A DELAY ANALYSIS OF GENERIC MULTICAST TRANSPORT 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
{maihoefer|rothermel}@informatik.uni-stuttgart.de

ABSTRACT

We present a delay analysis of three generic classes of multicast transport protocols. The first class considered is an unreliable scheme that works without retransmission of messages. This class also includes forward error correction approaches. The second class uses a positive acknowledgment and retransmission scheme to guarantee reliability. Finally, the third class is a hierarchical approach to avoid the well-known ACK implosion problem for large receiver groups.

Our results show that only the unreliable and the hierarchical protocol class provide scalability for large receiver groups. For delay sensitive applications we can conclude from the results that in case of low packet loss probabilities, reliable multicast protocols provide low average delays, which are only slightly increased compared to unreliable protocols. However, if we take the maximum delay, the delay is significantly increased.

1. INTRODUCTION

If a reliable multicast protocol is used for groupware or multimedia communication, the expected transmission delay is one of the most important issues. While most approaches in this area use forward error correction (FEC) to avoid retransmissions for lost messages we will focus on protocols using retransmissions since FEC alone cannot guarantee reliability. For some real time applications like interactive distributed simulations, distributed games, or the delivery of MPEG I-frames, guaranteed reliability can be quite useful or even necessary [1].

In this paper we present a delay analysis of three generic classes of reliable multicast protocols. In contrast to previous delay analysis [2, 3, 4], we assume a more realistic system model including loss of data *and* control packets and this work is the first that analyzes a generic tree-based reliable multicast protocol with positive acknowledgments. Furthermore, besides the average delay for a random receiver we also analyze the maximum delay to reliably deliver all data packets with a certain adjustable probability, called threshold delay.

The first reliable protocol class we consider is denoted as (A). Receivers return a positive acknowledgment (ACK) to the sender in order to indicate successful reception of the data packet. If the sender misses an ACK packet from one or more receivers, a multicast retransmission is sent to the whole group. As the number of acknowledgment messages overwhelms the sender for larger multicast groups, hierarchical schemes have been proposed in the past. They organize the receivers into a tree structure called ACK tree, which is responsible for collecting acknowledgments and sending retransmissions. The sender is the root of the tree. Our second considered protocol class (H) is such a tree-based reliable multicast protocol. Receivers sent ACKs to their parent node in the ACK tree to indicate correctly received packets. The parent nodes are called group leaders for their children which form a local group. Each group leader that is not the root node also sends an ACK to its parent group leader until the root node is reached. If a timeout for an ACK occurs at a group leader or the root, a multicast retransmission is invoked. Note that we have not considered schemes based on negative acknowledgments, since pure NACK schemes provides no guaranteed reliability.

To show the influence of the retransmission scheme on delivery delay we compare protocols (A) and (H) with a protocol class without retransmissions (U) which can be either an unreliable best effort protocol or a FEC based protocol. The results show that (U) and (H) provide scalability for large receiver groups and transmission rates whereas (A) provides scalabiliy only for a limited group size. By comparing the absolute delays we can conclude that the average delay is increased only moderately by reliable protocols. However, the threshold delay to reliably deliver all packets with high probability, say for example 99.9%, is significantly higher compared to unreliable protocols.

The remainder of this paper is structured as follows. In the next section we introduce our assumed system model followed by the detailed delay analysis. Numerical results are presented in Section 3. Finally, we will conclude with a brief summary.

2. DELAY ANALYSIS

2.1. 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 packet is corrupted or lost during the transmission to a single receiver. With probability p_A an ACK packet 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 protocols and no background load is considered.

2.2. Analytical Approach

Our analysis builds on the work in [2]. The goal is to determine the delays between the initial generation of a packet at the sender and the correct reception at a randomly chosen receiver. The 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 retransmissions for correct reception of data and control packets.

The processing time is determined by the load on the 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 poison 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 [5].

The number of necessary data packet transmissions, M, is determined by the packet loss probabilities q_D and p_A. M has already been determined for the considered protocol classes in our processing and bandwidth requirements analysis [6, 7]. Given the average processing times and the number of transmissions we can determine the delay experienced by a single data packet.

2.3. Unreliable Protocol (U)

First, we have to determine the mean waiting time for a packet between generation and start of processing.

2.3.1. Mean Waiting Times at the Sender

The sender has to process only one packet flow, the flow of data packets denoted as λ_t^S with sending rate λ. The processing time for a packet is X. By varying the processing time X we can derive results for a best effort (low processing time) as well as FEC approach (increased processing time). The load on the sender is given by the traffic intensity ϱ_S^U, which is the product of the traffic rate λ and mean processing time for a request [5]:

$$\varrho_S^U = \lambda E(X). \tag{1}$$

As explained in Section 2.2 the system can be modeled as a $M|G|1$ queue. The mean waiting time $E(W)$ for a packet until processing starts is given in [5]:

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

where S is the processing time for a request. Given this formula, the waiting time for protocol (U) at the sender is:

$$E(W_S^U) = \frac{\lambda_t^S E(X^2)}{2(1-\varrho_S^U)}. \tag{3}$$

2.3.2. Mean Waiting Times at a Receiver

The only packet flow is the reception of data packets λ^R with rate $\lambda(1-q_D)$ due to packet loss. The load on the receiver is:

$$\varrho_R^U = \lambda(1-q_D)E(X). \tag{4}$$

The mean waiting time of a packet at the receiver until processing starts is (see Eq. 2):

$$E(W_R^U) = \frac{\lambda^R E(X^2)}{2(1-\varrho_R^U)}. \tag{5}$$

2.3.3. Overall Delay of Protocol (U)

We assume τ to be the propagation delay for a packet transmission. The average time between the initial arrival of a data packet at the sender and the correct reception at a receiver $E(S_\phi^U)$ is:

$$E(S_\phi^U) = E(W_S^U) + E(X) + \tau + E(W_R^U) + E(X). \tag{6}$$

2.4. ACK-based Protocol (A)

2.4.1. Mean Waiting Times at the Sender

The mean waiting time for a packet at the sender is determined by the following three packet flows:

1. Data packets that are transmitted for the first time. This packet flow is denoted as λ_t^S and has rate λ. The processing time for a data packet is X.
2. Data packets that are retransmitted due to packet loss. This packet flow is denoted as $\lambda_{timeout}^S$ and has rate $\lambda(E(M^A)-1)$. $E(M^A)-1$ is the number of necessary retransmissions.
3. Control packets are received by the sender with flow λ_a^S and rate $\lambda RE(M^A)(1-q_D)(1-p_A)$. R is the number of receivers. The processing time for an ACK packet is Y.

The expected total number of necessary transmissions $E(M^A)$ to receive the data packet correctly at all receivers is given in [6]:

$$E(M^A) = \sum_{i=1}^{R} \binom{R}{i} (-1)^{i+1} \frac{1}{1-(q_D+(1-q_D)p_A)^i}. \tag{7}$$

The load on the sender is the sum of the packet rates:

$$\varrho_S^A = \lambda E(M^A)E(X) + \lambda RE(M^A)(1-q_D)(1-p_A)E(Y). \tag{8}$$

The mean waiting time for a packet until processing starts is:

$$E(W_S^A) = \frac{(\lambda_t^S + \lambda_{timeout}^S)E(X^2) + \lambda_a^S E(Y^2)}{2(1-\varrho_S^A)}. \tag{9}$$

2.4.2. Mean Waiting Times at a Receiver

The only packet flow at the receiver is the reception of data packets, λ^R, with rate $\lambda E(M^A)(1-q_D)$. 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 assumed to be independent random variables. The load on the receiver is:

$$\varrho_R^A = \lambda E(M^A)(1-q_D)\Big(E(X)+E(Y)\Big). \tag{10}$$

The mean waiting time of a packet at the receiver until processing starts is (see Eq. 2):

$$E(W_R^A) = \frac{\lambda^R E\Big((X+Y)^2\Big)}{2(1-\varrho_R^A)}. \tag{11}$$

With $E(V+W) = E(V)+E(W)$, $Var(V) = E(V^2)-(E(V))^2$ and $Var(V+W) = Var(V)+Var(W)$:

$$E(W_R^A) = \frac{\lambda^R\Big(E(X^2)+E(Y^2)+2E(X)E(Y)\Big)}{2(1-\varrho_R^A)}. \tag{12}$$

2.4.3. Overall Delay of Protocol (A)

T_s is assumed to be the sender timeout delay, τ the propagation delay and M_r^A the number of necessary transmissions for a single receiver r. The probability that a receiver needs j transmissions until correct reception of a packet is $P(M_r^A = j) = q_D^{j-1}(1-q_D)$. The average delivery delay is then:

$$\begin{aligned} E(S_\phi^A) &= \Big[\sum_{j=1}^{\infty} q_D^{j-1}(1-q_D)\Big(j(E(W_S^A)+E(X))+(j-1)T_s\Big)\Big] \\ &\quad + \tau + E(W_R^A) + E(X) \qquad (13) \\ &= \frac{E(W_S^A)+E(X)+q_D T_s}{1-q_D} + \tau + E(W_R^A) + E(X). \qquad (14) \end{aligned}$$

With $E(M_r^A) = \frac{1}{1-q_D}$ and $E(M_r^A)-1 = \frac{q_D}{1-q_D}$ [6]:

$$\begin{aligned} E(S_\phi^A) &= E(M_r^A)\Big(E(W_S^A)+E(X)\Big) + \Big(E(M_r^A)-1\Big)T_s \\ &\quad + \tau + E(W_R^A) + E(X). \end{aligned} \tag{15}$$

Besides the average delivery delay we can determine the expected maximum delay to reliably deliver all data packets with a certain probability, denoted as γ. We will call this delay threshold delay $E(S_\gamma^A)$, which can be obtained as follows:

$$E(M) = \frac{ln(1-\gamma)}{ln(q_D)}, \gamma \geq 1-q_D \tag{16}$$

$$\begin{aligned} E(S_\gamma^A) &= E(M)\Big(E(W_S^A)+E(X)\Big) + \Big(E(M)-1\Big)T_s \\ &\quad + \tau + E(W_R^A) + E(X). \end{aligned} \tag{17}$$

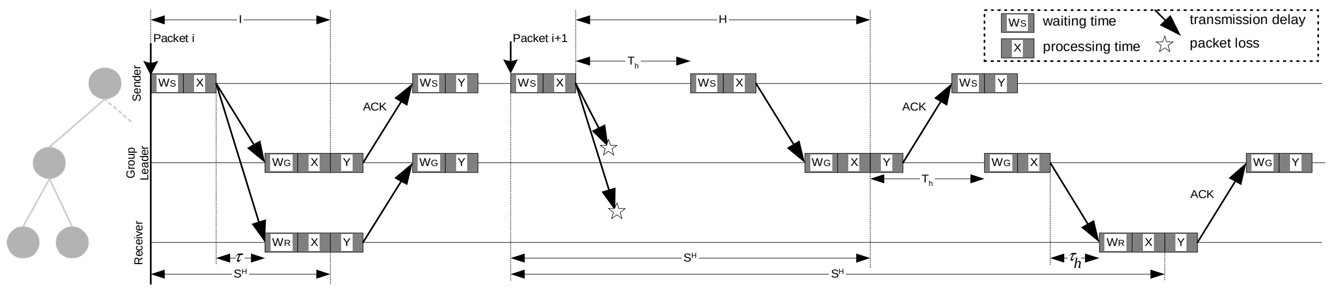

Figure 1: Examples of packet delivery delay for protocol (H)

2.5. Tree-based Protocol (H)

If a node in a tree-based protocol has lost a data packet and a retransmission is necessary, the missing ACK packet is detected by the group leader. If this group leader has lost the data packet as well, the group leader's group leader is queried and so forth. As a prerequisite for the delay analysis we will determine the height of the ACK 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:

$$h = \log_B \Big((R+1)(B-1)+1 \Big), \tag{18}$$

where B is the number of members in a local group (i.e. the branching factor of the ACK tree). To obtain the mean delay, we obtain the average tree height $\tilde{h}$:

$$\tilde{h} = \frac{\Big(\sum_{i=1}^{h-2} (i+1) * B^i \Big) + \Big(R - \sum_{j=1}^{h-2} B^j \Big) h}{R}. \tag{19}$$

For a delay analysis of tree-based protocols we have to consider sender and receiver nodes as well as inner nodes. Figure 1 shows the delay components.

2.5.1. *Mean Waiting Times at Sender and Receiver*

The packet flows, the sender and receiver loads ϱ_S^H, ϱ_R^H and the mean waiting times $E(W_S^H)$, $E(W_R^H)$ are analogous to protocol (A) with B instead of R for obtaining M^H.

2.5.2. *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:

$$\varrho_G^H = \lambda_{timeout}^S E(X) + \lambda_a^S E(Y) + \lambda^R E(X+Y). \tag{20}$$

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

$$E(W_G^H) = \frac{\lambda_{timeout}^S E(X^2) + \lambda_a^S E(Y^2)}{2(1-\varrho_G^H)} \\ \frac{+\lambda^R \Big(E(X^2) + E(Y^2) + 2E(X)E(Y) \Big)}{2(1-\varrho_G^H)}. \tag{21}$$

2.5.3. *Overall Delay of Protocol (H)*

We assume that T_h is the inner node timeout delay, $\tilde{h}$ the average number of hierarchy levels of the ACK tree and B the branching factor. If no retransmission is necessary, the delay from the initial transmission $E(I)$ is (see Eq. 6):

$$E(I) = E(W_S^H) + E(X) + \tau + E(W_G^H) + E(X). \tag{22}$$

Note that we have simplified this equation by assuming that the receiver is always a group leader and therefore take $E(W_G^H)$. Now we want to determine the delay for a hierarchical retransmission step on condition that the parent node has received the packet correctly. The time for a hierarchical retransmission step $E(H)$ is:

$$E(H) = \Big(E(M_r^H | M_r^H > 1) - 1 \Big) \Big(T_h + E(W_G^H) + E(X) \Big) \\ + \tau_h + E(W_G^H) + E(X). \tag{23}$$

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 a packet and the next parent retransmits it and so forth and multiply these probabilities with the expected delays. The overall delay is then:

$$E(S_\phi^H) = \Big[\sum_{i=o}^{\tilde{h}-2} q_D^i (1-q_D) \Big(E(I) + iE(H) \Big) \Big] \\ + q_D^{\tilde{h}-1} \Big((\tilde{h}-1)E(H) + E(W_S^H) + E(X) \Big). \tag{24}$$

Analogous to protocol (A) we can define the threshold delay $E(S_\gamma^H)$:

$$E(M) = \frac{ln(1-\gamma)}{ln(q_D)}, \gamma \geq 1 - q_D \tag{25}$$

$$E(H) = \Big(E(M|M>1) - 1 \Big) \Big(T_h + E(W_G^H) + E(X) \Big) \\ + \tau_h + E(W_G^H) + E(X) \tag{26}$$

$$E(S_\gamma^H) = E(W_S^H) + E(X) + (\tilde{h}-1)E(H). \tag{27}$$

3. NUMERICAL RESULTS

We examine the expected delays of the analyzed protocols by means of some numerical examples. The chosen delays are according to measurements in [8] $X = 500\mu s$ for data packets and $Y = 100\mu s$ for control packets. For protocol (U) we have chosen $X = 500\mu s$ as well as $X = 1000\mu s$ since FEC increases the processing time. Analogous to [2], 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 global propagation delay is chosen as $\tau = 10ms$ and the local propagation delay for protocol (H) as $\tau_h = 5ms$. The timeouts are chosen as the doubled propagation delay, i.e. $T_s = 20ms$ and $T_h = 10ms$.

Figure 2 plots the expected average delay and the expected threshold delay to reliably deliver all packets with probability 0.999, with data and control packet loss probability $q_D = p_A = 0.1$. On the left, the number of receivers are varying and the data rate is $\lambda = 0.01$. On the right, the data rate is varying and the number of receivers is 1000. As it can be seen, protocol (A) has only a limited scalability. If the data rate or number of receivers exceeds a certain limit, protocol (A) becomes quickly overwhelmed with

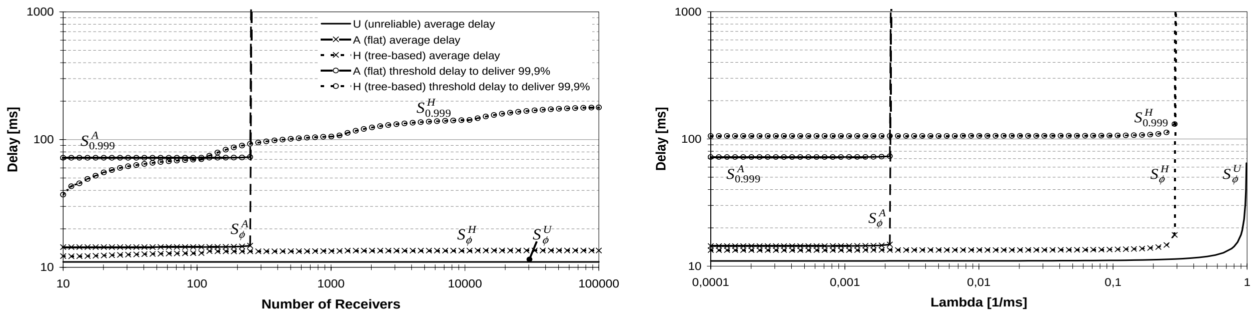

Figure 2: Average delay with packet loss probability 0.1

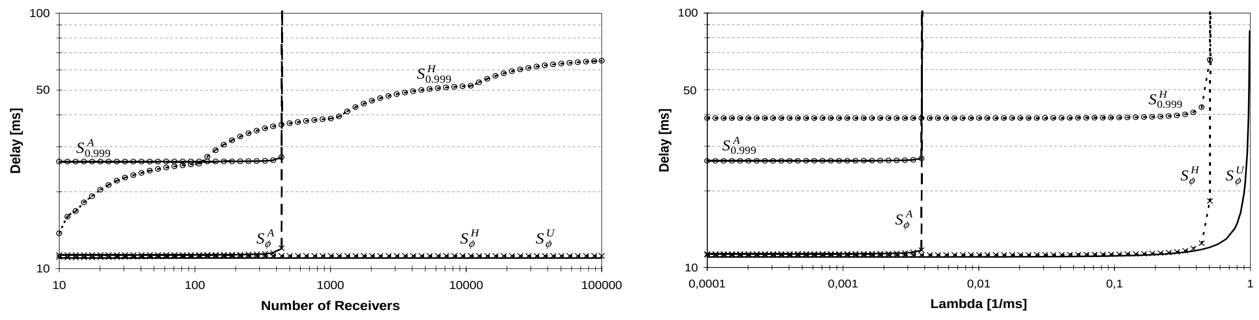

Figure 3: Average delay with packet loss probability 0.01

feedback control messages. Protocol (H) provides scalability for any number of receivers and even for high data rates before it becomes saturated at $\lambda = 0.3$. Protocol (U) is shown in the figure with $X = 500\mu s$. With $X = 1000\mu s$ the delay is increased only moderately by 1s or 9%.

By comparing Figure 2 with packet loss probability 0.1 and Figure 3 with packet loss probability 0.01 we can see that low packet loss probabilities result in low average delays comparable to unreliable or FEC based approaches. This means, in such a scenario, even reliable multicast protocols provide a good average delay behaviour. However, we can also conclude that the threshold delay for reliably deliver all packets with probability $\gamma = 0.999$ is significantly increased compared to the average delay. By increasing γ, the delay increases also. A probability of $\gamma = 1$ would result in an infinite delay. For protocol (H) the threshold delay increases with the tree-height, since with increased height, more hierarchical retransmission steps must be performed in the worst case. Which delay of both, average delay or threshold delay, is more important depends on the application requirements.

4. SUMMARY

We have analyzed the delay behaviour of three generic multicast protocols. The first protocol class (U) is an unreliable best effort or FEC based approach whereas classes (A) and (H) use a retransmission scheme that guarantees reliability. Protocol class (H) extends (A) with a hierarchical ACK tree.

The numerical results show that only the unreliable and the hierarchical protocol class provide scalability for large receiver groups. For delay sensitive applications we can conclude that reliable multicast protocols increase the average delay only moderately. If the packet loss probability is low, the average delays are almost identical for all considered protocol classes. Besides the average delay we have evaluated the delay to reliably deliver all data packets with a certain probability, for example 99.9%. Compared to the average delay, this delay is significantly higher.

5. 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. 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.

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

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.