Data Aggregation in Sensor Networks: Balancing Communication and Delay Costs

Peter Korteweg[1,⋆], Alberto Marchetti-Spaccamela[2], Leen Stougie[1,3], and Andrea Vitaletti[2]

[1] TU Eindhoven
Fax: + 31 40 246 5995
p.korteweg@tue.nl
[2] University of Rome "La Sapienza"
[3] CWI Amsterdam

Abstract. In a sensor network the sensors, or nodes, obtain data and have to communicate these data to a central node. Because sensors are battery powered they are highly energy constrained. Data aggregation can be used to combine data of several sensors into a single message, thus reducing sensor communication costs at the expense of message delays. Thus, the main problem of data aggregation is to balance the communication and delay costs.

In this paper we study the data aggregation problem as a bicriteria optimization problem; the objectives we consider are to minimize maximum energy consumption of a sensor and a function of the maximum latency costs of a message. We consider distributed algorithms under an asynchronous time model, and under an almost synchronous time model, where sensor clocks are synchronized up to a small drift. We use competitive analysis to assess the quality of the algorithms.

Keywords: distributed algorithms, sensor networks, data aggregation, bicriteria optimization.

1 Introduction

A wireless sensor network (WSN) 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 and, therefore, the key challenge in a sensor network is the reduction of energy

⋆ Corresponding author: TU Eindhoven, PO Box 513, 5600 MB Eindhoven, The Netherlands.

G. Prencipe and S. Zaks (Eds.): SIROCCO 2007, LNCS 4474, pp. 139–150, 2007.

consumption. 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 [7]. 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. In this paper we comply with most of the literature on sensor networks concentrating on *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. Total aggregation is possible if data are completely correlated, or can be described by a single value, e.g. when the required data is maximum temperature. Observe that even if total aggregation might be considered a simplistic assumption in other cases, it allows us to provide an upper bound on the expected benefits of data aggregation in terms of power consumption.

WSN deal with real world environments. In many cases, sensor data must be delivered within time constraints so that appropriate observations can be made or actions taken [11]. We assume that the routing network is a tree; this is a common assumption in data aggregation network problems [1,4].

In [3] we studied the Data Aggregation Sensor Problem as a unicriterion problem where we minimized the maximum communication costs subject to a budget on the latency costs. Here the budget constraint was a hard constraint.

The dynamics governing the monitored phenomena are often not well understood and/or defined at the beginning of the monitoring process. For this reason a strict constraint on latency could be unappropriate.

A common assumption in literature on data aggregation is that value of information degrades over time. E.g. Broder and Mitzenmacher [5] describe a data aggregation model where there is a reward function on the data collected by a server; the function increases with the quantity of data collected and decreases over time. A similar tradeoff holds for data aggregation in sensor networks: delaying data decreases the information value of the data, but increases network lifetime.

Both the above discussed tradeoffs and the partial knowledge of the monitored process at the beginning, suggest to use a bicriteria objective function to asses the quality of the algorithms instead of hard constraints.

The bicriteria data aggregation sensor problem

The Data Aggregation Sensor Problem (DASP) is to send all messages to the sink such as to minimize the communication costs, and to minimize the latency costs. For the first objective we have chosen to minimize the maximum communication costs per node. This is a natural objective in sensor networks because of limited and unreplenishable energy at nodes. The objective maximizes the network lifetime, i.e. the time that all sensors can communicate. For the second objective we have chosen to minimize the maximum latency cost.

The two objectives conflict with each other. We can easily find algorithms with low communication costs by delaying messages and aggregating them into

packets. As communication costs are independent of the size of packets sent, but linear in the number of packets sent, aggregation reduces the communication costs, at the expense of increased latency costs. Similarly we can find algorithms with low latency costs at the expense of high communication costs. The objective is to find algorithms where both costs are relatively good.

We formulate the problem as a bicriteria optimization problem: minimizing one of the objectives under a budget restriction on the other objective. We call a bicriteria optimization problem an (B, A)-bicriteria problem if we minimize objective A under a budget on objective B. In this paper we study the (B, A)-sensor problem where objective A is maximum communication costs and objective B is maximum latency costs. Quality of algorithms is assessed through the concept of (β, α)-approximation: allowing an excess of multiplicative factor β on the budget of objective B, the value produced is worst-case within ratio α from optimal with respect to objective A. For network design problems this was formalized in [9,10]. The concept is general in the sense that results hold regardless which of the two objectives is minimized, and which is budgeted.

Sensor nodes are equipped with a clock that can be used to measure the latency of messages. We distinguish three distributed on-line models, which are common in literature on distributed algorithms, see [12]. 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. In the *almost synchronous* model, all nodes are equipped with a clock and the clocks are almost synchronous, i.e. there is a relatively small drift between any two clocks. In practice, these clocks can easily drift seconds per day, accumulating significant errors over time [12].

Results

In this paper we present distributed on-line algorithms for sensor networks with a routing intree. The first main contribution is that we study for the first time sensor network problems in a bicriteria optimization framework. In Section 2 we formalize the model.

In Section 3, for the asynchronous model we present an algorithm which balances communication and latency costs. If δ is tree depth, and U is the ratio between maximum and minimum allowed delay, then the algorithm is $(2\delta^{\lambda}, 2\delta^{1-\lambda} \log U)$-competitive, for any λ, $0 < \lambda \leq 1$. The algorithm is member of a class of memoryless algorithms for which we show that no better competitiveness than $(\delta^{\lambda}, \delta^{1-\lambda})$ exists.

In Section 4 we present the second main contribution, which is the analysis of algorithms for sensor networks in which clocks in various nodes show small drifts. For this so-called almost synchronous model we present an algorithm which for sensors with a clock drift of at most Δ between any two nodes and latency budget L is $(1 + \Delta\delta/L, \log^2 \delta)$-competitive. For small drift, i.e. $\Delta\delta/L$ small, the competitiveness comes close to the best possible competitiveness in the synchronous model. We notice that no previous results are known for this model, which is in fact the more realistic one.

Related work
In [3] we studied the Data Aggregation Sensor Problem as a unicriterion problem where we minimized the maximum communication costs subject to a budget on the latency costs. Here the budget constraint was a hard constraint. Interpreted in the bicriteria setting the results imply $(1, O(\log U))$ for synchronous, and $(1, \delta \log U)$ for the asynchronous models. No results were given for the almost synchronous model.

In the past, many bicriteria optimization problems were formulated as a unicriterion optimization problem with as single objective a weighted sum of the two objectives. For aggregation problems with objectives to minimize communication costs and latency costs such a formulation as a unicriterion optimization problem can be found in [2,4,6,8].

Both Khanna et al. [8] and Brito et al. [4] consider the Multicast Aggregation Problem (MAP), or TCP Acknowledgment problem, on a tree. The Multicast Aggregation Problem is equivalent to the Data Aggregation Sensor Problem in the sense that messages, which arrive over time, have to be sent to a sink in the graph. The main difference with our problem is in the objectives. First, the objective of MAP is to minimize the sum of communication costs; this is a natural objective if nodes have permanent access to energy. This is not true for sensor networks, for which minimizing energy cost per node is more suitable. The other objective of MAP is to minimize the sum of latency costs, and latency costs do not depend on communication time to the sink. Second, the authors analyze the problem using a single objective which is a weighted sum of communication costs and latency costs.

A main drawback of formulating the problem using a single objective is that the choice of the weights influences the outcome. Especially if the objectives are measured in different units, e.g. energy and time, then the choice of weights is highly arbitrary. Thus, we believe that a bicriteria setting is more appropriate in this case.

2 Preliminaries

We study sensor networks $G = (V, A)$, which are *intrees* rooted at a *sink node* $s \in V$. Nodes represent sensors and arcs represent the possibility of communication between two sensors. 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; message j arrives at the sink via the unique $v_j - s$-path. Thus, each message is completely defined by the pair (v_j, r_j).

A *packet* is a set of messages which are sent simultaneously along an arc. Each initial message is a packet and two packets j and j' can be aggregated at a node v into a single packet. The resulting packet can be recursively aggregated with other packets.

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 the sake of simplicity we also assume that all communication times $\tau(a)$ are equal, namely we set $\tau(a) = 1\ \forall a \in A$.

For $v \in V$, τ_v is the total communication time of the path from v to s. We define $r'_j := r_j + \tau_{v_j}$ as *earliest possible arrival time* of j at s. We assume that each node v knows its total communication time τ_v to the sink. Finally, we define $\delta := \max_v \tau_v$ as the depth of the network in terms of the communication time. We assume $\delta \geq 2$, avoiding the trivial case of $\delta = 1$.

The value of information degrades over time. To model this we define the quality degradation cost of a message. Let d_j be the arrival time of message j. We assume that the quality degradation of a message j depends on the latency of a message $l_j := d_j - r_j$. In this paper we choose the latency as our quality degradation function, i.e. our function increases linearly over time. We also refer to these costs as *latency costs* and we say that a solution is L-bounded if $l_j \leq L$ for all i. Since $\delta = \max_v \tau_v$ a L-bounded feasible solution must satisfy $L \geq \delta$, as otherwise it is impossible to send all messages j to the sink such that their latency costs are within budget L.

The budget on the latency imposes an arrival time interval $I_j := [r_j + \tau_{v_j}, r_j + L]$ of any L-bounded solution. It also imposes a transit interval for each node u on the $v_j - s$ path: $I_j(u) := [r_j + \tau_{v_j} - \tau_u, r_j + L - \tau_u]$. I.e. in each L-bounded solution message j should transit at u in interval $I_j(u)$. Finally, we define $U = \frac{\max_j |I_j|}{\max\{1, \min_j |I_j|\}}$. Since $L \geq \delta$ we have $U \leq \delta$.

Given a solution S the communication cost of node v_i is the total energy cost spent by v_i and it is given by the total number of messages sent by v_i times the communication cost of v_i. We are interested minimizing maximum communication cost over all nodes.

Given a bound L on the latency and β, $\beta \geq 1$, we study the communication cost of algorithms that provide βL-bounded feasible solution: a βL-bounded feasible solution is (β, α)-approximate if its communication cost is at most α times the communication cost of the optimal L-bounded solution. An interesting special case is to find a minimum γ such that there exists a (γ, γ)-approximate algorithm [9].

In this paper we consider *distributed on-line* algorithms, in which nodes communicate independently of each other and messages are released over time. Therefore, at any time t the input of each node's algorithm is given by packets that have been released at or forwarded from that node in the period $[0, t]$. An algorithm is (β, α)-competitive if it is an (β, α)-approximation and the algorithm is an online algorithm.

2.1 The Synchronous Model

For the synchronous model we presented an algorithm for the latency constrained sensor aggregation problem in [3]. In the following we restate the algorithm

in a bicriteria setting, because we use the algorithm as a subroutine in our algorithm for the almost synchronous model. The algorithm is based on the following lemma.

Lemma 1. *[3] Given any interval $[a, b]$, such that $b - a \geq 1$. 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.*

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 1. The algorithm sends messages j to the sink at time $t(I)$ where interval I depends on message j and budget L on the latency costs. We choose as interval the interval of an L-bounded solution, i.e. I_j.

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.

The analysis of the competitive ratio of CC is based on the following lemma that will be used in the sequel. The lemma bounds the competitive ratio for instances in which the arrival intervals I_j differ by a factor at most 2 in length.

Lemma 2. *[3]* CC *is $(1, 3)$-competitive if there exists an $i \in \mathbb{N}$ such that $2^{i-1} < |I_j| \leq 2^i \ \forall j$.*

This result immediately implies the following theorem.

Theorem 1. *[3]* CC *is $(1, O(\log U))$-competitive*[1].

In the the CC algorithm no message incurs a delay cost which exceeds its budget. A simple modification of the CC algorithm which balances the communication and delay costs can be obtained by replacing $t(I_j)$ by $t(I_j^*)$ as follows. Let $\mu := \max\{1, \min_j(L_j - \tau_{v_j})\}$, and let $N_m = \{j \in N | (\frac{\log U}{\log \log U})^{m-1}\mu \leq |I_j| < (\frac{\log U}{\log \log U})^m \mu\}$ for $m \in \mathbb{N}$. The algorithm sends messages $j \in N_m$ to the sink at time $t(I_j^*)$ where $I_j^* = [r_j + \tau_{v_j}, r_j + \tau_{v_j} + (\frac{\log U}{\log \log U})^m \mu]$.

The proof of the following theorem is omitted.

Theorem 2. *There exists an algorithm that is $(\frac{\log U}{\log \log U}, \frac{\log U}{\log \log U})$-competitive.*

3 The Asynchronous Model

For the asynchronous model we present a modification of the algorithm Spread Latency (SL), as proposed in [3]. The algorithm assigns to message j a total waiting time of $2(\tau_{v_j})^\lambda$ times the allowed latency minus communication time, for some λ, $0 < \lambda \leq 1$. SL equally divides this waiting time over the nodes:

[1] All logarithms in this paper are base 2.

at each node of the $v_j - s$ path message j is assigned a waiting time of $2(L - \tau_{v_j})/(\tau_{v_j})^{1-\lambda}$ time units. When messages are 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 these messages has passed.

Theorem 3. *Algorithm* SL *is* $(2\delta^\lambda, 2\delta^{1-\lambda}\log U)$*-competitive for* λ, $0 < \lambda \leq 1$.

Proof. Consider algorithm SL for fixed λ, $0 < \lambda \leq 1$. First note that because no message is delayed due to aggregation the latency of each message j is at most

$$\tau_{v_j} 2(L - \tau_{v_j})/\tau_{v_j}^{1-\lambda} + \tau_{v_j} \leq 2\delta^\lambda L.$$

We prove that for all $a \in A$ the number of packets SL sends through a is at most $2\delta^{1-\lambda}\log U$ times that number in an optimal L-bounded solution. This proves the theorem.

Let $\mu := \max\{1, \min_j(L - \tau_{v_j})\}$. Consider a packet P of messages sent by an optimal L-bounded solution through (u, v) at t. To bound the number of packets sent by SL that contain at least one message from P, define $P_i := \{j \in P \mid 2^{i-1}\mu \leq L - \tau_{v_j} < 2^i\mu\}$, for $i = 1, \ldots, \lceil \log U \rceil$. 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_i is $2\delta^{1-\lambda}$. Since the waiting time of messages $j \in P_i$ at node u is at least $2 \cdot 2^{i-1}\mu/\delta^{1-\lambda}$, the delay between any two packets that are charged to messages in P_i is at least $2^i\mu/\delta^{1-\lambda}$. Since the optimal solution sends packet P at t through (u, v), we get $t \in I_j(u)\ \forall j \in P$ and thus $I_j(u) \subseteq [t - 2^i\mu, t + 2^i\mu]\ \forall j \in P_i$. Thus, the number of packets charged to messages in P_i is at most $2 \cdot 2^i\mu/(2^i\mu/\delta^{1-\lambda}) = 2\delta^{1-\lambda}$. □

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), budget L, communication time to the sink τ_{v_j} and clock time. The following lower bound shows that the competitive ratio of SL cannot be beaten by more than a factor $\log U$ 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 4. *No deterministic asynchronous memoryless algorithm is better than* $(\delta^\lambda, \delta^{1-\lambda})$*-competitive, for fixed* λ, $0 \leq \lambda \leq 1$.

Proof. Consider any deterministic asynchronous memoryless algorithm with latency costs at most δ^λ times the budget on the latency costs for fixed λ, $0 \leq \lambda \leq 1$. An adversary chooses a binary tree with root s and all leaves at distance δ from s. The adversary releases message 1 with latency L at time r_1 in a leaf v_1. There must be a node u where message 1 waits at most $\delta^\lambda(L - \tau_{v_1})/\delta$. The adversary releases message $j, j = 2, \ldots, \delta^{1-\lambda}$ at time $r_1 + j(L - \tau_{v_1})/\delta^{1-\lambda}$ such

that all messages j are sent over node u, 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 at equal distance, all messages are sent non-aggregated to and from u, whereas they are aggregated as early as possible in an optimal solution, in particular at u. □

Theorems 3 and 4 immediately imply the following corollary.

Corollary 1. *There exists a deterministic asynchronous algorithm that is* $(\sqrt{\delta}, \sqrt{\delta}\log U)$*-competitive and no deterministic asynchronous memoryless algorithm is better than* $(\sqrt{\delta}, \sqrt{\delta})$*-competitive.*

If we assume that $L \geq 2\delta$, which in practice is not a severe restriction at all, essentially the same analysis as in the proof of Theorem 3 gives $(2\delta^{\lambda}, 2\delta^{1-\lambda})$-competitiveness. Thus, in this case SL is a best possible on-line algorithm up to a constant multiplicative factor.

4 The Almost Synchronous Model

Typically in sensor networks clocks have a small drift. The CC-algorithm is not robust in the sense that its competitive ratio may be much worse if we assume existence of such clock drifts. However, the idea underlying the CC-algorithm gives rise to algorithms which have good competitive ratio even in the almost synchronous model. In this section we present such an algorithm. We assume that the difference between the time indicated at any two clocks is at most Δ. We assume all communication times to be equal and of unit length, i.e. $\tau(a) = 1\ \forall a$. We also divide nodes into classes; a node v is of class p if p is the maximal integer such that $\tau_v = h2^p + 1$ for some integer h, and v is of class 0 if $\tau_v = 1$. Note that $p \in \{0, \ldots, \lceil \log \delta \rceil\}$. The algorithm is the following:

Algorithm:AlmostSynchronousClock (ASC) Message j incurs 3 kinds of delay:

1. a delay of $t(I_j) - \tau_{v_j} - r_j$ at its release node v_j;
2. a delay of Δ at each node it traverses;
3. a delay which sums to $2^{p+1}\Delta$ at the first node of class $p, p > 0$, it traverses.

The waiting time of message j at a node v is the sum of the delays. A message is sent from a node v once its waiting time is over, unless some other message (packet) is sent from v earlier in which case j is aggregated with this packet.

Note that if $\Delta = 0$ the algorithm is identical to the CC-algorithm. To illustrate delay of the third kind we give an example: if a message traverses nodes of classes 1-4 in order 1,2,3,4 then its delay of the third kind of these nodes is respectively $4\Delta, 4\Delta, 8\Delta, 16\Delta$. If the order is 4,1,2,3 then its delay of the third kind is 32Δ at the node of class 4 and 0 elsewhere.

Now we analyze the competitive ratio of ASC. Let $V_k := \{v | 2^{k-1} < \tau_v \leq 2^k\}$ for some $k \in \mathbb{N}$, for $k = 1, \ldots, \lceil \log \delta \rceil$. First, we analyze the behavior of the algorithm for instances in which the release nodes of all messages is in V_k for some $k \in \mathbb{N}$.

Lemma 3. *If the* CC*-solution sends a packet from v, the* ASC*-solution sends at most $(k+1)$ packets from v which contain a message of the* CC*-packet, if $\forall j$ $v_j \in V_k$ for some $k \in \mathbb{N}$.*

Proof. Each packet, either CC or ASC, contains at least one message whose waiting time is completely over when the packet is forwarded. Hence without loss of generality we only consider messages whose waiting time is completely over when counting packets.

Consider a packet P_{CC} sent by the CC-solution from some node v at time t. In the remainder of the proof we only consider the messages in this packet. We analyze the number of ASC-packets which contain a message of P_{CC}. The delays of messages in P_{CC} are chosen such that all messages in this packet which traverse v, i.e. v is not the release node, arrive at this node at time t. As the delay of the first kind in the ASC-algorithm is identical to the delay incurred by the CC-algorithm we focus on the deviation from this time to analyze the number of packets ASC sends. This deviation may be caused either by delay of kind 2 and 3, or by the clock drift.

If $k = 0$ the lemma trivially holds, because all messages which are sent over some node $v \in V_0$ have this node v as release node. Hence, if they are sent in a single packet by the CC-solution they are also sent in a single packet in the ASC-solution.

For $k \geq 1$ we introduce the following notation: $V_{p,k} = \{v \in V_k | v \text{ is of class } p, \forall v' \in V_k \text{ of class } p, \tau_v \leq \tau_{v'}\}$ for $p \in \{0, \ldots, k-1\}$. $V_{p,k}$ is the set of nodes in V_k of class p with minimal communication time to the sink. Define $\tau(V_{p,k}) := \tau_v$ for some $v \in V_{p,k}$. The nodes of V_k are partitioned into layers $U_{p,k}$ for $p \in \{0, \ldots, k-1\}$ as follows:

$$\begin{aligned} U_{p,k} &:= \{v \in V_k | \tau(V_{p,k}) \leq \tau_v < \tau(V_{p+1,k})\} \text{ for } p \in \{1, \ldots, k-3\}, \\ U_{k-2,k} &:= \{v \in V_k | \tau(V_{k-2,k}) \leq \tau_v\}, \\ U_{k-1,k} &:= V_{k-1,k}. \end{aligned}$$

Note that $V_{p,k} \subseteq U_{p,k}$ for all p. Further, each message j with $v_j \in U_{p,k}$ traverses some node in $V_{p,k}$. See Figure 1 for a sketch of the layer structure.

We characterize a set of nodes S by its depth, which is $\max_{v \in S} \tau_v - \min_{v \in S} \tau_v$ and the *class string*. The class string is an ordered string representing the class of nodes in S by increasing communication time to the sink. I.e. V_3 has depth 4 and class string $\{2010\}$. In general, set V_k has depth 2^{k-1}. Node sets S and S' are *equivalent* if they have the same depth and class string.

We observe that all messages j with $v_j \in V_k$ are sent to a node in V_{k-1} from some node in $V_{k-1,k}$, i.e. a node of class $k-1$. Also, there are no nodes of higher class in V_k and this is the only node of class $k-1$ a node traverses in V_k. From these observations we may derive that all messages j with $v_j \in V_k$ which are sent over the same node $v \in V_{k-1,k}$ are sent from this node in a single packet. This can be seen as follows. The total accumulated delay of kind 2 and 3 that any message has incurred when sent from v is at least $2^k \Delta + \Delta$ because v is of class $k-1$. The total accumulated delay of kind 2 and 3 that any message has incurred

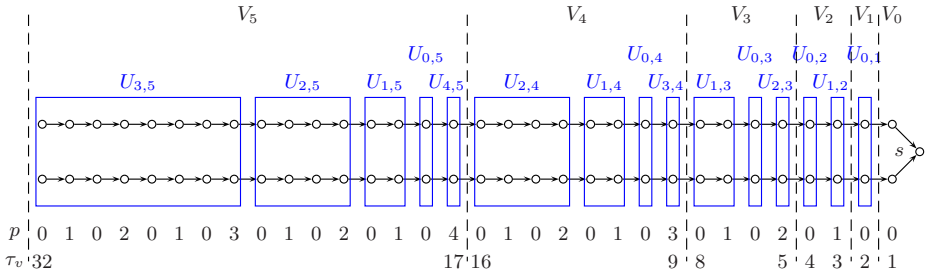

Fig. 1. Node set V_k and layers $U_{0,k}, \ldots, U_{k-1,k}$ for $k = 1, \ldots, 5$

when it arrives at v is at most $2^{k-1}\Delta + 2^{k-1}\Delta$, because the maximum class of any other node in V_k is $k-2$ and each message has traversed at most 2^{k-1} nodes. As the clock drift is bounded by Δ and the difference between the minimum and maximum delay of any two messages is at most $(2^k\Delta+\Delta)-(2^{k-1}\Delta+2^{k-1}\Delta) = \Delta$ all messages j with $v_j \in V_k$ which are sent over v in P_{CC} must be sent from this node in a single ASC-packet.

Now we are in position to prove our lemma using induction on k. Suppose the lemma holds for $V_0, \ldots, V_k$. Consider set V_{k+1}; this set is partitioned into layers $U_{0,k+1}, \ldots, U_{k,k+1}$. For $\ell = 0, \ldots, k-1$ layer $U_{\ell,k+1}$ is *equivalent* to set $V_{\ell+1}$, hence all messages j with $v_j \in U_{\ell,k+1}$ which are sent from the same node in $U_{\ell,k+1}$ are sent in a single packet. Thus there are at most k packets which arrive at any node $v \in U_{k,k+1}$. As $U_{k,k+1}$ has depth 1, all messages which have v as their release node, are sent from this node in a single packet. Hence, the total number of packets sent from any node in V_{k+1} is bounded by $k+1$. This proves the lemma. □

Theorem 5. ASC *is* $(1 + 4\Delta\delta/L, \log^2 \delta)$*-competitive.*

Proof. Consider a packet P sent by the optimal solution. Let $\mathcal{P}_{\mathrm{ASC}}$ be the set of packets sent by the ASC-algorithm which contain at least one message from P. Let $N_{i,k} = \{j \in N_i | \tau_{v_j} \in V_k\}$, for $i, k \in \mathbb{N}$, $1 \le i \le \lceil \log U \rceil, 1 \le k \le \lceil \log \delta \rceil$. Observe that for any choice of budget on the latency L, there are at most $2 \log \delta$ nonempty sets $N_{i,k}$. Using this, it follows from Lemma 2 and Lemma 3 that $|\mathcal{P}_{\mathrm{ASC}}| = O(\log^2 \delta)$. Hence, the communication costs of the ASC-solution are at most $O(\log^2 \delta)$ times the cost of an optimal L-bounded solution.

The latency of any message j is at most $L + \Delta\tau_{v_j} + 2\Delta\tau_{v_j} + \Delta$, where the sum consists of the delay of kind 1,2,3 and the clock drift. Thus, the latency of message j is at most $(1 + 4\Delta\delta/L)$ times the budget on the latency. □

If the drift is very small, competitiveness of ASC approaches the lower bound of $(1, \log \delta)$ of the synchronous case, which we proved in [3]. If the drift is of the same order as the latency, i.e. $\Delta = O(L)$, then the SL algorithm, with $\lambda = 1$, has strictly better (β, α)-competitive ratio, than the ASC algorithm. In case of

such drifts, it is not plausible anymore to consider the clocks to be synchronized in any sense.

5 Conclusions and Open Problems

We presented on-line distributed algorithms for data aggregation in sensor networks. We considered algorithms under two different models for sensor clocks. For the almost synchronous time model we presented an algorithm which minimizes communication costs under a small excess of the latency budget. These are the first analyses of algorithms for this model, which models actual sensor networks closer than the known ones. We emphasize that the results depend linearly on the drift, and that if the drift is very small our algorithms approach best possible competitive ratios.

For the asynchronous time model we presented an algorithm which balances the communication and latency costs up to a factor $\log U$, where U is the ratio between maximum and minimum allowed delay. We showed that no memoryless algorithm can have a competitive ratio which is more than a factor $\log U$ better than ours, and in case the latency budget is not too small our algorithm is best possible within the class of memoryless algorithms.

The competitive ratio of our asynchronous algorithm is almost balanced; it would be interesting to find an algorithm with balanced ratios, equal to the lower bounds we presented in this paper. Another path for future research is to make a more careful analysis of the almost synchronous time model, in order to determine the maximum clock drift for which almost synchronous algorithms have better competitive ratio than asynchronous algorithms.

References

1. Akyildiz, I., Su, W., Sanakarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: A survey. Computer Networks Journal 38(4), 393–422 (2002)
2. Albers, S., Bals, H.: Dynamic TCP acknowledgment: Penalizing long delays. SIAM Journal Discrete Mathematics 19(4), 938–951 (2005)
3. Becchetti, L., Korteweg, P., Marchetti-Spaccamela, A., Skutella, M., Stougie, L., Vitaletti, A.: Latency constrained aggregation in sensor networks. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 88–99. Springer, Heidelberg (2006)
4. Brito, C., Koutsoupias, E., Vaya, S.: 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), pp. 627–635 (2004)
5. Broder, A., Mitzenmacher, M.: Optimal plans for aggregation. In: Proceedings of the 21st Annual Symposium on Principles of Distributed Computing (PODC), pp. 144–152 (2002)
6. Dooly, D.R., Goldman, S.A., Scott, S.D.: On-line analysis of the TCP acknowledgment delay problem. Journal of the ACM 48(2), 243–273 (2001)
7. Heinzelman, W., Chandrakasan, A., Balakrishnan, H.: Energy efficient communication protocols for wireless microsensor networks. In: Proceedings of Hawaiian International Conference on Systems Science, pp. 3005–3014 (2000)

8. Khanna, S., Naor, J., Raz, D.: Control message aggregation in group communication protocols. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 135–146. Springer, Heidelberg (2002)
9. Marathe, M.V., Ravi, R., Sundaram, R., Ravi, S.S., Rosenkrantz, D.J., Hunt III, H.B.: Bicriteria network design problems. Journal of Algorithms 28(1), 142–171 (1998)
10. Ravi, R., Marathe, M.V., Ravi, S.S., Rosenkrantz, D.J., Hunt III, H.B.: Many birds with one stone: Multi-objective approximation algorithms (extended abstract). In: Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC), pp. 438–447 (1993)
11. Stankovic, J.A.: Research challenges for wireless sensor networks. SIGBED Rev. 1(2), 9–12 (2004)
12. Sundararaman, B., Buy, U., Kshemkalyani, A.D.: Clock synchronization for wireless sensor networks: a survey. Ad. Hoc. Networks 3(3), 281–323 (2005)