Reliable high-speed Grid data delivery using IP multicast

Karl Jeacle and Jon Crowcroft
University of Cambridge Computer Laboratory
15 JJ Thomson Avenue, Cambridge CB3 0FD, UK
{firstname.lastname}@cl.cam.ac.uk

Abstract

In recent years, much work has been done on attempting to scale multicast data transmission to hundreds or thousands of receivers. In today's Grid environments, however, a typical application might involve bulk data transfer to just ten or twenty sites. Using multicast for this type of application can provide significant benefits including reduced load on the transmitter, an overall reduction in network traffic, and consequently shorter data transfer times.

In this project, we are investigating how multicast can be exploited within such an environment without requiring major changes to applications or underlying networks. The approach taken is to modify TCP to support multicast transfers, and run this modified TCP engine over UDP as a userspace transport protocol. We describe the work to date on the design and implementation, and provide some early experimental results.

1 Introduction

The aspirations driving recent multicast research work have been to achieve scalability to hundreds or thousands of receivers - audio streaming over the Internet being a typical application. Within a Grid environment, however, a typical application might involve bulk data transfer to just ten or twenty sites. Using multicast for this type of application can provide significant benefits including reduced load on the transmitter, an overall reduction in network traffic, and consequently shorter data transfer times.

Most previous work on reliable multicast has proposed new APIs and protocols that are not present outside of a lab environment. This has led to much theoretical analysis, but little practical deployment.

Given the large file transfers taking place across the Grid, and unicast being the only transport mechanism available, an 80/20 rule can be said to apply: 80% of transfers are destined for a small number of receivers, while 20% of transfers are destined for a large number of receivers.

Therefore, the benefit of multicast in the Grid is not its potential for massive scalability, it is simply that some parallel replication of packets is taking place.

In order to investigate the benefits that multicast can offer in an existing wide-area network such as a Grid environment, a different approach is necessary.

2 Constraints

Before discussing a mechanism for introducing multicast capability, some clarification is necessary with regard to the scope of the problem being addressed.

Transmitting data to multiple destinations via multicast will always be more efficient than transmitting via unicast, but an increase in transfer speed will be possible only if the source, or a link close to the source, is a bottleneck in the network.

To achieve this increase, no additional application-level functionality is required, or indeed desired, at intermediary nodes in the network. Native multicast support is the sole requirement.

The constraining factors are the extent of multicast deployment in the network, and the impact of ACK implosion. But assuming symmetric links, sending 1500 byte data packets and receiving 40 byte acknowledgments will constrain the group size to just under 40 receivers. This is quite acceptable for today's Grid environments.

3 Modifying TCP

Today, applications use TCP for reliable unicast transfers. It is a mature and well-understood protocol. By modifying TCP to deliver data to more than one receiver at a time, and use multicast when available, an application can transparently send data reliably to multiple recipients. Using existing TCP mechanisms, the modified protocol ensures that data is delivered reliably. Multicast transmission is attempted for performance reasons, but fallback to unicast preserves backwards compatibility and reliability guarantees, and capitalizes on more than a decade of experience that TCP implementations enjoy.

Network protocols are typically implemented in the kernels of hosts and network devices. Any proposals that require modifications to these protocols imply changes to kernel code. This immediately restricts deployment opportunities. By limiting changes to code that runs in user-space on end stations, new protocols can be developed and tested on live networks.

TCP is a reliable end-to-end unicast transfer protocol implemented in host kernels. It is possible to modify TCP behaviour between end stations without any changes to intermediate devices, however this requires kernel changes. If TCP is moved into user-space, changes can be made without modifying the kernel, but again, some form of privileged access is needed by the user-space TCP implementation to directly send and receive packets on a host's network interfaces. While not as significant a barrier to widespread deployment as kernel changes, this privileged access requirement severely limits the ease with which new code can be widely tested.

One solution to this problem is to implement a modified multicast TCP over UDP. User-space applications can freely send and receive UDP packets, so a small shim layer can be introduced to encapsulate and decapsulate the TCP-like engine's packets into UDP. While there are performance implications by running in userspace, the instant deployment potential of a userspace library, coupled with the scalability of multicast, mean that any such limitations are more than acceptable.

Therefore, it is possible to:

- build a new end-to-end protocol, implemented as a userspace library
- allow Grid apps to avail of library
- have the option of moving the library into the kernel in future

The key advantage of this approach is that any Grid application can make use of this new protocol by simply linking against the supplied library.

No changes are required in the network (other than enabling IP multicast).

Because the protocol is not tightly coupled to the application, should it become adopted for widespread use, a native implementation can be built in the kernel to boost performance.

4 Previous work

4.1 Reliable Multicast

Reliable multicast has proved to be a difficult problem to solve, and over the last decade, much research has been carried out into how best to approach this problem.

Scalable Reliable Multicast [12] demonstrated how application level framing techniques could be used to provide reliable delivery of data using multicast, while the classic problem of ACK implosion was tackled by The Reliable Multicast Transport Protocol [15] with its hierarchy of designated receivers that could also cache data.

Receiver-driven Layer Multicast [16] introduced the concept of a receiver-driven approach where receivers could join a subset of groups leading to the best local reception of transmitted multicast packets.

The Reliable Multicast data Distribution Protocol [19] showed how Forward Error Correction (FEC) techniques along with NACK processing could be used to efficiently implement high-speed reliable data transfers.

The subsequent Digital Fountain Approach [5] generalized such FEC techniques and introduced its 'Tornado Codes' that would allow source data to be reconstructed from any subset of multicast packets equal in length to the original source data.

The issue of congestion control and fairness alongside TCP flows has been addressed by both PGM's pgmcc [18] and TFMCC [25, 24] for single rate transmissions. While multi-rate transmission have been tackled by [21].

Given such a wide range of approaches, and the realization that a one-size-fits-all solution would not be forthcoming for reliable multicast, the IETF formulated a series of guidelines. These include the criteria used in evaluating reliable multicast techniques (RFC2357), the design space for bulk data transfer (RFC2887), and a set of building blocks or common components essential to building a reliable multicast transfer protocol (RFC3048).

All of the above build on the existing IP multicast model, but some work has been done outside of this model and has proposed to replace it, or augment it, with a simpler multicast system that does not have group addresses. Multiple destination addresses are simply placed in the IP packet header. Routers along the path strip these out accordingly.

Sender Directed Broadcast Mode (RFC1770) was an early military proposal that hinted at this approach, while more recently, Small Group Multicast [3] and Sender-initiated Multicast [22] have eventually led to Xcast [2].

Some hybrid models have been proposed including IRMA [13] which use a hierarchy for ACK aggregation, but allows TCP connections to group addresses, but only when network routers are present that can translate these new TCP packets into conventional unicast TCP packets.

4.2 Multicast TCP

Some existing work has been done in proposing, or indeed modifying, TCP to support multicast communications.

UCL Extension [6] - UCL's Simple TCP Extension was a proposal to extend TCP to allow a connection from one application to many. SYN packets and data packets are sent to a group address. A number of mechanisms for dealing with internal data structures, API and dynamic window adjustment are suggested.

SCE [20] - Single Connection Emulation is a sublayer between the unicast transport layer and the multicast network layer i.e. between TCP and IP. This new layer allows an application to open a reliable connection using TCP to a group address. The SCE intercepts this connection and provides the mechanism necessary to send and receive data over the underlying multicast IP layer while fooling the upper TCP layer into believing that a single connection exists.

TCP-SMO [14] - Single-source Multicast Optimization extends TCP to allow senders and receivers to incorporate multicast transmission when communicating. A single-source server can maintain a common multicast channel that is associated with n unicast channels. The work addresses problems such as connection management, send window advancement, ACK processing, RTT estimation and packet retransmission, and congestion and flow control.

M/TCP [23] - Multicast-extension to TCP is an extension to TCP that introduces multicast capability, however not conventional IP multicast, but small group multicast schemes such as SIM or Xcast where the sender attaches a list of receiver addresses onto the packet header. M/TCP is sender-initiated, requires no changes at the receiver, and provides multiple transmission rates by classifying receivers into multiple groups.

4.3 Userspace TCP

Moving TCP out of the kernel into userspace is not a new idea. A number of projects have done this in the past, either as a byproduct of a larger project, or as an end in itself. Existing work includes:

Alpine [11] - this is a user-level implementation of the TCP stack from FreeBSD 3.3. A number of modifications were necessary, but these were made with easy integration into the kernel in mind.

Arsenic [17] - as part of gigabit ethernet work, a user-level IP networking library based on the protocol code from Linux 2.3.29 was built.

atou [8] - a userspace implementation of a TCP-like protocol that runs over UDP. It provides a way to evaluate variations of TCP over the Internet, however it is not a library with an interface, so applications cannot be compiled using atou.

HP [10] - Edwards and Muir used HP-UX 9.01 as a basis for this implementation as part of the Jetstream experimental gigabit network.

INRIA [4] - a TCP implementation by Braun et al built on SunOS 4.1.3 and AIX 3.2.5. TCP has been divided into two parts: kernel & userspace.

lwIP [9] - a small independent implementation of the TCP/IP stack.

Minet [7] - a user-level stack implemented for teaching.

Of these, lwIP is of particular interest. lwIP is full TCP/IP stack, but unlike BSD or Linux TCP implementations, it does not need to be extracted from a kernel. It can run independently with minor modifications.

5 The lwIP TCP/IP stack

lwIP is a small independent implementation of the TCP/IP protocol stack that has been developed by Adam Dunkels at the Swedish Institute of Computer Science (SICS).

The original focus of lwIP was to reduce RAM usage while still having a full TCP implementation. This made lwIP suitable for use in embedded systems with limited memory resources. Now open source, these small scale goals still remain.

5.1 lwIP modifications

Two modifications are essential at the outset:

1. Link library - the lwIP code has to be wrapped into a library that can be linked into userspace applications.
2. UDP driver - to facilitate operation in userspace, the stack has to run over UDP, hence an lwIP over UDP network interface is necessary.

An application linking the lwIP code runs as three threads. One thread runs the UDP driver to handle packets coming to and from other hosts; a second thread looks after the operation of the protocol; the third thread is the application mainline itself.

6 TCP-XM

This section describes TCP-XM - a version of TCP that has been modified and extended to support multicast data transfers.

The key difference between TCP and TCP-XM is that TCP-XM can open a connection to multiple destinations simultaneously, and choose between unicast and multicast for transmission of data segments to these destinations.

Some assumptions must be made:

- Sender initiated
- Primarily aimed at push applications
- Sender has advance knowledge of unicast destination addresses

Given these assumptions, the "call process" for a TCP-XM sender is as follows:

1. Application attempts connection to multiple unicast addresses
2. A random group address is allocated if not user-specified
   (and SSM is not available)
3. Multiple TCP PCBs are created
4. Independent 3-way handshakes
5. Group address sent as option in SYN
   Multicast dependent on presence of group option in SYNACK

PCB variable then dictates transmission mode for PCB

6. User data writes are replicated & enqueued on all PCB send queues
7. Data packets are multicast (or unicast) Packets have protocol type TCP, but destination is a multicast group
8. All retransmissions are unicast
9. Auto-switch to unicast transmission after n failed multicasts
10. Unicast & multicast sequence numbers stay synchronized
$min(cwnd)$ & $min(snd\_wnd)$ used
11. Connections closed as per TCP

From the receiver's perspective:

1. No API change is necessary
2. Normal TCP listen takes place
3. On TCP-XM connection, IGMP join
4. Accept data on both unicast and multicast addresses

6.1 Implementation

The protocol as described above has been implemented as a modification to lwIP and compiled and tested on FreeBSD and RedHat Linux. This section discusses some of the changes necessary.

6.1.1 API

The only API changes necessary for TCP-XM are when the sender initiates a new connection. The user needs a mechanism for specifying multiple destination addresses, and an optional multicast group address.

lwIP applications typically use its "netconn" API, although a socket API is also available. To open a new TCP-XM connection with the "netconn" API:

```
conn = netconn_new(NETCONN_TCPXM);
netconn_connectxm(conn,
  remotedest, numdests, group, port);
```

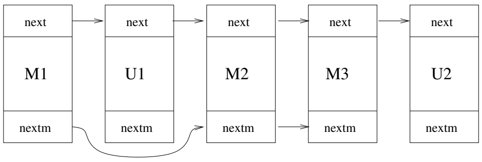

Figure 1: Multicast TCP PCBs

Instead of `netconn_connect()`, three argument changes are introduced for the `netconn_connectxm()` call: an array is used to specify destination address(es); the number of destination addresses is supplied; and an optional group address is supplied.

6.1.2 PCBs

TCP PCBs are stored on a linked list. This has not been changed, however some extra variables have been added to the PCB structure:

```
struct tcp_pcb {
    ...
    struct ip_addr group_ip;
    enum tx_mode   txmode;
    u8t            nrtxm;
    struct tcp_pcb *nextm;
}
```

The group address and transmission mode are dependent on the TCP group option discussed in the next section.

The `nrtxm` variable keeps track of how many times a unicast retransmission has been necessary for segments that have previously been multicast. Too many of these retransmissions will result in the transmission mode of the PCB falling back to unicast.

The `nextm` pointer references the next PCB that is part of a chain of PCBs associated with a single TCP-XM connection. Figure 1 shows a list of five PCBs – M1, M2 and M3 are all part of a single TCP-XM connection, while U1 and U2 are separate independent unicast connections.

6.1.3 TCP Group Option

When a TCP-XM connection is made, a SYN packet is sent that includes a TCP option specifying a multicast group address. If not

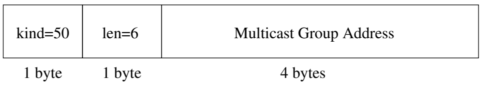

Figure 2: TCP Group Option

specified by the user, the group address is automatically chosen by the protocol. The presence of this option means that the originating host has TCP-XM capability. Figure 2 illustrates this option.

If the receiver is running TCP-XM, it can accept the group address offered or, in exceptional circumstances, return the SYNACK with an alternative proposed group address. On connection establishment, an IGMP join request is issued.

The PCB `txmode` variable's initial value is dependent on the SYNACK. Its value can be one of:

TX_MULTICAST - the SYNACK contained the group option

TX_UNICAST - the SYNACK did not contain the group option

TX_UNICAST_GROUP - the SYNACK contained the group option, but multicast transmission failures have caused a temporary fallback to unicast

If multicast is not possible, the PCB will immediately go into TX_UNICAST state. No attempt will be made to synchronize segment sequence numbers with other PCBs in the connection. If multicast is working, TX_MULTICAST is used, and sequence numbers will be kept in line. However, if multicast fails to a particular destination, the PCB can be moved into TX_UNICAST_GROUP mode whereby data segments will be unicast, but the sequence numbers will be kept in line with those being multicast. The reason for this is to keep the option of switching back to TX_MULTICAST mode a possibility.

6.2 Initial test results

Figures 3 and 4 show some initial results. The tests were run between desktop machines on the Computer Laboratory LAN, so a number of variable factors including processor speeds and versions of Linux had an impact on the results.

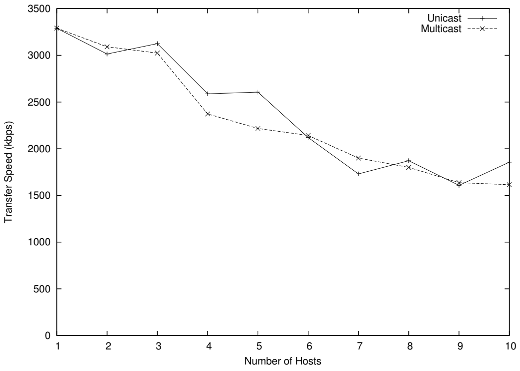

Figure 3: Speed: Unicast vs Multicast

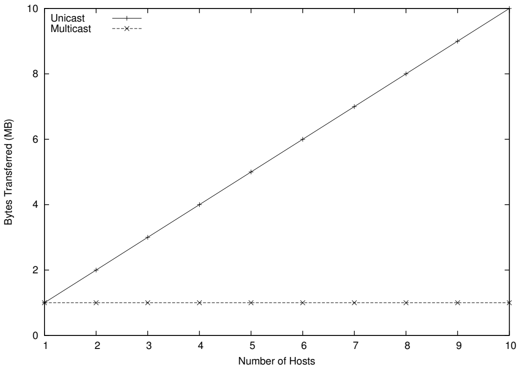

Figure 4: Efficiency: Unicast vs Multicast

Figure 3 shows the bits/sec achieved on transfers using individual TCP-XM connections and single TCP-XM connections using multiple destinations. Figure 4 shows the number of bytes transferred using both unicast and multicast.

Connections to multiple hosts using individual TCP-XM connections have their own distinct window variables, while a single TCP-XM connection to a group of hosts shares a common set of variables across destinations.

As can be seen from the graphs, the key benefit from multicast comes in the reduction in bytes transmitted on the network. The similarity in speed between unicast and multicast can partly be accounted for by the fact that there is no immediate bottleneck on the LAN, and also because the code in use is at an alpha stage.

7 Future work

In addition to Globus integration, future work revolves around how unicast and multicast transmissions modes can overlap and be combined to produce a mechanism that makes efficient use of partial multicast connectivity in a network. In particular, three techniques can be used as a starting point: Fallback / Fall Forward, Look Ahead, and Multiple Groups.

7.1 Fallback & fall forward

By default, all initial transmissions of data segments use multicast, but fallback to unicast transmission will take place should the destination consistently signal that segments are not being received.

How and when group members fallback, and indeed fall forward to rejoin multicast transmission, is an important part of how the protocol will operate. A mechanism is required to ensure that both sender and receiver make intelligent use of all data transmitted, and can make an informed decision about each other's state based on the pattern of packets received.

7.2 Look ahead

Instead of multicasting, the second approach unicasts packets by default, but a look-ahead scheme is used to multicast packets in advance.

When a sender connects to a group of destination hosts, the TCP PCB variables are synchronized. When sending data, the same segment sequence numbers will be sent to hosts at the same time. Given this synchronization, it is possible to use multicast to "send ahead". For example, before unicasting segments 1, 2 & 3, segments 4 & 5 could be multicast. Destination hosts will buffer the out-of-sequence segments, and await delivery of the early (expected) segments via unicast.

7.3 Multiple Groups

The third approach is essentially a variation on the first two. Rather than confine multicast transmission to a single group, destinations can be split into a number of different groups with similar characteristics.

7.4 Globus & The Grid

The Globus Toolkit is the de-facto standard for middleware used to implement Grid services. At present, all bulk data transfer is carried out using the GridFTP protocol [1]. This is based on the conventional FTP protocol, but includes some extra features to optimize bulk data transfer e.g. parallel data streams.

Implementations that make use of the protocol must support the Grid Security Infrastructure (GSI) so that user authentication can take place using Grid certificates.

A number of approaches can be be considered for the integration of the TCP-XM protocol into a Grid / Globus environment.

1. Modify the GridFTP protocol: it would be possible to take the existing GridFTP protocol and modify it to support multicast transfers.
2. Modify internal Globus components: the Global Access to Secondary Storage and `globus_io` libraries provide application programs with access to lower layer protocols such as GridFTP.
3. Add GSI support to an existing application: an existing file transfer application that uses the TCP-XM protocol can be integrated into a Grid environment simply by adding support for the Grid Security Infrastructure.

Further investigation is required in choosing the most suitable integration method.

8 Conclusion

We have described the work to date on the design and implementation of the TCP-XM protocol. By implementing this protocol as a userspace library above UDP, we are in a position to test the operation of the protocol in live networks, while delivering a reliable multicast file transfer application to the Grid community. Future protocol tests will be carried out between eScience centres around the UK.

References

[1] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, S. Meder, and S. Tuecke. GridFTP Protocol Specification. GGF GridFTP Working Group Document, September 2002.

[2] R. Boivie, N. Feldman, Y. Imai, W. Livens, D. Ooms, and O. Paridaens. Explicit Multicast (Xcast) Basic Specification. *IETF draft-ooms-xcast-basic-spec-04*, January 2003.

[3] Rick Boivie and Nancy Feldman. Small Group Multicast. *IETF draft-boivie-sgm-02*, February 2001.

[4] Torsten Braun, Christophe Diot, Anna Hoglander, and Vincent Roca. An Experimental User Level Implementation of TCP. Technical report, INRIA RR-2650, September 1995.

[5] John W. Byers, Michael Luby, Michael Mitzenmacher, and Ashutosh Rege. A Digital Fountain Approach to Reliable Distribution of Bulk Data. In *SIGCOMM*, pages 56–67, 1998.

[6] Jon Crowcroft, Zheng Wang, and Ian Wakeman. A Simple TCP Extension to Achieve Reliable 1 to Many Multicast. University College London, Internal Note, March 1992.

[7] Peter Dinda. The Minet TCP/IP Stack. Technical report, Northwestern University, NWU-CS-02-08, January 2002.

[8] Tom Dunigan and Florence Fowler. A TCP-over-UDP Test Harness. Technical report, Oak Ridge National Laboratory, ORNL/TM-2002/76, May 2002.

[9] Adam Dunkels. Minimal TCP/IP implementation with proxy support. Technical report, Swedish Institute of Computer Science, SICS-T-2001/20-SE, February 2001.

[10] Aled Edwards and Steve Muir. Experiences Implementing a High-Performance TCP in User-Space. Technical report, HP Laboratories Bristol, HPL-95-110, September 1995.

[11] David Ely, Stefan Savage, and David Wetherall. Alpine: A User-Level Infrastructure for Network Protocol Development. In *Proceedings of USENIX USITS*, 2001.

[12] Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, December 1997.

[13] Kang-Won Lee, Sungwon Ha, and Vaduvur Bharghavan. IRMA: A Reliable Multicast Architecture for the Internet. In *INFOCOM (3)*, pages 1274–1281, 1999.

[14] Sam Liang and David Cheriton. TCP-SMO: Extending TCP to Support Medium-Scale Multicast Applications. In *Proceedings of IEEE INFOCOM*, 2002.

[15] John C. Lin and Sanjoy Paul. RMTP: A Reliable Multicast Transport Protocol. In *INFOCOM*, pages 1414–1424, San Francisco, CA, March 1996.

[16] Steven McCanne, Van Jacobson, and Martin Vetterli. Receiver-driven Layered Multicast. In *ACM SIGCOMM*, volume 26,4, pages 117–130, New York, August 1996. ACM Press.

[17] Ian Pratt and Keir Fraser. Arsenic: A User-Accessible Gigabit Ethernet Interface. In *Proceedings of IEEE INFOCOM*, 2001.

[18] Luigi Rizzo. pgmcc: a TCP-friendly single-rate multicast. In *SIGCOMM*, pages 17–28, 2000.

[19] Luigi Rizzo and Lorenzo Vicisano. A Reliable Multicast data Distribution Protocol based on software FEC techniques. In *The Fourth IEEE Workshop on the Architecture and Implementation of High Performance Communication Systems (HPCS'97)*, Sani Beach, Chalkidiki, Greece, June 1997.

[20] Rajesh Talpade and Mostafa H. Ammar. Single Connection Emulation: An Architecture for Providing a Reliable Multicast Transport Service. In *Proceedings of 15th IEEE Intl Conf on Distributed Computing Systems*, Vancouver, June 1995.

[21] Lorenzo Vicisano, Luigi Rizzo, and Jon Crowcroft. TCP-Like Congestion Control for Layered Multicast Data Transfer. In *INFOCOM (3)*, pages 996–1003, 1998.

[22] Vasaka Visoottiviseth, Hiroyuki Kido, Youki Kadobayashi, and Suguru Yamaguchi. Sender-initiated multicast forwarding scheme. In *Proceedings of IEEE ICT2003*, Tahiti, February 2003.

[23] Vasaka Visoottiviseth, Takuya Mogami, Noritoshi Demizu, Youki Kadobayashi, and Suguru Yamaguchi. M/TCP: The Multicast-extension to Transmission Control Protocol. In *Proceedings of ICACT2001*, Muju, Korea, February 2001.

[24] Joerg Widmer and Mark Handley. TCP-Friendly Multicast Congestion Control (TFMCC) Protocol Specification. *IETF draft-ietf-rmt-bb-tfmc-01*, November 2001.

[25] Jörg Widmer and Mark Handley. Extending Equation-Based Congestion Control to Multicast Applications. In *SIGCOMM*, pages 275–286, 2001.