Tight Bounds for Delay-Sensitive Aggregation

Yvonne Anne Oswald
Computer Engineering and Networks Laboratory (TIK)
ETH Zurich, Switzerland
oswald@tik.ee.ethz.ch

Stefan Schmid
Institut für Informatik
Technische Universität
München, Germany
schmiste@in.tum.de

Roger Wattenhofer
Computer Engineering and Networks Laboratory (TIK)
ETH Zurich, Switzerland
wattenhofer@tik.ee.ethz.ch

ABSTRACT

This paper studies the fundamental trade-off between communication cost and delay cost arising in various contexts such as control message aggregation or organization theory. An optimization problem is considered where nodes are organized in a tree topology. The nodes seek to minimize the time until the root is informed about their states and to use as few transmissions as possible at the same time. We derive an upper bound on the competitive ratio of $O(\min(h, c))$ where h is the tree's height, and c is the transmission cost per edge. Moreover, we prove that this upper bound is tight in the sense that any oblivious algorithm has a ratio of at least $\Omega(\min(h, c))$. For chain networks, we prove a tight competitive ratio of $\Theta(\min(\sqrt{h}, c))$. Furthermore, the paper introduces a new model for online event aggregation where the importance of an event depends on its difference to previous events.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]

General Terms

Algorithms, Theory

Keywords

Competitive Analysis, Wireless Sensor Networks, Distributed Algorithms, Aggregation

1. INTRODUCTION

The analysis of distributed algorithms often revolves around time and message complexity. On the one hand, we want our distributed algorithms to be fast, on the other hand, communication should be minimized. Problems often ask to optimize one of the two—and treat the other only as a secondary target. However, there are situations where time and message complexity are equally important.

In this paper, we study such a case known as *distributed aggregation*. Nodes of a large distributed network may sense potentially interesting data which they are to report to a central authority. Not only should the data make its way fast through the network such that information is not unnecessarily delayed; but also, since message transmission is costly, one may reduce the number of transmissions by aggregating messages along the way. In other words, nodes may wait for further packets before forwarding them in order to decrease the number of transmission at the expense of a later arrival of the information at the sink. This problem has many applications. In the past it was mostly studied in contexts such as control message aggregation, or organization theory. In the heyday of wireless networking the first application that comes to mind is perhaps sensor networking. Due to energy constraints, it is necessary to minimize the number of transmissions. At the same time, it is desirable to aim at minimizing the time until the nodes are informed about changes of measured values.

This paper assumes that the communication network of the nodes forms a pre-computed directed spanning tree on which information about events is passed to the root node (the sink). We assume that data arrives at the nodes in an online (worst-case) fashion. A main challenge is to decide at what points in time data should be forwarded to a parent in the tree.

Our contributions are the following. We prove that a simple algorithm achieves a competitive ratio of $O(\min(h, c))$ where h is the tree's height, and c is the transmission cost per edge, which improves on an existing upper bound of $O(h \log(cn))$, where n is the network size. This algorithm is oblivious, i.e., decisions at each node are based solely upon the static local information available at the node. Being oblivious is a desirable property of distributed algorithms, since non-oblivious algorithms need dynamic updating mechanisms—a costly operation. We also demonstrate that this upper bound is tight in the sense that there exist problem instances where any oblivious algorithm has a ratio of at least $\Omega(\min(h, c))$. Earlier work proved a lower bound of $\Omega(\sqrt{h})$ on a chain network. Therefore, we examine this topology more closely and show that chain networks are inherently simpler than general trees by giving a competitive ratio of $\Theta(\min(\sqrt{h}, c))$ for oblivious algorithms. In the last part of this paper, we initiate the study of a new event aggregation model which takes into account that nodes often have non-binary data to aggregate and greater differences between values need to be reported to the root faster than small differences. We present a model comprising this additional constraint as well as an oblivious algorithm achieving a competitive ratio of $\Theta(c/\epsilon)$ on a one-link network, where ϵ is the minimum difference between two values. Moreover, we devise a polynomial algorithm that can compute an optimal aggregation strategy offline for chain networks.

This paper is organized as follows. We review related work in Section 2 and we introduce our model in Section 3. Section 4 contains our main technical contributions and Section 5 addresses value-sensitive aggregation. We conclude the paper in Section 6.

2. RELATED WORK

The trade-off between delay and communication cost appears in various contexts, and plays a role in the design of algorithms for wireless sensor networks, for Internet transfer protocols, and also appears in organization theory. This section gives a brief overview of related work on this topic.

Papadimitriou et al. [11, 12] investigate the following optimization problem: An organization is modeled as a tree where employees (leaves) receive messages to be sent to the boss (the root). The authors observe that before it is possible to accept some communication, humans typically must do a "context switch" in order to process the message properly, and that the cost of these context switches becomes large if communications are not scheduled carefully. Concretely, the cost function consists of two components, one capturing the total number of messages sent along the tree, and the other one capturing the total delay incurred by the messages before they reach the root. For the formal analysis of this dilemma of interruptions, a Poisson process queuing model is assumed.

A basic problem in the design of Internet transfer protocols such as the TCP protocol concerns the acknowledgements (ACKs) which have to be sent by a receiver in order to inform the sender about the successful reception of packets: In many protocols, a delay algorithm is employed to acknowledge multiple ACK packets with a single message or to piggy-back the ACK on outgoing data segments [14]. The main objective of these algorithms is to save bandwidth (and other overhead at the sender and receiver) while still guaranteeing small delays. The problem of aggregating ACKs in this manner is also known as the *TCP acknowledgment problem* [5]. Karlin et al. [7] pointed out interesting relationships of the single-link acknowledgment problem to other problems such as *ski-rental*, and gave an optimal, $e/(e-1)$-competitive randomized online algorithm. Brito et al. [4] calculated general upper and lower bounds for an asynchronous ACK delaying problem.

There are many variations of the theme, e.g., Albers et al. [1] seek to minimize the number of acknowledgments sent plus the *maximum* delay incurred for any of these packets. They propose a $\pi/6$-competitive deterministic algorithm for the single link, which is also a lower bound for any deterministic online algorithm. Frederiksen et al. [6] consider deterministic and randomized algorithms for bundling packets on a single-link network; their objective function measures the total time elapsed while packets are waiting at the leaf node, but have not been delivered yet.

Finally, there is much literature on aggregation in sensor networks [10, 13, 15, 16, 17]. E.g., Becchetti et al. [2] studied online and offline algorithms for scenarios where fixed deadlines must be met. They show that the offline version of the problem is strongly NP-hard and provide a 2-approximation algorithm.

The paper closest to ours is by Khanna et al. [8]. Our model is derived from the one by [8] which investigates the task of centralized and decentralized online control message aggregation on weighted tree topologies. In particular, [8] presents a $O(h \log \alpha)$-competitive distributed algorithm, where h is the tree's height, and α is the sum of the weights of the tree's edges. Moreover, the authors show that any oblivious distributed online algorithm has a competitive ratio of at least $\Omega(\sqrt{h})$. In this paper, we study the same algorithm and we give a new analysis for scenarios where the communication cost is c on all links, resulting in a better upper bound of $O(\min(h, c))$. We also derive a new generalized lower bound for edge cost which are different from h, and show that for any oblivious aggregation algorithm, the competitive ratio is at least $\Omega(\min(h, c))$. Moreover, by taking into account many interesting properties of our algorithm, we show that for chain graphs an upper bound of $O(\min(\sqrt{h}, c))$ holds. This is asymptotically tight.

Korteweg et al. [9] address the same problem, but they follow a *bicriterion approach* which considers time and communication as two independent optimization problems: a (B, A)-bicriterion problem minimizes objective A under a budget on objective B. Inter alia, the authors prove that if r is the ratio between the maximum and the minimum delay allowed, then the competitive ratio of their algorithm is $(2h^{\lambda}, 2h^{1-\lambda} \log r)$ for any λ in $(0, 1]$.

3. MODEL

Let there be a rooted tree $T = (V, E)$ of height h with root $r \in V$ and $n = |V|$ nodes. Every node u except for the root r (the *sink*) has a parent node v, i.e., an edge $(u, v) \in E$. The cost of transmitting a message over an edge is c.

We assume that *events* occur at the leaf nodes $L \subset V$ (e.g., a control message arriving at a node, or a sensor node detecting an environmental change). We will refer to the information about the occurrence of a single event as an *event packet*. Leaf l creates an event packet p for every event that happens at l.

Eventually, all event packets have to be forwarded to the root. Instead of sending each packet $p \in \mathcal{P}$ individually to a node's parent after the event took place, nodes can retain packets and send a *message* m consisting of one or more packets together later, thus saving on communication cost as we have to pay for a link only once *per message* (rather than per event). Messages can be *merged* iteratively with other messages on their way to the root.

We consider a synchronous model where time is divided into time slots. In each slot, an arbitrary number of events can arrive at each node. For an event packet p, $t_l(p)$ denotes the time slot its corresponding event occurred at a node and $t_r(p)$ the time when it reaches the root. For each time slot an event waits at a node, we add one unit to the delay cost, i.e., the delay cost $dc(p)$ the event accumulates until reaching the root is $dc(p) = t_r(p) - t_l(p)$.

Each message can only be forwarded one hop per round, i.e., a message always has to wait one round. Thus, the delay accumulated by an event is at least h_l, where h_l denotes the length of the path from the respective leaf l to the root. The total delay cost of all events accumulated up to time slot T is hence

$$dc_T = \sum_{p \in \mathcal{P}, t_r(p) \leq T} dc(p) + \sum_{p \in \mathcal{P}, t_r(p) > T} (T - t_l(p)).$$

Nodes can aggregate as many event packets as needed. At each time step t, a node may aggregate awaiting event packets and forward the resulting message to its parent. The cost of sending a message is c per edge no matter how many event packets are combined. Consequently, the total communication cost is the sum of the edge cost of all message transmissions. More formally, let S_t be the set of nodes sending out a message in time slot t, then the total communication cost cc_T up to time slot T is $cc_T = \sum_{t=1}^{T} |S_t|$. The total cost up to time T is the sum of both the delay and the communication cost,

$$cost_T = dc_T + cc_T.$$

Observe that the edge cost c allows us to weight delay and communication costs: a larger c implies that communication cost become relatively more important compared to the delay cost. Note that we neglect the energy consumption in idle listening mode and consider the nodes' transmission cost only. We believe that this is justified for networks where listening nodes have their radios turned off most of the time and only check for data transfers at the very beginning of each time slot.

Nodes do not know the future arrival time of events, and hence have to make the decisions on when to send messages *online*. We are in the realm of *competitive analysis* [3] and define the (strict) *competitive ratio* ρ achieved by an online algorithm $\mathcal{AGG}$ as the delay and communication cost of $\mathcal{AGG}$ divided by the total cost of an optimal offline algorithm $\mathcal{OPT}$.

DEFINITION 3.1 (ρ-COMPETITIVENESS). *An online algorithm $\mathcal{AGG}$ is (strictly) ρ-*competitive *compared to an optimal offline algorithm $\mathcal{OPT}$ if for all input sequences I, i.e., all possible event arrival sequences,*

$$cost^{\mathcal{ALG}}(I) \leq \rho \cdot cost^{\mathcal{OPT}}(I).$$

The goal of an online algorithm designer is hence to devise algorithms minimizing ρ, as a small ρ describes the guaranteed worst-case quality of an algorithm.

In this paper, we focus on *oblivious* online algorithms.

DEFINITION 3.2 (OBLIVIOUS ALGORITHMS). *A distributed online algorithm $\mathcal{ALG}$ is called* oblivious *if the decisions by each node $v \in V$ whether to transmit a message solely depends on the time slots when the packets currently stored at v arrived (at v).*

In particular, Definition 3.2 implies that the decisions of a node v do not depend on packets forwarded by v earlier or on v's location in the aggregation network.

4. ALGORITHM AND ANALYSIS

4.1 Algorithm

This section describes and analyzes the deterministic online algorithm $\mathcal{AGG}$ presented in [5, 8]. The algorithm is *oblivious* in the sense that given the same packet arrival times, each node will react in the same way, i.e., independently of its distance to the root (cf. Definition 3.2). Essentially, the event aggregation algorithm $\mathcal{AGG}$ seeks to balance the total delay cost and the total communication cost. In order to do so, it aggregates information about multiple events into one message until the forwarding condition is satisfied. Whenever a new event occurs or a message arrives, it is merged with the message waiting for transmission at the node.

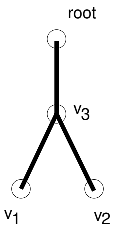
For each message m, we define $delay(m,t)$, denoting the delay associated with message m at time t. Informally, it is the sum of the accumulated delay cost of all the event packets the message m contains, minus the sum of the edge cost of every edge a message was sent over. Thus, not only the delay but also the communication cost paid already is taken into account. More formally, let a message m be a set of merged messages $\{m_1, \ldots, m_k\}$, where message m_i consists of $|m_i|$ packets and arrived at the current node in time slot t_i. The delay of message m at time t is defined by

$$delay(m,t) := \sum_{i=1}^{k} \left[delay(m_i, t_i) - c + |m_i|(t - t_i) \right].$$

When executing algorithm $\mathcal{AGG}$, a node v forwards a message m to its parent as soon as the current accumulated delay exceeds the transmission cost.

$$delay\,(m,t) \geq c.$$

We demonstrate the execution of $\mathcal{AGG}$ on a simple example. Consider the tree and the event arrival sequence in Figure 1. There are two events occurring at leaf node v_1, one in time slot $t = 1$, one at time $t = 2$. Node v_2 receives two packets at $t = 2$. The transmission cost is set to $c = 3$. For this input sequence, node v_1 sends its two packets after time $t = 2$ and node v_2 after time $t = 3$, i.e., as soon as the accumulated delay reaches or exceeds $c = 3$. Node v_3 incurs a delay of two after the message from v_1 arrives.



events at node	v_1	v_2
$t = 1$	1	0
$t = 2$	1	2

delay at node	v_1	v_2	v_3
$t = 1$	1	0	0
$t = 2$	3	2	0
$t = 3$	0	4	2
$t = 4$	0	0	7
$t = 5$	0	0	0

Figure 1: Example execution of $\mathcal{AGG}$ where the transmission cost c is 3.

In the next time slot, v_3's delay cost increases to 7, as the message from v_2 carries an "overflow delay" of $4 - c = 1$ and there are four messages at v_3. Adding this up to the delay cost of the previous slot results in $2 + 1 + 4 = 7$.

4.2 Tight Bound for Trees

We establish a new upper bound of $O(\min(h,c))$ on the competitive ratio of $\mathcal{AGG}$ by an astoundingly simple analysis. Instead of calculating the delay and the communication cost the event packets accumulate, we focus on the messages $\mathcal{AGG}$ and $\mathcal{OPT}$ send. We proceed as follows. First, we investigate the competitiveness of $\mathcal{AGG}$ for a single link network, then tackle the chain network, and finally generalize our analysis to tree topologies.

THEOREM 4.1. *On arbitrary trees, the competitive ratio of $\mathcal{AGG}$ is at most*

$$\rho = \frac{cost^{\mathcal{AGG}}}{cost^{\mathcal{OPT}}} \in O(\min{(h,c)}).$$

PROOF. The proof unfolds in several lemmas.

LEMMA 4.2. *[5] The competitive ratio of $\mathcal{AGG}$ on a single link is at most* 2.

PROOF. Consider a single link of cost c. Let t_i be the time slot $\mathcal{AGG}$ sends the message m_i containing information on k_i events. This transmission entails a communication cost of $cc_i^{\mathcal{AGG}} = c$. The total delay of the events in message m_i is $dc_i^{\mathcal{AGG}} = c + x_i$ for some $x_i \geq 0$, because m_i will be sent as soon as its events' delay exceeds c, and because there can be $(x_i + 1) \geq 0$ simultaneous event arrivals in the time slot immediately preceding t_i. Any optimal offline algorithm $\mathcal{OPT}$ will have at least delay cost x_i as well. In addition, $\mathcal{OPT}$ incurs a cost of at least c for the event packets contained in m_i, either because of a transmission or due to the accumulated delay, $cost_i^{\mathcal{AGG}}/cost_i^{\mathcal{OPT}} \leq (2c + x_i)/(c + x_i)$. This expression is maximized for $x_i = 0$, implying a competitive ratio of 2. □

In a next step we analyze the chain network. First, we assume that each message of the optimal offline algorithm which is sent from a given leaf node comprises packets of multiple messages sent by $\mathcal{AGG}$ at this leaf, and show that the claim indeed holds in this case. Second, we prove that the claim holds true also if $\mathcal{AGG}$ sends more messages than $\mathcal{OPT}$. Finally, our results are generalized for arbitrary sending sequences and for trees.

LEMMA 4.3. *If the optimal algorithm's messages sent from a given leaf l include all packets of multiple messages of $\mathcal{AGG}$, the competitive ratio is at most $O(\min{(h_l, c)})$.*

PROOF. Let m_i^A denote the i^{th} message leaf node l located at depth h_l sends to its parent. Let the total number of messages sent by

$\mathcal{AGG}$ at l be denoted by M_l^A. Due to our assumption, a message m_j^O of $\mathcal{OPT}$ contains the event packets of one or several messages of $\mathcal{AGG}$, i.e., $m_j^O = \cup_{k=i}^{i'} m_k^A$ for some $i' \geq i$. The cumulated cost for $\mathcal{AGG}$ is less than $(i'-i+1)(3h_l c)$, since any message incurs a delay cost of less than $2c$ and a transmission cost of c per hop towards the root.

Let the total number of messages sent by the optimal algorithm at node l be M_l^O. The cost a message m_j^O accumulates is at least $(i'-i+1)(c+h_l-1)$; the term $(i'-i)c$ is the delay cost the events accrue while waiting at the leaf and $(i'-i+1)(h_l-1)$ is the lower bound for the delay incurred on the way to the root, matched if $\mathcal{AGG}$'s messages contain one event each. The additional cost c stands for the communication cost the optimal algorithm has to pay for the first link. Note that this cost cannot decrease due to a possible aggregation closer to the root. Hence we can write

$$\begin{aligned} \rho &= cost^{\mathcal{AGG}}/cost^{\mathcal{OPT}} \\ &\leq \frac{\sum_{i=1}^{M_l^A} 3h_l c}{\sum_{j=1}^{M_l^O} (i'-i+1)(c+h_l-1)} \\ &= \frac{3h_l c}{c+h_l-1} \in O(\min(h_l, c)), \end{aligned}$$

which completes our proof. □

LEMMA 4.4. *If the optimal algorithm sends more messages than $\mathcal{AGG}$ from a given leaf node l, the competitive ratio is at most $\rho \in O(\min(h_l, c))$.*

PROOF. Let the total number of messages that node l at depth h_l sends be M_l^A and let the total number of messages by the optimal algorithm be M_l^O. The delay cost these messages accumulate for $\mathcal{OPT}$ is at least $M_l^O h_l$, regardless of aggregation operations closer to the root, and the communication cost is at least $M_l^O c$ since every message has to be sent over the first link separately. The cumulated cost for $\mathcal{AGG}$ is at most $M_l^A(3h_l c)$. Thus

$$\begin{aligned} \rho &\leq \frac{\sum_{i=1}^{M_l^A} 3h_l c}{\sum_{j=1}^{M_l^O} h_l + c} \leq \frac{3h_l c}{h_l + c} \\ &\in O(\min(h_l, c)). \end{aligned}$$

□

Continuing the proof of Theorem 4.1, it remains to consider an arbitrary sequence of event arrivals where the optimal algorithm's messages are not unions of the packets of several messages of the online algorithm $\mathcal{AGG}$ or where the optimal algorithm transmits more often at the leaf, but messages of $\mathcal{AGG}$ are split and recombined across several messages the optimal algorithm sends. We divide the sequence of event arrivals into sections where $\mathcal{AGG}$ sends more messages than $\mathcal{OPT}$ and into section where $\mathcal{AGG}$ sends fewer messages than $\mathcal{OPT}$. We proceed in the following way: Set $t := 0$. For every time slot t_i where $\mathcal{AGG}$ sends a message we check whether the number of messages so far exceeds the number of messages the optimal algorithm sends. As soon as this condition is not satisfied anymore (time slot t_k) we backtrack to the previous message at time slot t_{k-1}. We exempt $\mathcal{OPT}$ from paying for the messages it did not send towards the root up to this time slot. For the section $[0, t_{k-1}]$, Lemma 4.3 can be applied. We continue by resetting $t := 0$. For every time slot t_i where $\mathcal{AGG}$ sends a message we check whether the number of messages so far is below the number of messages the optimal sends. As soon as this condition is not satisfied anymore (time slot t_k) we backtrack to the previous message at time slot t_{k-1}. We exempt $\mathcal{OPT}$ from paying for the messages it did not send towards the root up to this time slot. For the section $[0, t_{k-1}]$ we can now apply Lemma 4.4. We repeat this examination for all remaining slots. Consequently it holds for each leaf that there is no section where $\mathcal{AGG}$'s ratio is higher than $O(\min(h, c))$. Since no assumptions on the behavior of the optimal algorithm have been made in the arguments above and since we only count the communication cost incurring on the edges between the leaves and their neighbors, the statements hold for general trees as well. □

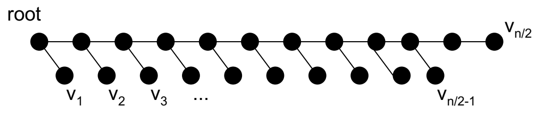


Figure 2: Lower bound topology.

We conclude our investigations of the tree network with a lower bound stating that $\mathcal{AGG}$ is asymptotically optimal for any *oblivious algorithm.* Recall from Definition 3.2 that for oblivious algorithms, it holds that the wait time w only depends on the packet arrival time of the packets currently stored by a given node.

THEOREM 4.5. *Any oblivious deterministic online algorithm has a competitive ratio of at least*

$$\rho = \frac{cost^{ALG}}{cost^{\mathcal{OPT}}} \in \Omega(\min(h, c))$$

on the tree.

PROOF. Consider the tree topology depicted in Figure 2 which consists of a chain network of $n/2+1$ nodes, where all nodes except for the two last ones have an additional neighbor. The leaf nodes are referred to by $v_1, ..., v_{n/2}$.

Assume an input sequence where all leaves simultaneously get one packet and consider any oblivious online algorithm ALG. Since ALG is oblivious, according to Definition 3.2, each leave node v_i will send the packet after waiting for w time slots to its parent, where the packet arrives at time $w+1$ (the value of $w > 0$ depends on the chosen algorithm). From there, the packet leaves at time $2w+1$. And so on. Generally, the packet of leaf node v_i will arrive at a node at distance j from v_i at time $jw+j$, and will stay there for w rounds. Observe that the packets of two nodes v_{i-1} and v_i are never merged into one message, as the time intervals $[(j-1)w+(j-1), jw+(j-1)]$ and $[jw+j, (j+1)w+j]$ are disjoint. Thus, ALG has communication cost in the order of $\Theta(h^2 c)$ and delay cost in the order of $\Theta(h^2 w)$. Now consider an algorithm which aggregates all these packets on the way to the root: the communication cost are in $\Theta(ch)$, and the delay cost are $\Theta(h^2)$. As the optimal algorithm $\mathcal{OPT}$ can only have lower cost, we have the following (asymptotic) competitive ratio:

$$\rho = \frac{cost^{ALG}}{cost^{\mathcal{OPT}}} \geq \frac{h^2 w + h^2 c}{2hc + h^2},$$

and the claim follows. □

Discussion. The analysis of Theorem 4.1 can be compared to the results obtained in [8]. There, an upper bound of $O(h \log \alpha)$ is derived for the competitive ratio of $\mathcal{AGG}$, where α is the total edge weight of the tree. If all edges have a weight c, this translates into $O(h \log(nc))$, which is $O(h^2 \log c)$ in balanced binary trees. In this case, our much simpler analysis is better by a factor of $\Theta(h \log c)$ if $h < c$. In other networks, for instance, on chain topologies, the gap between the two bounds narrows, although it always remains positive.

4.3 Tight Bound for Chains

In order to obtain our upper bound for trees, the previous section has already briefly studied the competitiveness of $\mathcal{AGG}$ on chain networks. In the following, it is shown by a more detailed analysis taking into account many intrinsic properties of $\mathcal{AGG}$, that the bound can be improved. Concretely, we prove that on the chain topology, $\mathcal{AGG}$ is $O(\sqrt{h})$-competitive, and that no oblivious algorithm is can be less than $\Omega(\min(c, \sqrt{h}))$-competitive.

THEOREM 4.6. *In chain graphs, the competitive ratio of $\mathcal{AGG}$ is*

$$\rho = \frac{cost^{\mathcal{AGG}}}{cost^{\mathcal{OPT}}} \in O\left(\min\left(\sqrt{h}, c\right)\right).$$

PROOF. We start analyzing input sequences where $\mathcal{AGG}$ does not merge any messages at inner nodes. We then show that merge operations cannot improve or deteriorate the competitive ratio.

LEMMA 4.7. *On chain graphs and for sequences where $\mathcal{AGG}$ does not have any merge operations, the competitive ratio of $\mathcal{AGG}$ is $O(\sqrt{h})$.*

PROOF. Consider a transmission of the optimal offline algorithm $\mathcal{OPT}$, and assume that $\mathcal{OPT}$'s message m^O consists of packets which are distributed over $x\sqrt{h}$ messages m_i^A of the online algorithm $\mathcal{AGG}$. Observe that $x \in \omega(1)$ since otherwise the claim trivially holds due to the communication cost.

Let n_i denote the number of packets in message m_i^A, and let t_i denote the time when this message departs from the leaf node. Without loss of generality we assume that $n_i \leq c$. In order to send the $x\sqrt{h}$ messages individually to the root, $\mathcal{AGG}$ incurs a transmission cost of $cc^{\mathcal{AGG}} = chx\sqrt{h}$ and the delay cost is bounded by $dc^{\mathcal{AGG}} < 2chx\sqrt{h}$. For the optimal algorithm this transmission entails a communication cost of $cc^{\mathcal{OPT}} = ch$. We now bound the delay cost accumulated by each message m_i^A which is merged into message m^O by the optimal algorithm. Without loss of generality, assume that for $1 \leq i \leq x\sqrt{h}$, all n_i packets arrive simultaneously: if packets arrived dispersed over time, $\mathcal{OPT}$ would incur higher delay cost. Let λ_i denote the time interval from the arrival of the n_i packets until the corresponding message departs from the leaf. Moreover, let δ_i be the time interval after the message has departed from the leaf until the next set of n_{i+1} packets arrives. The delay cost the $\mathcal{OPT}$ accumulates at the leaf is given by

$$\sum_{l=i}^{x\sqrt{h}-1} \sum_{j=l}^{x\sqrt{h}-1} n_l \left(\delta_j + \lambda_j\right).$$

We have $\lambda_l = \lceil c/n_l \rceil$ by the definition of $\mathcal{AGG}$. Under the assumption that λ_l is c/n_l, the delay of the optimal algorithm decreases while $\mathcal{AGG}$'s cost remain the same. In order to guarantee that consecutive messages cannot merge, it must hold for δ_l that

$$\delta_l \geq \max\left(\frac{h}{2}\left\lceil\frac{2c}{n_l}\right\rceil - \frac{h}{2}\left\lceil\frac{2c}{n_{l+1}}\right\rceil - \lambda_{l+1}, 0\right).$$

We show now that the delay cost for the optimal algorithm decreases if we balance the number of packets per message. Consider two consecutive messages m_i^A and m_{i+1}^A, where m_{i+1}^A is not the last message of m_j^O for some j. Assume $n_i \geq n_{i+1} + 1$. We want to compute the difference between the delay cost in this case (case a: dc_1) and the delay cost when we remove one packet from m_i^A and add it to m_{i+1}^A (case b: dc_2). To this end it suffices to examine the delay cost accrued in the time slots between the departure of m_{i-1}^A and the arrival of m_{i+2}^A. The relevant delay cost is $\sum_{l=1}^{i-1} n_l \cdot (\delta_{i-1} + \lambda_i + \delta_i + \lambda_{i+1} + \delta_{i+1}) + n_i\left(\lambda_i + \delta_i + \lambda_{i+1} + \delta_{i+1}\right) + n_{i+1}\left(\lambda_{i+1} + \delta_{i+1}\right)$. Note that m_{i+1}^A cannot catch up with m_i^A because of its size in dc_1, thus $\delta_i = 0$. For dc_2, δ_i can only exceed 0 if $n_i - 1 > n_{i+1} + 1$. The difference Δ_{dc} between the two costs is hence

$$\Delta_{dc} = \sum_{l=1}^{i-1} n_l \cdot (\delta_{i-1}^{a)} + \lambda_i^{a)} + \lambda_{i+1}^{a)} + \delta_{i+1}^{a)} - \delta_{i-1}^{b)} - \delta_i^{b)}$$

$$-\lambda_{i+1}^{b)} - \delta_{i+1}^{b)}) + n_i(\lambda_i^{a)} + \lambda_{i+1}^{a)} + \delta_{i+1}^{a)} - \lambda_i^{b)} - \delta_i^{b)} - \lambda_{i+1}^{b)}$$

$$-\delta_{i+1}^{b)}) + n_{i+1}(\lambda_{i+1}^{a)} + \delta_{i+1}^{a)} - \lambda_{i+1}^{b)} - \delta_{i+1}^{b)}) - \lambda_i^{b)} - \delta_i^{b)}$$

$$= \sum_{l=1}^{i-1} n_l \cdot \left(\frac{h}{2}\left(\left\lceil\frac{2c}{n_{i+1}}\right\rceil - \left\lceil\frac{2c}{n_i}\right\rceil\right) + \frac{c}{n_{i+1}}\right)$$

$$+n_i\left(\frac{h}{2}\left(\left\lceil\frac{2c}{n_{i+1}}\right\rceil - \left\lceil\frac{2c}{n_i - 1}\right\rceil\right) + \frac{c}{n_i} - \frac{c}{n_i - 1} + \frac{c}{n_{i+1}}\right)$$

$$+n_{i+1}\left(\frac{h}{2}\left(\left\lceil\frac{2c}{n_{i+1}}\right\rceil - \left\lceil\frac{2c}{n_{i+1}+1}\right\rceil\right) + \frac{c}{n_{i+1}} - \frac{c}{n_{i+1}+1}\right)$$

$$-\frac{c}{n_i - 1} - \frac{h}{2}\left(\left\lceil\frac{2c}{n_i - 1}\right\rceil - \left\lceil\frac{2c}{n_{i+1}+1}\right\rceil\right) + \frac{c}{n_{i+1}+1}.$$

Clearly, the sum of the terms multiplied by $h/2$ is at least zero. Observe that the following inequality holds for all $n_i \geq 2, n_i - 1 > n_{i+1}$, and arbitrary n_{i-1}:

$$n_i\left(\frac{1}{n_i} - \frac{1}{n_i - 1} + \frac{1}{n_{i+1}}\right) - \frac{1}{n_i - 1}$$
$$+n_{i+1}\left(\frac{1}{n_{i+1}} - \frac{1}{n_{i+1}+1}\right) + \frac{1}{n_{i+1}+1} \quad > \quad 0.$$

Hence, the difference of the cost is always positive, and a lower bound for the delay cost is reached if for all pairs of consecutive messages the second one contains at least as many packets as the first one.

We now compute the delay cost of this balanced arrival sequence. Let the last message with a size four times its predecessor's, i.e., $n_k \geq 4n_{k-1}$, be m_k^A. (If no such message exists, take $k = 1$.) The total sum of the delay cost in this case is

$$\sum_{i=1}^{x\sqrt{h}-1} \sum_{j=i}^{x\sqrt{h}-1} n_i\left(t_j - t_{j+1}\right) > \sum_{i=1}^{k-1}\sum_{j=i}^{k-1} n_i\delta_j + \sum_{i=k}^{x\sqrt{h}-1} \sum_{j=i}^{x\sqrt{h}-1} n_i\lambda_j.$$

The second summand is at least $(x\sqrt{h} - 1 - k)^2 c/4$ since the delay cost m_i^A accrues in between the arrival of m_j^A and m_{j+1}^A is at least $n_i\lambda_j = n_i c/n_j \geq c/4$ since $n_j \leq 4n_i$ for all $i, j > k$. In order to bound the first summand, we use the fact that because

$\forall\, i:\ \ n_i < n_{i+1}$ it holds that $\lceil 2c/n_i \rceil - \lceil 2c/n_{i+1} \rceil \geq 0$. Thus

$$\begin{aligned}\sum_{i=1}^{k-1}\sum_{j=i}^{k-1} n_i\delta_j &= \sum_{i=1}^{k-1} n_i \sum_{j=i}^{k-1} \frac{h}{2}\left(\left\lceil\frac{2c}{n_j}\right\rceil - \left\lceil\frac{2c}{n_{j+1}}\right\rceil\right)\\ &= \sum_{i=1}^{k-1} n_i \frac{h}{2}\left(\left\lceil\frac{2c}{n_i}\right\rceil - \left\lceil\frac{2c}{n_k}\right\rceil\right)\\ &> \sum_{i=1}^{k-1} n_i \frac{h}{2}\left(\frac{2c}{n_i} - \frac{2c}{n_k} - 1\right)\\ &> \frac{(k-1)\,hc}{4}.\end{aligned}$$

Note that if $k < x\sqrt{h}/2$ the delay cost amounts to more than $\Omega((x\sqrt{h}-1-k)^2c/4) = \Omega(x^2hc)$ and otherwise the delay cost exceeds $\Omega(xh\sqrt{hc})$. Thus, we can conclude our proof by

$$\rho \in O((x\sqrt{h}hc)(hc + hx\sqrt{h} + xhc)) \in O(\sqrt{h}).$$

□

LEMMA 4.8. *Consider a transmission of $\mathcal{OPT}$ and assume that $\mathcal{AGG}$ merges μ_i messages into one message at hop distance i from the leaf. Compared to a sequence where no messages are merged, $\mathcal{AGG}$ can reduce its cost by at least $\Omega(\mu_i \cdot c \cdot (h-i))$.*

PROOF. If the μ_i messages are sent to the root separately, the online algorithm pays $\mu_i(h-i)c$ communication cost for the transmissions between the node at distance i to the root node. By merging theses messages the cost for the transmission amounts to $(h-i)c$, consequently the reduction is in $\Omega(\mu_i \cdot c \cdot (h-i))$. □

LEMMA 4.9. *Consider a transmission of $\mathcal{OPT}$ and assume that $\mathcal{AGG}$ merges μ_i messages into one message at hop distance i from the leaf. Compared to a sequence where no messages are merged, $\mathcal{OPT}$ can reduce its cost by at most $O(\mu_i \cdot c \cdot (h-i))$.*

PROOF. We prove this lemma in two steps. First, we assume that in the execution of the online algorithm $\mathcal{AGG}$ the μ_i messages departed from the leaf node separately and did not merge with any messages before reaching the node at distance i. Thereafter we show how to generalize our results for arbitrary merges.

Consider the μ_i messages that $\mathcal{AGG}$ merges at node i. We denote the size of the messages under scrutiny by $n_1, ..., n_{\mu_i}$. Without loss of generality, assume that for $1 \leq j \leq \mu_i$, all n_j packets arrive at the same time: if packets arrived dispersed over time, only the delay cost of $\mathcal{AGG}$ would decrease. In the following, let X_s denote that variable X is considered in the scenario where $\mathcal{AGG}$ does not have any merges, and X_m denote a variable in the other scenario. Let δ^l denote the time interval between two consecutive arrival time slots of the set of n_l and the set of n_{l+1} packets. For sequences where $\mathcal{AGG}$ does not merge any packets, $\mathcal{OPT}$'s delay cost is given by

$$dc_s^{\mathcal{OPT}} = \sum_{l=1}^{\mu_i-1}\sum_{j=l}^{\mu_i-1} n_l\delta_j^s.$$

In order to guarantee that consecutive messages cannot merge by the definition of $\mathcal{AGG}$ we have $\delta_s^l = \max(\kappa_s, 1)$, where $\kappa_s = h/2\left(\lceil 2c/n_l\rceil - \lceil 2c/n_{l+1}\rceil\right) + \lceil c/n_l\rceil - \lceil c/n_{l+1}\rceil$.

If we assume the first merge operation of μ_i messages to happen at depth $h-i$, it must hold that $n_l < n_{l+1}$ and we can compute a lower bound for the reduce time interval between two messages. Observe that in order to ensure that messages do not merge too early, it must hold that $\delta_m^l > \max(\kappa_m, 1)$, where $\kappa_m = (i-1)/2\left(\lceil 2c/n_l\rceil - \lceil 2c/n_{l+1}\rceil\right) + \lceil c/n_l\rceil - \lceil c/n_{l+1}\rceil$.

Thus, $\mathcal{OPT}$ can reduce its delay cost by at most:

$$\begin{aligned}\Delta dc^{\mathcal{OPT}} &= dc_s^{\mathcal{OPT}} - dc_m^{\mathcal{OPT}}\\ &\leq \sum_{l=1}^{\mu_i-1} n_l \sum_{j=l}^{\mu_i-1}\left[\delta_j^s - \delta_j^m\right]\\ &\leq \sum_{l=1}^{\mu_i-1}\sum_{j=l}^{\mu_i-1} n_l \frac{h-i+1}{2}\left[\left\lceil\frac{2c}{n_l}\right\rceil - \left\lceil\frac{2c}{n_{l+1}}\right\rceil\right]\\ &= \frac{h-i+1}{2}\sum_{l=1}^{\mu_i-1} n_l\left[\left\lceil\frac{2c}{n_j}\right\rceil - \left\lceil\frac{2c}{n_{\mu_i}}\right\rceil\right]\\ &\leq \frac{h-i+1}{2}(\mu_i-1)3c.\end{aligned}$$

Hence, the claim holds for non-recursive merges. To see why the claim also holds for repeated merges, consider again $\mathcal{AGG}$'s messages which are aggregated by $\mathcal{OPT}$, and consider the locations in the chain topology where they merge the first time. If this is the only time a merge occurs, we are done. Otherwise, regard the node where the merge occurs as a new leaf node and consider the remaining chain until the root. For this network, the same arguments apply, and hence, the claim also holds in this case. □

LEMMA 4.10. *Fix a sequence of packet arrivals such that on a chain graph $\mathcal{AGG}$ sends ν messages individually to the root and merges μ messages at distance i from the leaf the competitive ratio is $O(\min(\sqrt{h}, c))$.*

PROOF. Fix a sequence of packet arrivals such that on a chain graph $\mathcal{AGG}$ sends ν messages individually to the root and merges μ messages at distance i from the leaf. If the packets that form the μ messages the online algorithm merges arrive before the ν messages sent to the root separately the claim follows directly from Lemmas 4.8 and 4.9. Otherwise we have to take the delay cost the ν messages can save since the to be merged messages can arrive closer to each other into account. Applying the Lemmas 4.8 and 4.9 leads to a total cost for $\mathcal{AGG}$ of less than $3\nu hc + 3hc + 3\mu ic$ and the total cost for $\mathcal{OPT}$ amounts to at least $\Omega\left(hc + \min(\nu^2 c, \nu hc) + \mu ic\right)$. As in the proofs above we can assume without loss of generality that $(\nu + \mu) \in \omega(\sqrt{h})$. If $\nu h < \nu^2$ the competitive ratio is in $O(1)$, otherwise the ratio is at most $O(\frac{\nu h+\mu}{h+\nu^2+\mu})$. Assume ν to be lager than μ. This implies that $\nu > \sqrt{h}$ and hence the ratio is $O(\nu h/\nu^2) = O(\sqrt{h})$. If μ is larger than ν, we have a ratio of $O(hnu/(h+nu^2))$, which is $O(\sqrt{h})$. □

The online algorithm performing several merge operations cannot increase the competitive ratio, and together with Theorem 4.1, it follows that the competitive ratio is at most $O(\min(\sqrt{h}, c))$ on chain graphs. □

We now show that $\mathcal{AGG}$ is asymptotically optimal for all oblivious online algorithms, i.e., we derive a lower bound for chain networks of $\Omega(\min(\sqrt{h}, c))$. For this lower bound, we consider a chain network with $h+1$ nodes. Let ALG denote any oblivious online algorithm, and assume that packets arrive one-by-one: at time 0, a packet p arrives at the leaf node l. The next packet arrives at the leaf exactly when ALG sends the packet at l. Let w denote the time a packet waits at l, and observe that the same waiting time holds for all nodes on the way from the leaf to the root. Thus, the total waiting time per packet is hw, and the communication cost is hc: $cost^{ALG} = hw + hc$. We now derive an upper bound on the optimal algorithm's cost for this sequence. We partition the packets into blocks of size $\sqrt{h}$, i.e., one message contains

$\sqrt{h}$ packets. Thus, the communication cost per packet of this algorithm is $hc/\sqrt{h} = \sqrt{h}c$. The delay cost per message at the leaf is $\sum_{i=1}^{\sqrt{h}-1} iw \in \Theta(hw)$. In addition, each packet experiences one unit of delay per hop on the way up to the root. Thus, the optimal cost per packet is $cost^{\mathcal{OPT}} \leq \Theta(\sqrt{h}c + w\sqrt{h} + h)$. Therefore, for this sequence, it asymptotically holds that

$$\rho \geq \frac{hc + hw}{(\sqrt{h}c + w\sqrt{h} + h} = \frac{h(c+w)}{\sqrt{h}(c+w) + h}.$$

The lower bound follows from distinguishing three cases. If h and $c+w$ are asymptotically equivalent, the above expression becomes $\Omega(\sqrt{h})$. If h is asymptotically larger than $c+w$, the best oblivious algorithm which chooses w as small as possible yields a lower bound of $\Omega\left(c\right)$. Finally, if $h < c+w$, we have $\Omega(\sqrt{h})$.

THEOREM 4.11. *The competitive ratio of any oblivious algorithm is at least*

$$\rho = \frac{cost^{ALG}}{cost^{\mathcal{OPT}}} \in \Omega\left(\min(\sqrt{h}, c)\right).$$

Discussion. Our findings can be compared to the analysis presented in [8]. Their $\Omega(\sqrt{h})$ lower bound holds for arbitrary oblivious algorithms on *trees*. In this paper, we have shown that this upper bound is too pessimistic, as general trees are inherently more difficult than chain topologies, and that the lower bound can be increased to $\Omega(\min(h, c))$. For chain networks, we have generalized their result to arbitrary edge cost, yielding a lower bound of $\Omega(\min(\sqrt{h}, c))$, which is proved tight by $\mathcal{AGG}$.

5. VALUE SENSITIVE AGGREGATION

Before we conclude this paper, we introduce a novel model for online event aggregation. In contrast to the aggregation model studied so far, this model is more appropriate in scenarios where the to be delivered information is not binary (e.g., event messages) but where arbitrary *value* aggregations need to be performed at the root. Take wireless sensor networks as a motivating example: A set of nodes measures the temperature at a certain outdoor location, and the root is interested to have up-to-date information on these measurements. Thereby, larger value changes are more important and should be propagated faster to the root, e.g., such that an alarm can be raised soon in case of drastic environmental changes.

In the following, we consider a most simple topology: a network consisting of a leaf and a sink. Let the value measured by the leaf l at time t be l_t. We assume that the leaf node can only send the value it currently measures. The root node's latest value of node l at time t is denoted by r_t. We seek to minimize the following optimization function: $cost = M \cdot c + \sum_t |l_t - r_t|$ where M is the total number of message transmissions and c the cost per transmission, i.e., $M \cdot c$ is the total communication cost.

Typically, the values measured by a sensor node do not change arbitrarily, but there is a bound on the maximal change per time unit. In the following, we assume that the value measured by a node changes by at most Δ per round. Moreover, we assume that the sensor nodes can only measure discrete quantities, and that the difference between two measured values is at least ϵ.

First observe that there exists a simple optimal (offline) algorithm which employs dynamic programming. $\mathcal{OPT}$ exploits the following optimal substructure: For all time slots i, we compute the minimal cost given that the node sends its value at time i; in order to find this minimum cost, we consider each optimal last transmission $j < i$ and add the inaccuracy cost which accrued at the root node between the two transmissions j and i. Observe that it is not necessary to iterate over all time slots i, because an optimal algorithm only sends a value immediately after it has changed, i.e., we only have to consider the time slots with value changes. Hence we can construct an array $OPT[\cdot]$ of size λ, where λ is the total number of value changes at the leaf node. We set $OPT[0] = 0$, as we assume that initially, the root stores the correct value. The remaining matrix entries are then computed as follows:

$$OPT[i] = \min_{j<i}\left(OPT[j] + c + \sum_{t=j+1}^{i} |l_t - l_j|\right).$$

We have the following theorem.

THEOREM 5.1. *In a link network, the optimal aggregation strategy can be computed in time $O(\lambda^3)$, where λ is the number of value changes at the leaf.*

We propose the following online algorithm $\mathcal{AGG}$, which can be seen as a generalization of the algorithm presented in the previous section: The leaf l sends the value it currently measures if and only if $\sum_{t=\tau}^{T} |l_t - l_\tau| \geq c$, where τ is the last time l has transmitted its value and T is the current time.

For the analysis of $\mathcal{AGG}$, we consider the time intervals between two transmissions of $\mathcal{AGG}$. For each such interval, we can bound the competitive ratio yielding an overall competitive ratio. We first need the following helper lemma.

LEMMA 5.2. *Let ρ be the competitive ratio of $\mathcal{AGG}$ when $\mathcal{AGG}$'s delay cost is c in each time interval I, then $3\rho/2$ is an upper bound on the total competitive ratio.*

PROOF. First observe that $\mathcal{AGG}$ can have a larger delay cost than c in an interval, e.g., if in a round where the accumulated delay cost is $c - \epsilon$ for an arbitrarily small $\epsilon > 0$ there is a large value change of size Δ at the leaf. Hence, the online algorithm's delay in any interval is at most $2(c-\epsilon) + \Delta$. Consider an interval I where the online algorithm's delay cost is $2(c-\epsilon) + k$ for some $k \leq \Delta$. Compared to the case studied so far, $\mathcal{AGG}$'s delay cost will increase by at most $k + c - 2\epsilon$. However, due to the large value change, we know that the optimal algorithm's delay cost must increase by at least k as well. The new competitive ratio ρ' is hence

$$\begin{aligned}
\rho' &= \frac{CC'_{\mathcal{AGG}} + DC'_{\mathcal{AGG}}}{CC'_{\mathcal{OPT}} + DC'_{\mathcal{OPT}}} \\
&\leq \frac{cc^{\mathcal{AGG}} + dc^{\mathcal{AGG}} + k + c - 2\epsilon}{cc^{\mathcal{OPT}} + dc^{\mathcal{OPT}} + k} \\
&= \frac{cc^{\mathcal{AGG}} + dc^{\mathcal{AGG}}}{cc^{\mathcal{OPT}} + dc^{\mathcal{OPT}} + k} + \frac{k + c - 2\epsilon}{cc^{\mathcal{OPT}} + dc^{\mathcal{OPT}} + k} \\
&< \rho + \frac{c - 2\epsilon}{cc^{\mathcal{OPT}} + dc^{\mathcal{OPT}}} \\
&< \rho + \frac{c - 2\epsilon}{(cc^{\mathcal{AGG}} + dc^{\mathcal{AGG}})/\rho} \\
&< \rho + \frac{c - 2\epsilon}{2c/\rho} \\
&< \frac{3\rho}{2}.
\end{aligned}$$

□

THEOREM 5.3. *The competitive ratio of $\mathcal{AGG}$ is*

$$\rho = \frac{cost^{\mathcal{AGG}}}{cost^{\mathcal{OPT}}} \in \Theta(c/\epsilon),$$

where c is the link cost and ϵ is the minimum difference between two values.

PROOF. We first prove that $\rho \in O(c/\epsilon)$, and subsequently show that $\rho \in \Omega(c/\epsilon)$.

Proof for $\rho \in O(c/\epsilon)$: First, the ratio is computed under the assumption that the delay cost of $\mathcal{AGG}$ is exactly c; we then apply Lemma 5.2. We classify the possible types of intervals I between two sending events of the online algorithm and consider them separately. Observe that for any interval where the optimal algorithm $\mathcal{OPT}$ transmits, the competitive ratio is at most 2, as $\mathcal{OPT}$ has cost at least c and the online algorithm $\mathcal{AGG}$ has communication cost c and delay cost c. It remains to examine the situations where $\mathcal{OPT}$ does not send.

Assume that at the beginning of this interval, $\mathcal{AGG}$ sends the value A_0, and at the end, it sends the value A_1. Furthermore we denote the optimal algorithm's value at the root at the beginning and at the end of this interval by O_0 and O_1, respectively. We define $\delta_0 := |A_0 - O_0|$ and $\delta_1 := |A_1 - O_1|$ and examine all possible cases under the assumption that the delay cost of $\mathcal{AGG}$ is c for every interval.

Case $\delta_0 = \delta_1 = 0$*:* If $\mathcal{OPT}$ has no transmission, it must have the same delay cost in this interval as $\mathcal{AGG}$, because the initial and final values are the same, and hence $\rho_I \leq 2$.

Case $\delta_0 = 0, \delta_1 \neq 0$*:* If $\mathcal{OPT}$ has no transmission, it must have at least the same delay cost as $\mathcal{AGG}$ in I , thus $\rho_I \leq 2$.

Case $\delta_0 \neq 0, \delta_1 = 0$*:* If $\mathcal{OPT}$ has no transmission, it incurs at least delay cost δ_0 in the first time slot, as $\mathcal{AGG}$ sends the correct value at the beginning of the interval. Hence, $\rho_I \leq 2c/\delta_0$.

Case $\delta_0 \neq 0, \delta_1 \neq 0$*:* In this case, $\mathcal{OPT}$ has delay cost δ_0 as well yielding $\rho_I \leq 2c/\delta_0$.

Thus, for each of these intervals, $\rho_I \leq 2c/\delta_0$. It must hold that $\delta_0 > \epsilon$, and the claim follows by applying Lemma 5.2.

Proof for $\rho \in \Omega(c/\epsilon)$: Consider the following sequence of values at the leaf node:

$$0, \epsilon^{c/\epsilon}, 0^{c/\epsilon-1}, -\epsilon, 0^{c/\epsilon-1}, \epsilon, 0^{c/\epsilon-1}, -\epsilon, \ldots$$

where α^β denotes that the value α remains for β rounds. Observe that for each subsequence $(0^{c/\epsilon-1}, -\epsilon, 0^{c/\epsilon-1}, \epsilon)$, 2ϵ is an upper bound on $\mathcal{OPT}$'s delay cost: It is the total delay cost if there are no transmissions at all in the entire sequence. In contrast, $\mathcal{AGG}$ has $2c$ delay cost plus two transmissions. That is, neglecting the cost of the first time slot, we have $\rho \geq 4c/2\epsilon = 2c/\epsilon$. □

6. CONCLUSION

This paper investigated an online aggregation problem which can be regarded as a generalization of the classic *ski-rental problem* to trees. This generalization is motivated by the increasing popularity of wireless sensor networks where a trade-off between event notification time and transmission cost exists. We have studied a simple distributed algorithm which achieves a competitive ratio of $\Theta(\min(h, c))$ in case of general trees and $\Theta(\min(\sqrt{h}, c))$ in case of chains. Apart from the efficiency criterion, this algorithm is attractive as it poses minimal hardware requirements on the sensor nodes (low memory and computational requirements). In addition to binary event aggregation, this paper has initiated the analysis of a new model where the root is interested in the nodes' *values* (e.g., the measured temperature or humidity).

We believe that both problems still pose interesting questions for future research. E.g., the exploration of asynchronous models, the study of non-oblivious algorithms, or algorithms which have a limited amount of information about the states of their neighbors, can yield deeper insights into the event aggregation problem and may also be useful in other applications. Moreover, it would be interesting to examine whether the techniques used in [7] can also be applied to our value aggregation problem in order to reduce the competitive ratio further by randomization.

Acknowledgments

We thank Thibaut Britz and Pascal von Rickenbach from ETH Zurich for interesting discussions.

7. REFERENCES

[1] S. Albers and H. Bals. Dynamic TCP Acknowledgement: Penalizing Long Delays. In *Proc. 14th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 47–55, 2003.

[2] L. Becchetti, P. Korteweg, A. Marchetti-Spaccamela, M. Skutella, L. Stougie, and A. Vitaletti. Latency Constrained Aggregation in Sensor Networks. In *Proc. European Symposium on Algorithms (ESA)*, pages 88–99, 2006.

[3] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[4] C. Brito, E. Koutsoupias, and S. Vaya. Competitive Analysis of Organization Networks or Multicast Acknowledgement: How Much to Wait? In *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 627–635, 2004.

[5] D. R. Dooly, S. A. Goldman, and S. D. Scott. On-line Analysis of the TCP Acknowledgment Delay Problem. *Journals of the ACM*, 48(2):243–273, 2001.

[6] J. S. Frederiksen and K. S. Larsen. Packet Bundling. In *Proc. 8th Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 328–337, 2002.

[7] A. R. Karlin, C. Kenyon, and D. Randall. Dynamic TCP Acknowledgement and Other Stories About $e/(e-1)$. In *Proc. 33rd ACM Symposium on Theory of Computing (STOC)*, pages 502–509, 2001.

[8] S. Khanna, S. Naor, and D. Raz. Control message aggregation in group communication protocols. In *Proc. of the International Colloquium of Automata, Languagues and Programming (ICALP)*, 2002.

[9] P. Korteweg, A. Marchetti-Spaccamela, L. Stougie, and A. Vitaletti. Data Aggregatcion in Sensor Networks: Balancing Communication and Delay Costs. In *Proc. Colloquium on Structure, Information, Communication, and Complexity (SIROCCO)*, 2007.

[10] B. Krishnamachari, D. Estrin, and S. Wicker. The Impact of Data Aggregation in Wireless Sensor Networks. In *Proc. Int. Conference on Distributed Computing Systems Workshops (ICDCSW)*, 2002.

[11] C. H. Papadimitriou. Computational Aspects of Organization Theory. In *Proc. European Symposium on Algorithms (ESA)*, 1996.

[12] C. H. Papadimitriou and E. Servan-Schreiber. The Origins of the Deadline: Optimizing Communication in Organizations. In *Manuscript*, 1999.

[13] I. Solis and K. Obraczka. The Impact of Timing in Data Aggregation for Sensor Networks. In *Proc. IEEE Int. Conference on Communications (ICC)*, 2004.

[14] W. R. Stevens. *TCP/IP Illustrated, Vol.1: The Protocols*. Addison-Wesley, 1994.

[15] P. von Rickenbach and R. Wattenhofer. Gathering Correlated Data in Sensor Networks. In *Proc. ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, 2004.

[16] O. Younis and S. Fahmy. An Experimental Study of Routing and Data Aggregation in Sensor Networks. In *Proc. IEEE Int. Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 2005.

[17] Y. Yu, B. Krishnamachari, and V. K. Prasanna. Energy-Latency Tradeoffs for Data Gathering in Wireless Sensor Networks. In *Proc. Joint Conference of the IEEE Computer and Communication Societies (INFOCOM)*, 2004.