Title: Scalable Reliable Multicast Using Receiver Grouping

Author 1: Elias G. Khalaf
Assistant Professor
Department of Mathematics and Computer Science
6363 St. Charles Ave., Campus Box 35
Loyola University New Orleans
New Orleans, LA 70118

Email: ekhalaf@loyno.edu

Phone: (504) 865-2024
Fax: (504) 865-2051

Author 2: S. Sitharama Iyengar
Professor and Chairman
Department of Computer Science
298 Coates Hall
Louisiana State University
Baton Rouge, LA 70803

Email: iyengar@csc.lsu.edu

Phone: (225) 578-1252
Fax: (225) 578-1465

Author Presenting Paper: Elias G. Khalaf

Keywords: Reliable Multicast Protocol Receiver Grouping

Scalable Reliable Multicast Using Receiver Grouping

Elias G. Khalaf
Department of Mathematics and Computer Science
Loyola University New Orleans
6363 St. Charles Ave., Campus Box 35
New Orleans, LA 70118

S. Sitharama Iyengar
Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803

Abstract

Scalable Reliable Multicast Protocols have been the subject of much research in recent years. We propose a new protocol that groups receivers for error recovery into fixed-size groups, thus reducing their processing requirement to O(1). This means that, regardless of the size of the multicast session, the processing requirements for receivers remains constant. The processing requirement on the sender is tunable according to its processing capabilities and/or the expected multicast session size. The concept of Local Recovery is then applied which further improves the processing requirements, especially on the sender. The processing requirements on both the sender and the receiver are analytically studied, and compared with another same-class protocol.

Keywords: Reliable multicast, retransmission scoping, receiver grouping.

1. Introduction

Internet Multicasting refers to the delivery of data packets from a source (*sender*) to a set of destinations (*receivers*), rather than just a single destination [1]. Data packets can be sent from the sender to each receiver separately, but that would be wasteful of network bandwidth and of the sender's processing power. Instead, using multicasting, network routers perform replication of data when necessary to eventually reach all destinations. The sender in this case only sends out one copy of the data. The receivers, collectively, make up a *multicast group*.

The area of reliable multicast transport protocols has been a very active research area of the past few years. Many applications such as shared whiteboard, file replication and update, stock quote dissemination, web cache update, among others, require reliable multicast. Many researchers have been attempting to create one generic protocol that will be suitable for all applications. It has been realized, however, that there is no one protocol that serves this purpose. Applications must have greater control over the segmentation and framing of data units.

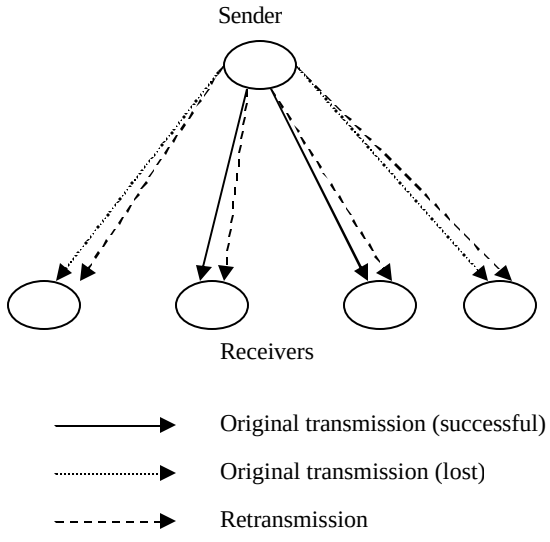


Fig. 1. Retransmission Scoping Problem

On the other hand, there has been a flood of new multicast protocols, each trying to serve a different multicasting need or trying to improve on an existing protocol. However, one major problem to overcome with all protocols is the *retransmission scoping problem* [5] as depicted in Fig. 1. This has to do with the unnecessary processing overhead that is imposed on a receiver that has already received a packet correctly. The problem in reliable multicast is how to scope retransmissions so as to shield receivers and the links leading to them form loss recovery due to other receivers [5].

In this paper we focus on processing requirements at the receivers and at the sender, with the conjecture that network bandwidth will keep outgrowing processing speeds for the next few years. Therefore, we devise new schemes for congestion control in multicast enabled internets that minimize processing requirements on receivers (which constitute the bulk of the network) as well as reduce or tune those requirements on senders. We devise a new approach to error recovery in negative acknowledgement (NACK)-based multicast transport protocols, and then we combine it with the well-known approach of Local Recovery to achieve even greater reduction in processing requirements.

2. Related Work

Several reliable multicast protocols and architectures have been proposed. One of the leading protocols is (scalable reliable multicast) SRM [2], which is NACK-based. SRM suffers from the unwanted redundant packets sent to receivers due to the fact that retransmissions are multicast to the whole group, which is wasteful of bandwidth and receiver processing power. The concept of local recovery helps SRM reduce global retransmissions by isolating domains of loss, but the local groups can themselves suffer from the same global problem should the size of the multicast session grow too large. Other protocols like LBRM [4] and RMTP [8] (and others like [3]) use a hierarchical approach with designated receivers to supply repairs to lower-level receivers. But the placement of such designated receivers and their processing requirements are not fully studied.

The first analytical attempt in [9] and [10] to study the performance of reliable multicast protocols paved the way for researchers in this area. In [9], the performance of two fundamental classes of reliable multicast protocols were compared, namely, the *sender-initiated* and the *receiver-initiated* classes. Many other protocols were designed that belonged to neither category. In effect, two more categories were added by [7], namely the *ring-based* protocols and the *tree-based* protocols. The analysis and results can be found in [11] and will be skipped here for space limitations.

3. Protocols and System Model

A. System Model

This model consists of one sender *S* multicasting a continuous stream of packets to *R* receivers. The model could possibly be extended to have *R* receivers where any receiver has an equal chance of being a sender itself. For our model we assume:

- All loss events at all receivers for all transmissions are *mutually independent*;
- The probability of packet loss, *p*, is independent of the receiver;
- ACKs (positive acknowledgments) and NACKs (negative acknowledgments) are never lost and these packets (referred to as *control packets*) are typically small;
- Processing requirements at the hosts are more important than network bandwidth in determining the throughput of reliable multicast protocols.

B. Protocol Description

Receiver Grouping is a process that randomly groups *n* receivers for the purpose of packet loss recovery. In the global mode of the protocol, *S* acts as a *Match Maker* and, upon request, groups *n* receivers with each other, *(n-1)* of which had been waiting to be grouped in a queue at *S*. While in the queue, these receivers will recover lost packets from *S*.

The new protocol proposed in this paper [6], called the *K* protocol, tries to achieve optimal processing requirements on both the sender and the receivers without incurring heavy additional

bandwidth. It utilizes a new and efficient error recovery scheme. The protocol exhibits the following properties:

- It utilizes a *receiver-initiated* loss recovery scheme;
- It is NACK-based;
- The sender is generally not involved in the recovery of lost packets, unless no other receiver can provide the missing packets;
- It introduces the concept of *receiver grouping*. A variation of this protocol uses *local receiver grouping* in order to reduce congestion further on the network.

The *K* protocol exhibits the following behavior:

- Upon joining a multicast group (session), a receiver R_i initially pairs up with the sender *S* until a *group G* of *n* receivers is formed;
- Receiver R_i informs *S* that it is looking for a group;
- *S* saves the address of R_i in a queue and waits until enough receivers request grouping;
- Upon receiving a request for grouping, *S* checks if it has *(n-1)* receivers waiting for grouping. If so, then *S* informs R_i and all other *(n-1)* receivers in the queue that all of them are going to be buddies in the same group for the purpose of error recovery. At that moment, R_i drops its pairing with *S* and groups with its new buddies in *G*. If there are no receivers waiting for grouping, *S* behaves as in the previous step;
- All receivers in *G* now use *point-to-point* (peer-to-peer) communication for error correction in a NACK-based fashion (as opposed to multicasting their NACKs). When group members detect lost packets, they use one of several possible *group recovery schemes* (see section E). The schemes work in such a way that all *n* receivers will end up having the missing packet. If at least one member has it, then *S* should not be bothered and all members should recover their loss from that one receiver. Only if none of the receivers in *G* has a missing packet does *S* get consulted;
- Upon leaving a multicast session, a receiver informs its group members that it intends to leave. The group members then break the group and act as if is they have just joined the session, i.e. they pair with *S* until enough members are found to form a new group;
- If some receiver in a group is not responding, then its buddies in the group can detect this either by polling it periodically, or by expiring timeouts, in which case they behave as in the previous step;
- If a receiver R_i has been dropped (dumped) by its group members, then upon reestablishing a contact with any of the former members, R_i is informed that it needs to find another group. R_i then behaves as if it's joining anew.

C. Observations

We observe the following properties for the *K* protocol:

- *S* acts as a *match maker* in forming *G*;
- Protocol *K* is *stateless* with respect to grouping, i.e. it keeps no record of which receivers are grouped with each other at any point in time. Statelessness is very important to have; otherwise, it would be very costly in terms of memory requirements for a server to remember all groups. In addition, that information will become stale very quickly due to the dynamic nature of receivers joining and leaving a multicast session;
- Unless there are delays in processing groupings, for example due to insufficient processing power, no more than *(n-1)* receivers should pair with *S* at any point in time, since they will end up being grouped with each other;
- If *R*, the total number of receivers, is a multiple of *n*, then *R/n* groups will be formed with no leftover receivers. Otherwise we will have at most (*R mod n)* receivers paired with *S* at any point in time.

It is worthy to note that all receivers are required to group under *K* and it is assumed that none of them are incapable of doing so.

D. Example

The operation of the *K* protocol is depicted in Fig. 1. In this example, there is one sender *S* and seven receivers, *R1* through *R7*. The requests for grouping randomly arrive at *S* from *R2*, *R4*, *R1*, *R6*, *R3*, *R5* and *R7* respectively in that order. The resulting groupings are shown in the figure. Receivers *R3* through *R7* lose the packet/segment

and request repairs from their buddies in the group. In group (*R2*, *R4*, *R1*), *R4* did not have the packet so it recovered it from *R4*. All receivers in the group *(R6, R3, R5)* lost the segment, so one of them (*R5* in this case) recovered the segment from *S* and supplied it to all others. Note that since *7 mod 3 = 1*, only one receiver (the last to request grouping), namely *R7*, did not find a group, so it paired with *S* for error recovery, awaiting a group.

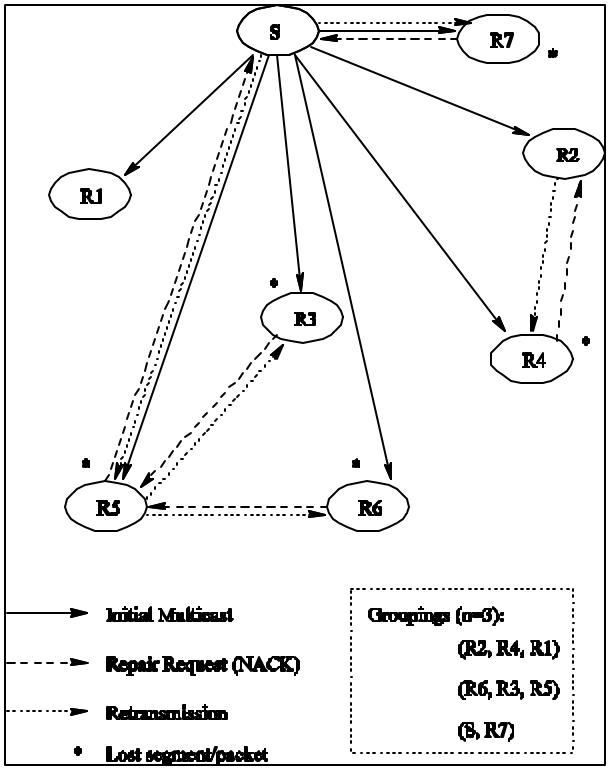


Fig. 2. Example of the operation of *K*

E. Group Recovery Schemes

Receivers in a group of size *n* can deploy several error recovery schemes, but some can be more efficient than others, especially when a large number of receivers in the group did not receive a multicast packet. However, since the number of receivers in a group is constant, even a total ordering on the sending of NACKs and retransmissions, an $O(n^2)$ operation, would not pose a severe threat on the processing load of the receivers in a group. Three recovery schemes can be found in [6], each with its advantages and disadvantages.

F. Effect of Grouping on Sender

The grouping operation involves the reception of *n* messages from *n* receivers, the storage of the addresses of *(n - 1)* receivers until they get grouped, and the sending of *n* messages to all *n* receivers informing them that they are to form a group. Message sizes in this operation are very small and the storage needed is insignificant. At most, *(n - 1)* additional storage buffers are needed on the server to store the addresses of *(n - 1)* receivers waiting to be grouped; once the n^{th} receiver requests grouping, all *(n - 1)* buffers are released. In addition, grouping is generally a one-time operation for each group of *n* receivers for the life of a multicast session, provided the receivers remain healthy for the duration of the session. Therefore, in the best case, grouping only requires the sender to process a total of *2R* messages during the life of the session. In the worst case, all receivers join the session simultaneously - an unlikely scenario - in which case *S* receives *R* requests for grouping and is then expected to perform *R/n* grouping operations. In reality, receivers dynamically join/leave multicast sessions at different points in time, although a surge could be expected at the beginning of a multicast session.

4. Processing Cost Analysis

In order to understand the improvements achieved with the *K* protocol, we now analyze the processing requirements on receivers as well as on the sender. We follow the same analysis and use the same notation presented in [10] and [9]. The notation used in the processing cost analysis is described in Table 1.

A. The Receiver

In order to analyze the processing requirements at a receiver R_i, we must first consider the following facts:

- R_i only receives NACKs from its buddies in the group, which amount to at most *(n - 1)* NACKs;
- R_i supplies its *(n - 1)* buddies with missing packets via point-to-point communication, at most *(n - 1)* times per packet transmission or loss;

- A receiver asks *S* for a lost packet only if none of its buddies in the group have it. This takes place at most once per packet transmission/loss.

In doing performance analysis at the receiver we must consider the time it takes each receiver to do the following:

- obtain data from higher layers;
- process NACK from *(n - 1)* buddies;
- send *(n - 1)* NACKs to its buddies or to *S* if none have the packet;
- process received packet from buddies (at most *(n - 1)* times) or from *S* (once);
- send packet to buddies (at most *(n - 1)* times).

Therefore, the expected processing time at the receiver can be expressed as the sum of the above times to get

$$Y^K = Y_f + (n-1)Y_n + (n-1)X_n + (n-1)Y_p + (n-1)X_p.$$

Taking the expectation of Y^K we get

$$E[Y^K] = E[Y_f] + E[(n-1)]E[Y_n] + E[(n-1)]E[X_n] + E[(n-1)]E[Y_p] + E[(n-1)]E[X_p].$$

Now since $E[n] = n$ for constant n, we get $E[Y^K] = O(n)$. Therefore, $E[Y^K] \in O(1)$.

This means that the processing requirement at the receiver is *constant* and is *independent of R*, which is a highly desirable property in order to achieve optimality, and is the major result of this paper.

B. The Sender

We are interested in the mean processing requirement per packet in order for the sender to multicast packets reliably to all of the receivers. Processing requirements at the sender can be expressed as the total time needed to perform the following:

- prepare and transmit the first packet
- process all NACKs sent by receivers
- process all retransmissions to receivers in response to the NACKs

Table 1. Notation

X_f	Sender time to feed packet from application to transport layer
X_p	Sender time to process transmission of a packet
X_n	Sender time to process a NACK
Y_p	Receiver time to process a newly received packet
Y_t	Receiver time to process a timeout
Y_n	Receiver time to process and transmit/receive a NACK
p	loss probability at a receiver
M	Number of receivers requesting repair from sender
R	Total number of receivers in session
n	Number of receivers is a group
X^w, Y^w	Send and receive packet processing time in protocol *w*.

B. The Sender

Therefore, the expected processing requirements at the sender under the *K* protocol can be expressed as $X^K = X_f + M(X_n + X_p)$, where *M* is the number of receivers requesting repair from *S* when their buddies cannot supply the lost packet. Taking the expectation of X^K we get

$$E[X^K] = E[X_f] + E[M](E[X_p] + E[X_n]).$$

It is worthy here to note that error correction from the sender is done on a one-to-one unicast communication channel. Therefore, we are interested in finding the number of receivers that will request correction from *S*.

The probability of a group *(R1, R2, ... ,Rn)* not receiving a packet is now reduced to p^n. We can now treat the system as a session of *R/n* receivers (each being a group of *n* receivers) with the probability of losing a packet of p^n. The expected value of *M* can now be given as $E[M] = (p^n/n)R$. Substituting in $E[X^K]$ we get

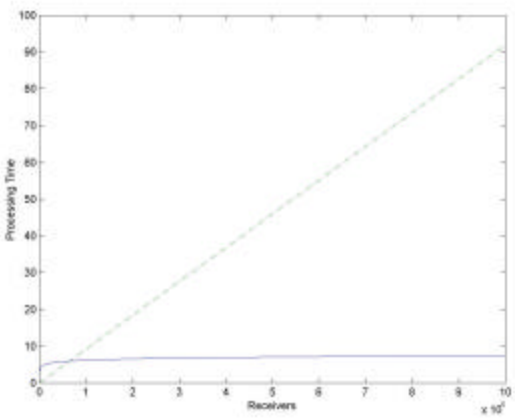
$$E[X^K] = E[X_f] + (p^n/n)R\,(E[X_p] + E[X_n]).$$

Please note that the *n* in X_n is for NACKs and is not the same as the size of a group. Therefore,

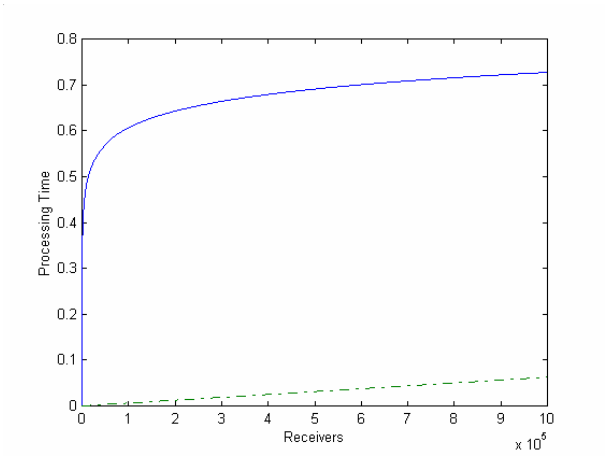
$$E[X^K] \in O(1 + (p^n/n)R).$$

As $p \to 0$, $E[X^K] = O(1)$. The sender's processing requirements can thus be tuned to its capabilities, by changing the size of the group (*n*). The larger

the receiver group size, the less the sender's processing requirements, as more error correction takes place within each group. This can be illustrated in Fig. 3 where a significant processing improvement was achieved by increasing the receiver group size from 5 to 7 in *K*, while maintaining a 20% probability of packet loss. The dashed line corresponds to *K*, and the solid line to a receiver-initiated class of protocols with NACK avoidance (RINA) [11] that exhibits a sender's requirement of $O(1 + pln(R)/(1 - p))$, and a receiver's requirement of $O(1 - p + pln(R))$. The major drawback of RINA is the receiver's processing requirements; this has been reduced to $O(1)$ in *K*.



(a) $p = 0.2$, $n = 5$



(b) $p = 0.2$, $n = 7$

Fig. 3. RINA (solid line) vs. *K* (dashed line) – increasing group size from 5 to 7.

5. Grouping with Local Recovery

We combine the concept of *local recovery* [3], [4], [8], with that of grouping to achieve even lower processing requirements on the sender. The major difference in this modified *K* protocol, which we will call *KL*, is that repair servers (special designated nodes in the network) now play the role of the sender in terms of grouping and error recovery. Only if a repair server does not have a lost packet does it consult the sender or another repair server in the hierarchy. The details of the *KL* protocol are given in [6], in which the sender's processing requirements is shown to be

$$E[X^{KL}] \in O(B(1-p) + Bpln(B))$$

where *B* is the branching factor of the network. Notice that if *p* is constant, then $E[X^{KL}] \in O(Bln(B))$, and that if $p \to 0$, then $E[X^{KL}] = O(1)$, which is always a desirable property. The receiver's processing requirements under *KL* remains constant since the group size does not have to change, and neither do the recovery schemes. In addition, as far as the receiver is concerned, the local repair server is indistinguishable from the sender. Therefore, $E[Y^{KL}] = O(1)$.

6. Conclusions

In this paper we introduced the new concept of receiver grouping as a new model for requesting error correction in a multicast session. The major contribution is that the processing requirements on the receiver side have been reduced to $O(1)$ – a constant! This was achieved with the *K* protocol, in which a receiver need not process unwanted packets resulting from multicast retransmissions due to other receivers. The sender's processing requirements under *K* can be tuned to its capabilities, by changing the size of the receiver group. The concept of local recovery was applied to further reduce the processing requirements on the sender resulting in the *KL* protocol, while maintaining the constant processing requirements on the receivers.

Our work can proceed in several directions since there are many unresolved issues to deal with. First, the choice of the group size is critical before

a multicast session starts, so the question is how to optimally choose *n*, and whether this choice can be altered dynamically during a multicast session and the effect this may have on the sender, receivers and network bandwidth. Another direction is to find the optimal group recovery scheme within each group, based on what the receivers know about each other and whether they have a lost packet or not. One other possibility is to consider electing a group leader, and then have the leaders join a separate multicast session for error recovery. Finally, the effect of grouping on network bandwidth can be studied, aside from the processing requirements on receivers.

7. References

[1] S. Deering, "Multicast Routing in a Datagram Internetwork," Ph.D. Dissertation, Stanford University, Palo Alto, CA, December 1991.

[2] S. Floyd, V. Jacobson, S. McCanne, C. Liu, and L. Zhang, "A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing," *IEEE/ACM Trans. Networking*, Vol. 5, pp. 784-803, December 1997.

[3] M. Hofmann, "Enabling Group Communication in Global Netoworks," *Proceedings of Global Networking '97*, Calgary, Alberta, Canada, November 1996.

[4] H. W. Holbrook, S. K. Singhal and D. R. Cheriton, "Log-Based Receiver Reliable Multicast for Distributed Interactive Simulation," *Proceedings of ACM SIGCOMM*, pp. 328-341, August 1995.

[5] S. Kasera, G. Hjalmtysson, D. Towsley and J. Kurose, "Scalable Reliable Multicast Using Multiple Multicast Channels," *IEEE/ACM Trans. Networking*, Vol. 8, No. 3, pp. 294-310, June 2000.

[6] Elias G. Khalaf, "Congestion Control Mechanisms for Internet Multicast Transport Protocols", Ph.D. Dissertation, Louisiana State University, Baton Rouge, Louisiana, May 2000.

[7] B. N. Levine and J. Garcia-Luna-Aceves, "A Comparison of Known Classes of Reliable Multicast Protocols," *Proceedings of ACM Multimedia*, November 1996.

[8] J. C. Lin and S. Paul, "RMTP: A Reliable Multicast Transport Protocol," *Proceedings of IEEE Infocom*, pp. 1414-1424, 1996.

[9] S. Pingali, J. Kurose and D. Towsley, "A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols", *Proceedings of ACM Sigmetrics Conference*, May 1994.

[10] D. Towsley, J. Kurose and S. Pingali, "A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols", *IEEE Journal on Selected Areas in Communications, Vol. 15, No. 3, pp. 398-406*, April 1997.

[11] S. Paul, *Multicasting on the Internet and its Applications*, Kluwer Academic Publishers, 1998.