GBT MEMO 182

Reliable Multicast Protocols and their Application on the Green Bank Telescope

Joseph J. Brandt[a]

[a]National Radio Astronomy Observatory, Green Bank, West Virginia USA

ABSTRACT

The Green Bank Telescope (GBT) is under construction in Green Bank, West Virginia. The GBT is expected to be operational in mid 1999. Networking the GBT will require innovative application of network technologies. Distribution of data from the originating source to large groups of consumers can be accomplished in at least two ways. The first is to 'unicast' or communicate individually with each consumer on the net. The second is to 'multicast' (a form of broadcasting) to a group of consumers. Multicasting is a technology which seems to have merit for distribution of information, but it is inherently unreliable. In some cases, such as observational data, packets must be delivered reliably, while in other cases such as informational displays, reliable delivery is not essential. This paper introduces the basic concepts of multicasting, and also discusses the design approaches of reliable multicast protocols.

Keywords: Remote Observing, Multicast protocols, Collaboration tools

1. INTRODUCTION

"The coming of the book must have seemed as if it would turn the world upside down in the way it spread, and above all democratized knowledge."[*]

Like the invention of the printing press, computers and computer communication networks are evolving the way we live, work, and play. Our children are beginning to use computers in the classroom, in some cases linking them to children half a world away. Telecommuting and teleconferencing is beginning to have a significant impact on both business and science. Where it was once impossible to observe without traveling hundreds of miles, it is now possible in some cases to observe from the desktop.

Development projects such as Internet2[†] will provide additional network bandwidth, and the Multicast Initiative[‡] will provide some of the mechanisms to efficiently use it.

Multi-media groupware has been available for some time now. Many of these programs currently use or could benefit from the use of IP multicasting. Although reliable multicast technology is not currently planned for immediate use on the GBT at first light, it is clear that the network design should not preclude its use.

The following sections introduce the concept of multicasting, highlight the design issues for multicast protocols, provide some details about the classification and various approaches to the problem of reliability, and discusses several protocols defined in the literature as they relate to these classifications. This paper is not intended to be a complete taxonomy of reliable multicast protocols, but rather an introduction to a promising new technology.

(Send correspondence to P.O. Box 2, Green Bank, WV 24944 USA): E-mail: jbrandt@nrao.edu

[*] James Burke, *Connections*

[†] Internet2 is a project of the University Corporation for Advanced Internet Development

[‡] MIST is a three year program sponsored by ARPA/ITO investigating the multicast requirements of the Internet and Distributed Interactive Simulation (DIS) environments.

2. MULTICAST

"The most important thing in the programming language is the name. A language will not succeed without a good name. I have recently invented a very good name and now I am looking for a suitable language."[§]

What is Multicast ? There are three basic models of information exchange on the Internet: unicast, broadcast, and multicast. Unicast exchange is analogous to direct person to person email. It represents a one-to-one relationship between sender and receiver. Multicast exchange is much like the Internet newsgroups, representing a one-to-many relationship, where the recipients are a select group of interested subscribers. Continuing with this analogy, broadcast exchanges are much like the dreaded email or newsgroup "bombings", where all recipients are affected, whether they want to be or not, in essence representing a one-to-everybody relationship.

Multicast communications therefore is a mechanism for the simultaneous distribution of information from a source to a group of interested subscribers. *Why is this important?*

Consider a shuttle launch, rock concert, or a Jovian comet impact video data stream being transferred from its source across limited bandwidth Wide Area Network (WAN) links. For one user to tune-in at a remote site requires some portion of the valuable WAN link bandwidth. With the unicast model, since each connection is unique, each additional user requires an equivalent additional amount of WAN link bandwidth. With multicasting, each additional user simply subscribes to the existing multicast video stream, without adding significant additional network load. In this way multicasting is a more efficient method of distributing data.

3. THE OSI NETWORK PROTOCOL MODEL AND MULTICAST

"Applying computer technology is simply finding the right wrench to pound in the correct screw."[¶]

In describing the different modules of a communications protocol, the modules are usually thought of as being broken down into layers. These layers together form what is known as a protocol stack. The OSI model defines these layers as (in order from wire to application): Physical, Link, Network, Transport, and Application. Each layer has a specialized function. (The Physical layer specifies the hardware characteristics of the transmission medium etc., so I will omit it for brevity.)

3.1. The Link layer

The Link layer normally encompasses the hardware device driver and media access control (MAC) addressing. Ethernet[‖] is a commonly used LAN medium that uses a 48 bit MAC address.

Ethernet uses address ranges to indicate the different classes of communication. Ethernet, like other link mediums, has a way to distinguish between unicast, broadcast, and multicast group addresses. Ethernet defines that the least significant bit of the most significant byte be set[**] to indicate a multicast, or all address bits set for a broadcast. With unicast packets the MAC address is completely unique, with only one destination matching the address exactly. Multicast MAC addresses are formed algorithmically, based on the multicast group IP address.

The internal GBT network uses link layer switching to partition traffic from segments not participating in a unicast data transfer. Broadcasts normally are sent to all ports of the switch, flooding all LAN segments. For multicast packets, this presents a problem. By default multicast packets are treated as broadcast packets. To restrict this, GBT LAN switches define *virtual LANs*, or VLANs. A VLAN can be thought of as a broadcast domain, which can span multiple segments, or switches. By the use of VLANs, the GBT network will be able to support fast efficient switching, and optimize (local) multicast packet distribution, without flooding all LAN segments.[1]

[§] D. E. Knuth, 1967

[¶] Author unknown

[‖] Ethernet is a trademark of Xerox Corporation

[**] i.e. the first bit on the wire

3.2. The Network Layer

The network layer (or IP layer in TCP/IP) handles the actual flow of information around the network. IP/Multicast uses class D addresses, those with their high-order bits as 1110, or in Internet form, addresses ranging from 224.0.0.0 to 239.255.255.255. Packets are given a besteffort delivery, with no guarantee of reliability.

Packet routing is accomplished in the network layer. Routing is the mechanism which guides a packet from the source towards its eventual destination, by network equipment called *routers*. Routers interconnect networks forming what we call the Internet. There are several routing protocols in existence to handle the exchange of routing information for unicast traffic[††]. Examples of multicast-specific routing protocols are the Internet Group Management Protocol[2] (IGMP), Multicast Open Shortest Path First[3] (MOSPF), Core Based Trees[4] (CBT), and Distance Vector Multicast Routing Protocol[5] (DVMRP). Routing for multicast packets can be quite complex, due to the fact that a single group address has multiple destinations. Routers must duplicate packets when routes diverge (in contrast unicast packets have a single destination). Multicast routing is a research topic in and of itself, and will not be addressed in this paper. Sites can be also tied together using a technique known as tunneling. The Multicast back-bone or MBone is based upon this concept, where multicast packets are distributed between sites by encapsulation inside unicast packets.

3.3. The Transport Layer

The transport layer provides a mechanism for data flow between hosts on a network. The two most common protocols in this layer are the User Datagram Protocol (UDP) and the Transmission Control Protocol (TCP). TCP provides a reliable point to point (unicast) connection. All information exchanged between hosts is acknowledged, not unlike registered mail. TCP senders maintain information about flow control, state, and timing of its receiver. The responsibility for error detection is placed on the sender. UDP provides a mechanism for applications to send data, with only best effort delivery semantics. This typically means that in the absence of network congestion, overloaded memory buffers, and intermittent network connectivity, all packets will be delivered. Any additional reliability semantics must be implemented by higher layer protocols (or the application itself).

3.4. The Application Layer

The Application layer gives meaning to the actual data exchanged. Typical application layer protocols are SMTP, FTP, Telnet, and RPC. In the Transport and Application layers is where most reliable multicast protocols come into the picture. In some implementations the reliability is built into the transport layer, while others implement it in the application layer.

4. RELIABLE MULTICAST PROTOCOL DESIGN ISSUES

"The process by which fundamental change comes about at times has nothing to do with diligence, or careful observation, but happens entirely by accident."[‡‡]

This section will address some of the difficulties and approaches in reliable multicast protocol design. With unicast protocols, the sender and receiver can dynamically adjust to changing round trip delays, loading, etc. in a relatively straight forward manner, since the relationship is one to one. In contrast, for multicast and broadcast protocols, the relationship is one to many, or many to many (the most likely real world case), complicating the algorithms for adjustment of changing network and host loading conditions.

First I should define the semantics of reliability. Bormann[6] defines reliability from the perspective of sender and receiver. From the viewpoint of a receiver a protocol is reliable if the receiver can determine when "...it is failing or being partitioned from active senders."[6] From the viewpoint of the sender, a protocol is reliable if "...it is assured (with sufficient probability) that all messages reach within a bounded time all recipients that are not failing or being partitioned."[6] Stated simply, a protocol is reliable if it works, and if not, that it at least is cognizant that it doesn't.

So to design a reliable protocol, we must consider: Does the sender or receiver have the responsibility for detecting errors and initiating repairs? How does the protocol scale? Is ordering preserved? How is flow control addressed? Is it robust? The following sections expand upon these questions.

[††] Popular routing protocols are HELLO, BGP, EGP, OSPF, and RIP only to name a few.

[‡‡] James Burke, *Connections*

4.1. Error Recovery Methods

Generally it can be stated that protocols are either *sender-initiated* (e.g. the sender is responsible for error recovery) or *receiver-initiated.*

A distinction can be made with respect to how errors are recovered. If retransmissions must be done by the sender, the system is *sender-oriented.* If NACKs are multicast to the group, often a neighboring receiver is able to perform retransmission, therefore is said to be *receiver-oriented.*

Levine *et al.*[7] decouple the definitions of flow control, needed for the pacing of transmissions, from the mechanisms used for memory deallocation at the sender. Using this concept, multicast protocols can be thought of as managing two windows: a *congestion window*, which advances based on feedback regarding errors and network conditions; and a *memory window*, which advances based on feedback that allows the sender to known when it is safe to release buffers of transmitted data.

4.1.1. Sender Initiated Recovery

This method of recovery is familiar, since it is similar to the mechanisms used by TCP. Receivers send acknowledgments of each message to the sender, which maintains state information about the status of each receiver's acknowledgment. This approach does not scale well, because the processing load of maintaining state information is concentrated at the sender. As the number of receivers increases, the sender is overwhelmed with processing acknowledgments. This effect is known as *implosion.*

4.1.2. Receiver Initiated Recovery

Receiver initiated schemes distribute the processing load of error recovery to the receivers. Errors are detected by gaps in a what should be an ordered packet sequence, or by timeouts. Since it is the receiver's responsibility to guarantee delivery, the receiver generates a negative acknowledgment (NACK) to signal the need for retransmission. This also provides feedback for the sender to manage the congestion window, allowing the receivers to pace the source.

Levine gives proof that "...the ideal receiver initiated protocol requires an infinite amount of memory to work correctly."[7] In order to implement a receiver initiated protocol with finite memory resources, the memory window must also be managed. If no acknowledgments are used the sender can never be sure that all receivers have successfully received all packets. Therefore most receiver initiated protocols concentrate on providing a scalable, efficient, acknowledgment mechanism.

4.1.3. NACK Avoidance

Receiver initiated protocols are also vulnerable to the *NACK implosion* effect. This can occur when a packets are lost close to the sender, or when losses are widespread. The sender is overwhelmed with task of processing NACK requests. To address this, several protocols typically delay for a period,* when an error is detected, and listen for NACKs from neighboring receivers. If a NACK is heard corresponding to the lost packet, the receiver will suppress its duplication of the NACK. The intent is to minimize the number of duplicate NACK requests. These mechanisms are usually referred to as receiver-initiated with NACK avoidance (RINA). The basic RINA approach is the basis for several protocols.[8–12]

Another powerful feature of receiver initiated recovery scheme is that since the it is the receiver's responsibility for error recovery, each receiver can determine the level of reliability required, independent of other receivers, by simply not requesting retransmission of missed data. This is important to applications where data may become obsolete with time.

*Floyd *et al.*[8] discuss how the calculation of this period is dependent upon the underlying network topology, and how adaptive algorithms can improve performance.

4.2. Participant Structure

"...the Church had a bishop-to-bishop communications network connecting one kingdom to another, carrying news as well as ecclesiastical business ... "[†]

In order to implement efficient use of WAN links, and also to address the acknowledgment implosion effect, many protocols arrange receivers into logical groupings. The functions of groups often are twofold: receivers can act as local senders, aiding in error recovery, and acknowledgment hierarchies can be structured to reduce the number of acknowledgments seen by the sender.

4.2.1. Flat Structure

In a flat structure, group members are each equal with respect to the sender. In flat receiver-oriented systems, receivers that successfully receive a record of data are often able to aid in error recovery by responding to neighboring receiver NACKs, improving reliability. "The flat receiver-oriented approach consumes similar amounts of bandwidth as the sender-oriented (broadcast NACK) approach since NACKs and repairs are global in scope."[13]

4.2.2. Tree Structure

As the name implies, tree structured protocols arrange the members of a group into a hierarchy. The responsibility of servicing retransmission requests, is distributed from the source down the hierarchy, with the root of each sub-tree servicing its immediate children. Special hierarchy acknowledgments or HACKs[‡] are used, indicating to the parent that its congestion window may be advanced.

Note that these special acknowledgments do not indicate that data has been reliably delivered to its eventual destination. In the event of a node failure, end-to-end correct operation cannot be guaranteed,[7] i.e. a subtree root cannot advance the memory window without confirmation of receipt from all dependent leaf nodes below it.

Aggregate acknowledgments are sent starting at the leaves, and propagating upward, advancing only when all other nodes below it report their aggregate acknowledgment.

4.2.3. Token Ring Structure

The ring based protocols were originally developed to support applications requiring total ordering. The Reliable Multicast Protocol (RMP) is a good example of this. It used a post-ordering rotating token, in which a token is rotated among group members and messages are ordered by the token site after they have been sent.

4.3. Flow Control

"The sooner you fall behind, the more time you have to catch up."[§]

Most connection-oriented protocols use acknowledgments for both error-recovery and flow control. Since not all receivers are homogeneous with respect to processing capability and network connectivity, the sender must throttle the rate at which data is transmitted when congestion is detected. The two basic approaches used for flow control are rate-based and window-based. In rate-based techniques, the sender limits the rate at which data is transmitted. The window-based technique, attempts to manage the congestion window, based on feedback, from its receivers.

In WAN links packets may often be dropped due to link quality, as they are dropped due to congestion. An increase in the number of NACKs generated may indicate either condition. Simply reducing transmission rate in the case of poor link quality may be inappropriate. Other factors such as round-trip timing and ICMP messages from routers should also be considered. Since the sender must accommodate the characteristics of many receivers, trade-offs must be made in each design. For example, should the sender slave its flow rate to the slowest receiver? or should some other statistical metric be used? The form in which these are addressed directly affects performance.

[†] James Burke, *Connections*

[‡] I love that name.

[§] Author unknown

4.4. Scalability

"Never underestimate the bandwidth of a station wagon full of tapes hurtling down the highway."[¶]

Evaluation of the scalability of a protocol is complex issue and continues to be a subject of much research, however, it can be stated generally that the approach to error recovery, flow control, and ordering semantics tend to have the most significant affects on scalability. Structure also plays an important part since protocol overhead can be reduced by localization of NACKs and repair transmissions.

4.5. Packet Ordering, Fragmentation, and Reassembly

"Always try to do things in chronological order; it's less confusing that way."[‖]

Although not a direct requirement for reliability, some protocols offer additional ordering semantics with respect to the application layer. This is important for data that spans multiple packets, so that the data retains its original form. A protocol may offer none, total-global, single-source, or causal ordering. Total global ordering, means packets from different senders will be delivered to each receiver's application in the same relative order. This type or ordering would be required for example in a distributed software development environment, where the application of code-patches must be performed in the same order by all participants. Single-source ordering means packets are delivered to a receiver's application in the same order as sent from the source. Causal ordering means that related packets, are delivered to a receiver's application in the same relative order they were sent.

It is important to note that ordering may increase the packet delivery latency, since a packet that has already been received, must be buffered, while a prior missing packet is awaited.

5. TAXONOMY OF MULTICAST PROTOCOLS

"In protocol design, perfection has been reached not when there is nothing left to add, but when there is nothing left to take away."[**]

This section is not intended to be a complete taxonomy, but rather a quick introduction to some of the current work in the field. I encourage interested readers to refer to more extensive studies.[7,13–15]

5.1. Adaptive File Distribution Protocol (ADFP)

ADFP uses a receiver-initiated error recovery scheme, with single-source ordering. ADFP was designed specifically to address file distribution, and uses a publishing metaphor. The publishing metaphor uses a publisher, subscriber, and secretary for sender, receiver, and master respectively. ADFP is able to dynamically select between multicast, unicast and broadcast mechanisms for data transfer, based on the number of receivers. For small groups unicasting may be used. This is also useful for systems that do not support multicasting. An implementation of ADFP in available from the authors.[16]

5.2. Local Group based Multicast Protocol (LGMP)

The LGMP protocol organizes receivers into subgroups called *local groups.* Each local group has a group controller which coordinates retransmission requests and handles status reports. Error recovery is first attempted within the group by using a receiver-initiated approach, with repairs requested from the sender only if no other receivers in the local group have the data. The group controller also performs local acknowledgment processing, and informs the sender of the summary status of the group.

Group management is not specifically part of the LGMP protocol, but a cooperating protocol called Dynamic Configuration Protocol (DCP) is used. LGMP provides single-source ordering semantics. Flow control is a rate-based algorithm with multiplicative decrease and additive increase, each local group handling congestion control separately. Simulations of more than 2000 receivers have been conducted.[10]

An implementation of LGMP is available from the LGMP web page[17] for Solaris, Digital Unix, and Linux. Work is also underway to integrate LGMP with the Reliable Multicast Framework (RMF) application interface.

[¶] Andrew Tannenbaum

[‖] Author unknown

[**] Variation of quote by the French aviator, adventurer, and author Antoine de Saint-Exup'ery

5.3. Multicast File Transfer Protocol (MFTP)

The Starburst Multicast File Transfer Protocol[18] (MFTP), designed specifically for file distribution, partitions a file into a set of blocks. At first all packets in a block are sent. Then the receivers send back a NACK with a bitmap each bit corresponding to a possibly lost packet. The sender collects the NACK messages and determines which packets have been requested at least once. The process repeats until all packets have been delivered.

5.4. Multicast Transfer Protocol, Version 2 (MTP-2)

MTP-2 is an improvement upon the MTP protocol, defined in RFC-1301.[19] It defines a three role model: sender, receiver, and master. The group master coordinates activity through the use of tokens. In the event the master fails, the remaining members elect a new master. This master recovery function addresses what otherwise would be a single point of failure.

MTP-2 uses receiver initiated NACKs for error recovery. Errors are detected by receivers when a sequence number gap is detected, or when no packets have been received for a period of time while the message is still incomplete. NACKs are unicast to the sender, but repairs are multicast to the entire group. Senders are expected to retain transmitted information for a period of time specified by the master, after which the sender's memory window is advanced. This interval is known as a retention window. Although most retransmissions can be handled within the retention window, there is a finite probability that all packets associated with a message may be lost. "...it is similar in effect to the maximum retransmission count found in many reliable data link and transport protocols."[6]

Rate control is accomplished by the master through the use of transmit tokens. A sender wanting to send data must request a transmit token from the master. The master dynamically adjusts a rate window based on the number of tokens granted.

Senders are allocated a portion of bandwidth available for the group. The master dynamically adjusts this allocation based on the number of tokens granted. Senders are not allowed to exceed the specified rate. In this way the master can ensure the maximum rate is not exceeded.

MTP-2 provides total global ordering, with the capability to manage several message streams, each with their own independent ordering. Applications can therefore avoid unnecessary delays by using the assignment of independent messages to different streams.

One of the strengths of MTP-2 is its ability to recover after the loss of a master. For larger WAN-based applications, MTP-2 probably will not scale well. "Although unicast NACKs are less expensive than multicast NACKs, retransmission requests for a missing packet may be made by more than one receiver, leading to duplicate multicasts of the same [lost] packet."[13] In network environments where the packet round trip delay exceeds the product of the heartbeat rate and retention window, reliability is compromised. This can be compensated by dynamic adjustment of these parameters, although this vulnerability may also be addressed with quality of service resource allocation mechanisms such as RSVP.[20] Another design limitation is the size of the message acceptance field, that limits the number of concurrent outstanding messages to 12. This is obviously a design trade-off, increasing the size of this field would increase the protocol overhead, since this field is present in all MTP-2 packets.

Xmc and Xy (X window sharing tools) have been developed with MTP-2 as a basis for reliable multicast communication. The software is available for SunOS, Solaris, and Linux operating systems, and may be obtained from the authors ftp site.[21]

5.5. Reliable Adaptive Multicast Protocol (RAMP)

The Reliable Adaptive Multicast Protocol (RAMP), developed by TASC, was originally defined in RFC-1458.[22] Two delivery modes are defined by RAMP, burst mode and idle mode. Burst mode is intended to address small highly interactive multicast groups, as is the case in most collaborative applications. Idle mode is intended to address larger groups. RAMP can operate in both reliable and unreliable modes, selectable by each receiver. The designers clear focus on collaborative environments make RAMP especially applicable for remote observing applications.

Receiver-initiated error recovery is used in both modes of operation. NACKs are sent if a gap in packet sequence numbers are seen. In the burst mode, the receivers periodically send ACKs to the sender, in response to packets with the acknowledge flag set. In the idle mode, receivers will issue a retransmit request if no idle messages or data packets are seen for an interval of time. Idle packets are multicast to the entire group. Repair packets may be either unicast or multicast to the group after a delay period. RAMP provides single-source ordering semantics. Flow control is based upon the number of NACKs or router ICMP quench messages during an interval of time.

The dual modes of RAMP address the two most common reliable multicast scenarios. Burst mode addresses highly interactive collaborative applications that usually need low-latency high-throughput connections, with a relatively small group size. On the other end of the scale large groups emphasize the need for scalability, at the sacrifice of latency and throughput.

RAMP is implemented in C++, and is available on IRIX, SunOS, and Solaris operating systems.[23] RAMP is also integrated with Reliable Multicast Framework (RMF), designed to provide a Application Program Interface (API) that can be used with a variety of multicast protocols.

5.6. Reliable Multicast Protocol (RMP)

The Reliable Multicast Protocol (RMP) provides a totally ordered, reliable delivery, using a combination of receiver-initiated error recovery and a rotating token. Acknowledgments are multicast to members of the group by the token holder when a packet is received, indicating a traditional positive acknowledgment to the sender, and passing the token to the next member in the group. When an error is detected, a NACK is multicast to the group after a statistical delay. If a neighboring receiver containing the missing data fails to see repairs from the sender (or other receiver) after an interval, it will post repairs to the group.

The default flow control policy for RMP is a modified sliding window protocol, based on the algorithms used for TCP. RMP's flow and congestion control policies are meant to be orthogonal to the rest of the protocol, allowing other flow control policies to be developed.

"RMP may best work in low loss environments such as LANs and controlled inter-networks, and applications which require high qualities of service, such as those that replicate data and files ..."[13]

RMP is available for SunOS, IRIX, Ultrix, OSF, Linux, and Win32 from GlobalCast Communications.

5.7. Reliable Multicast Transport Protocol (RMTP)

RMTP, developed by Bell Labs, uses a multi-level hierarchical structure in which leaf receivers periodically send status messages to designated receivers (DR), which act as local group controllers. The DRs in turn propagate status messages up the hierarchy. Error recovery is receiver-initiated, with local recovery by the handled by the DR. "...for WAN scenarios, the number of retransmissions showed how important DRs were in caching received data, processing ACKs, and handling retransmissions"[13]

RMTP uses a combination of rate control and window-based control. The rate is adjusted based on the receiver status. A TCP style slow start mechanism with multiplicative decrease, and additive increase is used when congestion is sensed. "The progress of the slowest receiver determines the window size."[13] Single-source ordering semantics are supported.

5.8. Single Connection Emulation (SCE)

Single Connection Emulation (SCE) is a modification to the TCP protocol, being developed at the University of Georgia. It is a sender-initiated scheme, extending TCP concepts for multicasting. Flow control, error correction, and retransmission are done by TCP.

5.9. Scalable Reliable Multicast (SRM)

Scalable Reliable Multicast (SRM) is a receiver-initiated protocol. The memory window is managed by periodic sessions messages, that report the highest sequence number seen. In addition, session messages provide receivers information about group membership, and one-way distance estimation, needed by the repair algorithm. To reduce the number of duplicate NACKs and repairs, SRM uses an adaptive algorithm, "...that is effective in controlling the number of duplicates over a range of scenarios."[8]

Rate control is accomplished by defining a peak rate for senders. A concept known as Application Level Framing (ALF) is used by applications to provide ordering semantics, hence ordering is not directly supported by the protocol itself.

SRM concepts have been tested on a global scale with the *wb* distributed white board application. Tests have been conducted "...with sessions ranging from a few to more than 1000 participants."[8]

5.10. Tree-based Multicast Transport Protocol (TMTP)

Tree-based Multicast Transport Protocol (TMTP) developed by the University of Kentucky, uses a hierarchical scheme to localize error recovery and acknowledgment processing. Groups known as domains are used with both sender-initiated and receiver-initiated error recovery mechanisms. NACKs are multicast with limited scope (i.e. the packet time-to-live field is set to localize packet traffic) when receivers detect a lost packet. Domain managers use a sender-initiated scheme, with positive acknowledgments and timeouts signaling the need for message retransmission.

Flow control is accomplished in TMTP by a combination of both rate control and window-based techniques. The window-based mechanism delays retransmissions as long as possible, to increase the likelihood of ACK reception. "TMTP is a reliable one-to-many multicast protocol that keeps the progress of reception between the various receivers tightly together, i.e. the slowest receiver sets the tone."[15]

5.11. Xpress Transport Protocol (XTP)

XTP is probably one of the more mature protocols, initially developed in 1987. XTP is a combined network and transport layer protocol that supports multicast. Error detection is receiver-initiated, but recovery is sender-oriented. The XTP protocol supports both unreliable and reliable packet delivery. XTP uses the go-back-N strategy, retransmitting missed packets to the entire group.

Table 1. Summary of Reliable Multicast Protocols.

Protocol	Error Recovery	Structure	Ordering	Flow Control
ADFP	Receiver-initiated	Flat	Single-source	Rate-based
LGMP	Receiver-initiated	Hierarchical	Single-source	Rate-based
MFTP	Receiver-initiated	Flat	Single-source	Rate-based
MTP-2	Receiver-initiated	Flat	Global	Rate-based
RAMP	Receiver-initiated	Flat	Single-source	Rate-based
RMP	Receiver-initiated	Token-Ring	global	Window-based
RMTP (Bell Labs)	Receiver-initiated	Hierarchical	Single-source	Window-based
SCE	Sender-initiated	Flat	Single-source	Window-based
SRM	Receiver-initiated	Flat	ALF/none	Rate-based
TMTP	Sender & Receiver-initiated	Hierarchical	Single-source	Window-based
XTP	Receiver-initiated	Flat	Single-source	Window-based

6. CONCLUSION

"There's an old story about the person who wished his computer were as easy to use as his telephone. That wish has come true, since I no longer know how to use my telephone."[†]

Clearly much has been left out of this paper. However, I have presented the major issues challenging multicast protocol designers today. It should be clear that there are different classes of multicast applications: those that emphasize scalability as the primary requirement; those that emphasize ordering; and those that must meet stringent latency requirements. Each set of requirements is best met by differing approaches. No doubt future work will be concentrated in algorithms which are adaptive to varying application requirements, network topology, loading, as well as network management and multicast routing.

To date, I have conducted minor tests with implementations of LGMP, MTP-2, RAMP/RMF, and RMTP protocols. The GBT monitor and control core communications library (RPC++) has been interfaced with the Solaris version of MTP-2. Although only minor tests have been run, the technology looks promising and may be applicable to the GBT in the near future.

[†] B. Stroustrup

Multicasting technology will enable many types of applications to efficiently transfer information, reducing network congestion and server loads. The design of the GBT local area network has been done with the need for efficient processing of both unicast and multicast data. Although no one protocol has been chosen, considerations for general support of reliable IP/multicast protocols have been incorporated in the GBT network design.

A recent article reads: "Advocates [of multicast] are overlooking one detail: with the coming of Gigabit Ethernet, network managers are trying to find ways to use all the bandwidth, not save it."[24] While possibly true today, just wait for *tomorrow*.

REFERENCES

1. A. Benett, J. Brandt, M. Clark, and J. Ford, "Monitor/Control Hardware Design Review," Tech. Rep. GBT Archive MC022, National Radio Astronomy Observatory, 1996.
2. S. Deering, "Host Extensions for IP Multicasting," *Network Working Group* **RFC-1112**, 1989.
3. J. Moy, "Multicast Extensions to OSPF," *Network Working Group* **RFC-1584**, 1994.
4. A. Ballardie, "Core Based Trees (CBT) Multicast Routing Architecture," *Network Working Group* **RFC-2201**, 1997.
5. C. P. D. Waitzman and S. Deering, "Distance Vector Multicast Routing Protocol," *Network Working Group* **RFC-1075**, 1988.
6. T. K. C. Bormann J. Ott, H. Gehrcke and N. Seifert, "MTP-2: Towards Achieving the S.E.R.O. Properties for Multicast Transport," *Proceedings of ICCCN* , 1994.
7. B. Levine and J. Garcia-Luna-Aceves, "A Comparison of Multicast Networking Protocols," *Multimedia Systems* , 1998.
8. V. J. S. Floyd and S. McCanne, "A Reliable Multicast Framework for Lightweight Sessions and Application Level Framing," *ACM SIGCOMM* , 1995.
9. J. Cooperstock and S. Kotsopoulos, "Why Use a Fishing Line When You Have a Net?," *Usenix Proceedings* , 1996.
10. M. Hofmann, "A Generic Concept for Large-Scale Multicast," *International Zurich Seminar on Digital Communication* , 1996.
11. S. K. B. Whetten and T. Montgomery, "A High Performance Totally Ordered Multicast Protocol," *INFOCOMM* , 1995.
12. K. S. S. Paul and J. Lin, "Reliable Multicast Transport Protocol," *IEEE INFOCOMM Proceedings* , 1996.
13. M. D. Petitt, "Solutions for Reliable Multicasting," *Thesis, Naval Postgraduate School* , 1996.
14. D. T. S. Pingali and J. Kurose, "A Comparison of SenderInitiated and ReceiverInitiated Reliable Multicast Protocols," *SIGMETRICS* , 1994.
15. C. Hänle, "A Comparison of Architecture and Performance between Reliable Multicast Protocols over the MBone," *Thesis, Institute of Telematics, University of Karlsruhe* , 1997.
16. J. Cooperstock and S. Kotsopoulos, *ADFP Protocol Implementation*, ftp://ftp.ecf.toronto.edu/pub/adfp.
17. M. Hofmann, *Local Group Concept, Univeristy Web site*, http://www.telematik.informatik.uni-karlsruhe.de/hofmann/lgc-publications.html, 1997.
18. A. T. K. Miller, K. Robertson and M. White, "Starburst Multicast File Transfer Protocol (MFTP) Specification," *Network Working Group* **draft-miller-mftp-spec-02**, 1997.
19. A. F. S. Armstrong and K. Marzullo, "Multicast Transport Protocol," *Network Working Group* **RFC-1301**, 1992.
20. R. Braden and L. Zhang, "Resource Reservation Protocol," *Network Working Group* **RFC-2205**, 1997.
21. J. O. C. Bormann and N. Seifert, *MTP-2 Protocol Implementation*, ftp://ftp.cs.tu-berlin.de.
22. R. Braudes and S. Zabele, "Requirements for Multicast Protocols," *Network Working Group* **RFC-1458**, 1993.
23. A. Koifman and S. Zabele, *RAMP/RMF Website*, http://www.tascnets.com/mist/RMF/release/index.html, 1997.
24. S. Steinberg, "Deflating this Month's Overblown Memes," *Wired magazine* **Nov. '97**, 1997.