Latency Constrained Aggregation in Sensor Networks *

Luca Becchetti[1], Peter Korteweg[2], Alberto Marchetti-Spaccamela[1], Martin Skutella[3], Leen Stougie[2,4], and Andrea Vitaletti[1]

[1] University of Rome "La Sapienza"
[2] TU Eindhoven
[3] University of Dortmund
[4] CWI Amsterdam

Abstract. A sensor network consists of sensing devices which may exchange data through wireless communication; sensor networks are highly energy constrained since they are usually battery operated. Data aggregation is a possible way to save energy consumption: nodes may delay data in order to aggregate them into a single packet before forwarding them towards some central node (sink). However, many applications impose constraints on the maximum delay of data; this translates into latency constraints for data arriving at the sink.
We study the problem of data aggregation to minimize maximum energy consumption under latency constraints on sensed data delivery, and we assume unique communication paths that form an intree rooted at the sink. We prove that the off-line problem is strongly **NP**-hard and we design a 2-approximation algorithm. The latter uses a novel rounding technique.
Almost all real life sensor networks are managed on-line by simple distributed algorithms in the nodes. In this context we consider both the case in which sensor nodes are synchronized or not. We assess the performance of the algorithm by competitive analysis. We also provide lower bounds for the models we consider, in some cases showing optimality of the algorithms we propose. Most of our results also hold when minimizing the total energy consumption of all nodes.

Categories and Subject Descriptors: F.1.2 [**Computation by Abstract Devices**]: Modes of Computation — Online Computation; G.2.2 [**Discrete Mathematics**]: Graph Theory — Graph Algorithms, Network Problems, Trees.

General Terms: Discrete algorithms for communication, discrete optimization and approximation.

Additional Keywords and Phrases: Wireless sensor networks, data aggregation, competitive analysis, distributed algorithms.

* A preliminary version of this article appeared in Springer Lecture Notes on Computer Science, Volume 4168 (2006).

1 Introduction

A sensor network consists of sensor nodes and one or more central nodes or *sinks*. Sensor nodes are able to monitor events, to process the sensed information and to communicate the sensed data. Sinks are powerful base stations which gather data sensed in the network; sinks either process this data or act as gateways to other networks. Sensors send data to the sink through multi-hop communication.

A particular feature of sensor nodes is that they are battery powered, making sensor networks highly energy constrained. Replacing batteries on hundreds of nodes, often deployed in inaccessible environments, is infeasible or too costly. Therefore, a key challenge in a sensor network is the reduction of energy consumption and the most natural objective is to minimize the maximum energy consumption over all nodes. Energy consumption can be divided into three domains: sensing, communication and data processing [1]. Communication is most expensive because a sensor node spends most of its energy in data transmission and reception [12]. This motivates the study of techniques to reduce overall data communication, possibly exploiting processing capabilities available at each node. Data aggregation is one such technique. It consists of aggregating redundant or correlated data in order to reduce the overall size of sent data, thus decreasing the network traffic and energy consumption.

Most literature on sensor networks assumes *total aggregation*, i.e. data packets are assumed to have the same size and aggregation of two or more incoming packets at a node results in a single outgoing packet. Observe that even if this might be considered a simplistic assumption, it allows us to provide an upper bound on the expected benefits of data aggregation in terms of power consumption. We refer here to a selection of papers, focused on the algorithmic side of the problem [3, 11, 14–17]. However, these papers mainly focus on empirical and technical aspects

We concentrate on data aggregation in sensor networks under constraints on the *latency* of sensed events; this means that data should be communicated to the sinks within a specified time after being sensed. Preliminary results are given in [13, 21], but formal proofs of their results are not provided.

Time synchronization, in the sense of the existence of a common clock for the nodes, may or may not be a requirement of the sensor network. In typical applications a time stamp forms a crucial part of the sensed data. Time synchronization in wireless sensor networks has been studied in [6–8, 10]. Time synchronization introduces overhead and in some scenarios, a synchronous model, in which all nodes share the same clock, may not be a requirement. We consider both the *synchronous model* and the *asynchronous model*. We restrict ourselves to the analysis of deterministic algorithms.

A sensor network is naturally represented by a graph whose nodes are the sensors and whose arcs are the wireless communication links. Data aggregation, latency constraints and energy savings, give rise to a large variety of graph optimization problems depending on the following issues.

- communication energy and time can be seen as functions of the size of the packet and the communication arc. Typically, these are concave functions exhibiting economies of scale in the size of the packets sent.

- The latency may depend on the (types of) sensor data or on the sensor nodes.

- Sensor networks can be modeled as synchronous or asynchronous systems.

- Data is delivered to one or more sinks.

- The overlay routing paths connecting nodes to the sinks can be fixed a priori, (e.g. a tree or a chain) or may also be chosen as part of the optimization process.

- There might be several objective functions; the most natural ones are to minimize the maximum energy consumption over all nodes or to maximize the amount of sensed data arriving at the sinks with a given energy constraint.

- Sensor networks are usually managed in a distributed on-line way thus reflecting most sensor networks in practice.

By considering the above issues, we formulate the sensor problem in a combinatorial optimization setting, which allows us to derive, what we believe to be, the first results on worst-case analysis for on-line algorithms on wireless sensor networks, as opposed to mainly empirical current results.

We concentrate here on a basic subclass of latency constrained data aggregation problems. We assume that communication times and communication costs, in terms of energy consumption, are functions of the arcs only, modeling the situation of total aggregation, while the objective is to minimize the maximum communication cost per node over all nodes. There is only one sink and the communication paths from the nodes to the sink are unique, forming an intree with the sink as the root. The tree is a typical routing topology in sensor networks; see [4, 11, 15, 18, 19].

In spirit [4] come closest to our paper. The authors considered optimization of TCP acknowledgement (ACK) in a multicasting tree. The problem is a data aggregation problem. However, energy consumption is not an issue in this problem and latency is considered as a cost instead of a constraint, resulting in an objective of minimizing the sum of the total number of communications and the total latency of the messages.

In [5] the authors studied the optimal aggregation policy in a single-hop scenario (i.e. the graph is a star). Namely an *aggregator* performs a request and starts waiting for answers from a set of sources. The time for each source to return its data to the aggregator is independent and identically distributed according to a known distribution F. The main differences with our paper are that they assume that F is known, and they focus on a single-hop scenario.

The outline of our paper is the following. In Section 2 we formalize the problem; for a thorough understanding of the problem we have studied both the off-line and the on-line version of the problem, although the latter version is the relevant one in practice.

In Section 3 we show that the off-line problem is **NP**-hard and we give a 2-approximate algorithm. We remark that our approximate solution is based on a

new rounding technique of the LP-relaxation of an Integer Linear Programming formulation of the problem, which might be useful for other applications.

In Section 4 we describe the distributed on-line problem, both in the synchronous and the asynchronous setting. Our main results are:
(a) *Distributed synchronous.* We present a $O(\log U)$-competitive algorithm, where U is the ratio between the maximum and the minimum time that a packet can wait in its route toward the sink. We also show a lower bound of $\Omega(\log U)$ on the competitive ratio, whence the proposed algorithm is best possible up to a multiplicative constant.
(b) *Distributed asynchronous.* We give an $O(\delta \log U)$-competitive algorithm, where δ is the depth of the tree, which belongs to a class of algorithms for which we can prove a lower bound of $\Omega(\delta)$ on the competitive ratio.

In Section 5 we discuss several extensions to the basic model. We demonstrate that most of our results also hold, when minimizing the total energy consumption of all nodes, or for concave costs functions, or when aggregation is possible only if messages are released within the same region.

Finally, in Section 6 we summarize our results and suggest possibilities for future research in this rich research area.

2 The sensor problem formalized

We study sensor networks $D = (V, A)$, which are *intrees* rooted at a *sink node* $s \in V$. Nodes represent sensors and arcs represent the possibility of communication between two sensors. Given an arc $a \in A$ we denote its head and tail nodes by $\text{head}(a)$ and $\text{tail}(a)$, respectively.

Over time, n messages, $N := \{1, \ldots, n\}$, arrive at nodes and have to be sent to the sink. Message j arrives at its *release node* v_j at its *release date* r_j and must arrive at the sink via the unique path from v_j to s at or before its *due date* d_j. Thus, each message is completely defined by the triple (v_j, r_j, d_j). Unless otherwise stated we assume that messages are indexed by increasing due date, i.e., $d_1 \leq d_2 \leq \cdots \leq d_n$. We refer to $L_j := d_j - r_j$ as the *latency* of message j.

A *packet* is a set of messages which are sent simultaneously along an arc. More precisely, each initial message is sent as one packet. Recursively, two packets j and ℓ can be aggregated at a node v. The resulting packet has due date $d = \min\{d_j, d_\ell\}$. This definition naturally extends to the case of more packets aggregated together.

Communication of a message along an arc takes time and energy (cost). In this paper we assume that the communication time $\tau : A \to \mathbb{R}_{\geqslant 0}$ and communication cost $c : A \to \mathbb{R}_{\geqslant 0}$ are independent of packet size. We often refer to the *communication cost* of a node as the communication cost of its unique outgoing arc. This models the situation in which all messages have more or less the same size and where *total aggregation* is possible, as discussed in the introduction. For $v \in V$, let τ_v and c_v be, respectively, the total communication time and total communication cost on the path from v to s. For message j and node u on the path from v_j to s, we define *transit interval* $I_j(u)$ as the time interval during

which message j can transit at node u: $I_j(u) := [r_j + \tau_{v_j} - \tau_u, d_j - \tau_u]$. In particular, $I_j(s) = [r'_j, d_j]$, where $r'_j := r_j + \tau_{v_j}$ is the *earliest possible arrival time* of j at s. We abbreviate $I_j(s)$ to I_j and call it the *arrival interval* of message j. We also write $|I|$ for the length of interval I; note that $|I_j(u)| = |I_j|$ for all j and for all u on the path from v_j to u.

Finally, we define $\delta := \max_v \tau_v$ as the depth of the network in terms of the communication time. All logarithms in this paper are base 2. In this paper when we refer to $O(\log p)$ for some parameter p it is to be understood that $O(\log p) = O(1)$ if $p = 1$.

The objective of the sensor problem is to send all messages to the sink in such a way as to minimize the maximum communication cost per node, while satisfying the latency restrictions. Given that communication costs are independent of the size of packets sent, but linear in the number of packets sent, it is clearly advantageous to aggregate messages into packets at tail nodes of arcs.

3 The off-line problem

In this paragraph we give some positive and negative results on the off-line sensor problem. We prove that the problem is strongly **NP**-hard. We formulate the problem as an ILP and we design a novel rounding technique for its LP-relaxation, yielding a 2-approximation.

We start by proving some properties of optimal off-line solutions.

Lemma 1. *There exists a minimum cost solution such that:*

(i) whenever two messages are present together at the same node, they stay together until they reach the sink;
(ii) a message never waits at an intermediate node, i.e., a node different from its release node and the sink;
(iii) the time when a packet of messages arrives at the sink is the earliest due date of any message in that packet.

Proof. (i): Repeatedly apply the argument that whenever two messages are together at the same node but split up afterwards, keeping the one arriving later at the sink with the other message does not increase cost.

(ii): Use (i) and repeatedly apply the following argument. Whenever a packet of messages arrives at an intermediate node and waits there, changing the solution by shifting this waiting time to the tail node of the incoming arc does not increase cost.

(iii): Follows similarly as (ii) by interpreting the time between the arrival of a packet at the sink and earliest due date as waiting time. □

3.1 NP-hardness

Theorem 1. *The off-line sensor problem is strongly* **NP***-hard.*

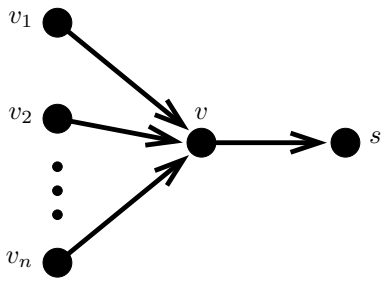

Fig. 1.

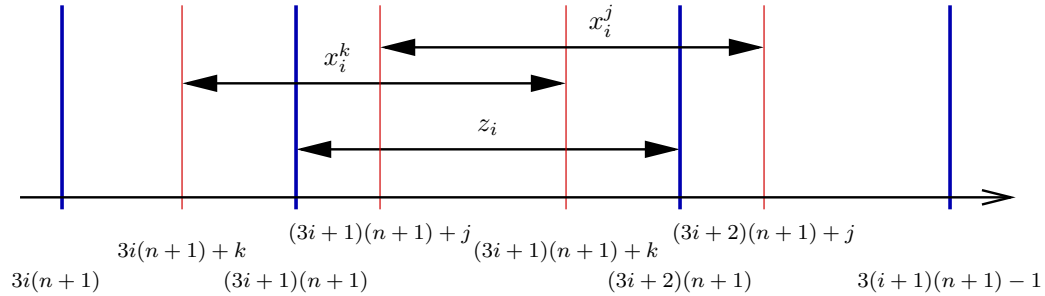

Fig. 2. Arrival intervals corresponding to clause C_i. In the depicted example, the clause has the form $C_i = (X_j \vee \neg X_k)$.

Proof. We prove the theorem using a reduction from the Satisfiability Problem. Given an instance of SAT with n boolean variables $X_1, \ldots, X_n$ and m clauses $C_1, \ldots, C_m$, we construct the intree on $n+2$ nodes depicted in Figure 1.

The nodes $v_1, \ldots, v_n$ on the left correspond to variables $X_1, \ldots, X_n$. There is one intermediate node v and the sink s on the right. The communication costs of the arcs are determined later. The communication times of all arcs are zero, whence the earliest arrival times of the messages coincide with their release dates.

For clause C_i we define a time interval $T(C_i) = [3i(n+1), 3(i+1)(n+1)-1]$ and a message $z_i = (v, r_i, d_i) := (v, (3i+1)(n+1), (3i+2)(n+1))$, $i = 1, \ldots, m$. Notice that the arrival interval $I_{z_i} = [r_i, d_i] \subset T(C_i)$. We also define two dummy messages $z_0 := (v, 0, 0)$ and $z_{m+1} := (v, 3(m+1)(n+1), 3(m+1)(n+1))$. Notice the crucial fact $|I_{z_0}| = |I_{z_{m+1}}| = 0$, leaving no choice in sending z_0 and z_{m+1}.

If variable X_j occurs unnegated in clause C_i, we create a message $x_i^j = (v_j, r_i^j, d_i^j) := (v_j, (3i+1)(n+1)+j, (3i+2)(n+1)+j)$. If X_j occurs negated in clause C_i, we create message $x_i^j := (v_j, 3i(n+1)+j, (3i+1)(n+1)+j)$. If X_j does not occur in C_i no message x_i^j is created. Notice that in both cases the arrival time interval $I_i^j \subset T(C_i)$. If X_j does not occur in C_i no message x_i^j is created. An illustration is given in Figure 2.

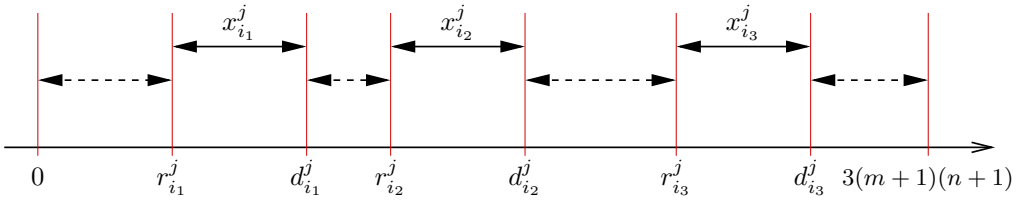

Fig. 3. Arrival intervals of messages with release node v_j. In the depicted example, variable X_j occurs in three clauses C_{i_1}, C_{i_2}, and C_{i_3}. Arrival intervals of the four auxiliary messages are represented by dashed arrows.

The rough idea behind the reduction is the following: in an optimal solution, message x^j_i is either sent at its release or at its due date (the reason for this will become clear later). Moreover, sending x^j_i at its release (due) date means setting X_j to true (false). Thus, message z_i can join message x^j_i at node v if and only if the value of variable X_j makes clause C_i true.

We continue with the description of the instance. Let $i^j_1 < \cdots < i^j_{k_j}$ denote the indices of the clauses in which variable X_j occurs. We create k_j+1 additional messages released at node v_j. The release and due dates of these messages are chosen such that the $2k_j + 1$ arrival time intervals formed by the release and due dates of all messages released at node v_j form a partition of the interval $[0, 3(m+1)(n+1)]$; see Figure 3.

We will demonstrate that for appropriately chosen cost functions any SAT problem reduces to our sensor problem. We define the cost function by $c(v_j, v) = (\max_l k_l + 1)/(k_j+1)$ for $j = 1, \ldots, n$, and $c(v, s) = (\max_l k_l + 1)/(\sum_{l=1}^n k_l + 2)$.

Claim 1. Every optimal solution to the subinstance obtained by ignoring messages $z_1, \ldots, z_m$ has the following properties:

(a) The cost of each node is $\max_l k_l + 1$;
(b) A message with release node v_j is either sent from v_j at its release date or at its due date, $j = 1, \ldots, n$;
(c) For each fixed $j = 1, \ldots, n$, either all messages x^j_i ($i = i^j_1, \ldots, i^j_{k_j}$) are sent at their release dates or all of them are sent at their due dates.

Proof of Claim 1. Let us consider a solution which minimizes the cost of nodes v_j. Since the $2k_j + 1$ arrival time intervals of messages with release node v_j form a partition of $[0, 3(m+1)(n+1)]$, at most one x^j_i-message and one of the auxiliary messages can be aggregated into a packet, which then has to be sent at the single intersection point of the two arrival time intervals. Thus the minimal number of packets that have to be sent from node v_j is k_j+1, i.e. k_j pair-packets and one packet containing a single message. Hence the minimal cost of node v_j is $c(v_j, v)(k_j + 1) = \max_l k_l + 1$ for all nodes v_j.

Each pair-packet is sent at the common release and due date of its two messages and by construction of these dates no two pair-packets emanating from different nodes can be aggregated into a single packet at node v. Thus,

there are $\sum_{l=1}^{n} k_l$ pair-packets passing v. And we have the two dummy messages z_0 and z_{m+1}, which are sent from node v at times 0 and $3(m+1)(n+1)$. Pair-packets cannot be aggregated with these dummy messages but each single-message-packet can be sent at time 0 or $3(m+1)(n+1)$ and hence it may join dummy message z_0 or z_{m+1} at node v. This gives a total of $\sum_{l=1}^{n} k_l + 2$ packets passing v. Thus, the cost of node v is $c(v,s)(\sum_{l=1}^{n} k_l + 2) = \max_l k_l + 1$. Notice that a single-message-packet contains either the first or the last auxiliary message released at node v_j. If the single-message-packet is the first auxiliary message then all pair-packets are sent on the due date of the x_i^j-message in the packet. Otherwise, all pair-packets are sent on the release date of the x_i^j-message in the packet.

Thus, we have constructed a solution which satisfies properties (a),(b) and (c). As the cost of node v_j is at least $\max_l k_l + 1$ the solution is an optimal solution. From the construction of this solution it can easily be verified that any solution which violates property (a),(b) or (c) has a node with a cost which exceeds $\max_l k_l + 1$. □

This claim suffices to prove the following claim which in its turn implies the proof of the theorem directly.

Claim 2. The sensor problem has a solution with maximum cost at most $\max_l k_l + 1$ if and only if the underlying instance of SAT is satisfiable.

Proof of Claim 2. Given a satisfying assignment for the SAT instance, a feasible solution to the sensor problem can be obtained as follows. Notice that in the construction of an optimal solution in the proof of Claim 1, for each j, there is a choice for the set of messages corresponding to X_j, to send either dummy message z_0 separately at time 0 or the dummy message z_{m+1} separately at time $3(m+1)(n+1)$. In both cases the cost of sending all messages corresponding to the variables and z_0 and z_{m+1} is $\max_l k_l + 1$ for each node. We make the choice now by sending z_{m+1} separately if X_j is true in the satisfying assignment and z_0 separately if X_j is false.

We claim that message z_i corresponding to clause C_i, $i = 1, \ldots, m$, can be aggregated at v with one of the pair-packets corresponding to a variable in the clause. Suppose that clause C_i is satisfied due to variable X_j. If X_j appears unnegated in C_i (thus X_j is true), then the pair-packet containing message x_i^j is sent at time $r_i^j := (3i+1)(n+1) + j \in [r_i, d_i]$, and message z_i can join this packet at no additional cost. Similarly, if X_j appears negated at C_i (thus X_j is false) then x_i^j is sent at $d_i^j := (3i+1)(n+1) + j \in [r_i, d_i]$. This concludes the proof of the "if" part.

It follows from Claim 1 that any feasible solution with maximum cost $\max_l k_l + 1$ yields an assignment of values to the boolean variables $X_1, \ldots, X_n$: variable X_j is set to true (false), if all messages x_i^j, $i = i_1^j, \ldots, i_{k_j}^j$, are sent at their release (due) dates. It also follows from Claim 1 that in an optimal solution message z_i, $i = 1, \ldots, m$ should not cause additional cost, therefore it must join one of the packets starting at a node v_j. Due to the construction of the instance,

this is only possible if the value of variable X_j causes clause C_i to be satisfied. This concludes the proof of the "only if" part for this objective function. □

3.2 A 2-approximation

We give an ILP-formulation of the problem, based on Lemma 1, and show that rounding the optimal solution of the LP-relaxation yields a 2-approximation algorithm. For every message-arc pair $\{i, a\}$, we introduce a binary decision variable x_{ia}, which is set to 1 if and only if arc a is used by some message j which arrives at s at time d_i. We use the notation $j_{\min}$ for the smallest index i such that $d_i \geq r'_j$; that is, $j_{\min} := \min\{i : d_i \geq r'_j\}$. We use a_j to denote the first arc on the unique path from v_j to s. The LP relaxation of the sensor problem is

$$\begin{array}{l} \min\ z \\ \text{s.t. } z \geq c(a) \sum_{i=1}^{n} x_{ia} \quad \forall a \in A, \\ \quad \sum_{i=j_{\min}}^{j} x_{ia_j} \geqslant 1 \quad \forall\ 1 \leq j \leq n, \\ \quad x_{ia} \geqslant x_{ia'} \quad \forall\ 1 \leq i \leq n\ \forall\ a, a' \in A \text{ with head}(a') = \text{tail}(a), \\ \quad x_{ia} \in \{0,1\} \quad \forall\ 1 \leq i \leq n\ \forall\ a \in A. \end{array} \tag{1}$$

The first set of constraints ensures that z is at least the communication cost of any node. The second set of constraints forces each message to leave its release node in time to reach the sink before its due date. By the third set of constraints a message does not wait at intermediate nodes.

In the following lemma we develop a tool for rounding the corresponding LP-relaxation, which is obtained by replacing the integrality constraints with non-negativity constraints $x_{ia} \geqslant 0$.

Lemma 2. *Let $\alpha_1, \ldots, \alpha_n \in \mathbb{R}_{\geqslant 0}$ and $\beta_1, \ldots, \beta_n \in \{0, 1\}$ with*

$$\sum_{i=j}^{k} \alpha_i \geqslant 1 \quad \Longrightarrow \quad \sum_{i=j}^{k} \beta_i \geqslant 1 \quad \forall 1 \leq k \leq n\ \forall 1 \leq j \leq k. \tag{2}$$

By decreasing some of the β_i's from 1 to 0, one can enforce the inequality

$$\sum_{i=1}^{n} \beta_i \leqslant 2 \sum_{i=1}^{n} \alpha_i \tag{3}$$

while maintaining property (2). Moreover, this can be done in linear time.

Proof. Consider the β_i's in order of increasing index. If $\beta_i = 1$, then round it down to 0, unless this yields a violation of (2). It is not difficult to see that this greedy algorithm can be implemented to run in linear time. It remains to be proven that inequality (3) holds for the resulting numbers $\beta_1, \ldots, \beta_n$.

For $h \in \{1, \ldots, n\}$, let $\bar{h} := \min\{i > h \mid \beta_i = 1\}$; if $\beta_i = 0$ for all $i > h$ or $h = n$, then $\bar{h} := n + 1$. Similarly, let $\underline{h} := \max\{i < h \mid \beta_i = 1\}$; if $\beta_i = 0$ for all $i < h$ or $h = 1$, then $\underline{h} := 0$. We prove the following generalization of (3):

$$\sum_{i=1}^{h} \beta_i \leqslant 2 \sum_{i=1}^{\bar{h}-1} \alpha_i \quad \forall\ 1 \leq h \leq n. \tag{4}$$

By contradiction, consider the smallest index h violating (4). Since h is chosen minimally, it must hold that $\beta_h = 1$; rounding β_h down to 0 would yield a violation of (2). In particular this would yield

$$\sum_{i=\underline{h}+1}^{\bar{h}-1} \alpha_i \geqslant 1 \tag{5}$$

while $\sum_{i=\underline{h}+1}^{\bar{h}-1} \beta_i = 0$. Notice that $\underline{h} \geqslant 1$, since, by choice of h,

$$\sum_{i=1}^{h} \beta_i > 2 \sum_{i=1}^{\bar{h}-1} \alpha_i \overset{(5)}{\geqslant} 2 \ .$$

Thus, $\beta_{\underline{h}} = \beta_h = 1$. We get a contradiction to the choice of h:

$$\sum_{i=1}^{h} \beta_i = \sum_{i=1}^{\underline{h}-1} \beta_i + 2 \overset{(4)}{\leqslant} 2 \sum_{i=1}^{\underline{h}-1} \alpha_i + 2 \overset{(5)}{\leqslant} 2 \sum_{i=1}^{\underline{h}-1} \alpha_i + 2 \sum_{i=\underline{h}+1}^{\bar{h}-1} \alpha_i \leqslant 2 \sum_{i=1}^{\bar{h}-1} \alpha_i \ .$$

The first inequality follows from (4) since $\overline{(\underline{h}-1)} = \underline{h}$. □

Theorem 2. *There is a polynomial time* 2*-approximation algorithm for the sensor problem on intree* $D = (V, A)$*.*

Proof. We round optimal (fractional) solution (x, z) of the LP relaxation of (1) to an integral solution $(\bar{x}, \bar{z})$. Consider the arcs in order of non-decreasing distance from s. For arc a with $\text{head}(a) = s$, set $\hat{x}_{ia} = 1 \ \forall \ i = 1, \ldots, n$. Modify these values to $\bar{x}_{1a}, \ldots, \bar{x}_{na}$ by applying Lemma 2 to $x_{1a}, \ldots, x_{na}$ and $\hat{x}_{1a}, \ldots, \hat{x}_{na}$.

For an arc a' with larger distance to s, take the arc a with $\text{head}(a') = \text{tail}(a)$ and set $\hat{x}_{ia'} := \bar{x}_{ia} \ \forall i = 1, \ldots, n$. We also modify these values into $\bar{x}_{1a'}, \ldots, \bar{x}_{na'}$ by applying Lemma 2 to the values $x_{1a'}, \ldots, x_{na'}$ and $\hat{x}_{1a'}, \ldots, \hat{x}_{na'}$. Premise (2) of Lemma 2 is satisfied for $x_{1a'}, \ldots, x_{na'}$ and $\hat{x}_{1a'}, \ldots, \hat{x}_{na'}$ since (2) holds for $x_{1a}, \ldots, x_{na}$ and $\bar{x}_{1a}, \ldots, \bar{x}_{na}$ and since $x_{ia'} \leq x_{ia}$.

By construction, the final solution $(\bar{x}, \bar{z})$ is feasible if we choose $\bar{z} = 2z$. □

4 The distributed on-line problem

We consider a class of *distributed on-line* models, in which nodes communicate independently of each other, while messages are released over time. Each node is equipped with an algorithm, which determines at what times the node sends its packets to the next node on the path to the sink. The input of each node's algorithm at any time t is restricted to the packets that have been released at or forwarded from that node in the period $[0, t]$.

We assume that all nodes are equipped with a clock to measure the latency of messages. We distinguish two distributed on-line models: In the *synchronous* model all nodes are equipped with a *common clock*, i.e. the times indicated at all

clocks are identical. A common clock may facilitate synchronization of actions in various nodes. In the *asynchronous* model there is no such common clock; still, the duration of the time unit is assumed to be the same for all nodes.

We also assume in both models that each node v knows its total communication time τ_v to the sink. Moreover, for the asynchronous model we assume that all communication times $\tau(a)$ are equal, and without loss of generality we set $\tau(a) = 1\ \forall a \in A$.

4.1 The synchronous model

For the synchronous model we propose an algorithm, which we prove to be best possible (up to a multiplicative constant) within the class of deterministic algorithms. The algorithm is based on the following simple lemma.

Lemma 3. *Given any interval $[a, b]$, $a \geq 0$, such that $[a, b]$ contains at least one integer. Let $i^* = \max\{i \in \mathbb{N} \mid \exists k \in \mathbb{N} : k2^i \in [a, b]\}$, then k^* for which $k^*2^{i^*} \in [a, b]$ is odd and unique.* □

Proof. Assume that $k_1 2^{i^*} \in [a, b]$ and $k_2 2^{i^*} \in [a, b]$, with $k_1 < k_2$. We may assume that $k_2 = k_1 + 1$, for if $k_2 > k_1 + 1$, then obviously also $(k_1 + 1)2^{i^*} \in [a, b]$. This means that either k_1 or k_2 is even. Suppose this is k_1 (if k_2 is even the arguments are analogous). Then, $k_1/2 \in \mathbb{N}$ and $(k_1/2)2^{i^*+1} \in [a, b]$, contradicting the definition of i^*. □

We use notation $t(I)$ to represent the unique point in the interval $I = [a, b]$ which equals $k^*2^{i^*}$ with i^* and k^* as defined in Lemma 3. Note, that we assume I to contain at least one integer.

> **Algorithm:CommonClock** (CC): Message j is sent from v_j at time $t(I_j) - \tau_{v_j}$ to arrive at s at time $t(I_j)$ unless some other packet passes v_j in the interval $[r_j, t(I_j) - \tau_{v_j}]$, in which case j is aggregated and the packet is forwarded directly.

First we derive a bound on the competitive ratio of CC for instances in which the arrival intervals I_j differ by at most a factor 2 in length.

Lemma 4. *If there exists an $i \in \mathbb{N}$ such that $2^{i-1} < |I_j| \leq 2^i$ for all messages j, then* CC *has a competitive ratio of at most* 3.

Proof. We will prove that the communication cost of each arc in the CC-solution is at most 3 times the communication cost of this arc in the optimal solution.

Assume that in an optimal solution packets arrive at s at times $t_1 < \cdots < t_\ell$. Let N_h^* be the packet arriving at t_h at s. Since $t_h \in I_j\ \forall j \in N_h^*$ and $|I_j| \leq 2^i\ \forall j$, we have $I_j \subset [t_h - 2^i, t_h + 2^i] =: I\ \forall j \in N_h^*$, and $|I| = 2 \cdot 2^i$. If $t_h = k2^i$ then in the CC-solution all messages in N_h^* may arrive at s at times t_h, $t_h - 2^i$ or $t_h + 2^i$. If $t_h \neq k2^i$ then I contains two different multiples of 2^i, say $k2^i$ and $(k+1)2^i$, such that $k2^i < t_h < (k+1)2^i$. In this case, since $|I_j| > 2^{i-1}\ \forall j$, we have $\forall j \in N_h^*$

that $I_j \cap \{k2^i, k2^i + 2^{i-1}, (k+1)2^i\} \neq \emptyset$. Lemma 3 implies that in a CC-solution every message $j \in N_h^*$ arrives at s at one of $\{k2^i, k2^i + 2^{i-1}, (k+1)2^i\}$. Hence, $\forall h = 1, \ldots, \ell$, all messages in N_h^* arrive at s at at most 3 distinct time instants in the CC-solution. CC does not delay messages at intermediate nodes. This implies that the arcs used by messages in N_h^* are traversed by these messages at most 3 times in the CC-solution, proving the lemma. □

Let $U = \frac{\max_j |I_j|}{\max\{1, \min_j |I_j|\}}$.

Theorem 3. CC *is* $\Theta(\log U)$*-competitive.*

Proof. For each $i \in \mathbb{N}$ with $\log(\max\{1, \min_j |I_j|\}) \leq i \leq \lceil \log(\max_j |I_j|) \rceil$, CC sends the messages in $N_i := \{j \in N \mid 2^{i-1} < |I_j| \leq 2^i\}$, at a cost of no more than 3 times the optimum, by Lemma 4. This proves $O(\log U)$-competitiveness if $\min_j |I_j| \geq 1$. In case $\min_j |I_j| = 0$ we observe that restricted to the class of messages $N_0 = \{j \in N \mid |I_j| = 0\}$ CC's cost equals the optimal cost, because there is no choice for these messages.

To prove $\Omega(\log U)$ consider a chain of 2^{n+1} nodes $u_1, \ldots, u_{2^{n+1}} = s$ for some $n \in \mathbb{N}$. Take $\tau(a) = 1$ and $c(a) = 1$ $\forall a$. For $j = 1, \ldots, n$, $v_j = u_{2^j}$, $r_j = 0$, and $d_j = 2^{n+1} - 1$. Hence $r'_j = 2^{n+1} - 2^j = k2^j$ for some odd $k \in \mathbb{N}$ and $|I_j| = 2^j - 1$. Therefore, CC makes each message j arrive at s at time r'_j, no two messages are aggregated, and the cost is $\sum_{j=1}^{n}(2^{n+1} - 2^j) = (n-1)2^{n+1} + 2$. In an optimal solution all messages are aggregated into a single packet arriving at s at time $2^{n+1} - 1$ at a cost of $2^{n+1} - 2$. Notice that $U = 2^n - 1$ in this case. □

The following theorem shows that CC is best possible (up to a multiplicative constant).

Theorem 4. *Any deterministic synchronous algorithm is* $\Omega(\log U)$*-competitive.*

Proof. Consider an intree of depth $\delta = 2^{n+1}$ with n the number of messages, and where each node, except the leaves, has indegree n. We assume $\tau(a) = 1$ for all $a \in A$. For any on-line algorithm we will construct an adversarial sequence of n messages all with latency $L = \delta$, such that there exists a node at which the adversary can aggregate all messages in a single packet, but at which none of them is aggregated by the on-line algorithm. Using a similar argument as in the proof of Lemma 1 (i) the fact that all messages can be aggregated in a single packet implies that there exists a solution such that every node sends at most one packet, hence the cost of the adversarial solution is 1, whereas the cost of the on-line algorithm is n.

Fix any on-line algorithm. Given an instance of the problem, let $W_j(u)$ be the time interval message j spent at node u by application of the algorithm, i.e. the waiting time interval of message j on u. We denote its length by $|W_j(u)|$. Note that $\sum_u |W_j(u)| \leq |I_j|$ for each message j. We notice that the waiting time of a message in a node can be influenced by the other messages that are present at that node or have passed that node before. Since the algorithms are distributed the waiting time of a message in a node is not influenced by any message that will pass the node in the future.

The adversary chooses the source node v_j with total communication time $\tau_{v_j} := \delta - 2^j$ from s, for $j = 1, \ldots, n$, so that $|I_j| = 2^j$. Thus, $U = 2^{n-1} = \delta/4$. The choice of the exact position of v_j and the release time r_j is made sequentially and, to facilitate the exposition, described in a backward way starting with message n. The proof follows rather directly from the following claim.

Claim. For any set of messages $\{k, \ldots, n\}$ the adversary can maintain the properties:
(i) all messages in $\{k, \ldots, n\}$ pass a path p_k with 2^k nodes;
(ii) $I_k(u) = \bigcap_{j \geq k} I_j(u)\ \forall u \in p_k$;
(iii) if $k < n$, then $W_{k+1}(u) \bigcap I_k(u) = \emptyset\ \forall u \in p_k$;
(iv) if $k < n$, then $W_i(u) \bigcap W_j(u) = \emptyset\ \forall u \in p_k$, $i = k, \ldots, n$, $j > i$.

We notice that for any message j and any node u on the path from v_j to s, $W_j(u)$ may have length 0 but is never empty; it contains at least the departure time of message j from node u.

Note that properties (i) and (ii) for $k = 1$ imply that all messages can indeed be aggregated into one packet, hence as argued above, the adversarial solution has a cost of 1. Properties (iv) and (i) for $k = 1$ imply that the on-line algorithm sends all messages separately over a common path with 2 nodes, yielding a cost of n. This proves the theorem.

We prove the claim by induction. The basis of the induction, $k = n$, is trivially verified. Suppose the claim holds for message set $\{k, \ldots, n\}$ and p_k is the path between nodes $\overline{v}$ and $\hat{v}$. We partition p_k into two sub-paths $\overline{p}$ and $\hat{p}$ consisting of 2^{k-1} nodes each, such that $\overline{v} \in \overline{p}$ and $\hat{v} \in \hat{p}$. We denote the last node of $\overline{p}$ by $\overline{u}$ and the first node of $\hat{p}$ by $\hat{u}$. We distinguish two cases with respect to the waiting times the algorithm has selected for message k in the nodes on p_k.

Case a: $\sum_{u \in \overline{p}} |W_k(u)| \geq (1/2)|I_k|$. The adversary chooses v_{k-1} with total communication time $\tau_{v_{k-1}} = \delta - 2^{k-1}$ such that its path to s traverses $\hat{p}$ but not $\overline{p}$. More precisely, we ensure that the first node message $k-1$ has in common with any other message is $\hat{u}$. This is always possible, since the node degree is n. This choice immediately makes that setting $p_{k-1} = \hat{p}$ satisfies property (i). The release time of $k-1$ is chosen so that $I_{k-1}(\hat{u})$ and $I_k(\hat{u})$ start at the same time, implying that $I_{k-1}(u)$ and $I_k(u)$ start at the same time for every $u \in \hat{p}$. Since $|I_{k-1}(u)| = |I_k(u)|/2$ we have $I_{k-1}(u) \subset I_k(u)$ for all $u \in \hat{p}$, whence property (ii) follows by induction.

Note that, as we consider distributed algorithms, message $k-1$ does not influence the waiting time of $j, j > k-1$, on $\overline{p}$ as $\hat{u}$ is the first node which both j and $k-1$ traverse. In particular, $W_k(u)\ \forall u \in \overline{p}$ is not influenced by $k-1$.

Now, the equal starting times of $I_{k-1}(\hat{u})$ and $I_k(\hat{u})$ together with $\sum_{u \in \overline{p}} |W_k(u)| \geq (1/2)|I_k|$ and $|I_{k-1}(\hat{u})| = |I_k(\hat{u})|/2$ imply that k will not reach $\hat{u}$ before interval $I_{k-1}(\hat{u})$ ends. This, together with the consideration above, implies property (iii).

To prove (iv), note that by induction it is sufficient to prove that $W_{k-1}(u) \cap W_j(u) = \emptyset\ \forall j > k-1\ \forall u \in \hat{p}$. Since, as just proved, $W_k(u) \cap I_{k-1}(u) = \emptyset\ \forall u \in \hat{p}$ we have $W_{k-1}(u) \cap W_k(u) = \emptyset\ \forall u \in \hat{p}$. We have by induction that, for $j > k$,

$W_j(u) \cap I_{j-1}(u) = \emptyset \ \forall u \in \hat{p}$ and we just proved that $I_{k-1}(u) \subset I_{j-1}(u) \subset I_j(u) \ \forall u \in \hat{p}$, which together imply $W_{k-1}(u) \cap W_j(u) = \emptyset \ \forall j > k \ \forall u \in \hat{p}$.

Case b: $\sum_{u \in \overline{p}} |W_k(u)| < (1/2)|I_k|$. As in the previous case, the adversary chooses v_{k-1} with total communication time $\tau_{v_{k-1}} = \delta - 2^{k-1}$ such that its path to s traverses $\overline{p}$ (therefore also $\hat{p}$) but does not intersect any of the paths used by messages $\{k, \ldots, n\}$ before it reaches $\overline{p}$ in $\overline{v}$. Again, this is always possible since the indegree of each node is n. Hence, choosing $p_{k-1} = \overline{p}$ satisfies property (i). The release time of $k-1$ is chosen so that $I_{k-1}(\overline{v})$ and $I_k(\overline{v})$ end at the same time, implying that $I_{k-1}(u)$ and $I_k(u)$ end at the same time for every $u \in \overline{p}$. Since $|I_{k-1}(u)| = |I_k(u)|/2$ we have $I_{k-1}(u) \subset I_k(u)$ for all $u \in \overline{p}$, whence property (ii) follows by induction.

The equal ending times of $I_{k-1}(\overline{u})$ and $I_k(\overline{u})$ together with $\sum_{u \in \overline{p}} |W_k(u)| < 1/2|I_k|$ and $|I_{k-1}(\overline{u})| = |I_k(\overline{u})|/2$ imply that k has left $\overline{u}$ before $I_{k-1}(\overline{u})$ begins, implying property (iii). Indeed, this gives $W_{k-1}(u) \cap W_k(u) = \emptyset \ \forall u \in \overline{p}$. It also implies that $k-1$ could not influence the waiting time of k on $\overline{p}$.

The proof of (iv) follows the very same lines as in Case a, with the difference that we now refer to nodes in $\overline{p}$ instead of $\hat{p}$. □

Since in the proof $U = \delta/4$ we also have the following lower bound on the competitive ratio of any deterministic synchronous algorithm.

Corollary 1. *Any deterministic synchronous algorithm is $\Omega(\log \delta)$-competitive.* □

4.2 The asynchronous model

In this paragraph we consider deterministic algorithms for the asynchronous model. We propose a deterministic algorithm and analyze its competitive ratio. We also provide a lower bound on the competitive ratio for a broad class of algorithms including this algorithm.

In the asynchronous model nodes are equipped with a clock and a distributed algorithm. All clocks have the same time unit, but neither the time nor the start of a new time unit on clocks is synchronized. We assume that $\tau(a) = 1$ for all a, such that τ_{v_j} is equal to the number of nodes on the path from v_j to s.

We propose algorithm Spread Latency (SL) for this model, which divides the latency minus communication time of each message j equally over the nodes on the path from v_j to s: at each node of this path the message is assigned a waiting time of $(L_j - \tau_{v_j})/\tau_{v_j}$ time units. As soon as messages appear simultaneously at the same node they get aggregated into a packet, which is sent over the outgoing arc as soon as the waiting time of at least one of its messages at that node has passed. In this way, no message is delayed due to aggregation and thus the algorithm yields a feasible solution.

As in the previous subsection, let $U := \frac{\max_j |I_j|}{\max\{1, \min_j |I_j|\}} = \frac{\max_j (L_j - \tau_{v_j})}{\max\{1, \min_j (L_j - \tau_{v_j})\}}$.

Theorem 5. *The algorithm* SL *is $O(\delta \log U)$-competitive.*

Proof. We prove that for all $a \in A$ the number of packets SL sends through a is at most $O(\delta \log U)$ times that number in an optimal solution. This proves the theorem.

Let $\lambda := \max\{1, \min_j(L_j - \tau_{v_j})\}$. Consider a packet P of messages sent by an optimal solution through (u, v) at t. Without loss of generality we do not consider messages for which $\min_j(L_j - \tau_{v_j}) = 0$ as these messages have to be sent upon release by both SL and the optimal algorithm. To bound the number of packets sent by SL that contain at least one message from P, define $P_k := \{j \in P \mid 2^{k-1}\lambda \leq L_j - \tau_{v_j} < 2^k\lambda\}$, for $k = 1, \ldots, \lceil \log U \rceil$. We *charge* any sent packet to the message that caused the packet to be sent due to its waiting time being over. It suffices to prove that the number of packets charged to messages in P_k is $O(\delta)$.

Since the waiting time of messages $j \in P_k$ at node u is at least $2^{k-1}\lambda/\delta$, the delay between any two packets that are charged to messages in P_k is at least $2^{k-1}\lambda/\delta$. Since the optimal solution sends packet P at time t through arc (u, v), we get $t \in I_j(u)\ \forall j \in P$ and thus $I_j(u) \subseteq [t - 2^k\lambda, t + 2^k\lambda]\ \forall j \in P_k$. Thus, the number of packets charged to messages in P_k is at most $2 \cdot 2^k\lambda/(2^{k-1}\lambda/\delta) = 4\delta$. □

The competitive ratio of SL can be $\Omega(\delta \log \delta)$-competitive. Consider the chain $u_m, \ldots, u_1, s$ with unit transit times; i.e. $m = \delta$. We assume $\delta > 4$. An adversary releases messages $j_{i,k}$ at node u_m for $i = 1, \ldots, \delta/4$ and $k = 0, \ldots, \log_2 \delta - 1$. The release time of message $j_{i,k}$ is $r(i, k) = \delta + \frac{2^{k+1}}{\delta} i - 2^k$ and the latency is $L(i, k) = 2^k + \delta$. It follows from the observations $r(i, k) < r(i + 1, k)$ and $r(\delta/4, k) < r(1, k - 1)$ that messages are released ordered by decreasing values of k and then by increasing values of i. This induces the total order $\prec$ on pairs (i, k) and (i', k'). Formally, $(i, k) \prec (i', k')$ if either $k > k'$ or $k = k'$ and $i \leq i'$.

We have $I_{j_{i,k}} = [r(i, k) + \delta, r(i, k) + L(i, k)]$. As $r(i, k) + \delta \leq 2\delta$ and $r(i, k) + L(i, k) \geq 2\delta$ for each message $j_{i,k}$, we have $\bigcap_{j \in j_{i,k}} I_j \neq \emptyset$, hence the adversary may aggregate all messages at their common release node u_m. Let $W_j(u)$ be, as defined before, the waiting time interval of message j on u determined by SL. We have $W_{j_{i,k}}(u_m) = [r(i, k), r(i, k) + \frac{L(i,k)-\delta}{\delta}]$. It follows from simple arithmetics that $r(i, k) + \frac{L(i,k)-\delta}{\delta} < r(i', k')$ for all pairs $(i, k), (i', k')$ such that $(i, k) \prec (i', k')$. Hence, $W_j(u_m) \cap W_{j'}(u_m) = \emptyset$ for any two messages j and j' and SL sends all messages separately from u_m. This implies that SL is $\Omega(\delta \log \delta)$-competitive for the sensor problem with unit costs.

SL determines the waiting time of each message at the nodes it traverses independently of all other messages. We call such an algorithm a *memoryless* algorithm. To be precise, in a memoryless algorithm node v determines the waiting time of message j based only on the message characteristics (v_j, r_j, d_j), communication time to the sink τ_v and clock time. The following lower bound shows that the competitive ratio of SL cannot be beaten by more than a factor $O(\log U)$ by any other memoryless algorithm. In the derivation of the lower bound we restrict to memoryless algorithms that employ the same algorithm in all nodes with the same communication time to s. This is not a severe restriction,

given that communication time to s is the only information about the network that a node has.

Theorem 6. *Any deterministic asynchronous memoryless algorithm is $\Omega(\delta)$-competitive.*

Proof. Consider a binary intree with root s and all leaves at distance δ from s. An adversary releases message 1 with latency L at time r_1 in a leaf v_1. There must be a node v where message 1 waits at most $(L - \tau_{v_1})/\delta$. The adversary releases message $j, j = 2, \ldots, \delta$, at a leaf v_j at time $r_1 + j(L - \tau_{v_1})/\delta$ such that all messages j are sent over node v, and no two messages can be aggregated before reaching v. Because $\tau_{v_j} = \tau_{v_1}\ \forall j$ and we assumed that any memoryless algorithm applies the same algorithm in nodes with the same communication time to the sink, all messages are sent non-aggregated to and from v, whereas they are aggregated as early as possible in an optimal solution, in particular at v. □

The lower bound does not hold for arbitrary algorithms as a node may adjust the waiting time of subsequent messages that traverse that node. However, we notice that only if a node delays subsequent messages longer the competitive ratio might be better. If the waiting time at a node does not increase for subsequent messages, the competitive ratio remains $\Omega(\delta^{1-\epsilon})$. The following theorem shows that the lower bound remains $\Omega(\delta)$ if release nodes do not delay subsequent messages longer than preceding messages.

Theorem 7. *Any asynchronous memoryless algorithm for which the waiting time of message j at its release node is at most $\frac{L-\tau_{v_j}}{K}$ is $\Omega(K)$-competitive.*

Proof. Consider a chain which consists of two nodes v and s. We assume constant latency L for each message. The adversary releases $K - 1$ messages with an interval of $(L - \tau_{v_j})/(K - 1)$ at v. Since the waiting time of message j at v is at most $(L - \tau_{v_j})/K$, none of these messages are aggregated in the on-line solution, whereas they are all aggregated in one packet in an optimal solution. □

The theorems prove that SL is $\Omega(\delta)$-competitive. For arbitrary asynchronous algorithms we do not have any better lower bound than the one in Theorem 4. Furthermore, notice that for any memoryless algorithm to have a competitive ratio better than some constant times the number of messages, it should delay messages at their release node.

Improved algorithm for the case of a chain. We describe an algorithm with improved competitive ratio for the case that the network is a chain with s in one of its ends, Line-SL. We first introduce a classification of the nodes on the chain. Initially, all nodes are unclassified. Let $2^p|n := \max\{p \in \mathbb{N}, \exists k \in \mathbb{N} \text{ s.t. } n = k2^p\}$. For $p = \lfloor \log \delta \rfloor, \ldots, 0$ we assign all unclassified nodes v with $2^p|\tau_v$ to class p. Notice that the resulting classification of nodes has the property that between any pair of nodes of the same class there is at least one node of higher class.

We describe the algorithm Line-SL. For a message j which is released at node v_j we set $w_j := (L_j - \tau_{v_j})/(\lfloor \log \tau_{v_j} \rfloor + 1)$. The rough idea of the algorithm is as follows: When message j reaches a node v of class p and j has not visited a node of higher class yet, it waits at node v for w_j time units. Otherwise, if j has already visited a node of class larger than p, it does not wait at node v at all. Since between any pair of nodes of the same class there is at least one node of higher class, a message will wait at most once at a node of class p for each $p = 0, \ldots, \lfloor \log \delta \rfloor$. Moreover, the highest class of a node message j will find on its path from v_j to the sink is $\lfloor \log \tau_{v_j} \rfloor$. Thus, the overall waiting time of message j accumulates to no more than

$$w_j(\lfloor \log \tau_{v_j} \rfloor + 1) = L_j - \tau_{v_j}.$$

Therefore the message arrives at the sink at or before time $r_j + \tau_{v_j} + L_j - \tau_{v_j} = d_j$.

As in the SL algorithm, messages are aggregated into a packet if they appear simultaneously at an intermediate node. This packet is sent over the outgoing arc as soon as the waiting time of at least one of its messages at that node has passed (in particular, if a packet contains a message that does not wait at a particular node, then the packet does not wait at that node). Notice that no message is delayed due to aggregation and therefore the algorithm yields a feasible solution.

Theorem 8. *Line-*SL *is $O(\log^3 \delta)$-competitive.*

Proof. We prove that the number of packets sent through an arc by Line-SL is at most $O(\log^3 \delta)$ times the number of packets sent by an optimal solution through this arc.

Let $\lambda := \max\{1, \min_j (L_j - \tau_{v_j})\}$. Consider a packet P of messages sent by an optimal solution through arc $a \in A$ at time t. We derive a bound on the number of packets sent by Line-SL that contain at least one message from P.

As above, we define $P_k := \{j \in P \mid 2^{k-1}\lambda \leq L_j - \tau_{v_j} < 2^k \lambda\}$ for $k = 1, \ldots, \lceil \log U \rceil$. Moreover, for $p = 0, \ldots, \lfloor \log \delta \rfloor$, let $P_{k,p}$ denote the subset of P_k that consists of all messages j that have visited a node of class p but no node of class $p+1$ before being sent through arc a. Notice that the number of subsets $P_{k,p}$ is $O(\log \delta \log U) = O(\log^2 \delta)$.

We charge any packet sent by Line-SL to one of its messages $j \in P_{k,p}$ where k and p are chosen maximally. The waiting time of a message $j \in P_{k,p}$ at a node of class p is at least $2^{k-1}\lambda/(\log \delta + 1)$. On the other hand, messages in $P_{k,p}$ can only pass this node of class p within a time interval of $2 \cdot 2^k \lambda$ (see proof of Theorem 5). Thus, the number of packets charged to messages in $P_{k,p}$ is at most $4(\log \delta + 1) \in O(\log \delta)$. Since the number of subsets $P_{k,p}$ is $O(\log^2 \delta)$, the result follows. □

5 Variations and generalizations

We consider some variations and generalizations to the sensor model studied in the previous sections. A first variation is obtained by changing the objective

to minimizing the sum of energy consumption. This objective is common in networks where nodes have access to a replenishable energy source. In a technical report of this paper [2] we show that most of our positive results (upper bounds) in this paper also hold under this min-sum objective. In particular the distributed algorithms in Section 4 can be applied and their competitive ratios are equal to those given in Theorems 3 and 5. The lower bounds do not hold. Hence, there are bigger gaps between lower and upper bounds in the competitive analysis for this objective.

For the off-line version of the problem, a minor adaption makes the **NP**-hardness proof hold also for the min-sum objective. Also, in [2] we have devised a dynamic programming algorithm that works in polynomial time for the problem on a chain. In this case the problem is equivalent to a batch processing problem, for which Finke et al. [9] developed the same dynamic programming algorithm independently of us. It is an open question if a polynomial time algorithm exists for the min-max objective on a chain.

As a second variation we consider generalizations of the assumption of total aggregation. Under total aggregation any two messages can be aggregated into a single packet regardless of their release times and release nodes. In practice total aggregation is not always possible; whether messages can be aggregated depends on the data. We consider two generalizations which model these limitations: concave cost functions, and geographically bounded aggregation.

In the *concave cost function model* the communication cost of a packet is a non-decreasing concave function in data size. The reason to choose a concave cost function is that typically in short-range communication the costs are determined by a (significant) start-up cost, and a communication cost which is linear in the data size [12, 20]. Even in the case that aggregation is no more than appending messages to a packet, a concave cost function is a very natural cost function. In case aggregation results in a packet with data size less than the sum of the original packets, an aggregation function models to what extend packets can be aggregated; such a function is typically a concave non-decreasing function [11]. Thus a concave cost function reflects both the economies of scale of sending an aggregate packet, and the gain obtained by data compression. Most results of the previous sections generalize to concave non-decreasing cost functions. First, since the constant cost function of the total aggregation model is both concave and non-decreasing, all lower bounds derived in this paper hold for any such cost function. Thus, we focus on the positive results. Consider the same algorithms, CC for the synchronous model and SL for the asynchronous model. The proofs of Theorem 3 and Theorem 5 are both based on bounding the number of packets the algorithm sends for each packet that the optimal solution sends. It follows from a straightforward analysis that these proofs generalize to cost functions which are both concave and non-decreasing.

In practice, the aggregation of packets can be subject to geographical constraints. In particular, it may be unfeasible to aggregate two messages originating

from sensors that are too far apart. In order to model this kind of constraints we introduce the *geographically bounded total aggregation model*, which is defined as follows. Messages i and j can be aggregated into a single packet if both i and j can reach a common ancestor node in time at most ρ; i.e., there is a node v on the intersection of the path from v_i to s and the path from v_j to s, such that $\tau_{v_i} - \tau_v \leq \rho$ and $\tau_{v_j} - \tau_v \leq \rho$. Otherwise, the messages can not be aggregated. If two messages can be aggregated they can be totally aggregated, i.e., the cost of a packet is independent of the number of messages it contains. The total aggregation model studied before is a special case, with $\rho = \delta$.

For the synchronous model we propose the CC algorithm again. We briefly discuss the proof of the competitive ratio of CC and we discuss how the lower bound proof can be adapted.

CC is $O(\log U)$-competitive for any choice of ρ, because Lemma 4 remains valid; we only have to adapt to the fact that multiple packets may arrive at the sink, at a single time. Also, Theorem 4, which gives a lower bound of $\Omega(\log U)$ on any deterministic synchronous algorithm, remains valid. The class of instances on which the lower bound is based becomes more restricted. In the proof of Theorem 4 the instances had to satisfy $U = O(\delta)$, now they have to satisfy $U = O(\rho)$. As a result Corollary 1 becomes:

Corollary 2. *Any deterministic synchronous algorithm is $\Omega(\log \rho)$-competitive for the sensor problem.* □

For the asynchronous model we present the Geographic Spread Latency (GSL) algorithm: For each message j assign to this message a waiting time of $(L_j - \tau_{v_j})/\rho$ time units at the first ρ nodes it traverses. As soon as messages appear simultaneously at the same node, and they can be aggregated, they get aggregated into a packet, which is sent over the outgoing arc as soon as the waiting time of at least one of its messages at that node has passed. The algorithm is an adaptation of the SL-algorithm, and the algorithms are identical if $\rho = \delta$. A straightforward adaptation of the proofs of Theorems 5 and 6 gives the following result:

Corollary 3. *The algorithm* GSL *is $O(\rho \log U)$-competitive, and any deterministic asynchronous memoryless algorithm is $\Omega(\rho)$-competitive.*

6 Conclusions

The results we presented in this paper are the first results on a rich class of problems which are both theoretically and practically interesting. We described the great variety of problems in this class in the introduction.

For the off-line problem we proved that the problem is **NP**-hard on a tree; we presented a 2-approximation, but we have no lower bounds on the approximability. It would be interesting to see if there exists an approximation preserving reduction from an **APX**-hard problem.

For the synchronous model we presented an $O(\log U)$-competitive algorithm and we showed that this algorithm is best possible up to a multiplicative constant. For the asynchronous model we presented an $O(\delta \log U)$ -competitive algorithm. There remains a gap between this bound and the lower bound of $\Omega(\log U)$ from the synchronous model. Theorem 6 shows that improvements in the upper bound should come from algorithms that are essentially different from the one we presented here. We also demonstrated that most of our results can be generalized to models with a concave cost function, or where aggregation is regionally bounded.

ACKNOWLEDGMENTS. Supported by EU Integrated Project AEOLUS (FET-15964), EU project ADONET (MRTN-CT-2003-504438), EU COST-action 293, Dutch project BRICKS, DFG Focus Program 1126, "Algorithmic Aspects of Large and Complex Networks", grant SK 58/5-3, MIUR-FIRB Israel-Italy project RBIN047MH9.

References

1. I. Akyildiz, W. Su, Y. Sanakarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks Journal*, 38(4):393–422, 2002.
2. L. Becchetti, P. Korteweg, A. Marchetti-Spaccamela, M. Skutella, L. Stougie, and A. Vitaletti. *Latency Constrained Aggregation in Sensor Networks*. SPOR-report 2006-08, TU Eindhoven, www.win.tue.nl/bs/spor, 2006.
3. A. Boulis, S. Ganeriwal, and M. B. Srivastava. Aggregation in sensor networks: an energy - accuracy tradeoff. *Ad-hoc Networks Journal*, 1(2-3):317–331, 2003.
4. C. Brito, E. Koutsoupias, and S. Vaya. Competitive analysis of organization networks or multicast acknowledgement: how much to wait? In *"Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 627–635, 2004.
5. A. Broder and M. Mitzenmacher. Optimal plans for aggregation. In *Proceedings of the 21st Annual Symposium on Principles of Distributed Computing (PODC)*, pages 144–152, 2002.
6. J. Elson and D. Estrin. Time synchronization for wireless sensor networks. In *Proceedings of the 2001 International Parallel and Distributed Processing Symposium (IPDPS), Workshop on Parallel and Distributed Computing Issues in Wireless and Mobile Computing*, pages 1965–1970, 2001.
7. J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the 5th ACM Symposium on Operating System Design and Implementation (OSDI)*, pages 147–164, 2002.
8. J. Elson and K. Römer. Wireless sensor networks: a new regime for time synchronization. *Computer Communication Review*, 33(1):149–154, 2003.
9. G. Finke, V. Jost, M. Queyranne, and A. Sebö. Batch processing with interval graph compatibilities between tasks. In *Les Cahiers du Laboratoire, 118*, Leibniz-IMAG, 2004.
10. S. Ganeriwal, R. Kumar, and M. Srivastava. Timing-sync protocol for sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys)*, pages 138–149, 2003.
11. A. Goel and D. Estrin. Simultaneous optimization for concave costs: single sink aggregation or single source buy-at-bulk. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 499–505, 2003.

12. W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy efficient communication protocols for wireless microsensor networks. In *Proceedings of Hawaiian International Conference on Systems Science*, pages 3005–3014, 2000.
13. F. Hu, X. Cao, and C. May. Optimized scheduling for data aggregation in wireless sensor networks. In *International Conference on Information Technology Coding and Computing (ITCC)*, pages 557–561, 2005.
14. C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS)*, pages 414–458, 2002.
15. C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Mobile Computing and Networking*, pages 56–67, 2000.
16. K. Kalpakis, K. Dasgupta, and P. Namjoshi. Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks. *Computer Networks*, 42(6):697–716, 2003.
17. S. Lindsey and C. S. Raghavendra. Pegasis: Power-efficient gathering in sensor information systems. In *Proceedings of IEEE Aerospace Conference*, pages 1125–1130, 2000.
18. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. In *Proceedings of the 5th ACM Symposium on Operating System Design and Implementation (OSDI)*, pages 131–146, 2002.
19. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems (TODS)*, 30(1):122–173, 2005.
20. G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, 2000.
21. W. Yuan, V. S. Krishnamurthy, and S. K. Tripathi. Synchronization of multiple levels of data fusion in wireless sensor networks. In *Proceedings of IEEE Globecom*, pages 221–225, 2003.