Scalable and Reliable Multimedia Multipeer Communication in the M-Connection Service

Andreas Mauthe[1], José F. de Rezende[2], Serge Fdida[2] and David Hutchison[1]

[1] Computing Department
Lancaster University
Lancaster LA1 4YR
UK
email: {andreas, dh}@comp.lancs.ac.uk

[2] Laboratoire MASI - CNRS
Université Pierre et Marie Curie
4, Place Jussieu - 75252 Paris Cedex 05
France
email: {rezende,fdida}@masi.ibp.fr

Abstract

New applications such as conferencing and co-operative systems operate in a group environment. Many networks have multicast capabilities, yet, a lack of transport protocols and services that offer end-to-end group communication support can be observed. End-to-end specific issues in this context are communication integrity and the problem of reliability versus scalability. Full reliability is required by many applications. This usually generates high management overhead and restricts the number of receivers. On the other hand, some applications do not require guaranteed reliability, and a small number of unrecoverable errors can be acceptable as long as a high number of participants can be accommodated. The M-Connection Service presented in this paper provides end-to-end support for distributed multimedia systems. Four service classes to support discrete as well as continuous media have been developed. Further, different reliability schemes to satisfy the requirements of various types of applications have been designed and implemented. In this paper we introduce and discuss the algorithms and mechanisms to provide these reliability semantics in detail. The performance and scalability of each reliability scheme is evaluated using a test suite of simulations and their applicability in an Internet as well as an ATM environment is also assessed.

1 Introduction

Distributed computing, conferencing and co-operative applications require support for multipeer communication. The basic multipeer communication mode is (1:N) multicast, i.e. a topology with one sender and multiple receivers. Many LANs and MANs (e.g. Token Ring, Ethernet, FDDI, DQDB) support multicast based on their inherent broadcast characteristics. Networks such as ATM offer multicast support by providing copy functions in routers and switches. Multicast is also defined for some network protocols like for example IP. However, a lack of end-to-end group communication support for multimedia systems can be observed.

The M-Connection Service presented in this paper is a protocol independent transport layer service which offers multicast support for multimedia group applications [1]. It provides QoS support as well as reliability and group integrity. A major problem in this context is how to provide a reliable service to a potentially large number of receivers. Data transfer is reliable if all receivers will get an error free copy of the transmitted data. Error recovery algorithms are necessary to ensure reliable data transmission. For continuous media error recovery through retransmission is not appropriate. Forward error recovery schemes and special encoding techniques are proposed to provide more reliability for audio and video [2, 3, 4]. For discrete media lost or corrupted data can be retransmitted to provide reliability. However, with an increasing group size the probability that one of the recipients receives a corrupted message increases and hence the likelihood that messages have to be retransmitted is also increasing. Further, a high overhead is caused by the required management of the receiver group. In a sender-based approach the so called source implosion effect can be observed with a growing number of receivers, i.e. the sender is mainly busy managing the group and therefore not able to transmit more data. On the other hand, there are also applications for which a reliable transmission is desirable but which allow a number of unrecoverable errors. A shared white-board is an example for such an application.

Related to reliability is the problem of group integrity. Group integrity refers to the number and/or identity of receivers who have to receive the transmitted data correctly to consider the communication successful. For example a certain number of receivers might be required to cover the cost of communication or it has to be ensured that certain mandatory receivers are receiving data correctly. If the group integrity is not met any longer the communication has to be suspended or released.

The discussion in this paper concentrates on the problems of reliability vs. scalability and on multipeer integrity in the M-Connection Service. Four service classes offer different degrees of reliability and integrity. A new concept of reliability, the scalable probabilistic reliability [5] is introduced in service class D. With this concept it is possible to provide a high degree of reliability to large and dynamic user groups. The performance and scalability of the two reliability algorithms is compared using simulation results.

This paper is organised in seven sections. The M-Connection Service is presented in section two. Section three discusses the different reliability semantics provided by this service. In section four group integrity is introduced. Further, this section also discusses the implications of the different reliability semantics on integrity. A summary of the simulation work and a discussion of the results is presented in section five. Full reliability and scalable probabilistic reliability are compared according to results gathered during simulations. Section six discusses the implementation of the reliability algorithms in an Internet environment and over ATM. Conclusions are provided in section seven.

2 M-Connection Service

The M-Connection service is part of a wider group communication service architecture comprising communication services as well as management services. Each module of this architecture provides a specific service at the most appropriate location. Redundant service definitions are avoided and policies and mechanisms are clearly separated. The M-Connection Service is a fundamental module of the architecture with the aim of providing end-to-end communication support between one sender and multiple receivers for multimedia applications. It is a transport service that interfaces multiple transport protocols which support multicast (1:N) and a certain degree of QoS. The service offered by the M-

Connection Service is independent of any specific protocol or network. The service can be provided over an IP style network as well as over ATM. Its functionality is not depending on transmission characteristics or network structure. Two transport protocol architectures have so far been considered as potential candidates for our service, viz. XTP 4.0 [6] and a proprietary multimedia transport protocol architecture developed at Lancaster University [7, 8]. To show the feasibility of the M-Connection Service and how it can be provided we mapped all defined service elements onto XTP calls and mechanisms [1].

The M-Connection Service is provided by a restricted set of service elements which support all functionality needed for communication between one sender and multiple receivers. The service offered by the M-Connection Service allows service users to establish, join, leave, change properties of and terminate *m-connections*. All service primitives that constitute the M-Connection Service are shown in table 1.

service element	*primitives*
M-Connection set-up	listen_mconn, open_mconn.req
	open_mconn.ind, open_mconn.rsp
	open_mconn.cnf, open_mconn.ack
Leave M-Connection	leave_mconn.req, leave_mconn.ind
	leave_mconn.cnf
Join M-Connection	join_mconn.req, join_mconn.ind
	join_mconn.rsp, join_mconn.cnf
Change M-Connection Properties	chg_mconn_qos.req, chg_mconn_qos.ind
	chg_mconn_qos.rsp, chg_mconn_qos.cnf
	chg_mconn_qos.ack
	chg_mconn_policy.req, chg_mconn_policy.cnf
Terminate M-Connection	close_mconn.req, close_mconn.ind

Table 1: M-Connection Service primitives.

M-Connections are characterised by a minimal set of properties such as QoS, and by policies to determine the success of the communication. The aim of the M-Connection Service is twofold, i) to provide the user with a full set of clearly specified service elements for multicast communication and ii) to hide the details and specific characteristics of different transport protocols.

2.1 Service Classes and QoS Schemes

To support continuous as well as discrete media data transfer and a wide variety of multimedia applications, four distinct service classes to accommodate the requirements of different kinds of traffic are provided:

A - best-effort
B - group integrity and/or QoS
C - group integrity and data transfer reliability
D - group integrity and scalable probabilistic reliability

Class A provides a best-effort service without QoS negotiation. However, the sender can specify the QoS of the data stream to inform the receivers about the traffic type and its QoS. This service class was designed for audio video dissemination, for example security announcements or regular training sessions in a company.

The second class, Class B, offers group integrity and the possibility of negotiating QoS. Group integrity in this context refers to the possibility of specifying integrity conditions for an m-connection. QoS can be negotiated among all participants. If QoS negotiation without group integrity is required the parameters determining group integrity are set to nil. If only group integrity is required, no QoS will be specified. Service class B is intended to be used for continuous data transfer and for discrete data transfer where no data transfer reliability is required. Audio-video conference applications are potential users of this service class.

Class C provides group integrity as well as a totally reliable data transfer. Data transfer reliability is provided through retransmission of any lost or corrupted data based on state information from all receivers. The specified QoS in this class gives an indication of bandwidth and timing requirements for an m-connection. Any reservation of network or end-system resources is done according to this QoS specification. No guarantees for the timely transmission of correct data units are given. Data reliability in this class always has priority over timing constraints. This service class is intended to be used for discrete media transfer to small to medium size receiver groups where full reliability and group integrity is essential. Examples for such applications are software distribution services or the transfer of electronic documents during an audio-video conference.

Class D is especially designed to provide a high degree of data transfer reliability to large and dynamic receiver groups. The total number and/or full identity of the receiver set is not necessarily known to the sender. Also, the state of the receiver group is not completely known to the sender. Receivers request the retransmission of data by sending negative acknowledgments to the sender. The decision about retransmission is based on partial knowledge of the global group state. QoS is based on the sender specified transmission rate. No QoS negotiation takes place. This service can for instance be used by applications such as a shared white-board.

2.2 Quality of Service Negotiation

QoS negotiation in classes B and C takes place between the sender, the service provider and the receivers. During the connection establishment all necessary resources are reserved using the capabilities of the respective network service provider. Three different negotiation schemes are considered:

- imposed-QoS: The sender specifies a distinct QoS which can not be negotiated. Individual receivers can only accept or reject it.
- common-denominator: This has to be agreed on a common QoS among all participants.
- receiver-selected: The sender specifies the maximum QoS it can provide. Each receiver selects the most appropriate QoS which can be supported through adaptation mechanisms such as filtering [9].

The common-denominator case is distinguished from the other cases in two ways. First, the sender might give a range of QoS rather than a clearly specified value. Receivers can propose their QoS within this range. Second, a third message (handshake) is required to inform the receivers of the common QoS.

The QoS parameters we consider for continuous media are throughput, end-to-end delay, delay jitter, and error rate. A flow specification such as the one proposed in [10] can be used to fully describe the QoS requirements in such a case. The only relevant QoS parameter for Classes C and D (discrete media) is throughput which determines the sending rate. Since reliability is provided through retransmission no guarantee for timely delivery of messages can be given. In service class D an acceptable error rate can be defined. This gives only an indication of the number of non-recoverable errors acceptable to a receiver.

3 Reliability in Multicast Communication

3.1 Reliability Mechanisms and Algorithms

Reliability in multicast communication does not only depend on the kind of network reliability and on the retransmission functions employed but also on the size of the receiver group. A way to ensure that all receivers of the group will receive all data correctly is to keep track of the receiver status. This can be done by keeping a list of all receivers at the sender and to retransmit data whenever it was corrupted or not received by a receiver. Such an algorithm is for instance proposed for XTP4.0 [6]. The drawback of this approach is that status information from every receiver has to be send frequently to the sender. With a growing number of receiver the management overhead in the sender increases and the source implosion effect can be observed.

The problem of source implosion was addressed by a number of authors [11, 12, 13, 14, 15]. Early work in this area has concentrated on acknowledgment schemes and their effect on source implosion [11]. For large groups it is proposed to split the group into sub-groups. The size of the sub-group has to be adjusted to the maximum number of control messages that the sender can handle. The sender asks for acknowledgments from sub-groups and decides about the retransmission of messages once all members of each sub-groups have acknowledged a message. The RAMP algorithm introduced in [12] 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.

The algorithm described in [13] (called a framework for scalable reliable multicast (SRM)) was especially designed for wb (a white-board application). Scalability is achieved by a receiver-based reliability scheme in conjunction with Application Level 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 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 [14] 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 [15] 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. Further, at least some dedicated receivers are aware of other receivers and the structure of the multicast group.

In [16] an error control algorithm (so called *bucket algorithm*) is specified to provide multicast reliable communication. Every time the sender asks for status information from the receivers it stores information about the data sent so far in a data structure called bucket. This bucket serves as repository for all acknowledgement packets from the receivers associated with the respective sender request. There are a number of buckets; once all buckets are full the oldest bucket is emptied and data is retransmitted if necessary. In conjunction with this algorithm two mechanisms using acknowledgment aggregation are defined to address the problem of source implosion. The first mechanism, called slotting, is used by the receivers to distribute the acknowledgements over time. This is done by using a random time-out before sending an acknowledgement. Acknowledgements are always triggered by a control messages from the sender and are multicast to all members of the group. If a receiver receives a request for retransmission from another receiver for the same data, it refrains from sending its own status information. This second mechanism is known as damping. This strategy reduces the amount of reverse traffic from the receiver even with increasing group size. It is ensured that at least one receiver will reply and have a correct copy of the data. However, slotting and damping do not perform very well especially in a WAN environment.

In [17] a new paradigm of communication (multicast-to-some) is defined to address scalability issues. It allows for a multicast message to be delivered to a random subset of the multicast group. This mechanism can be used by the multicast source to get replies from only a subset of destinations in the group. To do this an acceptance probability p is defined for each message sent by the source. It shows that with an appropriate setting of parameters the cost of multicast can be made to depend on number of desired responses rather than the size of the multicast group.

3.2 Reliability in the M-Connection Service

The M-Connection Service is a pure end-to-end multicast service and it does not assume any additional capability from the underlying network other than the delivery of multicast packets. It would be possible to achieve other distribution models for instance by using a multicast server. However, the complexity of this approach negates the benefits of the multicast concept and is also inconsistent with the design philosophy of the M-Connection Service.

To provide full reliability, a group management scheme which keeps track of the state of all receivers is required. In the M-Connection Service this is done by using list-based group management. The list-based approach was chosen despite its known deficiencies because it complies with the design philosophy of the M-Connection Service and is relatively easy to implement on-top of 1:N multicast. In addition to full reliability, the M-Connection Service provides also a scheme where reliability to a very high degree can be achieved. This is a new concept called Scalable Probabilistic Reliability.

3.2.1 Fully Reliable Communication in the M-Connection Service

In the scheme used to provide full reliability a receiver group list is created from the positive replies to an *open_mconn.req* of the receivers. Therefore, initially a message is send to a group address announcing the set-up of a communication among members of the group. 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 Fully Reliable Multicast (FRM) algorithm.

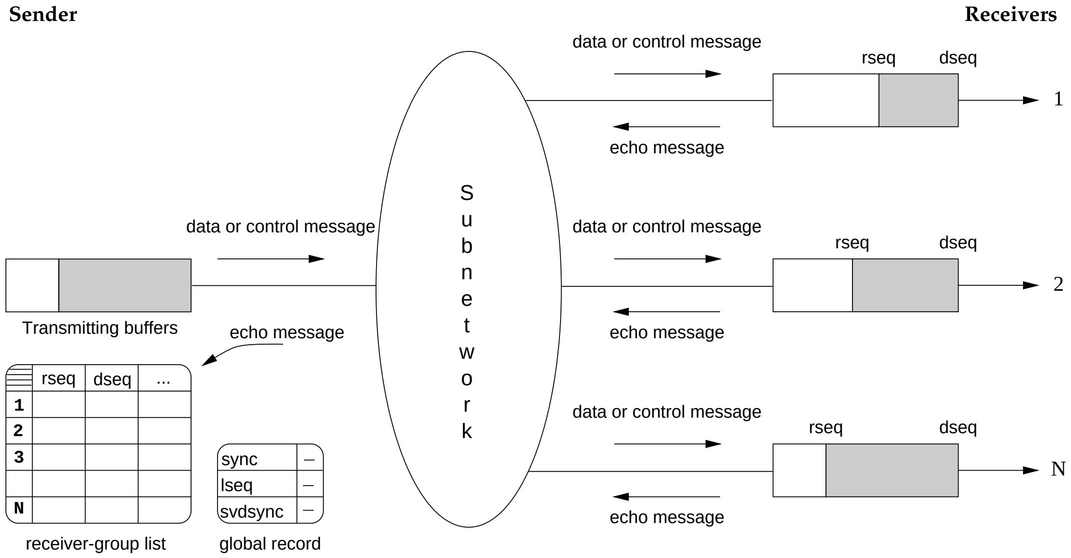

Figure 1: FRM Algorithm Structure.

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 this 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. There are two reasons why a sender might not get a report from a receiver. Either because the receiver has left unexpectedly due to machine or network failure or because the report from a receiver is lost. In a packet-switched network such as the Internet the latter can happen frequently. This will have a negative affect on the performance especially in an already overloaded system. To ensure that this does not happen, a secure out-of-band signalling as proposed in [18] is used.

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:

$$\begin{array}{l} \text{if } (smaxRTT > maxRTT) \\ \quad smaxRTT := smaxRTT + (maxRTT - smaxRTT)/16; \\ \text{else } smaxRTT := maxRTT; \end{array}$$

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. Nevertheless, in the FRM algorithm the user may specify conditions under which slow receivers that jeopardise the success of the communication are ejected. An ejection condition based upon the release rate throughput at the sender is defined for this purpose. This guarantees that the QoS does not drop below an acceptable value specified in advance.

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, a higher value of k has a negative impact on the throughput of the multicast communication because transmission buffers in the sender 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 multicast 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 currently used because of implementation reasons.

List-based multicast offers reliable multicast by tracking the explicit membership of the receiver group and by providing retransmission for all lost or corrupted data. Hence, full reliability is ensured for the whole receiver group. The main problem of this scheme is its relatively poor scalability, i.e. the number of receivers is usually restricted to around a hundred. However, it also has a number of advantages which makes it especially suitable for the M-Connection Service. It complies with the strict sender based philosophy of the M-Connection Service, its implementation complexity and the implementation overhead is low, it can be easily adapted to most environments and it provides a strict control over the whole group which is important if strong group integrity is required.

3.2.2 Scalable Probabilistic Reliability

To accommodate the requirements of applications with large and/or dynamic receiver groups with less strict demands on reliability, a new error control mechanism was developed. The reliability semantic provided by this mechanism differs from the full reliability achieved by FRM in that it only provides reliability to a high degree, i.e. there is a small probability that some errors can not be recovered.

In contrast to a list-based reliable scheme the decision to (re-)transmit data packets is based on partial knowledge of the receivers' state rather than the full, global knowledge of their state. The philosophy behind this reliability semantic is that it is the receivers' responsibility to detect and recover from packet loss. In conjunction with this, a sender based acknowledgement strategy is used, i.e. the sender asks for status information to outdate buffers containing transmitted packets. Therefore, a hybrid scheme is used where a negative acknowledgment scheme is augmented with a polling scheme used to force positive acknowledgments.

The Scalable Probabilistic Reliable Multicast Algorithm

The algorithm was inspired by the bucket algorithm first proposed for XTP 3.6 [16]. However, it differs substantially in a number of ways. The most important difference is that it does not assume that receivers have knowledge of each other or that replies to a control message are transmitted to the whole group.

The sender keeps a number of so called buckets represented by the bucket list. Periodically the sender sends a control message asking for replies (so called echo messages) from the receivers. This control message is associated with a bucket; for every control message sent the sender creates a new bucket. Within the bucket the highest sequence number of all packets transmitted so far is kept. After receiving a control message a recipient will reply with the highest sequence number of all correct packets received so far. Echo messages contain also information which is used to update state variables required for such mechanisms as flow- or rate-control, round-trip delay estimation, etc.

Further, a list of global variables exists to keep track of the receiver state. This list contains for instance the lowest sequence number of a data packet requested for retransmission (*minrseq*). A receiver which detects an error will send a negative acknowledgement (NACK) to the sender. If the sequence number of the requested packet is lower than the current sequence number stored in *minrseq*, it will be updated according to this request. It will be also updated if an echo message acknowledges a lower sequence number of correct received messages than the one which is stored in the respective bucket.

The number of buckets is finite. The optimal number of buckets depends on the amount of buffer available at the sender, the number of receivers and the maximum round-trip time. However, a problem only occurs if the number of buckets is estimated too small. It was found that the optimal number usually lies between 5 to 10 buckets. Within this range the number of buckets can be over estimated without any performance loss. If all buckets are full the sender *empties* the oldest bucket to replace it with a new one. At this time it checks if there is a request for retransmission. A retransmission request is only considered if it is ensured that the same request from another receiver has not already been served. By this strategy, redundant retransmissions are avoided [19]. All data from the requested packet onwards is retransmitted to the group (i.e. multicast go-back-N retransmission). Also, buffers are freed from packets for which no retransmission request is expected anymore. Data is only retransmitted when a bucket is out-dated, i.e. a negative acknowledgement is not served immediately. The retransmission is optimised by this in a sense that a number of requests is served with just one retransmission rather than only one request at a time.

The list of buckets corresponds to a finite time period, i.e. a bucket always remains in the list for this time period. Thus, this time period has to be large enough to allow the receivers to reply to the related control message before a bucket is out-dated. The time period used in the M-Connection Service is the worst case estimate of the round-trip-time. For contingency this is multiplied by a factor ($cfactor$) of 2. This factor can be changed, for instance in an unstable environment an increase of this factor ensures that not too many buckets are out-dated prematurely. The number of buckets in the list actually determines the time period in which control packets are sent, retransmission requests are served and buffers are freed.

Scalability and Reliability Issues

To avoid source implosion and improve scalability, a back-off algorithm is used to reduce the number of echo messages. According to this algorithm a receiver does not reply to each control packet. The full set of receivers will only reply to the first control packet. To the second control packet a receiver will reply with a probability of 0.5, to the third with a probability of 0.25, etc. To ensure that in the average case at least one receiver will reply to each control packet the probability with which a receiver replies should never be less than $\frac{1}{2^k}$ where $k = \ln_2 N$. The receivers are informed about the value of N by the sender that estimates the current number of receivers in the multicast session. This estimate is based on information gained during connection establishment and from leaving and joining receivers. Whenever a receiver has to send a negative acknowledgement the probability with which it replies to the next control packet is reset to one. This mechanism also ensures that a receiver replies on average to one in every 2^k control packets and does not fall completely silent.

Error recovery is mainly based on negative acknowledgements. A sensible choice of the time period (i.e. RTT multiplied by contingency factor, $cfactor$) for which a bucket is valid should prevent buffers from being released before all receivers have had a chance to request retransmission. However, there may be a situation where a negative acknowledgement arrives after the buffers of the respective packet have been released. This situation can for instance occur if the delay between sender and a receiver suddenly increases by a factor greater than $cfactor$. In this case the sender signals to the respective receiver that this error can not be recovered. In this message it also includes the lowest sequence number which is still available. Because of these non-recoverable errors this reliability mechanism can not provide a totally reliable service.

A tradeoff between reliability and performance exists. The larger is the value of $cfactor$ the smaller is the probability of having non-recoverable errors. A large $cfactor$ allows the sender to get negative acknowledgments from receivers suffering from network congestion. However, with a large $cfactor$ the buffers at the source are released less frequently with the same number of buckets. Each new bucket causes an additional overhead and if the number of buckets is too high the performance starts to deteriorate [19]. With the Scalable Probabilistic Reliable Multicast (SPRM) algorithm it is guaranteed that at least one receiver has received a correct copy of the message and that the sender will at least get one reply from the group.

4 Group Integrity in Multicast Communication

In multicast communication a determined sub-set of receivers have to receive the transmitted data correctly[1]. However, not all members of the group might be mandatory for the success of the communication or only a certain number is required (e.g. in distributed file systems). Hence conditions on the integrity of the receiver group are required according to which the success of the establishment and data transfer can be determined. These conditions give the number and/or identity of recipients who have to receive the transmitted data correctly. This concept called group integrity is closely related to reliability. However, whereas reliability deals with the problem of how to provide reliable data transfer to a group, integrity is concerned with the question under which condition the communication of discrete as well as continuous media data can be considered successful.

The following conditions are considered: *key-member* (identity) and/or k (number), *all*, *quorum*

4.1 Continuous Media and Group Integrity

The time dependency of continuous media makes it virtually impossible to correct errors through retransmission. Also, a small number of errors is quite acceptable for most multimedia applications. However, some group applications employing continuous media require group integrity. In this case a (sub-)group of users have to receive the transmitted data with a certain QoS.

In the M-Connection Service these applications are considered in service class B. The communication integrity conditions specify only the number and/or identity of users who

[1]Note, for continuous media data these constraints are tied to all QoS parameters and do not just apply to a single data packet [10].

have to be present. Each receiver controls the received QoS and determines if it is according to the specification. If this is not the case and all attempts to improve the situation fail, the receiver leaves the communication. It is then the task of the sender to validate the integrity conditions and to take necessary actions. If, for example, the receiver was a key member or if the number of receivers drops below the mandatory number, the sender has to release the m-connection.

Further, to avoid the possibility that integrity conditions are breached because a receiver *dies silently*[2], the sender requests periodically state information from the receivers. The periodicity with which these requests are sent, and hence the integrity is validated (so called *integrity interval*) depends on application requirements. Hence the importance of group integrity for an application determines the length of the integrity interval.

4.2 Full Reliability and Group Integrity

The strongest notion of group integrity in the M-Connection Service is provided in service class C, the fully reliable service. In this class it is ensured that either all 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 and not because it suffers from unrecoverable errors or insufficient QoS. A situation where the group integrity is momentarily not valid can only occur between two control messages.

For service class C, integrity conditions are validated every time when a receiver leaves or is forced to leave and when the sender detects that a mandatory receiver has died silently. If the integrity conditions are not satisfied any longer the sender will issue a close request.

4.3 Scalable Probabilistic Reliability and Group Integrity

Group integrity for service class D has to be regarded in a wider context. It refers not only to the interval during which the presence of mandatory receivers has to be validated as in service class B. It is also not just another reliability parameter which identifies the (sub-)group that has to receive a correct copy of the data under any circumstances. Group integrity here ensures that a mandatory set of receivers is, on average, present and that it receives in general a correct version of the transmitted data. This is reflected, for example, in the fact that an error rate is defined which specifies the acceptable number of non-recoverable errors.

All receivers on the m-connection reply to the first control message. According to these replies the sender is able to establish the number and identity of the participating recipients. Hence both, identity and quantitative integrity conditions can be validated. The frequency with which participants reply to control messages varies but is bounded by the back-off algorithm. The integrity interval has to be long enough to ensure that (on average) all receivers will have replied at least once to a control message sent during this interval. At the end of this interval it is then possible to validate the integrity conditions.

The main factor determining the length of the integrity interval in service class D is the back-off algorithm. The minimum length of the integrity interval is 2^k times the period between two subsequent control messages since every receiver has (on average replied once within this interval. However, because all receivers reply with a certain probability only, there is also the probability (though very small) that they do not reply within the integrity interval even though they are present and receiving. In the M-Connection Service an *out-*

[2]Receivers might for instance be cut off because of machine failure or network partitioning.

of-band synchronising handshake is therefore used if not all key-members have replied at the end of an integrity interval.

In contrast to the handshake of the fully reliable service the transmission of application data is not stopped during this time. There is also a certain probability that quantitative conditions are not fulfilled even though a sufficient number of recipients is present and receiving. Hence the m-connection might be unnecessarily released. To reduce the probability of such a premature action, the m-connection is released only once the minimum number of receivers has not replied within two subsequent integrity intervals.

5 Performance Analysis

To evaluate the service and the influence of different scenarios on its efficiency and performance, it was analysed using a test suite of simulations. This was preceded by a mathematical analysis of the algorithms to provide the theoretical background for the performance study [20]. The main focus of this study are the two different reliability semantics and their scalability since they generate the largest management overhead and are therefore most interesting in this context.

The simulator tool employed is QNAP_2. For the simulation a multicast communication model with a variable number of receivers (N) was used (figure 2). 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 155.5Mbit/s and a independent packet loss 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 exponential service rate of mean $TSData$. Table 2 comprises all above described simulation parameters.

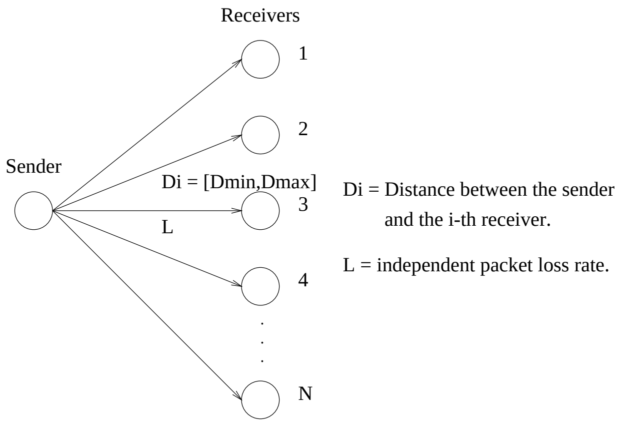

Figure 2: Multicast Tree Topology.

The main parameter used to evaluate the performance is the mean response time (MRT) or data packets. The MRT is the mean time elapsed between the sender application submitting a data packet for transmission and its delivery at the receiver user. The loss recovery algorithms employed in service classes C and D are compared according to the results obtained for this performance parameter.

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
$TSData$	mean exponential service rate of data packets at the receivers
MRT	mean response time

Table 2: Simulation parameters.

5.1 Scalability of classes C and D

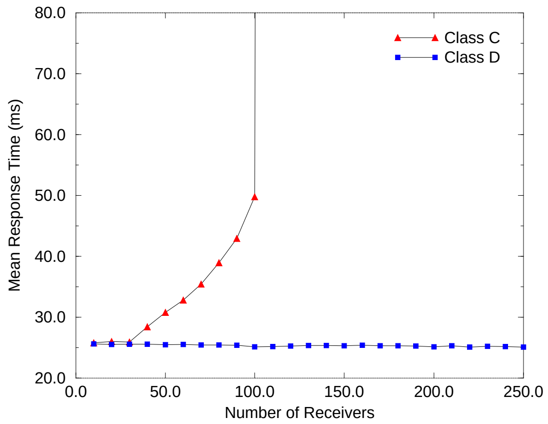

Figure 3: Mean Response Time Comparison between Classes C and D.

The scalability of classes C and D are compared using the mean response time for data packets as a function of N. Figure 3 shows the different performance of the two classes. The x-axis shows the number of receivers and y-axis shows the mean response time. In these simulations the propagation delay is uniformly distributed between [15,28] 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, $L = 10^{-4}$, $SData = 10$ ms, $TSData = 0.5$ ms. For the SPRM algorithm the number of buckets (N_{BUCK}) is 5.

As figure 3 shows, the fully reliable algorithm is stable up to a number of 30 receivers.

From this point on a steady increase of the MRT up to a number of 100 receivers can be observed. With more than 100 receivers a sudden and very rapid increase of the MRT occurs. The increase of the mean response time can be mainly explained by two factors, the source implosion effect and buffer restrictions in the sender. With 100 receivers a threshold is reached after which the sender is no longer able to handle the number of receivers efficiently.

Measurements for service class D based on the SPRM algorithm show that under the same conditions and for the same number of receivers the mean response time remains constant. For the fully reliable algorithm the mean response time is virtually the same as for the class D algorithm for up to 30 receivers. It increases to double the MRT of the SPRM algorithm with a number of 100 receivers.

In this suite of simulations no case of non-recoverable errors for the SPRM algorithm occurred. However, full reliability can not be guaranteed and the probability of such errors remains. Guaranteed full reliability does always generate additional costs. Further optimisation might allow a higher number of receivers with less overhead, but it will not be possible to achieve full reliability without any extra costs.

5.2 Influences, Side Effects and Interfering Factors

Both algorithms are influenced by different external factors related to the heterogeneity of the network and receiver group, the topology, error rate, etc. Each algorithm is affected differently and therefore optimisations to adapt the services to specific conditions have to be considered separately.

5.2.1 Full Reliability

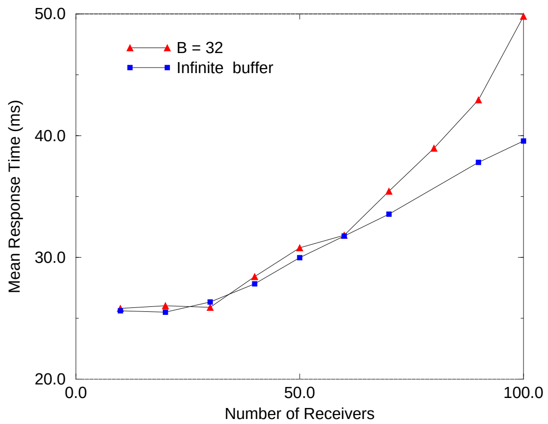

Figure 4: Mean Response Time with infinite buffers.

The two main factors influencing the mean response time with the list-based fully reliable algorithm are management overhead and buffer restrictions in the sender. The management overhead leads to an 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 hardly able to send any application data.

However, buffer restrictions also considerably influence the performance of the fully reliable algorithm. Figure 4 shows that without any restriction on buffer space the mean response time does not increase as quickly as in the case where buffers are limited.

With an increasing number of receivers the maximum round-trip time is increasing because the increased management overhead adds to the $maxRTT$ calculation. 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.

The increase of the MRT in the case of infinite buffers which can be observed in figure 4 can be solely attributed to the source implosion effect and to an increased number of retransmission requests because of the higher failure probability.

Because of the effects that the $maxRTT$ has, especially when buffers are restricted, its value has to be estimated carefully. If it is too large, buffers are not freed as early as possible. On the other hand, if it is too small the sender will enter into a synchronising handshake more often. During this handshake data transmission is also stopped which has the impact of increasing the mean response time.

The maximum round-trip time is influenced by the network delay and receiver load. A change of this value immediately affects the performance of the fully reliable algorithm. In a heterogeneous environment the maximum round-trip time estimation has to be tuned to the value of the slowest receiver. If the $maxRTT$ is very large (because for instance one participant is connected via satellite) this can result in a very poor performance even if the number of receivers is relatively small.

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 a reduced sensitivity towards $maxRTT$. The latter can be realised by distributing echo messages over time.

5.2.2 Scalable Probabilistic Reliability

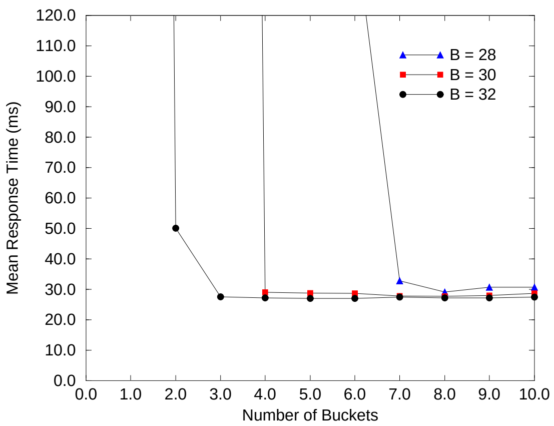

Figure 5: Number of buckets influence on the Mean Response Time.

Using the SPRM algorithm the buffer problems of the list-based approach are avoided by releasing buffers more frequently. The frequency with which buffers are released is determined by the number of buckets. However, this number becomes a major factor in influencing the performance of the SPRM algorithm. Figure 5 shows the influence of the number of buckets on the performance for different buffer sizes (using the same traffic load). With the increase of the number of buckets, fewer buffers are required at the sender to achieve the same performance. In any event, after a certain number of buckets no performance gains can be achieved by increasing the number of buckets any further. Previous work indicates that if the number of buckets is too high the performance starts to deteriorate [19]. This is due to the increased number of echo messages, more frequent status updates and retransmissions. Thus, the best value for the number of buckets is the minimum number for which the performance is calculated to be optimal. However, if this value is not known a conservative estimate should be used since there is a range in which no performance decrease with an increased number of buckets can be observed.

The SPRM algorithm is less sensitive to variations in the estimated round-trip time because this does not determine the release of buffers and consequently the pace of transmission. Nevertheless, large variations can provoke some non-recoverable errors.

In a highly dynamic and heterogeneous environment two problems occur. First, dynamic changes influence the round-trip time which can cause large variations. This has negative effects on the number of non-recoverable errors. Second, if the receiver set is very heterogeneous the algorithm always adapts to the "weakest" receiver. This will lead to a deterioration in the performance.

6 Implementation

The M-Connection Service is currently being implemented in different environments. At the moment we concentrate on service classes C and D, especially on the reliability aspects. To show the applicability and to validate the general nature of the proposed algorithms they are being implemented on top of the multicast backbone (MBone) and the ATM AAL5 delivery services. For our tests we use in both cases a user-level process at the sender that reads a local file and uses multicast transmission to the receiver group. In the case of MBone, the implementation resides on top of the User Datagram Protocol (UDP) [21]. 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 [22] 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 algorithms discussed in this paper could be part of such a protocol.

The sender transmits all multicast packets using a multicast group address determined before the communication is set-up. The receivers reply to the IP (unicast) address of the sender. Only the designated sender sends to the multicast group address. No other sender must use this address. 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 packet size and which algorithm will be used in the communication. 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 initialises his variables accordingly. 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. First results show that the FRM algorithm has performance problems in case of heavy loss. This is especially critical in the current MBone environment. The results for the probabilistic algorithm indicate that it is better adapted to this kind of environment.

In the case of ATM, a multicast connection between the sender and the receiver group and unicast ATM connections from each receiver to the sender are set-up. All packets originating at the sender are transmitted over the multicast connection to the receiver group. The replies from the receivers are sent over the unicast connection to the sender only.

The AAL5 implementation is currently being tested locally using PCs running Linux connected via three Bay Networks switches (fig 6). In the near future, we will conduct experiments on a wider scale over a national ATM backbone network connecting three sites in France (Paris, Toulouse and Sophia-Antipolis). Initial results in the local set-up confirm the simulation results

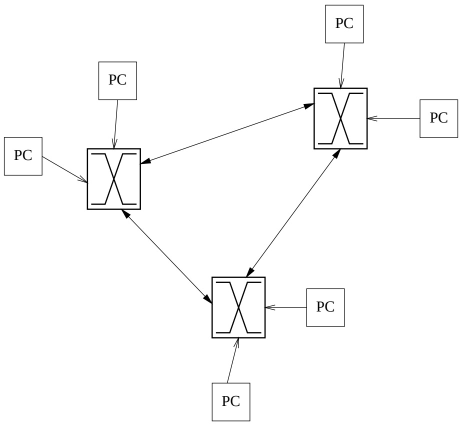

Figure 6: Local ATM Network experiments.

7 Conclusions

The M-Connection Service is a transport service that offers multicast support for multimedia applications. It is an independent service which can be provided over any kind of network with multicast capabilities and QoS support. The M-Connection Service provides four different service classes for different media types and reliability requirements. Classes A and B are designed for dissemination and continuous media services. They do not offer any kind of transfer reliability. However, for class B, group integrity can be defined according to which a mandatory (sub-)group has to receive data with a specified QoS. Class C offers full reliability ensured by means of a list-based group management and retransmission scheme. For Class D a new reliability scheme called Scalable Probabilistic Reliability was developed. Reliability here is provided through retransmission in response to negative-acknowledgement. This scheme was designed for large receiver groups and applications than can cope with a small number of errors.

A number of different reliability algorithms for multicast communication have been proposed. For the M-Connection Service it was not only important to find algorithms which perform well and scale to a certain extent, they also have to be practicable, efficient to implement, independent from the underlying network structure, and consistent with the overall design of the service. Most of the algorithms proposed so far could not be used because they do not meet these requirements. They are very often designed for specific tasks or environments, or their complexity is too high so that they cannot be integrated into to the M-Connection Service. Simulation results and our practical experience in implementing the reliability schemes introduced in this paper in an Internet and ATM environment show that both schemes accomplish these requirements well. Hence a large number of multimedia applications can be supported.

An important issue closely related to reliability but generally omitted in this context is the problem of group integrity. Group integrity is a measure according to which data transmission is successful as long as a specified sub-group of participants is present and receives the data correctly. In the M-Connection Service group integrity is provided for service classes B, C and D. Further, all service classes of the M-Connection Service offer QoS, and QoS negotiation schemes are defined for class B and C.

Some interesting and challenging problems emerged during our work on the problem of providing scalable and reliable multipeer services within a general multimedia group communication framework. For the SPRM algorithm we are experimenting with a dynamic number of buckets. The number of buckets would be adjusted according to traffic load and round-trip times. To remedy the buffer problems of the fully reliable algorithm we are investigating a scheme akin to the one in [11] in which buffers are divided into sections, where a section represents the smallest management unit. A constant and more frequent release of the buffers in these sections is achieved by a special acknowledgement, release and error recovery algorithm. Source implosion is avoided by distributing receiver acknowledgements over time.

8 Acknowledgements

For research support the authors of Lancaster University are indebted to the Engineering and Physical Sciences Research Council of Great Britain. The work on this project is funded by the EPSRC project GR/J47804 on Multimedia Group Communications Services.

References

[1] 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.

[2] E. W. Biersack, "Performance Evaluation of Forward Error Correction in an ATM Environment," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 4, pp. 631–640, 1993.

[3] B. Lamparter, O. Boehrer, W. Effelsberg, and V. Turau, "Adaptable Forward Error Correction for Multimedia Data Streams," in *Reihe Informatik 9/93*, (Universiteat Mannheim, Mannheim), 1993.

[4] V. Hardman, A. Sasse, M. Handley, and A. Watson, "Reliable Audio for Use over the Internet," in *INET'95*, 1995.

[5] S. Fdida, "Multimedia Transport Protocol and Multicast Communication", in *High-Speed Networking for Multimedia Applications.* W. Effelsberg, O. Spaniol, A. Dan-

thine, D. Ferrari (eds.), Kluwer Academic Publishers, Boston/Dordrecht/London, 1996.

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

[7] F. García, "Continuous Media Transport and Orchestration Services", *PhD Thesis*, Lancaster University - Computing Department - Lancaster, UK, May 1993.

[8] G. Danner, "Multimedia Transport Systems", *Diploma Thesis*, Lancaster University - University of Mannheim, Dec. 1995.

[9] N. Yeadon, F. García, D. Hutchison, and D. Shepherd, "Filters: QoS Support Mechanisms for Multipeer Communications," To appear in *IEEE Journal on Selected Areas in Communications*, 1996.

[10] F. García, D. Hutchison, A. Mauthe, and N. Yeadon, "QoS Support for Distributed Multimedia Communication," in *Proc. IFIP/IEEE International Conference on Distributed Platforms*, (Dresden, Germany), Feb. 1996.

[11] L. Hughes, and M. Thomson, "Implosion-Avoidance Protocols for Reliable Group Communications," in *Proc. of 19th Conference on Local Computer Networks*, Minneapolis, USA, Oct. 1994.

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

[13] 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 *Proceedings of ACM SIGCOMM'95*, (Cambridge, USA), Aug. 1995.

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

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

[16] W. T. Strayer, B. J. Dempsey, and A. Weaver, *XTP: The Xpress Transfer Protocol.* Addison and Wesley, 1992.

[17] M. H. Ammar, "Probabilistic Multicast: Generalizing the Multicast Paradigm to Improve Scalability," in *IEEE INFOCOM'94*, (Toronto, Canada), pp. 848–855, June 1994.

[18] P. Bhagwat, P. P. Mishra, and S. K. Tripathi, "Effect of Topology on Performance of Reliable Multicast Communication," in *IEEE INFOCOM'94*, (Toronto, Canada), pp. 602–609, June 1994.

[19] H. Santoso and S. Fdida, "Transport Layer Multicast: An Enhancement for XTP Bucket Error Control", in *Proc. of HPN'92, 4th IFIP Conference on High Performance Networking*, Liège, Belgium, pp. G2-1-18, Dec. 1992.

[20] A. Brandwajn and S. Fdida, "Modelling and Analysis of a Transport Multicast Protocol", MASI Report, Laboratoire MASI, Univ. Pierre et Marie Curie, Nov. 1995.

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

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