A Classification of Reliable Multicast Protocols

J. William Atwood, Concordia University

Abstract

The range of user requirements on multicast protocols is so wide that no single protocol will ever satisfy them. The set of multicast protocols can be classified using the user requirements, and the architectures, mechanisms, communications patterns, and policies used to satisfy these requirements. We provide such a classification, and illustrate it with several example protocols chosen to cover the range of features described.

For point-to-point (unicast) communication, the Transmission Control Protocol (TCP) [1] has dominated the Internet data communications landscape for many years.

TCP provides absolutely reliable service unless the underlying network fails. It assumes the user requires that all data be delivered, in sequence.

To provide multicast services (point-to-multipoint and multipoint-to-multipoint), a variety of protocols is available. However, when considering reliable multicast equivalents to TCP, the situation is vastly different from the unicast situation, as the number of possible failure modes is larger, and the definition of "reliable" can take on many shades of meaning. Each protocol proposed in the literature is intended to meet the needs of a particular set of applications. Each has a slightly different definition of reliability, and each operates in a slightly or significantly different environment. Given the wide variety of requirements and environments, there will never be a "one size fits all" multicast protocol design [2]. However, this does not mean it is impossible to develop a family of protocols (or a single protocol with a variety of selectable features) that satisfies a wide range of requirements.

As a step toward identifying the features of such a family, we have developed a taxonomy of reliable multicast protocols. Previous taxonomies include Diot *et al.* [3], which is a survey of multicast protocols in terms of functionality and mechanisms for reliable multicast transmission, and Obraczka [4], which reviews several existing multicast transport mechanisms and classifies them according to their distinct features. The Reliable Multicast Transport Working Group (RMTWG) within the Internet Engineering Task force (IETF) has a mandate to standardize reliable multicast transport. It has developed a framework for standardization of reliable multicast transport for the specific subset of reliable multicast protocols that relate to bulk data transfer (point-to-multipoint), and is now populating that framework. The reader is referred to the RMTWG Website [5] for current information on the progress and status of the RFCs and Internet drafts.

To help identify the components that are common across all classes of reliable multicast protocols (point-to-multipont and multipoint-to-multipoint), we have developed a taxonomy based on requirements, architectures, mechanisms, communications patterns, and policies. This brings out the orthogonality between the various requirements of reliable multicast transport protocols and the ways they are used to satisfy those requirements.

Requirements

Traditional protocol design has started with a statement of the requirements, and followed with the exploration of possible designs that can meet these requirements. In this section we identify the range of requirements that characterize and differentiate reliable multicast protocols.

To make the issues more concrete, capsule summaries of the following multicast protocols are given later in the article: UDP, XTP (two variants), RMP, PGM, SRM, LGMP, RMTP (two variants), and LPC. (LPC is a representative hybrid forward error eorrecting protocol, which is included in the study to illustrate an upper bound on receiver scalability.) For each requirement stated below, we note the example protocols that have this requirement. Table 1 gives the correlation between user requirements and the example protocols.

Multicast applications vary in their requirements along a number of dimensions:

- Number of senders
- Group organization and receiver scalability
- Data reliability
- Congestion control
- Group management
- Ordering

These parameters all interact with each other: for example, the data reliability requirements that can be met are dependent on the receiver scalability required.

Number of Senders

The first dimension of our classification is based on the number of senders: single-sender (point-to-multipoint) and multiple-sender (multipoint-to-multipoint) applications.

Point-to-multipoint or 1-to-*N* multicast applications require data delivery from a single source to multiple receivers, and usually run without human interaction. A few example applications for this set include software distribution, data distribution and replication, and mailing list delivery.

Multipoint-to-multipoint or *M*-to-*N* applications require data delivery from multiple sources to multiple receivers.

0890-8044/04/$20.00 © 2004 IEEE

Protocol	# Senders	Scalability	Data reliability					Congestion control	Group management	Ordering
			Weak				Strong			
			BE	BL	MR	RC	AB			
UDP	M	Large	x					No	None	no
XTP 3.6	1	Medium	x			x		No	Weak	no
XTP 4.0	1	Small	x				x	No	Strong	no
RMP	M	Medium					x	Yes	Strong	yes
SRM	M	Large				x		No	Weak	appl
PGM	M	Medium				x		No	None	no
LGMP	1	Large				x		Yes	Weak	no
RMTP	1	Large				x		Yes	None	no
RMTP-II	M	Large		x		x		Yes	Weak*	appl
LPC	1	Enormous				x		No	None	no

Data reliability key:
BE = best effort
BL = bounded latency
MR = most recent
RC = receiver-centered
AB = Absolute

Ordering key:
Appl = application

*But accurate reporting

Table 1. *Correlation between reliable multicast requirements and example protocols.*

They often have direct human involvement and stringent latency requirements due to their interactive nature. They may also have a requirement to order the delivery of the data from the multiple senders. In most cases, an *M*-to-*N* protocol operates in *M*-to-*M* mode (all senders are also receivers). Example applications include interactive distributed simulations, networked gaming, and other collaborative applications such as teleconferencing and videoconferencing.

Group Organization and Receiver Scalability

Multicast protocols have very different characteristics depending on the number of senders and the number of receivers. We will use the categories *one*, *few*, and *many* to describe these two dimensions. Few is less than 20. Many can be several hundred or several thousand.

One-to-one is unicast transmission. Most collaboration applications fall in the few-to-few category. Given the nature of human interaction, there is a practical limit on the number of users who can simultaneously carry on a useful session. Hence, it is unlikely that such an application would ever require the protocol to be scalable to even 20 users.

One-to-few and one-to-many represent data distribution applications. One-to-few applications are relatively simple to design, but one-to-many applications may require communication support for groups of hundreds or thousands of members, which may undergo a very high number of group changes per second [6]. Few-to-many could be a collaboration application with a large audience, or a data distribution application with a small number of senders and a large number of receivers. True many-to-many is unlikely, given the constraints imposed by social interaction noted above, but not impossible. As an example, during the NASA Shuttle broadcast, the actual data (audio and video signals) constituted a one-to-many group, but the overall operation was a many-to-many group, with over 200 participants [6], because each listener announces its presence to the entire group by multicasting to the group address.

We differentiate four levels of scalability: small groups (with few members); medium groups, where the group is on a single LAN or within a single administration, and the cost of multicasting from a receiver is small; large groups, where there is geographical distribution, or the cost of multicasting from a receiver is high; and enormous groups, where the losses on separate branches of the distribution tree are uncorrelated, or there is *no* reverse path to provide feedback for error control.

XTP is a 1-to-*N* protocol whose architecture limits it to serving small to medium groups. LGMP and RMTP are 1-to-*N* protocols designed to serve large groups. Recently, RMTP-II has been introduced; it supports multiple senders into a group, but remains fundamentally a 1-to-*N* protocol. LPC is a 1-to-*N* protocol designed to scale to enormous groups.

RMP, SRM, and PGM are *M*-to-*N* protocols designed to serve medium to large groups.

Data Reliability

There are many "reliable multicast" protocols, each of which has its own view of what data reliability is (or how important certain data reliability features are). The mechanisms used in a protocol will be determined to a great extent by the actual data reliability requirements. For unicast transmission, the notion of data reliability is simple: there is only one recipient, and that recipient either does or does not receive all the data sent by the sender. For multicast transmission, the situation is much more fluid, given that there are many more ways in which the communication can fail.

For small groups with a single sender, a very strong meaning of reliability can be assigned, similar to the meaning assigned for unicast transmission using TCP: communication is reliable if all data provided by the sender are received, in order and without duplication, by the receivers. For multiple senders, this definition must be applied to each individual sender. The test of reliable reception can be made by the sender or receiver. The term *strong* reliability will be used for this case.

For large groups, a relaxed definition of data reliability must be accepted, because the test can be applied only by the receivers, to avoid the possibility of overwhelming the sender(s) with status-response packets. This also implies weak or nonexistent knowledge at the sender(s) about the membership of the group and the progress of data reception at these members. Handley *et al.* [7] call this *semi-reliable* delivery, but we will use the term *weak* reliability.

Given these disparate views of the term reliable, multicast applications can be classified based on how reliably the transport connection delivers data to the layer above it.

Best Effort Reliability — Although *best effort* is a term traditionally applied to the network layer, it can also be used to describe the data reliability provided by a transport protocol that makes no effort to improve the reliability provided by the network layer. This can be useful for applications where loss can be tolerated but end-to-end semantics are required. It is best exemplified by the UDP service model.

Bounded Latency Reliability — In this case, each packet adheres to a specified lifetime over which the data are useful to the receiver, thereby defining an upper bound on its delivery latency. Packets arriving outside this timeframe are discarded. A common application that requires such a reliability guarantee is a video stream, where each packet has a playback time, and any packet not meeting this deadline is of no further use.

Most Recent Reliability — This type of reliable transmission targets applications where only the most recent data for a particular parameter is of interest. A simple example would be a service that provides reliable updates of stock quotes. If a particular stock quote is lost and a new update is received before retransmission can occur, the old data are rendered useless. Thus, it is possible that the data may take on a value that is never known to some or all of the receivers. Other example applications include digital interactive simulations (DIS) and situational awareness dissemination in shared WAN environments.

Receiver-Centered Reliability — In some applications (especially with large numbers of receivers), reliability is determined at the receiver: the sender has no responsibility for error recovery (other than possibly providing retransmission of packets), and often no knowledge of the success of the delivery. However, this service is superior to best effort reliability, since an effort (possibly quite significant) is made to recover lost data.

Absolute Reliability — This type of reliability requires that all of the transmitted multicast packets be delivered to the active group. If any of the data are missing at the receiver, even after repeated retransmissions, none of the data will be useful. This is analogous to the reliability supported by TCP for unicast sessions. Multicast file transfer is an example of an application requiring absolute reliability.

Handley *et al.* [7] distinguish reliability of application data units (ADUs) from reliability of individual packets. (This is a variation of the end-to-end reliability argument of Saltzer *et al.* [8].) If the ADUs span several packets, it is possible to obtain absolute reliability for ADUs while lowering the feedback requirements for individual packets by using receiver-centered reliability.

Among the example protocols, UDP offers only best effort service. XTP 3.0 offers the *choice* of best effort or receiver-centered reliability, while XTP 4.0 offers a choice between best effort and absolute reliability. RMP offers absolute reliability, even when receivers are partitioned from the group. SRM, PGM, LGMP, RMTP, RMTP-II, and LPC offer receiver-centered service, with RMTP-II offering bounded latency.

Congestion Control

Congestion control is one of the most difficult requirements to satisfy in multicast due to its multipoint nature. Uncontrolled propagation of multicast data by even a few sources can potentially cause havoc on large-scale internetworks. Thus, for reliable multicast to be accepted and embraced, it must address congestion control requirements in an efficient manner. Multicast protocols must also coexist with the more common unicast protocols such as TCP, because one strong requirement imposed by the IETF [2] on any (new) protocol is that it be "TCP-friendly," in the sense that it obey similar rules concerning congestion control to those mandated for TCP.

The need for multicast congestion control can be expressed in two forms. The first is to require maintenance of the same speed for all receivers, in the presence of congestion. The second is to permit certain receivers to lower their rate requirements on experiencing congestion. An excellent review of multicast congestion control for the first requirement may be found in the papers by Golestani and Sabnani [9], and Basu and Golestani [10]. A novel congestion control algorithm for layered multicast, which responds to the second requirement, is described by Vicisano, Crowcroft, and Rizzo [11].

Congestion control uses evidence of packet loss as its signal to reduce traffic flow. Since packet loss is also indicative of the need to repair, congestion control will often be integrated with error recovery, using the same mechanisms.

Congestion control is an area of active research in the IETF [12], and will not be discussed further in this article.

Group Management

The notion of strongly reliable delivery of data is meaningless if the membership of the destination group is not defined. However, not all applications need an absolute definition of the group membership. It is generally true that as the group gets larger, interest in maintaining accurate knowledge of the group membership decreases. For this reason, we classify multicast protocols according to the degree of member knowledge required.

One way to express the requirement for group management is to define the set of conditions that must be true for a transport connection to enter or remain in the data transfer phase. The term *active group integrity* (AGI) identifies the necessary conditions in accordance with various levels of group reliability.

Two extremes for these conditions are 0-reliable multicast, where the sender is not required to have any knowledge of its set of receivers, and *K*-reliable multicast. A group of N receivers is said to be K-reliable, where $(0 \leq K \leq N)$, if at least K members of the group are alive (i.e., responding) at any particular instant. If a receiver failure is detected, that particular receiver is removed from the group, and the remaining group members can continue as long as K members remain.

Multicast group management deals with managing the group in accordance with the AGI. The level of reliability provided by a multicast protocol depends on the control algorithms and group policies used to manage the group.

In keeping with our use of data reliability, we use the terms *weak* and *strong* to differentiate protocols that make some effort to manage the group membership from those that ensure complete knowledge, supplemented with the use of *none* for those that provide no support at all.

For some applications, the group will form at the beginning of the session and continue without change during its lifetime. This is the case of $K = N$. For others, there is a requirement for ongoing group membership modification. In both cases, receivers must be able to join, and in the second case able to join after the group has started to operate. Similarly, members

should also be allowed to leave multicast groups at any time. (This may or may not destroy the group, depending on the particular conditions for maintaining the AGI.)

If members are permitted to join after the session has been established, policies must also exist to determine valid join points — whether the joining member is allowed to "catch up," or must join at some sort of "synchronization point".

A variation on this idea is a requirement to re-form a group after it has become partitioned.

XTP-4 and RMP provide strong control over group membership. The other protocols (in the set of examples) adopt weak or nonexistent group control to avoid overwhelming the sender(s).

Ordering

Few-to-few applications often carry with them a requirement for global ordering. This applies only to applications with multiple senders, since a single-sender application only has one possible ordering. A protocol serving this class of applications will typically provide at least mechanisms for ensuring complete ordering among the packets issued by the session members.

Packets may arrive out of sequence at their destination due to packet losses or changes in the datagram routes. Many distributed applications require ordered reception of packets, because misordering may give a different view of the state of the group. Ordered reliable multicast communication among a set of users is defined based on how the objects of a sending user are presented to the receiving user and how the receiving user gets objects from the sender.

In the case of single-sender ordering, the objects generated by the sending user must be delivered to each receiving user in the active group in the same order in which they were sent. In the case of multiple sending users, ordering is determined in terms of the relative sequencing of objects received from multiple sending users. The ordering relationship defines the arrangement or interleaving of objects from multiple senders, and can be classified as no ordering, local ordering, causal ordering, partial ordering, or total ordering [13].

A protocol that ensures, in addition to reliability, a total ordering of the delivered messages is called an *atomic* protocol. Such a protocol may be used for reliable validation, atomic operations, group memberships, and so on, while a causal protocol (ensuring causal and not total ordering) may be used for ensuring consistency in updates to replicated data.

UDP provides no ordering guarantees. (In fact, it provides no guarantee that the packets will be delivered at all.)

All *reliable* multicast protocols must, of necessity, provide local ordering; guaranteeing that a sequence of packets will be delivered requires that the packets be sequenced (ordered). Partial, causal, and total ordering are meaningful only when multiple senders are allowed, and reasonable only when there are few senders. RMP provides strong control of ordering, as it is intended to serve collaborative applications and a (relatively) small number of users. SRM and RMTP-II leave ordering to the application (i.e., they consider ordering a task to be done at a superior level). PGM considers multiple sources to be independent. For the single-sender protocols (XTP, LGMP, and RMTP), ordering (beyond local ordering) is not relevant.

Architectures for Multicast Protocols

The multicast requirements outlined in the previous section are met using a variety of mechanisms and supporting architectures. This section describes the architectures that have been used; the next section describes the mechanisms and policies.

Data Distribution Architectures

Reliable multicast protocols use the basic delivery service provided by IP multicast [14] (and, in many cases, the IP unicast delivery service as well) to distribute the data, control traffic, and error recovery traffic required to ensure the necessary level of reliability. The properties of the IP multicast delivery service have a significant effect on the design (and hence classification) of the protocols.

Multicast routing protocols are responsible for construction of multicast packet delivery trees and forwarding the multicast packets within a specific domain. An optimal delivery tree is constructed for a group using the specific multicast routing algorithm employed in that domain. Two basic forms of these algorithms are used: *source-specific*, where a distribution tree is constructed from each source to the set of receivers, and *shared tree*, where a single tree is built and then shared by all senders. A detailed study of the various multicast routing algorithms and protocols can be found in Almeroth [15].

In addition to the distribution properties just discussed, being a multicast sender may result in significantly higher charges from a service provider, which has a strong influence on the communications patterns employed.

Error Recovery Architectures

To make a protocol reliable, the missing data must be discovered, and then a decision must be made to retransmit these data. Given that only the sender knows what was sent, and only the receiver knows what was received, status packets must be exchanged between the two to permit a retransmission decision to be made. Feldmeier and Biersack [16] define the *loss estimation system* (LES) as the system that decides whether data have been lost. If the LES is at the sender, the status reporting packets must go from the receiver to the sender. Protocols that adopt this strategy are called *sender-reliable*. If the LES is at the receiver, information about the transmitted packets must flow from the sender to the receiver. Protocols that adopt this strategy are called *receiver-reliable* [17].

If the multicast group is small, it is possible for the LES to be located at either the sender or the receiver. Locating it at the sender has the advantage that the sender retains complete knowledge about the progress of all members of the group. If there are multiple senders, either each sender must maintain information about its own transmissions, or responsibility must be delegated to a single *coordinator*, responsible for overall control of error recovery for the group. This latter case is more likely if there is an additional requirement for ordered delivery. The error recovery architecture in this case is *flat*; the sender(s) and receivers communicate directly, with no intervening hosts.

As the size of the multicast group grows, the flow of information from the receivers to the sender can overwhelm the sender. For this reason, all such larger groups use an LES located at the receiver. If the group is of medium size, it is possible for the receivers to only indicate loss when it occurs using negative acknowledgments (NACKs), and retain the flat structure. Saturation of the sender can be reduced by suppressing redundant error reports.

For large groups, error recovery cannot be managed by a single entity, because that entity will be swamped by error management packets. It is necessary to create an error recovery hierarchy that parallels the data distribution tree. Tree-based reliable protocols have been shown to be the most scalable [18]. Various researchers [19–21] have noted that it is important to match the error recovery tree to the data distri-

bution tree as closely as possible. Receivers request retransmission from some host that is *near* to them (a *parent* or *sibling*). This reduces the delay for error recovery, and minimizes the load on the original sender. If there are multiple senders, it is extremely likely that error recovery will be delegated to a single host managing a single tree, because of the high cost of duplicating the recovery tree. Protocols that use hierarchy are called *hierarchical receiver reliable*.

For enormous groups, and especially when the losses in different parts of the distribution tree are uncorrelated, hierarchy may still be employed to report the losses, but the error recovery unit is likely to be multicast over the whole data distribution tree. Alternatively, there may be no error recovery architecture, because the original data are sent redundantly or repeatedly.

We will call the object at the root of the error recovery structure the *global controller*.

We will call an object in the interior of the error recovery structure a *local controller*. As noted, these objects are likely to be located close to a branching point in the data distribution tree, although this is not required. If they are collocated with network-level routers, there may exist a symbiotic relationship between the routers and the local controller, which can improve performance.

Group Management Architectures

Group management is very similar to error recovery, in that small groups can be managed absolutely with a simple one-level structure, and large groups must be managed in a hierarchical fashion. In keeping with our model for error recovery, we will reuse the terms global controller and local controller, and rely on context to indicate whether it is data reliability or group management being controlled.

As the groups get larger, the sender will of necessity retain less and less information about the group.

Assistance from Network Elements

A reliable multicast protocol must involve mechanisms running in end hosts, and routers forwarding multicast packets. In standard IP multicast, the routers merely forward the data packets, leaving the reliability mechanism to the transmitters and receivers. If the group is large enough that hierarchy is necessary, the local controller will exist at (or close to) the interior branching points of the data distribution tree. These local controllers can be application-level objects, or directly implemented in the routers (which *ensures* that they are collocated with the branching points [19]).

One approach to router assistance is to use the router as a *turning point* [20], implementing *forwarding services* to redirect error recovery packets to a local controller, called a *replier*. This requires minimal overhead at the routers and no maintenance of state.

Another approach is to implement extensive services in the router itself [21], to permit more efficient and sophisticated multicast routing options and encourage communication and cooperation between IP and higher-layer protocols.

One advantage of router-assisted mechanisms is that error recovery packets can be directed to a subtree that has experienced loss, thus minimizing *exposure* (the receipt of error recovery packets by receivers that have not experienced loss) [20, 22]. (The alternative would be to assign a separate multicast address to the group contained within the subtree.)

Routers can also be used, with or without local controllers, to aggregate NACKs and use stored information about these NACKs to minimize exposure [22].

Mechanisms, Communications Patterns, and Policies to Satisfy the Requirements

The basic mechanism for the provision of a reliable multicast service is packet exchange. These packets contain specific fields that are pertinent to the function being implemented.

In addition to the packets distributing the original copy of the data, there is a substantial amount of communication among the participants. We note three *communications patterns* that recur in many situations: unicast (point-to-point), single-sender multicast (point-to-multipoint), and multiple-sender multicast (multipoint-to-multipoint). These patterns are combined in many ways to permit achieving the goals of the users.

Finally, there are policies followed that determine which packets may be sent, to whom, under what conditions. We will formulate our classification based on these criteria.

We investigate five sets of requirements: data distribution, error detection and reporting, error recovery, group management, and ordering. Each will be viewed from the perspective of basic mechanisms, communications patterns, and possible policies.

Data Distribution

Data distribution is the original distribution of the data from the sender(s) to the receivers. It takes place using the multicast service provided by IP. For point-to-multipoint distribution, there is a single sender; for multipoint-to-multipoint distribution, there are multiple senders. These senders have *no* responsibility for error recovery or group management.

Thus, the mechanism is IP multicast, the communications pattern is determined by the underlying routing protocol, and there is no specific policy, as the goal is just to deliver the packets in a best effort way.

Error Detection and Reporting

As noted previously, multicast protocols can be grouped into two broad classes, sender-reliable and receiver-reliable, depending on whether the sending or receiving end is responsible for implementing the reliability mechanisms (i.e., depending on where the LES is located). In this subsection the mechanisms, communications patterns, and policies for error detection and reporting are outlined. The next subsection will consider error recovery. A summary of the communications patterns and policies for both error reporting and error recovery is given in Table 2.

Mechanisms — The mechanism used to detect missing data is the sequence number. All packets are labeled with a sequence number, which permits the packet to be put in its proper place prior to delivery to the user. For the single-sender case, this sequence number is simply the position of the packet in the series of packets issued by that sender. For the multiple-sender case, this sequence number is a (token number, position) pair, especially if any form of global ordering is required.

Communications Patterns — The appropriateness of an error reporting method depends very heavily on the scalability requirements. For sender-reliable error recovery, it will normally be appropriate for the receiver to unicast its error report to the sender, since the sender "needs to know" everything anyway. For receiver-reliable error recovery, two patterns are evident:

- Send the error report using unicast transmission, and let the "recovery agent" decide whether to use unicast repair or multicast repair.

Protocol	Error reporting							Error recovery		
	Comm pat		Policy					Comm Pat		Policy
	uc	mc	ACK	NACK	Supp	SNACK	TRACK	uc	mc	
XTP 3.6		x	x	x	x				x	SR
XTP 4.0	x		x	x					x	SR
RMP		x	x	x					x	SO
SRM		x		x	x				x	FRO
PGM		x		x	x				x	HRO
LGMP	x	x	x	x		x		x	x	HRO
RMTP	x		a	x				x	x	HRO
RMTP-II	x		x	x			x	x	x	HRO
LPC	x			x					x	HFEC

uc: Unicast
mc: Multicast
ACK: (Positive) acknowledgment
NACK: Negative acknowledgment
Supp: Suppression of acknowledgments
SNACK: Semi-negative acknowledgment
TRACK: Tree-based acknowledgment

SR: Sender-reliable
SO: Receiver-reliable, sender-oriented
FRO: Receiver-reliable, flat receiver-oriented
HRO: Receiver-reliable, hierarchical receiver-oriented
HFEC: Hybrid forward error correction

Table 2. *Error management communications patterns and policies.*

• Multicast the repair request so that other receivers can suppress their announcements.

In the second case, the repair must of necessity be multicast, to ensure that all receivers who need the repair actually see it. Depending on the error recovery architecture, the recovery agent may be the sender or some other entity. Further details are given later in this section.

Policies — The effectiveness of the sequence number as an error detection mechanism depends very much on the reordering properties of the underlying delivery mechanism. If the network level always delivers packets in order, a snaphot of the received sequence will give an accurate indication of what has been lost. However, if the network level can reorder packets [23], a particular snapshot may erroneously categorize packets as lost, even though they will soon arrive at the receiver. In this case, it is a worthwhile policy to delay reporting an error for a period of time.

If the error recovery is based on resending the lost packet(s), the error reporting mechanism can provide the highest contiguous packet (or byte) received (Go-Back-*N*), or the set of packets received (or not received) (Selective Repeat). If the error recovery is based on sending forward error correction (FEC) packets, it is sufficient to report the number of packets lost in a specific FEC block [24]. Finally, if the return path is nonexistent, there may be no way to report the error; in this case the protocol must rely on the FEC being sufficient to deal with the level of errors experienced.

Possible policies for *when* to report are:

- Always report a missing packet, perhaps with some time delay to account for network misordering.
- Report only when asked (i.e., sender controls).
- Report under specific circumstances (i.e., when specific conditions are met).

Always reporting a lost packet works well when the likelihood of loss is low and the underlying network is order-preserving. However, this is becoming less and less likely in the Internet [23]. Reporting only when asked is appropriate for sender-reliable protocols. It puts the control over status reporting in the hands of the sender, and permits the sender to control the frequency and timing of status reports. Receiver-reliable protocols report when the receiver detects loss.

Error Recovery

This section is concerned with the actions taken to recover from errors, that is, to make the protocol reliable. It is here that the entire concept of reliability takes its meaning. After stating the mechanisms used and the possible communications patterns, we explore the various policies in use, beginning with those that are capable of strong reliability and moving to those that are only capable of weak reliability, because to do otherwise would overload the sender.

The communications patterns and policies for error recovery are shown in the rightmost two columns of Table 2.

Mechanisms — The following are the mechanisms used:

- Send a replacement for the specific packet that was lost.
- Send an FEC packet.

The general term *recovery object* will be used in the sequel.

Communications Patterns — The following are some possible communications patterns:

- The sender unicasts the recovery object to the receiver.
- The sender multicasts the recovery object to the entire set of receivers.
- The local controller unicasts the recovery object to the receiver.
- The local controller multicasts the recovery object to the entire set of local receivers.

As noted previously, use of multicast by the receiver forces the recovery agent (original sender or local agent) to multicast the recovery object. In addition, if the specific receivers are not identified, or the lost objects are not identified, the recovery object *must* be multicast.

Policies — Macker *et al.* [25] provides a taxonomy of various error recovery policies:

Sender-reliable (SR): The LES is at the sender. The sender has absolute control, and is likely to require a positive acknowledgment (ACK) from each receiver.

Receiver-reliable (RR): The LES is at the receivers. The sender has some or no control. This class is subdivided into three cases, each more scalable than the previous one:

- Sender-oriented: (SO) The receiver sends a negative acknowledgment (NACK) to the sender, which issues the repair.
- Flat, receiver-oriented: (FRO) The receiver multicasts the NACK, and any receiver that has the data can issue the repair. Suppression (Supp) of redundant NACKs and repairs can assist in making this case more scalable.
- Hierarchical, receiver-oriented: (HRO) The receiver issues the NACK to a local controller, which aggregates it with others for forwarding up the hierarchy. The concepts of semi-NACK (SNACK) and tree-based ACK (TRACK) are further explained in the section on example protocols. The local controller may also retransmit the missing data, thus reducing the effect on receivers that have not lost the packets.

When a group is enormous, and the individual branches of the distribution tree have uncorrelated losses, it is sufficient to report the *number* of packets lost and issue FEC packets to effect the repairs. Lacher *et al.* [24] call this *hybrid FEC* (HFEC).

If there is no feedback from the receivers at all, a fixed number of FEC packets can be sent along with the original data. The advantage of this solution is that it can scale to an unbounded number of receivers. However, the maximum error rate must be below the rate covered by the ratio of FEC packets to data packets [24].

Group Management

The range of group management requirements is very wide. These requirements are satisfied by using the network-level mechanisms for joining and leaving groups provided by IGMP [26], the best effort data transfer service of IP, and (in certain cases) the data caches established by the protocol or application.

Mechanisms — At the transport layer, if there is no requirement to maintain group integrity, there is no requirement for specific join mechanisms at that level. The network-level join must be done by the receiver, but no action is taken (or necessary) at the transport level.

However, if group membership must be known and/or controlled, there must be an exchange of packets between the host that wishes to be a member and the host that requires knowledge of the group membership. Six actions are required: invitation to join, request to join, join ACK, invitation to leave, request to leave, and leave ACK. (The invitation to leave may be a requirement, not a request.) The mechanism may be an explicit packet (e.g., join) or a field in an existing packet (e.g., using an END bit in a packet header to request/force a leave action).

Communications Patterns — An invitation to join is of necessity multicast to the whole group. If configuration information is available to the new receiver, or a host is responding to an invitation to join, a request to join can be unicast from the new receiver to the global or local controller. Otherwise, the new receiver must multicast its request to the global or local controller. The join ACK, if issued, will be unicast by the appropriate controller to the new receiver to ensure that it is aware of its acceptance to the group (or local group).

An invitation to leave will normally be unicast to an individual receiver, unless it is multicast to the entire group (in which case it is a directive to dissolve the group). A request to leave will be unicast to the local or global controller, and a leave ACK will be unicast by the appropriate controller to the departing receiver.

Policies — The idea of an LES can be used to improve our understanding of group management. Similar to data reliability, mechanisms for group management require feedback from the receivers. The location of the LES determines not only what type of error recovery can be attempted, but also how reliably the overall group can be managed.

Most of the example protocols listed in Table 2 have no built-in policy for group management. The policies that exist are sufficiently application-dependent that they cannot serve as a classification tool. If the group membership is to be managed, either one member must be appointed as a master or all members must reach consensus as to the (acceptable) membership. In either case, the LES will of necessity be sender-oriented, and the one or several masters will have complete knowledge of the group. This approach is only valid for small groups.

For protocols where membership change is permitted, after the group forms there must be a join/leave policy. This policy may be centrally enforced (for small groups), or distributed within the hierarchy (for large groups).

Depending on how valuable the past data is to late joiners, the protocol may require the transmitter to buffer all or part of the data using techniques such as multilevel caching. The late joiners may then look at the sequence number of the packet that was currently multicast by the transmitter and send a special request for retransmission of all the packets with sequence numbers lower than the current one.

The question of how long the data need to be retained by the transmitter also depends on the needs of the application. If the application requires that all the data be available throughout the life of the transport connection, sufficient buffering must be available to satisfy this demand. On the other hand, applications may specify a retention time for the data; once this time expires, the data may be discarded.

Ordering

Mechanisms — Ensuring that ordering constraints are honored requires two mechanisms: one to negotiate the labels used to ensure the ordering, and the second to distribute the data using these labels. The label is sometimes called a *token*.

Communications Patterns — Two approaches can be used to manage the token: centralized and decentralized. In the centralized approach, the token is owned by a master sender. This results in simpler management of the token, but does not scale well. The master sender is also a single point of failure. In this case, the reliable transport protocol that supports ordering must provide (at a minimum) a unicast link to the master sender (the one handing out the tokens) and the ability to multicast from any participant to all other participants.

A receiver requesting a token will typically unicast to the master sender. The response will be unicast to the receiver.

The release of the token may involve an explicit message to the master sender or be implicit (e.g., a bit set in the final packet).

In the distributed approach, the token circulates on a (virtual) ring among the senders. Management is more difficult, but the solution has much better scalability and exhibits no single point of failure.

In this case, each participant needs a unicast link to its neighbors on the ring and the ability to multicast to all participants. The token will be circulated on the unicast links, while

other functions (membership management, ring formation, etc.) will use the multicast capability.

In both cases, data will be distributed using the normal multicast data distribution mechanisms.

Policies — The policies used depend on the ordering policy required by the application. Since a wide variety of policies can be envisioned, and the exact policy required may need knowledge of application-level data to achieve it, the most successful approach is likely to have no (built-in) policies at all, but provide a set of comprehensive token generation and distribution primitives on top of a communications subsystem that offers a choice of 1-to-1, 1-to-*N*, or *M*-to-*N* communications paths.

Example Protocols

In this section, several examples of multicast protocols are given. They illustrate the range of requirements and solutions that have been adopted. Recall that Table 1 gives the correlation between user requirements and these example protocols.

User Datagram Protocol

UDP provides an end-to-end variant of the network-level best effort service, with no error recovery mechanisms, group control, or ordering. There can be an arbitrary number of senders and receivers, as long as each knows the group address.

Xpress Transport Protocol

XTP was originally introduced by Chesson as the Xpress Transfer Protocol. It was a network- and tranport-layer protocol, with unified unicast and multicast features [27]. The design was pure mechanism, and was based on a set of independent functions (error control, flow control, rate control, priorities, etc.) that could be individually selected. With the introduction of Revision 4.0, the name was changed to Xpress *Transport* Protocol, and the multicasting features were considerably strengthened. Revision 4.0b [28] further strengthened the multicast features. All revisions allow only a single multicast sender. We will use the short forms XTP-3 and XTP-4 for XTP up to revision 3.6 and XTP since Revision 4.0, respectively.

Requirements — XTP is a single-sender (point-to-multipoint) protocol, intended to serve small to medium groups. XTP-3 provides weak reliability (because it uses slotting and damping to suppress redundant error reports), while XTP-4 provides strong reliability. XTP-3 has relatively weak group management capabilities, while XTP-4 provides strong *mechanisms* for group management.

Architecture — The error recovery architecture is flat, with the LES at the sender. The group management architecture is flat. XTP has no concept of assistance from network elements.

Mechanisms and Policies — Data reliability is provided by reporting the highest sequence number received, and (if one or more gaps have been detected in the data stream) a list of the additional sequence numbers received (Go-Back-*N* and selective repeat are both supported). The error reports are multicast in XTP-3 and unicast to the sender in XTP-4. Status is only reported when the sender explicitly asks for it.

Error recovery is achieved by resending the missing packet(s); the policy is sender-reliable.

There are no explicit congestion control policies in the specification, although all implementations provide this capacity.

An XTP multicast group is normally formed when an invitation to join is issued by the sender. This invitation is a FIRST packet with the multicast bit set in the header. The receiver responds to the invitation with a JCNTL (join control) packet (XTP 4.0b and later), and the sender acknowledges the member with another JCNTL. This JCNTL has specific fields in it to negotiate quality of service. It is possible to join late by multicasting a JCNTL to the group.

Receivers depart by exchanging the CLOSE bits in the header, or can be forced to leave with the END bit in the header.

There are no *policies* for managing the group; this is left to the application.

Reliable Multicast Protocol

RMP was developed by Whetten, Montgomery, and Kaplan [13].

Requirements — RMP provides a totally ordered, reliable, atomic multicast service on top of an unreliable multicast service such as IP multicasting. RMP provides a wide range of guarantees, from unreliable delivery to totally ordered delivery, to *K*-resilient, majority resilient, and totally resilient atomic delivery.

Architecture — The error recovery architecture is flat, with the LES at the receivers. The group management architecture is flat. RMP has no concept of assistance from network elements.

Mechanisms and Policies — Each packet is marked with a unique identification, and multicast to the entire group. NACKs are used for faster error recovery, and a limited number of ACKs is used to guarantee delivery. These ACKs are sent by a *token site* that constantly rotates among the members of the multicast group, thus making the protocol fully and symmetrically distributed so that no site bears an undue portion of the communication load.

Responsibility for managing join and leave rests with the current token site, and a new member becomes the next token site to ensure that it has really joined the token ring.

RMP provides a windowed flow and congestion control algorithm that allows RMP to provide high performance over both LANs and WANs, even in the face of congestion.

Scalable Reliable Multicast Protocol

SRM is a reliable *M*-to-*N* multicast framework that has been designed for lightweight sessions and application-level framing. The algorithms of this framework are efficient and robust, and scale well to both very large networks and very large sessions [29]. SRM has been prototyped in the distributed whiteboard application (*wb*). The design of SRM is based on the application-level framing (ALF) concept, which says that the best way to meet diverse application requirements is to leave as much functionality and flexibility as possible to the application. Therefore, SRM is designed to meet only the minimum definition of reliable multicast; that is, eventual delivery of all data to all the group members, without enforcing any delivery order [29].

Requirements — SRM is a multiple-sender (multipoint-to-multipoint) protocol serving large groups. It provides weak data reliability and minimal group management.

Architecture — The error recovery architecture is flat, with the LES at the receivers. The group management architecture is flat. SRM has no concept of assistance from network elements.

Mechanisms and Policies — SRM uses flat receiver-oriented receiver-initiated error recovery. When a receiver detects a hole in the data stream, it multicasts a repair request to the entire group. However, to avoid implosion as well as identical requests for the same data, SRM uses the slotting and damping mechanisms used by XTP-3. Receivers delay themselves randomly before sending a repair request, and refrain from sending if a similar request is sent within the delay time. As with the original data, repair requests and retransmissions are always multicast to the entire group. To avoid transmitting duplicate repairs, hosts that wish to transmit a repair delay themselves randomly before transmitting the repair. The delay time is a function of the distance in seconds between the member that wishes to send the repair and the one that triggered the request or repair [29]. Thus, it is more likely that a host closer to the point of failure will time out first and multicast the request.

Reliable data delivery in SRM is ensured as long as each datum is available from at least one member. This has the advantage of reducing the buffering requirements on all members within the communication session.

Pragmatic General Multicast

PGM was specified by Speakman *et al.* in RFC 3208 [22].

Requirements — PGM is a reliable multicast transport protocol for applications that "require ordered or unordered, duplicate-free, multicast data delivery from multiple sources to multiple receivers." (Note that the ordering referred to in this quotation is single-source ordering; multiple sources are considered by PGM to be independent.) PGM guarantees that a receiver in the group either receives all data packets from transmissions and repairs, or is able to detect unrecoverable data packet loss. PGM is specifically intended as a workable solution for multicast applications with basic reliability requirements. Its central design goal is simplicity of operation with due regard for scalability and network efficiency [22].

Architecture — The error recovery architecture is hierarchical, with the LES at the receivers. There is no group management. PGM is designed to use a significant amount of state in the routers, to both aggregate NACKs and permit the use of stored information about NACKs to minimize exposure.

Mechanisms and Policies — PGM uses hierarchical receiver-oriented error recovery. Under the premise that NACKs are for repair and ACKs for buffer management, the choice was made to use NACKs exclusively, and use something else to decide on buffer expiry. (The transmit window is advanced by a source according to a purely local strategy.) The NACKs are unicast towards the network element that the multicast data are coming from, aggregated and forwarded up towards the source, unless the repair data are provided by a *Designated Local Repairer*. NACKs from other receivers on the same subnet are suppressed through receipt of NACK confirmations, which are multicast to the subnet. Since there is no concept of group management in PGM, there are no group management mechanisms or policies. Similarly, there are no policies for multisender ordering.

Local Group Based Multicast Protocol (LGMP)

LGMP was developed by Hofmann [30, 31]. It supports reliable and semi-reliable transfer of both continuous media and data files. The protocol improves scalability and performance by using subgroups (also called *local groups*) for local ACK processing and error recovery, as defined by the local group concept.

Local groups are formed by dynamic organization of receiver sets and managed in turn by a special receiver called a *group controller* (GC), which handles status responses and coordinates local retransmissions. The selection of appropriate receivers as GCs is based on the current state of the network and the receivers themselves. This process is not part of the LGMP itself, and is implemented by a separate configuration protocol called the Dynamic Configuration Protocol (DCP).

Requirements — LGMP is a single-sender (point-to-multipoint) protocol, intended to serve large groups. It provides reliable and semi-reliable transfer.

Architecture — The error recovery architecture is hierarchical, with the LES at the receivers. The group management architecture is also hierarchical, using DCP, which is managed by the GCs. LGMP has no concept of assistance from network elements.

Mechanisms and Policies — Data distribution is 1-to-*N*. Data reliability is provided when a receiver unicasts an ACK to release data units from the sender's buffer or a NACK to request retransmission of missed data units.

If a member of a Local Group does not receive a data unit, local error recovery is first attempted. In this case the GC will send a SNACK to its parent, indicating that the packet has been received in the group, but not by all members. The local recovery is managed by the GC, which will unicast or multicast the repair, depending on how many members of the subgroup have indicated data loss.

LGMP defines two different modes of performing local retransmissions: *load-sensitive* and *delay-sensitive*. It is up to the application to choose the appropriate mode according to its requirements. It is also possible for different GCs to operate in different modes.

LGMP separates the signal for indicating congestion from the algorithm for congestion control. It provides mechanisms to detect network congestion based on the status reports of each receiver, but leaves it up to the application to choose the algorithm to deal with congestion.

Instead of integrating mechanisms into LGMP to define local groups, and establish and maintain these logically structured group hierarchies, a new protocol named Dynamic Configuration Service (DCS), has been defined that performs these tasks.

Reliable Multicast Transport Protocol

RMTP [32] was developed by Paul *et al.* It permits a single sender to deliver bulk information to a large community of users. A new version of the protocol, RMTP-II, was introduced by Whetten and Taskale [33]. It improves RMTP by providing for multiple senders, and introduces a number of new features to meet the needs of emerging applications.

Requirements — The primary requirement for RMTP and RMTP-II is receiver scalability. RMTP provides indefinite availability of data through the use of a two-level cache. RMTP-II provides only bounded time reliability. Within the bounded time, RMTP-II provides a guarantee of delivery. The RMTP sender has no knowledge of the set of receivers; RMTP-II provides mechanisms to ensure reporting of the set of receivers.

Architecture — The error recovery architecture is hierarchical with the LES at the receivers. The group management architecture is hierarchical. Both architectures are populated with

designated receivers and ACK processors to manage the nodes of the hierarchy. RMTP has no concept of assistance from network elements, while RMTP-II can use such assistance, but does not depend on it.

Mechanisms and Policies — An RMTP sender divides the data to be sent into fixed-size packets and transmits them using the global multicast tree. Every packet is assigned a unique sequence number that defines the overall order before they are multicast to the group.

Designated receivers learn of missing packets from the ACKs unicast to them by the receivers in their local region. The ACK packets contain the sequence number of the first in-sequence packet not received by the receiver, plus an ACK bitmap indicating which out-of-sequence packets have been successfully received.

The DRs keep track of the sequence number of the lowest in-sequence packet that has not been successfully received by a receiver. The data packets not received are added to a retransmission queue, along with the identity of the receiver that requested the retransmission. If the number of receivers requesting retransmission of a packet exceeds a predefined threshold, the packet is remulticast to all the members of the group; otherwise, the packet is retransmitted via unicast to the receivers who requested the retransmission.

Late joining receivers can make an immediate transmission request to permit retrieving data sent earlier. To permit going back to the original start of transmission, data are stored at sender and DRs, partly in memory, and partly in a disk cache.

An ACK processor is chosen to be the one with the least number of hops from the receiver.

RMTP uses a rate-based windowed flow control mechanism to avoid overloading slow receivers and links with low bandwidth. RMTP also implements the slow-start algorithm to prevent a sender from flooding an already congested network.

The tree-based error recovery algorithms used in RMTP-II form the basis for the ongoing work on TRACK schemes in the RMT working group [5].

Hybrid FEC Protocols

Lacher *et al.* [24] offer a performance analysis of a variety of protocols that are based on the idea of reporting the *number* of packets lost in a (sub)tree and then sending a set of FEC packets to repair the damage. The advantage is that the individual packets do not need to be resent; sending n FEC repair packets is sufficient to compensate for loss of up to n packets at a receiver, without knowing which ones have been lost.

The protocol designated *Protocol C* in their paper (LPC) is included to show an extreme situation. (It scales to 10^6 receivers.)

Analysis

Reliable multicasting protocols span a wide range of requirements. While they have many attributes in common, there are factors that distinguish them sufficiently to make the idea of a single generic multicast protocol impossible. These are the number of senders, number of receivers (the need for scalability), data reliability needed, group knowledge required, and need for ordering.

Number of Senders

The existing reliable multicast protocols divide into three groups: single-sender protocols, multiple-sender protocols where the additional senders essentially (or actually) work through a single master, and multiple-sender protocols.

Scalability

Scalability is the issue that has the most profound effect on the design of a multicast protocol. It is the primary reason why the goal of a single generic multicast protocol is unattainable. We summarize below the effect this one issue has on the whole area of reliable multicast protocol design.

Architecture — Small groups exhibit a flat peer-to-peer architecture. The participants are all aware of each other.

As the group size gets larger, the architecture must become hierarchical. The separation into groups is achieved by assignment of distinct multicast addresses, unless routers provide specific assistance for the separation. The protocol will then come to depend on the local controllers, which may be specific to that protocol or (relatively) protocol-independent. Development of a generic model for this functionality will help to facilitate the deployment of multicast as a commercial offering, because it will allow families of multicast protocols to be supported without having to install protocol-specific code in routers throughout the served domain.

Communications Patterns — For small groups, the set of senders and set of receivers are likely to be the same. In this case, both unicast and multicast error reporting can be used effectively. Error repair can be done either way. With unicast reporting and unicast or multicast repair, absolute reliability can be achieved.

For medium groups, multicast error reporting with suppression can be used effectively. This must of necessity be combined with multicast repair. Absolute reliability is not achievable, although it is possible to come very close.

For large groups, the likelihood of a receiver also being a sender is very small. If the cost of being a multicast sender is larger than the cost of being a receiver, the receivers are likely to unicast their error reports, and the recovery agent can choose between unicast and multicast for the repair traffic. If the receivers multicast their error reports, the recovery agent must multicast the repair traffic.

Policies for Data Reliability — Small and medium groups tend to use replacement of the lost packet as the mechanism for data reliability. As the group gets larger, hybrid and pure FEC become necessary. Hybrid FEC is still capable of replacing all lost packets, but pure FEC is limited by the (pre-assumed) estimate of the number of lost packets per grouping of packets sent.

Small groups can achieve absolute reliability if they use a policy where the LES is at the sender. Protocols exist that can provide reliability even in the presence of (temporary) network partitioning.

As the group size gets larger, however, the responsibility for determining loss must move to the receivers, and the protocols become at best semi-reliable. They can, however, still deal with arbitrary loss probabilities.

As the group size gets enormous, the amount of feedback that can be used becomes small, and it is necessary to use FEC or hybrid FEC approaches.

Finally, if the amount of feedback is reduced to zero, FEC must be used, regardless of group size. It is capable of recovering only from a fixed error percentage. If this percentage is exceeded, delivery of the data cannot be guaranteed.

Policies for Group Management — Managing group membership requires an exchange of control packets that are separate from the packets used to carry the data flow.

Small groups can achieve complete knowledge of the group

membership within all members. As the group size gets larger, however, the group knowledge must become weaker, and the policies for determining the right to be in a group must become distributed.

However, if the turnover of group members is not too rapid, it is possible to have summary reports on group membership that will provide information (at least to the root of the distribution tree) about the entire membership.

As the group membership becomes enormous, it becomes infeasible to garner any knowledge about the set of receivers.

Ordering

The need for ordering only applies when there are multiple senders. By the nature of the algorithms that must be used to maintain the ordering, it is unlikely to be required for groups larger than a few members.

Conclusion

Many reliable multicast protocols have been designed that respond to a wide range of user requirements and network environments.

These protocols can be differentiated by examining the set of requirements they meet, the architecture they use for interaction among the participants, the mechanisms used to effect this interaction, and the policies that determine how and when to use the mechanisms.

Using several representative reliable multicast protocols as examples, the mapping between requirements and the other dimensions of the taxonomy has been explored. This permits identification of the dependencies, and is expected to lead to the development of additional building blocks that will aid in the design of future reliable multicast protocols.

Acknowledgments

I thank the anonymous reviewers for their detailed and helpful comments. The support of the Natural Sciences and Engineering Research Council of Canada, through its Discovery Grants program, and the support of Concordia University are gratefully acknowledged.

References

[1] J. Postel, "Transmission Control Protocol," IETF RFC 793, Sept.1981.
[2] A. Mankin *et al.*, "IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols," IETF RFC 2357, June1998.
[3] C. Diot, W. Dabbous, and J. Crowcroft, "Multipoint Communication: A Survey of Protocols, Functions, and Mechanisms," *IEEE JSAC*, vol. 15, no.3, Apr. 1997, pp. 27–90.
[4] K. Obraczka, "Multicast Transport Protocols: A Survey and Taxonomy," *IEEE Commun. Mag.*, Jan. 1998, pp. 94–102.
[5] "Reliable Multicast Transport Working Group Charter," http://www.ietf.org/html.charters/rmt-charter.html.
[6] K. Almeroth and M. Ammar, "Multicast Group Behavior in the Internet's Multicast Backbone(MBone)," *IEEE Commun. Mag.*, vol. 35, no. 6, June 1997.
[7] M. Handley *et al.*, "The Reliable Multicast Design Space for Bulk Data Transfer," IETF RFC 2887, Aug. 2000.
[8] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end Arguments in System Design," *ACM Trans. Comp. Sys.*, vol. 2, no. 4, 1984, pp.277–88.
[9] S. J. Golestani and K. Sabnani, "Fundamental Observations on Multicast Congestion Control in the Internet," *Proc. INFOCOM* New York, NY, Mar. 1999.
[10] A. Basu and S. J. Golestani, "Architectural Issues for Multicast Congestion Control," *Proc. Int'l. Wksp. Net. and Op. Sys. Support for Digital Audio and Video*, Basking Ridge, NJ, June 1999.
[11] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-Like Congestion Control for Layered Multicast Data Transfer," *Proc. INFOCOM*, San Francisco, CA, Mar./Apr. 1998.
[12] V. Jacobson *et al.*, "Requirements for Congestion Control for Reliable Multicast," Tech. rep., Slides from ftp://ftp.ee.lbl.gov/talks/sf-RMreq.ps, Sept. 1997.
[13] B. Whetten, T. Montgomery, and S. M. Kaplan, "A High Performance Totally Ordered Multicast Protocol," *Dagstuhl Sem. Distrib. Sys.*, 1994, pp. 33–57.
[14] S. E. Deering, "Host Extensions for IP Multicasting," IETF RFC 1112, Aug. 1989.
[15] K. C. Almeroth, "The Evolution of Multicast: From the Mbone to Interdomain Multicast to Internet2 Deployment," *IEEE Network*, Jan./Feb. 2000, pp. 10–20.
[16] D. C. Feldmeier and E. W. Biersack, "Comparison of Error Control Protocols for High Bandwidth-Delay Product Networks," *2nd IFIP WG6.1/WG6.4 Int'l. Wksp. Protocols for High-Speed Networks*, M. J. Johnson, Ed., Palo Alto, CA, Nov. 1990, pp. 271–95.
[17] D. F. Towsley, J. F. Kurose, and S. Pingali, "A Comparison of Sender-initiated and Receiver-initiated Reliable Multicast Protocols," *IEEE JSAC*, vol. 15, no.3, Apr. 1997, pp. 398–406.
[18] B. N. Levine and J. J. Garcia-Luna-Aceves, "A Comparison of Reliable Multicast Protocols," *Multimedia Sys.*, vol. 6, no. 5, 1998, pp. 334–48.
[19] P. Radoslavov *et al.*, "A Comparison of Application-level and Router-assisted Hierarchical Schemes for Reliable Multicast," *Proc. INFOCOM*, Anchorage, AK, Apr. 2001.
[20] C. Papadopoulos, G. Parulkar, and G. Varghese, "An Error Control Scheme for Large-Scale Multicast Applications," *Proc. INFOCOM*, San Francisco, CA, Mar./Apr. 1998, p. 1188.
[21] B. N. Levine and J. J. Garcia-Luna-Aceves, "Improving Internet Multicast with Routing Labels," *Int'l. Conf. Network Protocols*, Atlanta, GA, Oct. 1997.
[22] T. Speakman *et al.*, "PGM Reliable Transport Protocol Specification," IETF RFC 3208, Dec. 2001.
[23] J. C. R. Bennett, C. Partridge, and N. Shectman, "Packet Reordering is not Pathological Network Behavior," *IEEE/ACM Trans. Net.*, vol. 7, no. 6, Dec. 1999, pp. 789–98.
[24] M. S. Lacher, Jörg Nonnenmacher, and E. W. Biersack, "Performance Comparison of Centralized Versus Distributed Error Recovery for Reliable Multicast," *IEEE/ACM Trans. Net.*, Apr. 2000, pp. 224–38.
[25] J. P. Macker, J. E. Klinker, and M. S. Corson, "Reliable Multicast Data Delivery for Military Networking," *Military Commun. Conf.*, 10 1996, vol. 2, pp. 399–403.
[26] B. Cain *et al.*, "Internet Group Management Protocol, Version 3," IETF RFC 3376, Oct. 2002.
[27] T. W. Strayer, B. J. Dempsey, and A. C. Weaver, *XTP — The Xpress Transfer Protocol*, Addison-Wesley, 1992.
[28] W. T. Strayer, "Xpress Transport Protocol (XTP) Specification, version 4.0b," Tech. rep., XTP Forum, June 1998.
[29] S. Floyd *et al.*, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing," *IEEE/ACM Trans. Net.*, vol. 5, no. 6, Dec. 1997, pp. 784–803.
[30] M. Hofmann, "Adding Scalability to Transport Level Multicast," *Proc. 3rd Int'l. COST 237 Wksp.: Multimedia Telecommun. and Apps.*, Barcelona, Spain, Nov. 1996.
[31] M. Hofmann, "Enabling Group Communication in Global Networks," *Proc. Global Networking '97*, vol. II, June 1997, pp. 321–30.
[32] S. Paul *et al.*, "Reliable Multicast Transport Protocol (RMTP)," *IEEE JSAC*, vol. 15, no.3, Apr. 1997, pp. 407–21.
[33] B. Whetten and G. Taskale, "An Overview of Reliable Multicast Transport Protocol II," *IEEE Network*, vol. 14, no. 1, Jan. 2000.

Biography

J. WILLIAM ATWOOD [SM '81] (bill@cs.concordia.ca) received his B.Eng. from McGill University, his M.A.Sc. from the University of Toronto, and his Ph.D. from the University of Illinois at Urbana. He is a professor of computer science at Concordia University, where he does research on specification, validation, and evaluation of communications protocols.