Discrete Mathematics and Theoretical Computer Science DMTCS vol. **12:1**, 2010, 38–58

Tight Bounds for Delay-Sensitive Aggregation†

Yvonne Anne Pignolet[1] Stefan Schmid[2] Roger Wattenhofer[3]

[1] *IBM Research, Zurich Research Laboratory, Switzerland.*‡
[2] *T-Labs, TU Berlin, Germany.*‡
[3] *Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland.*‡

received August 11, 2009, *revised December 14, 2009*, *accepted January 19, 2010*.

This article studies the fundamental trade-off between delay and communication cost in networks. We consider an online optimization problem where nodes are organized in a tree topology. The nodes seek to minimize the time until the root is informed about the changes of their states and to use as few transmissions as possible. 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, we introduce a model for value-sensitive aggregation, where the cost depends on the number of transmissions and the error at the root.

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 article, we study such a case known as *distributed aggregation* (a.k.a. *multicast aggregation*). Nodes of a large distributed network may sense potentially interesting data which they are to report to a central authority (sink). 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

‡*Email:* `yvo@zurich.ibm.com` `stefan@net.t-labs.tu-berlin.de` `wattenhofer@tik.ee.ethz.ch`

†A preliminary version of this work has been published at the 27th ACM Annual Symposium on Principles of Distributed Computing (PODC), 2008 [OSW08].

1365–8050 © 2010 Discrete Mathematics and Theoretical Computer Science (DMTCS), Nancy, France

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. 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 sink is informed about changes of measured values.

We assume 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). Data arrives at the nodes in an online (worst-case) fashion. The main challenge is to decide at what points in time the data should be forwarded to a parent in the tree.

1.1 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.

A basic problem in the design of Internet transfer protocols such as the TCP protocol concerns the acknowledgments (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 [Ste94]. 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*, which was proved to be 2-competitive [DGS98]. Karlin et al. [KKR01] 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 (where $e = 2,7182...$ is the *Euler number*).

There are many variations of the theme, e.g., Albers et al. [AB03] seek to minimize the number of acknowledgments sent plus the *maximum* delay incurred for any of these packets. They propose a $\pi^2/6$-competitive deterministic algorithm for the single link (where $\pi = 3.1415...$), which is also a lower bound for any deterministic online algorithm. Frederiksen et al. [FL02] consider deterministic and randomized algorithms for bundling packets in a single-link network. They observe that because of physical network properties, a certain minimum amount of time must elapse in-between the transmission of two packets; the trade-off is investigated that on the one hand waiting delays the transmission of the current data, and on the other hand sending immediately may delay the transmission of the next data to become available even more.

There is also much literature on aggregation in sensor networks [KEW02, KLS08, SBAS04, SO04, vRW04, YF05, YKP04]. E.g., Becchetti et al. [BKMS$^+$06] 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. More complexity results for settings with fixed deadlines have been derived in [NS09].

Korteweg et al. [KMSSV09] address a similar problem following 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 $(0, 1]$.

The paper closest to our work is by Khanna et al. [KNR02] which investigates the task of centralized and decentralized online control message aggregation on weighted tree topologies. In particular, [KNR02] 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 $\Omega(\sqrt{h})$ (an oblivious algorithm bases the decisions at each node solely upon the static local information available at the node).

In this article, 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 is 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 intrinsic properties of the algorithm, we show that for chain graphs an upper bound of $O(\min(\sqrt{h}, c))$ holds. This is asymptotically tight. Brito et al. [BKV04] extended the work of Khanna et al. by proving general upper and lower bounds for the asynchronous and the centralized ACK aggregation problem. Interestingly, the asynchronous model yields higher lower bounds in the distributed case than the synchronous ("slotted") model, e.g., the lower bound for the chain network is $\Omega(h)$ in the distributed asynchronous model, a factor of $\sqrt{h}$ greater than the bounds in the synchronous model.

Finally, our value-sensitive aggregation model is reminiscent of the recent "online tracking" work by Yi and Zhang [YZ09]. The authors study a 1-lookahead scenario where Alice outputs a function $f : \mathbb{Z}^+ \rightarrow \mathbb{Z}^d$, and Bob, knowing all values at $t \leq t_{\text{now}}$ needs to guess $f(t)$ in an online fashion. Different scenarios are examined, and competitive strategies are presented that result in small errors only and save on communication cost. Our model differs from [YZ09] as it describes a 0-lookahead model and as in their model, nodes are *forced* to send updates if the measured value differs sufficiently from the value stored at the root.

1.2 Model

The network to be considered is modeled by 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 $c \geq 1$; we assume that nodes know the value c. Besides general trees, we will also study *chain networks* (sometimes also referred to as linked lists networks or linear arrays in literature) where each node has exactly two neighbors except for the root and the leaf node which have 1 neighbor.

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 time slot, i.e., a message always has to wait one time slot at a node before being transmitted to the next node. 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 it contains. 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 cost: 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* (see, e.g., the work by Sleator and Tarjan [ST85], or the introductory book by Borodin [BEY98]) and define the (strict) *competitive ratio* ρ achieved by an online algorithm $\mathcal{ALG}$ as the delay and communication cost of $\mathcal{ALG}$ divided by the total cost of an optimal offline algorithm $\mathcal{OPT}$ that knows the event arrival sequences *a priori*.

Definition 1.1 (ρ-competitiveness) *An online algorithm $\mathcal{ALG}$ is (strictly) ρ-*competitive *compared to an optimal offline algorithm $\mathcal{OPT}$ if for all input sequences S, i.e., all possible event arrival sequences,*

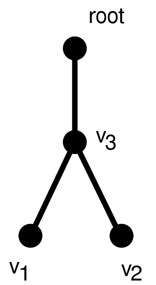
$$cost^{\mathcal{ALG}}(S) \leq \rho \cdot cost^{\mathcal{OPT}}(S).$$

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.

We focus on *oblivious* online algorithms.

Definition 1.2 (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 number of events currently stored at node v and on the arrival times of the corresponding messages at node v.*

In particular, Definition 1.2 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. Being oblivious is a desirable property of distributed algorithms, since non-oblivious algorithms need dynamic updating mechanisms—a costly operation.



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	6
$t=5$	0	0	0

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

1.3 Our Contribution

We prove that the simple algorithm introduced in [KNR02] achieves a competitive ratio of $O(\min(h, c))$ where h is the tree's height, and c is the transmission cost per edge. This improves an existing upper bound of $O(h \log(cn))$, where n is the network size. 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 article, we present an 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 $O(c/\epsilon)$ on a one-link network, where ϵ is the minimum difference between two values. We also describe an optimal offline algorithm (requiring polynomial time) for this model.

2 Oblivious Online Algorithm

This section describes and analyzes the deterministic online algorithm $\mathcal{KNR}$ presented in [DGS98, KNR02]. The algorithm is *oblivious* in the sense that decisions are reached independently of the distance to the root (cf Definition 1.2). Essentially, the event aggregation algorithm $\mathcal{KNR}$ 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 a message m at node v, we define $delay_v(m, t)$, denoting the delay associated with message m currently stored at node v at time t. Informally, it is the sum of the accumulated delay cost at node v of all the event packets the message m contains. In every time step a message remains at a node, $delay_v(m, t)$ is increased by the number of packets in the message. If a message arrives at a node where another message is already waiting to be forwarded, the two messages are merged. 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 node v at time t is defined by $delay_v(m, t) := \sum_{i=1}^{k} |m_i|(t - t_i)$.

When executing algorithm $\mathcal{KNR}$, a node v forwards a message m to its parent as soon as the current accumulated delay exceeds the transmission cost: $delay_v\,(m, t) \geq c$.

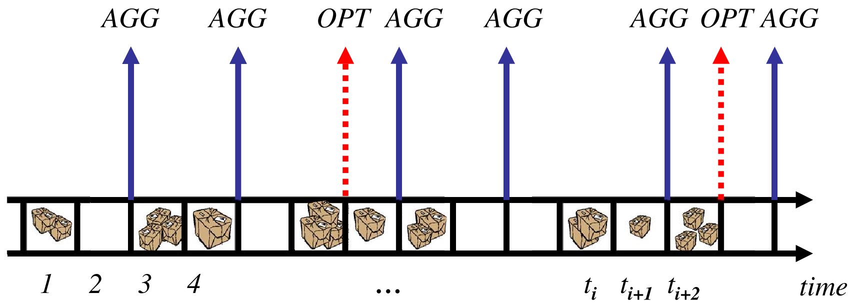


Fig. 2: Illustration of a sequence of event arrivals at a leaf and the corresponding forwarding actions of the online algorithm $\mathcal{KNR}$ for $c = 4$. In addition, the time slots where the optimal offline algorithm forwards its messages are marked as well. Note that we cannot verify in this limited view of a leaf node how much better $\mathcal{OPT}$'s decisions are because of our lack of knowledge on the situation at the other nodes.

We demonstrate the execution of $\mathcal{KNR}$ 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. In the next time slot, v_3's delay cost increases to 6, as the message from v_2 arrives with two additional packets, so there are four messages at v_3 now.

3 Tight Bound for Trees

Upper Bound

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

Theorem 3.1 *On arbitrary trees, the competitive ratio of $\mathcal{KNR}$ is at most*

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

Proof: We begin by analyzing one path from a leaf l to the root. Given a sequence of event arrivals at the leaf, we can compute when $\mathcal{KNR}$ sends messages to the leaf's parent node. We study the cost accumulated by these messages in groups of three messages. Then our results are generalized to trees.

Lemma 3.2 *For the events occurring at a given leaf l, the competitive ratio is at most $O(\min(h_l, c))$.*

Proof: Given a sequence of event arrivals at the leaf, we can compute when $\mathcal{KNR}$ sends messages to the leaf's parent node. We define m_i^A to be the i^{th} message that leaf node l, located at depth h_l, sends to its parent. Let t_0 be the first time slot, and the time slot when message m_i^A is forwarded towards the root by l is denoted by t_i. Let the total number of messages that leaf l transmits be M_l^A. If such a message m_i^A contains less than c packets it incurs a transmission cost of c and a delay cost of less than $2c$ per hop towards the root. If a message consists of more than c packets, the delay cost per hop is bounded by the number of packets. Thus, the total cost for $\mathcal{KNR}$ amounts to at most $h_l(c + \max(2c, |m_i^A|))$ for m_i^A.

In order to determine the minimum cost an offline algorithm accumulates, we divide the time into intervals of three message transmissions. More precisely, we consider time periods $[t_j, t_{j+3}]$, where $j \mod 3 = 0$. There are two possibilities for algorithm $\mathcal{OPT}$: Either it sends one or more messages in the period $[t_j, t_{j+3}]$ as well, yielding at least a transmission cost of c, or the delay accumulated by these packets is at least c. Either way, the cost accumulated by $\mathcal{OPT}$ at leaf l for these packets is at least c. In addition, the delay cost accumulating until the packets arrive at the root is at least $(h_l - 1)(|m_j^A| + |m_{j+1}^A| + |m_{j+2}^A|)$, since every packet in m_j^A, m_{j+1}^A and m_{j+2}^A incurs a delay cost of at least one for each hop towards the root. Thus, $\mathcal{OPT}$'s total cost for a period $[t_j, t_{j+3}]$ is at least

$$c + (h_l - 1)(|m_j^A| + |m_{j+1}^A| + |m_{j+2}^A|).$$

Unless $M_l^A \mod 3 = 0$, we cannot consider all messages in groups of three. If there are one or two last messages that do not belong to such a group, the respective cost of $\mathcal{OPT}$ for the period $[t_{3\lfloor M_l^A/3 \rfloor}, t_{M_l^A}]$ is at least $c + (h_l - 1)|m_{M_l^A}|$ or $c + (h_l - 1)(|m_{M_l^A}| + |m_{M_l^A - 1}|)$.

This implies that for any given sequence of event arrivals at a leaf the competitive ratio is

$$\begin{aligned}
\rho &= \frac{cost^{\mathcal{KNR}}}{cost^{\mathcal{OPT}}} \\
&\leq \frac{\sum_{j=1}^{M_l^A} h_l(c + \max(2c, |m_j^A|))}{c\lceil M_l^A/3 \rceil + \sum_{j=1}^{M_l^A} h_l |m_j^A|} \\
&\leq \frac{3M_l^A h_l c + \sum_{j=1}^{M_l^A} h_l |m_j^A|}{cM_l^A/3 + \sum_{j=1}^{M_l^A} h_l |m_j^A|} \\
&< \frac{3M_l^A h_l c}{M_l^A(c/3 + h_l)} + 1 \\
&\in O\left(\min\left(h_l, c\right)\right).
\end{aligned}$$

□

Observe that we make no assumptions on the behavior of the optimal algorithm in Lemma 3.2. Moreover, we charge the optimal algorithm $\mathcal{OPT}$ only for the transmissions over the edges between the leaves and their neighbors when bounding $\mathcal{OPT}$'s communication cost, i.e., we basically exempt the optimal algorithm from paying for any other transmission. Furthermore, we ignore the cost $\mathcal{KNR}$ could potentially save by merging messages on their way to the root: We assign the highest possible delay and communication cost to every message of $\mathcal{KNR}$. These properties allow us to extend Lemma 3.2 to trees without modification: More precisely, if we consider the longest path in the tree the statement holds for general trees as well. □

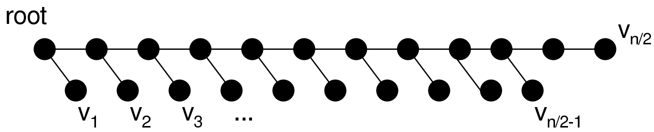


Fig. 3: Lower bound topology.

Lower Bound

We conclude our investigations of the tree network with a lower bound stating that $\mathcal{KNR}$ is asymptotically optimal compared to any *oblivious algorithm* (Definition 1.2). 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 3.3 *On trees, any oblivious deterministic online algorithm ALG has a competitive ratio*

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

Proof: Consider the tree topology depicted in Figure 3 which consists of a chain network of $n/2 + 1$ nodes, where all nodes except for the two last ones have an additional neighbor (n even). The leaf nodes are referred to by $v_1, ..., v_{n/2}$.

We compute the minimum delay and communication cost any oblivious online algorithm ALG will accrue for input sequence where all leaves simultaneously get one packet. Since ALG is oblivious, according to Definition 1.2, each leaf node v_i will send the packet to its parent after waiting for w time slots, where the packet arrives at time $w + 1$. (The value of $w \geq 0$ depends on the chosen algorithm; observe that w is the same for all nodes, because we assume ALG to be oblivious.) From there, the packets leave at time $2w + 2$. This process is repeated until all packets reach the root. Generally, the packet of leaf node v_i will arrive at a node at distance j from v_i at time $j + jw$, and will stay there for w time slots. Observe that the packets of two nodes v_{i-1} and v_i are never merged into one message. Thus, ALG has communication cost in the order of $\Theta\left(h^2 c\right)$ and delay cost in the order of $\Theta\left(h^2\right)$.

The optimal algorithm $\mathcal{OPT}$ has to send at least one message over each edge, resulting in a communication cost of $\Theta\left(ch\right)$. The minimum delay cost until all packets reach the root is $\Theta\left(h^2\right)$. If all the packets are merged into one message on their way to the root, the delay and communication cost is in this order of magnitude. Thus we have the following (asymptotic) competitive ratio:

$$\rho = \frac{cost^{ALG}}{cost^{\mathcal{OPT}}} \in \Omega\left(\frac{h^2 + h^2 c}{2hc + h^2}\right).$$

□

This result implies that $\mathcal{KNR}$ is asymptotically optimal in the sense that there exists no oblivious online algorithm achieving a better performance.

Discussion

Note that the above analysis for upper and lower bounds carries over to the model where events can appear at *any* node, not only at the leaves: The case where events occur at inner nodes can be reduced to a problem instance where events only occur at leaves by simply creating a new tree with additional leaves attached to any node where events occur; we have to map the arrival time slots to earlier time slots, ensuring that they arrive at the inner nodes at the correct time.

Theorem 3.1 can be compared to the results obtained in [KNR02]. There, an upper bound of $O(h \log \alpha)$ is derived for the competitive ratio of $\mathcal{KNR}$, where α is the total communication cost (sum of edge weights) of the tree. If all edges have a weight c, this translates into $O(h \log(nc))$, which is $O\left(h^2 \log c\right)$ in balanced binary trees. In this case, the analysis presented above 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 there always remains a factor in the order of $\Omega(\log(\max(h, c))$: Let us only take the edges on the longest path into account for the upper bound of [KNR02]. Thus the bound reduces to $O(h \log(hc))$. If $c > h$ this is $O(h \log c)$ whereas the bound presented here is $O(h)$. If $c < h$ the bound from [KNR02] gives $O(h \log h)$ and our analysis results in $O(c)$. Technically the difference between our approach and the approach in [KNR02] is the way we sum up the delay cost of the algorithms. In [KNR02], the sum of the delay cost per packet for $\mathcal{KNR}$ is expressed in terms of the delay cost per packet of the optimal offline algorithm: By a potential function argument, it is shown that the number of packets that reach the root later than in the optimal case decreases quickly over time. This implies that the total delay cost of the online algorithm cannot be much larger than the total delay cost of the offline algorithm. In contrast, in our approach we consider the absolute cost the online and offline algorithms accumulate. Like in [KNR02], there is no need to study merge behavior explicitly.

4 Tight Bound for Chains

Upper Bound

The previous section has already discussed the implications of our tree analysis on the performance of $\mathcal{KNR}$ on chain networks. In the following, we go one step further and take a closer look at chains. We show by a more detailed analysis that the bound derived there can be improved. In particular, we leverage the fact that the algorithm can merge messages and forward them together, i.e., save on transmission cost. Moreover, a merged message moves faster towards the root since it now contains more packets and reaches the forwarding threshold in fewer time slots than the two separate messages consisting of fewer packets. Concretely, we prove that on the chain topology, $\mathcal{KNR}$ is $O(\min \sqrt{h}, c)$-competitive, and that no oblivious algorithm can be less than $\Omega(\min(c, \sqrt{h}))$-competitive.

Theorem 4.1 *In chain graphs, the competitive ratio of $\mathcal{KNR}$ is*

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

Proof: Since a chain is a tree, we already know that $O(\min(c, h))$ and hence $O(c)$ is an upper bound on the competitive ratio due to Theorem 3.1. It remains to prove that $O(\sqrt{h})$ is an upper bound on the competitive ratio for chain networks. We use the following lemmata repeatedly.

Lemma 4.2 *When executing $\mathcal{KNR}$, a message consisting of n_i packets that arrive at the leaf at the same time reaches the node at distance l from the leaf after $l\lceil c/n_i\rceil$ time slots if the message does not merge with any other message.*

Proof: Let t_j denote the time the message spends at depth h_j, i.e., at the j^{th} node of its path to the root. Since $\mathcal{KNR}$ is oblivious and no messages are merged, $t_j = \lceil c/n_i\rceil$ is the same for every node of the chain. Thus the number of time slots the message spends on a path of length l is $\sum_{j=0}^{l-1} t_j = l\lceil c/n_i\rceil$. □

Lemma 4.3 *If $\mathcal{KNR}$ does not merge any messages then there are at least*

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

time slots between the arrival of the packets of m_i^A and the arrival of the packets of m_{i+1}^A at the leaf node, if m_{i+1}^A contains more packets than m_i^A.

Proof: To ensure that m_i^A reaches the root separately even though it is followed by a larger message, m_i^A has to have arrived at the root before m_{i+1}^A arrives at the node in depth 1. By Lemma 4.2 we know that m_i^A reaches the root $h\lceil c/n_i\rceil$ time slots after its arrival at the leaf. If m_{i+1}^A arrives at the leaf δ_i time slots after the departure of m_i^A then it reaches the node in distance 1 to the root$(h-1)\lceil c/n_{i+1}\rceil$ time slots later. Hence we can derive the inequality $h\lceil c/n_i\rceil < \delta_i + (h-1)\lceil c/n_{i+1}\rceil$ and the claim follows. □

Since a smaller message moves at a lower speed, a message m_{i+1} containing n_{i+1} packets can never catch up with a message m_i with n_{i+1} or more packets, even if there is only one time slot between their departure and arrival.

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

Let m_i^A denote the i^{th} message of $\mathcal{KNR}$ and let n_i be the number of packets contained in message m_i^A. Without loss of generality, we assume that for each message m_i^A of $\mathcal{KNR}$, all n_i packets arrive simultaneously, since an optimal offline algorithm $\mathcal{OPT}$ would clearly incur the same or higher delay cost if packets arrived dispersed over time. We will be referring to this assumption as **Assumption 1**.

Lemma 4.4 *Given a sequence of packet arrivals where $\mathcal{KNR}$ does not merge any messages, the ratio between the cost of $\mathcal{KNR}$ and the cost of $\mathcal{OPT}$ is maximized if the number of time slots between the arrival of the packets of m_i^A and the packets of m_{i+1}^A is*

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

Proof: Due to Lemma 4.3, the minimum number of time slots between the arrival of the packets of m_i^A and m_{i+1}^A is $h\lceil c/n_i\rceil - (h-1)\lceil c/n_{i+1}\rceil$. For $\mathcal{KNR}$, the delay cost and the communication cost is neither reduced nor increased if there are more time slots in-between the arrivals. If $\mathcal{OPT}$ merges two or messages of $\mathcal{KNR}$, the delay cost is increased if there are more time slots between the arrival of packets of consecutive messages. Hence, the claim follows, as the competitive ratio is maximized when $\mathcal{OPT}$'s cost is minimized. □

Thus we can assume without loss of generality that there are exactly $h\lceil c/n_i \rceil - (h-1)\lceil c/n_{i+1} \rceil$ time slots between the arrival of consecutive messages thanks to Lemma 4.4. We will be referring to this assumption as **Assumption 2**.

Lemma 4.5 *For sequences where $\mathcal{KNR}$ does not merge any messages, the competitive ratio of $\mathcal{KNR}$ is $O(\sqrt{h})$ on chain graphs.*

Proof: We pick a transmission of the optimal offline algorithm $\mathcal{OPT}$ and compare the cost accumulated for the packets within this message to the cost the online algorithm $\mathcal{KNR}$ incurs for the same packets. We show that the competitive ratio for each such transmission is $O(\sqrt{h})$ by proving an upper bound on the cost for the online algorithm $\mathcal{KNR}$ followed by a lower bound on the cost the optimal algorithm accrues.

Let us consider the case where $\mathcal{OPT}$'s message m^O consists of packets distributed over z messages m_i^A by the online algorithm $\mathcal{KNR}$. In other words, we assume that for a certain number of packets, the online algorithm sends messages m_i^A where $i \in \{1, \ldots, z\}$ and $\mathcal{OPT}$ sends one message m^O. If there are any packets in the first or the last message of $\mathcal{KNR}$ that the optimal algorithm $\mathcal{OPT}$ does not include in m^O, we ignore them when computing the delay and communication cost for $\mathcal{OPT}$. For $i = 1$ or $i = z$ we set n_i to the number of packets that are in m_i^A and in m^O . When sending the z messages individually to the root, $\mathcal{KNR}$ incurs a transmission cost of $cc^{\mathcal{KNR}} = zhc$ and a delay cost of $dc^{\mathcal{KNR}} < 2zhc$. After having bounded the cost for $\mathcal{KNR}$ we now turn to the minimum cost an offline algorithm faces. Clearly, the transmission of m^O entails a communication cost of $cc^{\mathcal{OPT}} = ch$ for $\mathcal{OPT}$. As a next step we want to determine the minimum delay cost accumulated by the z messages m_i^A, which are merged into message m^O by the optimal algorithm.

We consider two cases: a) the minimum delay cost if the majority of messages is large, i.e., contains more than $c/2$ packets, and b) the minimum delay cost if the majority of the packets is small, i.e., consists of at most $c/2$ packets. We show that in both cases the delay cost of the optimal algorithm is $\Omega(cz^2)$.

a) If more than $z/2$ messages contain more than $c/2$ packets, we ignore all other messages and focus on the messages with $n_i > c/2$. Let δ_j denote the number of time slots between the arrival of message m_j^A and the arrival of m_{j+1}^A. Assume there are y messages with $n_i > c/2$. They are responsible for a total delay cost of at least $dc^{\mathcal{OPT}} > \sum_{i=1}^{y} n_i \sum_{j=i}^{y} \delta_j$. Even when $\delta_j = 1$ for all j, this sum amounts to

$$dc^{\mathcal{OPT}} > \sum_{i=1}^{y} n_i \sum_{j=i}^{y} \delta_j \geq \sum_{i=1}^{y} \frac{c}{2}(y-i) \in \Omega(y^2 c).$$

Thus this sum is $\Omega(z^2 c)$, because $y > z/2$.

b) If the majority of messages m_i^A contain at most $c/2$ packets, we ignore the other messages. For simplicity's sake we let z denote the number of messages with $n_i \leq c/2$. Let λ_i denote the time period between the arrival of the n_i packets and the time slot when the corresponding message m_i^A departs from the leaf. Moreover, let δ_i be the time period after the message has departed from the leaf until the next set of n_{i+1} packets arrives. See Figure 4 for an example.

The total sum of the delay cost $\mathcal{OPT}$ accumulates at the leaf for all messages m_i^A is given by

$$dc^{\mathcal{OPT}} = \sum_{i=1}^{z-1} n_i \sum_{j=i}^{z-1} (\lambda_j + \delta_j) .$$

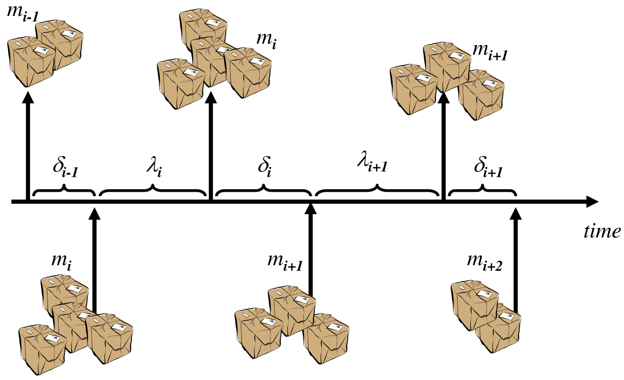


Fig. 4: A sequence of message arrivals and departures for $\mathcal{KNR}$ at a leaf. The message m_i^A is forwarded after δ_i time slots at the leaf. δ_i time slots later the next message arrives.

The following two key facts enable us to derive a bound for the delay cost the optimal algorithm incurs by waiting until the packets of all messages m_i^A have arrived at the leaf:

Fact 1 If the number of packets between two messages differs by a factor of at most two, the time they spend at the leaf differs by the same factor. More formally, if $2n_i > n_j,\ j > i$ then, $\lambda_j = \lceil c/n_j \rceil > c/2n_i \geq \lambda_i/2$.
Fact 2 If the difference between the size of two messages is larger than a factor of two, we can apply Lemma 4.3 to compute the minimum number of time slots between their arrival that ensures that a larger message cannot catch up with a smaller message on its path to the root. It holds that if $2n_i < n_j,\ j > i$ then the number of time slots m_j^A arrives later than m_i^A is at least

$$\begin{aligned} h\left\lceil \frac{c}{n_i} \right\rceil - (h-1)\left\lceil \frac{c}{n_j} \right\rceil &\geq h\frac{c}{n_i} - (h-1)\,(\frac{c}{n_j}+1) \\ &> \frac{2hc-(h-1)(c-2n_i)}{2n_i} \\ &> \frac{hc}{4n_i}, \end{aligned}$$

as $n_i < c/4$ due to our assumption that the largest message contains at most $c/2$ packets and no messages are merged after they leave the leaf node.

We construct a binary vector a where $a_i = 1$ if there exists a message m_j^A with $j > i$ and $2n_i < n_j$. Using this vector we can rewrite the delay cost $dc^{\mathcal{OPT}}$ the optimal algorithm incurs at the leaf.

$$
\begin{aligned}
dc^{\mathcal{OPT}} &= \sum_{i=1}^{z-1} n_i \sum_{j=i}^{z-1} (\lambda_j + \delta_j) \\
&> \sum_{i=1}^{z-1} n_i \left(a_i \frac{hc}{4n_i} + (1-a_i) \sum_{j=i}^{z-1} \frac{c}{2n_i} \right) \qquad (1) \\
&> \sum_{i=1}^{z-1} \left(\frac{a_i hc}{4} + \frac{(1-a_i)(z-1-i)c}{2} \right) \\
&= \frac{c}{4} \sum_{i=1}^{z-1} \left(a_i(h-2z+2+2i) + 2z-2-2i \right). \qquad (2)
\end{aligned}
$$

In (1), we used Facts 1 and 2 to bound $\sum_{j=i}^{z-1} \lambda_j + \delta_j$. If $a_i = 1$, we apply Fact 2 and the sum is replaced by the number of time slots until the large message responsible for $a_i = 1$ arrives, in the other case, we have a lower bound for λ_j due to Fact 1.

If $h > z-2$ the sum in (2) is minimized if $a_i = 0$ for all messages. In this case it holds that

$$
\begin{aligned}
dc^{\mathcal{OPT}} &> \frac{c}{4} \sum_{i=1}^{z-1} (2z-2-2i) \\
&= \frac{c}{4}(z-1)(2z-2-z) \\
&> \frac{c}{4}(z-2)^2 \\
&\in \Omega(cz^2).
\end{aligned}
$$

If $h < z-2$, then $\sum_{i=1}^{z-1} a_i(h-2z+2+2i)$ can be negative. This sum is minimized if the messages with $a_i = 1$ occur first. Let the number of these messages be y. In this case, we have to consider $\sum_{i=1}^{y} h-2z+2+2i = y(h-2z+3+y)$, which reaches its minimum of $-(1/4)(2z-h-3)^2$ for $y = z - h/2 - 3/2$. Consequently,

$$
\begin{aligned}
dc^{\mathcal{OPT}} &> \frac{c}{4} \sum_{i=1}^{z-1} \left(a_i(h-2z+2+2i) + 2z-2-2i \right) \\
&> -\frac{c}{16}(2z-h-3)^2 + \frac{c}{4}(z-2)^2 \\
&\in \Omega(cz^2).
\end{aligned}
$$

Hence $dc^{\mathcal{OPT}}$ is $\Omega(cz^2)$ in both cases a) and b); finally:

$$
\rho = \frac{cost^{\mathcal{KNR}}}{cost^{\mathcal{OPT}}} \in O\left(\frac{zhc}{cz^2 + ch} \right) = O\left(\min\left(\frac{h}{z}, z \right) \right) = O\left(\sqrt{h} \right).
$$

□

We now investigate what happens if there are fewer time slots between packet arrivals, i.e., situations where merge operations can occur. To this end, we consider a sequence of packet arrivals without merge operations and compare it to the same sequence where the number of time slots between consecutive arrivals is reduced. More precisely, let $S = \{(n_1, t_1), (n_2, t_2), \ldots\}$ denote a sequence of packet arrivals, where (n_i, t_i) describes the arrival of n_i packets in time slot t_i. We compute the difference between the cost accumulated for sequences $S = \{(n_1, t_1), (n_2, t_2), \ldots\}$ without merged messages and $S' = \{(n'_1, t'_1), (n'_2, t'_2), \ldots\}$, where for all $i > 0$ it holds that $n_i = n'_i$ and $t_{i+1} - t_i \geq t'_{i+1} - t'_i$.

Lemma 4.6 *Consider a transmission of $\mathcal{OPT}$ in the sequence S' and assume that $\mathcal{KNR}$ merges μ_i messages into one message at hop distance i from the leaf. Compared to a sequence S where no messages are merged, $\mathcal{KNR}$ 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 difference is in $\Omega(\mu_i \cdot c \cdot (h-i))$. □

Lemma 4.7 *Consider a transmission of $\mathcal{OPT}$ in the sequence S' and assume that $\mathcal{KNR}$ merges μ_i messages into one message at hop distance i from the leaf. Compared to a sequence S where no messages are merged, $\mathcal{OPT}$ can reduce its cost by at most $O(\mu_i \cdot c \cdot (h-i))$.*

Proof: Consider the μ_i messages that $\mathcal{KNR}$ merges at node i. Note that message j of size n_j cannot catch up with a message of greater size $n_j > n_i$ unless it merges with another message first. This implies, that in order to have μ_i messages merging at distance i from the leaf, the message size of $\mathcal{KNR}$'s messages has to be strictly monotonically increasing.

In the following, let the superscript s identify variables considered in the scenario where $\mathcal{KNR}$ does not merge any messages, and the superscript m identify variables in the other scenario. Let $\lambda_l + \delta_l$ denote the time period between two consecutive arrival time slots of the set of n_l and the set of n_{l+1} packets, where $\lambda_l = \lceil c/n_l \rceil$ is the number of time slots the l^{th} message spends at the leaf δ_l is the number of time slots between m_l^A's departure and m_{l+1}^A's arrival that ensure that m_{l+1}^A does not catch up (too early) with m_l^A. For sequences where $\mathcal{KNR}$ does not merge any packets, $\mathcal{OPT}$'s delay cost is given by

$$dc^s = \sum_{l=1}^{\mu_i - 1} \sum_{j=l}^{\mu_i - 1} n_l(\delta_j^s + \lambda_j).$$

In order to guarantee that consecutive messages are not merged by $\mathcal{KNR}$ it holds that $\delta_l^s = h \lceil c/n_l \rceil - (h-1) \lceil c/n_{l+1} \rceil - \lceil c/n_l \rceil$ due to Lemma 4.3 and Assumptions 1 and 2. Since $n_l < n_{l+1}$, we know that $\delta_l^s < h(\lceil c/n_l \rceil - \lceil c/n_{l+1} \rceil)$. Note that this entails that

$$\sum_{j=l}^{\mu_i - 1} \delta_j^s \leq h \left(\lceil c/n_l \rceil - \lceil c/n_{\mu_i} \rceil \right).$$

If we assume the merge operation of μ_i messages to happen at depth $h - i$ we can compute a lower bound for the shortened time period between two messages. Observe that in order to ensure that messages do not merge too early, it must hold that $\delta_l^m \geq i \lceil c/n_l \rceil - (i-1) \lceil c/n_{l+1} \rceil - \lceil c/n_l \rceil$. Due to $n_l < n_{l+1}$, $\delta_l^m \geq (i-1)(\lceil c/n_l \rceil - \lceil c/n_{l+1} \rceil)$ and thus

$$\sum_{j=l}^{\mu_i - 1} \delta_j^m \geq (i-1) \left(\lceil c/n_{\mu_i} \rceil - \lceil c/n_l \rceil \right).$$

Combining the above, $\mathcal{OPT}$ can reduce its delay cost by at most

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

□

Lemma 4.8 *Consider a sequence of packet arrivals on a chain graph satisfying the following conditions: $\mathcal{KNR}$ sends some messages individually to the root and performs one merge operation on the other messages. If $\mathcal{OPT}$ merges all these messages into one at the leaf then the competitive ratio is $O(\sqrt{h})$.*

Proof: Let the number of messages $\mathcal{KNR}$ sends individually to the root be denoted by ν and the number of messages $\mathcal{KNR}$ merges at distance i from the leaf be denoted by μ. If the packets that form the μ messages $\mathcal{KNR}$ merges arrive before the ν messages sent to the root separately the claim follows directly from Lemmas 4.6 and 4.7. 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 Lemmas 4.6 and 4.7 leads to a total cost for $\mathcal{KNR}$ of less than $3(\nu hc + hc + \mu ic)$ 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 $O(1)$, 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 $O(hn\mu/(h + n\mu^2))$, which is $O(\sqrt{h})$. □

Hence, we have shown that the competitive ratio does not deteriorate for one occurrence of a merge operation. We can apply the above lemma and induction on the number of merge operations one message takes part in: Treat the node where a merge occurs as a new leaf node and consider the merge operations on the remaining chain until the root independently. For this network, the same arguments apply, and we can conclude that the cost the online and the offline algorithm save by merging does not inflict a change in the competitive ratio. Hence it follows that the competitive ratio is at most $O(\min(\sqrt{h}, c))$ on chain graphs. □

Lower Bound

We now show that $\mathcal{KNR}$ is asymptotically optimal for all oblivious online algorithms, i.e., we derive a lower bound for chain networks of $\Omega(\min(\sqrt{h}, c))$.

Theorem 4.9 *The competitive ratio of any oblivious algorithm ALG is at least*

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

Proof: 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 due to the oblivious nature of ALG. 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, it asymptotically holds for this sequence 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, this yields $\Omega(\sqrt{h})$. If h is asymptotically larger than $c + w$, the best oblivious algorithm choosing w as small as possible gives a lower bound of $\Omega(c)$. Finally, if $h < c + w$, we have $\Omega(\sqrt{h})$. □

Discussion

Compared to the trees studied in the previous section, the analysis of chain networks is more involved. This can be explained with the fact that the tree bound is in the order of the tree height and hence merely investigating the cost occurring at any given depth rather than on the whole path is sufficient to prove the claim; there is no need to perform a detailed comparison of merge sequences of the online algorithm and the offline algorithm.

Our findings can be compared to the analysis presented in [KNR02]. Their $\Omega(\sqrt{h})$ lower bound holds for arbitrary oblivious algorithms on *trees*. 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 $\mathcal{KNR}$.

5 Value-Sensitive Aggregation

There are many scenarios where, in contrast to the aggregation model considered above, the information to be delivered is not binary (e.g., event messages) but where arbitrary *value* aggregations need to be performed at the root. E.g., consider a set of sensor nodes measuring the temperature or oxygen levels at certain outdoor locations, and the root is interested to have up-to-date information on these measurements. In this case, 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. We refer to the sum $\sum_t |l_t - r_t|$ as the error 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 time slot. Moreover, we assume that the sensor nodes can only measure discrete quantities, and that the difference between two measured values is at least ϵ.

Optimal Offline Algorithm for Link

There exists a simple optimal (offline) algorithm $\mathcal{OPT}$ which applies dynamic programming. $\mathcal{OPT}$ exploits the following optimal substructure: If we know the best sending times for all slots $t' < t$ given that a message is sent at time t', we can compute the optimal sending times assuming that $\mathcal{OPT}$ sends at time t. Let the number of time slots under scrutiny be T. Note that, we only have to consider the time slots with value changes, because an optimal algorithm either sends a value immediately after it has changed or not until the value has changed again. Let time slot t_i denote the time slot when the i^{th} value change occurred and we set $t_0 = 0$. Determining all time slots with value changes requires iterating over all T time slots. To compute the minimum cost accumulated until time slot t_i, we consider each possible last transmission $j < i$ and add the inaccuracy cost which accrued at the root node between the two transmissions j and i. Hence we can construct an array $OPT[\cdot]$ of size $\lambda + 1$, 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:

$$OPT[i] = \min_{j<i} \left(OPT[j] + c + |l_{t_i} - l_{t_j}| + \sum_{k=j+1}^{i-1} |l_{t_k} - l_{t_j}|(t_{k+1} - t_k) \right).$$

In time $O(\lambda^2)$ we construct a matrix $A[\cdot][\cdot]$ of size $(\lambda + 1) \times (\lambda + 1)$, where for all $1 \leq j, i \leq \lambda$ the entry $A_{i,j} = |l_{t_i} - l_{t_j}|(t_{i+1} - t_i)$. This allows us to compute any sum $\sum_{k=j+1}^{i-1} |l_{t_k} - l_{t_j}|(t_{k+1} - t_k)$ for $1 \leq i \leq j \leq \lambda$ in constant time. For a given i, $1 \leq i \leq \lambda$, computing $OPT[i]$ takes time $O(\lambda)$ using these pre-computed values. Thus, we have the following theorem.

Theorem 5.1 *In a link network, the optimal aggregation strategy for T time slots can be computed in time $O(T + \lambda^2)$, where λ is the number of value changes at the leaf.*

If the input for the offline algorithm consists of a sequence of value changes and the time slots when they happened, the time complexity is $O(\lambda^2)$.

Online Algorithm for Link

We propose the following online algorithm $\mathcal{AGG}$, which can be seen as a generalization of the algorithm $\mathcal{KNR}$ discussed in the previous sections: 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 periods between two transmissions of $\mathcal{AGG}$. For each such period, 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 error cost is c in each time period I, then $3\rho/2$ is an upper bound on the total competitive ratio.*

Proof: First observe that $\mathcal{AGG}$ can have a larger error cost than c in a period, e.g., if in a time slot where the accumulated error cost is $c - \varepsilon$ for an arbitrarily small $\varepsilon > 0$ there is a large value change of size Δ at the leaf. Hence, the online algorithm's error in any period is at most $2(c - \epsilon) + \Delta$. Consider a period I where the online algorithm's error cost is $2(c - \epsilon) + k$ for some $k \leq \Delta$. Compared to the case studied so far, $\mathcal{AGG}$'s error cost will increase by at most $k + c - 2\epsilon$. However, due to the large value change, we know that the optimal algorithm's error 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: *Upper bound.* First, the ratio is computed under the assumption that the error cost of $\mathcal{AGG}$ is exactly c; we then apply Lemma 5.2. We classify the possible types of periods I between two sending events of the online algorithm and consider them separately. Observe that for any period 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 error cost c. It remains to examine the situations where $\mathcal{OPT}$ does not send.

Assume that at the beginning of this period, $\mathcal{AGG}$ 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 period 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 error cost of $\mathcal{AGG}$ is c for every period.

Case $\delta_0 = \delta_1 = 0$: If $\mathcal{OPT}$ has no transmission, it must have the same error cost in this period 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 error 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 error cost δ_0 in the first time slot, as $\mathcal{AGG}$ sends the correct value at the beginning of the period. Hence, $\rho_I \leq 2c/\delta_0$.

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

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

Lower bound. 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 time slots. 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. Thus, neglecting the cost of the first time slot, we have $\rho \geq 4c/2\epsilon = 2c/\epsilon$. □

Please note that the lower bound in Theorem 5.3 only holds for our algorithm $\mathcal{AGG}$. The analysis of the claim for arbitrary algorithms is left for future research.

6 Conclusion

This article attended to the subject of online information aggregation, which can be regarded as a generalization of the classic *ski-rental problem* to trees. The studied optimization problem captures the trade-off between speed and energy prevalent in many wireless networks. We were able to show that a most simple algorithm achieves a best possible asymptotic competitive ratio in the class of oblivious, deterministic and fully distributed algorithms. In particular, the algorithm we consider fulfills its task without knowledge of the presence and the state of devices in its vicinity, and it does not base its decisions on previous events. Thus, apart from the efficiency criterion, the new analysis we have presented in this work suggests that the algorithm introduced in [KNR02] is attractive for practical applications as it poses minimal hardware requirements on sensor nodes in terms of computational requirements. Observe however that we assume that bursty arrivals of events and packets can be handled by the nodes. An interesting future research direction would be to allow nodes to inform their children that a certain packet could not be accepted due to memory constraints, and should be re-sent later. This makes the protocol more complex and requires nodes to know more about each other's state. Moreover, while such a solution can mitigate a problem in the short run, there is a natural bound on what can be stored in the total network with backlogging, namely the overall storage capacity of the nodes.

Acknowledgments

We would like to thank Thibaut Britz and Marcin Bienkowski for interesting discussions on this work.

References

[AB03] Susanne Albers and Helge Bals. Dynamic TCP Acknowledgement: Penalizing Long Delays. In *Proc. 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 47–55, 2003.

[BEY98] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[BKMS+06] Luca Becchetti, Peter Korteweg, Alberto Marchetti-Spaccamela, Martin Skutella, Leen Stougie, and Andrea Vitaletti. Latency Constrained Aggregation in Sensor Networks. In *Proc. Annual European Symposium on Algorithms (ESA)*, pages 88–99, 2006.

[BKV04] Carlos Brito, Elias Koutsoupias, and Shailesh Vaya. Competitive Analysis of Organization Networks or Multicast Acknowledgement: How Much to Wait? In *Proc. 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 627–635, 2004.

[DGS98] Daniel R. Dooly, Sally A. Goldman, and Stephen D. Scott. TCP Dynamic Acknowledgment Delay: Theory and Practice. In *Proc. 30th Annual ACM Symposium on Theory of Computing (STOC)*, pages 389–398, 1998.

[FL02] Jens S. Frederiksen and Kim S. Larsen. Packet Bundling. In *Proc. 8th Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 328–337, 2002.

[KEW02] B. Krishnamachari, D. Estrin, and S. Wicker. The Impact of Data Aggregation in Wireless Sensor Networks. In *Proc. 22nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 575–578, 2002.

[KKR01] Anna R. Karlin, Claire Kenyon, and Dana Randall. Dynamic TCP Acknowledgement and Other Stories About $e/(e-1)$. In *Proc. 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 502–509, 2001.

[KLS08] Fabian Kuhn, Thomas Locher, and Stefan Schmid. Distributed Computation of the Mode. In *Proc. 27th ACM Symposium on Principles of Distributed Computing (PODC)*, 2008.

[KMSSV09] P. Korteweg, A. Marchetti-Spaccamela, L. Stougie, and A. Vitaletti. Data Aggregation in Sensor Networks: Balancing Communication and Delay Costs. *Theoretical Computer Science*, 410(14):1346–1354, 2009.

[KNR02] S. Khanna, S. Naor, and D. Raz. Control Message Aggregation in Group Communication Protocols. In *Proc. 29th International Colloquium of Automata, Languagues and Programming (ICALP)*, pages 135–146. Springer, 2002.

[NS09] T. Nonner and A. Souza. Latency Constrained Aggregation in Chain Networks Admits a PTAS. In *Proc. 5th International Conference on Algorithmic Aspects in Information and Management (AAIM)*. Springer, 2009.

[OSW08] Yvonne Anne Oswald, Stefan Schmid, and Roger Wattenhofer. Tight Bounds for Delay-Sensitive Aggregation. In *Proc. 27th ACM Symposium on Principles of Distributed Computing (PODC)*, 2008.

[SBAS04] Nisheeth Shrivastava, Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. Medians and Beyond: New Aggregation Techniques for Sensor Networks. In *Proc. 2nd international conference on Embedded networked sensor systems (SENSYS)*, pages 239–249. ACM, 2004.

[SO04] I. Solis and K. Obraczka. The Impact of Timing in Data Aggregation for Sensor Networks. In *Proc. IEEE International Conference on Communications (ICC)*, volume 6, 2004.

[ST85] Daniel D. Sleator and Robert E. Tarjan. Amortized Efficiency of List Update and Paging Rules. *Commun. ACM*, 28(2):202–208, 1985.

[Ste94] W. R. Stevens. *TCP/IP Illustrated, Vol.1: The Protocols*. Addison-Wesley, 1994.

[vRW04] P. von Rickenbach and R. Wattenhofer. Gathering Correlated Data in Sensor Networks. In *Proc. ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 60–66. ACM New York, NY, USA, 2004.

[YF05] O. Younis and S. Fahmy. An Experimental Study of Routing and Data Aggregation in Sensor Networks. In *Proc. IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 50–57, 2005.

[YKP04] Y. Yu, B. Krishnamachari, and V. K. Prasanna. Energy-Latency Tradeoffs for Data Gathering in Wireless Sensor Networks. In *Proc. 23rd Annual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM)*, 2004.

[YZ09] Ke Yi and Qin Zhang. Multi-dimensional Online Tracking. In *Proc. 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1098–1107, 2009.