Worldwide MBone Experiments on Reliable Multicast

Markus Hofmann

Institute of Telematics, University of Karlsruhe[1]
E-Mail: markus@mhof.com
URL: http://www.mhof.com /

Abstract: Reliable multicast protocols are presently subject to intensive research. In the past, numerous protocols and mechanisms have been proposed and their respective performance been analyzed or simulated. Little progress has been made, though, to evaluate reliable multicast protocols in large-scale Internet measurements. In this paper, we describe the results of worldwide MBone experiments on reliable multicast using the *Local Group based Multicast Protocol (LGMP)*. The experiments involved up to 50 machines located at 16 geographically dispersed sites in Canada, England, France, Germany, and the USA. We compare the results for sender-based and subgroup-based reliable multicast and assess different error correction schemes. Besides the performance evaluation, the paper discusses problems encountered during the experiments and presents the design and the implementation of a flexible measurement suite for large-scale experiments on reliable multicast over the Internet.

Keywords: Reliable Multicast, MBone, Internet Measurements.

1. Introduction

In the recent years, the *Multicast Backbone (MBone)* [1] has been created to support IP multicast on the Internet. The MBone is an overlay network that is constantly growing and covers already a big part of the Internet. While it originally has been created as an experimental network for early multicast deployment, it is currently evolving towards a more robust production network. The availability of a multicast service on the Internet enabled widespread use of applications for group communication, such as conferencing tools, multicast file transfer, web caching, or distributed games. The large variety of application requirements increases the complexity of multicast, especially in reliable multicast and in congestion control. Today, it is clear that no one-size-fits-all protocol can serve optimally the needs of all group applications. Instead, several schemes and protocols have been proposed to improve reliability of multicast communication on the Internet [2].

Thorough evaluation and assessment of different approaches is an integral part of current research activities in reliable multicast. Extensive work has been done on analytical performance evaluation [3, 4]. Likewise, numerous results from simulations have been reported [5, 6]. In contrast, very little work has been done on evaluating reliable multicast protocols in real-life networks. The few studies to date of network measurements on reliable multicast are confined to including only a handful of receivers, because of the great logistical difficulties presented by large-scale multicast measurements. Consequently, their findings cannot be used to assess protocol scalability. Furthermore, known studies on reliable multicast im-

[1] Currently, Markus Hofmann is a member of the Networking Software Research Department at Bell Laboratories in Holmdel, NJ, USA.

plementations restrict themselves to a single protocol and do not compare different approaches and mechanisms.

As communication groups grow larger, performing multicast measurements grows harder. Part of the problem is the instability and the heterogeneity of the current MBone infrastructure. Another part, though, is the enormous expenditure necessary for manually setting up all the remote sites. For this reason, we devised a measurement suite in which remote sites run special agents to facilitate automated measurements. With this suite, it is possible to define and to execute measurements from a single machine using a graphical user interface. We used the distributed measurement suite to conduct various experiments over the MBone. This article reports on the major findings and describes the problems we encountered during our experiments. It represents a first step towards in depth evaluation of reliable multicast implementations in large-scale networks and it draws first results from real-life protocol comparisons.

The rest of the article is organized as followed. Section 2 briefly describes the Local Group based Multicast Protocol (LGMP), which has been used for the MBone experiments. The design and the implementation of a distributed measurement suite are presented in Section 3. Section 4 reports on the MBone experiments and presents some of the results. Finally, Section 5 concludes the article.

2. The Local Group based Multicast Protocol (LGMP)

While not necessarily typical of all (tree-based) reliable multicast protocols (see [2] for an overview on different protocols), we use the LGMP protocol as an example for analysis and discussion. The *Local Group based Multicast Protocol (LGMP)* [9] supports reliable and semi-reliable transfer of both continuous media and data files. It is based on the principle of subgrouping for local error recovery and for local feedback processing, as defined by the *Local Group Concept (LGC)* [8]. Receivers dynamically organize themselves into subgroups, which are called *local groups*. They automatically select a *Group Controller* to coordinate local retransmissions and to process feedback messages. The selection of appropriate receivers as Group Controllers is based on the current state of the network and of the receivers themselves. In contrast to other reliable multicast protocols, no manual or administrative intervention is necessary. LGMP subgroups are self-organizing and self-adapting, thus, improving fault-tolerance and system reliability. The selection of Group Controllers is not part of the data transfer protocol itself. Instead, a separate protocol has been defined and implemented, which is named *Local Group Configuration Protocol (LGCP)* [9]. LGCP provides mechanisms for automated self-configuration of local groups and for dynamic reconfiguration in accordance with the current network load and group membership.

In LGMP, packet errors are firstly recovered inside local groups using a receiver-initiated approach. Missing data units are requested from the sender or a higher level Group Controller only if not even a single member of the local group holds a copy of the missing data unit. Otherwise, errors will be recovered by local retransmissions within the subgroup. We will refer to this mechanism as *local error recovery*. Local error recovery is different from tree-based protocols using *strict hierarchical error recovery* [7, 16]. With strict hierarchical error recovery, Group Controllers always request missing data packets from their parent, even if the data packet has been successfully received by a member of their subgroup. LGMP uses packet-based selective retransmissions, and it precisely restricts the scope of local retransmissions by using separate multicast addresses for each local group. Full reliability and efficient buffer utilization are achieved using a three-state acknowledgment scheme. This scheme enables dynamic reconfiguration of subgroups and hot swapping of Group Controllers in the case of system failures without loosing synchronization.

LGMP implements a hierarchical, rate-based congestion control scheme. The problem of congestion detection is handled for each local group separately, thus, being able to run a TCP-like scheme on acceptable timescales. A receiver, whose packet loss rate exceeds a pre-defined threshold, sets the CONGES-

TION bit in its outgoing feedback messages. On receiving such a message, Group Controllers perform two steps: First, they adapt their retransmission rate according to the "multiplicative decrease and additive increase" principle. Second, they set the CONGESTION bit in outgoing feedback messages, thus, forwarding the congestion notification upstream towards a higher level Group Controller and the sender. Finally, the sender adapts the transmission rate according to received congestion notifications. Note that the congestion control scheme is not the focus of the measurements presented in this paper. More work and careful evaluation needs to be done and other schemes might be applied, especially in the context of the IRTF and IETF efforts on multicast congestion control.

LGMP as well as LGCP have been implemented and tested on a variety of different platforms, such as Linux, Solaris, Digital Unix, SCO UnixWare, Windows 95 and Windows NT. The implementation provides a multi-threaded user space library that can be linked to any application. There is also a Java-API available. The source code of both protocols is available free of charge on the World Wide Web [9].

3. A Measurement Suite for Multicast Protocols in the Internet

The first problem one has to face when planning large-scale network measurements is searching for an adequate number of remote sites that are willing to participate in the measurement. The lower the requirements for participation are, the easier this task will be. For example, system administrators are more likely to support a network experiment if it is not necessary to provide a dedicated account on the machines. Ideally, remote administrators get an easy to install software package. Once the software has been started, it will automatically initiate all steps necessary to perform a measurement according to a previously defined measurement configuration without the need for remote login.

In addition, the design and implementation of a measurement suite for multicast protocols is motivated through the large expenditure for manual experiments. At the beginning of every measurement, one has to connect to all remote sites in order to start the required measurement programs. Afterwards, log files and probes need to be collected for evaluation purposes. For example, in a test scenario with 50 receivers, one has to open 50 different sessions for login to remote sites. After entering username and password 50 times, the appropriate test programs need to be started at all sites. Finally, log files need to be collected manually by doing 50 file transfers. This example illustrates the enormous amount of work for manual set up of large-scale multicast measurements and motivates the development of a measurement suite for large-scale multicast experiments. The measurement suite should meet the following requirements:

- *Automated execution of measurements*: The distributed measurement suite must be able to execute experiments automatically. The starting time should be freely selectable. As far as possible, network problems should not impair ongoing measurements. Errors and results need to be logged for evaluation purposes, and users should be able to check the status of ongoing measurements.
- *Remote Control*: Parts of the measurement suite, such as components to send and receive data, need to be installed and started on remote sites all over the world. In order to facilitate the execution of measurements, it should be possible to control and manage these distributed modules from a centralized site. This will reduce the administrative overhead at remote sites.
- *Simple installation*: The different components of the measurement suite should be easy to install and to activate. Ideally, there will be no more administrative or configuration overhead for remote partners after installing and starting the piece of software. This goal is very important because it facilitates the search for partners willing to join the measurement.
- *Simple use*: The definition of measurement configurations as well as the execution and control of measurements should be very simple and easy to use. Therefore, a graphical user interface supporting each of those tasks needs to be designed. Ideally, the GUI allows defining measurements by simply drag'n drop the name of Internet hosts for definition of multicast groups and subgroups.

- *Flexibility*: In order to allow comparisons of different approaches, new multicast protocols need to be integrated into the measurement suite easily. Therefore, the measurement suite should work independent of the protocol to be measured. A modular architecture is required to allow easy switching between different protocol implementations.

These requirements result in two main tasks. On the one hand, measurements are defined at a local machine and distributed to the participating remote sites. In addition, results are collected and evaluated at a local machine. These actions form the so-called *management task*. On the other hand, the measurement suite executes the actual measurements by participation of remote sites and it logs all the results. This is called the *measurement task*. The task sharing results in a two-part architecture according to the functionality of both tasks. The architecture is illustrated in Figure 3-1.

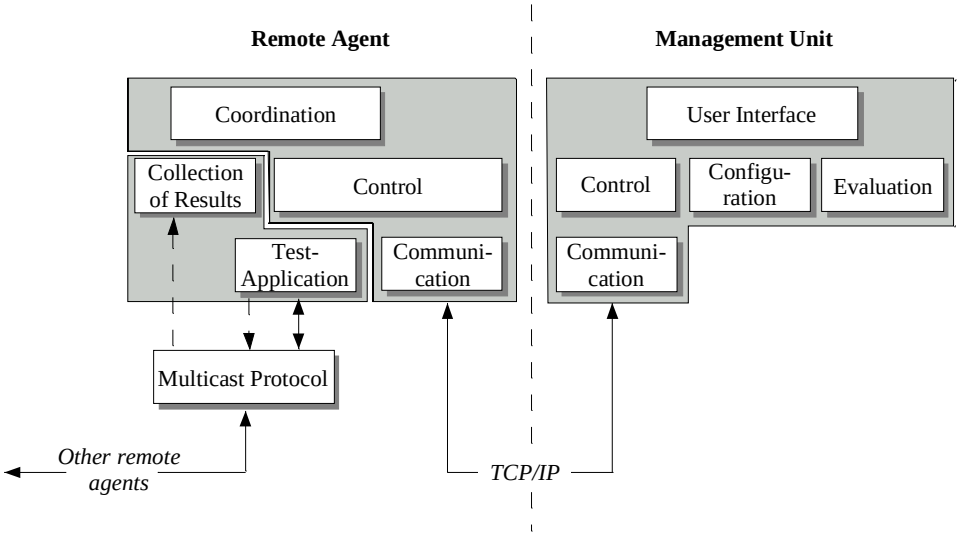

Figure 3-1: Architecture of the Measurement Suite

The so-called *Management Unit* sets up the central component of the measurement suite. It supports the user in defining measurements. A configuration editor with graphical user interface (see Figure 3-2) is used to comfortably define the sender, a set of receivers, and the starting time of a measurement. Besides these, the user may define various other parameters such as the source rate or the packet size. All parameters are stored in a configuration file, which is automatically transferred to the hosts involved in a measurement. At the end of each measurement, the Management Unit supports the user in processing and preparing the obtained results. For this purpose, it provides a set of powerful statistical functions and tools for graphical representation. Another task supported by the Management Unit is the control of active measurements. This includes starting, aborting, and managing measurements. Functions are provided to query the status of each remote site at any time.

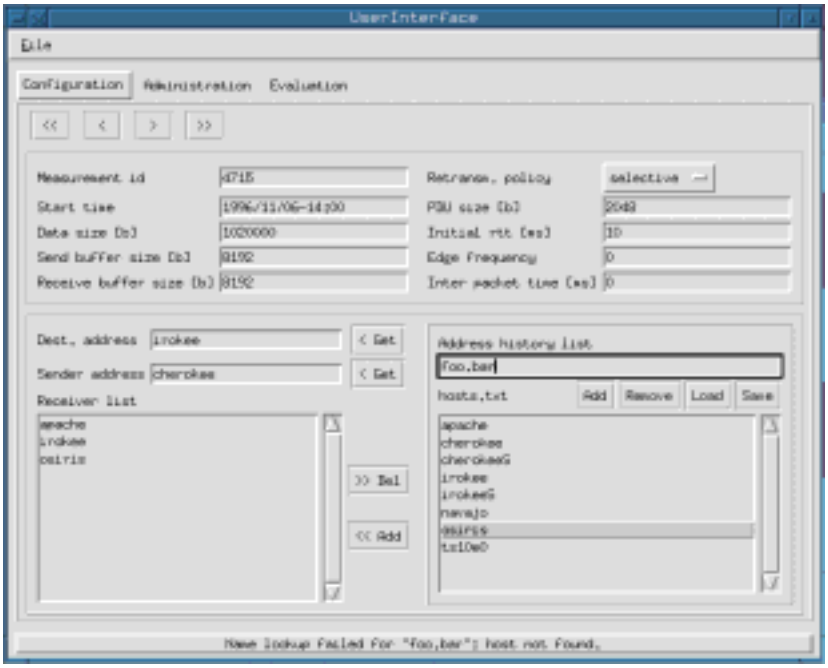

Figure 3-2: Graphical User Interface of the Configuration Editor

The Management Unit has to be installed on a single machine only. It allows the user to control a measurement from a single, centralized host without the necessity to logon to any remote site. For easy portability, a first version of the Management Unit has been implemented in *Python* [10]. With the advance of the *Abstract Window Toolkit (AWT)* for Java, developing graphical user interfaces in Java has become more feasible. Version 2 of the Management Unit has been completely written in Java and the message exchange with remote agents is now based on *Remote Method Invocation (RMI)* rather than native TCP/IP.

The actual measurement (e.g. sending and receiving data, probing, etc.) is done by so-called *Remote Agents*. Earlier versions of the agent software have been written in C, while the current release is implemented using Java. This allows an automated software update when new versions become available and improves portability. Once installed and started, remote agents listen to receive configuration files from the Management Unit. On receiving a configuration file, agents schedule all the operations listed in the file and start the corresponding programs at the given time using the given parameters. Examples for such operations are sending or receiving data using the protocol to be measured. During data transfer, agents measure several parameters (e.g. throughput, packet loss rate, round trip time, number of packets received, etc.) and store their value in a log file. The log files are automatically transmitted to the Management Unit at the end of a measurement and can be processed using its statistical and graphical tools. Remote agents do also respond to queries and commands of the Management Unit during data transfer. In such a way, they answer status queries or may cancel or abort a scheduled measurement.

The different steps necessary to perform a measurement could be summarized as follows: First, the agent software needs to be distributed and installed at remote sites. The respective administrators could install and activate the agent software. It is not necessary to get a guest account at remote sites. The agents are now ready to receive configuration files. The operator defines measurements using the configuration editor of a Management Unit and distributes the generated configuration files by simply clicking the “OK” button. On receiving configuration files, remote agents schedule all required operations, monitor the measurement and return their log files back to the Management Unit. No further intervention is necessary at remote sites once the software package has been installed and activated. It is possible to execute or to

repeat as many measurements as desired without any manual support by remote partners and without the necessity to manually login to remote sites.

4. Measurements on the MBone

Before being able to run a single measurement in our global target environment, we spent several weeks together with local and national network administrators on fixing MBone related problems. The problems ranged from incorrectly configured multicast routers up to more serious problems concerning the routing configuration in national backbones. Section 4.4 gives a short summary of the main issues. Finally, we were able to conduct measurements involving up to 50 machines located at 16 geographically dispersed sites in Canada, England, France, Germany, and the USA. We performed several measurements using different configuration parameters, different protocol mechanisms, and different receiver subsets. All measurements were executed multiple times to ensure their correctness and to identify possible runaways due to temporary atypical network conditions. In the following sections, we summarize our findings by presenting the results of a medium size experiment. It has been picked out of the large number of data sets because it nicely represents the basic results of all the conducted experiments.

4.1. Methodology

The experiment that is discussed in this paper is based on the topology given in Figure 4-1. It consists of a sender located in London, England, and of 31 receivers located at 12 geographic areas in France, Germany, and the USA.

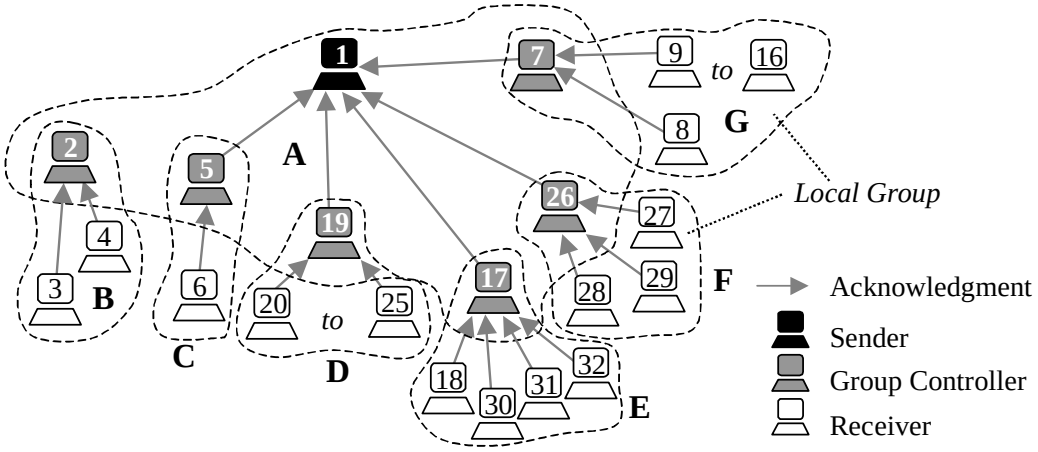

Figure 4-1: Network Topology and Subgroup Structure

In detail, the following hosts were involved in the experiment:

<1>	*thud.cs.ucl.ac.uk* (London)	(128.16.64.16)
<2>	*sahir.cs.umass.edu* (Amherst, USA)	(128.119.40.229)
<3>	*age.cs.columbia.edu* (New York, USA)	(128.59.22.100)
<4>	*wednesday.cc.gatech.edu* (Atlanta, USA)	(130.207.8.24)
<5>	*maia.lip6.fr* (Paris, France)	(132.227.61.3)
<6>	*violette.eurecom.fr* (Sophia Antipolis, France)	(193.55.114.53)
<7>	*lanai.rvs.uni-hannover.de* (Hannover, Germany)	(130.75.5.239)
<8>	*ibdr122.inf.tu-dresden.de* (Dresden, Germany)	(141.76.49.22)
<9>	*molinari.ibr.cs.tu-bs.de* (Braunschweig, Germany)	(134.169.34.163)
<10>	*bacardi.ibr.cs.tu-bs.de* (Braunschweig, Germany)	(134.169.34.168)
<11>	*johnnie.ibr.cs.tu-bs.de* (Braunschweig, Germany)	(134.169.34.165)

<12>	*walker.ibr.cs.tu-bs.de* (Braunschweig, Germany)	(134.169.34.176)
<13>	*jaeger.ibr.cs.tu-bs.de* (Braunschweig, Germany)	(134.169.34.167)
<14>	*meister.ibr.cs.tu-bs.de* (Braunschweig, Germany)	(134.169.34.161)
<15>	*metaxa.ibr.cs.tu-bs.de* (Braunschweig, Germany)	(134.169.34.173)
<16>	*sierra.ibr.cs.tu-bs.de* (Braunschweig, Germany)	(134.169.34.179)
<17>	*eratosthenes.informatik.uni-mannheim.de* (Mannheim, Germany)	(134.155.48.125)
<18>	*sokrates.informatik.uni-mannheim.de* (Mannheim, Germany)	(134.155.48.105)
<19>	*osiris.telematik.informatik.uni-karlsruhe.de* (Karlsruhe, Germany)	(129.13.3.121)
<20>	*moon.telematik.informatik.uni-karlsruhe.de* (Karlsruhe, Germany)	(129.13.3.122)
<21>	*mond.telematik.informatik.uni-karlsruhe.de* (Karlsruhe, Germany)	(129.13.3.123)
<22>	*luna.telematik.informatik.uni-karlsruhe.de* (Karlsruhe, Germany)	(129.13.3.124)
<23>	*kuu.telematik.informatik.uni-karlsruhe.de* (Karlsruhe, Germany)	(129.13.3.125)
<24>	*papago.telematik.informatik.uni-karlsruhe.de* (Karlsruhe, Germany)	(129.13.35.18)
<25>	*quechan.telematik.informatik.uni-karlsruhe.de* (Karlsruhe, Germany)	(129.13.35.16)
<26>	*cembalo.informatik.uni-stuttgart.de* (Stuttgart, Germany)	(129.69.210.24)
<27>	*gambe.informatik.uni-stuttgart.de* (Stuttgart, Germany)	(129.69.210.27)
<28>	*spinett.informatik.uni-stuttgart.de* (Stuttgart, Germany)	(129.69.210.28)
<29>	*violine.informatik.uni-stuttgart.de* (Stuttgart, Germany)	(129.69.210.26)
<30>	*fiddle.kom.e-technik.tu-darmstadt.de* (Darmstadt, Germany)	(130.83.139.122)
<31>	*viola.kom.e-technik.tu-darmstadt.de* (Darmstadt, Germany)	(130.83.139.121)
<32>	*lyre.kom.e-technik.tu-darmstadt.de* (Darmstadt, Germany)	(130.83.139.123)

The hosts are all workstation-class computers (e.g. SUN Sparc10, Digital Alpha) running Solaris 2.5.1, Solaris 2.6, or Digital Unix 4.0. Future experiments will also include Windows NT based PCs.

Using the subgroup based LGMP, receivers were arranged as illustrated in Figure 4-1. For example, receivers located in the USA (receivers 2, 3, and 4) formed a subgroup represented by a Group Controller in Amherst, MA, USA. The local group represented by receiver 19 combined all the receivers located in Karlsruhe, Germany (receivers 20 to 25). The remaining receivers were arranged as illustrated in Figure 4-1.

Each receiving host automatically executed an LGMP-based client application at the time defined in the corresponding configuration file[2]. A few seconds later, the sender started transmission of a one Mbytes sized data file. Each receiver and each Group Controller measured transfer time independently and logged the received and transmitted packets. By analyzing the log files, we were able to determine the total number of (re-) transmitted packets in each subgroup, as well as the number and the origin of local retransmissions. LGMP payload size was set to 1442 Bytes, thus, resulting in an overall number of 694 original data packets to be transmitted. A maximum data rate of 160 Kbits/s has been chosen to avoid overloading the MBone tunnels to the partners. Each measurement was executed five times. While the tables in this paper will show the results of each measurement, the figures will only illustrate the mean of all five measurements.

Our measurements aimed for a fair comparison of sender-oriented protocol mechanisms with LGMP specific mechanisms, such as local error recovery and hierarchical feedback processing. For this purpose, we also used our LGMP implementation in combination with a static version of LGCP, which allowed manual configuration of local subgroups. By configuring a single subgroup with the sender as its Group Controller, we were able to force all receivers to send their feedback directly to the sender. Accordingly, the only system performing retransmissions was the sender. This behavior exactly corresponds to a sender-oriented protocol scheme. Using the same protocol implementation for assessment of both, sender-oriented and LGMP-based schemes ensures a fair comparison of protocol mechanisms. In this way, our

[2] Our measurement suite provides mechanisms for automated synchronization of local system times.

comparison more accurately exposes the differences in the protocol mechanisms rather than the differences in the skill of protocol implementers.

4.2. Performance Metrics

Our assessment of different protocol techniques is based on two performance metrics: *network load* and *transfer time*. Network load is measured by the number of packets that are transmitted to each receiver for successful transmission of the entire file. This includes original transmissions as well as global and local retransmissions. While network load is of great importance to network providers, users are much more interested in the time it takes to transmit a certain amount of data. For this purpose, we also measured the transfer time for each receiver separately. The transfer time is measured by the time between starting the data transmissions and receiving the entire file successfully. In general, good performance is indicated by low network load and low transfer time.

4.3. Experiments

The following sections compare LGMP with alternative approaches and explain the obtained results. Since each of the 32 hosts measured its own performance indices, some Tables and Figures will show the average numbers reported by receivers of the same subgroup. For the sender-oriented approach, we calculated the mean of the same receiver subsets, although no subgroups have been used for data transmission. The numbers of transmitted packets that are shown in the tables do include retransmissions. In the sender-oriented approach based on global retransmissions, this number will be the same for all receivers. For subgroup based LGMP, however, the number of transmitted packets will be different for each local group.

4.3.1. Local Error Recovery vs. Sender-based Error Recovery

One goal of our experiments was to investigate the benefits of local error recovery over a traditional scheme with global retransmissions performed by the sender. For this purpose, we compared the measured network load as well as the observed transmission time for both approaches.

Network load for the sender-oriented approach with global retransmissions is given in Table 4-1. It shows for each of the five measurements the number of data packets that originated at the sender in London (including original data packets and retransmissions). For the sender-oriented approach, all those packets are forwarded to all the receivers.

Group	***Result 1***	***Result 2***	***Result 3***	***Result 4***	***Result 5***	***Mean***
Global	3984	3079	3160	3768	5048	**3808**

Table 4-1: Network load for sender-oriented approach [number of packets]

It is remarkable that, on average, 5.5 times the data of the original file was sent to each receiver (3808 packets compared to 694 packets for the entire file). This result is due to a large number of mainly uncorrelated packet losses in the MBone. Indeed, average per receiver packet loss rate varied from 48% in Mannheim up to 77% in New York (see Figure 4-2), and packet losses in different subgroups were mainly uncorrelated. The observed loss characteristics were similar to the ones reported in [11].

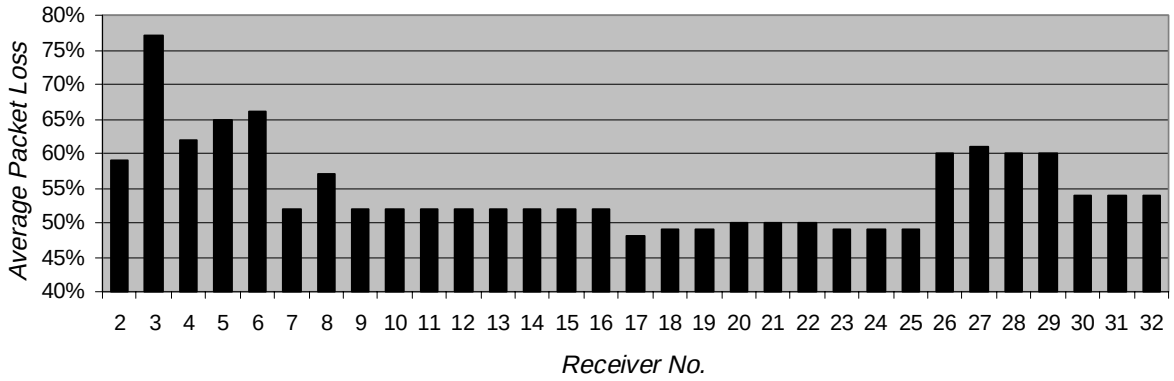

Figure 4-2: Average Packet Loss Rates of Receivers

The results for the subgroup-based approach are given in Table 4-2, whereas network load is shown for each subgroup separately. The network load for a subgroup is composed of packets that originate at the sender and that are addressed to the global group plus local retransmissions within the subgroup.

Group	***Result 1***	***Result 2***	***Result 3***	***Result 4***	***Result 5***	***Mean***
Subgroup A	899	850	857	889	922	**883**
Subgroup B	1592	1418	1077	1498	1521	**1421**
Subgroup C	903	853	870	890	931	**889**
Subgroup D	697	698	711	719	714	**708**
Subgroup E	729	716	718	718	739	**724**
Subgroup F	836	810	823	797	832	**820**
Subgroup G	704	727	729	725	1326	**842**

Table 4-2: Network load for subgroup-based approach [number of packets]

From the results, we make two major observations. First, local retransmissions significantly reduce the number of repair packets required to transmit a data file successfully and, second, they avoid delivery of duplicate packets and flooding of repair packets by restricting the scope of local retransmissions[3]. Both facts result in a significant decrease of aggregate network load. In our case, network load was decreased around 76%.

The first observation becomes clear on considering the mean values of all the subgroups. In the sender-based approach, for example, there were 3808 – 694 = 3114 repair packets necessary to transmit the file correctly. In contrast, the aggregate number of repair packets was reduced to ((883 + 1421 + 889 + 708 + 724 + 820 + 842) – 7 · 694) = 1429 retransmissions for the subgroup-based approach. These are 1685 retransmissions less than for the sender-based approach. These benefits partly arise from lower packet loss rates in local regions compared to global environments, which results in fewer losses of (local) retransmissions.

Despite the aggregate number of retransmissions, sub-grouping also reduces network load and the number of duplicate packets by restricting the scope of (local) retransmissions. Without scoping, all retransmis-

[3] Remember: LGMP restricts the scope of local retransmissions by using separate multicast addresses, and *not* by using TTL scoping.

sions are forwarded to all group members, which is why the entire group has to suffer from relative high packet loss rates in some parts of the network [6]. In our sender-based experiments, high loss rates in New York caused a large number of retransmissions to be sent to the entire group, thus increasing network load along the entire multicast routing tree. As shown in Figure 4-3, network load was the same in all the subgroups, although receivers in Germany, for example, observed much lower packet loss rates and did not need all the repair packets.

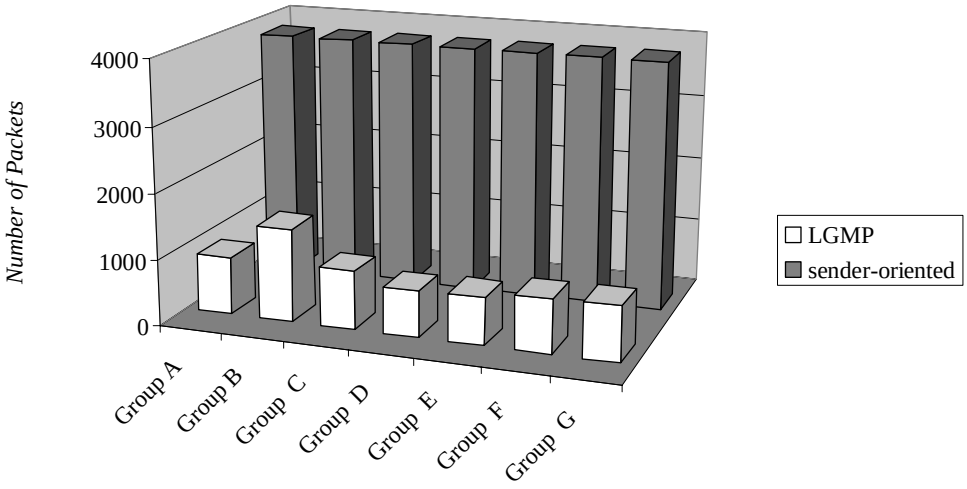

Figure 4-3: Comparison of Mean Network Load

In the subgroup-based approach, in contrast, most packet errors were resolved using local retransmissions with restricted scope. As shown in Figure 4-3, high packet loss rates in subgroup B increased network load in this specific subgroup, but did hardly effect the other ones. For example, an average of 708 packets was sent to receivers in subgroup D. This number includes all the original data packets as well as global and local retransmissions. In the sender-based approach, in contrast, an average of 3808 packets was forwarded to the same receiver set.

Besides the network load, we also measured the transfer time for each receiver separately. The results are shown in Figure 4-4.

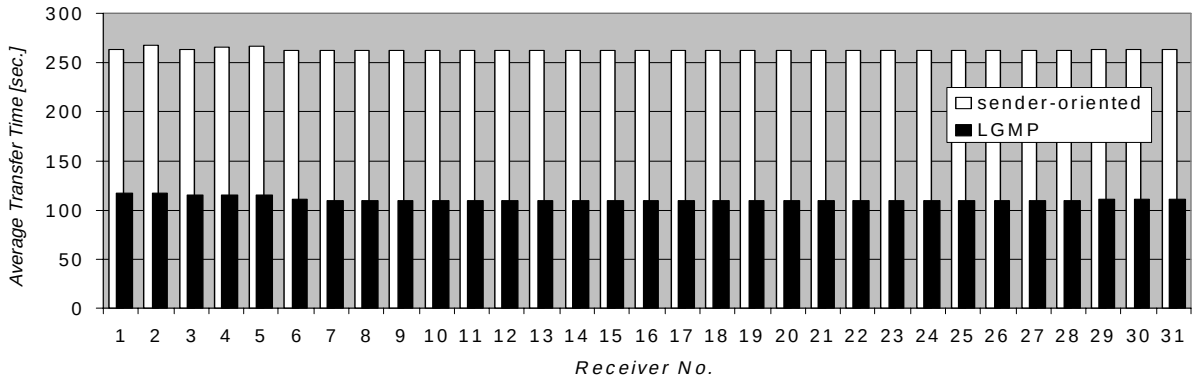

Figure 4-4: Comparison of Average Transfer Time

Again, we make two major observations from the results. First, average transfer time decreased around 58% when using the subgroup-based approach. While users of the sender-oriented scheme had to wait about 263 seconds to receive the entire file correctly, LGMP users successfully finished transmission after

112 seconds on average. The significant reduction of transfer time arises from local error recovery, which reduces the number of necessary retransmissions and hastens error correction by local exchange of repair packets. Another observation is that there is almost no variation in transfer time amongst the receivers. For the sender-oriented scheme, lowest transfer time reported by any of the receivers was 262 seconds while the highest one was 267 seconds. For the subgroup-based approach, mean transfer time varied from 110 seconds to 117 seconds. The very small variation in transfer time is due to the conservative, TCP-like congestion control of LGMP. The transfer rate and the sending window are adapted according to the slowest receiver, which is acceptable for a large variety of applications (e.g. bulk data transfer). If this behavior is not wanted, an application must set a minimum transfer rate and receivers that fall below this rate must leave the group. Another multicast group can be set up to serve those receivers with a lower rate. Handling heterogeneous multicast groups is not a congestion control issue per-se, but an application issue.

Besides network load and transfer time, we were also interested to see which systems performed local retransmissions and how LGMP's coordinated local error recovery compares to strict hierarchical error recovery. The next section will answer these questions.

4.3.2. Local Error Recovery vs. Strict Hierarchical Error Recovery

In contrast to strict hierarchical error recovery schemes, receivers in LGMP are actively involved in local error correction. LGMP receivers are able to perform local retransmissions, thus, helping other subgroup members to recover from transmission errors. Therefore, we were interested to see how often regular receivers participate in error correction and what the benefits are compared to strict hierarchical approaches. As an example, Figure 4-5 shows the number of local retransmissions that originated at each of the members of subgroup G.

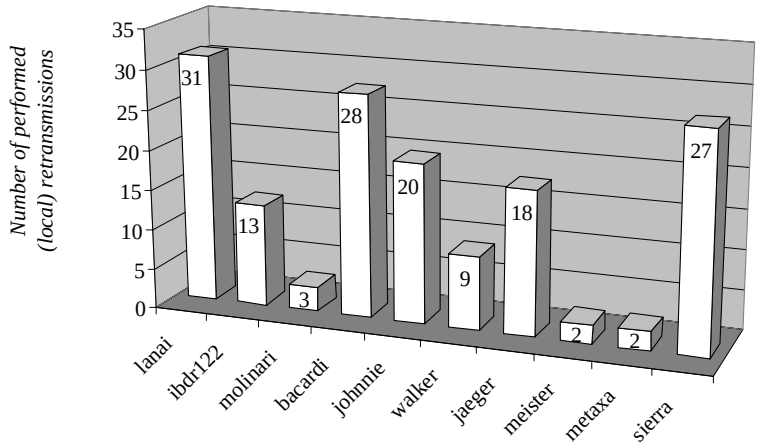

Figure 4-5: Sources of Local Retransmissions in Subgroup G

In this specific subgroup, a high percentage rate of all local retransmissions was not performed by the Group Controller, but by regular receivers. This is true despite the fact that the receiver with the lowest packet loss rate has been chosen to act as Group Controller. Indeed, Group Controller *lanai* performed most of the retransmissions, although 80% of all local repair packets originated at regular receivers. In strict hierarchical approaches, these local repair packets would have been requested from higher level Group Controllers, thus, increasing aggregate network load and repair delay.

The number of local retransmissions originating at regular receivers was not that high in all subgroups. In subgroup B, for example, 5% of local repair packets were sent out by regular receivers. Using strict hier-

archical error recovery, these packets would have been retransmitted across the Atlantic, which is not the case for LGMP style error correction. There also were subgroups (e.g. subgroups E and F) in which not a single local repair originated at a regular receiver. In these cases, LGMP acts like a protocol using strict hierarchical error recovery and it will provide the same network performance.

Our experiments show that active involvement of regular receivers is advantageous over strict hierarchical error recovery. However, the achievable benefits strongly depend on the placement and the capacity of Group Controllers. For example, an ISP might provide powerful Group Controllers with excellent network connection at strategic locations within its network. In such an environment, strict hierarchical error recovery is very likely to perform equally good as LGMP style error correction. Without explicit support by the ISP, regular Internet hosts take over the role of Group Controllers. Depending on the current system load and network connection, these hosts are likely to observe packet losses themselves. Our measurements clearly show that in such situations active involvement of regular receivers in error correction can provide notable performance gains.

4.4. Experience

During our experiments, we encountered a variety of difficulties that made worldwide data exchange over the MBone hard or even impossible. First, not all test partners had a connection established to the MBone and it took some time to help local network administrators setting up and configuring a tunnel to the MBone. Other sites were connected to the MBone, but incorrect tunnel configuration resulted in improper operation. At one site, for example, a multicast routing loop had been configured accidentally. Depending on the TTL value, a single multicast packet was received up to 128 times, thus, increasing network load dramatically. Some sites had meaningless TTL thresholds configured on their local multicast routers making it difficult to scope multicast packets correctly.

Another problem occurred with network configurations running PIM-SM [12] in the backbone and DVMRP [13] in the client network. In this environment, the multicast routing table was not set up correctly and no multicast packets were forwarded into the client network if the group members were listening but never sending to the multicast group. This is not critical to most existing multicast applications (e.g. the MBone tools), because they do send multicast feedback to the group. In contrast, LGMP and other reliable multicast protocols send unicast feedback to the Group Controller. In other words, LGMP receivers never send data to the global IP multicast group. This simplex-style multicast scheme triggered the mentioned problem so that we were not able to receive a single LGMP packet in these client networks. As a workaround, LGMP receivers started an additional process that periodically sent empty dummy packets to the global IP multicast group. Today, the “Member must be Sender” problem has been widely acknowledged and first patches have been announced for PIM-DM [14] client networks.

We used some of the tools described in [15] for allocating and debugging the encountered problems. Although the tools proved to be very useful in local environments, they often failed in the global scale. For example, it was rarely possible to trace a global multicast routing tree using the `mtrace` utility. It was very likely that at least one router on the tree did not answer correctly. However, it needs to be emphasized that the problems were not caused by the tools themselves, but rather by wrongly configured multicast routers running old software versions. For theses reasons, we mainly used very simple ‘sender’ and ‘receiver’ programs to check MBone connectivity and to track down possible problems. The programs are part of the LGMP distribution package [9].

Our experiments also showed that the packet loss rate over the MBone increased dramatically when transport level multicast packets were fragmented. In order to avoid fragmentation, payload size of transport level packets should not exceed the network’s MTU size minus the length of underlying protocol headers. These include IP headers, optional UDP header and the transport level protocol header. Note that IP tunneling is used to transfer multicast packets across non-multicast-capable routers. Therefore, an addi-

tional IP header for IP encapsulation should be taken into account when calculating the maximum payload type.

Overall, worldwide MBone connectivity turned out to be very unreliable. It was rarely possible to reach all the test sites simultaneously via multicast. Most of the times, either international MBone connectivity was interrupted or problems in one of the national backbones or at one of the test sites prevented successful multicast transmission. These recurring failures made large-scale experiments in a worldwide environment very hard and time-consuming. Fortunately, the MBone is currently leaving its experimental status and is moving towards a production network. Stepwise deployment of multicast-capable routers in the backbone makes MBone connectivity much more robust and removes the inefficiency and instability of manual MBone tunnels. However, the current MBone is not yet simply plug'n'play and more work needs to be done to improve its availability and to provide easy to handle and reliable management tools.

5. Conclusions

In the past, extensive work has been done on analytical and simulative performance evaluation of reliable multicast schemes. We believe that these results should be complemented by protocol measurements in real-life networks. Some results have been reported from smaller-scale network measurements, each of them focusing on a specific reliable multicast protocol. The work presented in this paper compares different approaches and addresses scalability in global environments. It represents a first step towards large-scale network measurements on reliable multicast. We have performed various experiments on the MBone and we discussed the encountered problems. Based on the gained experience, we have presented the design and the implementation of a distributed measurement suite that facilitates scheduling and automated execution of measurements, especially in a global environment. It allowed us to conduct a variety of large-scale experiments getting insights into the performance of different protocol mechanisms for reliable multicast. The results prove that local error recovery yields significant performance benefits over sender-based schemes in terms of network load and transfer time. They also show that active involvement of regular receivers in error recovery is advantageous over strict hierarchical error recovery. More experiments will be done to examine other issues such as congestion control and automatic subgrouping.

6. References

[1] M. Macedonia, and D. Brutzman. *MBone provides Audio and Video across the Internet;* IEEE Computer Magazine, page 30-36, April 1994.

[2] C. Diot, W. Dabbous, J. Crowcroft: *Multipoint Communication: A Survey of Protocols, Functions, and Mechanisms*; IEEE Journal on Selected Areas in Communications, Vol. 15, No. 3, Page 277-290, April 1997.

[3] B.N. Levine, J.J. Garcia-Luna-Aceves: *A Comparison of Known Classes of Reliable Multicast Protocols*; Proceedings of ACM Multimedia'96, November 1996.

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

[5] S. Floyd, V. Jacobson, S. McCanne, C.-G. Liu, L. Zhang: *A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing*; Computer Communication Review, Vol. 25, No. 4, Proc. of ACM SIGCOMM'95, Boston, MA, USA, August 1995.

[6] C. Hanle, M. Hofmann: *A Comparison of Reliable Multicast Protocols using the Network Simulator ns-2*; Proceedings of IEEE Conference on Local Computer Networks (LCN), Page 222 - 237, Boston, MA, USA, October 11-14, 1998.

[7] S. Paul, K.K. Sabnani, J.C.-H. Lin, S. Bhattacharyya*: Reliable Multicast Transport Protocol (RMTP)*; IEEE Journal on Selected Areas in Communications, Vol. 15, No. 3, April 1997.

[8] M. Hofmann: *A Generic Concept for Large-Scale Multicast;* Proceedings of International Zurich Seminar on Digital Communication, Zurich, Switzerland, February 1996.

[9] Local Group Concept (LGC) Website at http://www.telematik.informatik.uni-karlsruhe.de/~hofmann/lgc/.

[10] Python Language Website at http://www.python.org/.

[11] M. Yajnik, J. Kurose, D. Towsley: *Packet Loss Correlation in the MBone Multicast Network*; Technical Report 96(32), University of Massachusetts at Amherst, 1996.

[12] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, L. Wei: *Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification*; Internet Request for Comments RFC 2117, June 1997.

[13] S. Deering, C. Partridge, D. Waitzman: *Distance Vector Multicast Routing Protocol*; Internet Request for Comments RFC 1075, November 1988.

[14] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, A. Helmy, D. Meyer, L. Wei: *Protocol Independent Multicast Version 2: Dense Mode Specification*; Work in Progress, Internet Draft draft-ietf-pim-v2-dm-03.txt, June 7, 1999.

[15] D. Thaler, B. Aboba: *Multicast Debugging Handbook*; Work in Progress, Internet Draft draft-ietf-mboned-mdh-01.txt, October 1998.

[16] R. Yavatkar, J. Griffioen, M. Sudan: *A Reliable Dissemination Protocol for Interactive Collaborative Applications*; Proceedings of ACM Multimedia '95 Conference, November 1995.