A Delay Analysis of Tree-based Reliable Multicast Protocols

Christian Maihöfer and Kurt Rothermel
University of Stuttgart, Institute of Parallel and Distributed High Performance Systems (IPVR),
Breitwiesenstr. 20-22, D-70565 Stuttgart, Germany
{christian.maihoefer|kurt.rothermel}@informatik.uni-stuttgart.de

Abstract—**We present a comparative delay analysis of tree-based reliable multicast protocols and show the influence of varying sending rates, group sizes, packet loss probabilities and branching factors of the control tree. Besides the average delivery delay we consider the delay to reliably deliver all packets and the round trip delay. The former two examines the delay between generation of a packet at the sender and correct reception at a randomly chosen receiver or all receivers, respectively. The latter is the delay between generation of a packet at the sender and reception of all acknowledgement packets at the sender.**

Our numerical results show that all tree-based protocols provide low delays and good scalability. From the four considered protocol classes, NAK-based protocols achieve the best scalability but ACK-based protocols achieve the lowest delays.

I. Introduction

In analysis and simulation studies concerning bandwidth and processing load, tree-based reliable multicast protocols have proven to provide scalability for a large number of receivers. In tree-based protocols, the members of a multicast group are organized in a so-called control tree to overcome the well-known acknowledgment implosion problem of flat approaches, i.e., overwhelming of the sender by a large number of positive (ACKs) or negative acknowledgements (NAKs). A positive acknowledgement returned by a receiver confirms correct message delivery, whereas a negative acknowledgement asks for a message retransmission. Since acknowledgements are propagated along the edges of the control tree in a leaf-to-root direction, the implosion problem can be avoided by limiting the branching factor of a node and thus the number of acknowledgment messages.

In this paper we present a delay analysis of tree-based reliable multicast protocols. The message delivery delay is an important issue for multimedia applications. For example, real time applications like interactive distributed simulations, distributed games, or the delivery of MPEG I-frames [1] benefit from guaranteed reliability and low delays. Besides time constraints of some applications, low delays are vital for providing high throughput with a window based sending scheme [2].

In contrast to previous delay analysis we assume a more realistic system model as explained later in Section II. Besides analyzing the average delay between sender and receiver we determine the threshold delay, too. Threshold delay is the delay to reliably deliver all packets with a certain probability. For applications with time bounds for the delivery of messages, threshold delay should be considered rather than average delay since it gives a more realistic impression of the delay behaviour of reliable multicast protocols. For example, for low packet loss probabilities and within the scalability range of the various protocol classes, the average delivery delays of all classes are rather similar and only moderately higher than the message propagation delay of the network, which means they are rather similar to unreliable protocols without retransmissions. In contrast to average delay, threshold delay allows to compare the protocols and the performance of their retransmission schemes in more detail. Finally, we analyze the round trip delay between sending a data packet and receiving the last corresponding control packet at the sender. The round trip delay determines the time after a data packet can be removed from memory and influences the sending rate if the sender uses a window based sending scheme [2]. Furthermore, knowledge about this delay is important to adjust the retransmission timeout at the sender.

Our numerical results show that all tree-based protocols provide good scalability and low delays compared to non-hierarchical approaches. To be more precise, NAK-based protocols achieve the best scalability but ACK-based protocols achieve the lowest delays. With respect to the branching factor, the optimal value depends on several parameters like packet loss probability, protocol class and whether average delivery delay, threshold delivery delay or round trip delay is of interest. We can conclude, though, that a tuned branching factor can significantly reduce delay. To assess the analytical results we have simulated the RMTP [3] and TMTP [4] protocol and compared the analytical results with simulation results. The simulations confirm the analytical results [13].

The remainder of this paper is structured as follows. In the next section related work is discussed. In Section III we discuss the analyzed protocol classes. In Section IV we introduce our assumed system model followed by the detailed delay analysis. Numerical results are presented in Section V. Finally, we will conclude with a brief summary.

II. Background and Related Work

The first comparative delay analysis of sender- (ACK) and receiver-initiated (NAK) approaches was presented by Yamamoto et al. [5] and DeCleene [6]. Yamamoto et al. have analyzed the expected average delivery delay and showed that receiver-initiated protocols with NAK suppression provide best scalability. However, their analytical model for this class was simplified in assuming that all receivers are perfectly synchronized and thus only one NAK is sent back to the sender in case of message loss. Another simplification they made is that control packets are reliably delivered. While the analysis in [5] is independent of the network topology, in [6] a delay analysis of generic ACK- and NAK-based protocols operating over star and linear topologies was presented. In [7] the effect of local recovery and retransmission of parity packets on bandwidth and delay of NAK-based protocols is examined. While the bandwidth analysis is made in detail, the delay analysis is rather brief and comparatively simple. For example, they do not consider queuing delay in detail and neglect feedback processing. They concluded that local recovery techniques and parity packets outperforms other approaches.

Our paper extends previous work in four significant ways. First, to our knowledge this is the first comparative analysis of generic classes of tree-based reliable multicast protocols, which considers feedback traffic and queuing delays. Second, we consider the loss of control packets rather than assuming reliable delivery. Third, we assume that local clocks are not synchronized which affects the NAK suppression scheme. Finally, besides average delivery delay we examine the threshold delay and the round trip delay, too. An analysis of non-hierarchical protocols considering the issues above can be found in [8].

III. Classification of Tree-based Multicast Protocols

A. ACK-based Protocol (H1)

The first considered scheme is denoted as (H1). As in all other protocol classes we assume that the initial sender is the root of the control tree and that the initial transmission is multicasted to the global group. Global group denotes the whole multicast group in contrast to a local group, which is described below. (H1) uses ACKs sent by receivers to their parent in the control tree, called group leader, in order to indicate correctly received packets. Each group leader that is not the root node also sends an ACK to its parent as soon as a data packet has been received. If a timeout for an ACK occurs at a group leader, a multicast retransmission is invoked for this local group. A local group encompasses a group leader and its directly attached children. Such a retransmission can be sent to a separate multicast address for this local group or sent to the global group address and limited in scope by the TTL value. An example of a protocol similar to our definition of (H1) is RMTP [3]. RMTP uses subtree multicasting to limit the retransmission scope.

B. NAK-based Protocol (H2)

The second scheme (H2) is based on NAKs with NAK suppression [9]. NAKs are sent by means of multicast to the group leader and other nodes of this local group. A receiver that misses a data packet sends a NAK provided that it has not already received a NAK from another receiver that also misses the data packet. NAKs alone do not allow a deterministic decision when packets can be removed from memory at the sender. Therefore, selective ACKs (SAKs) are sent after a certain number of packets has been received or after a certain time period has been expired, to propagate the state of a receiver to its group leader. TMTP [4] is an example for class (H2).

C. Protocol with Aggregated Acknowledgments (H3) and (H4)

Protocol (H3) and protocol (H4) are based on protocol (H1) or (H2), respectively. Additionally, they implement aggregated ACKs, so called AAKs. In contrast to normal ACKs, they are sent to confirm the correct message delivery for a whole subhierarchy of the control tree. AAKs are necessary to guarantee reliable delivery even in case of node failures. A detailed discussion of these protocol classes is given in [10].

IV. Analysis

A. System Model

We assume the following system model for our analytical evaluations. A single sender multicasts a message to a set of R identical receivers. With probability q_D the multicast message is corrupted or lost during the transmission to a single receiver. With probability p_A for ACKs and q_N for multicast NAKs, a control message is corrupted or lost. We assume that nodes do not fail and that the network is not partitioned, i.e. retransmissions are finally successful. All nodes work exclusively for the multicast protocol and no background load is considered.

B. Analytical Approach

Our goal is to determine the delays between the initial generation of a packet at the sender and the correct reception at a receiver as well as the reception of the last control packet at the sender. These delays are determined by the necessary processing times for a packet at the sender and receivers, transmission delays, timeout delays to wait for a data or control packet and finally the number of necessary transmissions for correct reception of data and control packets.

The processing time at a node is determined by the load of such a node, i.e. the processing of data and control packets. We first determine the rates for initial sending and arrival of packets. Arrival times are modeled as a poisson distribution, which results in exponentially distributed inter-arrival times. As we assume general distributed service times this queue type is defined as $M|G|1$ queue [11].

The number of necessary data packet transmissions M is determined by the packet loss probabilities q_D, p_A, and q_N. M has already been determined for the various protocol classes assuming independent losses in our processing and bandwidth requirements analysis [12], [10]. In [13] we have analyzed M for spatially dependent losses. Given the average processing times and the number of transmissions we can determine the delay experienced by a single data packet.

C. Protocol Independent Methods

If a node in a tree-based protocol has lost a data packet and a retransmission is needed, the retransmission request (either by a NAK packet or a missing ACK packet) is sent to the group leader. If this group leader has lost the data packet as well, the group leader's group leader is queried an so forth. As a prerequisite for the delay analysis we will determine the height of the control tree. We define the root node's height as 1. The height of every other node is the height of the parent node plus 1. With this definition, the height can be obtained as follows, where R is the number of receivers:

$$R = \sum_{i=0}^{h-1} B^i = \frac{1-B^h}{1-B} \rightarrow h = \log_B \Big(R(B-1)+1 \Big). \tag{1}$$

B is the number of members in a local group (i.e. the branching factor of the control tree). We assumed, that the control tree is balanced, which can be achieved with the Token Repository Service [14, 15]. To obtain the mean delay, we obtain the average tree height $\tilde{h}$:

$$\tilde{h} = \frac{\Big(\sum_{i=1}^{h-2} (i+1) * B^i\Big) + \Big(R - \sum_{j=1}^{h-2} B^j\Big) h}{R}. \tag{2}$$

D. ACK-based Protocol (H1)

For a delay analysis of tree-based protocols we distinguish among sender, receivers and group leaders. Although the sender is a group leader as well, here and in the following we will denote only inner nodes as group leaders. All delay components are shown in Figure 1.

1) Mean Waiting Times at the Sender (Root Node)

First, we have to determine the mean waiting time for a packet between generation or arrival and completion of processing or sending. The mean waiting time is determined by the load of a node, i.e. the processing of incoming and outgoing packet flows. The sender has to process the following three arriving packet flows:

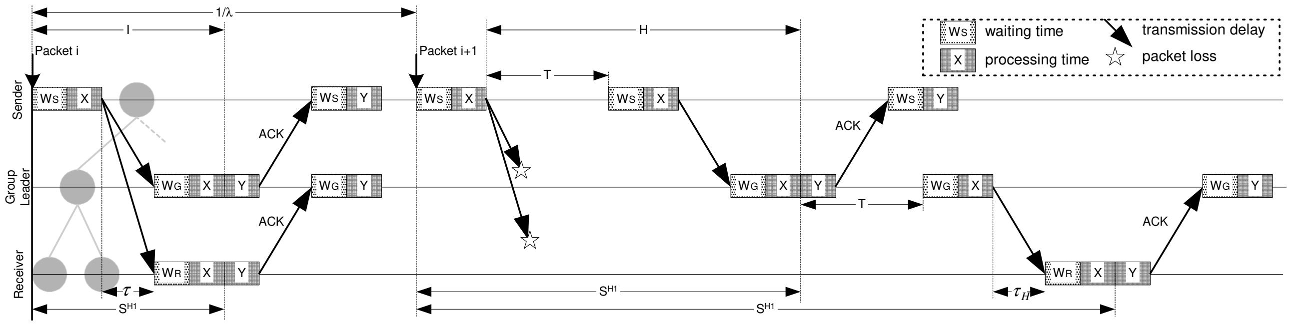

Figure 1: Packet delivery delay for protocol (H1)

1. Data packets from the higher protocol layer that are transmitted for the first time. This packet flow is referred to as λ_t^S and has rate λ. The processing time for a data packet is assumed to be X.
2. Data packets that are retransmitted due to packet loss. This packet flow is referred to as λ_r^S and has rate $\lambda(E(M^{H1})-1)$, since every packet is $(E(M^{H1})-1)$-times retransmitted. $E(M^{H1})$ is the expected total number of transmissions per packet until all multicast group members have received it correctly.
3. Control packets are received by the sender with flow λ_a^S and rate $\lambda B E(M^{H1})(1-q_D)(1-p_A)$. B is the branching factor of the ACK-tree, i.e. the number of child nodes per group leader. The processing time for an ACK packet is assumed to be Y.

The expected total number of necessary transmissions $E(M^{H1})$ to receive the data packet correctly at all receivers is given in [12]:

$$E(M^{H1}) = \sum_{i=1}^{B} \binom{B}{i} (-1)^{i+1} \frac{1}{1-(q_D+(1-q_D)p_A)^i}. \tag{3}$$

The load on the sender is given by the traffic intensity ϱ, which is generally the product of the traffic rate λ and mean processing time for a request (data transmission, retransmission or request) $E(S)$:

$$\varrho = \lambda E(S). \tag{4}$$

The load on the sender (ϱ_S^{H1}, traffic intensity) is then the sum of the packet rates:

$$\varrho_S^{H1} = \lambda E(M^{H1})E(X) + \lambda B E(M^{H1})(1-q_D)(1-p_A)E(Y). \tag{5}$$

As explained in Section B, the system can be modeled as a $M|G|1$ queue. The mean waiting time $E(W)$ for a packet until processing starts in this queuing system is [11]:

$$E(W) = \frac{\lambda E(S^2)}{2(1-\varrho)} \tag{6}$$

$$E(W_S^{H1}) = \frac{(\lambda_r^S+\lambda_t^S)E(X^2)+\lambda_a^S E(Y^2)}{2(1-\varrho_S^{H1})}. \tag{7}$$

2) Mean Waiting Times at a Receiver (Leaf Node)

The only packet flow at a receiver is the reception of data packets which are acknowledged by an ACK, λ^R, with rate $\lambda E(M^{H1})(1-q_D)$. The processing time is $X+Y$ since the arrival of a data packet is followed by replying an ACK packet to the sender. Note that X and Y are independent random variables.

The load on the receiver is:

$$\varrho_R^{H1} = \lambda E(M^{H1})(1-q_D)\Big(E(X)+E(Y)\Big). \tag{8}$$

The mean waiting time of a packet at the receiver until processing starts is (see Eq. 6):

$$E(W_R^{H1}) = \frac{\lambda^R E\Big((X+Y)^2\Big)}{2(1-\varrho_R^{H1})}. \tag{9}$$

With X and Y are independent random variables:

$$E(W_R^{H1}) = \frac{\lambda^R\Big(E(X^2)+E(Y^2)+2E(X)E(Y)\Big)}{2(1-\varrho_R^{H1})}. \tag{10}$$

3) Mean Waiting Times at a Group Leader (Inner Node)

The load on an inner node is the sender load without the initial transmission and the receiver load:

$$\varrho_G^{H1} = \lambda_r^S E(X) + \lambda_a^S E(Y) + \lambda^R E(X+Y).$$

The mean waiting time of a packet at an inner node is:

$$E(W_G^{H1}) = \frac{\lambda_r^S E(X^2)+\lambda_a^S E(Y^2)}{2(1-\varrho_G^{H1})} + \frac{\lambda^R\Big(E(X^2)+E(Y^2)+2E(X)E(Y)\Big)}{2(1-\varrho_G^{H1})}. \tag{11}$$

4) Overall Delay of Protocol (H1)

T is the group leader timeout delay, τ is the network propagation delay, $\bar{h}$ is the average number of hierarchy levels of the control tree and B is the branching factor. If no retransmission is necessary, the delay from the initial transmission $E(I)$ is:

$$E(I) = E(W_S^{H1}) + E(X) + \tau + E(W_G^{H1}) + E(X). \tag{12}$$

Note that a simplifying pessimistic assumption we made is that the receiver is always a group leader and therefore take $E(W_G^{H1})$ in the above equation.

Now we want to determine the delay for a hierarchical retransmission on condition that the parent node has received the packet correctly. The time for a hierarchical retransmission $E(H)$ is:

$$E(H) = \Big(E(M_r^{H1}|M_r^{H1}>1)-1\Big)\Big(T+E(W_G^{H1})+E(X)\Big) + \tau_H + E(W_G^{H1}) + E(X). \tag{13}$$

M_r^{H1} is the number of transmissions for a single receiver r and τ_H the network propagation delay for a hierarchical retransmission. For obtaining the overall delay, we determine the probabilities that no data loss occurs, that a node misses a packet but the parent node is able to retransmit it, that a node and its parent misses that packet and the next parent retransmits it and so forth and multiply these probabilities with the expected delays. The overall delay is then:

$$E(S_\phi^{H1}) = \Big[\sum_{i=o}^{\tilde{h}-2} q_D^i(1-q_D)\Big(E(I)+iE(H)\Big)\Big] + q_D^{\tilde{h}-1}\Big(E(W_S^{H1})+E(X)+(\tilde{h}-1)E(H)\Big). \quad (14)$$

Besides the delay for delivering data packets to the receivers we want to examine the delay for receiving all ACK packets at the sender:

$$E(S_{RTD}^{H1}) = \underbrace{\Big(E(M^{H1})-1\Big)\Big(T+E(W_S^{H1})+E(X)\Big)}_{\text{sending transmissions}} + \underbrace{E(X)+E(W_S^{H1})+\tau_H+E(W_G^{H1})+E(X)}_{\text{receiving last successful retransmission}} + \underbrace{E(Y)+E(W_G^{H1})+\tau_H+E(W_S^{H1})+E(Y)}_{\text{send and receive last ACK}}. \quad (15)$$

Besides the mean delivery delay we can determine the expected delay to reliably deliver a certain percentage of data packets. We assume that γ is the percentage of data packets that has to be reliably delivered and $E(S_\gamma^{H1})$ is the expected delay, which can be obtained as follows:

$$\gamma = 1 - q_D^{M_\gamma^{H1}} \quad (16)$$

$$M_\gamma^{H1} = \frac{ln(1-\gamma)}{ln(q_D)}, \gamma \geq q_D \quad (17)$$

$$E(H) = \Big(E(M_\gamma^{H1}|M_\gamma^{H1}>1)-1\Big)\Big(T+E(W_G^{H1})+E(X)\Big) + \tau_H + E(W_G^{H1}) + E(X) \quad (18)$$

$$E(S_\gamma^{H1}) = E(W_S^{H1}) + E(X) + (\tilde{h}-1)E(H). \quad (19)$$

In Eq. 19 retransmissions encompass all nodes on the path from the sender to a random receiver r. Our assumption here is that all parent nodes first have to receive a message with the desired probability γ before retransmissions can be sent to r.

E. NAK-based Protocol (H2)

1) Mean Waiting Times at the Sender (Root Node)

At the sender we distinguish among the following four packet flows: First, the flow for the initial data packet transmissision λ_t^S with rate λ and processing time X. Second, the NAK flow that trigger a retransmission λ_r^S with rate $\lambda\Big(E(M^{H2})-1\Big)$ and processing time $X+Y$. Third, the flow of additional NAKs, which are not necessary to trigger a retransmission λ_n^S with rate $\lambda E(\widetilde{L}^{H2}) - \lambda_r^S$ and processing time Y. And finally, the flow of selective (periodical) acknowledgments (SAKs) λ_s^S with rate $\lambda B(1-p_A)$ and processing time Z. All delay components are shown in Figure 2.

The number of transmissions is (see Eq. 3 and [12]):

$$E(M^{H2}) = \sum_{i=1}^{B}\binom{B}{i}(-1)^{i+1}\frac{1}{1-q_D{}^i}. \quad (20)$$

The number of received NAKs $E(\widetilde{L}^{H2})$ is given in the processing requirements analysis [12].

Given these flows, the load on the sender is:

$$\varrho_S^{H2} = \lambda E(M^{H2})E(X) + \lambda E(\widetilde{L}^{H2})E(Y) + \lambda B(1-p_A)E(Z). \quad (21)$$

The mean waiting time of a packet at the sender until it is processed is:

$$E(W_S^{H2}) = \frac{\lambda_t^S E(X^2) + \lambda_r^S\Big(E(X^2)+E(Y^2)+2E(X)E(Y)\Big)}{2(1-\varrho_S^{H2})} + \frac{\lambda_s^S E(Z^2)}{2(1-\varrho_S^{H2})}. \quad (22)$$

2) Mean Waiting Times at a Receiver (Leaf Node)

At the receiver we distinguish among four packet flows. λ^R is the flow of data packets from the sender with rate $\lambda E(M^{H2})(1-q_D)$ and processing time X. The second flow $\lambda_{n,g}^R$ consists of the submitted NAK packets with rate $\lambda\Big(E(O_r)-1\Big)\frac{\vartheta_2}{\vartheta_3}$ and processing time Y. ϑ_2 is the average number of NAKs sent in each round and ϑ_3 is the mean number of receivers that did not receive a data packet and therefore want to send a NAK. The third flow $\lambda_{n,r}^R$ are the received NAKs from other receivers with rate $\lambda(1-q_N)\Big[\vartheta_2\Big(E(O)-1\Big) - \frac{\vartheta_2}{\vartheta_3}\Big(E(O_r)-1\Big)\Big]$ and processing time Y. The last flow λ_s^R are the sent SAKs with rate λB and processing time Z.

The total number of rounds O, the number of rounds for a single receiver O_r, as well as ϑ_2 and ϑ_3 are given in the processing requirements analysis [12].

With these flows, the load on the receiver is:

$$\varrho_R^{H2} = \lambda E(M^{H2})(1-q_D)E(X) + (\lambda_{n,g}^R + \lambda_{n,r}^R)E(Y) + \lambda_s^R E(Z). \quad (23)$$

Therefore the mean waiting time of a packet at a receiver is:

$$E(W_R^{H2}) = \frac{\lambda^R E(X^2) + (\lambda_{n,g}^R + \lambda_{n,r}^R)E(Y^2) + \lambda_s^R E(Z^2)}{2(1-\varrho_R^{H2})}. \quad (24)$$

3) Mean Waiting Times at a Group Leader (Inner Node)

The load on an inner node is the sender load without the initial transmission and the receiver load:

$$\varrho_G^{H2} = \Big(E(M^{H2})-1\Big)E(X) + \lambda\widetilde{L}^{H2}E(Y) + \lambda B(1-p_A)E(Z) + \lambda E(M^{H2})(1-q_D)E(X) + (\lambda_{n,g}^R + \lambda_{n,r}^R)E(Y) + \lambda_s^R E(Z). \quad (25)$$

The mean waiting time of a packet at an inner node is:

$$E(W_G^{H2}) = \frac{\lambda_r^S\Big(E(X^2)+E(Y^2)+2E(X)E(Y)\Big)}{2(1-\varrho_G^{H2})} + \frac{\lambda_n^S E(Y^2) + \lambda_s^S E(Z^2)}{2(1-\varrho_G^{H2})} + \frac{\lambda^R E(X^2) + (\lambda_{n,g}^R + \lambda_{n,r}^R)E(Y^2) + \lambda_s^R E(Z^2)}{2(1-\varrho_G^{H2})}. \quad (26)$$

4) Loss Detection Phase

To obtain the overall delay, we distinguish between the following phases (taken from [5]):

1. Loss Detection Phase. This phase encompasses the time between the initial arrival of a packet i at the sender and the triggering of a NAK at one of the receivers, which have lost the first data packet. The loss is detected with the arrival of a packet j, where $j > i$.

2. Loss Recovery Phase. This phase encompasses the time between the end of the first phase and the correct reception of a packet at the considered receiver. As NAKs can be lost, this phase includes the periodical sending of NAKs until the data packet is received correctly.

For the loss detection phase we must consider the time to unsuccessfully send data packets, the time to send and receive the first successful data packet and the time to send an initial NAK for the first lost data packet. The random variable L is the number of consecutive lost packets at the $k+1$ unsuccessful receivers. Given that $K=k$, the conditional probability distribution of L is:

$$P(L=l|K=k) = q_D^{l(k+1)}(1-q_D^{k+1}), \quad l = 0,1,\ldots \text{ and } k = 0,1,\ldots,R-1. \quad (27)$$

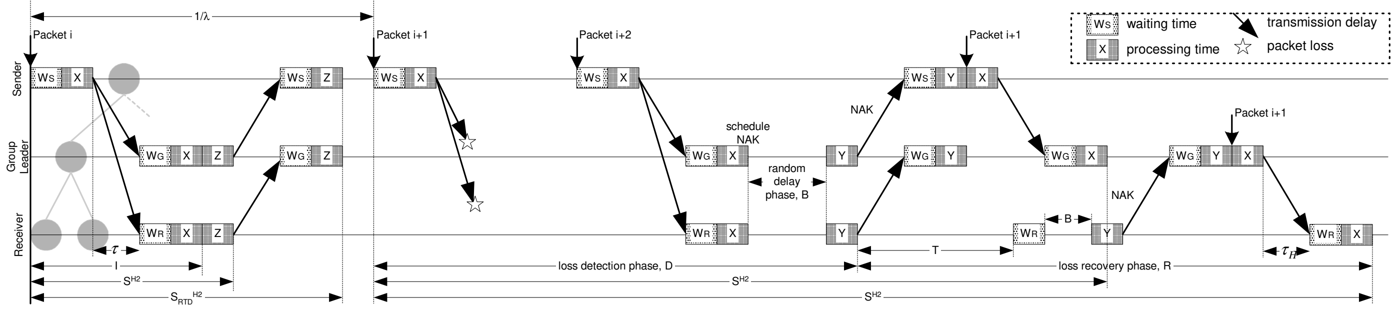

Figure 2: Packet delivery delay for protocol (H2)

The number of subsequent lost packets at $k+1$ receiver is:

$$E(L|K=k) = \frac{q_D^{k+1}}{1-q_D^{k+1}}. \tag{28}$$

To obtain the mean k among the possible ones between 0 and $B-1$ we have:

$$E(L|K) = \sum_{k=0}^{B-1} \binom{B-1}{k} q_D^k (1-q_D)^{B-1-k} \frac{q_D^{k+1}}{1-q_D^{k+1}}. \tag{29}$$

Note that Eq. 29 differs from the result of Yamamoto et al. [5]. Now we multiply the mean number of subsequent lost packets with the time $\frac{1}{\lambda}$, to process a packet. The delay of the $L+1$st packet is:

$$\underbrace{E(W_S^{H2}) + E(X)}_{\text{sender processing delay}} + \underbrace{\tau}_{\text{propagation delay}} + \underbrace{E(W_G^{H2}) + E(X)}_{\text{receiver processing delay}}. \tag{30}$$

Finally, the first phase can be expressed as follows:

$$E(D^{H2}) = \sum_{k=0}^{B-1} \binom{B-1}{k} q_D^k (1-q_D)^{B-1-k} \frac{q_D^{k+1}}{1-q_D^{k+1}} \frac{1}{\lambda} \tag{31}$$

$$+ E(W_S^{H2}) + E(X) + \tau + E(W_G^{H2}) + E(X) + E(B^{H2}) + E(Y). \tag{32}$$

Note that here and in the following we make a pessimistic simplification in assuming that the receiver is always a group leader and therefore take $E(W_G^{H2})$.

5) Loss Recovery Phase

From the viewpoint of a random receiver, this phase encompasses a number of timeout rounds. This means, the initial sent NAK in $E(D^{H2})$ was unsuccessful. The following receiver or group leader timeouts have the length $T + E(W_G^{H2}) + E(B^{H2}) + E(Y)$, where T is the timeout period and W_G^{H2} the waiting time before processing starts at a group leader. B^{H2} is the random delay a receiver or group leader waits before a NAK is sent. This delay starts with the discovery of packet loss at the first receiver. In case of no NAK suppression it ends with the expiration of the backoff timer and the transmission of the initial NAK. After a number of unsuccessful sent NAKs, this round ends with a final successful sent NAK to the sender. This includes the propagation delay to the sender, the sending of the data packet, the propagation delay to the receiver and receiver processing of the received data packet. The mean loss recovery delay is:

$$\begin{aligned} E(R^{H2}) = &\Big(E(M_r^{H2}) + E(O_{e,r}) - 1\Big) \\ &* \Big(T + E(W_G^{H2}) + E(B^{H2}) + E(Y)\Big) \\ &+ \tau_H + E(W_G^{H2}) + E(Y) \\ &+ E(X) + \tau_H + E(W_G^{H2}) + E(X). \end{aligned} \tag{33}$$

The number of necessary transmissions for a single receiver r, $E(M_r^{H2})$, as well as the number of empty rounds, in which no retransmission is sent due to NAK loss, $E(O_{e,r})$, are given in [10].

6) Overall Delay of Protocol (H2)

If no retransmission is necessary, the delay from the initial transmission $E(I)$ is:

$$E(I) = E(W_S^{H2}) + E(X) + \tau + E(W_G^{H2}) + E(X). \tag{34}$$

For obtaining the overall delay, we determine the probabilities that no data loss occurs, that a node misses a packet but the parent node is able to retransmit it, that a node and its parent misses that packet and the next parent retransmits it and so forth and multiply these probabilities with the expected delays. The overall delay is then:

$$\begin{aligned} E(S_\phi^{H2}) = &(1-q_D)E(I) \\ &+ \Big[\sum_{i=1}^{\tilde{h}-2} q_D^i (1-q_D)\Big(E(I) + E(D^{H2}) + iE(R^{H2})\Big)\Big] \\ &+ q_D^{\tilde{h}-1}\Big(E(D^{H2}) + (\tilde{h}-1)E(R^{H2})\Big). \end{aligned} \tag{35}$$

Now we obtain the delay of a SAK. Although SAKs are sent periodically, we want to determine here the delay starting from the sending of a data packet and the reception of the last SAK packet belonging to this data packet. As there is no retransmission mechanism for lost SAKs in (H2) we assume that they are reliably delivered.

$$\begin{aligned} E(S_{RTD}^{H2}) = &(1-q_D)^B E(I) \\ &\underbrace{+\Big(1-(1-q_D)^B\Big)\Big(E(D^{H2}) + E(R^{H2})\Big)}_{\text{sending transmissions}} \\ &+ \underbrace{E(Y) + E(W_G^{H2}) + \tau_H + E(W_S^{H2}) + E(Y)}_{\text{sending and receiving the last SAK}} \end{aligned} \tag{36}$$

We have to use the processing delay of control packets $E(Y)$ instead of periodic control packets since we obtain the delay on condition that a SAK is actually being sent.

The delay to reliably deliver a certain percentage γ of data packets is denoted by $E(S_\gamma^{H2})$. It can be obtained with a modified R^{H2} as follows:

$$\begin{aligned} E(R^{H2}) = &\Big(E(M_\gamma^{H2}) + E(O_{e,r}) - 1\Big) \\ &* \Big(T + E(W_G^{H2}) + E(B^{H2}) + E(Y)\Big) \\ &+ \tau_H + E(W_G^{H2}) + E(Y) \\ &+ E(X) + \tau_H + E(W_G^{H2}) + E(X) \end{aligned} \tag{37}$$

$$E(S_\gamma^{H2}) = E(W_S^{H2}) + E(X) + E(D^{H2}) + (\tilde{h}-1)E(R^{H2}). \tag{38}$$

M_γ^{H2} is obtained analogous to Eq. 17 of protocol (H1).

The analysis of protocol classes (H3) and (H4) is similar to the analysis of classes (H1) and (H2). The detailed formulas can be found in [13].

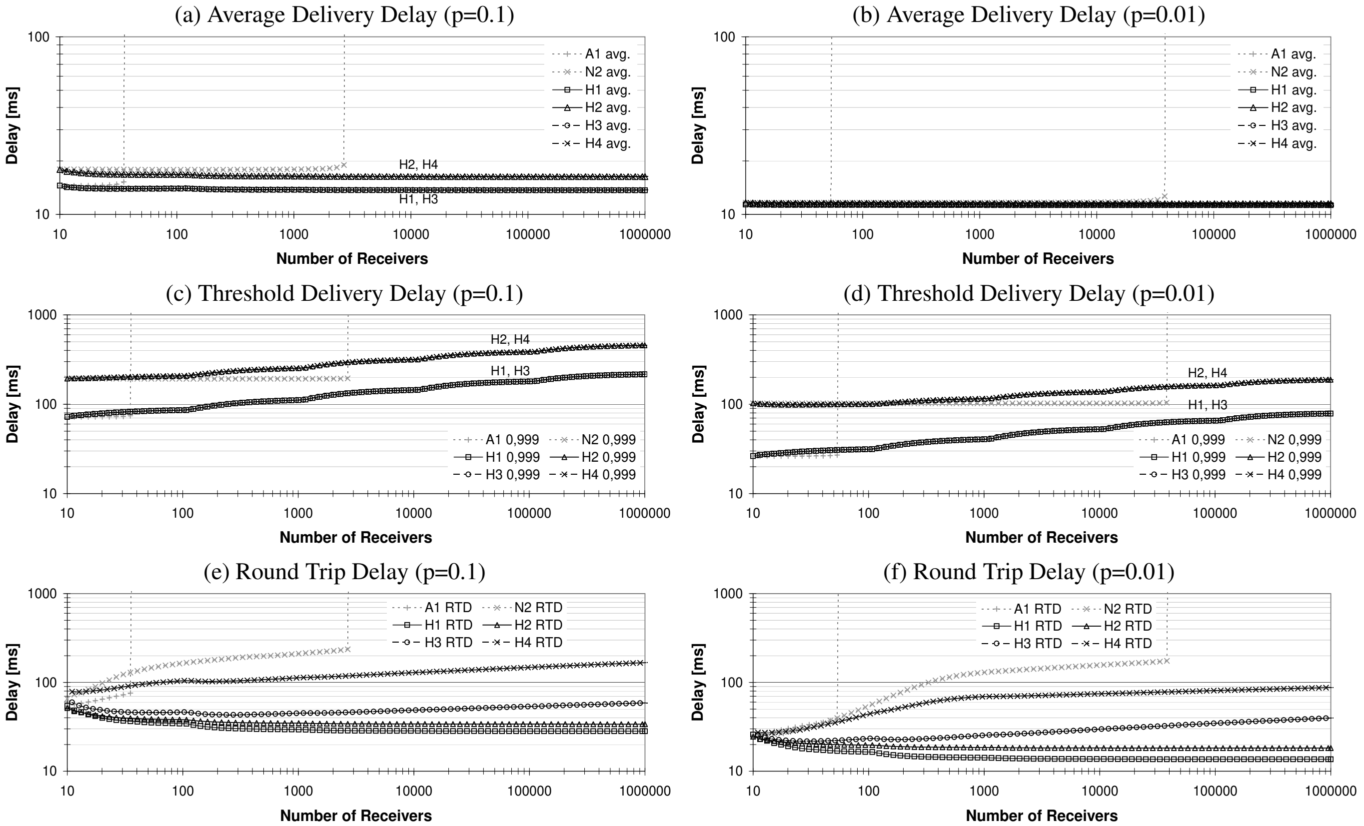

Figure 3: Delays with respect to the number of receivers

V. Numerical Results

We examine the expected delays of the analyzed protocols by means of some numerical examples. According to measurements in [16] we have chosen the delay $X = 500\mu s$ for data packets and $Y = 100\mu s$ for control packets. Analogous to [5], the packet processing times are assumed as constant with no variability, i.e. according to $Var(X) = E(X^2) - (E(X))^2 = 0$, the second moments are determined as $E(X^2) = (E(X))^2$. The propagation delay is chosen as $\tau = 10ms$. The timeouts are chosen as the trebled propagation delay, i.e. $T_S = T_R = 30ms$. For the NAK suppression time we have assumed $B = 30ms$. A discussion of reasonable values for the NAK suppression time can be found in [5]. We have chosen a lower suppression time due to our smaller local group sizes.

Figure 3 shows the expected average delay, the threshold delay to reliably deliver all packets with probability 0.999 and the round trip delay for all considered protocol classes with varying number of receivers. Additionally, two non-hierarchical approaches (A1) and (N2) are included to compare their scalability. Protocol (A1) is an ACK-based protocol similar to (H1) and (N2) is a NAK-based protocol with NAK suppression similar to (H2). Protocol (A1) and (N2) are explained in more detail in [10]. Data and control packet loss probability is 0.1 (left side) and 0.01 (right side). The data rate is $\lambda = 0.1\frac{1}{ms}$. Figure 4 plots the delays for 5000 receivers with varying sending rate λ.

In contrast to non-hierarchical approaches, all tree-based protocols provide scalability for all group sizes and in most cases, they result in lower delays even within the scalability range of (A1) and (N2). While the low packet loss probability of Figure 3.b results in almost identical average delivery delays close to the propagation delay of the network, Figure 3.a allows a more detailed view. The NAK protocols result in higher average delivery delays as well as threshold and round trip delays. This results from the receiver-initiated loss detection. Recall that receiver-initiated protocols detect packet loss by a gap in the sequence number, i.e. not before a subsequent packet is correctly received, which results in higher delays for retransmissions.

Figure 3.c and 3.d plots the threshold delivery delay to deliver all messages with probability 0.999. While the average delivery delay of all protocol classes within their scalability range and with low loss probability is close to the propagation delay of $\tau = 10ms$, since most nodes need no retransmissions, the threshold delivery delay is significantly higher. For applications having a time constraint to deliver all messages, threshold delay may be more important than average delivery delay. Analogous to average delivery delay, ACK-based protocols have a significantly lower threshold delay. With probability 1 for reliably deliver all packets, the threshold delay of our analysis would be infinite for all protocol classes, since there exists a low but non-zero probability that an infinite number of retransmissions is necessary. Our simulation results in [13] indicate that the threshold delay for $\gamma = 0.999$ is a good approximation for the delay to deliver all packets correctly.

Figure 3.e and 3.f shows the expected average round trip delay of the analyzed protocol classes. After this time, the sender can remove the data packet from memory. Besides freeing buffer space, the round trip delay is important if a window based sending scheme for flow and congestion control is used. In this case the round trip delay may limit the throughput, since throughput is basically given by $\frac{bufferspace}{rtd}$ [2]. Recall that the round trip

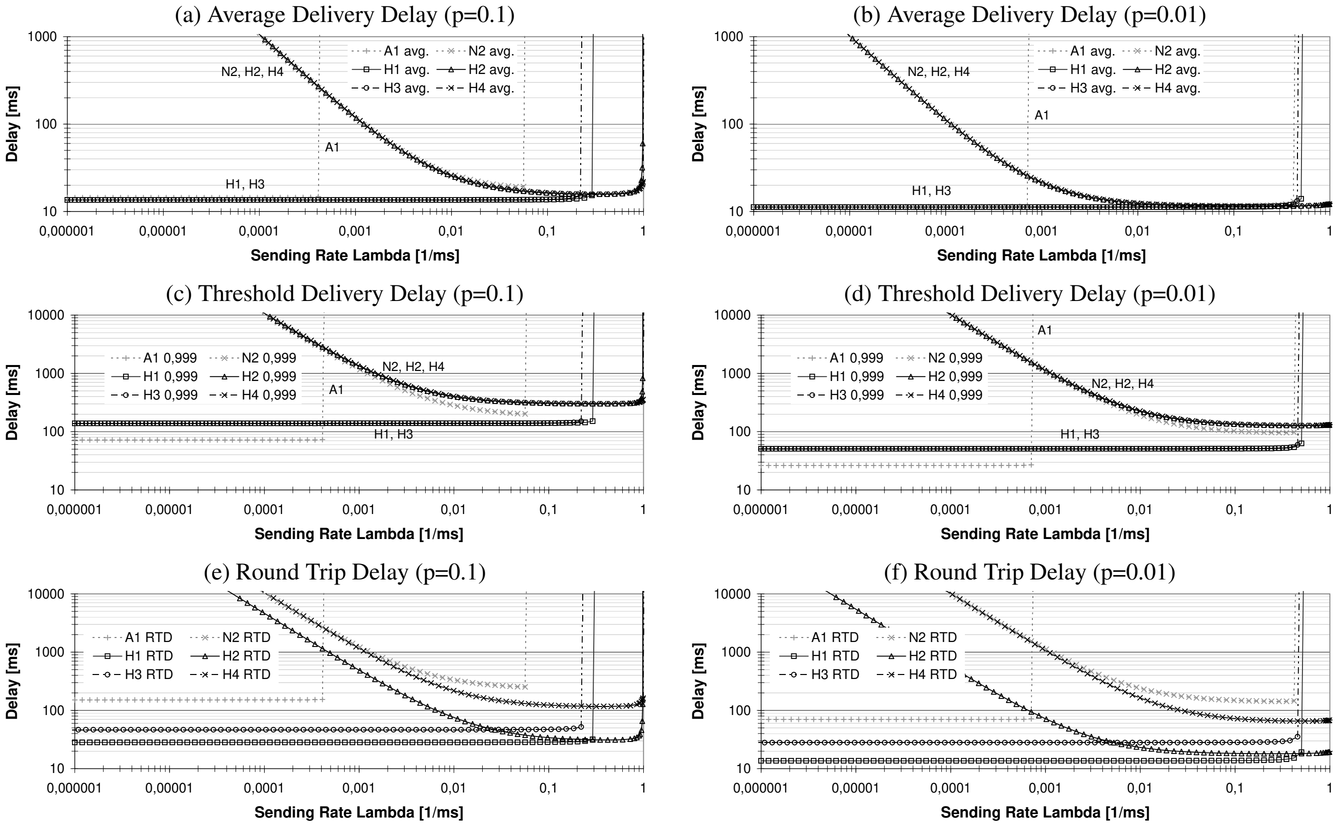

Figure 4: Delays with respect to the sending rate

delay for protocol (H1) and (H2) is assumed to encompass only the direct child nodes of the sender while the round trip delay for protocol (H3) and (H4) encompass the whole control tree. The round trip delay for (H1) and (H2) decreases with the tree height. This is caused by our assumption that an increased tree height results in lower delays between the sender and its direct child nodes, since all members of a local group are nearby. As the round trip delay with AAK protocols considers all local groups from a leaf node to the sender, for (H3) and (H4) it increases with the tree height.

The effect of the receiver-initiated loss detection can be studied in detail in Figure 4 with varying sending rates. For low sending rates this loss detection delay is the dominant delay. For example, with sending rate $\lambda = 0.0001\frac{1}{ms}$ it takes about 10s ($1/\lambda$) to detect packet loss. You can see in Figure 4.c and 4.d that the threshold delay is indeed about 10s, almost independent of the loss rate. With respect to the average delivery delay, only nodes that have lost a packet are affected by the loss detection delay. Therefore, for loss probability 0.1 only 10% of all nodes need to wait 10s for detecting a packet loss; all other nodes receive the packet from the initial transmission. Therefore, the average delay is about 1s. With packet loss probability 0.01, the average delay is decreased by about the factor 100. As packet loss is detected by the sender for protocol (H1) and (H3), their delays are independent of the sending rate. If the sending rate exceed a certain limit ($\lambda = 0.5\frac{1}{ms}$ for packet loss rate 0.01 and $\lambda = 0.2\frac{1}{ms}$ for packet loss rate 0.1), though, the sender is overwhelmed with ACK messages. For protocol (H2) and (H4) this limit is increased due to less control messages ($\lambda = 1\frac{1}{ms}$ for packet loss rate 0.1).

Figure 5 shows the delay results for varying branching factors with 1000 receivers and sending rate $\lambda = 0.1\frac{1}{ms}$. Within the scalability range of a protocol class, the average delivery delay is hardly influenced by the branching factor. Since an ACK-based protocol is only scalable for up to 50 nodes with the given sending rate, a branching factor of more than 50 nodes cannot be supported by (H1) and (H3). The threshold delay and round trip delay provides more interesting results. For all protocol classes, the threshold delay decreases with increasing branching factor until a protocol class is saturated by the feedback implosion. This is caused by the decreased tree height and therefore faster retransmissions in the worst case. With respect to round trip delay, there is a minimum delay at a branching factor of 10 to 20 for protocol (H3) and (H4). Since (H1) and (H2) uses normal ACKs rather than hierarchical ones, the lowest round trip delay is achieved with a small number of child nodes.

In [13] we have simulated the RMTP and TMTP reliable multicast protocols to compare the simulation results with our numerical results. The simulations show, that our analysis is able to predict very precisely the behaviour of the various protocol classes.

VI. Summary

We have presented a comparative delay analysis of tree-based reliable multicast protocols. Besides the average delivery delay we have considered the delay to reliably deliver all packets and the round trip delay.

Our numerical results showed that all tree-based protocols provide low delays and good scalability compared to non-hierarchical approaches. From the four considered protocol classes, NAK-based protocols achieve the best scalability but

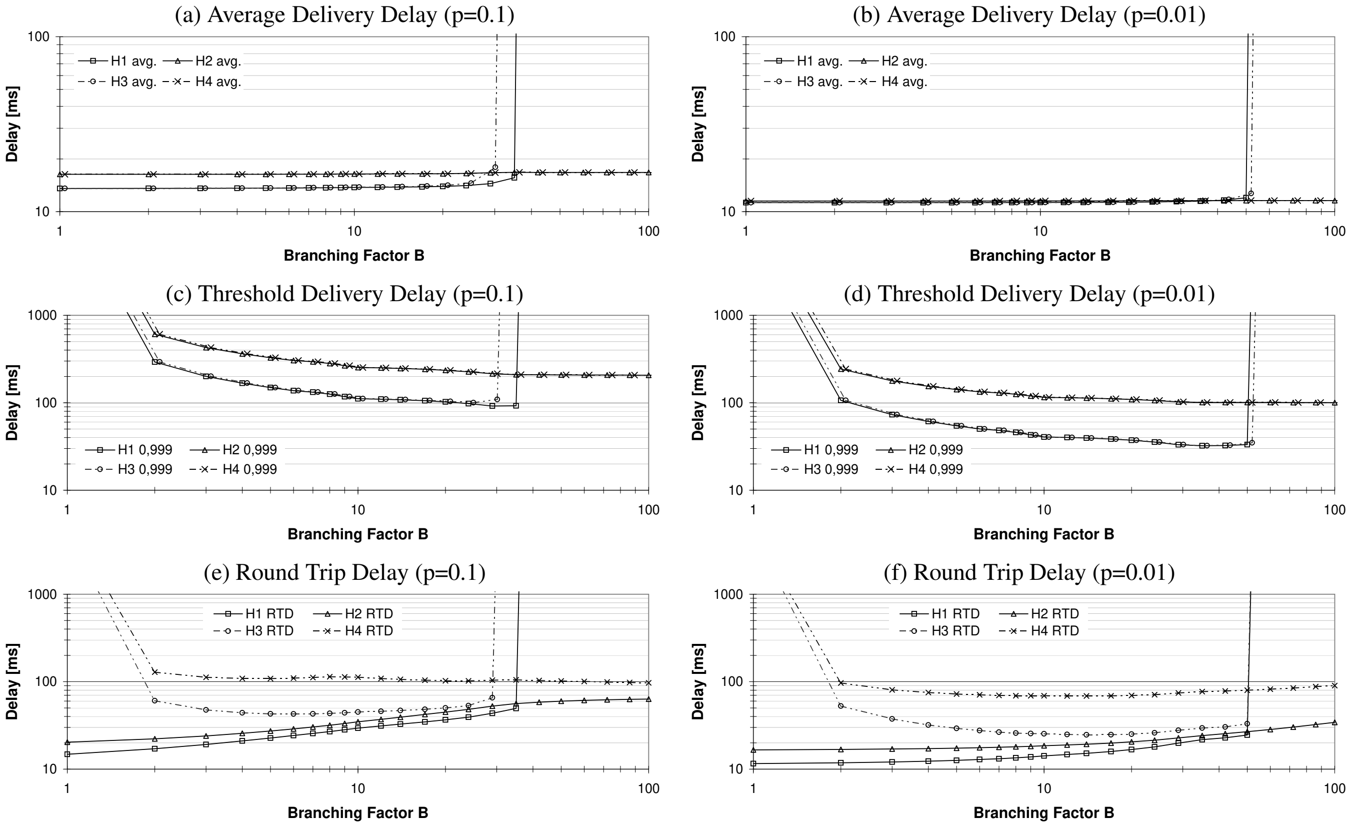

Figure 5: Delays with respect to the branching factor

ACK-based protocols achieve the lowest delays. With respect to protocols with aggregated ACKs (AAKs), which provide reliability even in case of node failures, we can conclude that the increase in delay compared to protocols without AAKs is negligible.

VII. References

[1] S. Pejhan, M. Schwartz, and D. Anastassiou, "Error control using retransmission schemes in multicast transport protocols for real-time media," *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, pp. 413–427, 1996.

[2] M. Allman, D. Glover, and L. Sanchez, "Enhancing TCP over satellite channels using standard mechanisms," RFC 2488, 1999.

[3] S. Paul, K. Sabnani, J. Lin, and S. Bhattacharyya, "Reliable multicast transport protocol (RMTP)," *IEEE Journal on Selected Areas in Communications, special issue on Network Support for Multipoint Communication*, vol. 15, no. 3, pp. 407–421, Apr. 1997.

[4] R. Yavatkar, J. Griffioen, and M. Sudan, "A reliable dissemination protocol for interactive collaborative applications," in *The Third ACM International Multimedia Conference and Exhibition (MULTIMEDIA '95)*, New York, Nov. 1996, pp. 333–344, ACM Press.

[5] M. Yamamoto, JF. Kurose, DF. Towsley, and H. Ikeda, "A delay analysis of sender-initiated and receiver-initiated reliable multicast protocols," in *Proceedings of IEEE INFOCOM'97*, Los Alamitos, Apr. 1997, pp. 480–488, IEEE.

[6] B. DeCleene, "Delay characteristics of generic reliable multicast protocols," Technical Report TR-08150-3, Logicon TASC, 1996.

[7] J. Nonnenmacher, M. Lacher, M. Jung, G. Carl, and E. Biersack, "How bad is reliable multicast without local recovery," in *Proceedings of IEEE INFOCOM'98*, New York, Apr. 1998, pp. 972–979, IEEE.

[8] C. Maihöfer and K. Rothermel, "A delay analysis of reliable multicast protocols," in *Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2001)*, Orlando, July 2001.

[9] S. Ramakrishnan and B. N. Jain, "A negative acknowledgement with periodic polling protocol for multicast over lans," in *Proceedings of IEEE INFOCOM*, Mar. 1987, pp. 502–511.

[10] C. Maihöfer, "A bandwidth analysis of reliable multicast transport protocols," in *Proceedings of the Second International Workshop on Networked Group Communication (NGC 2000)*, Palo Alto, Nov. 2000, pp. 15–26, ACM.

[11] Leonard Kleinrock, *Queueing Systems, Volume II: Computer Applications*, Wiley Interscience, New York, 1976.

[12] C. Maihöfer, K. Rothermel, and N. Mantei, "A throughput analysis of reliable multicast transport protocols," in *Proceedings of the Ninth International Conference on Computer Communications and Networks*, Las Vegas, Oct. 2000, pp. 250–257, IEEE.

[13] C. Maihöfer and K. Rothermel, "A delay analysis of tree-based reliable multicast protocols," Technical Report TR 2001/03, University of Stuttgart, 2001.

[14] K. Rothermel and C. Maihöfer, "A robust and efficient mechanism for constructing multicast acknowledgment trees," in *Proceedings of the Eight International Conference on Computer Communications and Networks*, Boston, Oct. 1999, pp. 139–145, IEEE.

[15] C. Maihöfer and K. Rothermel, "Constructing height-balanced multicast acknowledgment trees with the Token Repository Service," Technical Report TR 1999/15, University of Stuttgart, 1999.

[16] S. Kasera, J. Kurose, and D. Towsley, "Scalable reliable multicast using multiple multicast groups," in *Proceedings of ACM SIGMETRICS*, Seattle, jun 1997, pp. 64–74, ACM.