(\delta/\epsilon)\log\Delta)$ faster than the optimal offline solution; here δ is the radius of the network and Δ the maximum degree. We finally extend this result to a more complex interference model.

Keywords: wireless networks, data gathering, approximation algorithm, distributed algorithm, on-line algorithm, resource augmentation

1 Introduction

Wireless networks are used in many areas of practical interest, such as mobile phone communication, ad-hoc networks, and radio broadcasting. Moreover, recent advances in miniaturization of computing devices equipped with short range radios have given rise to strong interest in sensor networks for their relevance in many practical scenarios (environment control, accident monitoring etc.) [1, 21].

In many applications of wireless networks data gathering is a critical operation for extracting useful information from the operating environment: information collected from multiple nodes in the network should be transmitted to a sink that may process the data, or act as a gateway to other networks. In the case of wireless sensor networks, sensor nodes have limited computation capabilities, thus implying that data gathering is an even more crucial operation. For these reasons, data gathering in sensor networks has received significant attention in the last few years; we restrict ourselves to cite only two contributions, where further pointers to the literature can be found [1, 10]. The problem also finds applications in Wi-Fi networks when many users need to access a gateway using multi-hop wireless relay-routing [5].

[*]A preliminary version appeared in *Proc. 4th Conf. on Algorithmic Aspects of Information and Management*, pp. 72–83, Springer, 2008.

[†]Max-Planck-Institut für Informatik, Saarbrücken, Germany, email `bonifaci@mpi-inf.mpg.de`.

[‡]Eindhoven University of Technology, the Netherlands, email `p.korteweg@tue.nl`.

[§]Sapienza Università di Roma, Italy, email `alberto@dis.uniroma1.it`.

[¶]Vrije Universiteit Amsterdam and CWI, Amsterdam, the Netherlands, email `stougie@cwi.nl`.

An instance of the *Wireless Gathering Problem* (WGP) is given by a static wireless network which consists of several stations (nodes) and one base station (the sink), modeled as vertices of a graph; over time data packets arrive at stations that have to be gathered at the base station. In the sequel we assume that time is discrete and that stations have a common clock, hence time can be divided into rounds. Following Bar-Yehuda et al. [2], we adopt the half-duplex model, in which nodes can either send or receive during a single round. Typically, not all nodes in the network can communicate with each other, hence packets have to be sent through several nodes before they can be gathered at the sink; this is called *multi-hop* routing. The crucial issue to be considered is *interference*: the communication between two pairs of nodes causes interference if either of the receiver nodes also receives the communication signal intended for the other node. In case of interference, the receiver node does not receive the packet (we do not assume any collision detection mechanism).

Realistic models of communication among nodes depend on many parameters that influence the performance of transmissions (see for example [1,25]); this raises many combinatorial optimization problems that received significant attention in the last few years. We notice that most solution methods proposed so far focus on polynomial time algorithms that achieve provably good performance in the worst case. Unfortunately these methods are not suitable for practical implementations; in fact they are centralized and/or require solving complex combinatorial optimization problems [5,15,16]. Since sensors have limited computational power and are unable to perform sophisticated coordination activities, sophisticated algorithms that require solving complex combinatorial optimization problems are impractical for implementations and have mainly theoretical interest. Communication algorithms that have been implemented and tested in real scenarios are not only distributed but can be implemented with very limited overhead (see e.g. BMAC and DMAC [18,23]).

We notice that a formal analysis of the performance of simple algorithms suitable for practical implementation is still missing. In this paper we continue the work initiated in [2,3,8] on the problem of analyzing *simple distributed* algorithms that have *good approximation guarantees* in *realistic scenarios*. We emphasize that the algorithms analyzed are very simple and that the challenge is not in their design but in their analysis. Namely, we consider fully distributed algorithms, i.e., each node makes decisions of when to transmit independently of other nodes. Our model is more restricted than decentralized algorithms which may allow information exchange between neighboring nodes before transmission to decrease the possibility of interference even further.

In order to perform our analysis we make several assumptions that are discussed in the sequel. First of all we assume that time is discrete and that nodes share a common clock. We observe that this assumption is common to most (if not all) research that aims to formally evaluate communication algorithms and that a synchronized network is necessary in many applications of sensor networks. For these reasons, there is an extensive study on network synchronization; as a result, known synchronization algorithms are very effective in practice, being able to reduce the time difference among clocks to a few tens of microseconds (see for example [24]).

We also assume that each node has a unique identifier. We observe that there are scenarios where there are no node identifiers; nevertheless, the assumption that nodes are uniquely identified is coherent with the limitations imposed by the technology: it is known that even RFID tags have an identifier plugged in during their production. Moreover, there are basic tasks that require unique identification of nodes in order to break symmetries.

Furthermore, we assume that all nodes of the network are connected by a routing tree and that each node is aware of the distance (number of hops) to the sink. We observe that methods for building a routing tree are well-known and adopted in practice (see for example [11,17,26]). Namely, we assume that packets are routed through a breadth-first search (BFS) tree; such a tree can be constructed efficiently using distributed algorithms [3]. The routing tree is then used for communication. Because of interference, a node can receive at most one message from its children: if two or more children try to reach their parent in the same round, then a conflict arises. Randomization is a tool that is well-known for its simplicity and effectiveness in practice in order to decrease the possibility of conflicts, in particular in the context of collision avoidance protocols.

A relevant parameter that has been considered in the literature is Δ, the maximum degree of a node in the network [3, 15, 16]. We assume that all nodes know Δ (or an upper bound on it); we observe that such an information can be easily obtained once the routing tree has been constructed.

To assess the quality of the algorithm we use resource augmentation, a technique that, in the context of machine scheduling, was introduced by Kalyanasundaram and Pruhs [12]. The idea is to study the performance of on-line algorithms which are given processors faster than those of the adversary. Intuitively, this is done to compensate an on-line scheduler for its lack of future information. Such an approach has led to a number of interesting results showing that moderately faster processors are sufficient to attain satisfactory performance guarantees for different scheduling problems, see for example [12, 22].

Related work. All of the above assumptions also underlie the seminal paper by Bar-Yehuda et al. [3], where the problem of data gathering has been considered as a subproblem in wireless routing. The authors prove that a simple randomized algorithm requires in expectation $O((m+\delta)\log\Delta)$ rounds to gather m packets, where δ is the radius of the network, and Δ is the maximum degree of a node. Hence, their algorithm is in expectation an $O(\log\Delta)$-approximation of the optimal maximum completion time. A restrictive assumption in Bar-Yehuda et al. [3] is that all packets are released simultaneously. The authors pose as open problems the study of packets released over time and the analysis of objective functions different from the maximum completion time.

The wireless gathering problem in a centralized setting was studied by Bermond et al. [4–6] and Bonifaci et al. [7, 8]. Bermond et al. [5] showed that the problem of minimizing maximum completion time is **NP**-hard, even when all release times are zero. In Ref. [7, 8] we studied centralized and distributed algorithms for the wireless gathering problem with arbitrary release times. For the case of minimizing the maximum completion time, we presented a 4-competitive algorithm, and proved that no shortest paths following algorithm can be better than 4-approximate [7]. For the case of minimizing maximum flow time, we presented an algorithm which, under resource augmentation by sending packets a factor 5 faster than an offline algorithm, produces a solution with value at most the optimal off-line solution value. For both the minimization of maximum flow time and of average flow time, we also showed that for any $\epsilon > 0$ no algorithm can be better than $\Omega(m^{1-\epsilon})$-approximate, without resource augmentation, unless $\mathbf{P} = \mathbf{NP}$ [8]. We remark that distributed algorithms considered in [8] require coordination among neighboring nodes to avoid interferences during transmission; this can be accomplished through a suitable signaling protocol that however is not realistic in current sensor networks.

Kumar et al. [15, 16] considered decentralized algorithms for wireless packet routing. They provide near-optimal polynomial time routing and scheduling algorithms for solving throughput maximization problems in wireless ad-hoc networks. However, their algorithms are not distributed in the sense that nodes use information about neighboring nodes in order to decrease interference and in some case hinge on the solution of a linear or a convex optimization problem. For this reason the above results are mainly of theoretical interest.

As we remarked, there has been much work in recent years on various optimization issues in communication algorithms for wireless and sensor networks: see [13, 14, 20, 27] and references therein. However, to the best of our knowledge, a formal analysis of the worst case performance of simple distributed algorithms that are suitable for implementation in real sensor networks scenarios is still missing.

2 Preliminaries

We formulate WGP as a graph optimization problem. Given is an undirected graph $G = (V, E)$ with $|V| = n$, sink $s \in V$, and a set of data packets $M = \{1, 2, \ldots, m\}$ which arrive over time. We assume that each edge has unit length. For each pair of nodes $u, v \in V$ we define the *distance* between u and v, denoted by $d(u, v)$, as the length of a *shortest path* from u to v in G. A *layer* of nodes is a set of all nodes at distance d from sink s, for some d. Each packet $j \in M$ has an *origin* $v_j \in V$ and a release time $r_j \in \mathbb{Z}_+$ at which it enters the network.

We assume that time is discrete; we call a time unit a *round.* Packet j can be sent for the first time in round r_j. The rounds are numbered $0, 1, 2, \ldots$. During each round a node may either be *sending* a packet, be *receiving* a packet, or be *inactive.* If $d(u, v) = 1$ then u can send some packet j to v during a round. If node u sends a packet j to v in some round, then the pair (u, v) is called a *call* of packet j during that round. Two calls (u, v) and (u', v') *interfere* if $d(u', v) = 1$ or $d(u, v') = 1$; otherwise the calls are *compatible.* The solution of WGP is a schedule of compatible calls such that all packets are sent to the sink.

The above definition of interference is known in the literature as primary interference; we consider in section 5 the extension to incorporate secondary interference, i.e., interference between non-adjacent but proximate links in the network.

Given a schedule, let v_j^t be the node where packet j resides at time t. The quantity $C_j := \min\{t : v_j^t = s\}$ is called the *completion time* of packet j. We define $F_j := C_j - r_j$ as the *flow time* of packet j. We assume that packets cannot be aggregated. As extra notation, let $\delta_j := d(v_j, s)$ be the minimum number of calls required for packet j to reach s, and let $\delta := \max_{j \in M} \delta_j$ (that is, δ is the radius of the network).

In this paper we analyze distributed algorithms under the following assumptions. A network with a shortest paths routing tree is given, and each node knows the next node on the path to the tree root (the sink), as well as $d(v_j, s)$, its distance to the root. Further, we assume that each node is equipped with a clock and the clocks are synchronized, i.e., they indicate the same time. This enables nodes to synchronize packet communications. Also, each node knows an upper bound on the maximum node degree, called Δ.

3 A lower bound for distributed algorithms

It is known from previous work that WGP is **NP**-hard when minimizing the maximum completion time, the maximum flow time or the total flow time [7, 8]. The latter problems regarding flow times are in fact **NP**-hard to approximate within a factor $\Omega(m^{1-\epsilon})$, for any $\epsilon > 0$ [8]. Here we give an *unconditional* lower bound on the approximability of the maximum flow time in a distributed setting where routing is done through an in-tree. We consider a scenario in which the conflicts between transmissions from one layer of the tree to the next are resolved randomly: whenever several transmissions from a layer occur in the same round, only a uniformly chosen one succeeds. We call this the *random selection* model. This assumption seems natural for distributed algorithms, as they have no simple means of coordinating the transmitting nodes (or more precisely, coordinating the transmitting nodes is as hard as the original communication task). We will give our lower bound on a star network with the sink at the center. In this case, the gathering problem becomes similar to the problem of accessing a single multiple-access channel. We can assume that every station processes packets in the order they are injected at the station. As is standard in the literature, a (deterministic) distributed gathering protocol can be modeled as an automaton [19], determined by an internal state and a transition function. Each internal state stipulates whether the pending packet is to be transmitted towards the sink in the current round. Our other assumption is that the protocol is *acknowledgment-based* [9]. This means that the state of every node depends only on: (1) the number of nodes in the network; (2) the identity of the node; (3) the number of rounds elapsed since the currently processed packet became the first in the FIFO buffer at the node.

Proposition 3.1. *In the random selection model, there are instances for which the expected competitive ratio of any acknowledgment-based protocol is at least $\Omega(n)$ when minimizing the maximum flow time.*

Proof. Consider any acknowledgment-based protocol in the random selection model; we give the lower bound on a star network of n nodes, with the center of the star being the sink node. Because the protocol is acknowledgment-based, there are two possibilities: either (1) there is some node v for which the protocol prescribes to wait for at least a round after a new packet comes at the top of the node's FIFO buffer; or (2) all nodes try to transmit in the round in which a new packet comes at the top of the buffer.

In case (1), injecting n packets at node v, one every round for n rounds, allows the adversary to deliver every packet the round after it has been released, so that the maximum flow time is 1. On the other hand, the maximum flow time of the protocol is $\Theta(n)$, because when the last packet is injected at most $n/2$ packets have been delivered (by our assumption that v waits one extra round before transmitting). Thus in case (1) the lower bound is established. In the following we focus on case (2).

In this case, the adversary will release $n2^n$ packets. The sequence consists of 2^n consecutive phases, each of n rounds. In the first round of each phase, two packets are released simultaneously on nodes 1 and 2 of the star, and in the j-th round $(1 < j < n)$ a packet is released on node $j + 1$. In the last round of each phase no new packet is released. The adversary delivers to the sink one of the packets released in the first round of a phase immediately, and the other packet in the following round. The adversary then sends all other packets one round after their release dates, hence the maximum flow time of the adversary is 2.

The protocol can obviously send only one of the packets released in the first round of the phase, and has to send the other packet in a later round. We call the packet which is released but not sent by the protocol the *target packet*. We prove the proposition by showing that the expected flow time of one of the n target packets is at least n.

Since we are in case (2), every node will transmit in the first round in which a new packet is injected into it. This means that in each round of a phase some packet different from the target packet is also trying to access the sink.

Let X be the random variable indicating the flow time of a target packet. We know that $X \geq 2$. Because of the random selection rule, at every round the target packet has a chance of $1/2$ of reaching the sink (or less, depending on the protocol, but that can only increase the flow time). In formulas, for all t such that $1 \leq t < n$,

$$\Pr[X = t + 1 \,|\, X > t] = 1/2.$$

This means that $\Pr[X = t + 1] = 2^{-t}$.

Consider the set of 2^n target packets. For $\ell = 1, \ldots, 2^n$, let $X_\ell = 1$ be the event that target packet ℓ has flow time n. Otherwise $X_\ell = 0$. Then $\mathbb{E}[X_\ell] = 2^{-n+1}$ for all $\ell = 1, \ldots, n$ and therefore the expected number of target packets with flow time at least n is $\mathbb{E}[\sum_{\ell=1}^{2^n} X_\ell] = \sum_{\ell=1}^{2^n} \mathbb{E}[X_\ell] = 2$. Hence, in expectation at least one target packet has flow time $\Omega(n)$. $\square$

It is useful to compare Proposition 3.1 with the above cited result, that there is no polynomial time algorithm which can approximate WGP within a factor sublinear in the number of packets when minimizing the maximum flow time, unless $\mathbf{P} = \mathbf{NP}$. The latter condition is not required in Proposition 3.1. In [8] it was shown that, allowing a constant increase in speed, one can obtain a solution with maximum flow time which is less than that of the optimal solution for the original instance. Proposition 3.1 indicates that such a result is not attainable by distributed algorithms that choose randomly the packets to be advanced. Most of the distributed algorithms that have been proposed for the gathering problem (for example, the one in [3]) are of this type, but it would be interesting to extend this negative result to all distributed algorithms.

In the next section, we will analyze a distributed algorithm. It cannot determine which packet is advanced from each layer to the next. Proposition 3.1 and the observations above should thus explain why we focus on the analysis of the total flow time (as opposed to maximum flow time) and why we need to use resource augmentation.

4 A distributed algorithm and its analysis

4.1 The algorithm

We consider a distributed algorithm for Wgp first introduced by Bar-Yehuda et al. in the context of minimizing the maximum completion time for the gathering problem without release dates [3]. We focus on flow times.

To reduce interference between nodes, the algorithm uses node *labels*. A node at distance d from the sink is assigned label $d \bmod 3$. Each node can be either *active* during a round or *inactive*; only active nodes will transmit a packet. A node will not be active if its packet buffer is empty.

Before we describe the algorithm, we introduce a basic but crucial procedure which enables communication from a set of active nodes. The procedure, first introduced and studied by Bar-Yehuda, Goldreich and Itai [2], is called Decay and requires $2\log\Delta$ rounds; the time needed for a single execution of the procedure is called a *phase*.

Algorithm 1 Decay(u, v) [2]

```
for $j = 1, 2, \ldots, 2\log\Delta$ do
  $u$ sends to $v$ the oldest packet from its buffer;
  $u$ deactivates itself for the rest of the phase with probability $1/2$.
end for
```

We can now state a distributed algorithm for Wgp (Algorithm 2). The protocol requires

Algorithm 2 DistributedGreedy (DG) [3]

```
Construct a breadth-first search tree with root $s$
for each next phase $k = 1, 2, \ldots$ do
  Activate each node with label $k \bmod 3$ having a nonempty packet buffer;
  Execute Decay$(u, \text{parent}(u))$ in parallel for each active node $u$.
end for
```

the existence of a basic communication structure. Therefore, as a first step of the algorithm a breadth-first search (BFS) tree with the sink as the root is constructed. This can be done in expected time $O((n + \delta\log n)\log\Delta)$ [3]. Since it is done only once, when starting up the system, we will not count this time in defining the cost of the schedule.

Although the algorithm does not model acknowledgement of packets explicitly, it is easy to include them, e.g. by doubling the number of rounds, having communication in odd rounds and acknowledgements in even rounds, as in [3]. Using this, we can assume that successful receipt of a packet (by the parent of the sending node in the BFS tree) is acknowledged immediately. Only at that time it gets removed from the sender's buffer.

By the transmission protocol in DG, where in phase k only nodes of layer $k \bmod 3$ transmit, if two nodes transmit, then either they are at the same layer or they are at least distance 3 apart. Hence, in DG two nodes can only interfere if both sender nodes are in the same layer.

A *superphase* consists of three consecutive phases (thus it lasts $6\log\Delta$ rounds). Another important ingredient in the analysis of DG is the following.

Theorem 4.1 ([3]). *Let i be a layer of the tree containing some packet at the beginning of a superphase. There is probability at least $\mu := e^{-1}(1 - e^{-1})$ that, during this superphase,* DG *sends a packet from a node u in layer i successfully to* parent(u) *in the BFS tree.*

This theorem shows that, during a superphase, each nonempty layer forwards a packet with probability μ to the following layer. Notice, however, that there is no guarantee on which particular packet is advanced.

4.2 The analysis

For our analysis we define three solution models. For a given instance of WGP, we relate the completion times of the packets in these three models. This approach is similar to the approach of Bar-Yehuda et al. [3]. However, one additional difficulty is that, because packets have release dates, the extra speed given to the algorithm does not directly translate into shorter flow times.

We formulate the three models below. By *advancing* a packet we mean that the packet is sent from its current node to its parent in the BFS tree.

- Model 1 is the previously discussed radio network model, where packets are routed according to the BFS tree, and each layer advances with probability μ at least one packet during every superphase.
- Model 2 consists of a chain of $\delta + 1$ nodes, where packets at the ith level of the previous model reside in node i of the path. From each node of the path at most one packet can advance during every superphase, and the probability that this occurs is exactly μ.
- Model 3 also consists of a chain, but each layer advances with probability $p := 1 - (1 - \mu)^\alpha$ at least one packet every round (for some parameter $\alpha \geq 1$ to be fixed later).

Note that the probabilities stated above (in all three models) assume that the node or layer considered contains some packet to be advanced; otherwise, the probability that a packet is advanced from that node is obviously zero. It follows from Theorem 4.1 that the solution generated by DG (at unit speed) is a solution which fits into Model 1.

Motivated by the negative results of Proposition 3.1 we focus on deriving a bound on the expected average flow time of DG. We use resource augmentation to analyze the performance of DG. A σ-speed algorithm sends data packets at a speed that is σ times faster than an offline algorithm.

Through a sequence of steps we relate the expected flow times of Model 1 to those of Model 2. Subsequently, we demonstrate that the expected flow time of σ-speed DG is bounded by the expected flow time of a Model 3-solution. Finally, we upper bound the expected flow times of a Model 3-solution in terms of the expected flow times of an optimal offline solution.

Lemma 4.2. *The expected sum of flow times of a Model 1-schedule is at most the expected sum of flow times of a Model 2-schedule, for every* WGP*-instance.*

The proof of this Lemma is given later as it requires some preliminary results. The lemma 4.2 extends the work of Bar-Yehuda et al. [3] to the case of WGP with release dates. We use their proof techniques to derive a bound on the sum of flow times for instances where packets may have release dates.

For the proofs we have to introduce some notation. Consider a distribution of packets among the levels, that is a vector $\mathbf{x} = (x_0, \ldots, x_\delta)$ such that $x_i \geq 0$ (level 0 consists of the sink node). A *move vector* is any $(\delta + 1)$-dimensional vector of nonnegative integers, $\mathbf{m} = (m_0, \ldots, m_\delta)$. Similarly, we can define an *arrival vector* $\mathbf{a} = (a_0, \ldots, a_\delta)$. Partition $\mathbf{x}' = \text{Move}(\mathbf{x}, \mathbf{m}, \mathbf{a})$ is obtained when m_i packets are moved from level i to level $i - 1$ and a_i new packets are released on level i. We allow $m_i > x_i$ by stipulating that, if $m_i > x_i$, then only x_i packets are moved. More precisely, the number of packets moved from level i is $\min(x_i, m_i)$. So $x'_i = x_i + a_i - \min(x_i, m_i) + \min(x_{i+1}, m_{i+1})$.

A *move sequence* is an infinite sequence $\mathbf{M} = (\mathbf{m}^1, \mathbf{m}^2, \cdots)$ of move vectors; similarly we can define an *arrival sequence* $\mathbf{A} = (\mathbf{a}^1, \mathbf{a}^2, \cdots)$. We denote with $\text{Move}^*(\mathbf{x}, \mathbf{M}, \mathbf{A}, t)$ the result of making t moves according to $\mathbf{M}$ and $\mathbf{A}$, that is

$$\begin{aligned}\text{Move}^*(\mathbf{x}, \mathbf{M}, \mathbf{A}, 0) &= \mathbf{x}\\ \text{Move}^*(\mathbf{x}, \mathbf{M}, \mathbf{A}, t+1) &= \text{Move}(\text{Move}^*(\mathbf{x}, \mathbf{M}, \mathbf{A}, t), \mathbf{m}^t, \mathbf{a}^t).\end{aligned}$$

We say that a move vector $\mathbf{m}$ *dominates* a move vector $\widetilde{\mathbf{m}}$ if $m_i \geq \widetilde{m}_i$ for all i. This notion can be extended to move sequences: $\mathbf{M}$ dominates $\tilde{\mathbf{M}}$ if $\mathbf{M}^t$ dominates $\tilde{\mathbf{M}}^t$ for all t. We also define a partial order $\preceq$ on the set of distribution vectors, where $\mathbf{x} \preceq \mathbf{y}$ if and only if there exists a move sequence $\mathbf{M}$ and an integer t such that $\mathbf{x} = \text{Move}^*(\mathbf{y}, \mathbf{M}, \mathbf{0}, t)$. Bar-Yehuda et al. showed the following relation between move vectors and distribution vectors.

Lemma 4.3 ([3])**.** *If* $\mathbf{m}$ *dominates* $\widetilde{\mathbf{m}}$ *and* $\mathbf{x} \preceq \mathbf{y}$ *then* $\text{Move}(\mathbf{x}, \mathbf{m}, \mathbf{0}) \preceq \text{Move}(\mathbf{y}, \widetilde{\mathbf{m}}, \mathbf{0})$.

Corollary 4.4. *If* $\mathbf{m}$ *dominates* $\widetilde{\mathbf{m}}$ *and* $\mathbf{x} \preceq \mathbf{y}$ *then* $\text{Move}(\mathbf{x}, \mathbf{m}, \mathbf{a}) \preceq \text{Move}(\mathbf{y}, \widetilde{\mathbf{m}}, \mathbf{a})$ *for any arrival vector* $\mathbf{a}$.

Proof. Follows from Lemma 4.3, the fact that $\text{Move}(\mathbf{x}, \mathbf{m}, \mathbf{a}) = \text{Move}(\mathbf{x}, \mathbf{m}, \mathbf{0}) + \mathbf{a}$ and the fact that $\mathbf{x} \preceq \mathbf{y}$ implies $\mathbf{x} + \mathbf{a} \preceq \mathbf{y} + \mathbf{a}$ for any vector $\mathbf{a}$. □

Let $\text{tft}(\mathbf{M}, \mathbf{A}) = \sum_{t=0}^{\infty} \sum_{i \neq 0} (\text{Move}^*(\mathbf{0}, \mathbf{M}, \mathbf{A}, t))_i$. Notice that this gives the total flow time when the move sequence of the algorithm is $\mathbf{M}$ and the arrival sequence is $\mathbf{A}$.

Lemma 4.5. *If* $\mathbf{M}$ *dominates* $\tilde{\mathbf{M}}$ *then* $\text{tft}(\mathbf{M}, \mathbf{A}) \leq \text{tft}(\tilde{\mathbf{M}}, \mathbf{A})$.

Proof. By applying Corollary 4.4 repeatedly to the sequence of move vectors we obtain $\text{Move}^*(\mathbf{0}, \mathbf{M}, \mathbf{A}, t) \preceq \text{Move}^*(\mathbf{0}, \tilde{\mathbf{M}}, \mathbf{A}, t)$ for all t. In particular this implies $(\text{Move}^*(\mathbf{0}, \mathbf{M}, \mathbf{A}, t))_0 \geq (\text{Move}^*(\mathbf{0}, \tilde{\mathbf{M}}, \mathbf{A}, t))_0$ for all t, and the claim follows from the definition of tft. □

Armed with this lemma we can proceed to prove Lemma 4.2.

Proof of Lemma 4.2. The technique of the proof is similar to the one used by Bar-Yehuda et al. [3]. For completeness, we give the entire proof. Consider an instance $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^T$ of model 1, where x_i^t is the number of packets at level i at time $t \leq T$. Recall that m_i^t is the number of packets that moved from level i to level $i-1$ at time t. When $x_i^t \geq 1$ then, by Theorem 4.1, $\Pr[m_i^t \geq 1] \geq \mu$. On the other hand when $x_i^t = 0$ then $m_i^t = 0$ and clearly $\Pr[m_i^t \geq 1] = 0$, which violates the probabilistic assumption of model 2. We therefore define the move sequence $\bar{\mathbf{M}} = (\bar{\mathbf{m}}^1, \bar{\mathbf{m}}^2, \cdots)$ as follows: if $x_i^t \geq 1$ then $\bar{m}_i^t = m_i^t$, otherwise $\bar{m}_i^t = 1$. Also, let $p_{i,t} := \Pr[\bar{m}_i^t \geq 1]$. By Theorem 4.1, $p_{i,t} \geq \mu$. Since $\mathbf{M}$ and $\bar{\mathbf{M}}$ differ only when $x_i^t = 0$, for every $t \leq T$, $\text{Move}^*(\mathbf{0}, \mathbf{M}, \mathbf{A}, t) = \text{Move}^*(\mathbf{0}, \bar{\mathbf{M}}, \mathbf{A}, t)$. In particular, $\text{tft}(\mathbf{M}, \mathbf{A}) = \text{tft}(\bar{\mathbf{M}}, \mathbf{A})$.

We now construct a move sequence $\tilde{\mathbf{M}} = (\widetilde{\mathbf{m}}^1, \widetilde{\mathbf{m}}^2, \cdots)$ with $\widetilde{m}_i \in \{0, 1\}$. If $\bar{m}_i^t \geq 1$ then $\widetilde{m}_i^t = 1$ with probability $(p_{i,t} - \mu)/p_{i,t}$; otherwise $\widetilde{m}_i^t = 0$. Since $\bar{\mathbf{M}}$ dominates $\tilde{\mathbf{M}}$, by Lemma 4.5 the total flow time of $\bar{\mathbf{M}}$ is less than or equal to that of $\tilde{\mathbf{M}}$. Also, by construction of $\tilde{\mathbf{M}}$,

$$\begin{aligned}\Pr[\widetilde{m}_i^t = 0] &= \Pr[\bar{m}_i^t = 0] + \Pr[\bar{m}_i^t \geq 1 \text{ and } \widetilde{m}_i^t = 0]\\ &= \Pr[\bar{m}_i^t = 0] + p_{i,t} \cdot \frac{p_{i,t} - \mu}{p_{i,t}}\\ &= 1 - \mu.\end{aligned}$$

Thus the probability distribution of $\tilde{\mathbf{M}}$ is in accordance with model 2. The result follows by taking expectations. □

Lemma 4.6. *The expected sum of flow times of σ-speed* DG *is at most the expected sum of flow times of a Model 3-schedule, for $\sigma \geq 6\alpha \log \Delta$.*

Proof. Let $F_{j,\sigma}$ be the flow time of packet j in a σ-speed DG schedule of the given instance. We define the flow time of a packet j in a Model k solution with speed factor σ as $F^{(k)}_{j,\sigma}$, $k = 1, 2, 3$.

From Lemma 4.2 and the fact that DG is a Model 1-solution it follows that

$$\mathbb{E}[\sum_j F_{j,\sigma}] = \mathbb{E}[\sum_j F^{(1)}_{j,\sigma}] \leq \mathbb{E}[\sum_j F^{(2)}_{j,\sigma}]. \tag{1}$$

On the other hand, consider the consequence of speeding up the Model 2-schedule by a factor $\sigma \geq 6\alpha \log \Delta$: now, every $1/(6 \log \Delta)$ superphases (that is, every round) the probability for each layer of advancing at least a packet becomes $1 - (1 - \mu)^{\alpha}$; that is, the solution satisfies the constraints of a Model 3-schedule so that

$$\mathbb{E}[\sum_j F^{(2)}_{j,\sigma}] \leq \mathbb{E}[\sum_j F^{(3)}_{j,1}]. \tag{2}$$

The claim follows by combining (1) and (2). □

We define a *deterministic tandem queue* (with unit processing times) as a chain of processors consisting of a sink M_0, and a set of machines $M_i, i = 1, \ldots, \delta$. A job which has been processed on machine M_i is sent to machine $M_{i-1}, i = 1, \ldots, \delta$. The processing time of a job is 1 on each machine. New jobs can be released at integer times on any machine. In the next lemma we relate the expected flow time of a Model 3-schedule to the flow time of a tandem queue, in which the machines are the layers of the gathering tree. We then show in Lemma 4.8 the intuitively clear point that the optimal solution cannot do better than such a tandem queue.

Lemma 4.7. *The expected sum of flow times of a Model 3-schedule is at most $1/p^{\delta}$ times the sum of flow times of a deterministic tandem queue with unit processing times.*

Proof. Consider a Model 3 schedule S. In a given round, the probability that *all* layers advance a packet is (at least) p^{δ}. Suppose that we modify the schedule by not advancing any packet at all whenever some layer fails to advance a packet. In this way we can only increase flow times. On the other hand, apart from a number of empty rounds in which no packet is advanced, the schedule is equivalent to a schedule of a deterministic tandem queue with unit processing times, where each packet j of the Model 3 instance is transformed in a job arriving at machine M_{δ_j}. Call this last schedule S' and consider any interval of k rounds of S'. Since the probability that all layers advance a packet in S is at least p^{δ}, the expected number of rounds required in S to advance the same number of packets as these k rounds do in schedule S' is

$$\begin{aligned}
&k \cdot p^{\delta} \cdot \sum_{i=0}^{\infty} (1 - p^{\delta})^i \cdot (i + 1) = k \cdot p^{\delta} \cdot \sum_{i=0}^{\infty} \sum_{j=0}^{i} (1 - p^{\delta})^i = \\
&= k \cdot p^{\delta} \cdot \sum_{j=0}^{\infty} \sum_{i=j}^{\infty} (1 - p^{\delta})^i = k \cdot p^{\delta} \cdot \sum_{j=0}^{\infty} (1 - p^{\delta})^j \sum_{i=0}^{\infty} (1 - p^{\delta})^i = \\
&= k/p^{\delta}.
\end{aligned}$$

If F'_j is the flow time of packet j in S', we then have $\mathbb{E}[F^{(3)}_{j,1}] \leq p^{-\delta} F'_j$, for each packet j. □

Lemma 4.8. *The sum of flow times of a deterministic tandem queue with unit processing times is at most the sum of flow times of an optimal* WGP *schedule.*

Proof. Let S be a deterministic tandem queue schedule, and let S^* be a schedule where in each round each layer, except layer 1, can advance any number of packets, and layer 1 can advance at most one packet. Notice that the optimal schedule cannot do better than S^*.

Consider schedule S^*. Let M_t^* be the set of packets which have not arrived at the sink in round t. Because the schedule can advance any number of packets over an edge, we have that if no packet is sent to the sink in S^* in round t, then no packet in M_t^* can arrive at the sink before or at round t, i.e., $r_j + \delta_j > t$ for each $j \in M_t^*$. We prove the lemma by demonstrating that if some packet is sent to the sink in S^* in round t, then also some packet is sent to the sink in S in round t. This suffices to prove the lemma, because an optimal WGP schedule cannot advance packets faster than in schedule S^*.

Suppose to the contrary that there is a first round t in which some packet is sent to the sink in S^*, but no packet is sent to the sink in S. Let $t', t' < t$, be the last round before t in which no packet is sent to the sink in S^*. Then, there is a set of $t - t'$ packets in S^* which arrive at the sink in rounds $(t', t]$. Hence, it follows from this and the observation above that there are $t - t'$ packets j such that $t' < r_j + \delta_j \leq t$. Now consider schedule S; in this schedule $t - t' - 1$ packets are sent to the sink in rounds $(t', t-1]$, hence there is a packet j with $t' < r_j + \delta_j \leq t$ which has not arrived at the sink in round t. But then, j must have been in layer $1 + i$ or higher in rounds $t - i$, $i = 1, \ldots, \delta_j$. I.e., j must have been in layer $1 + \delta_j$ or higher in round $r_j \leq t - \delta_j$ which gives a contradiction. □

Theorem 4.9. *Let* $0 < \epsilon \leq 1$ *and* $\sigma = 6\mu^{-1} \cdot \log \Delta \cdot \ln(\delta/\epsilon)$. *Then* σ*-speed* DG *is in expectation* $(1 + 3\epsilon)$*-competitive when minimizing the average flow time.*

Proof. It follows from Lemmas 4.7 and 4.8 that the expected sum of flow times of σ-speed DG is at most $1/p^\delta$ times the sum of flow times of an optimal offline solution, for $\sigma = 6\alpha \log \Delta$. As σ-speed DG is an online algorithm, DG is σ-speed $p^{-\delta}$-competitive when minimizing average flow times. The probability $p = 1 - (1-\mu)^\alpha$ depends on the choice of the speedup α. We set $\alpha := \mu^{-1} \ln(\delta/\epsilon)$, which gives $p = 1 - (1-\mu)^{\mu^{-1}\ln(\delta/\epsilon)} \geq 1 - e^{-\ln(\delta/\epsilon)} = 1 - \epsilon/\delta$, so that $p^{-\delta} \leq (1 - \epsilon/\delta)^{-\delta} \leq e^\epsilon \leq e^{2\ln(1+\epsilon)} = 1 + 2\epsilon + \epsilon^2 \leq 1 + 3\epsilon$. □

It also follows from the theorem that the competitive ratio of DG can be made arbitrarily close to 1 with an appropriate increase in speed.

5 An extension

The model that we considered so far for WGP assumes that a node can only cause interference at nodes that are adjacent to it in the network. In practice, the interference caused by the radio signal can go beyond the transmission radius [25]. This can be modeled by an integer $d_I \geq 1$ that specifies the distance (expressed in number of hops) up to which the signal can cause interference: two calls (u, v) and (u', v') are now compatible if $d(u, v') \leq d_I$ and $d(u', v) \leq d_I$ (we still require that u is adjacent to v, and u' to v').

Algorithm DG can be extended to this setting by assigning to a node in layer d the label d mod $d_I + 2$ and then using superphases consisting of $d_I + 2$ phases each. In this way one can avoid interference between nodes from different layers of the tree. It is easy to see that Theorem 4.1 can be extended to this setting.

Using this fact, we can extend our analysis in the previous section to prove that, for $\epsilon > 0$ and $\sigma = \Theta(d_I^2 \cdot \log \Delta \cdot \ln(\delta/\epsilon))$, σ-speed DG is in expectation $(1+\epsilon)$-competitive when minimizing the average flow time. We notice that one d_I factor is due to the longer superphases, and the other one is due to DECAY having to cope with larger neighborhoods (of size Δ^{d_I}).

6 Conclusion and open problems

We considered the wireless gathering problem with the objective of minimizing the average flow time of data packets when nodes are restricted in their computational and communication capabilities. We showed that a simple on-line algorithm has favorable behavior when the objective function is minimizing the average flow: although the problem is extremely hard to approximate in general, augmenting the transmission rate allows to remain within a small factor of the cost of an optimal solution for the problem without augmentation. We also showed for maximum flow a lower bound on the competitive ratio of any acknowledgment-based algorithm, which depends on the size of the network.

The proposed algorithm is simple; however it assumes the existence of a common clock that is used to reduce interferences. It would be interesting to extend the results by removing such an assumption. It is interesting for future research to study other objective functions and to allow other routing than through a tree, in which case it will be challenging to design and analyze congestion-avoiding algorithms with better ratios than those developed for trees.

Acknowledgments

Research of the first and fourth authors partially supported by the Future and Emerging Technologies Unit of EC (IST priority - 6th FP), under contract no. FP6-021235-2 (project ARRIVAL). Research of third author partially supported by EU ICT-FET 215270 FRONTS. Research of first and third authors partially supported by MIUR-FIRB Italy-Israel project RBIN047MH9. Research of fourth author supported by the Dutch BSIK-BRICKS project.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.

[2] R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time-complexity of broadcast in multi-hop radio networks: an exponential gap between determinism and randomization. *Journal of Computer and Systems Sciences*, 45(1):104–126, 1992.

[3] R. Bar-Yehuda, A. Israeli, and A. Itai. Multiple communication in multihop radio networks. *SIAM Journal on Computing*, 22(4):875–887, 1993.

[4] J.-C. Bermond, R. C. Corrêa, and M.-L. Yu. Optimal gathering protocols on paths under interference constraints. *Discrete Mathematics*, 309(18):5574–5587, 2009.

[5] J.-C. Bermond, J. Galtier, R. Klasing, N. Morales, and S. Pérennes. Hardness and approximation of gathering in static radio networks. *Parallel Processing Letters*, 16(2):165–183, 2006.

[6] J.-C. Bermond, L. Gargano, and A. A. Rescigno. Gathering with minimum completion time in sensor tree networks. *Journal of Interconnection Networks*, to appear.

[7] V. Bonifaci, P. Korteweg, A. Marchetti-Spaccamela, and L. Stougie. An approximation algorithm for the wireless gathering problem. *Operations Research Letters*, 36(5):605–608, 2008.

[8] V. Bonifaci, P. Korteweg, A. Marchetti-Spaccamela, and L. Stougie. Minimizing flow time in the wireless gathering problem. *ACM Transactions on Algorithms*, to appear.

[9] B. S. Chlebus, D. R. Kowalski, and M. A. Rokicki. Maximum throughput of multiple access channels in adversarial environments. *Distributed Computing*, 22(2):93–116, 2009.

[10] C. Florens, M. Franceschetti, and R. J. McEliece. Lower bounds on data collection time in sensory networks. *IEEE Journal on Selected Areas in Communications*, 22:1110– 1120, 2004.

[11] C. Intanagonwiwat, R. Govindan, D. Estrin, J. S. Heidemann and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 11(1):2–16, 2003.

[12] B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. *Journal of the ACM*, 47(4):617–643, 2000.

[13] K. Kar, M. S. Kodialam, T. V. Lakshman, and L. Tassiulas. Routing for network capacity maximization in energy-constrained ad-hoc networks. In *Proc. 22nd IEEE Int. Conference on Computer Communications*, 2003.

[14] M. S. Kodialam and T. Nandagopal. Characterizing achievable rates in multi-hop wireless mesh networks with orthogonal channels. *IEEE/ACM Transactions on Networking*, 13(4):868–880, 2005.

[15] V. S. Anil Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan. End-to-end packet-scheduling in wireless ad-hoc networks. In J. I. Munro, editor, *Proc. 15th Symp. on Discrete Algorithms*, pages 1021–1030, 2004.

[16] V. S. Anil Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan. Provable algorithms for joint optimization of transport, routing and MAC layers in wireless ad hoc networks. In *Proc. of 4th Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 2007.

[17] S. Lindsey, C. Raghavendra and K. M. Sivalingam. Data gathering algorithms in sensor networks using energy metrics. *IEEE Transactions on Parallel and Distributed Systems*, 13(9):924–935, 2002.

[18] G. Lu, B. Krishnamachari, and C. S. Raghavendra. An adaptive energy-efficient and low-latency MAC for tree-based data gathering in wireless sensor networks. *Wireless Communications and Mobile Computing*, 7(7):863–875, 2007.

[19] N. A. Lynch. *Distributed Algorithms.* Morgan Kaufmann, 1996.

[20] T. Moscibroda and R. Wattenhofer. The complexity of connectivity in wireless networks. In *Proc. 25th IEEE Int. Conf. on Computer Communications*, 2006.

[21] K. Pahlavan and A. H. Levesque. *Wireless information networks.* Wiley, New York, 1995.

[22] C. A. Phillips, C. Stein, E. Torng and J. Wein. Optimal time-critical scheduling via resource augmentation. *Algorithmica*, 32(2):163–200, 2002.

[23] J. Polastre, J. L. Hill, and D. E. Culler. Versatile low power media access for wireless sensor networks. In *Proc. 2nd Int. Conf. on Embedded Networked Sensor Systems*, pages 95–107, 2004.

[24] K. Römer, P. Blum, and L. Meier. Time synchronization and calibration in wireless sensor networks. In *Handbook of Sensor Networks: Algorithms and Architectures*, pages 199–237. John Wiley and Sons, Hoboken, 2005.

[25] S. Schmid and R. Wattenhofer. Algorithmic models for sensor networks. In *Proc. 20th Int. Parallel and Distributed Processing Symp.*, 2006.

[26] L. Subramanian and R. H. Katz. An architecture for building self-configurable systems. In *Proc. of 1st Symp. on Mobile Ad Hoc Networking and Computing*, pages 63–73, 2000.

[27] L. Xiao, M. Johansson, and S. P. Boyd. Simultaneous routing and resource allocation via dual decomposition. *IEEE Transactions on Communications*, 52(7):1136–1144, 2004.