Control Message Aggregation in Group Communication Protocols

Sanjeev Khanna[1], Joseph (Seffi) Naor[2], and Dan Raz[2]

[1] Dept. of Computer & Information Science, University of Pennsylvania, Philadelphia, PA 19104
sanjeev@cis.upenn.edu,
[2] Computer Science Dept., Technion, Haifa 32000, Israel
{naor,danny}@cs.technion.ac.il

Abstract. Reliable data transmission protocols between a sender and a receiver often use feedback from receiver to sender to acknowledge correct data delivery. Such feedback is typically sent as control messages by receiver nodes. Since sending of control messages involves communication overhead, many protocols rely on aggregating a number of control messages and sending them together as a single packet over the network. On the other hand, the delays in the transmission of control messages may reduce the rate of data transmission from the sender. Thus, there is a basic tradeoff between the communication cost of control messages and the effect of delaying them.
We develop a rigorous framework to study the aggregation of control packets for multicast and other hierarchical network protocols. We define the multicast aggregation problem and design efficient online algorithms for it, both centralized and distributed.

1 Introduction

Reliable transmission of data across an unreliable packet network (such as IP) requires end to end protocols (such as TCP) that use feedback. Such protocols use control messages to set up connections, to acknowledge the correct reception of (part of) the data, and possibly to control the transmission rate. This introduces a communication overhead, that may result in decreasing the overall effective throughput of the network. Since control messages are typically small, one can aggregate several control messages into a single packet in order to reduce the communication overhead. However, such an approach introduces extra delay since a control message may need to wait at the aggregation points for additional control messages. In many cases (e.g. TCP) delayed control messages may result in a reduction of the overall transmission protocol throughput. Thus, there is a tradeoff between the amount of reduction in the communication cost one can get from aggregating control messages, and the affect of the possible delay caused by it. This paper studies this tradeoff in the context of multicast and hierarchical protocols.

We use the term *multicast protocols* to describe a transmission protocol that delivers data from a single sender to a (possibly large) set of receivers. Reliable

P. Widmayer et al. (Eds.): ICALP 2002, LNCS 2380, pp. 135–146, 2002.

multicast protocols provide a mechanism to ensure that every receiver receives all the data. Since the underlying network (IP) is unreliable (i.e. packets may be lost), such protocols must detect errors (i.e. lost packets) in order to request their retransmission. One approach is to have the sender detect that an error has occurred, this is done by making each receiver acknowledge receiving each packet. A common practice is to send a special control packet called ACK (positive acknowledgment) for each received packet. However, this may result in an implosion of control packets being sent to the sender. An alternative is to make each receiver independently responsible for detecting its errors, e.g., a gap in the sequence numbers may signal a lost packet. In such a case the common practice is to send a NAK (negative acknowledgment). In this case, an implosion of control packets may occur as well when the same packet is lost for many receivers.

Several reliable multicast protocols were developed (see for example [7]) which use different techniques to overcome the ACK/NAK implosion problem. One commonly used technique is to use control message aggregation where several control messages to the root are aggregated into a single control message, thus saving on the communication cost at the expense of delaying individual messages to achieve aggregation. Most protocols use ad hoc methods based on user-defined timers to perform such an aggregation. Consider, for example, the Local Group based Multicast Protocol (LGMP) [4]. This protocol uses the local group concept (LGC) in order to overcome the ACK-implosion problem. Local groups are organized in an hierarchical structure. Each receiver sends control message to its local group controller, which aggregates the data, tries to recover locally, and when needed reports to its local group controller (which belongs to a higher level in the hierarchy tree). LGMP uses timer timeout in order to decide when to send a control message up the tree (see Section 2.4 of [4]).

Using hierarchical structures is not restricted to multicast algorithms. Many network protocols use hierarchical structures to address the scalability problem. Examples include the PNNI [8] routing algorithm in ATM and the RSVP [9] protocols. In many of these protocols there is a need to report status to the higher level in the hierarchy. This leads to exactly the same tradeoff between the amount of control information and the delay in the reported information.

A very recent example is the Gathercast mechanism proposed by Badrinath and Sudame [1]. Gathercast proposes to aggregate small control packets, such as TCP acknowledge messages, sent to the same host (or to hosts in the same (sub)-network). The authors show that such an aggregation significantly improves performances under certain conditions. In the proposed scheme there is a time bound on the amount of delay a packet can suffer at the gathering point, and once this timer has expired the packet is sent.

1.1 Our Model and Results

This paper offers a theoretical framework to study the aggregation of control messages in such situations. Specifically, we investigate the global optimization problem of data aggregation in multicast trees. We are given a multicast tree with a *communication cost* associated with each link. When a packet arrives at

a receiver node (leaf or internal node), a control message (typically an ACK control message) has to be sent to the root. Nodes can delay control messages in order to aggregate them, and save on the communication cost. The cost paid by a control message to traverse a tree link is independent of the number of aggregated control messages that it represents. Also, note that control messages that originate at the same receiver node as well as the ones originating at different nodes are allowed to be aggregated. However, the delay of each original control message contributes to a *delay cost*. The total delay cost is defined to be the *sum* of the delays of all the messages. Thus, our optimization problem can be stated as follows: Given a multicast tree and a sequence of packet arrivals at the receiver nodes, determine a schedule of minimum cost (communication cost plus the delay cost) to send back the control messages to the root. We refer to this problem as the *multicast aggregation* problem. This is an online optimization problem in which decisions must be made based upon current state without knowing future events.

We present both centralized and distributed online algorithms for the multicast aggregation problem. Our centralized online algorithm assumes a global information model, where a central entity determines how control messages are sent. However, future arrivals of messages are not known to the algorithm. Our distributed online algorithm is a "local" algorithm that makes decisions in the nodes of the tree based on the packets waiting there. Clearly, in practice, multicast aggregation algorithms are distributed, thus making the centralized model to be largely of theoretical interest. However, we believe that studying the centralized online model provides important insight into the combinatorial structure of the problem. Indeed, it turns out that the multicast aggregation problem is highly non-trivial even in the centralized model. Both of our algorithms are based on a natural strategy that tries to balance the communication costs and the delay costs incurred by the control messages.

For the centralized online case, we give an $O(\log \alpha)$-competitive algorithm, where α is the total communication cost associated with the multicast tree. We also show that our analysis of the competitive factor of the algorithm is tight. For the distributed online case, we give an $O(h \cdot \log \alpha)$-competitive algorithm, where h is the height of the multicast tree. We show a lower bound of $\Omega(\sqrt{h})$ on the competitive ratio of *any* distributed online algorithm which is *oblivious*, i.e., uses only local information. This notion will be defined more precisely later.

In order to study the performance of our distributed algorithms in practice, we conducted a performance study using simulations. We compared our algorithm to two commonly used methods: one that reports all events immediately, and another that works with timers. It turns out that in most scenarios our distributed online algorithm outperforms the other heuristics, and in some relevant cases it performs significantly (up to 40%) better. One of the most notable characteristics of our online algorithm is its robustness, i.e., it performs well across a broad spectrum of scenarios. It follows from our simulations that in a sense, our online algorithm works like a *self adjusting timer*, since on one hand it aggregates messages, but on the other hand it does not wait too long before

sending them. Due to lack of space the detailed description of the simulation results are omitted from this version of the paper.

1.2 Related Work

Dooly, Goldman and Scott [3] study the problem of aggregating TCP ACK packets over a single link. They observed that the off-line case of the single link version can be optimally solved by using dynamic programming. For the online case of this problem they designed a 2-competitive algorithm in the spirit of rent-to-buy algorithms. Bortnikov and Cohen [2] give several online heuristics for a local scheduling problem for the more general hierarchical case.

A model similar to ours was introduced by Papadimitriou and Servan-Schreiber [6] in the context of organization theory (see also [5]). They model an organization as a tree where the messages arrive at the leaves and must be sent to the root as soon as possible. Messages model pieces of information about the "world" outside the organization, and the "boss" needs to have an up-to-date view of the world as soon as possible. These messages are allowed to be aggregated and the objective function is the sum of the communication cost and the delay cost. However, their work is primarily focused on the case where message arrivals are modeled by a Poisson process.

2 The Model

In this section we formally define our model and assumptions. We are given a rooted tree T that we refer to as the multicast tree. This tree may be a real multicast tree, or a tree describing a hierarchical structure of a protocol, where each link actually represents a path in the real network. We view the tree T as being directed from the leaves towards the root r so that each arc is uniquely defined by its tail. Thus, for each node $v \in T$ (except for the root) we denote by $e(v)$ the arc leaving it. Each arc (a tree edge) has a communication cost (or just cost) denoted by $c(\cdot)$. We assume that the communication cost of each arc is at least 1.

Packets arrive at the tree nodes over time which is slotted. An arrival of a packet at a tree node generates a control message (ACK) that needs to be delivered to the root. Our goal is to minimize communication cost of control messages by aggregating them as they make their way up the tree. We denote by τ the time at which the last packet arrives. In general, packets can arrive at any node, internal node or leaf. However, we can assume without loss of generality that packets arrive only at leaves.

For a given packet p, let $t_a(p)$ denote the time at which it arrives at a leaf node v. As mentioned before, for each packet we need to send a control message to the root. Control messages waiting at a node accumulate delay and the delay accumulated until time t by the control message for a packet p is denoted by $d_t(p)$. We assume that each control message must wait at least one unit of time at a leaf before being sent out. Thus, the delay accumulated by a control message

is at least 1. From here on, we will identify each packet with its control message, and simply use the word packet and control messages interchangeably. Our delay metric is the sum of the delays of the packets, where the delay of each packet is linear in the time it waits at a node, i.e. $d_t(p) = \beta(t - t_a(p))$ for some constant β.

In general, nodes can aggregate as many packets as needed and send them up the tree to their parent. We make the simplifying assumption of no propagation delay along the links (this avoids having to deal with synchronization issues). In the distributed model, we assume that at each time step t, each node v may aggregate awaiting packets and send the aggregated packet to its parent. The cost of sending the message is $c(e(v))$, which is independent of the number of aggregated packets. In the centralized model we assume that at each time step t, an online algorithm broadcasts a subtree T_t (possibly empty) that collects all packets present at the leaves of T_t and sends them to the root. These packets are aggregated together in a bottom-up manner so that each link of the tree T_t is traversed exactly once during this process. We refer to such an action as broadcasting the subtree T_t, and the cost of this broadcast is given by $\mathsf{cost}(T_t) = \sum_{v \in T_t} c(e(v))$.

To summarize, our total cost is the sum of the delay costs of all the packets (*total delay cost*) together with the total communication cost. In the centralized model, the total communication cost is the sum of the costs of the subtrees that are broadcast. In the distributed model, the communication cost is the sum of the costs of the tree edges that are used by the (aggregated) messages.

3 The Centralized Online Algorithm

In this section we present log-competitive centralized online algorithms for our problem. We assume a global information model, where a central entity determines how control messages are sent. However, future arrivals of messages are not known to the algorithm. Our online algorithm is based on the following natural strategy: at any time t, we broadcast a maximal subtree (wrt containment) such that the cost of the subtree is roughly equal to the accumulated delay of the packets at its leaves.

Let $d_t(p)$ denote the delay accumulated by a packet p until time t. Our algorithm broadcasts at each time t, a maximal subtree $T' \subseteq T$ that satisfies

$$\sum_{p \in T'} d_t(p) \geq \mathsf{cost}(T'),$$

where $p \in T'$ ranges over all packets waiting at any leaf node in T'. It is easily seen that at each broadcast of a tree T' by the online algorithm, we must also have $\sum_{p \in T'} d_t(p) \leq 2 \cdot \mathsf{cost}(T')$.

Fix an optimal solution OPT and let $\mathcal{T}^* = \{T_1, ..., T_\tau\}$ denote the trees (possibly empty) broadcast by OPT. Let P denote the set of all packets received during the algorithm's execution. For any packet $p \in \mathsf{P}$, let $\mathsf{delay}(p)$ and $\mathsf{delay}^*(p)$ denote the delay incurred by a packet p in the online solution and the optimal solution, respectively. Clearly, the cost C^* incurred by OPT is given by

$$C^* \;=\; \sum_{T_i \in \mathcal{T}^*} \mathsf{cost}(T_i) + \sum_{p \in P} \mathsf{delay}^*(p).$$

Define $L = \{p \mid \mathsf{delay}(p) > \mathsf{delay}^*(p)\}$ to be the set of packets that the online algorithm broadcasts later than OPT (late packets), and let $E = \{p \mid \mathsf{delay}(p) \leq \mathsf{delay}^*(p)\}$ be the set of packets that are broadcast no later than OPT (early packets). The key fact that we will use in our analysis is the following lemma that relates the delay incurred by the late packets in the online algorithm to one in the optimal solution.

Lemma 1.

$$\sum_{p \in L} \mathsf{delay}(p) \;\leq\; \sum_{p \in L} \mathsf{delay}^*(p) \;+\; 4\left(\sum_{T_t \in \mathcal{T}^*} \mathsf{cost}(T_t)\right) \cdot \log \alpha$$

Proof. Consider the set $L_t \subseteq L$ of packets that OPT sends at time t in a tree $T_t \in \mathcal{T}^*$. Let L_t denote this subset of packets and let $\ell = |L_t|$. Define $t_i = t + (1/\beta) \cdot 2^i(\mathsf{cost}(T_t)/|L_t|)$. We claim that the number of packets in L_t that are still alive at time t_i is at most $|L_t|/2^i$. Suppose not, then the total accumulated delay of these packets exceeds $\mathsf{cost}(T_t)$. Since they have not yet been broadcast, this contradicts the online broadcasting rule. Thus at time $t_{1+\lceil \log(\mathsf{cost}(T_t)) \rceil}$, no packets from the set L_t remain. The total delay incurred by packets in L_t in the online algorithm is thus bounded as below:

$$\begin{aligned}
\sum_{p \in L_t} \mathsf{delay}(p) &\leq \sum_{p \in L_t} \mathsf{delay}^*(p) \;+\; \sum_{i=1}^{1+\lceil \log(\mathsf{cost}(T_t)) \rceil} \beta \cdot \left(\frac{|L_t|}{2^{i-1}}\right)\left(\frac{1}{\beta}\right)\left(\frac{2^i \mathsf{cost}(T_t)}{|L_t|}\right) \\
&= \sum_{p \in L_t} \mathsf{delay}^*(p) \;+\; 2\log\left(\mathsf{cost}(T_t)\right) \cdot \mathsf{cost}(T_t) \;+\; 4\mathsf{cost}(T_t) \\
&\leq \sum_{p \in L_t} \mathsf{delay}^*(p) \;+\; 4\log\left(\mathsf{cost}(T_t)\right) \cdot \mathsf{cost}(T_t)
\end{aligned}$$

Since $\mathsf{cost}(T_t) \leq \alpha$, the lemma now follows by summing over all sets $L_1, ..., L_\tau$.

Recall that in the centralized algorithm, a subtree $T' \subseteq T$ is broadcast if it satisfies $\sum_{p \in T'} d_t(p) \geq \mathsf{cost}(T')$. Therefore, the communication cost of the algorithm is no more than the delay cost, and the cost C incurred by the online centralized algorithm is given by:

$$\begin{aligned}
C \leq 2\sum_{p \in P} \mathsf{delay}(p) \;&\leq\; 2\left(\sum_{p \in E} \mathsf{delay}^*(p) + \sum_{p \in L} \mathsf{delay}(p)\right) \\
&\leq 2\left(\sum_{p \in E} \mathsf{delay}^*(p) + \sum_{p \in L} \mathsf{delay}^*(p)\right) \;+\; 2\left(4\left(\sum_{T_i \in \mathcal{T}^*} \mathsf{cost}(T_i)\right) \cdot \log \alpha\right)
\end{aligned}$$

$$= 2\left(\sum_{p\in P} \mathsf{delay}^*(p) + 4\left(\sum_{T_i\in\mathcal{T}^*} \mathsf{cost}(T_i)\right)\log\alpha\right) \;\leq\; 8C^*\cdot\log\alpha$$

We note that we are not trying to optimize constants. Thus we have the following result.

Theorem 1. *There is an $O(\log\alpha)$-competitive centralized algorithm for the multicast aggregation problem.*

3.1 A Lower Bound

We now show that our analysis above is essentially tight. Consider a two-level tree T as shown in Figure 1. Assume $\beta = 1$ for clarity of exposition. The tree T has a single edge coming into root r from u with a cost of k^{2k+1} for some integer k. The node u has k children $v_0, ..., v_k$ where the cost of each edge (v_i, u) is k^{2k}. The total cost α of the tree T is thus $2k^{k+1}$. We will now describe a packet arrival sequence such that our online algorithm pays $O(\log\alpha/\log\log\alpha)$ times the optimal cost on this sequence. The arrival sequence consists of a sequence of blocks. The jth block comprises of arrival of packets at each leaf at time $t_j = (2j)k^{2k+1}$ where $j \geq 0$. The leaf node v_i receives $k^{2(k-i)}$ packets in each block. It is easily seen that the optimal solution is to immediately broadcast the entire tree at time $t_j + 1$.

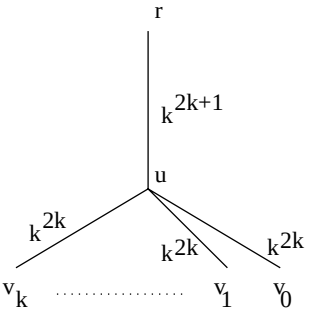

Fig. 1. A lower bound for our online algorithm

We now analyze the behavior of our online algorithm on this sequence. Let T_i denote the subtree of T that only contains the nodes r, u and v_i. First observe that all requests that arrive in any block j are broadcast before the arrival of the requests in the next block; this follows from the fact that the starting time of two consecutive blocks are at least α time units apart. So it suffices to analyze the cost paid by the online algorithm in each block. In each block j, the first

subtree is broadcast at time $t_j + (k+1)$; node v_0 accumulates a delay that is equal to $\mathsf{cost}(T_0)$ and we broadcast tree T_0. Observe that by our choice of edge weights, no other leaf v_i is able to connect to this subtree at this time. The next subtree is now broadcast at time $t_j + (k^3 + k^2)$ and is simply the subtree T_1. In general, at time $t_j + (k^{2i+1} + k^{2i})$ we broadcast subtree T_i. Thus the total cost paid by our algorithm in each block is $O(k \cdot k^{2k+1})$ which is $O(\log \alpha / \log\log \alpha)$ times the cost paid by the optimal solution.

4 The Distributed Online Model

We now present a distributed version of the centralized online algorithm. Consider node v at time t, and denote by $P(v,t)$ the set of packets that are waiting at v at time t. The set $P(v,t)$ is aggregated into a single message, $m(v,t)$, waiting to be sent from v. We denote by $\mathsf{delay}(m(v,t))$ the delay associated with $m(v,t)$. At each unit of time, $\mathsf{delay}(m(v,t))$ is increased by $\beta|P(v,t)|$, i.e., each message in the set $P(v,t)$ contributes towards the delay accumulated by $m(v,t)$. We note that if node v receives a message m' at time t', then it aggregates m' with $m(v,t')$, and the delay associated with the aggregated message is $\mathsf{delay}(m') + \mathsf{delay}(m(v,t'))$. We assume here that each message must wait at least one unit of time at a node before being sent out. Thus, the delay accumulated by a message is at least β times the length of the path from the leaf (where it originated) to the root.

The algorithm sends message $m(v,t)$ at time t if $\mathsf{delay}(m(v,t)) \geq c(e(v))$. Suppose node u is the head of arc $e(v)$. Then, the delay associated with message $m(v,t)$ upon arrival at u is $\mathsf{delay}(m(v,t)) - c(e(v))$, i.e., message $m(v,t)$ has "paid" for crossing $e(v)$. Note that if message $m(v,t)$ is sent at time t, then $\mathsf{delay}(m(v,t)) \leq 2c(e(v))$.

We now analyze the competitive factor of the distributed online algorithm. Fix an optimal solution OPT and let $\mathcal{T}^* = \{T_1, ..., T_\tau\}$ denote the trees (possibly empty) broadcast by OPT. Let P denote the set of all packets received during the algorithm's execution. For any packet $p \in \mathsf{P}$, let $\mathsf{delay}(p)$ and $\mathsf{delay}^*(p)$ denote the delay incurred by a packet p in the online solution and the optimal solution, respectively. Clearly, the cost C^* incurred by OPT is given by

$$C^* = \sum_{T_i \in \mathcal{T}^*} \mathsf{cost}(T_i) + \sum_{p \in P} \mathsf{delay}^*(p).$$

As in the previous section, define $L = \{p \mid \mathsf{delay}(p) > \mathsf{delay}^*(p)\}$ to be the set of packets that reached the root in the online algorithm later than the time they reached the root in OPT, and let $E = \{p \mid \mathsf{delay}(p) \leq \mathsf{delay}^*(p)\}$ be the set of packets that reached the root in the online algorithm no later than the time they reached the root in OPT. The key fact that we will use in our analysis is the following lemma that relates the delay incurred by the late packets in the online algorithm to one in the optimal solution.

Lemma 2.

$$\sum_{p\in L} \mathsf{delay}(p) \;\le\; \sum_{p\in L} \mathsf{delay}^*(p) \;+\; \left(8\sum_{T_t\in\mathcal{T}^*} \mathsf{cost}(T_t) + 4\sum_{p\in P} \mathsf{delay}^*(p)\right)\cdot h\cdot\log\alpha$$

Proof. Consider a $T_t \in \mathcal{T}^*$. Denote by W the set of late packets that are broadcast in tree T_t. Define the following sequences, $\{t_i\}$ and $\{W_i\}$, where $W_i \subseteq W$ denotes the packets that have not reached the root by time t_i. Define $t_0 = t$ and $W_0 = W$; t_i $(i \ge 1)$ is defined to be the first time since t_{i-1} by which the packets belonging to W_i have accumulated delay of at least $2\cdot\mathsf{cost}(T_t)$. Since t_i is the earliest time at which this event occurs, the accumulated delay cannot exceed $2\cdot\mathsf{cost}(T_t) + \sum_{p\in T_t}\mathsf{delay}^*(p)$. (Since the packets in OPT also had to wait one unit of time.)

We now claim that for all $i \ge 1$, $|W_i| \le |W_{i-1}|/2$. Suppose that this is not the case. Then, the delay accumulated by the packets that are alive at time t_i is:

$$\sum_{v\in T_t}\sum_{p\in v} \mathsf{delay}(p) > \mathsf{cost}(T_t)$$

Also,

$$\mathsf{cost}(T_t) = \sum_{v\in T_t} \mathsf{cost}(e(v))$$

Hence, there exists a node v such that the delay accumulated by the packets alive at v at time t_i is strictly greater than $\mathsf{cost}(e(v))$. This is a contradiction, since according to the algorithm node v should have sent its packets before time t_i.

We now define the potential at time t_i, Φ_i, to be the sum of the distances of the packets belonging to W_i from the root of the tree T. The distance of a node v from the root is defined to be the number of links in the path from v to the root. Clearly, $\Phi_i \le |W_i|h$, and since $|W_i| \le |W_{i-1}|/2$, we get

$$\Phi_i \;\le\; \Phi_{i-1} - \frac{|W_{i-1}|}{2} \;\le\; \Phi_{i-1}\cdot\left(1-\frac{1}{2h}\right)$$

Hence, by time t_f, where $f = 4h\log(h|W|)$, we get that $\Phi_f = 0$. The total delay incurred by packets in W in the distributed online algorithm is thus bounded as below:

$$\sum_{p\in W} \mathsf{delay}(p) \;\le\; \sum_{p\in W} \mathsf{delay}^*(p) \;+\; \left(8\mathsf{cost}(T_t) + 4\sum_{p\in W} \mathsf{delay}^*(p)\right) h\log(h|W|)$$

We now claim that $|W|$ can be bounded by $|T_t|\cdot\alpha$. To see this, observe that any node that contains more than α packets will never be part of the late set. Thus, $|W| \le |T_t|\cdot\alpha$, and $\log(h|W|)$ is $O(\log\alpha)$, since we assume that the cost of each tree arc is at least 1.

The lemma now follows by summing over all sets of late packets in the trees $\{T_1, ..., T_\tau\}$.

Recall that in the distributed algorithm, if message $m(v,t)$ is sent at time t, then $\mathsf{delay}(m(v,t)) \geq c(e(v))$. Therefore, the communication cost of the algorithm is no more than the delay cost, and the cost C incurred by the online distributed algorithm is given by:

$$\begin{aligned}
C &\leq 2 \sum_{p \in P} \mathsf{delay}(p) \\
&\leq 2 \left(\sum_{p \in E} \mathsf{delay}^*(p) + \sum_{p \in L} \mathsf{delay}(p) \right) \\
&\leq 2 \left(\sum_{p \in E} \mathsf{delay}^*(p) + \sum_{p \in L} \mathsf{delay}^*(p) \right) \\
&+2 \left(\left(8 \sum_{T_i \in\ T^*} \mathsf{cost}(T_i) + 4 \sum_{p \in P} \mathsf{delay}^*(p) \right) \cdot h \cdot \log \alpha \right) \\
&= 2 \sum_{p \in P} \mathsf{delay}^*(p) \\
&+2 \left(\left(8 \sum_{T_i \in \mathcal{T}^*} \mathsf{cost}(T_i) + 4 \sum_{p \in P} \mathsf{delay}^*(p) \right) \cdot h \cdot \log \alpha \right) \\
&\leq 16 C^* \cdot h \cdot \log \alpha
\end{aligned}$$

Thus we have the following result.

Theorem 2. *There is an $O(h \cdot \log \alpha)$-competitive distributed algorithm for the multicast aggregation problem.*

4.1 A Lower Bound

We now provide evidence that it is inherently difficult to obtain significantly better bounds on the competitive ratio.

A distributed online algorithm is called *oblivious* if decisions at each node are based solely upon the static local information available at the node. In particular, in such algorithms the wait time of a packet is independent of the location of the node in the tree. We note that all the distributed algorithms considered in this paper are oblivious. In general, being oblivious is a desirable property of distributed algorithms, since non-oblivious algorithms require a dynamic updating mechanism at each node that informs it about changes in the hierarchical structure. Such a mechanism can be both expensive and complicated to implement.

Consider a path of length h, as shown in Figure 2, where the vertices on the path are $r = v_1, \ldots, v_{h+1} = z$, such that r is the root and z is the (only)

receiver. The cost of each link in the path is h. We assume that $\beta = 1$. We will now describe a packet arrival sequence such that any oblivious online algorithm pays $O(\sqrt{h})$ times the optimal cost on this sequence.

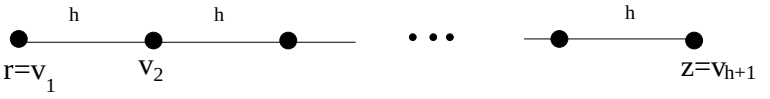

Fig. 2. A lower bound for oblivious online algorithms

Denote by ALG the online algorithm. The packets arrive one-by-one. At time 0, packet p_1 arrives at z; for $i \geq 2$, packet p_i arrives at z when packet p_{i-1} leaves node z. Denote by $wait(h)$ the time a single packet waits in z before being sent out towards r. Since the algorithm is oblivious, and since the local static information at all the nodes is the same (one out-going link with a cost of h), the same waiting time applies to all nodes on the way from z to r. Thus, the total waiting time of each packet paid by the online algorithm is $h \cdot wait(h)$ and the communication cost of each packet is h^2. Then, the cost of each packet in the online algorithm is

$$\mathsf{cost}(\text{ALG}) = h \cdot wait(h) + h^2$$

We now derive an upper bound on the cost of OPT, the optimal algorithm. We partition the packets into blocks of size $O(\sqrt{h})$. The packets in each block will be broadcast together to the root r. Clearly, the communication cost of each block in OPT is h^2, and thus the communication cost per packet is $h \cdot \sqrt{h}$. Next we bound the average delay cost of a packet in OPT. Since each block has $\sqrt{h}$ packets, a packet waits $O(\sqrt{h} \cdot wait(h))$ at node z, and then one unit at each node on its way to r. Adding the communication and the delay costs, we get that the average total cost of a packet in OPT is

$$O(\sqrt{h} \cdot wait(h) + h + h \cdot \sqrt{h})$$

Now if $h \leq wait(h)$, then OPT is $O(\sqrt{h} \cdot wait(h))$ while the online algorithm pays $\Omega(h \cdot wait(h))$, and if $h > wait(h)$, then OPT is $O(\sqrt{h} \cdot h)$ while the online algorithm pays $\Omega(h^2)$, yielding the desired lower bound. We conclude with the following theorem.

Theorem 3. *The competitive ratio of any oblivious distributed online algorithm for the multicast aggregation problem is at least* $\Omega(\sqrt{h})$.

Notice that the lower bound that we proved for the centralized online algorithm also applies to our distributed online algorithm, yielding a lower bound of $\Omega(\sqrt{h} + \log \alpha / \log \log \alpha)$.

5 Concluding Remarks

Many important questions remain open. One direction would be to study the performance of non-oblivious algorithms. It would be very useful to understand how much can be gained from the knowledge of the hierarchical tree structure. Another research avenue is to find algorithms that work well for trees with special properties. For example, many group communication protocols might have a very flat tree, with a bounded height of, say, two or three levels. Can they use better aggregation protocols? Another direction which is worth looking into is a model where events in the input sequence are not independent. For example, consider the case where the same tree is used for both multicasting and for collecting Acks (NAKs). In this case, packet loss will trigger an event in all the leaves belonging to its subtree. This requires a different model for the input sequence. Another interesting problem is whether there exists a centralized online algorithm with a constant competitive factor. In this model we are only able to show a constant factor lower bound on the competitive ratio of any algorithm.

Acknowledgments. We thank Guy Even, Baruch Schieber, Bruce Shepherd, and Francis Zane for many stimulating discussions. A special thanks to Yair Bartal for his insightful comments on many aspects of this work.

References

1. B. R. Badrinath and P. Sudame. Gathercast: The design and implementation of a programmable agregation mechanism for the internet. Submitted for publication, 1999.
2. E. Bortnikov and R. Cohen. Schemes for scheduling of control messages by hierarchical protocols. In *IEEE INFOCOM'98*, March 1998.
3. D. R. Dooly, S.A. Goldman, and S. D. Scott. On-line analysis of the TCP acknowledgement delay problem. *Journal of the ACM*, 48:243–273, 2001.
4. M.Hofmann. A generic concept for large-scale multicast. In B. Plattner, editor, *International Zurich Seminar on Digital Communication*, number 1044, pages 95 – 106. Springer Verlag, February 1996.
5. C. Papadimitriou. Computational aspects of organization theory. In *ESA '96*, 1996.
6. C. Papadimitriou and E. Servan-Schreiber. The origins of the deadline: optimizing communication in organization. Workshop on Complexity in Economic Games, Aix-en-Provence, 1999. To appear, Handbook on the Economics of Information.
7. Reliable multicast protocols. `http://www.tascnets.com/newtascnets/Projects/Mist/Documents/RelatedLinks.html`.
8. The ATM Forum technical committee. Private network-network interface specification version 1. 0 (PNNI). Technical report, March 1996. af-pnni-0055.000.
9. Lixia Zhang, Stephen Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala. RSVP: a new resource ReSerVation protocol. *IEEE Network Magazine*, 7(5):8 – 18, September 1993.