

Reliable Multicast for the Grid: a comparison of protocol implementations

M. P. Barcellos^{1,2}, M. Nekovee², M. Daw¹, J. Brooke¹, S. Olafsson²

¹University of Manchester, Oxford Road, Manchester M13 9PL

²BT Research, Adastral Park, Martlesham, Suffolk IP5 3RE, UK

Abstract

Many Grid applications can benefit from reliable multicast (RM). One example is a simulation that generates Terabytes of data, which needs to be reliably transmitted to multiple visualization servers (and archiving machines), and from these to clients. There has been much research on scalable RM protocols, and the literature has many examples of protocols and analytical/simulation comparisons. Unlike previous work, this paper describes an experimental evaluation of a selected set of reliable multicast protocol implementations, in the context of Grid applications. The experiments are performed on UK's high-speed academic network SuperJanet. Our results show that the use of these protocols greatly reduces the network overhead that is incurred in point-to-multipoint transfers using TCP. However, we found that the current implementation of the protocols fail to satisfy the high-throughput requirements of the Grid.

1. Introduction

Grids [1] are distributed infrastructures that join together, and coordinate, computing resources of multiple organisations in order to enable communities of users to solve problems and share results. A Grid infrastructure may include specialised measurement instruments, super-computers and PC clusters, data repositories and high-speed (wide-area) networks to connect them all.

Certain Grid applications require the delivery of same content to multiple destinations. One example is distribution of hundreds of Terabytes of data that is generated by a large instrument at one location to several storage facilities elsewhere [2]. Another prime example is in Grid-enabled computational steering applications, such as those of RealityGrid [3]. In such applications large scale simulations can generate hundreds of Terabytes of data at one location that need to be delivered to a set of remote sites [4]. Researchers can interact with simulations and steer accordingly, based on simulation results that are being processed and visualized. The data need to be delivered intact and promptly, either as streams or in smaller files, since scientists and engineers depend on the data to be able to change the course of remote simulations.

Such point-to-multipoint transfer scenarios could be handled efficiently and elegantly using reliable multicast communication. In its simplest form, multicast [5,6] is the process of sending every single packet from the source to multiple destinations in the same logical multicast group. Multicast may be present at multiple layers in the IP protocol stack. At the network layer, IP multicast allows efficient dissemination of the packet through the network with the help of multicast-enabled routers. The source transmits a single copy of the packet to a multicast group address. This is then duplicated by routers, where required, and delivered to all group members [7].

In the last few years, a number of authors have pointed out the potentially great value of a multicast service for the Grid [8,9], and recently a multicast-TCP approach for grid applications was put forward in [9]. To date, however, the most important use of multicast for the Grid is for Access Grid (AG), a multi-site multimedia collaboration and application-sharing platform which deploys multicast for group-to-group audio and video transmissions [10]. AG, however, uses only best-effort multicast, which does not guarantee safe delivery of all packets transmitted. This is far less challenging than reliable multicast, which is needed in the Grid data transfer scenarios discussed above. In addition to reliability, desirable features in such applications are high throughput and low end-to-end delays [8].

There has been substantial research on reliable multicast, including scalable mechanisms, new protocol descriptions, and comparisons via analysis and simulations [11]. However, there have been very few experimental studies of RM protocols [12,13], and none of these addresses Grid applications.

In this paper we use extensive experiments performed on SuperJanet [14], to examine the performance of a selected set of reliable multicast protocols, in the context of Grid applications. The final aim of this work is to build a reliable multicast communication platform that is tailor-made for the Grid and integrate it with RealityGrid applications [15] (see Fig. 1).

The rest of this paper is organized as follows. In section 2 we give a brief overview of the protocols considered, and what motivated the choice of these protocols. In section 3 we describe our network and experimental setup, and discuss a distributed application that we have developed for automation of the experiments. This is followed by a description and some analysis of our key findings. We close this paper in section 4 with conclusions and an outlook of future work.

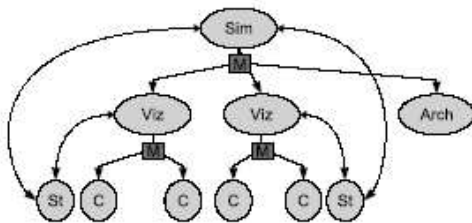


Figure 1. Schematic diagram of the use of multicast (M) in computational steering (St) and visualization (Viz) applications, such as those in RealityGrid.

2. Overview of protocols

The two main algorithms which underlie basic IP multicast service are the Internet Group Management Protocol (IGMP), which is used by hosts to join or leave a multicast group, and various algorithms for setting up multicast forwarding trees along which packets are routed to destinations [7]. Reliable transport is built on top of this basic service using some method of

loss detection and repair of these and/or forward error correction techniques. Some protocols also provide mechanisms for congestion and rate control, secure transmission etc.

Our basic criteria in selecting a subset of RM protocols from the large number of protocols proposed to date were:

- Availability of open source implementations, preferably for several operating systems.
- Provision of an end-to-end solution to reliability and scalability (no requirement for additional router support).
- Relative implementation maturity.

Based on the above criteria we examined the following protocols.

Multicast Dissemination Protocol (MDP)

MDP has been developed by Naval Research Laboratory (NLR) [16]. It is a protocol framework and software toolkit for reliably multicasting data objects, including files and application memory blocks.

MDPv1 relies on packet-based erasure techniques and adaptive group timing mechanisms. It employs negative acknowledgements (NACKs) for loss discovery and a timer-based backoff mechanism at receivers to avoid the so-called “Nack implosion” which plagued a number RM protocols.

MDPv2 (henceforth called MDP for brevity) adds a parity-based repair mechanism and forward error correction (FEC). MDP was implemented using C++ and the Protolib communication protocol library. It supports most Unix platforms, including Linux, and Windows.

Nack-Oriented Reliable Multicast (NORM)

The primary design goals of the NORM family of protocols [18] are to provide efficient scalable and robust bulk data transfer across possibly heterogeneous IP networks and topologies protocols. NORM uses a selective negative acknowledgement mechanism similar to that employed in MDP for transport reliability. A congestion control mechanism is specified to allow the NORM protocol fairly shares available network bandwidth with other transport

protocols, such as TCP. The protocol offers a number of features to allow different types of applications or possibly higher-level transport protocols to utilize its service in different ways. The protocol leverages the use of FEC-based repair and other IETF reliable multicast transport (RMT) building blocks in its design. NORM is currently being standardized by IETF's reliable multicast transport protocol working group [19].

We examined two implementations of NORM protocols:

- NORM implementation by NLR (version 1.1b3-17th February 2004) [19]. This is a follow up to MDP, and hence shares many of its characteristics. **NORM/NLR** is being implemented in C++ and relies on Proto-Lib. A congestion control mechanism is under development.
- NORM implementation by the French National Institute for Research in Computer Science and Control (INRIA) (MCL 2.99.2, 19th December 2003) [20]. This implementation of NORM is part of a larger project on reliable multicast communication, called MCL, which aims to provide an easy-to-use and integrated solution for, reliable and highly scalable multicast deliver of data. MCL is composed of a C/C++ library and applications built on top of it. According to [20], **NORM/MCL** is still experimental and hence several limitations exist, including the lack of congestion control.

Local Group-Based Multicast Protocol (LGMP)

LGMP [21] is a protocol implementation based on the ideas defined by the local group concept. It supports reliable and semi-reliable transfer of both continuous media and data files. LGMP is based on the principle of sub-group formation for local error recovery and feedback processing. Receivers dynamically organize themselves into subgroups, which are called local groups. They select a Group Controller to coordinate local retransmission of lost packets and to process feedback messages. This is in contrast with MDP and NORM in which repairs are always handled at sender. The use of local groups could be advantageous in reliable multicast over global wide area networks, where there are long delays between the source and some of the receivers.

Our comparison of the implementations of the above protocols used a set of criteria, including code-based maintenance (to match new versions of libraries and compilers, for example), support from authors (to help determine the optimal protocol parameters, debug the protocols and understand results), as well as ease of installation, usage and source code documentation. MDP, NORM/NRL and NORM/MCL are actively supported by their authors. Both implementations of NORM have the NORM Internet Drafts to help explain their working principles, but these Drafts do not help very much to understand their respective implementations. MDP is reasonably documented, and there is some documentation available for LGMP.

For our experimental evaluation the most important feature of the above protocols was whether they compiled and ran on all machines and platforms that comprised our testbed. We were able to achieve this for MDP and both implementations of NORM. Despite all our attempts, however, LGMP didn't compile at all or crashed immediately upon execution. For this reason we were unable to evaluate this protocol experimentally.

3. Experimental evaluation

The above protocol implementations compared in the previous section were used in a set of WAN performance trials on SuperJanet. This section describes the network and experiment setup, experiment automation and some key findings of our experiments.

3.1 Experiment and network setup

Our setup comprised 9 PCs (1 source and 8 receivers) all running GNU/Linux. The multicast source was at BT Research, which is located at Adastral Park in Martlesham. It was connected to SuperJanet via a 150 Mbs ATM link that provides the connection between the UCL (University College London) campus at Adastral Park ([UCL@Adastral](#)) and the main UCL campus in London. Receiver nodes were located at multiple UK e-Science centres and were connected to each other and the source over SuperJanet.

Multicast data is transported on the SuperJanet backbone and is delivered to each regional network where it connects to a Backbone Access Router (BAR) [14]. Each regional network

is responsible for forwarding multicast data to its member organisations that are in turn responsible for internal distribution to end-users over their LAN infrastructure. Multicast routing on the SuperJanet backbone uses PIM-SM [7] and each BAR is configured to transport both multicast data and routing information to the regional networks to which it connects.

3.2 Input parameters and settings

To experimentally assess the implementations files are sent among machines using both multicast and multiple-unicast streams (we call this MultiTCP). Each experiment is comprised of a large set of runs. Each of these runs consist of sending reliably a file from one machine to one or more machines, and obtaining confirmation that all data have been successfully received at each destination.

The maximum speed a protocol can archive depends on a number of factors, including link capacity, network environment, the available multicast management and routing service, and protocol settings. As a baseline for comparison we performed TCP transfers between the source at BT Research and receiver nodes. These produced Goodputs between 77 and 92 Mbps. The performance of multicast protocols depends also on the input parameters that are provided by user, and in particular on sending rates. To achieve high throughput and robustness with low network cost, it was necessary to explore the input parameter space of each protocol. This is a challenging task when performed over a wide area network due to many potential sources of disturbance that make it necessary to perform each measurement many times, in order to obtain reliable statistics. Other issues we faced were the heterogeneity of operating systems and hardware of the machines involved, the necessity of detecting and recovering from protocol failures and measurement and collection of performance data from remote sites. The above challenges were addressed by creating a distributed application, which we describe below.

3.3 Experiment automation

In essence, the distributed application performs thousands of individual experimental runs, varying executables and their parameters. A single experiment run is activated by executing the *file transfer tool* that is specific to each protocol. The minimal parameters for this tool

are the IP multicast address and port; other parameters depend on the role of the machine (sender or receiver) and the protocol that is being used. For example, the sender (receiver) may take parameters such as time-to-live (TTL) and buffer size as well as name and location of the files to be transmitted (stored). Each tool has its own syntax and parameter passing style. The incorrect use of parameters may crash the tool or make it very inefficient.

The distributed application, written in Python, elegantly accommodates such variations in number of parameters, meaning and syntax. It employs a *source agent* at the source host and a *remote agent* at each destination host. The remote agents are first activated and wait for connection requests from the source agent. Subsequently the source agent is started and connects to each of the remote agents, instructs them in setting up the experiment, synchronizes with them and then starts an experiment run. All parameters for the run, including the protocol to be used and its parameters, are configured at the source agent and sent to sender agents. After the run, the source agent collects information from remote agents, determines if the transfer was successful and extracts any relevant quantity from this information. In the meantime, receiver agents waits for the arrival of the next message from the source, which will indicate whether a new experiment is being launched and if so, which parameters it is using.

There are many reasons that can cause termination of transfer without completion. These include protocol deadlocks; livelocks, source and receiver crash failures (segmentation faults), and performance failure (throughput drops to 1% or less of what is expected). The distributed application deals with these by detecting the problem, killing the protocol processes, re-synchronizing sender and receiver agents, and then starting a new experiment.

3.4 Results

In evaluating the performance of RM protocols we used the following metrics.

- **Success (S):** This metric gives the percentage of successful experiments, and is an indication of how robust a protocol is under real-life network conditions. An experiment is counted as not being a success if the file does not arrive completely within a “reasonable” time to all destinations. The time

limit is set by the sender at the beginning of a transfer, and is chosen based on the file size, available bandwidth and sending rate.

- **Goodput (G):** This metric is computed by dividing the file size by the total transfer time. The total transfer time consists of the time taken for completion of the file transfer plus the time to check the integrity of the file at the receiver and acknowledging this to sender.
- **Network Overhead (N):** We measure this quantity by sniffing the network at the sender and counting all those packets that are used in either multicast or multiple unicast transmission of the same file. The difference between the total size of the packets transferred and the file size is then defined as the network overhead incurred by a transmission.

Since results can depend very much on network conditions it is necessary to perform a large number of measurements and average the results. For example, in our measurements of S we performed at least 50 measurements for each file size and each protocol, resulting in a total of over 1,600 runs.

In Fig. 2 results are shown for the variations of transmission success (S) as a function of the file size (denoted as SZ). The groups size (denoted as GS) was kept fixed at $GS=4$ in these experiments, and we used the best set of parameters, as obtained from our searches, for each protocol. It can be seen that MultiTCP is the only protocol that reaches 100% robustness regardless the file size. The success of NORM/NLR (denoted as NORM) and MDP remains consistently above 95%. On the other hand, NORM/MCL (denoted as MCL) was robust for all file sizes below 16 MB but after this S drops to below 86%.

The variations of Goodput G with the number or receivers is shown in Fig. 3 for the multicast protocols considered, and is compared with that of MultiTCP. Fig. 3 shows that MultiTCP by far outperforms the multicast protocols for all group sizes considered. However, it can be seen that, with regards to Goodput, TCP's performance is not scalable, and drops sharply as GS grows. In contrast, after an initial transient behavior, the Goodput of the RM protocols stabilizes, and show only small variations with the group size. Ideally, of course, for $GS=1$ (point-to-point transfer) the RM protocols

should achieve a Goodput that matches the TCP's performance, and in this respect the performance seen here is rather disappointing. However, our results clearly show the superiority of multicasting in terms of scalability.

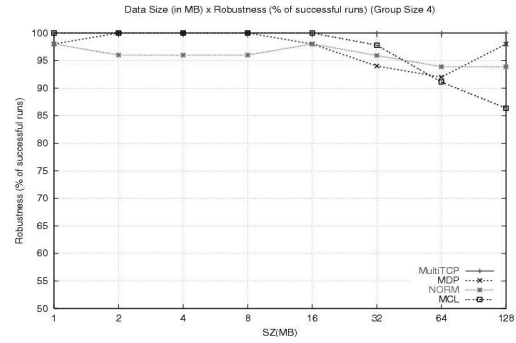


Figure 2. Success rate S is shown as a function of the size of the file SZ that is transmitted. The group's size is $GS=4$.

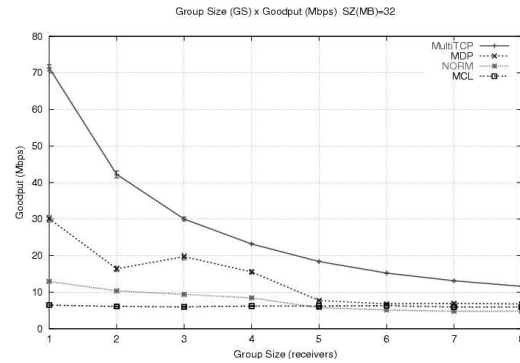


Figure 3. Variations of Goodput (G) as group size increases. The size of the transmitted files is fixed at 32 MB.

Finally, we consider the network overhead N resulting from point-to-multipoint transmissions of a 32 MB file. Fig. 4 shows how N varies as the group's size is increased from 1 to 8 receivers. For $GS=1$ (point-to-point transmission) TCP generates around 1.4 MB overhead, which is much smaller than the overhead generated by RM protocols. However, as expected, TCP's overhead increases linearly with the group size (note the logarithmic scale of the y axis in this figure), and reach around 224 MB when the group size is 8. Once again, it can be seen that, unlike TCP, the RM protocols show a scalable behavior and, relative to TCP, the network overhead generated increases only slowly with the group size.

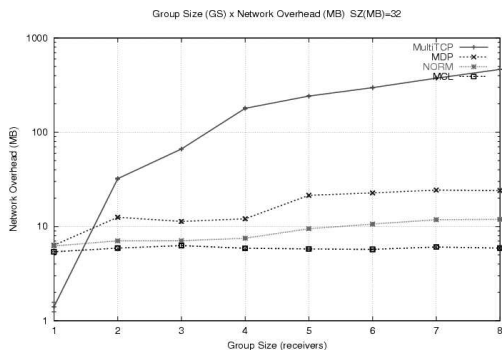


Figure 4. Variations in Network bandwidth overhead (N) is shown as the group size increases. The size of the transmitted files is fixed at 32 MB.

4. Conclusions

In this paper we explored the use of reliable IP multicast for efficient point-to-multipoint data transfer in Grid applications. Our particular emphasis was on applications such as Grid-enabled computational steering that require high-throughput reliable delivery of very large data files.

In this context, we experimentally evaluated a selection of reliable multicast protocols for which open source implementations were available. The experiments were carried out on one of the UK's high-performance networks Super-Janet, and involved up to 9 machines located at several e-Science centres, using a distributed application that we used for automations. All experiments were performed using both multicast protocols and multiple TCP transfers, and the results were compared.

As expected, we found in multipoint transfers a sharp drop in TCP's Goodput as the number of destinations increases, combined with a linear increase in the network overhead generated by TCP. These results confirm that, due to the large file sizes involved in many Grid applications, the use of TCP in point-to-multipoint transfers can become quickly prohibitive, even if only a few sites are involved. As regards multicasting, the reliable multicast protocols we considered show a scalable behaviour (in terms of Goodput and generated network overhead) with the group size. These results strengthen the case for the use of reliable multicast for the Grid.

Our evaluations also showed that the reliable multicast protocols considered here are unable to achieve the high-performance (in terms of

Goodput) that is required for the Grid. In fact they seem to require much improvement in order to match TCP's performance in this respect.

Our future work will focus on understanding, and hopefully mitigating, the reasons behind the low Goodput of the protocols considered in the current work. We also plan to explore other reliable multicast protocols, as well as alternatives such as Digital Fountain [22], which promises to achieve both 100% reliability and high-throughput in point-to-multipoint transfers.

Acknowledgements

We are grateful to Tom Crummey (University College London) and Mick Russell (BT Exact) for their invaluable help with multicast setup. Our work was jointly sponsored by the e-Science Northwest Centre (in association with RealityGrid), and BT's Long Term Research Venture. We thank Brian Adamson and Vincent Roca for their help with NORM and Martin Koyabe for useful discussions.

References

- [1] I. Foster and C. Kesselman, eds., *The Grid: Blueprint for a Future Computing Platform*, Morgan Kaufman, San Francisco (1999).
- [2] LHC Grid Computing Project's website <http://lcg.web.cern.ch/LCG/>
- [3] RealityGrid's website <http://www.realityGrid.org>
- [4] S. Pickels *et al.* The TeraGyroid Project, in *Proceedings of the 10th Global Grid Forum (GGF10) Workshop on Case Studies on Grid applications*.
- [5] D. R. Cheriton, S. E. Deering, Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, 8 (2), 85-110, September 1990.
- [6] S. Deering, *Multicast Routing in Datagram Internetwork*, PhD thesis, Stanford University, December 1991.

- [7] D. Koisur, IP multicasting: The complete guide to interactive corporate networks, Wiley Computer Publishing, John Wiley & Sons, New York (1998).
- [8] V. Sander, W. Allcock, P. CongDuc, I. Monaga, P. Padala, M. Tana, F. Travostino, Networking issues of Grid infrastructures, Grid High-Performance Networking Group, Global Grid Forum, June 2003, draft-ggf-ghpn-netissues-0.txt.
- [9] K. Jeacle and J. Crowcroft, Reliable high-speed Grid data delivery using IP multicast, Proceedings of UK All Hands Meeting 2003, Nottingham, UK.
- [10] Access Grid website <http://www.accessgrid.org>
- [11] M. W. Koyabe and G. Fairhurst, Reliable multicast via satellite: A comparison survey and taxonomy, International Journal of Satellite Communication, Vol. 24, 21-26, 2001.
- [12] M. Yanjik, J. Kurose, D. Towsley, Packet loss correlations in the Mbone multicast network, in Proceedings of IEEE Global Internet Conf., London , Nov 1996.
- [13] M. Hoffman, World-wide Mbone experiments on reliable multicast, Technical Report TR2000-14, January 2000.
- [14] SuperJanet website <http://www.ja.net/>
- [15] The MUST Project, Multicast Streaming Technology for the Grid website, <http://www.sve.man.ac.uk/Research/AToZ/MUST/>
- [16] J. P. Macker, The Multicast Dissemination Protocol (MDP) Toolkit, in Proceedings of IEEE MILCOM, volume 1, 626-630, 1999
- [17] IETF Working Group on Reliable Multicast Transport website, <http://www.ietf.org/html.charters/rmt-charter.html>
- [18] B. Adamson, C. Bormann, M. Handley, J. Macker, NACK-Oriented Reliable Multicast Protocol (NORM), RMT Working Group Internet Draft, January 2004.
- [19] NRL - NORM website <http://norm.pf.itd.nrl.navy.mil>
- [20] INRIA – MCL website <http://www.inrialpes.fr/planete/people/roca/mcl/mcl.html>.
- [21] M. Hofmann, Scalable multicast communication in wide area networks, PhD Thesis, University of Karlsruhe, Germany, 1998.
- [22] J. W. Byers, M. Luby, M. Mitzenmacher, A. Rege, A digital fountain approach to reliable distribution of bulk data, in Proceedings of SIGCOMM, 56-67, 1998.