Fully Reliable Multicast in Heterogeneous Environments

José F. de Rezende[1], *Andreas Mauthe*[2], *Serge Fdida*[1] *and David Hutchison*[2]
[1] *Laboratoire MASI - CNRS*
Université Pierre et Marie Curie
4, Place Jussieu - 75252 Paris Cedex 05
France
[2] *Computing Department*
Lancaster University
Lancaster LA1 4YR
UK
email: {rezende,fdida}@masi.ibp.fr
email: {andreas, dh}@comp.lancs.ac.uk

Abstract

Multicast (1:N) is now supported by a number of networks and communication protocols. A problem in this context is how to provide fully reliable data transmission to the receiver group (or specified sub-group) in a heterogeneous environment. To achieve this an algorithm is required which is independent of any underlying network and multicast scheme. In this paper we present such an algorithm which provides fully reliable data transfer over Internet style networks as well as over ATM networks. The algorithm was initially inspired by mechanisms proposed for XTP 4.0 and proves to work well using XTP 4.0 protocol mechanisms. To evaluate its feasibility and performance it was tested with a series of simulations and implemented over MBone. The results presented in this paper show that it can provide fully reliable multicast to a relatively large number of receivers on top of different networks, protocols and protocol architectures.

1 Introduction

Multicast communication (i.e. the communication between a single sender and multiple receivers) raises various new issues and problems related to data transmission and reliability. Different degrees of reliability are required for different applications. Full reliability is for instance needed for the transfer of a file to a group of users. In this case all receivers have to receive a correct and complete copy of the data. An efficient method to provide this kind of reliability while still fully exploiting the advantages of multicast communication (e.g. bandwidth saving) is the key issue in the design of reliable multicast protocols.

In a heterogeneous communication environment a number of different networks are connected ranging from LANs (e.g. Ethernet) to Internet style WANs and high-speed networks such as ATM. Whereas with Internet style multicast every member of the group receives every message sent to the group, with point-to-multipoint multicast (as proposed for ATM) there is only one designated sender from which the members of the group can receive. Any algorithm designed for a heterogeneous environment has to take this into account.

Another issue which is frequently addressed in this context is the problem of slow receivers that jeopardise the success of the data transmission because of their inability to process data at a given rate. Such receivers can slow down the communication considerably by sending delayed acknowledgments and requesting frequent retransmissions [1]. A flexible method to deal with this problem is required. A new concept in multicast communication is that of mandatory receivers or a minimum number of receivers that have to be present in order to consider the communication successful [2]. It should be possible to specify a (sub-)set of receivers that are essential for the success of the communication and hence determine its outcome. Others, less important participant can also have less strict reliability requirements. Hence they might not have to be considered as far as error recovery is concerned.

The Fully Reliable Multicast (FRM) algorithm for end-to-end multicast communication introduced in this paper is motivated by our work on multipeer communication services in a heterogeneous network environment [3, 4]. This algorithm provides full reliability to a receiver group (or mandatory sub-group) independently of the underlying network. A list-based scheme using positive acknowledgments is employed to keep track of the global state of the multicast communication. Retransmission of lost or corrupted data is used for error recovery. Conditions can be specified under which slow receiver that can not keep pace with the data transmission are ejected to keep the QoS acceptable for the rest of the group. Main design principles of the algorithm are independence, compatibility to various protocols and protocol architecture, and efficiency. The FRM algorithm was initially developed as part of an independent multicast transport service which can amongst others operate over XTP 4.0 [3]. The algorithm proved to be ideally suited for this kind of environment. To evaluate the algorithm it was implemented on top of MBone and analysed using a series of simulations.

This paper is organised in six sections. Section two discusses reliable multicast communication including proposals of different methods and algorithms to provide reliable multicast in different environments. In section three the FRM algorithm is introduced and described in detail. The results of a performance analysis of the algorithm is presented in section four. Subsequently in section five the implementation of the algorithm over MBone is introduced and discussed. Finally, section six gives conclusion and outlines some future research issues.

2 Reliable Multicast Communication

Multicast communication is a transmission mode which now is supported by a variety of local and wide area networks. Many group applications require reliable data transfer to all receivers. To provide data transfer reliability in an heterogeneous environment, an algorithm is necessary which can easily be supported over ATM style networks as well as over meshed nets like the Internet. In contrast to the distribution group model supplied by IP/Multicast (multicast group abstraction), ATM [5] provides unidirectional point-to-multipoint connections where a multicast tree is created with a single sender as the root node and receivers as leaf nodes (1-N multicast tree). In this model the receivers do not have any knowledge of each other nor is there any communication among them. This model of communication can be regarded as a subset of the more general model proposed by IP/Multicast where everybody can send to the group address and every member is receiving messages addressed to the group. The distribution model of IP/Multicast could actually be achieved at the ATM layer by using a multicast server [6]. However, the complexity of this approach negates the benefits gained by the simplicity of the ATM concept. It also may prevent the users from utilising the QoS capabilities provided by ATM networks. Hence we concentrate on multicast communication with a single source which in our view is much more common than the multisource case. Further, in our research we found that most algorithms specifically designed for the IP environment are, in general, not operational in an ATM-like network.

Reliable multicast communication entails a range of new problems affecting end-to-end mechanisms. For instance loss and error recovery is different from traditional unicast. An entirely new task is group membership management. Main problems in this context are receiver group integrity and QoS maintenance.

In order to achieve full reliability the sender has to keep a copy of a transmitted packet until all receivers in the receiver group have positively acknowledged its reception. In this case an algorithm that keeps track of the state of all receivers is required. This is usually done by using a list-based group management scheme. List-based loss recovery algorithms offer reliable multicast by tracking the explicit membership of the receiver set and by providing retransmission for any lost or corrupted data. Hence full reliability is ensured for the whole receiver group.

Compared to the unicast case it is much more difficult in multicast to determine the success or failure of the communication. Multicast communication introduces a new problem related to the integrity of the receiver group. Group integrity refers to conditions according to which the data transfer is deemed successful. These conditions are called communication integrity conditions. They give the number and/or identity of recipients that have to be *present* and/or to receive the transmitted data correctly. The following conditions are considered: *key-member* (identity) and/or k (number), *all*, *quorum*[1].

Another important issue in this context is the problem of misbehaving receivers. A slow receiver can jeopardise the communication by not responding in time or requesting too many retransmissions. In general, if a receiver falls behind the others in the sequence space, the sender has to keep up with the pace of this slow receiver. This causes a decrease of the end-to-end throughput noticed by all receivers. To avoid this degradation on the performance, a condition can be specified under which a receiver is forced to leave the receiver group because it jeopardises the communication. This condition is called *ejection condition*.

The problem of reliability in multicast communication was recently addressed by a number of authors [7, 8, 9, 10]. The RAMP algorithm introduced in [7] guarantees reliable and orderly delivery to all multicast recipients using two different modes. In the burst mode the sender requires acks for each data burst. With the idle model the sender keeps sending all the time. In this mode, a negative acknowledgment scheme is used where a receiver notifies a sender immediately upon detecting a gap within the received sequence. It maintains explicit group membership at the sender and therefore does not scale very well for large groups. RAMP uses a simplex model for multicast communications.

RAMP is the only model that strictly follows the single sender multiple (passive) receiver paradigm. The algorithm described in [8] (called a framework for scalable reliable multicast (SRM)) was especially designed for wb (white-board application). Scalability is achieved by a receiver-based reliability scheme in conjunction with Application Level

[1]*quorum* and *all* refer to the number of positive replies during connection establishment.

Framing (ALF). All receivers are able to retransmit since all data packets (application data, retransmission requests and all other data) are always distributed to the entire group. SRM requires that the application is aware of the operational environment and that it is able to restore the data for retransmission. Further, an IP multicast distribution model is assumed. SRM is strongly focused on the specific environment it was designed for and on the kind of application it aims to support. RMTP [9] achieves reliability by using a packet-based selective repeat retransmission scheme. The ACK-implosion problem is avoided using a multi-level hierarchical approach, in which the receivers are grouped into a hierarchy of local regions, with a Designated Receiver (DR) in each local region. DRs cache received data and respond to retransmission requests of the receivers in their local regions. It assumes a group distribution model as provided by IP/Multicast (receiver-based approach). In [10] an hierarchical approach, named Local Group Concept (LGC), is used as well. Here, global multicast groups are divided into separate subgroups. The receivers which pertain to the same subgroup should be located in close physical vicinity. In each subgroup a Local Controller is responsible for collecting status information from its receiver group and is informing the sender with a single composite message to prevent source implosion. The Local Controller is also responsible for retransmission within the local group. Data that has to be retransmitted is obtained either from the local receivers or the original sender. All of the above described algorithms assume that it is possible to retrieve data for retransmission from the receivers as well as from the sender. To exploit their full potential it would be necessary to have knowledge about the underlying network. None of the introduced algorithms so far addresses the problem of misbehaving receivers and the consequences of this for the multicast transmission.

The FRM algorithm described in this paper provides fully reliable data transfer and group membership management for multicast communication. It defines proper mechanisms for loss recovery, integrity conditions validation and throughput monitoring and maintenance by using ejection conditions. All these mechanisms rely on the use of a list-based group management scheme which is akin to the one proposed by XTP 4.0 [11].

3 The Fully Reliable Multicast (FRM) Algorithm

The algorithm is part of a protocol independent transport service and as such uses and shares information with other components which are gathered continuously during an ongoing communication. However, the main design goal was *independence*, i.e. the algorithm should be operable with all kinds of end-to-end protocols and services in different environments.

3.1 Algorithm Description

Initially a message is send to a group address announcing the set-up of a communication among members of the group. A receiver group list is created from all positive replies to this request. During the communication, the sender gathers status information in regular intervals from the receiver group to update this list. To do this the sender periodically multicasts a *control message* to the receiver group to which each receiver immediately responds by unicasting an *echo message* to the sender. To correlate control and echo messages, each control message is identified by a value (*sync*) which is incremented each time a new control message is sent. Each receiver must copy the *sync* value from incoming control messages into the echo messages returned to the sender. The control message is used to solicit the status of the receivers and the echo message is used to transfer this status information to the sender. Thus, the current state of the communication is determined by the sender according to the replies of the receivers.

The status information carries the reception status of the receiver which is represented by two variables called *rseq* (received sequence number) and *dseq* (delivered sequence number). The *rseq* contains a value one larger than the largest monotonic sequence number of the data packet received without error. This information is used to trigger a retransmission if necessary. The *dseq* value gives the sequence number of the last data message delivered to the user plus one. Buffers at the sender are released using the former value. Further information exchanged in the status information is used to update state variables required for such mechanisms as flow- or rate-control, maximum round-trip delay estimation, etc. The figure 1 depicts the basic structure of the FRM algorithm.

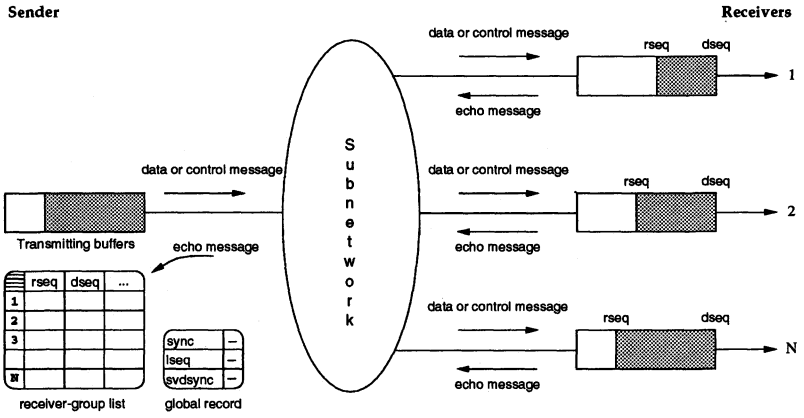

Figure 1: FRM Algorithm Structure.

Each time the sender multicasts a control message it sets a timer (*wtimer*). The value of *wtimer* determines the time interval during which the sender gathers the replies to the control message. When the timer expires the sender gets the lowest *rseq* (*minrseq*) and *dseq* values from all replies (*mindseq*). It starts retransmission if needed and releases output buffers accordingly. Further, it estimates the maximum round-trip time and updates the variables used in the flow- and rate-control mechanisms. Subsequently a new control message is sent to start this procedure again.

If a response from a receiver does not arrive for some reason during the *wtimer* interval, a synchronising handshake is initiated by the sender. During the synchronising handshake data transmission is stopped and the sender requests status information from all receivers. The sender tries for a user-specified number of times to get replies from all receivers. If a receiver does not reply at all during this period, this receiver is considered *dead* and is dropped from the receiver group. At each new request, the sender increases the period in which it collects the replies.

Because of the effects that the estimate maximum round-trip time has on the algorithm's performance, its value has to be estimated carefully. In our model the round-trip time for each receiver is calculated by timestamping the control message with the sending time. Each receiver must copy this time value from the control messages into the echo messages returned to the sender. When the sender processes the echo messages, it calculates the round-trip time for each receiver. Note, the round-trip time includes the network round-trip time, the waiting time of the echo message in the input buffers and the processing time of this message. When the worst case round-trip time ($maxRTT$) is smaller than the current estimation ($smaxRTT$), the worst case round-trip time is smoothed, i.e. it assumes a lower value according to the equation below. Otherwise, the current estimation assumes this new value. Therefore, the estimation of the maximum round-trip time for each control message is done according to the following algorithm:

```
if ($smaxRTT > maxRTT$)
    $smaxRTT := smaxRTT + (maxRTT - smaxRTT)/16;$
else $smaxRTT := maxRTT;$
```

Network delay and receiver load are the main factors influencing the value of the maximum round-trip time. The FRM algorithm is sensitive to changes of this value. In general, the maximum round-trip time estimation adapts to the capabilities of the slowest receiver. This can cause problems in a heterogeneous environment where its value might be very large for some receivers (they might for instance be connected via satellite) compared to the rest of the receiver group. This can result in a very poor performance even if the number of receivers is relatively small.

The value of the *wtimer* has to be large enough to allow all receivers to reply to the related control message before it expires. The *wtimer* is calculated using the following equation:

$$wtimer := k * smaxRTT, \text{ where } k > 1$$

The multiplication factor k in the above formula is used for contingency reasons. This factor can be changed, for instance in an unstable environment an increase of this factor ensures that the sender does not enter in a synchronising handshake too often. Thus, it ensures that all replies will be received in the *wtimer* interval. The larger the value of k the higher is the probability that all replies will be received in this interval. Hence unnecessary synchronising handshakes are avoided. However, the higher value of k has a negative impact on the throughput of the multicast communication because the sender transmitting buffers cannot be released as frequently as with a smaller value for k.

The loss recovery algorithm uses a global record which is kept at the sender. This record is a data structure containing the current state of the communication in terms of data (re)transmission. This global record keeps the *sync* value of the last control message sent, the value one larger than the largest sequence number of the data packet ever (re)transmitted just before sending the control message (*lseq*) and the *sync* value of the control message during which data packets were previously retransmitted (*svdsync*). After collecting the status information of the receiver set and by using the information contained in the global record, the sender can decide if retransmission is needed or not while avoiding redundant retransmissions. If retransmission is needed, the packet will be multicast to the whole group. A go-back-n scheme for retransmission is used in the first version.

3.2 Group Integrity

The FRM algorithm provides mechanisms to check communication integrity conditions. A strong notion of group integrity is provided by the FRM algorithm since it ensures that either all mandatory recipients have received the data correctly or that it is still available. Hence, integrity conditions are only breached when a mandatory receiver is not present any longer. A situation where the group integrity is momentarily not valid can only occur between two control messages.

Integrity conditions are validated every time the sender discovers that a receiver is not present any more. This may be the case when a receiver decide to leave or is forced to leave the receiver group. Further, the sender also validates the integrity conditions at the end of each *integrity interval* which is determined by the current *wtimer* value. Every time the sender detects that a mandatory receiver is not present any more, the sender will release the communication since the integrity conditions are not satisfied any longer.

3.3 Ejection

In the FRM algorithm the user may specify a minimum degree of end-to-end throughput in order to consider the communication successful. An ejection condition based upon the release rate throughput at the sender (*relthr*) is defined for this purpose. This parameter indicates the mean number of buffers released during a certain time interval (*ejectcdtintv*). Note that buffers are only released when all receivers have correctly received the associated messages. Thus, the user specifies a minimum release throughput (*minrelthr*) as ejection condition. When the calculated release throughput falls below the *minrelthr*, the ejection condition is satisfied. The slowest receiver(s) (if any can be determined) is then forced to leave the receiver group. Network problems which affect all receivers or a sub-set of receivers may be the cause for the degradation in the end-to-end throughput as well as problems in the receivers themselves. Hence, the main issue is how exactly to determine the cause of the ejection situation. Thus, before ejecting any receiver(s), it has to be ensured that the offending receiver(s) is the sole reason for the poor performance.

To establish which receiver is causing the problem, additional information is kept in the receiver group list. The *slowest receiver* is the one which acknowledges the lowest sequence number (*dseq*) for a certain consecutive number of times. In addition, the sender calculates the mean release throughput of each receiver (*mrelthr*). This value is calculated based on the replies to each control message. It gives the mean number of buffers that would be released if only this receiver took part in the communication. When the ejection condition is met, the sender has to decide which receiver(s) have to leave the communication. To do this it checks if a *slowest receiver* exists. In this case, the sender ejects this receiver. Otherwise, it compares the *mrelthr* of each receiver against the *minrelthr*. All receivers that have a *mrelthr* smaller than the *minrelthr* are forced to leave the communication. This strategy ensures that the sender only forces the offending receiver(s) to leave.

The figure 2 shows the influence of the ejection condition when applied to multicast communication using the FRM algorithm. In this example we consider ten receivers of which two can not keep up with the rest and fall behind the minimum release throughput. The minimum release throughput specified by the sender is 180 bytes/s. As soon as the mean throughput drops below this value, the sender determines the slow receivers and ejects them. This action shows the anticipated effect, i.e the communication recovers and returns to the initial better performance. Hence it prevents that slow receivers jeopardise the communication by slowing it down beyond an in advanced agreed threshold. In comparison, the curve without ejection condition shows that the sender synchronises with the slow receivers.

3.4 The algorithm

For each echo message received the sender updates the receiver-group list and whenever the *wtimer* expires the sender runs the FRM algorithm. A complete description of this algorithm including all its procedures is presented in figure 3.

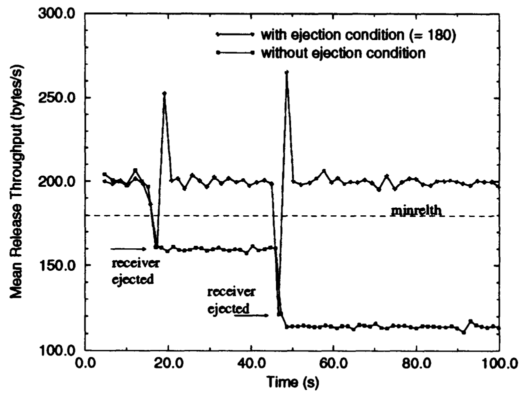

Figure 2: Ejection Condition Performance.

4 Performance Analysis

To evaluate the FRM algorithm and the influence of different scenarios on its efficiency and performance, it was analysed using a test suite of simulations. This section introduces and discusses the simulation results.

The simulator package used is $QNAP_2$ [12]. For the simulation a multicast communication model with a variable number of receivers (N) was developed. Both sender and receivers have a restricted buffer capacity of B packets. A constant data packet size S is assumed. The processing times for control (C), data (D) and echo (E) packets are as follows: C, $E = x$, and $D = 3*x$. The execution time of the loss recovery algorithm is x. The underlying network is ATM characterised by a transfer rate of 155.5 Mb/s and a packet error rate of L. The distance between one sender and the i-th receiver is determined by a propagation delay of D_i, where D_i is uniformly distributed in the interval $[D_{min}, D_{max}]$. The sender generates data traffic with the constant time span $SData$ between two subsequent packets, and the receiver user consumes data with a constant rate of $TSData$. Table 1 comprises all above described simulation parameters.

The main parameter used to evaluate the performance is the mean response time for data packets (MRT). The MRT is the mean time elapsed between the sender application submits a data packet for transmission and its delivery at the receiver side. The loss recovery algorithm is analysed according to the results obtained for this performance parameter.

The algorithm is evaluated using the MRT as a function of N. In these simulations the propagation delay is uniformly distributed between [10,15] ms. The processing time for the packets are obtained using $x = 0.5$ ms. In addition, the following parameter values are assumed: $B = 32$ packets, $S = 2000$ bytes, $SData = 10$ ms, $L = 10^{-5}$, $TSData = 0.5$ ms.

Figure 4 shows the scalability of the algorithm. The x-axis shows the number of receivers and y-axis shows the mean response time. The FRM algorithm presents a steady increase of the MRT. After a certain number of receivers the sender is no longer able to handle them efficiently. Therefore, the algorithm does not scale well when the number of

```
if (all receivers replied) {
    update_variables; /* mindseq, minrseq, maxRTT, ... */
    calculate_smaxRTT;
    release_output_buffers; /* until mindseq */
    if (ejectcdtintv) {
        check_ejection_condition;
        if (eject_cdt_satisfied) leave_offending_receiver(s);
    }
    check_comm_integrity_conditions;
    if (!comm_int_cdt_satisfied) release_communication;
    if (minrseq < lseq) {
        if (sync > svdsync) {
            /* retransmission is not redundant */
            retransmit_buffers; /* from minrseq */
            svdsync := sync + 1;
        }
    }
    sync := sync + 1;
    send_control_message;
    set wtimer; /* k * smaxRTT */
}
else synchronising handshake;
```

Figure 3: FRM Algorithm Description.

receivers reaches a certain threshold (in our example 150 receivers) but further optimisation to allow a large number of receivers be accommodated are currently being investigated.

4.1 Influences and Interfering Factors

The algorithm is influenced by different external factors related to the heterogeneity of the network and receiver group, the topology, error rate, etc. However, the main factors influencing the mean response time with our list-based FRM algorithm are management overhead and network error rate in conjunction with buffer restrictions in the sender. The management overhead, usually referred to as source implosion effect, leads to a steady increase of the MRT because with every new receiver the sender has to process more control packets and to manage a growing number of receivers. During this time it is not able to send any data. In the worst case the sender is only dealing with management tasks and is thus hardly able to send any application data.

However, buffer restrictions also considerably influence the performance of the FRM algorithm. Figure 5 shows that with less buffer space in the sender the MRT increases suddenly and very rapidly. With an increasing number of receivers the maximum round-trip time is increasing because the increased management overhead adds to the estimate maximum round-trip time. The sender can only send as long as there is enough buffer space left. Once all buffers are full the transmission has to be stopped until all receivers have acknowledged the correct reception of the data. With an increased round-trip time the frequency with which buffers are freed becomes lower. Hence, transmission of data has to stop more often because of insufficient buffer space. This in turn has a negative effect on the encountered MRT.

parameters	*meaning*
N	number of receivers
B	buffer capacity in number of packets
S	data packet size
C	control packet processing time ($= x$)
D	data packet processing time ($= 3 * x$)
E	echo packet processing time ($= x$)
x	constant used to determine processing times
L	packet error rate
D_i	propagation delay between the sender and the i-th receiver
D_{min}	minimum propagation delay
D_{max}	maximum propagation delay
$SData$	constant time between two data packets generated by the sender
$TSData$	constant service rate of data packets at the receivers
MRT	mean response time

Table 1: Simulation parameters.

Data loss or corruption has the same effect on the performance of the algorithm since retransmission prevents that buffers in the sender are released. Only when all receivers have acknowledged the correct reception of the retransmitted data, buffers can be released. With an increasing number of receivers the failure probability increases as well leading to a large number of retransmissions.

Our results indicate that to achieve a better performance while providing full reliability it is necessary to optimise buffer utilisation and to remedy the source implosion effect. The former requires a more frequent release of buffers, i.e. the time in which buffers can be released should be decreased. This would also result in reduced sensitivity towards the estimate maximum round-trip time. The latter can be realised by distributing echo messages over time. Further optimisation might allow a large number of receivers with less overhead, but it will not be possible to achieve full reliability without any extra costs.

5 Implementation over MBone

To show its applicability in an Internet environment and to validate the general nature of our FRM algorithm, we have also implemented it using the multicast backbone (MBone) delivery service. The implementation resides on top of the User Datagram Protocol (UDP) [13]. Apart from development reasons the main motivation for this solution was compatibility with other Internet protocols and the current Internet philosophy. In the same way RTP [14] provides support for real-time communication over UDP, a protocol or a set of protocols can be used to provide the required reliability for multicast communication over UDP. The FRM algorithm could be part of such a protocol.

The sender transmits all multicast packets (DATA and CNTL messages) to a Class D multicast address determined before the communication is set-up. The receivers reply

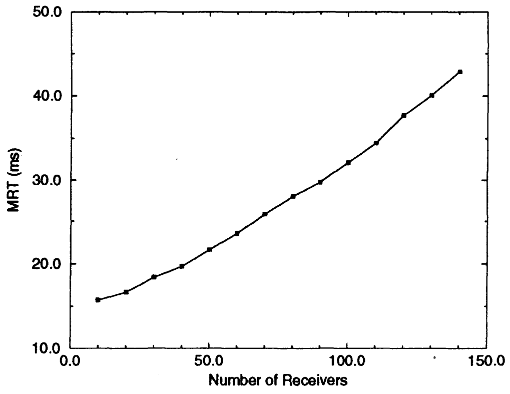

Figure 4: FRM Algorithm Scalability.

(ECHO messages) to the unicast address of the sender. Therefore, all packets originated at the sender are transmitted via a 1 to N multicast tree to the receiver group, and the replies from the receivers are sent using the conventional unicast delivery service. Only the designated sender sends to the multicast address. No other sender must use this address for this kind of fully reliable multicast communication.

Before transmitting data the multicast sender initialises the communication by sending a *INIT* packet to the multicast group address. This packet informs the receivers about the sender's name and unicast address, the minimum sustainable release throughput and the packet size. Upon reception of this packet each receiver willing to participate replies with an *INIT* packet informing the sender of his globally unique identifier and his buffer size. In this so called initialisation phase the sender estimates the maximum round-trip time for the first time and it initialises his variables accordingly.

To test the FRM algorithm over the Internet and in order to conduct experiments on a wider scale a user-level process at the sender which reads a local file and performs a multicast file transfer is used. At the moment we are conducting a number of experiments between the Laboratoire MASI in Paris, Lancaster University in the UK, and Sophia-Antipolis in Nice to measure the implementation' s performance.

6 Conclusion

Multicast (1:N) communication is an efficient way to transmit the same data to a group of receivers. A number of networks (such as Ethernet, token ring, ATM) and protocols (e.g. IP, XTP 4.0) support multicast to some extent. A major problem in this context is how to deal with data transfer reliability in a heterogeneous environment where different multicast schemes might be employed. A way to support full reliable data transmission over a number of different networks while still utilising the advantages of multicast communication is required.

The FRM algorithm introduced in this paper provides fully reliable data transmission to a group (or specified sub-group) independently of the underlying protocols or network

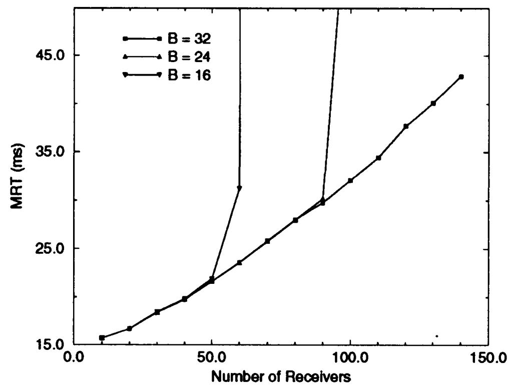

Figure 5: MRT with different buffers size.

structure. A list-based group management scheme, positive acknowledgments and retransmission for error recovery are employed. Conditions under which slow receivers that jeopardise the success of the communication are ejected can be specified. This guarantees that the QoS does not drop below an acceptable value specified in advance.

Major design goals of our service were protocol and network independence, compatibility and efficiency. To evaluate our algorithm and to assess its performance it was tested in different environments using real protocols and simulation tools. It has proved to work with XTP 4.0 mechanisms as well as on top of a UDP/IP protocol architecture. Our experience with the algorithm shows that it provides an efficient and effective way to ensure fully reliable multicast in a heterogeneous environment. It is also flexible and can even be used when different degrees of reliability are needed for a sub-set of the receiver group.

The maximum sustainable number of receivers is mainly restricted by two factors, namely limited buffer in the sender and the source implosion effect. Source implosion refers to a situation where the sender is too busy processing acknowledgments from receivers and thus is unable to send new data. To remedy these problems we are currently working on a scheme where buffers are released more frequently and source implosion is avoided by distributing receiver acknowledgments over time.

References

[1] S. Fdida, "Multimedia Transport Protocol and Multicast Communication". *High-Speed Networking for Multimedia Applications* W. Effelsberg, O. Spaniol, A. Danthine, D. Ferrari (eds.), Kluwer Academic Publishers, Boston/Dordrecht/London, 1996.

[2] L. Mathy, G. Leduc, O. Bonaventure and A. Danthine, "A Framework for Group Communications". *Third International Broadband Islands Conference.* O. Spaniol, W. Bauerfeld and F. Williams (eds.), Elsevier Science Publishers (North-Holland), Hamburg, June 1994.

[3] J. F. de Rezende, A. Mauthe, S. Fdida and D. Hutchison, "M-Connection Service: A Multicast Service for Distributed Multimedia Applications", in *Proc. of Multimedia Transport and Teleservices, International COST237 Workshop*, Copenhagen, Denmark, Nov. 1995.

[4] J. F. de Rezende, A. Mauthe and S. Fdida, "La Communication Multimédia de Groupe de Bout-en-Bout", in *Proc. of Colloque Francophone sur l'Ingénierie des Protocoles (CFIP'96)*, Rabat, Maroc, Oct. 1996.

[5] *UNI 3.1 - User Network Interface.* Prentice Hall, 1994.

[6] G. Armitage, "Support for Multicast over UNI 3.0/3.1 based ATM Networks", Bellcore, Feb. 1996.

[7] A. Koifman and S. Zabele, "RAMP: A Reliable Adaptive Multicast Protocol", in *Proc. of IEEE INFOCOM'96*, San Francisco, CA, Mar. 1996.

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

[9] J. C. Lin and S. Paul, "RMTP: A Reliable Multicast Transport Protocol", in *Proc. of IEEE INFOCOM'96*, San Francisco, CA, Mar. 1996.

[10] M. Hofmann, "A Generic Concept for Large-Scale Multicast", in *Proc. of International Zurich Seminar on Digital Communication (IZS'96)*, Zurich, Switzerland, Feb. 1996.

[11] XTP Revision 4.0, "Xpress Transport Protocol Specification", XTP Forum, Santa Barbara, CA, Mar. 1995.

[12] Simulog, "QNAP$_2$ (Queueing Network Analysis Package) version 2", Simulog, 1992.

[13] J. Postel, "User Datagram Protocol", Request for Comments, RFC-768, Aug. 1980.

[14] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", Request for Comments, RFC-1889, Jan. 1996.