Optimal Deterministic Timeouts for Reliable Scalable Multicast

Matthias Grossglauser, *Student Member, IEEE*

Abstract— **Reliable multicast protocols suffer from the problem of feedback implosion. To avoid this problem, the number of receivers sending feedback in case of loss must be small. However, losses experienced by different receivers are strongly correlated, since receivers share common resources in the multicast tree.**

One approach to feedback implosion avoidance relies on delaying feedback at the receivers. We present DTRM (Deterministic Timeouts for Reliable Multicast), a distributed algorithm to compute optimal deterministic timeouts for each receiver in a multicast tree as a function of the tree topology and the sender-to-receiver round-trip delays. DTRM has several desirable properties. First, feedback implosion is provably avoided for a single loss anywhere in the tree, provided delay jitter is bounded. Second, the computation of the timeouts can be entirely distributed; receivers and intermediate nodes only rely on local topology information. Third, the timeouts computed by DTRM are optimal with respect to the maximum response time.

Keywords— **Multicast, reliability, feedback implosion**

I. Introduction

THE availability of high speed transmission technology, the ever increasing performance of workstations and personal computers, and the convergence of the data and telecommunications industry will foster the creation of a powerful networking infrastructure. This environment will enable a whole panoply of new applications, including multimedia and distributed computing. It becomes increasingly clear today that many of these applications will rely on an efficient multicasting service. The recent success of Internet multicasting on the MBone confirms this. Videoconferencing on the Internet is rapidly becoming a standard tool in the research community, despite the rather limited video and audio quality.

Some applications require a *reliable multicast service*, i.e., the error-free transport of information to a group of recipients. Such applications include distribution of "non fault-tolerant" information (such as software, news, or webpages), distributed computing, such as distributed interactive simulation (DIS), or network management. We believe that the importance of this type of service will increase in the future, when distributed computing becomes commonplace.

It has been observed [2] that a receiver-based multicast protocol achieves better scalability than a sender-based one. But even a receiver-based scheme suffers from the NACK[1]-implosion problem if the losses at different receivers are correlated. This is very likely due to the resource-sharing in a multicast tree. When a packet is lost on some link, then all receivers on the subtree fed by this link experience this loss.

There are two approaches to solve this problem: *structure-based* and *timer-based*. In structure-based approaches [3], [4], intermediate nodes in the tree process and combine feedback information. Timer-based approaches, such as the one discussed in this paper, do not rely on processing by network nodes. Rather, they rely on *delayed feedback* to avoid an implosion [5], [6], [7]. They have the advantage of not requiring network support for implosion avoidance, but the potential disadvantage of higher application-to-application delays.

Another dimension in the solution space is receiver collaboration: if receivers are to collaborate, then they have to buffer correctly received data in order to be prepared to respond to other receivers' NACKs [6]. This also implies that feedback has to be multicast to all receivers, i.e. that each receiver is also a multicast source. In this work, we assume that receivers are greedy in the sense that they immediately consume data they have already received. Only the sender buffers data for possible repairs. The receivers *unicast* feedback to the source. Receiver collaboration can be undesirable because (1) the cost of maintaining copies of data at all the receivers instead of only at the source is too high, or because (2) receivers do not trust each other or want to withhold their identity from each other, or because (3) the underlying network does not offer (scalable) multipoint-to-multipoint communication.

Proposals have been made to use random NACK delay timers, which allows to reduce the expected number of NACKs [5], [8], [6]. One reason why only these heuristic approaches have been proposed is that the current Internet does not provide for any Quality of Service guarantees. Next generation networks are expected to offer a more elaborate service model with delay and delay jitter guarantees. Currently, various bodies are working on standardizing such services, e.g. the ATM Forum or the Internet Engineering Task Force (IETF).

This paper presents a rigorous approach to the NACK-implosion problem, based on the assumption that end-to-end delay variation is bounded and known to the endpoints. We compute *deterministic* timeouts as a function of the multicast tree topology and end-to-end delays. We establish an optimality criterion and present an algorithm, called DTRM (for Deterministic Timeouts for Re-

M. Grossglauser is with INRIA, High Speed Networking Group, BP 93, 06902 Sophia Antipolis Cedex, France (email: matthias.grossglauser@inria.fr)

This work was supported in part by a grant from France Telecom/CNET. Part of this work was done while the author was a visitor at AT&T Bell Laboratories, Murray Hill, NJ, USA. See [1] for an earlier version of this paper.

[1] Negative Acknowledgment

liable Multicast), that computes a timeout value for every receiver. In a networking environment with delay guarantees, NACK-implosion is provably avoided. The timeout computation is efficient and can be distributed, in which case neither sender nor receivers need to have knowledge of the group membership or of the complete tree topology. Each receiver only needs to know its round-trip delay to the sender.

The remainder of this paper is structured as follows. Section II discusses related work. Section III presents the context of the problem in more detail. Section IV explains how timeouts have to be set to fulfill the *single-NACK condition*, and establishes an optimality criterion. Section V presents a distributed algorithm to compute an optimal timeout allocation. Section VI discusses the robustness and scalability of the algorithm, and Section VII concludes the paper.

II. Related Work

In [2], Pingali *et al.* present an analytical comparison of three generic reliable multicast protocols. They conclude that a receiver-based approach, where the receivers carry the burden of detecting losses and requesting retransmissions, is preferable to a sender-based approach. They also compare two receiver-based schemes, called N1 and N2. In N1, a receiver unicasts a NACK to the sender immediately after it experiences a loss; the sender then retransmits the packet. In N2, the receiver, upon seeing a loss, waits for a random time and then broadcasts the NACK to the entire multicast group. The authors conclude that N2 performs better than N1, but unfortunately, there is no indication as to whether this advantage stems from the fact that N1 immediately responds to a loss, whereas N2 waits for possible NACKs from other receivers, or from the fact that N2 broadcasts the NACK, whereas N1 unicasts it to the sender. Furthermore, it is argued that only processing overhead, but not bandwidth overhead, is costly, which seems to apply more to LANs than to WANs.

Crowcroft and Paliwoda discuss the implosion problem in the context of group communication, where multiple servers reply to a request from a client [5]. They analyze the performance of randomly delaying responses if all members share the same transmission medium. The idea of randomly delaying feedback by receivers (slotting) and having receivers suppress redundant feedback (damping) has been introduced in the XTP protocol [8]. The same ideas are used in the SRM protocol [6]. The authors of SRM present an adaptive algorithm that adjusts the intervals from which the random timers are chosen to the network condition. This heuristic scheme seems to work well in a networking environment without guarantees. However, scalability is unfortunately not fully achieved, as each multicast group member has to maintain a delay estimate between itself and each other group member.

Papadopoulos and Parulkar [7] define a reliable multicast taxonomy around an hypothetical optimal algorithm. In their taxonomy, our approach is called *sender-controlled implosion control*. They also outline an algorithm that organizes receivers into *buckets*, based on their round trip delay, but not on topology. Before requesting retransmission upon loss, each receiver has to make sure that all the receivers in buckets with smaller round trip delays have not already requested this retransmission. The maximum number of requests can therefore be limited by limiting the maximum number of receivers in each bucket.

Two approaches for structure-based implosion avoidance are discussed in [3], [4]. The fundamental difference between the two approaches is that in [3], a group member (called *Designated Receiver*) is responsible for processing a region's feedback, while in [4], dedicated servers (called *Logging Servers*) perform this function.

The feedback implosion problem has also been addressed in a probabilistic context, such as in congestion control for video distribution [9] and under a relaxed form of multicast reliability called *multicast-to-some* [10].

III. Context

The advantage of timer-based approaches to the feedback implosion problem is that no network processing of feedback is necessary. The solution is entirely *end-to-end*. The network's only role is in forwarding the feedback from the receivers back to the sender.

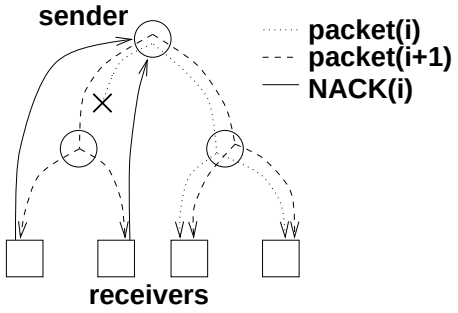


Fig. 1. A packet loss experienced by multiple receivers results in a NACK being sent back to the sender by each of these receivers.

We assume that the end-to-end delay for a packet sent from the sender to a receiver α is constant and equal to d_α^P. Also, we assume that the receiver-to-sender delay for NACKs is constant and equal to d_α^N. We call $d_\alpha = d_\alpha^P + d_\alpha^N$ the *round trip delay* (RTD) of receiver α. Furthermore, we assume that processing at the sender and at the receiver is immediate (retransmission due to a NACK packet at the source, detection of loss at a receiver upon reception of an out-of-sequence packet).

Figure 1 shows how the NACK-retransmission scheme works: if packet i is lost on some link, then each receiver in the subtree attached to that link observes this loss upon reception of packet $i+1$, and starts a timer. When a receiver's timer expires, it sends a NACK(i) packet back to the sender, requesting retransmission of packet i. When a receiver that has started a retransmission timer for packet i receives the retransmitted packet i, it stops the timer without further action.

A NACK-implosion can be avoided if we can make sure that for any loss, one NACK arrives early enough such that

the retransmission of the lost packet caused by this NACK prevents further NACKs from other receivers. For this, the retransmitted packet has to arrive at these other receivers before their timers expire. We refer to this requirement as the *single-NACK condition*. The implicit assumption here is that it is reasonable to prevent NACK-implosion at the cost of increased delays in the case of loss.

IV. Computing Timeouts

In this section, we first discuss how to determine *some* set of timeouts such that the single-NACK condition is met in the entire multicast tree. Then we establish an optimality criterion. Finally, we describe a distributed algorithm to determine an optimal set of timeouts.

Let us introduce some notation. We assume the existence of a multicast tree. The root of this tree is the *sender*. Internal nodes are called *switches*, and leaf nodes *receivers*. Finally, by *downstream* we mean "towards the receivers", and by *upstream* we mean "towards the sender".

A. How to Choose Consistent Timeouts

Assume a packet is lost on a link, and call the subtree fed by this link the *loss subtree*. All receivers in the loss subtree, and only they, experience this loss. Consider two receivers α and β in the loss subtree with round trip delays d_α and d_β and timeouts t_α and t_β, respectively.

Assume that an out-of-sequence packet, i.e. the packet following the lost packet, is sent at time 0. The out-of-sequence packet reaches receiver α at time d_α^P and receiver β at time d_β^P. The timer of receiver α therefore expires at time $t_\alpha + d_\alpha^P$, the one of receiver β at $t_\beta + d_\beta^P$. The resulting NACK from receiver α reaches the source at time $t_\alpha + d_\alpha$, and the retransmitted packet reaches receiver β at time $t_\alpha + d_\alpha + d_\beta^P$. Therefore, to ensure that only receiver α sends a NACK back to the sender when both receiver α and β experience a loss, we must have

$$t_\beta > t_\alpha + d_\alpha. \tag{1}$$

We say that β fulfills the single-NACK condition *with respect to* α. Given a timeout value t_α for receiver α, we enforce the single-NACK condition be setting

$$t_\beta := t_\alpha + d_\alpha + \epsilon, \tag{2}$$

where $\epsilon > 0$ is a small constant to allow for some delay variation without violating (1).

Our goal is to set timeouts for receivers in a multicast tree such that any single loss on a link only results in a single NACK coming back to the source. This can be achieved by assigning to each subtree A a *representative receiver* α with round trip delay d_α and timeout t_α such that every receiver $\beta \neq \alpha$ in this subtree A fulfills condition (1).

We make the following definitions.

Definition 1: We call *subtree* A the subtree rooted at node A[2].

[2]We use the classical definition of *subtree rooted at* A, consisting of its root A and *all* of its descendants [11, page 93].

Definition 2: A *receiver allocation* ϕ for a tree is a mapping of the set τ of nodes onto the set ρ of leaves (receivers), such that $\phi(A)$ is a descendant of A for any $A \in \tau$[3]. The receiver $\phi(A)$ is called the *representative receiver* of node A.

Definition 3: A *consistent timeout allocation* (in subtree A) associates with each receiver i (in subtree A) a non-negative timeout value t_i, such that for any node B with representative receiver $\alpha = \phi(B)$, all other receivers $\beta \neq \alpha$ in subtree B fulfill the single-NACK condition (1) with respect to α.

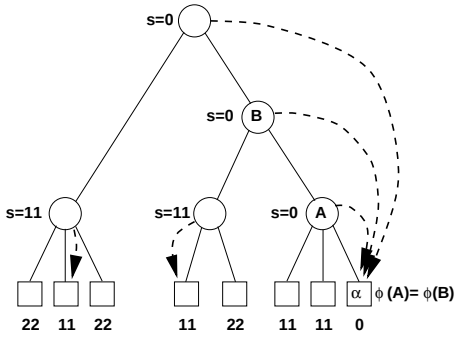


Fig. 2. An example receiver allocation, and the corresponding consistent timeout allocation.

Figure 2 gives an example of a receiver allocation and the corresponding consistent timeout allocation. The round trip delay d_i has been set to 10 for all receivers i, and $\epsilon = 1$. A dashed arrow from a node to a receiver means that this receiver is the representative receiver of this node. The numbers below the receivers are the timeout values.

Note the following property that is formalized in the lemma below. Suppose a node A is a descendant of a node B (cf. Fig. 2). Furthermore, suppose that B's representative $\alpha = \phi(B)$ lies in subtree A. All receivers in subtree B, and in particular all receivers in subtree A, must fulfill the single-NACK property with respect to α. Therefore, α is also A's representative.

Lemma 1: If $\phi(B) = \alpha$, then any descendant A of B such that α is in subtree A has $\phi(A) = \alpha$.

Proof: This follows directly from the previous paragraph. ■

Another way to put this is the following. A receiver α is a representative receiver for a consecutive sequence of ancestors up to the first ancestor that has a different representative receiver. No ancestor further upstream has α as its representative receiver.

We now give the algorithm that determines the consistent timeout allocation in a subtree A, given a receiver allocation ϕ. We need the following definitions.

Definition 4: The *set of cotrees* of a receiver α is the set of subtrees of A not containing α, that are rooted at children of some node on the path from the sender to α.

The following definition will be used later in the discussion of lost NACKs.

[3]Note that if A is a receiver, $\phi(A) = A$.

Definition 5: The *set of dependent cotrees* of a receiver α for a receiver allocation ϕ contains those cotrees of α whose root's parent has α as its representative receiver.

Consider Figure 3 for an illustration. The idea behind cotrees is the following. Assume that the timeout allocation in each cotree of α is already consistent. Then each cotree is a possible source of *one* unwanted NACK when A is the loss subtree and α is A's representative. By adding a constant offset s to each receiver's timeout within a cotree B, we can fulfill the single-NACK property with respect to α while maintaining the timeout allocation consistent in cotree B. This is the idea behind the algorithm below. Given a receiver allocation ϕ, it recursively computes this offset s to be added to every cotree of the representative receiver of every node in the tree (cf. Figure 2.) This results in the consistent timeout allocation in subtree A.

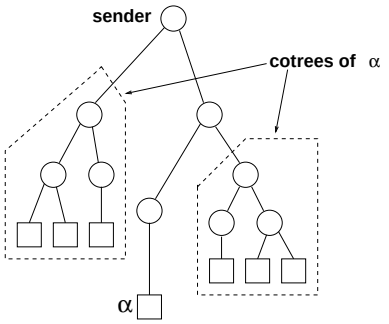


Fig. 3. The cotrees of receiver α.

Algorithm 1: proc **comp (A: node, s: non-negative real, ϕ: receiver allocation)**

```
1   α:=φ(A).
2   t_α:=s.
3   for all cotrees B of α,
4       comp (B, t_α + d_α + ε, φ).
5   endfor
```

The first two steps set the timeout of node A's representative receiver $\phi(A)$ such that it respects the single-NACK condition with respect to representative receivers of A's ancestors that lie outside subtree A. The remaining steps recursively compute the consistent timeout allocations in $\phi(A)$'s cotrees, with an offset such that all the receivers in these cotrees fulfill the single-NACK condition with respect to $\phi(A)$. Therefore, given a receiver allocation ϕ, **comp (sender, 0,** ϕ**)** computes the consistent timeout allocation for the entire tree.

B. Optimal Timeouts

Now that we are able to compute the consistent timeout allocation given the receiver allocation ϕ, i.e., given a representative receiver for each node, we need to establish a sensible cost function to compare receiver allocations. For this, let us consider for a moment how a reliable end-to-end transport protocol based on acknowledgments (negative or positive) works. The goal of the protocol is to deliver data to the application error-free and in order. This means that the transport protocol layer buffers all data following a loss, until the lost data has been retransmitted. Only then can the data be handed over to the application. Now, in our framework, observe what happens to a receiver α in the worst case (the case where nobody else's NACK arrives at the sender earlier than α's) between the moment when the receiver realizes a packet has been lost, and the moment when this packet arrives at this receiver. The receiver runs its timer for time t_α, and then sends a NACK, which takes time d_α^N to arrive at the sender. The retransmitted packet then takes d_α^P to arrive at α. The total *response time* is $t_\alpha + d_\alpha$.

It makes sense to minimize the largest response time $t_\alpha + d_\alpha$, both to minimize the buffering necessary in the transport protocol layer, and to minimize the delay seen by the application. Furthermore, the transport layer of the sender also has to perform buffering, to allow for the retransmission of lost packets. The sender can be sure that all receivers have received a packet if it has not received a NACK after $\max\{t_\alpha + d_\alpha\}$, where we are maximizing over all the receivers in the multicast tree[4].

We therefore want to minimize

$$T = \max_{\alpha \in \rho} \{t_\alpha + d_\alpha\} \tag{3}$$

where ρ is the set of all receivers in the multicast tree.

C. Computing an Optimal Receiver Allocation

In the previous section, we have established a cost function for receiver allocations. We now turn to the core of the problem, which is to find an efficient algorithm to compute the optimal receiver allocation, i.e., the one minimizing the cost function (3). This is a combinatorial problem, where the number of possible receiver allocations is potentially very large. In this section, we show that an efficient algorithm does indeed exist. It is based on the fact that only a small number of receiver allocations within any subtree can correspond to the entire tree's optimal receiver allocation. Therefore, this problem allows for a *divide-and-conquer* approach, where the possible receiver allocations within a subtree A can be computed from the possible allocations of all the subtrees rooted at A's children.

Consider a subtree A with n links and n children $A_1, \ldots, A_n$ (cf. Figure 4). Assume that the receiver allocation *inside* the children are fixed, and that the timeout

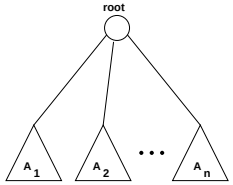


Fig. 4. A subtree, consisting of a root, n links and n children.

[4]provided the NACK and the retransmitted packet are not lost themselves.

allocation inside the children are consistent, as given by **comp** $(A_i, \mathbf{0}, \phi)$.

Lemma 1 says that our only degree of freedom in choosing an allocation for A is choosing one of its children's representative as A's representative. If we choose the representative α_i of A_i as the representative of A, then we have to add an offset $s_{A_i}[\phi]$ to all the timeouts of receivers in the other children $\{A_1, \ldots, A_n\} \setminus \{A_i\}$, because these children are cotrees of α_i. The timeouts of the receivers in A_i remain unchanged.

We now make the following definition:

Definition 6: For a node A and a receiver allocation ϕ, the triple $(s_A[\phi], T_A[\phi], \phi)$ is given by

$$\begin{aligned} s_A[\phi] &= d_{\phi(A)} + \epsilon \\ T_A[\phi] &= \max_{\alpha \in \rho_A} \{t_\alpha + d_\alpha\} \end{aligned} \tag{4}$$

where ρ_A is the set of receivers of subtree A and where the t_α represent a consistent timeout allocation in subtree A.

For a subtree A, $T_A[\phi]$ is the maximum response time (i.e. the cost) of subtree A, and $s_A[\phi]$ is the offset that we have to add to A's siblings if we choose A's representative as A's parent's representative. We call the set of all triples $(s_A[\phi], T_A[\phi], \phi)$ for a subtree A the *characteristic set* of A. Note from (4) that $s_A[\phi]$ and $T_A[\phi]$ only depend on the receiver allocation in subtree A. In other words, two different receiver allocations that are equal within subtree A do not give rise to two elements in the characteristic set. For ease of notation, we do not make this explicit. As an

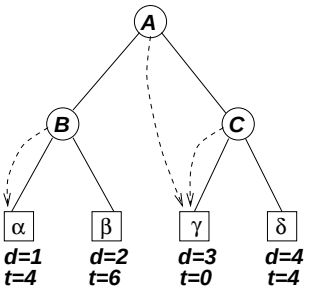


Fig. 5. An example tree; an optimal allocation and the resulting timeouts with $\epsilon = 1$ are shown.

example, the following tables gives the characteristic sets of nodes A, B and C of the tree in Figure 5, as well as the corresponding consistent timeout allocation. The optimal allocation shown in Figure 5 is framed.

$s_B[\phi]$	$T_B[\phi]$	ϕ	t_α	t_β
2	4	$B \mapsto \alpha$	0	2
3	4	$B \mapsto \beta$	3	0

$s_C[\phi]$	$T_C[\phi]$	ϕ	t_γ	t_δ
4	8	$C \mapsto \gamma$	0	4
5	8	$C \mapsto \delta$	5	0

$s_A[\phi]$	$T_A[\phi]$	ϕ	t_α	t_β	t_γ	t_δ
2	10	$A \mapsto \alpha$, $B \mapsto \alpha$, $C \mapsto \gamma$	0	2	2	6
2	10	$A \mapsto \alpha$, $B \mapsto \alpha$, $C \mapsto \delta$	0	2	7	2
3	11	$A \mapsto \beta$, $B \mapsto \beta$, $C \mapsto \gamma$	3	0	3	7
3	11	$A \mapsto \beta$, $B \mapsto \beta$, $C \mapsto \delta$	3	0	8	3
4	8	$A \mapsto \gamma$, $B \mapsto \alpha$, $C \mapsto \gamma$	4	6	0	4
4	8	$A \mapsto \gamma$, $B \mapsto \beta$, $C \mapsto \gamma$	7	4	0	4
5	9	$A \mapsto \delta$, $B \mapsto \alpha$, $C \mapsto \delta$	5	7	5	0
5	9	$A \mapsto \delta$, $B \mapsto \beta$, $C \mapsto \delta$	8	5	5	0

We now show that the characteristic set of each node can be computed from the characteristic sets of its children, which means that the optimal allocation ϕ_{OPT} can be computed by divide-and-conquer. If $\phi(A_l) = \phi(A)$, in other words, if A_l's representative is also A's representative, then it follows from Definition 6 and Algorithm 1 that

$$\begin{aligned} s_A[\phi] &= s_{A_l}[\phi] \\ T_A[\phi] &= \max(T_{A_l}[\phi], \max_{i \in \{1,\ldots,n\} \setminus \{l\}} \{s_{A_l}[\phi] + T_{A_i}[\phi]\}) \end{aligned} \tag{5}$$

For a receiver α, the characteristic set contains only the following element.

$$\begin{aligned} s_A[\phi] &= d_\alpha + \epsilon \\ T_A[\phi] &= d_\alpha \\ \phi(A) &= \{A \mapsto \alpha\} \end{aligned} \tag{6}$$

This follows readily from Definition 6.

The previous observations can now be used to devise a recursive algorithm that finds an optimal allocation ϕ_{OPT} that minimizes $T_A[\phi]$. Any non-leaf node (switch or sender) in the tree can compute its characteristic set based solely on the characteristic sets of its children using (5). The optimal allocation ϕ_{OPT} is then simply the one minimizing $T_S[.]$, where S is the sender. To find each receiver's timeout, Algorithm 1 can be applied.

Let us look at the size of A's characteristic set as a function of the sizes of the children's characteristic sets. Let m_i be the size of A_i's characteristic set. From (5) we see that we have n choices for placing A's representative in one of A's children. Furthermore, in each child A_i, we can choose among m_i allocations. Thus, the size of A's characteristic set is

$$m = n \cdot \prod_{i=1}^{n} m_i. \tag{7}$$

This algorithm suffers from the combinatorial explosion of the size of the characteristic set, as expressed by (7). In fact, this algorithm does nothing else than explicitly enumerate all possible receiver allocations for the entire tree and compute the cost for each of these allocations. This, of course, becomes rapidly impractical as the tree size increases. In the remainder of this section, we show that only a small subset of the characteristic set, which we refer to as the *constrained characteristic set* (CCS), is necessary for each node in the computation of ϕ_{OPT}, and we find a tight upper bound on the size of this set.

Lemma 2: Consider two elements $(s_A[\phi], T_A[\phi], \phi)$ and $(s_A[\psi], T_A[\psi], \psi)$ in the characteristic set of a node A, where ϕ and ψ differ in subtree A. Assume that $s_A[\phi] \geq s_A[\psi]$ and $T_A[\phi] \geq T_A[\psi]$. Then an optimal receiver allocation ϕ_{OPT} can be found without considering $(s_A[\phi], T_A[\phi], \phi)$ in the computation of the characteristic set of A's parent[5].

Proof: Assume that ϕ and ψ are identical outside subtree A, i.e. that for any X not a descendant of A, we have either $\phi(X) = \psi(X) \notin \rho_A$ (the set of subtree A's receivers), or both $\phi(X) \in A$ and $\psi(X) \in A$ (but possibly different). This poses no restriction on the proof, as the elements of A's characteristic set only depend on the allocation within subtree A. Let B be A's parent.

Each element in a node's characteristic set corresponds to a receiver allocation in the subtree rooted at this node. Call $(s_B[\phi], T_B[\phi], \phi)$ and $(s_B[\psi], T_B[\psi], \psi)$ the elements of B's characteristic set resulting from (5).

Consider the two cases where (a) B's representative is in subtree A, and (b) where it is not. In case (a), it follows from Definition 6 and the assumption in Lemma 2 that

$$s_B[\phi] = s_A[\phi] \geq s_A[\psi] = s_B[\psi]. \tag{8}$$

Thus,

$$\begin{aligned} T_B[\phi] &= \max(T_A[\phi], \max_{X \in c(B) \setminus \{A\}} \{s_A[\phi] + T_X[\phi]\}) \\ &\geq \max(T_A[\psi], \max_{X \in c(B) \setminus \{A\}} \{s_A[\psi] + T_X[\psi]\}) \\ &= T_B[\psi]. \end{aligned} \tag{9}$$

because $T_A[\phi] \geq T_A[\psi]$ by assumption and $T_X[\phi] = T_X[\psi]$ because X is outside subtree A (i.e. X is not a descendant of A), and where $c(B)$ represents the set of B's children.

In case (b), let Y be B's child such that subtree Y contains B's representative receiver $\phi(B) = \psi(B)$. Then

$$s_B[\phi] = s_Y[\phi] = s_Y[\psi] = s_B[\psi]. \tag{10}$$

The fact that X and Y are outside subtree A implies that

$$\begin{aligned} T_B[\phi] &= \max(T_Y[\phi], \max_{X \in c(B) \setminus \{Y,A\}} \{s_Y[\phi] + T_X[\phi]\}, \\ &\quad s_Y[\phi] + T_A[\phi]) \\ &\geq \max(T_Y[\psi], \max_{X \in c(B) \setminus \{Y,A\}} \{s_Y[\psi] + T_X[\psi]\}, \\ &\quad s_Y[\psi] + T_A[\psi]) \\ &= T_B[\psi]. \end{aligned} \tag{11}$$

It follows that the same property that holds for $(s_A[\phi], T_A[\phi], \phi)$ and $(s_A[\psi], T_A[\psi], \psi)$ holds for the two elements of B's characteristic set as well, namely

$$\begin{aligned} s_B[\phi] &\geq s_B[\psi] \\ T_B[\phi] &\geq T_B[\psi] \end{aligned}$$

By induction, it holds for two elements of the characteristic set of sender S. As we need to find the element $(s_S[\phi], T_S[\phi], \phi)$ that minimizes $T_S[\phi]$, it follows that there cannot be a single optimal allocation ϕ as there always exists another allocation ψ that is at least as good, i.e. has an equal or lower T_S. Therefore, to find *some* ϕ_{OPT} through (5), it is not necessary to consider $(s_A[\phi], T_A[\phi], \phi)$. This completes the proof. ■

The next lemma states that there exists a global optimum such that if B is any node and A is a child of B, then the receiver allocation in subtree A is locally optimal (i.e. it minimizes $T_A[\phi]$) if B's representative $\phi(B)$ is *not* in subtree A.

Lemma 3: Assume a node A with n children $A_1, \ldots, A_n$, and assume that A_i's representative $\phi(A_i)$ is selected as A's representative. Then in order to find a globally optimal allocation, it is only necessary to consider the elements $(s_{A_j}[\phi], T_{A_j}[\phi], \phi)$ of the characteristic set of A_j, $j \in \{1, \ldots, n\} \setminus \{i\}$ that minimize $T_{A_j}[\phi]$.

Proof: As A's representative is not in subtree A_j, $s_A[\phi] = s_{A_i}[\phi]$ is independent of the choice of receiver allocation in subtree A_j. Also,

$$T_A[\phi] = \max(T_{A_i}[\phi], \max_{j \in \{1,\ldots,n\} \setminus \{i\}} \{s_{A_i}[\phi] + T_{A_j}[\phi]\}) \tag{12}$$

shows that selecting a receiver allocation ϕ that does not minimize $T_{A_j}[\phi]$ can only increase $T_A[\phi]$. Thus, selecting a ϕ that does not minimize $T_{A_j}[\phi]$ results in elements in A's characteristic set that need not be considered by virtue of Lemma 2. Therefore, if A's representative is not in subtree A_j, we only have to consider elements of A_j's characteristic set that has minimal $T_{A_j}[\phi]$. ■

Lemma 2 and Lemma 3 increase the efficiency of the recursive algorithm to find ϕ_{OPT}. Lemma 2 limits the size of the characteristic set of a node, and Lemma 3 makes the computation of a node's characteristic set from its children's characteristic sets more efficient, by limiting the number of combinations that have to be considered. We call the subset of the characteristic set that needs to be taken into account at each node the *constrained characteristic set (CCS)*.

We now prove a theorem that gives a tight upper bound on the size of the constrained characteristic set if in the recursive computation, elements are deleted at each node according to Lemma 2.

Theorem 1: Let h denote the height of a node (and of the subtree rooted at this node), defined as the number of links on the longest path from this node to some receiver in the subtree rooted at this node[6]. If elements of the characteristic set of a node are deleted according to Lemma 2, then the size of this set is at most $\max(h, 1)$.

Proof: The proof makes use of the following two lemmas.

Lemma 4: A trivial node of height 0 (i.e. a receiver) has a constrained characteristic set of size 1.

Proof: This follows readily from the fact that there is only one possible receiver allocation for a single node, as noted previously (cf. Eqn. (6)). ■

[5]In the special case of two equalities, we need to consider one of the two elements, but not both.

[6]For example, the tree shown in Figure 2 is of height 3.

Lemma 5: A tree of height 1 has a constrained characteristic set of size 1.

Proof: A subtree A of height 1 consists of a node and n receivers $\alpha_1, \ldots, \alpha_n$ directly attached to this node. We can assume w.l.g. that $d_{\alpha_1} < d_{\alpha_2} < \ldots < d_{\alpha_n}$. If two round-trip delays were equal, allocating one or the other is equivalent, and it is therefore enough to consider one of them.

There are n elements in the characteristic set of this tree, corresponding to selecting one of the n receivers as the representative. If receiver α_i is chosen as representative, then the element of the characteristic set, denoted by $(s_A[\phi_i], T_A[\phi_i], \phi_i)$ can be computed from (6) and (5).

$$\begin{aligned} s_A[\phi_i] &= d_{\alpha_i} + \epsilon \\ T_A[\phi_i] &= \max(T_A[\phi_i], \max_{j \in \{1,\ldots,n\} \setminus \{i\}} \{s_A[\phi_i] + T_A[\phi_j]\}) \\ &= d_{\alpha_i} + \epsilon + \max_{j \in \{1,\ldots,n\} \setminus \{i\}} \{d_{\alpha_j}\} \end{aligned} \quad (13)$$

Therefore, $s_A[\phi_1] < s_A[\phi_2] < \ldots < s_A[\phi_n]$. Also, as $\max_{j \in \{1,\ldots,n\} \setminus \{i\}} \{d_{\alpha_j}\} = d_{\alpha_n}$ whenever $i < n$, it follows that $T_A[\phi_1] < T_A[\phi_2] < \ldots < T_A[\phi_{n-1}]$. As for $T_A[\phi_n]$,

$$T_A[\phi_n] = d_{\alpha_n} + \epsilon + d_{\alpha_{n-1}} \geq T_A[\phi_1] = d_{\alpha_1} + \epsilon + d_{\alpha_1} \quad (14)$$

In particular, equality only results for $n = 2$. Therefore, there is only the element corresponding to allocating receiver α_1 with smallest delay as A's representative in CCS_A. ■

We have proved the assertion for the cases where $h = 0$ and $h = 1$. We are now ready to complete the proof for any h. For this, consider a subtree A and denote the receiver in A with the smallest round-trip time with $\alpha_{\min}$ (assuming it is unique.) Denote the roots of the cotrees of $\alpha_{\min}$ in subtree A with $A_1, \ldots, A_n$, as shown in Figure 6. We prove the theorem in two steps. First, we show

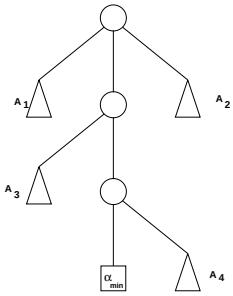


Fig. 6. The cotrees of receiver $\alpha_{\min}$ with the lowest round-trip delay in the entire subtree.

that there is exactly one element in CCS_A corresponding to allocating $\alpha_{\min}$ as A's representative. Second, we show that all other possible elements in CCS_A correspond to selecting A's representative in one particular cotree of $\alpha_{\min}$. This results in the desired bound on the size of CCS_A.

To show that there is exactly one element in CCS_A corresponding to selecting $\alpha_{\min}$ as A's representative, note that this choice corresponds to an element $(s_A[\phi], T_A[\phi], \phi)$ of CCS_A that minimizes $s_A[\phi] = d_{\alpha_{\min}} + \epsilon$, by definition of $\alpha_{\min}$. Among all allocations that have $\alpha_{\min}$ as A's representative, by virtue of Lemma 3, only one allocation minimizing $T_A[\phi]$ corresponds to an element in CCS_A.

To bound the number of other elements in CCS_A, let $T^{OPT}_{A_i}$ be the minimal $T_{A_i}[.]$ in cotree A_i. Let $T^{OPT}_{DC} = \max_{i=1,\ldots,n}\{T^{OPT}_{A_i}\}$, and the corresponding cotree the *dominant cotree DC* (assuming, for the moment, that it is unique). This is the cotree with the highest cost for its local optimum.

Lemma 3 states that if $\alpha_{\min}$ is A's representative, then a necessary condition for an allocation to be in CCS_A is that it corresponds to the locally optimal receiver allocation in each cotree of $\alpha_{\min}$. Therefore, $T_A[\phi] = s_A[\phi] + T^{OPT}_{DC}$.

Let ψ be another allocation such that A's representative $\alpha = \psi(A) \neq \alpha_{\min}$ is not in the dominant cotree DC. It follows that DC is either a cotree of α, or it is a subtree of a cotree of α. For the single-NACK property to hold with respect to α, the offset $s_A[\psi] = d_\alpha + \epsilon$ has to be added to all the timeouts in α's cotree, and in particular, to the timeouts in DC. Then $T_A[\psi]$ must be at least $s_A[\psi] + T^{OPT}_{DC}$, as by definition T^{OPT}_{DC} minimizes $T_{DC}[.]$. As $s_A[\psi] > s_A[\phi]$ and therefore $T_A[\psi] > T_A[\phi]$, ψ is not in A's CCS (Lemma 2).

Thus, A's CCS has always exactly one element corresponding to $\alpha_{\min}$ as A's representative, and possibly other elements corresponding to allocations that place A's representative α in $\alpha_{\min}$'s dominant cotree DC. For each such choice of α in DC, the same argument as before can be used to show that there is only one allocation ψ possible in A's CCS: α uniquely determines $s_A[\psi]$, and thus only the element minimizing $T_A[\psi]$ survives.

It follows that $\|\mathrm{CCS}_A\| \leq \|\mathrm{CCS}_{DC}\| + 1$. Also, height (A) $\geq$ height (dominant cotree)$+1$. Therefore, by induction anchored on Lemma 5, the assertion is proved.

If $\alpha_{\min}$'s dominant cotree DC is not unique, i.e. $T^{OPT}_{DC_1} = T^{OPT}_{DC_2} = T^{OPT}_{DC}$, then A's CCS has only one element, corresponding to $\phi(A) = \alpha_{\min}$. To see this, assume there are two dominant cotrees, DC_1 and DC_2. Let ψ be an allocation such that $\psi(A)$ is in DC_1. Due to DC_2, $T_A[\psi]$ must be at least $s_A[\psi] + T^{OPT}_{DC}$. Therefore, there is no element in A's CCS corresponding to ψ. This is easily extended to more than two dominant cotrees. This completes the proof. ■

V. Distributed Optimal Timeout Computation

This section explains how the computation of each node's CCS, and therefore the computation of the optimal receiver allocation and of the corresponding consistent timeout allocation, can be distributed. The key result from the previous section, namely the bound on the size of a node's CCS, limits the size of the messages that have to be exchanged in the distributed computation. We also show that only little state has to be maintained in nodes.

Each element of a node's CCS corresponds to a "candidate receiver allocation" in the subtree rooted at this node, i.e. an allocation that *might* be globally optimal. Suppose that a node that has computed its own CCS from its children's CCSs does not know the complete allocation

function ϕ corresponding to each element of its CCS, but only (a) in what child subtree lies its own representative receiver, and (b) which among the possible candidate allocations should be chosen in each child. Then it can be seen that any node's representative receiver for some receiver allocation in this node's CCS can be determined in the following way: start at this node and note which is the candidate receiver allocation in the child containing the representative receiver. Go to this child node, which knows in which of its children is its representative receiver for this candidate application. Do this recursively until you reach the representative receiver.

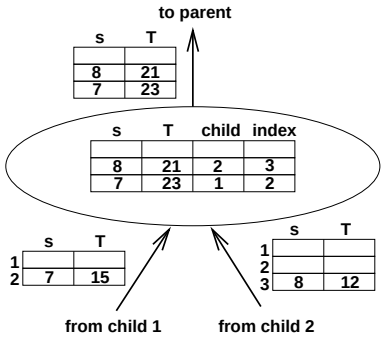


Fig. 7. The topology information associated with the allocations in a node's CCS can be distributed.

Node	child CCS	own CCS
B	$\text{CCS}_\alpha = \{(2,1)\}$ $\text{CCS}_\beta = \{(3,2)\}$	$\{(2,4,\alpha,1)\}$
C	$\text{CCS}_\gamma = \{(4,3)\}$ $\text{CCS}_\delta = \{(5,4)\}$	$\{(4,8,\gamma,1)\}$
A	$\text{CCS}_B = \{(2,4)\}$ $\text{CCS}_C = \{(4,8)\}$	$\{(2,10,B,1), \underline{(4,8,C,1)}\}$

TABLE I

The upstream part of the distributed timeout computation.

Node	child	index	offset
A	B	1	4
	C	1	0
B	α	1	4
	β	1	6
C	γ	1	0
	δ	1	4

TABLE II

The downstream part of the computation.

Furthermore, Lemma 3 states that we need to know (b) above actually only for the child subtree containing the representative receiver, because the other children's allocation is locally optimal. Thus, instead of the full allocation function ϕ, we only need to associate with each CCS element (a) a pointer to the child containing the representative receiver, and (b) an index into that child's CCS. Figure 7 illustrates this: the node receives two CCSs from its two children, and computes its own CCS. Internally, it maintains for each element of its CCS the pointer to the child subtree containing the representative receiver, and an index into that child's CCS. It sends its own CCS to the parent. In the end, the root knows its own CCS. It can then pick the element (s_{OPT}, T_{OPT}) that minimizes the cost function T.

The downstream part of the algorithm computes the actual timeouts. Each node receives an index into its CCS and an offset from its parent. Call this offset S and the selected CCS element $(s, T, child, index)$. Each node now sends to its representative child *child* the offset S and *index*. It sends to all of its other children an offset $S + s$, and as index the one corresponding to the child's local optimum (cf. Lemma 3). If the node is a receiver, then the offset S it receives from its parent is its timeout value.

The following example illustrates the distributed algorithm. The topology of the network is depicted in Figure 5. Table I shows the computation each node performs in the upstream part of the protocol. The element chosen in the root's CCS as the best is underlined. Compare this to A's full characteristic set in Section IV-C.

Table II shows the downstream part of the protocol. The resulting timeouts are reported in Figure 5. It is interesting to note that the receiver with the smallest round-trip delay is not the tree's representative in the optimal allocation.

VI. Discussion

The algorithm shown in the previous section computes an optimal set of timeouts. Theorem 1 bounds the size of the upstream messages by the height of a node. In a balanced tree with n receivers and outdegree m, this height is $O(\log_m(n))$. For example, assume that a multicast group has 4096 participants, and that each switch only has two children ($m = 2$). Then the constrained characteristic set of the root is at most of length $\log_2(4096) = 12$. Note that no node in the tree has to know the entire topology, and that communication is local: each node only communicates with its parent and its children. For general trees, the number of elements in the upstream messages can never exceed the diameter of the underlying network topology.

In this section, we discuss other aspects of scalability and robustness. We address the issues of lost NACKs and delay variation. We then examine the impact of the tree topology on the timeout values and on the response times as computed by DTRM. Finally, we discuss how to handle dynamic group membership changes.

A. Lost NACKs and Delay Variation

Let us briefly consider what happens if the unique NACK gets lost. In this case, every dependent cotree (cf. Definition 5) of the receiver having produced the NACK sends an additional NACK. This number is bounded above by

$h_l(m-1)$, where h_l is the height of the loss tree and m is the largest outdegree of any node in the loss tree. In the example above with 4096 receivers, this number would be 12 if the loss tree is the entire tree. This illustrates that a NACK loss results in a reasonable number of additional NACKs, placing only a small burden on the sender.

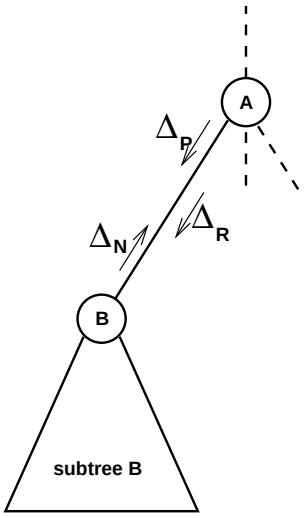


Fig. 8. Delay variation on a link (A, B): Δ_P is the delay variation on this link of the out-of-sequence packet, Δ_N is the delay variation of the NACK packet, and Δ_R is the delay variation of the retransmitted packet.

While the underlying assumption of DTRM is that delay variation remains bounded and can be absorbed by the parameter ϵ, it is interesting to observe that the same property of graceful degradation holds for delay variation on a link (cf. Fig. 8).

Let us assume that on some link AB, the delay of the original out-of-sequence packet is varied by Δ_P and that of the retransmitted packet by Δ_R. Also, if the NACK originates in subtree B, let its delay be varied by Δ_N. Let L denote the root of the loss subtree. We have to distinguish three cases.

1. The loss occurs on or below AB: In this case, if $\Delta_N + \Delta_R > \epsilon$, one additional NACK is produced by each dependent cotree of $\phi(L)$. The situation is equivalent to a lost NACK.
2. The loss occurs above AB and $\phi(L)$ lies in subtree B: Let us first look at additional NACKs from within subtree B. If $\Delta_N + \Delta_R > \epsilon$, then each cotree of $\phi(L)$ in subtree B produces an additional NACK (as $\phi(L)$ is B's representative, all of these cotrees are dependent). Next, let us look at additional NACKs from outside subtree B. If $\Delta_P + \Delta_N > \epsilon$, then each dependent cotree of $\phi(L)$ outside subtree B produces an additional NACK. Thus, if both conditions are true, then there is an additional NACK from each dependent cotree of $\phi(L)$. Again, this is equivalent to a lost NACK.
3. The loss occurs above AB and $\phi(L)$ is not in subtree B: In this case, no additional NACKs can be produced outside subtree B. If $-\Delta_P + \Delta_R > \epsilon$, then subtree B produces one additional NACK.

Host	RTD	Optimal timeout
`ibp.fr`	22ms	118ms
`sics.se`	70ms	118ms
`umd.edu`	108ms	0ms
`virginia.edu`	113ms	118ms
`umass.edu`	119ms	241ms
`uky.edu`	154ms	198ms
`utexas.edu`	157ms	241ms
`berkeley.edu`	172ms	241ms
`usc.edu`	177ms	241ms
`washington.edu`	181ms	241ms
`fokus.gmd.edu`	273ms	118ms

TABLE III
THE RECEIVERS OF THE MBONE SESSION, WITH THE RTDS FROM `pax.inria.fr`.

Therefore, delay variation at a single link can at most produce as many additional NACKs as a lost NACK. The situation where delay variations occur on all links is hard to analyze. The number of NACKs produced per loss will likely depend on many factors, such as the distribution of delay variation, and its temporal and spatial correlation. We do not propose DTRM for such an environment of unpredictable, unbounded delay variations. It is likely that only adaptive schemes that compute timeouts based on measurements of the network condition will be sufficiently robust. It is a reasonable goal for such heuristic, dynamic schemes to get as close as possible to the performance of a provably optimal scheme in idealized conditions. Thus, we advocate the use of DTRM as a performance evaluation baseline of such heuristics. On the other hand, if the network can make some delay jitter guarantees, which is a quality of service parameter that future networks, such as ATM, are expected to support, then the desirable properties of DTRM (timeout optimality, graceful degradation) can be fully exploited.

B. Timeouts and Response Times

We study the timeouts and response times computed by DTRM in two different scenarios. First, we take a typical MBone session, with participants in the US and in Europe, and apply DTRM in order to get a feel for the timeout values obtained in a realistic setting. Second, we give a bound on the maximum response time, and we compute the average response times for balanced trees where all the receivers have equal round-trip delays.

The MBone session membership we have used is reported in [12] (cf. Table III). We assume that the IP host `pax.inria.fr` at INRIA is the sender and the other 11 hosts are the receivers. We have used `ping` to measure round-trip delays between `pax` and all the receivers. We have then used `traceroute` to determine the tree topology. Note that `traceroute` provides us not with the MBone topology, but with the tree topology that would result if all the IP routers were multicast-capable. Given that the

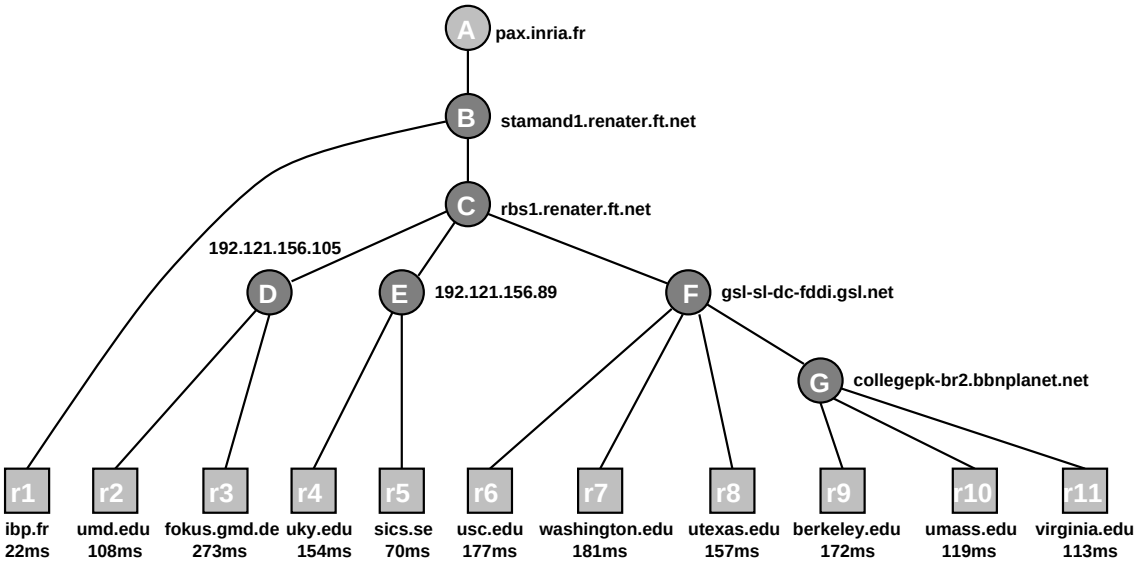


Fig. 9. The tree topology and the round-trip delays for a typical MBone multicast session.

MBone is an overlay network to deal with old (i.e., non multicast-capable) IP routers, this approach yields more realistic topologies than simply using the current MBone tree.

We can make several interesting observations from the timeout values obtained in this setting (with $\epsilon = 10$ms). First, the largest timeout (241ms) in the tree is smaller than the largest round-trip delay[7] of 273ms. The cost of this timeout allocation, i.e. the maximum response time, is 422ms, for receiver `washington.edu` (181ms + 241ms). Thus, the timeout values obtained in this example group are quite reasonable. A maximum response time of less than twice the largest round-trip delay in a multicast session spanning such a large geographical area is reassuring.

However, the above group, of course, is quite small (11 receivers). We therefore finish this subsection with a discussion of upper bounds of the maximum response time, and with an expression for the average response time for balanced trees. We first obtain an upper bound on the largest timeout value t_i occurring in the tree. For this, assume an arbitrary allocation. As mentioned earlier, the timeout of each receiver is the sum of all $d_i + \epsilon$ for all receivers i that are representatives of ancestors of i. It is straightforward to observe that in a tree of height h, this sum cannot be larger than $h(d_{\max} + \epsilon)$, where $d_{\max}$ is the largest delay of any receiver in the tree. Thus, the maximum response time is bounded from above by

$$h(d_{\max} + \epsilon) + d_{\max}. \tag{15}$$

Again, if the tree is balanced, then $h = O(\log_m(n))$.

We now discuss the average response time, which we define as follows. Assume that a single loss occurs with equal probability on any link of the tree[8]. Then the average response time is the expected value of $d_i + t_i$, i.e. the response time of receiver i, where i is the representative receiver of the loss subtree. For the sake of exposition, assume the following balanced tree. Each switch in the tree has outdegree m. The tree is of height h and has exactly m^h receivers. Furthermore, all receivers have identical round-trip delays equal to d. Using this tree has the advantage of making all allocations equivalent in terms of the resulting timeouts. We denote the average response time by $\overline{t}_h$.

The average response time $\overline{t}_h$ can be found through a simple recursion. For a tree of height h, the average response time can be expressed as

$$\overline{t}_h = \frac{T_h}{n_h}, \tag{16}$$

where

$$T_h = \sum_{A \in \tau} d_{\phi(A)} + t_{\phi(A)} \tag{17}$$

is the sum of the response time of the representative receivers of all tree nodes, and where

$$n_h = \frac{m^{h+1} - 1}{m - 1} \tag{18}$$

is the number of nodes in the tree. Note that T_{h+1} can be written in the following way

$$T_{h+1} = mT_h + (m-1)n_h(d + \epsilon) + d. \tag{19}$$

The term mT_h is due to the fact that a tree of height $h+1$ contains m subtrees of height h. The term $(m-1)n_h(d + \epsilon)$ is due to the offset $d + \epsilon$ that we have to add to each receiver's timeout in all but one of these subtrees in order to fulfill the single-NACK condition with respect to the tree's representative. Finally, the term d is due to the fact that the tree's representative (which has $t = 0$) has a response time of d. This recursion yields

$$\overline{t}_h = \frac{m^h(m-1)}{m^{h+1} - 1}[d + h(d + \epsilon)] - \frac{m^h - 1}{m^{h+1} - 1}\epsilon. \tag{20}$$

[7] Note that he largest RTD is between INRIA, France, and the geographically second-closest site, GMD Fokus in Germany.

[8] Including a "virtual" link feeding the root, as we do not exclude the case where the loss subtree is the entire tree.

In summary, both the maximum and the average response time scale like $O(\log_m(n))$, which ensures scalability even to very large groups. Furthermore, we have observed that these bounds are quite pessimistic for heterogeneous topologies and RTD distributions. For example, note that for the tree topology depicted in Fig. 9, the actual maximum response time is 422ms, while $h = 5$ and $d_{\max} =$ 273ms results in a bound of 1688ms. The actual average response time is 194.5ms.

C. Dynamic Membership Changes

Another concern for scalability are group membership changes. Because it must be expected that in large groups, the rate of receiver joins and leaves is high [13], their associated overhead must remain small (e.g. in terms of signalling, resource allocation, updating internal state, but also timeout computation.) We outline a solution to this problem. For this, we have to assume that after a timeout computation, all on-tree switches must maintain the following state from the last timeout computation:

1. The offset S they have received from their parent.
2. The child subtree A in which lies their representative receiver.
3. The variable s from the entry in CCS_A that was selected in A.

First, consider the case when a new receiver joins. The new receiver is attached to the existing tree at a node X. Our solution consists in assuming that this new receiver is not a representative (except for itself). This means that X can simply instruct the new receiver to use the timeout value $S + s$ to insure the single-NACK condition.

Second, consider the case when a receiver leaves. If the receiver is not a representative of any subtree, then clearly the receiver can go away without harming the single-NACK property. However, if the receiver *is* a representative of one or several ancestors, then the situation is different. Denote by C the highest ancestor for which the receiver in question is the representative receiver. Then, if we just let the receiver go away, all the cotrees of that receiver within C would produce a NACK if C were the loss subtree - the situation is equivalent to C's NACK being lost.

In other words, before we release the receiver, we need to find a new representative for C. This can be achieved by applying the distributed timeout computation locally within subtree C, while pretending that the receiver wishing to leave did not exist. Once a new set of timeouts has been computed within C, the receiver can leave.

There is no guarantee that the join and leave operations described above conserve the global optimality of the receiver allocation. However, note that their impact tends to be very locally constrained. For example, in a tree with outdegree m, the probability that a randomly picked receiver is not a representative is equal to $1 - m^{-1}$, and the probability that it is only a local representative (i.e. of its parent only) is equal to $m^{-1}(1-m^{-1})$, etc. In other words, in practice the impact of join and leave operations has a high probability of remaining very locally constrained.

When the cost of the tree moves too far away from the optimum, it becomes necessary to recompute all the timeouts. This can be done in two ways, namely on a periodic basis, or on a demand-driven basis. In the latter case, the source could distribute a threshold cost in the downstream part of the protocol. Any receiver in the tree whose timeout exceeds this threshold would spawn the recomputation.

In conclusion, we observe that dynamic membership changes can be accommodated efficiently in DTRM. The probability that a membership change can be treated locally is high, which is an important requirement for scalable group management. The tradeoffs involved in the recomputation of the timeouts will be the subject of further research.

VII. Conclusion and Future Work

Timer-based approaches to avoid feedback implosion in reliable multicast protocols have the advantage of requiring no network-level processing of feedback. We have presented a distributed algorithm to compute deterministic timeouts. This algorithm has several advantages.

First, the computation of the timeouts can be entirely distributed. No node in the tree (sender, switches, receivers) needs to maintain global information, and communication to compute timeouts is local between neighbors in the tree. The messages remain small even for very large multicast groups. Second, the single-NACK property is deterministic, i.e. we are guaranteed that only one NACK results from a single loss, provided that the NACK and the retransmitted packet are not lost themselves. If they are lost, then the number of additional NACKs remains acceptably small. Third, this algorithm computes a set of timeouts that is optimal with regard to the maximum response time experienced by the sender and the receivers. This maximizes the efficiency of the multicast service by minimizing the buffering requirements in the transport layer, and by minimizing the delays seen by the application. Fourth, both the maximum and the average response time scale as $O(\log_m(n))$. Dynamic group membership changes can be efficiently handled.

The usefulness of timer-based feedback implosion control is not limited to reliable multicast, of course. Any multicast protocol that relies on feedback from receivers about events occurring on links inside the multicast tree could take advantage of it. An important example is multicast congestion control, where feedback is used to reduce the source rate in case of congestion on a link.

We expect DTRM to prove useful also as a benchmarking baseline for heuristics for timeout computation, such as those designed to operate in networks without delay guarantees. The potential of these heuristics can be estimated from a comparison with this optimal algorithm.

Acknowledgments

I am indebted to S. Keshav for suggesting this study and many stimulating discussions. My thanks to the anonymous reviewers for many helpful comments. Discussions with M. Ammar, J-C. Bolot, C. R. Kalmanek, C. Lund,

S. Phillips and K. K. Ramakrishnan were a valuable contribution to this work. The idea of using `traceroute` to compute the tree topology given a sender and a set of receivers is due to K. C. Almeroth and M. Ammar.

References

[1] M. Grossglauser, "Optimal Deterministic Timeouts for Reliable Scalable Multicast," in *Proc. IEEE INFOCOM '96*, (San Francisco, California), March 1996.

[2] S. Pingali, D. Towsley, and J. F. Kurose, "A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols," in *Proc. ACM SIGMETRICS '94*, pp. 221–230, May 1995.

[3] J. C. Lin and S. Paul, "RMTP: A Reliable Multicast Transport Protocol," in *Proc. IEEE INFOCOM '96*, (San Francisco, Calif.), pp. 1414–1424, March 1996.

[4] H. W. Holbrook, S. K. Singhal, and D. R. Cheriton, "Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation," in *Proc. ACM SIGCOMM '95*, (Cambridge, Mass.), pp. 328–341, September 1995.

[5] J. Crowcroft and K. Paliwoda, "A Multicast Transport Protocol," in *Proc. ACM SIGCOMM '88*, pp. 247–256, August 1988.

[6] S. Floyd, V. Jacobson, S. McCanne, C.-G. Liu, and L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing," in *Proc. ACM SIGCOMM '95*, August 1995.

[7] C. Papadopoulos and G. Parulkar, "Implosion Control for Multipoint Applications," in *10th IEEE Workshop on Computer Communications*, (Seattle), September 1995.

[8] W. T. Strayer, B. J. Dempsey, and A. C. Weaver, *XTP: The Express Transfer Protocol.* Addison-Wesley (Reading, Mass.), 1992.

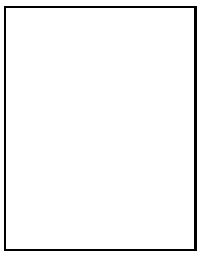
[9] J.-C. Bolot, T. Turletti, and I. Wakeman, "Scalable Feedback Control for Multicast Video Distribution in the Internet," in *Proc. ACM SIGCOMM '94*, (London, UK), September 1994.

[10] M. H. Ammar, "Probabilistic Multicast: Generalizing the Multicast Paradigm to Improve Scalability," in *Proc. IEEE INFOCOM '94*, (Toronto, Canada), June 1994.

[11] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms.* McGraw-Hill, New York, NY, 1990.

[12] M. Yajnik, J. Kurose, and D. Towsley, "Packet Loss Correlation in the MBone Multicast Network," Tech. Rep. UMASS CMPSCI Technical Report #96-32, University of Massachusetts at Amherst, Amherst, Mass., USA, 1996.

[13] T. Billhartz, J. B. Cain, E. Farrey-Goudreau, D. Fieg, and S. Batsell, "Performance and Resource Cost Comparisons for the CBT and PIM Multicast Routing Protocols in DIS Environments," in *Proc. IEEE INFOCOM '96*, (San Francisco, Calif.), pp. 85–93, March 1996.



Matthias Grossglauser received the Diploma in Communications Systems Engineering from the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, and the M. Sc. from the Georgia Institute of Technology, Atlanta, GA, both in 1994. He is currently pursuing his Ph. D. at INRIA, Sophia Antipolis, France.

He spent a year at Georgia Tech, working in the PICA group on adaptive routing strategies for massively parallel computers. In 1993, he was at Institut EURECOM, Sophia Antipolis, France, specializing in Corporate Networking. In 1994 and in summer 1995, he was a visitor at AT&T Bell Laboratories, Murray Hill, NJ, to do research on service models and scheduling disciplines for high speed networks, and on scalable multipoint communications.

Currently, his research interests are in network traffic analysis and modeling, measurement-based call admission control, and reliability and scalability in multipoint communications.