A Reliable Message Multicast Transport Protocol for Virtual Circuits

Jie Li and Malathi Veeraraghavan
University of Virginia
Charlottesville, USA

***Abstract*—Motivated by the need to distribute large volumes of scientific data to large numbers of subscribers, we propose a reliable multicast transport protocol. Specifically, this protocol is developed for use on virtual circuits, since dynamic circuit services are now being offered by large providers, and virtual circuit networking is well suited to multicasting as it eliminates the data-plane congestion control problem of connectionless IP. The new protocol is called Virtual Circuit Multicast Transport Protocol (VCMTP). A key concept is to execute retransmissions (required due to flow control problems) at the end, i.e., after the message (file or memory data) is multicast. This leads to scalability, where the throughput for the receivers that can keep pace with the sending rate is independent of the number of receivers. A prototype was tested on Emulab, and measurements obtained. Our findings are that for disk-to-disk file transfers at a sending rate of 600 Mbps, the Emulab hosts can support multicasting with less than 0.5% retransmission rates, but at a 800 Mbps sending rate, the average throughput is only 650 Mbps because the retransmission rate increases to 9%.**

***Index Terms*—Reliable multicast; transport protocols; high speed; virtual circuits**

I. Introduction

Scientific data collected by instruments, from experimental studies, and from simulations executed on high-performance computing platforms, often need to be distributed across the Internet. Some types of scientific data are collected and distributed on a daily basis. For example, the University Corporation for Atmospheric Research (UCAR) Internet Data Distribution (IDD) project [1] distributes large amounts of meteorology data on a near real-time basis, on the order of 10 GB/hour, to a subscriber base of 170 institutions. Over 30 types of data products (referred to as *feedtypes*) are distributed through the IDD project. Examples include CONDUIT high-resolution model data from the National Centers for Environmental Prediction (NCEP), and NEXRAD-II, which is radar data. The software used for this data distribution is called Local Data Manager (LDM) [2]. LDM uses RPC based communications, and allows for a downstream server to subscribe to feedtypes from upstream servers in a tree structure. Different feedtypes have different tree structures. For example, the CONDUIT feedtype has 163 subscribers, of which 103 are served directly by the UCAR LDM servers, and 57 of these, in turn, serve other receivers. Unicast TCP connections are used between all upstream and downstream LDM servers.

While the use of unicast TCP connections over the basic IP-routed service has the ease-of-use advantage, there are disadvantages to this mode of data delivery. The IDD project requires 9 LDM servers at UCAR, and generates 5 Gbps of traffic on the 10 Gbps UCAR access link. Also, the latency of delivering the data products is sensitive to the number of receivers. While this particular application is not highly latency sensitive, in other applications, e.g., financial data distribution, this can be an issue. In addition to scientific data distribution of the type described above, newspapers and magazines could also leverage such multicast services for daily/weekly data distribution.

Therefore, the *problem statement* of this work is to design, prototype, and evaluate a multicast transport protocol for reliable message transfer across virtual circuits. First consider the *multicast* aspect. If a sender can simultaneously send a message to multiple receivers, the server capacity and link bandwidth required can be lowered. Second consider the term "reliable message transfer." Reliability is required as this protocol is designed for data distribution, and not audio/video transmission. The reason for using the term "message" is that individual files, or data products in IDD terminology, can be regarded as messages. Contrast message transport service as offered by UDP with byte stream transport service offered by TCP. Our requirement is to transfer messages reliably to large numbers of receivers, not byte streams.

Finally, consider our proposed usage of *Virtual Circuits (VC)*. We have two reasons for considering virtual circuits. First, for the IDD project, several feed types generate data almost continually, making the use of virtual circuits more suitable than IP routed service. Second, VC networking shifts the burden of congestion control to the control plane. VC networks have a connection setup phase in which requests for bandwidth are accepted/rejected based on resource availability. As long as the sending rates are limited to the VC rates, there is no possibility of congestion in the data plane. This simplifies the reliable multicast transport protocol problem to just handling bit errors, and flow control problems (receiver buffer overflows). Other reliable multicast proposals, such as the Scalable Reliable Multicast (SRM) framework [3], note that multicast congestion control, which is required if the underlying network service is a connectionless service such as the IP-routed service, is a difficult proposition. For example, should the sending rate be tuned to that of the worst-case receiver? One approach proposed in SRM is to use reserved resources as with the Integrated Services (IntServ) architecture [4], which is a VC networking solution. With

the growing interest in dynamic (virtual) circuit services [5], especially in the scientific computing community, we hence propose to target our reliable message transport protocol for virtual circuits. Technologies that are commonly used today for virtual circuits include MultiProtocol Label Switching (MPLS) and Ethernet Virtual LANs (VLANs), both of which support multicasting, as with MPLS virtual private networks (VPNs), and with multipoint VLANs.

Our key contributions are the design, prototyping, and evaluation of a new transport protocol called Virtual Circuit Multicast Transport Protocol (VCMTP). The main design concept is to transmit the whole message first and then handle retransmissions. This is coupled with the concept of bandwidth adaptation as proposed in [6], whereby a multitasking receiver could change multicast groups to receive data on a lower-rate VC if it experiences consistent receive buffer overflows. By controlling membership of receivers and using multiple (different-rate) virtual circuits with the sending rates tuned to the VC rates, only few retransmissions are likely to be required. These are handled at the end to avoid one or more slow receivers from slowing down data reception for the other receivers.

Section II provides a brief overview of prior work on reliable multicast protocols. Section III discusses the overall design of VCMTP, and Section IV presents a prototype implementation. Section V presents the experimental settings and evaluation results with the prototype. Section VI concludes this work.

II. Related Work

Reliable multicast transport protocols have been proposed since the mid-1990s [3], [7]–[11]. Several concepts developed in these proposals are adopted in our work. For example, the session manager in RMTP [7] is similar to our VCMTP multicast manager, as is the use of receivers tracking the status of successful or failed packet transmissions instead of the sender. However, there are two key differences between RMTP and VCMTP. Positive ACKnowledgments (ACKs) are used in RMTP with a tree of designated receivers handling ACKs to avoid the ACK implosion problem [8], while VCMTP uses negative acknowledgements (NACKs). Second, RMTP uses window-based flow control, while VCMTP uses a flow-control problem avoidance technique through control-plane methods.

In SRM both repair requests and retransmissions are multicast to the entire group. Repair requests differ from negative ACKs (NACKs) in that the latter are sent to a specific sender, while repair requests are sent to all participants. While this approach will be considered in future work, VCMTP uses TCP unicast connections for the NACKs and retransmissions, as proposed in MTP-2 [9].

The NACK-Oriented Reliable Multicast (NORM) protocol is described in [11]. It uses FEC to improve reliability and "uses probabilistic suppression of redundant feedback based on exponentially distributed random backoff timers" for scalability. A TCP friendly congestion control algorithm is proposed as NORM is developed for multicast IP.

A difference between MTP-2 and VCMTP is that the former claims scalability only under no loss scenarios, while in VCMTP using our design concept of completing the whole message transmission before executing retransmissions, we aim for scalability in the presence of losses. This concept is similar to that used in Reliable Blast UDP (RBUDP) in [12], which is a unicast transport protocol for use across dedicated optical circuits.

A Reliable Adaptive Multicast Protocol (RAMP), described in IETF RFC 1458, was enhanced for use over an all-optical, circuit-switched, gigabit network in an ARPA-sponsored Testbed for Optical NEtworking (TBONE) [10]. The question of not slowing down other receivers due to flow control problems in some slow receivers is not addressed; instead retransmissions are executed immediately after reports of losses.

III. Virtual Circuit Multicast Transport Protocol

The two objectives in designing VCMTP are reliability and scalability. As with other reliable transport protocols such as TCP, VCMTP will include error control and flow control functions. Congestion control is not required in the data plane because congestion is handled in the control plane during VC setup (VC setup requests will be rejected if all bandwidth resources are used up for existing virtual circuits). For *error control*, to avoid senders having to maintain state information about every receiver, retransmission requests will be receiver driven through NACKs. This avoids the (positive) acknowledgement implosion problem in which the sender host is overloaded by acknowledgement messages from a large number of receivers. For packet retransmissions to individual receivers, MVCTP will use unicast TCP connections over IP-routed paths. To achieve scalability, unlike in TCP in which packet retransmissions are interleaved with the main data transfer process, the VCMTP sender executes retransmissions only after the whole message is multicast to all receivers. When receivers detect packet loss during the main data transfer period, the receivers immediately send retransmission requests to the sender, but the sender stores the requests and executes them at the end. The sender unicasts retransmissions to each receiver on individual TCP connections. This approach prevents the retransmission process from slowing down the main data transmission process at the sender, which can be a problem at high speed, and when the number of receivers is large. Only those receivers that have packet loss will experience a reduction in the overall throughput (because of the retransmissions at the end), while other receivers that can keep up with the sending rate will experience the higher throughput.

As the sender receives TCP connect requests, it sends a control message on the TCP connections informing each receiver of all available VCs with their corresponding rates. If a receiver consistently experiences a high packet loss rate, a VCMTP multicast manager can have it switch to receiving data on a lower-rate VC if there is one.

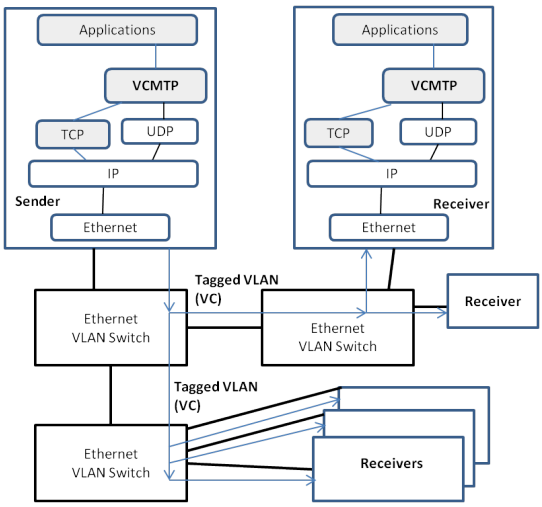

Fig. 1: An example configuration

For *flow control*, there is choice of three mechanisms: window-based, rate-based, and on/off [13]. Window-based flow control is not suitable for multicast because the free-space available in the receive buffer will differ from one receiver to the next. Rate-based flow control allows a sender to adjust its sending rate dynamically. Such a scheme would again be difficult to implement in a multicast scenario as each receiver's receive rate can vary dynamically due to multitasking. The on/off mechanism allows a receiver to send an on/off message to the sender based on its buffer occupancy. The sender will start/stop sending data as per these control messages. Again, this scheme will slow down the reception rate for the fast receivers.

The solution to the flow control proposed for VCMTP is as follows. First, the sending rates of multiple VCs used to transmit the same data are sent to all receivers in the control plane allowing each receiver to choose the multicast group corresponding to the VC whose rate is less than or equal to the rate it can sustain. Measurements are required at the receiver to have it continually evaluate whether its multicast receiving process is able to keep up with the selected rate. If it finds itself losing too many packets from its receive buffer, it will switch to a lower-rate VC if one is available. In spite of this arrangement, packets are still likely to be lost due to multitasking, and these losses are handled with NACKs and retransmissions.

IV. VCMTP Prototype

Fig. 1 illustrates how an Ethernet VLAN based virtual circuit is provisioned between a sender and multiple receivers. Tagged VLANs with rate policing and scheduling are used to create rate-guaranteed virtual circuits. Other VC technologies such as MPLS can also be used. We use VLANs just to illustrate the concept. These virtual circuits are reserved and provisioned by schedulers such as the ESnet On-Demand Secure Circuits and Advance Reservation System (OSCARS) [14]. Class-D multicast IP addresses in the (224 to 239)/8 range will be assigned, one per multipoint VLAN. Each receiver binds a UDP socket to this address. As shown in Fig. 1, the multicast forwarding action is performed in the Ethernet switches on the VLAN ID and multicast MAC address, which is derived from the Class D multicast IP address.

The inner blocks shown with the sender and a receiver in Fig. 1 illustrate the concept that VCMTP uses UDP for the multicast transmission of the message (the arrows show the flow of this transmission), and TCP connections to send/receive NACKs and retransmissions. VCMTP adopts a block-oriented data transfer model in which a message (disk file or memory data) on the sender is first fragmented into multiple blocks (VCMTP packets), which are then encapsulated into UDP datagrams for multicasting. Upon receiving a multicast packet, the receiver process extracts the VCMTP packet from the UDP datagram, and then adds the VCMTP payload to the corresponding data block given its position in the message. Unlike the byte streaming transfer model, the block-oriented data transfer model does not require data to be passed to the application in sequence. However, it requires that receivers be aware of the length of the message before the transfer, so that they can allocate the data storage (either on disk or in memory) correspondingly. In VCMTP, this information is communicated between the sender and all the receivers over the TCP connections before the message is multicast.

A. Sender Implementation

The VCMTP sender is implemented as a user-space library. The `VCMTPSender` class provides a set of APIs that are related to both control-plane functions, such as initializing a VCMTP multicast group, and data-plane functions, such as sending data to a multicast group. The application issues a VCMTP `JoinGroup` function call specifying a Class D IP address, allowing the VCMTP code to open a UDP socket using that IP address. The VCMTP code also starts a `VCMTP retransmission thread`, whose functions are to listen for TCP connection requests from multicast receivers, and to handle NACKs and retransmissions.

To multicast a message, the application can call one of two VCMTP functions in the `VCMTPSender` class:

1) *int VCMTPSender::Send(void* buffer, size_t length)*
2) *int VCMTPSender::Send(char* file_name)*

The first API is used for transferring in-memory data, and the second API is used for transferring disk files. For the second call, the VCMTP send function reads directly from the disk. In both functions, the sender first determines the size of the message and communicates this information to all the receivers that are connected to it via TCP connections. It then divides the data into blocks that can fit into the payload of a VCMTP packet. The VCMTP header includes a source port number, destination port number, block sequence number, and payload length. The VCMTP `send` function writes the VCMTP packets to the UDP socket.

The `select()` system call is used in the VCMTP retransmission thread for handling unicast TCP connections

from all receivers. As it receives NACKs, it stores the block number with the receiver TCP socket ID in a buffer called a *retransmission store*. Retransmissions are executed at the end of the message multicast on a receiver-by-receiver basis, where the receiver with the smallest number of missing blocks will be served first. The retransmission store maintains in a memory cache the missing blocks that have been retransmitted to receivers so far. This design allows the sender to leverage memory caching to retransmit a missing block to multiple receivers efficiently. For messages stored on disk that are larger than the cache, disk reads are required with a specific offset for each block that needs retransmission. For large files, this design has a disadvantage when compared to the TCP approach in which data read from disks are held in a retransmission buffer in memory and retransmissions are completed immediately after loss detection.

B. Receiver Implementation

The receiving application calls a `VCMTP Receiver` method with a set of class D IP addresses, allowing the latter to bind one UDP socket for each Class D IP addresses corresponding to each of the virtual circuits (recall the concept of using different-rate circuits to handle the flow control problem). It also starts a `VCMTP retransmission thread` to send NACKs and receive retransmissions. Upon receiving a UDP datagram carrying a VCMTP packet, it compares the current received block sequence number with that of the last received block sequence number to determine if there is a missing packet. If a packet loss is detected, the receiver stores the block number in a retransmission store buffer and sends it in a NACK to the VCMTP sender. For the correctly received VCMTP packet, it extracts the payload and copies it to the specified application data block, which involves a disk write for a file transfer. The VCMTP retransmission thread handles all retransmissions received on the TCP connections, and likewise writes the payloads into the application data block.

C. VCMTP Multicast Manager

Besides the VCMTP sender and receivers, a separate component called the *VCMTP Multicast Manager* is implemented to support a set of management-plane functions, such as performance monitoring, fault management, and configuration management. The performance monitoring module keeps track of the retransmission rates of receivers, and initiates a receiver's switch to a lower-rate VC when needed. The fault management module detects exception conditions during the message multicast, and initiates recovery. The configuration management handles the setup of VCs through circuit schedulers such as OSCARS.

V. Evaluation

The VCMTP prototype was tested on the Emulab testbed [15]. We chose hosts with 1 Gbps Ethernet NICs to execute relatively high-speed experiments up to 800 Mbps. The nodes have 2 GB memory and commodity disk facilities. Section V-A describes a multicast experiment that illustrates a positive aspect of the VCMTP design, while Section V-B describes a unicast experiment that demonstrates a negative feature.

TABLE I: VCMTP Multicast Throughput (Unit: Mbps)

	512 MB	**1 GB**	**2 GB**	**4 GB**
Avg. (SD) throughput of receptions in no-loss runs	579.49 (1.73)	574.56 (1.60)	588.25 (0.30)	582.17 (0.87)
Avg. (SD) throughput of no-loss receptions in loss runs	N/A	575.65 (0.81)	588.27 (0.74)	582.22 (0.98)
Avg. (SD) throughput of loss receptions in loss runs	N/A	561.4 (1.73)	580.32 (4.94)	576.1 (4.43)

A. Multicast Performance

In this experiment, one VCMTP sender multicasts disk files of different sizes to 7 receiver nodes. The sending rate is set to 600 Mbps, a rate at which there are only a few receive buffer overflows, if any. For each file size, the experiment was repeated 10 times, and the average throughput was computed. The results are shown in Table I. For the 512 MB file size, there were no retransmissions in all 70 receptions (7 receivers and 10 runs). For the other three file sizes, in some runs there were no losses, while in other runs, one or more receivers experienced losses. The first row shows the throughput values for the no-loss runs, while the second and third rows show the throughput values for the runs in which there were losses, with the averaging done over receptions that had no losses, and receptions that had losses, respectively. Table I shows that the average throughput of the receptions in no-loss runs is very close to that of the no-loss receptions in the loss runs. Furthermore, the average throughput of loss receptions in loss runs are lower than that of no-loss receptions in both no-loss and loss runs. Thus in a single multicast, if 6 out of the 7 receivers experienced no packet losses, their (higher) throughput was unaffected by the losses incurred in the 7^{th} receiver. This illustrates the key design concept of VCMTP that allows for scalability.

B. Choice of sending rate

This experiment compares unicast TCP with a single-sender/single-receiver VCMTP configuration to illustrate the need for VCMTP receivers to choose a VC whose rate matches the rate at which the VCMTP application can deplete the receive buffer. Disk-to-disk file transfers with different sizes ranging from 512 MB to 4 GB were executed and measurements obtained. Two different sending rates, 600 Mbps and 800 Mbps, were used. For each file size and each experimental setting, 10 runs were executed, and the average throughput computed from the measurements. The results are shown in Fig. 2.

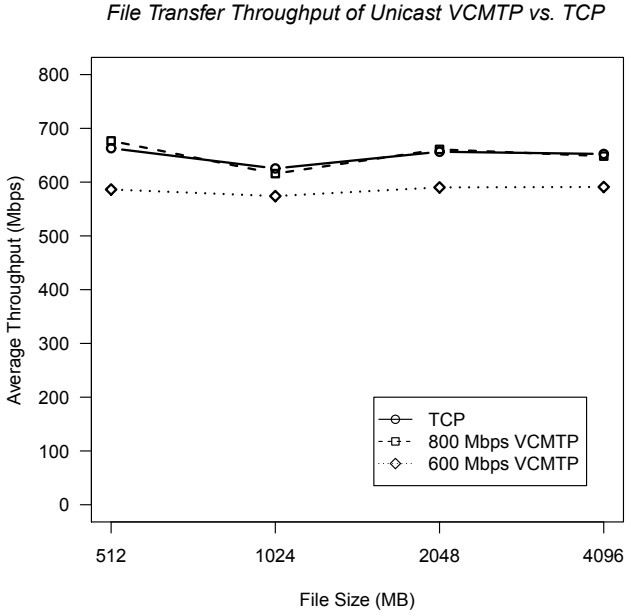

Fig. 2: Unicast File Transfer Throughput for VCMTP vs. TCP

With TCP, the sending rate is automatically determined by the sender as part of the Slow Start/Congestion Avoidance algorithms. As shown in the figure, TCP achieves a throughput of around 650 Mbps for all the file sizes considered. With VCMTP, there is a greater likelihood of receive buffer overflows at 800 Mbps than at 600 Mbps. The average retransmission rates for 512 MB, 1GB, 2 GB, and 4 GB file transfers are 4.46%, 11.04%, 8.63%, and 9.27% at 800 Mbps, and 0%, 0.26%, 0.07%, and 0.19% at 600 Mbps respectively. The 800 Mbps VCMTP transfer achieves the same throughput as TCP throughput as seen in Fig. 2, but the more cautious setting of 600 Mbps yields close to that throughput but with few retransmissions if any.

VI. Conclusions

A new reliable multicast transport protocol was proposed for virtual circuits. This work was motivated by two considerations. First, there is a need for data distribution to large numbers of subscribers both in the scientific community and in the commercial domain. Second, new dynamic circuit services are being offered by both research-and-education network and commercial providers. As virtual circuits (VCs) are provisioned prior to data transfer with a guaranteed rate, there is no congestion in the data plane, making VCs more suitable than multicast IP. A new Virtual Circuit Multicast Transport Protocol (VCMTP) is proposed. A key design concept of executing the whole message multicast before handling retransmission requests provides scalability but does incur a cost due to disk reads when the file size is larger than host memory. A prototype of VCMTP was implemented and evaluated for high-speed file transfers. Our conclusions are that as long as the receivers can estimate the rate at which their receive buffers can be depleted and choose a VC with the corresponding rate, packet losses can be kept minimal. Both the sender computing resources and bandwidth can be reduced through the use of multicasting.

Acknowledgment

This work was supported by the NSF grants OCI-1038058 and OCI-1127340, and DOE grants DE-SC002350 and DE-SC0007341.

References

[1] Internet Data Distribution. [Online]. Available: http://www.unidata.ucar.edu/software/idd/

[2] Local Data Manager. [Online]. Available: http://www.unidata.ucar.edu/software/ldm/

[3] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing," in *ACM SIGCOMM*, Aug. 1995, p. 342.

[4] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC 1633 (Informational), Internet Engineering Task Force, Jun. 1994. [Online]. Available: http://www.ietf.org/rfc/rfc1633.txt

[5] M. Veeraraghavan, M. Karol, and G. Clapp, "Optical dynamic circuit services," in *IEEE Communications Magazine*, vol. 48, Nov. 2010, pp. 109–117.

[6] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '96. New York, NY, USA: ACM, 1996, pp. 117–130. [Online]. Available: http://doi.acm.org/10.1145/248156.248168

[7] S. Paul, K. K. Sabnani, J. C.-H. Lin, and S. Bhattacharyya, "Reliable Multicast Transport Protocol (RMTP)," *IEEE Journal on Selected Areas in Communications*, vol. 15, p. 407, Apr. 1997.

[8] B. Whetten and G. Taskale, "An overview of reliable multicast transport protocol II," *Network, IEEE*, vol. 14, no. 1, pp. 37 –47, jan/feb 2000.

[9] C. Bormann, J. Ott, H.-C. Gehrcke, T. Kerschat, and N. Seifert, "MTP-2: Towards achieving the S.E.R.O. properties for multicast transport," in *Internet Conference on Computer Communications and Networks (ICCCN)*, Sep. 1994.

[10] A. Koifman and S. Zabele, "RAMP: A reliable adaptive multicast protocol," in *IEEE Infocom '96*, Mar. 1996, p. 1142.

[11] B. Adamson, C. Bormann, M. Handley, and J. Macker, "NACK-oriented reliable multicast (NORM) transport protocol," Internet Engineering Task Force, RFC 5740, 2009. [Online]. Available: http://tools.ietf.org/html/rfc5740

[12] E. He, J. Leigh, O. Yu, and T. Defanti, "Reliable blast UDP : predictable high performance bulk data transfer," in *Cluster Computing, 2002. Proceedings. 2002 IEEE International Conference on*, 2002, pp. 317 – 324.

[13] X. Zheng, A. P. Mudambi, and M. Veeraraghavan, "FRTP: Fixed Rate Transport Protocol - A modified version of SABUL for end-to-end circuits," in *Pathnets Workshop, held in conjunction with Broadnets 2004*, Oct. 2004.

[14] Oscars. [Online]. Available: http://www.es.net/OSCARS/docs/index.html

[15] Emulab. [Online]. Available: http://www.emulab.net/