

# Light-weight Reliable Multicast Protocol

Tie Liao  
INRIA, Rocquencourt, BP 105  
78153 Le Chesnay Cedex, France  
Tie.Liao@inria.fr

## Abstract

This paper describes the design and implementation of LRMP, the Light-weight Reliable Multicast Protocol, which has been in use by a significant number of projects. LRMP provides a minimum set of functions for end-to-end reliable multicast network transport suitable for bulk data transfer to multiple receivers. LRMP is designed to work in heterogeneous network environments and support multiple data senders. A totally distributed control scheme is adopted for local error recovery so that no prior configuration and no router support are required. Subgroups are formed implicitly and have no group leaders. Packet loss is reported upon a random timeout first to the lowest level subgroup, then to a higher subgroup and so on until it is repaired. This simple scheme is rather efficient in duplicate NACK and repair suppression. Some congestion control mechanisms are included to fairly share network bandwidth with other data flows.

## 1. Introduction

The emergence of IP Multicast [1] has made possible to efficiently transfer data from one source to multiple receivers simultaneously, i.e., without duplication in the network. While group communication applications based on unreliable multicast such as real time multimedia conferencing have flourished since several years, applications based on reliable multicast are continuously growing, ranging from file transfer to collaborative work.

A number of reliable multicast transport protocols have been designed to meet this increasing requirement. Earlier reliable multicast protocols either had the scalability problem or were designed for a special environment (e.g. LAN) [2] [3] [4]. Consequently they have not been widely accepted. Recent work attempts to address more specifically the scalability and congestion control issues, two important issues that have to be dealt with by any reliable multicast protocol. MFTP [5] makes use of redundant encoding (FEC) to reduce the feedback traffic from receivers. It is particularly suitable for data transfer over satellite links where there is little or no bandwidth for upstream channel. RMTP [6] organizes receivers in a hierarchical tree structure to aggregate feedback information. It was originally designed for data transfer from one source to a group of receivers. PGM [7] tries to add router support to

do error report and retransmission. Although such support could be very useful, its availability is not clear at present. Indeed there are still a lot of reliable multicast protocols not mentioned here, but none of them has well solved the scalability and congestion control issues.

It was strongly felt the need for a flexible yet simple reliable multicast transport protocol that places less constraints on the deployment and can be used in the global Internet. A variety of application services have this demand in common, such as database update propagation, data replication (e.g., proxy server, object distribution), shared work space, etc. A common characteristic of these applications is that they can tolerate some degree of end-to-end transmission delay. In contrast to real time continuous media applications, the reliability can be achieved at the expense of end-to-end delay. We attempt to provide a usable reliable multicast protocol through a careful study of all the related issues. While some design principles of unicast reliable protocols such as TCP can be applied, the design philosophy for reliable multicast protocols is rather different. Reliable multicast protocols have to offer good scalability and to be fair to competing TCP flows, otherwise they may be potentially dangerous to the Internet if used in a large scale.

In this paper, we describe LRMP, the Light-weight Reliable Multicast Protocol, which is a receiver-initiated reliable multicast transport protocol. LRMP guarantees sequenced and reliable data delivery from one sender to a group of receivers. LRMP allows multiple data senders in a multicast session and places no restriction on receiver's membership. Often reliable multicast protocols face the problem of complexity that grows significantly with the level of scalability and fairness of congestion control. That makes the protocol difficult to be engineered. LRMP is designed to be light-weight in terms of protocol overhead and simplicity in control mechanisms. This point is often ignored by many reliable multicast protocols.

The novel aspects of LRMP mainly include a new local error recovery mechanism, called random expanding probe (REP); a congestion control scheme based on NACK (negative acknowledgements) and congestion indication from receivers; a protocol independent FEC integration scheme. Local error recovery is a technique used to isolate reception errors in a limited region when possible. Existing approaches [6] [8] try to organize receivers into a hierarchy of subgroups. In each subgroup, there is a designated group

leader which is responsible for aggregation of receiver feedback and interaction with a higher level leader. The group leaders are either statically configured or dynamically designated. However it is not clear how well they can adapt to highly dynamic groups and the case where every receiver may become a data sender.

LRMP uses a different approach for local error recovery. In LRMP, hierarchical subgroups are formed implicitly. When a reception error occurs, a receiver probes the lost packets starting from the lowest level subgroup and up to the whole group. Each probe is controlled by a random timer to avoid NACK duplication, similar to SRM [9]. The holder of the lost packets which may not be the original sender sends the repair packets also upon a random timeout for duplicate repair suppression. Due to this particularity, this scheme is called random expanding probe.

The remainder of this paper is organized as follows: after a description of the main design objectives in Section 2, Section 3 describes the local error recovery mechanism. Section 4 presents the flow and congestion control scheme. Section 5 describes additional features such as the selective receiver report mechanism and FEC integration. Section 6 discusses implementation issues and gives some measurement results. Finally we give a summary of this paper and outline future work.

## 2. Design Objectives

The design of LRMP is motivated by the error control scheme of SRM [9] and other existing reliable multicast protocols. LRMP is intended to take advantages from the existing protocols and avoid their drawbacks. The design is mainly guided by the top level design goals which must be met at the first stage. Since many of reliable multicast issues are still under research, some second level goals are established and are expected to be fully met at the second stage.

In the Internet environment, hosts are generally interconnected via heterogeneous links and dispersed world wide. The quality of service of their network links such as the bandwidth and packet propagation time varies from one to another. The primary goal is thus to provide a reliable data delivery service over such a heterogeneous environment. The top level goals of LRMP are summarized as follows:

- It must be reasonably reliable over the current infrastructure of Internet. That means that it must ensure reliable data delivery under normal network conditions without intermediate router support.
- It must scale well to large groups of users, i.e., it must be cost effective in terms of network resource usage.
- It must continue to work for majority of receivers in case where some receivers are either extremely slow or behave abnormally, or the network is partitioned somewhere in the multicast routing tree.

These goals influence significantly the choice of control mechanisms in LRMP. The level of reliability is the most important argument in the design of a reliable multicast protocol. To achieve fully reliable data transfer, some handshake (connection establishment) is required before the data transmission and synchronization points (ACKs) are necessary during the data transmission as in the case of unicast (e.g., TCP). It is quite different in multicast where three scenarios can be distinguished:

1. Tightly coupled group. The hosts participating in group communication are registered upon permission. An unknown host may be rejected from the group. In this case, data should be guaranteed to be delivered to all registered hosts. Therefore reception acknowledgements or something similar to TCP SACK [10] are required to synchronize data transmission for all receivers. RMTP [6] looks like this type of approach.
2. Loosely coupled group. Any host can join or leave the group at any time and at any location. Synchronization points can not be applied to this case since there is no permission in joining the group and a malicious receiver may block the data transmission for the whole group. Data transmission can only be controlled on the behave of majority of receivers.
3. Mixed group. Some hosts are permitted and the others are not (they are just listeners). In this case, data transmission is synchronized only for the permitted hosts. Not permitted hosts can act as in a loosely coupled group.

LRMP is particularly designed for loosely coupled groups where there is generally no single starting point for all receivers. LRMP will work in NACK-only mode to ensure reliability for the majority of receivers. In addition, a provision is also made for the implementation of mixed NACK and ACK mode to achieve reliability for all receivers in a tightly coupled group. Hereafter all discussions refer to loosely coupled groups except indicated otherwise.

Other important arguments include the scalability and the normal behaviour upon the interference or attacks from a few receivers. Scalability generally means that the control and retransmission traffic remains at a reasonable level in a large multicast group. Besides the multicast protocol should be easily usable over an arbitrary set of senders and receivers. Since it is almost impossible that all receivers behave uniformly in an arbitrary multicast group, it is thus extremely important to protect a multicast protocol against a few receivers that either badly behave or have serious network problems.

The above goals lead to the adoption of a distributed control scheme which is thought to be the most flexible and scalable (see the next section), otherwise a scheme like SCE [4] and organized tree structure [8] could be adopted instead.

The second level goals include:

- Fairness in congestion control.
- It must support a variety of application services.
- To provide reasonable performance.
- Easy evolution.

Congestion control is as important as reliability. Though it generally should be among the top level goals, it is considered as a second level goal since it concerns many strategic and technical issues that are still open at present. For example, it is still not clear at present how a multicast data flow should share the bottleneck network bandwidth with unicast flows (e.g. equally or not equally). A data flow from one sender to one hundred receivers could be considered more important than a unicast flow.

In addition, performance was initially considered not as critical as the protocol is targeted for bulk data transfer. Quickly it is considered as a second level goal due to the fact that the protocol is being used by an increasing number of interactive applications such as chat, whiteboard, etc. LRMP is thus intended to offer reasonable performance in terms of throughput and transmission delay.

The second level goals are partially met in the current design and should be entirely met in near future. So it is important to ensure easy evolution and make the protocol extensible. This paper is not intended to give all the details on how these goals are achieved, instead it will be focused on error control and congestion control mechanisms. More details can be found in the technical specification [11].

### 3. Local Error Recovery

In TCP, when a packet is lost due to network congestion, it is retransmitted by the sender. In a multicast group the retransmission can also be performed by a third party which has successfully received the lost packets and is located close to the receiver which encountered the packet loss. This can avoid some control and retransmission traffic in the whole group. The transmission scope of an IP multicast datagram can be controlled through the scope field (TTL) of the IP header. In addition LRMP packets contain a scope field which equals to the initial scope value of IP datagram and remains constant during propagation.

The random expanding probe (REP) mechanism used here for local error recovery consists of three steps: divide a multicast session into a hierarchical subgroups; report error to a subgroup; send repair(s) to a subgroup. The contention in error report and sending repair is resolved using a random timer. Though there is no leader in each subgroup, we will show that REP can behave as if there are leaders in a subgroup.

#### 3.1 Hierarchical Subgroups

Yet another way to define the distance between two hosts is to use the number of hops, in other words, the number of intermediate routers to transit from the source to the destination. When an application sends multicast datagrams, it

must provide a scope value, known as the time-to-live (TTL) parameter, to indicate at which scope the datagram must be propagated. This scope value is contained in IP header and decreased by one when a datagram transits a router. A datagram will be dropped or not forwarded to the next node by a router if the scope field reaches zero or under a certain threshold. The scope field thus controls the radius of distance that a multicast datagram can reach.

To do local error recovery, a multicast group is broken into a hierarchy of subgroups. The division of group is receiver specific: to a particular receiver, a subgroup represents the receivers located in a region limited by a distance radius. For example, a subgroup with the scope value 15 includes all receivers in the local site. A subgroup with the scope value 63 includes all receivers national-wide<sup>1</sup>. Generally a datagram sent to a subgroup can not reach other entities outside that subgroup. By limiting error recovery in a subgroup, reception errors could be isolated to a region. Note that a subgroup contains another subgroup if its scope value is larger than the latter.

Figure 1 shows the hierarchy of subgroups viewed by the receiver  $R_i$ . The first level subgroup<sup>2</sup> contains only  $R_j$  and  $R_i$  itself. The second level subgroup is with the distance  $d_2$  which contains all entities except the sender  $S$  and the receiver  $R_n$ . The top level group is with the distance  $d_3$  and contains all entities in the session (e.g., the sender  $S$  and all receivers).

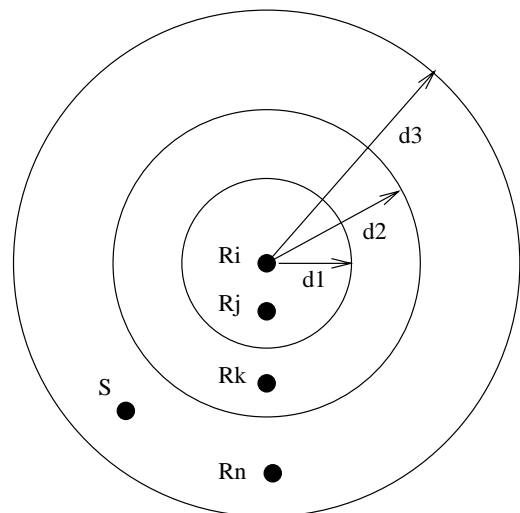


Figure 1. Hierarchy of subgroups

It is possible that the multicast path between two hosts is asymmetric, i.e., the number of hops from host A to host B may be different from that from host B to host A. As the gap of the subgroup scope values is generally enough large, in a given subgroup, if host A can reach host B, it is

1. This could be slightly different in different countries due to different routing policies.
2. The group level is counted bottom up.

most probable that host B can also reach host A.

There is trade-off between the number of subgroup hierarchies and the efficiency of error recovery. More the number of hierarchies, finer the local error recovery but larger the error recovery time. While the subgroup hierarchy could be configured at runtime by an application, LRMP uses by default the following rule for organization of subgroups. Given the time-to-live value (the nominal TTL) for a session, the session is organized into the following hierarchical subgroups:

Top level: whole group@TTL

Level 3: subgroup@63 if TTL > 63

Level 2: subgroup@47 if TTL > 47

Level 1: subgroup@15 if TTL > 15

The top level group, i.e. the whole group, must be present in every session. A descendant subgroup is created only if the nominal TTL is greater than the subgroup TTL value. Therefore there could be at maximum four levels of hierarchies. For example, a session with the TTL value equal to 63 has three levels. A session with TTL value less than and equal to 15 has only one level of hierarchy.

Every LRMP packet carries a scope field equal to the TTL value with that the packet is sent to the IP layer. When a receiver receives a packet (data or control), it knows that the packet sender is surely within a distance less than or equal to the scope value carried in the packet. The scope field allows a protocol entity to determine whether or not the packet sender belongs to one of its subgroups. An LRMP entity responds to a control packet using the same scope value as the received request packet. In this way, LRMP can work independently of how subgroups are divided at a particular receiver.

In each subgroup, every receiver plays a strictly equal role, there is no subgroup leader. Every receiver is responsible for recovery of its own lost packets. So no join or leave messages are necessary to explicitly form subgroups while they may still be implemented by applications in case of strong membership control.

### 3.2 Error Report using NACKs

In a normal operation scenario, a source multicasts a set of data packets to the session group address using the session TTL, i.e., the whole group. The transmission is controlled by a transmission interval (derived from the transmission rate). A receiver detects packet loss either by checking if there is a gap between the sequence numbers of successively received packets or the sequence number given by a sender report packet does not correspond to the expected sequence number for that sender.

LRMP makes no effort to recover data packets that were sent before a receiver joins the session. Otherwise an infinite buffer space is required at the sender side and a new security hole is opened for malicious attacks. The sequence number of the first data packet or sender report

packet is considered as the starting sequence number for a particular sender.

When packet loss is detected, the error report starts from the lowest level subgroup and then goes up one level until the top level group is reached. An exponential back-off timer, called NACK timer, is used for duplicate NACK suppression. The basic error report algorithm is described in detail as follows, the influence of transmission interval is not included for clarity:

0. Set the subgroup level  $i = 1$ .
1. Set the number of tries  $N = 0$ .
2. Schedules a random timer whose value is uniformly distributed in the range  $[t1, 2*t1]$ , where  $t1 = MRTT * 2^N$  and MRTT is the estimated mean round trip time to other protocol entities reachable in this subgroup.
3. During the timer period, if a similar NACK packet is received, goto step 5.
4. When the timer expires, if all lost packets have been received, the repair process terminates, otherwise send a NACK packet to the subgroup.
5. If the level of subgroup is not the top level, the level is increased by one,  $i \leq i + 1$ , goto step 1; otherwise, the number of tries is increased by one,  $N \leq N + 1$ . If  $N$  is less than the prefixed maximum number of tries (e.g. 8), goto step 2, otherwise the repair process is abandoned.

Loss report should be sequential, that is, the lost packets with lower sequence numbers should be reported first. Unless the current repair process is terminated normally or abnormally, the next loss report should not be started. In the worst case, an error report could be carried out once in each subgroup, and eight times in the top group. A NACK packet contains the sequence number of the first lost packet and a bitmap (32 bits) of the following lost packets. That allows to report up to 33 adjacent lost packets at once.

### 3.3 Retransmission

LRMP entities keep recently sent and received data packets in their buffer space. The buffer size, depending on the used transmission rate, should be enough large to allow loss repair. However if a receiver has less buffer space, it can drop in order packets after they have been processed by the application. In this case, this receiver will be not able to send local repairs to a subgroup.

Upon receipt of a NACK packet, a protocol entity is said eligible for sending repairs if it holds a part of the requested lost packets. An eligible entity checks first if the NACK is a duplicate for the purpose to eliminate possible duplicate retransmission. If a NACK is duplicate, it is simply ignored.

If a NACK is not duplicate, an eligible protocol entity schedules a random timer whose value is in the range

[MRTT, 2\*MRTT], where MRTT is the estimated mean round trip time to other protocol entities reachable in the corresponding subgroup. When the timer expires, if the lost packet(s) or a NACK reply packet have not been heard from other protocol entities, the protocol entity sends first a NACK reply and then the lost packet(s).

The reasons for sending a NACK reply are the following: first it is used to measure the round trip time between the NACK sender and the responder. Second it informs the sequence numbers of repair packets that will be sent by the responder. This allows partial repair, i.e., other eligible protocol entities can send repair packets other than those indicated by this NACK reply. If there remains a part of lost packets that no one can send in the subgroup, the NACK sender can report only this part to a higher subgroup. For example, suppose that the first lost packet and the first repair packet are of the same sequence number, the remaining part (new\_bitmap) of lost packets that can not be repaired by the responder is calculated using,

$$\text{new\_bitmap} = \text{nack\_bitmap} \& \sim\text{repair\_bitmap}$$

The transmission of repair packets are scoped with the same TTL value as the received NACK packet. It should be carried out with higher priority than the transmission of new data packets. Otherwise buffer overflow error may occur at receivers.

It should be noted that local error recovery represents additional vulnerability to malicious attacks since a third party may send modified data packets as repairs. It is thus desirable to use some mechanism to authenticate the received information. However this functionality is left to upper layers to perform and is out of the scope of this paper.

### 3.4 Optimizations

The basic algorithm described above can recover packet loss in a scalable way. To further increase efficiency and suppress duplicate NACK and repair packets, the basic algorithm can still be optimized. The following optimization procedures are an integral part of the REP mechanism but presented separately for clarity.

#### 3.4.1 Enabling/Disabling a Subgroup

To avoid unnecessary probe for missing packets in a subgroup, the following heuristic algorithm is proposed to determine whether or not a reception error can be recovered in a given subgroup. Basically if there are no other protocol entities within a subgroup, certainly no need to report error in that subgroup. If there are other protocol entities in a subgroup, errors may or may not be recoverable in that subgroup. A receiver enables a subgroup if it considers that there are some chances to recover errors in that subgroup, or disables otherwise. The enable/disable method allows to improve the error recovery time for missing packets.

A subgroup is enabled if one of the following conditions is true,

- at start-up.
- a repair packet with the subgroup TTL is heard.
- the disabled time is greater than the maximum disabled time, e.g., 3 minutes.

The last condition is to prevent the deadlock, that is, all receivers are in the disabled state. In this case, no receivers will send error report packets within that subgroup. So a timeout mechanism is necessary to retrigger the error recovery in that subgroup.

A subgroup is disabled if the number of consecutive failed error reports (i.e., upon which no repair packets were heard) in that subgroup is greater than or equal to the prefixed number, e.g., 5. The top level group should never be disabled since at least the sender is there and will reply to error report packets. If a subgroup is disabled for any reason, a receiver should jump to the next higher subgroup to report error.

#### 3.4.2 Delaying NACK and Repair

When a set of data packets are lost simultaneously at multiple receivers, the concerned receivers will all try to report packet loss. Though the probability of duplicate reports can be significantly reduced by using the random timeout mechanism, it is not null. In particular, in large groups and due to the variation of round trip time, duplicate reports may often occur.

In LRMP, every entity is identified by an integer identifier which is randomly chosen and unique in the session. This identifier is useful not only in the case where there are more than one entities at a single host address but also for duplicate NACK and repair suppression.

When a receiver receives an error report (i.e., NACK) which is a duplicate to its own, it must cancel the sending of its NACK for the current timer period. Besides when the next NACK timer expires, the receiver must not send its NACK if (1) it did not send a NACK in the previous timer period or (2) its identifier is higher than that of the sender of the NACK heard from the network. In this way it is the reporter with the lowest identifier will have higher priority in sending the next NACK packet(s). But other receivers keeps the repair process active since the reporter with the lowest identifier may leave the session.

Similarly, when there are duplicate repair packets in response to a NACK packet, only the receiver with the lowest identifier among all repair senders continues to send repair packets, other receivers must stop sending repairs.

With this mechanism, the entity with the lowest identifier among all reporters behaves like a group leader in error report even though there is no static leader in a subgroup. This leadership is dynamic and may change at the next error report. However it is important to make loss repair not solely dependent on another entity which is generally not trustable.

### 3.4.3 Original Sender

Other optimization can still be made at the sender side. Since there is only one source for each data stream, the original sender can respond immediately to NACK packets while other receivers are in waiting state. This can reduce the error recovery time and prevent duplicate repair packets in subgroups where the original sender is present.

In addition, if the scope field of a NACK packet is the session TTL, all receivers must not reply to the NACK packet and give the chance to the original sender to send the repair(s). This will eliminate duplicate repair packets at the session scope level.

The REP scheme will work still better if there is one fixed protocol entity responsible for error control in each subgroup. This means if a unique entity can apply the retransmission procedure of the original sender, for example, the router, the error recovery time and duplicate repairs will get even better.

## 4. Flow and Congestion Control

Flow and congestion control constitutes a vital component of this protocol. First applications need flow control to adjust the transmission rate so that the data flow does not go above the processing capacity of the application. Otherwise packets may be dropped at end hosts due to lack of processing power. Second the network can generally offer limited throughput. Even if there is a single data flow, it should still be adapted to the available network bandwidth (generally limited by the most bottleneck network path) to avoid congestion and achieve better performance. Third when there are several data flows in the network, it is extremely important that one flow does not shutdown the others. Instead the network bandwidth should be fairly shared among them. Besides a data flow should be competitive to ensure the responsiveness for applications. The rate-based control mechanism described here is intended to be friendly with other data flows under flow and congestion control such as TCP. Simulation results and statistics will be presented later in a separate report.

### 4.1 Send and Receive Windows

A data sender maintains a sliding send window. Data packets with sequence numbers falling into the send window are kept in the buffer space. Similar to TCP, the send window represents the maximum number of outstanding packets that a sender can send to the network. In addition the send window determines the number of packets that a sender can go back for retransmission from the current sequence number. Different from TCP, the send window size is generally constant during a session which could be adapted to the transmission rate.

Every receiver maintains a sliding receive window. Received data packets (in order or out of order) are maintained in the buffer space if the difference between the sequence number and the highest sequence number seen

from the sender is lower than the receive window size. The receive window determines the number of packets that a receiver can repair from the highest sequence number maintained by the receiver. Therefore if the expected sequence number is less than the highest sequence number minus the receive window size, the buffer space will not be enough to hold repairs without dropping new packets. Flow and congestion control should prevent that this case occurs.

### 4.2 Transmission Rate

Data transmission rate is kept within a range specified by an application, in form of  $[R_{min}, R_{max}]$ , where  $R_{min}$  and  $R_{max}$  are the lower and upper bounds of the data rate, respectively. If an application does not specify the rate range, the default range could be applied, for example, in [8 kbits/s, 64 kbits/s]. While the specification of rate range requires some experience in network communications, it is not a difficult task in most cases since the range does not need to be very precise.

At start-up, the data rate  $R$  should be increased gradually up to the maximum rate, similar to the slow start in TCP [12]. The initial rate is set to  $(R_{min} + R_{max})/2$ . Then for every eighth of send window data sent, increase the rate by  $0.125R$  if no contrary indications. Under normal network conditions (see below), the data rate will reach the maximum rate after an amount of data corresponding to one and half send window have been sent.

As a closed-loop approach is adopted for congestion control, if the network is partitioned near the sender, no feedback at all will be returned from the network. In this case, it is desirable that the sender keeps the transmission at the minimum rate.

### 4.3 Congestion Control

One of the objectives of congestion control can be viewed as to minimize the number of retransmissions, hence to avoid to overload the network. In a NACK based protocol, congestion control is also very important to ensure the reliability. Without such a control, loss recovery could be impossible since the lost sequence numbers may go out of the sender window.

Flow and congestion control is performed at the sender side according to the feedback information gathered from the network. A network congestion problem can be reflected by the following parameters:

- the information carried in NACK packets, such as the lowest sequence number to be retransmitted and the bitmap. Congestion will cause more packets being lost, thus the number of retransmissions will increase.
- the indications from receivers. Due to local error recovery, additional traffic can be generated in local region that may not be seen by the sender. This can cause local congestion problem which

needs to be reported to the sender.

- the variation of round trip time. Most network congestion will cause output queue full at intermediate routers, thus make the packet transit time longer.

Adaptation based on the round trip time is difficult and may not be adequate. In large groups, it is hard to get the round trip time to every receiver. Suppose that the round trip times are available, the rate can be adapted either individually to each receiver or to the average RTT. In both case, it is hard to determine the right function to apply. In our experiments, the adaptation based on the round trip times can make the data rate either unnecessarily too low or too high.

Our congestion control scheme is based on the loss information contained in NACK packets, the congestion indication from the receivers and the send window size. To avoid unnecessary oscillations of transmission rate, two adjustment operations should not be performed in too small time interval. It is assumed that the adjustments are carried out in regular time intervals, for example, every eighth of send window data sent.

#### 4.3.1 Adaptation to the Loss

NACKs can be used together with the send window to limit the number of outstanding packets in the network. However it is a bad idea that a rate based control mechanism reacts on every single loss as long as it is under a certain threshold. This may lead to useless rate adjustments and make impossible to reach a stable data rate [18]. It is important that a rate based control mechanism reacts on the right moment and on the right signals.

Three parameters can be derived from NACK packets that may be useful for congestion control: the number of received NACKs (in a given time interval); the number of lost packets reported a NACK; the lowest lost sequence number. The first parameter, the number of NACK packets heard from the network in a time interval, may not timely indicate a congestion problem due to NACK suppression and the fact that one NACK packet can report more than one lost packets. Thus the use of this information can be excluded.

In the long term, the total number of lost packets reported by NACKs is roughly the same as the number of retransmissions if the receivers are still synchronized with the data stream and the duplicates are excluded. The overall loss rate  $l$  can be approximately estimated from the following expression:

$$l = P_r/P$$

$P_r$  is the number of retransmissions and  $P$  is the total number of packets transmitted. This equation does not make much sense for short time intervals and high loss rates. Thus the estimated loss rate is not suitable for congestion control, but could be used, for example, to adjust the level of redundancy in the data stream if FEC is used. Another way is to make rate adjustments if the number of

lost packets exceeds a threshold. But it is very hard to fix the threshold such that the adjustments could be too early or too late.

Finally the lowest lost sequence number is a good indication of congestion problem and can be easily used for congestion control, as illustrated in Figure 2. If the lowest sequence number requested for retransmission is going out of the range maintained within the send window, that means that the sender is going ahead too fast and the error recovery will be soon impossible. If the sender does not slow down the data transmission, some receivers will not be able to synchronize with the data flow. Note that the expected sequence number will be generally reported as the lowest lost sequence number by a NACK.

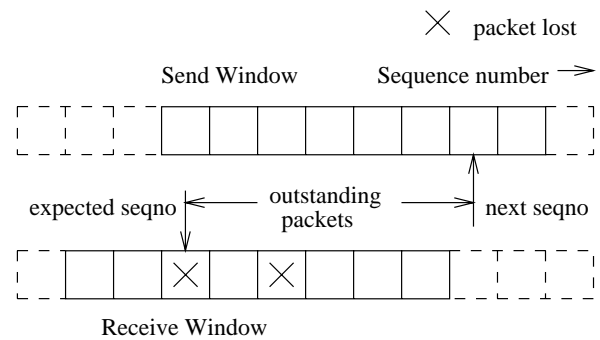


Figure 2: Sequence number space

To respond to this problem, a sender should calculate the difference between the current sequence number in transmission and the lowest lost sequence number contained in a NACK packet. The rate adjustment is determined as follows:

- If the difference is greater than a quarter of send window size, the transmission rate is reduced to  $0.75 * R$ .
- If the difference is greater than one third of send window size, the transmission rate is reduced to  $0.5 * R$ .
- If the difference is greater than half send window size, the transmission rate is reduced to  $0.25 * R$ .

This policy allows to respond enough quickly to congestion problems. Here we can see that in LRMP, NACK packets have a similar role in congestion control to ACKs in TCP.

#### 4.3.2 Adaptation Based on the Congestion Indications from Receivers

As described above, additional flow and congestion control is needed for the following reasons. First a receiver may not be able to synchronize with the data flow due to the fact that the application can not timely process received data. That may cause receive buffer overflow. Second the traffic introduced by local error recovery may cause local congestion problem. Congestion indication from receiver is intended to report possible local flow and congestion problem that is useful for congestion avoidance. It was first

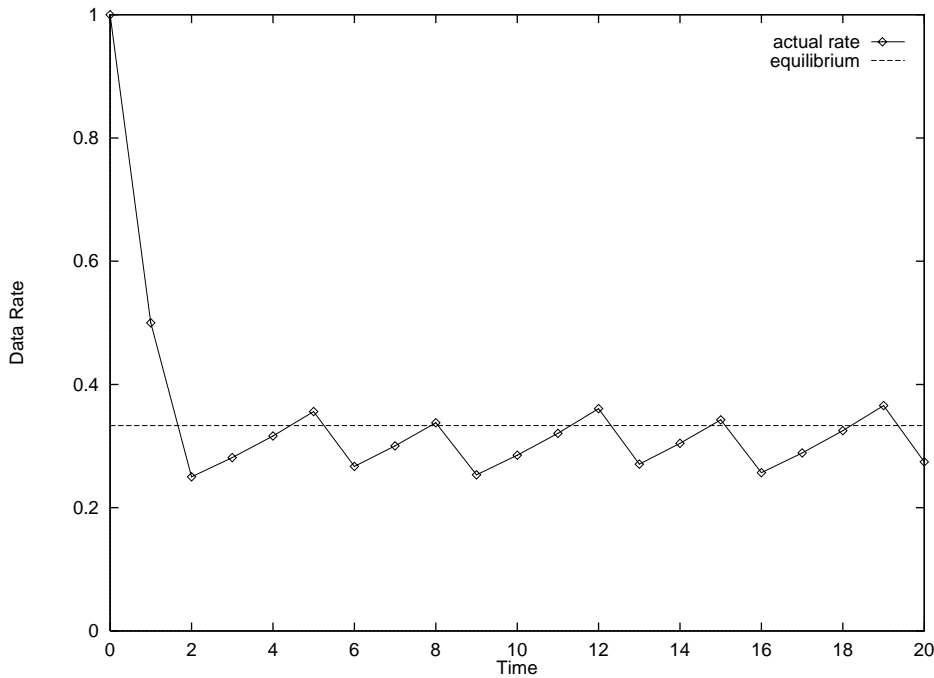


Figure 3: Behavior near the equilibrium

proposed by [14].

The congestion flag in receiver report is used for this purpose. It is set by a receiver if the difference between the last sequence number delivered to the application and the highest sequence number heard from the sender is greater than half the receive window (buffer) size. This means that the number of outstanding packets is going to be too high so that the receiver will not be able to repair loss due to the lack of buffer space. It is thus desirable for the sender to reduce the transmission rate in this case. Upon reception of a congestion indication, a sender will reduce the transmission rate to  $0.5 \cdot R$ .

#### 4.3.3 Getting to the Equilibrium: Rate Increase

If no decrease condition is true, as described above, and one eighth of send window time has elapsed since the last decrease or increase, the rate is increased by a factor of 0.125 if it is less than the upper bound  $R_{max}$ , otherwise the rate is set to  $R_{max}$ . With this control policy, one decrease to the half rate requires roughly six increases to reach the original rate.

Figure 3 shows the behavior of the congestion control scheme near the equilibrium. Time is in units of congestion check interval. It is assumed that the loss rate will increase above the equilibrium and no loss under the equilibrium. It is also assumed that the data rate is enough high to quickly cause the rate adjustment if the packet loss is present. From this figure, it can be seen that the data rate is roughly in a range of 12 percent above and 25 percent below the equilibrium in the steady state. Small oscillations are useful to explore addition bandwidth. In practice, the rate oscillation

will be more slow since some response time is needed to decrease the rate.

## 5. Other Features

This section gives a brief description on some additional features which are intended to make the protocol more robust and thus increase the application range.

### 5.1 Per Packet Selective Reliability

LRMP provides a per packet selective reliability, i.e., application data can be sent either reliably or in a best effort way. The reliability (a boolean value) must be indicated by the upper layer when sending data to LRMP. This allows an application to use the same transport protocol to send reliable and unreliable data. Unreliable data is less costly than reliable data in terms of resource usage. Proper use of unreliable data can also alleviate the burden to manage many sender states at receivers.

Unreliable packets are not subject to flow control at the sender side and not subject to sequence control at the receiver side. They can be considered as out-of-band packets which are delivered with priority but without guarantee of arrival at receivers. Due to this particularity, unreliable packets are expected to be sent occasionally by an application. When an unreliable packet is received by a receiver, it is delivered immediately to the application<sup>1</sup>. In addition a receiver does not need to cache such packets in the buffer space.

## 5.2 Selective Receiver Report

LRMP includes a selective feedback mechanism allowing to monitor the quality of service at receivers. Receiver reports are useful for both sender and receivers. First they are used for estimation of the session population. It could be interesting to know how many participants are actually in a session. Based on this estimation, the sender could decide, for example, to go on or stop the session.

Receiver reports are also used to get quality of service feedback from receivers, such as the loss rate, the number of packets and octets received. The feedback can be used to do flow and congestion control and adjust other transmission parameters, for example, the redundancy in data streams. In addition they allow to know where there is a serious reception problem.

The idea is that the sender tells the receivers whether and how to send receiver reports, instead of every receiver systematically sending reports which is not scalable to large groups. LRMP supports two ways to get receiver reports:

- a receiver sends once a report with a specified probability. This probability could be adapted by the sender to the number of receivers in a session.
- a receiver sends a report periodically, but the total traffic is limited to a small percentage of the total bandwidth, like RTP[19].

A data sender sends a receiver report selection packet for this purpose. The selected receivers may be a list of known entities or all receivers (via a wild card identifier). If there are multiple receiver reports to send, a receiver can multiplex several receiver reports in one outgoing packet.

## 5.3 Forward Error Correction

Forward error correction is supported as an independent optional module. FEC is based on the principle of adding some redundant packets to the transmitted data stream so that the original data packets can be recovered without retransmission up to certain loss. This can avoid generating feedback packets (NACK) [16] [17]. Proper use of FEC technique can substantially improve the scalability and is particularly advantageous over asymmetric network links where no or little bandwidth is available on the feedback channel.

While FEC is attractive, its use is not simple. If there is no loss or very little loss, the redundant packets will add unnecessary overheads and cause inefficiency. If the loss is too important, lost packets will not be recoverable without retransmission. The level of redundancy and the transmission scheme need to be adapted to the loss rate and pattern. LRMP only prescribes a packet encoding mechanism, no restriction is placed on how FEC packets should be transmitted as long as they allow proper decoding at receivers.

- 
1. This can also be done by inserting the received packet at the head of output queue so that the application can process it at next read.

Since the use of FEC is optional, the FEC integration scheme described here allows that implementations without FEC support can correctly receive data stream by ignoring FEC encoded packets.

In LRMP, a special packet type is defined for FEC packets which can be sent either with the data stream or as a separate stream. Each FEC packet is self-contained and refers to a set of original data packets. The decoding can be performed before the whole group have been received. More specifically, a FEC packet contains the following data fields:

- the block size (n), i.e., the number of original data packets plus the number of FEC packets in a block.
- the number of FEC packets (k) in a block.
- the relative FEC packet number in a block.
- a base sequence number from which FEC packets are calculated.
- a spacing number between the original data packets.

Let the base sequence number be b and the spacing parameter be s. The sequence numbers of data packets in a block are composed of:

$$b, b + s, b + 2s, b + 3s, \dots, b + (n - k - 1)s$$

FEC packets that have the same base sequence number belongs a same block. The spacing parameter is introduced to increase the efficiency of recovery in case of burst loss. Larger spacing parameter requires more buffer space at receivers. In addition, if large block size is used, receivers that newly joined the session may need longer time to be able to start decoding FEC packets.

It is worth noting that the original data packets do not have reference to FEC packets. This yields a protocol independent FEC integration scheme, i.e., no matter how the data stream is encoded and transmitted, redundant data can be added independently.

## 6. Implementation

LRMP has been implemented in Java as a reusable library. This library has experienced substantial improvements from its first version and will continue to evolve. As a reusable library, a stable API is extremely important. Fortunately LRMP has an API enough stable which has undergone little changes through different versions. Java is appropriate up to a certain degree of performance. As expected, the library can offer acceptable performance for bulk data transfer. Current version does not have FEC implementation.

Many lessons have been learned from the implementation and tests. As the data transmission is rate controlled, the transmission interval has an important role in NACK report algorithm which is intentionally missed in the above presentation for clarity. When an entity receives a NACK, the repair packets will be sent generally in regular time

# of receivers	NACKs	Repairs	Dup. NACKs	Dup. Repairs
1	998	984	0 (0.0%)	0
2	1047	984	43 (4.1%)	2
4	1203	984	193 (16%)	0
8	1286	1004	252 (20%)	1
16	1432	996	406 (28%)	3

Table 1: Statistics without local recovery

intervals. So after a NACK is sent, a receiver needs to wait some additional time to send the next NACK since the repair packets may be queued for transmission. Otherwise the number of duplicate NACK packets will significant grow under heavy loss. Security is a major concern in multicast data transfer over Internet, in particular, when local error recovery is used. The minimal requirement (for public groups) is the authentication, i.e., to be sure that the data comes from the right place. If the information is addressed only to the intended people, encryption is highly recommended. At the protocol level, implementations should protect them-selves from malicious attacks or misbehaving of a few receivers. It was observed that unknown traffic sometimes flows in the test sessions, perhaps due to a conflict of group address. In general, a receiver is not trustable in a loosely coupled group. Care should be taken to prevent a particular receiver blocks the data transmission for the whole group.

There are several options for implementations. The most simple implementation may only implement reception functions and depends on other receivers to recover packet loss. An implementation can also forbid the error recovery at the session scope level in very large groups and unicast the repair packets to a receiver located in a particular region, the later then multicasts the repairs to the subgroup. For small groups, an implementation can make use of receiver reports as positive ACKs to acknowledge the received packets.

LRMP has been used in a number of applications. Examples include chat, whiteboard, file transfer, software distribution, sharing of data objects, distance learning, etc. That allowed to get useful feedback and further improve the implementation. More information on the deployment can be found at <http://webcanal.inria.fr/lrmp/>.

A number of measurements have been carried out, among which two typical ones are presented here. For these measurements, a sender sends data to a variable number of receivers at a constant rate, 64 kbits/s. Each receiver runs at a different host. Random loss is artificially introduced at around 10% on the failed link. The number of transmitted data packets (excluding repairs) is 10000.

The first measurement uses the configuration shown in Figure 4. All receivers are located at the same site and encounter the same packet loss. There is no subgroup, errors are reported directly to the sender. The purpose of this

measurement is to see if the basic error recovery mechanism works well, including sending NACKs, duplicate NACK suppression, and sending repairs in the simplest case.

The statistics were collected at the receiver side and are shown in Table 1. The number of repair packets is about 10% of the total number of data packets (corresponding to 10% loss rate). The number of NACKs (including duplicates) increases with the number of receivers, but the increase is not significant with regard to the increase of the number of receivers. The duplicate repair packets are very rare since only the original sender can send repair packets. As the packet loss is uncorrelated, i.e., individual, roughly one NACK corresponds to one repair packet. In case of burst loss, we observed that one NACK can repair on the average more than six lost packets. From the statistics, we can see that the duplicate NACK suppression mechanism works enough well.

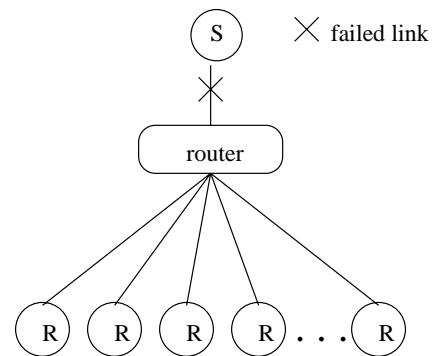


Figure 4: Configuration 1: homogeneous receivers

The second measurement uses the configuration shown in Figure 5. There are two level subgroups: a subgroup (the dashed circle) contains all receivers and the whole group (the solid circle). Only the receivers behind the failed link (between router 2 and router 3) encounter packet loss, the receivers which do not have packet loss will send local repair packets. The receivers behind router 3 are of the same number as those behind router 4. Thus contention will exist in both error report and sending repairs. The purpose of this measurement is to see the behavior of the random expanding probe scheme in local error recovery.

The statistics are given in Table 2. In all cases, about 85%

# of receivers	Local/Total NACKs	Local/Total Repairs	Dup. NACKs	Dup. Repairs
2	808/911	832/990	0 (0.0%)	0
4	914/1005	853/992	83 (8.3%)	7
8	966/1080	892/1001	101 (9.4%)	45
16	976/1092	862/991	151 (14%)	51

Table 2: Statistics with local recovery

of lost packets were repaired in the subgroup. As there is one repair try in a subgroup for every loss, if repair packets or NACKs are lost, further repair tries are carried out in the whole group. Thus a number of NACKs went to the whole group. The number of duplicate NACKs and that of duplicate repairs increase with the number of receivers, but they are reasonable. Again, in case of burst loss, the REP scheme can behave even better due to the fact that one NACK can report a number of adjacent lost packets and a receiver can behave as the dynamic leader in a subgroup.

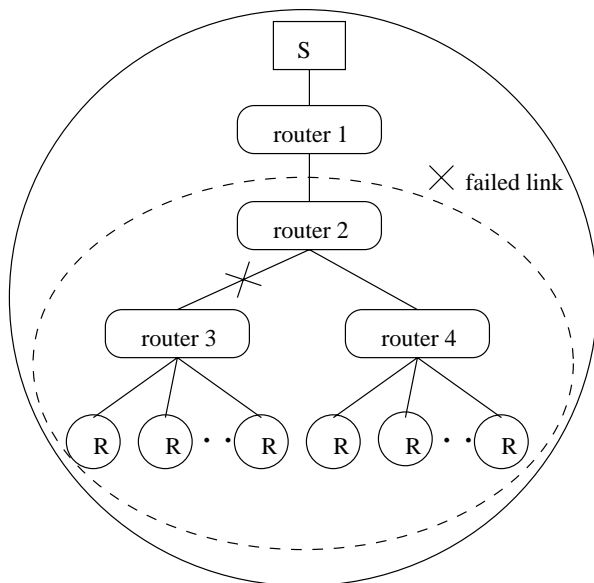


Figure 5: Configuration 2: heterogeneous receivers

## 7. Conclusions

LRMP is suitable for reliable multicast bulk data transfer. Control schemes are carefully chosen to achieve this purpose. The new local error recovery scheme is particularly appropriate for loosely coupled groups. To some extent, it can behave as if there are group leaders. Congestion control is performed using NACK packets and congestion indication from receivers. But one should be very careful in applying any congestion control scheme in multicast data transfer since one receiver may block the data transmission for the whole group. In other words, some tolerable range should be defined to exclude extreme cases. In LRMP, applications are allowed to choose the congestion control scheme that meets best their needs. Further work includes

the implementation of FEC and the simulation of the congestion control mechanism to see what is the fairness with TCP data flows.

## References

- [1] S. Deering, "Host extensions for IP multicasting", RFC 1112, August 1989.
- [2] J. Chang and N.F. Maxemchuk, "Reliable Broadcast Protocols", ACM Transactions on Computer Systems, 2(3). August 1984.
- [3] S. Armstrong, F. Freier and K. Marzullo, "Multicast Transport Protocol", RFC 1301, February 1992.
- [4] R. Talpade and M. H. Ammar, "Single Connection Emulation (SCE): An Architecture for Providing a Reliable Multicast Transport Service", Proceedings of the 15th IEEE Intl Conf on Distributed Computing Systems, Vancouver, June 1995.
- [5] K. Robertson, K. Miller, M. White and A. Tweedly, "StarBurst Multicast File Transfer Protocol (MFTP) Specification", Internet Draft, Work in Progress, July 1998.
- [6] B. Whetten et al., "The RMTP-II Protocol", Internet Draft, Work in Progress, April 1998.
- [7] T. Speakman, D. Farinacci, S. Lin, and A. Tweedly, "Pragmatic Group Multicast (PGM) Transport Protocol Specification", Internet Draft, Work in Progress, Jan. 1998.
- [8] Markus Hofmann, "Enabling Group Communication in Global Networks", Proceedings of Global Networking'97, pp321-330, Canada, June 1997.
- [9] S. Floyd, V. Jacobson, S. McCanne, C. Liu, L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", ACM Transactions on Networking, Nov. 1996.
- [10] M. Mathis et al., "TCP Selective Acknowledgement Options", RFC 2018, April 1996.
- [11] Tie Liao, "Light-weight Reliable Multicast Protocol Specification", available at [http://webcanal.inria.fr/lrmp/lrmp\\_v1.html](http://webcanal.inria.fr/lrmp/lrmp_v1.html).
- [12] V. Jacobson, "Congestion Avoidance and Control", Computer Communications Review, vol.18, pp314-329, August 1988.
- [13] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet", submitted to IEEE/ACM Transactions on Networking,

- Feb. 1998.
- [14] K.K. Ramakrishnan and R. Jain, "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks", *ACM Transactions on Computer Systems (TOCS)*, Vol. 8, No. 2, pp 158-181, May, 1990.
  - [15] R. Yavatkar, J. Griffioen, and M. Sudan, "A Reliable Dissemination Protocol for Interactive Collaborative Applications", *Proc. of the ACM Multimedia '95 Conference*, November 1995.
  - [16] L. Rizzo, "Effective erasure codes for reliable computer communication protocols", *ACM Computer Communication Review*, Vol.27, no.2, April 1997.
  - [17] J. Nonnenmacher, E. W. Biersack and D. Towsley, "Parity-Based Loss Recovery for Reliable Multicast Transmission", *Proc. of SIGCOMM '97*, Sept. 1997.
  - [18] Todd Montgomery, "A Loss Tolerant Rate Controller for Reliable Multicast", NASA technical report: NASA-IVV-97-011, August, 1997.
  - [19] Henning Schulzrinne, Stephen Casner, Ron Frederick and Van Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.