Network Working Group
Request for Comments: 2887
Category: Informational

M. Handley
S. Floyd
ACIRI
B. Whetten
Talarian
R. Kermode
Motorola
L. Vicisano
Cisco
M. Luby
Digital Fountain, Inc.

August 2000

The Reliable Multicast Design Space for Bulk Data Transfer

Status of this Memo

Copyright Notice

Abstract

The design space for reliable multicast is rich, with many possible solutions having been devised. However, application requirements serve to constrain this design space to a relatively small solution space. This document provides an overview of the design space and the ways in which application constraints affect possible solutions.

1. Introduction

The term "general purpose reliable multicast protocol" is something of an oxymoron. Different applications have different requirements of a reliable multicast protocol, and these requirements constrain the design space in ways that two applications with differing requirements often cannot share a single solution. There are however many successful reliable multicast protocol designs that serve more special purpose requirements well.

In this document we attempt to review the design space for reliable multicast protocols intended for bulk data transfer. The term bulk data transfer should be taken as having broad meaning - the main limitations are that the data stream is continuous and long lived -

constraints necessary for the forms of congestion control we currently understand. The purpose of this review is to gather together an overview of the field and to make explicit the constraints imposed by particular mechanisms. The aim is to provide guidance to the standardization process for protocols and protocol building blocks. In doing this, we cluster potential solutions into a number of loose categories - real protocols may be composed of mechanisms from more than one of these clusters.

The main constraint on solutions is imposed by the need to scale to large receiver sets. For small receiver sets the design space is much less restricted.

2. Application Constraints

Application requirements for reliable multicast (RM) are as broad and varied as the applications themselves. However, there are a set of requirements that significantly affect the design of an RM protocol. A brief list includes:

- Does the application need to know that everyone received the data?
- Does the application need to constrain differences between receivers?
- Does the application need to scale to large numbers of receivers?
- Does the application need to be totally reliable?
- Does the application need ordered data?
- Does the application need to provide low-delay delivery?
- Does the application need to provide time-bounded delivery?
- Does the application need many interacting senders?
- Is the application data flow intermittent?
- Does the application need to work in the public Internet?
- Does the application need to work without a return path (e.g. satellite)?
- Does the application need to provide secure delivery?

In the context of standardizing bulk data transfer protocols, we can rule out applications with multiple interacting senders and intermittent data flows. It is not that these applications are unimportant, but that we do not yet have effective congestion control for such applications.

2.1. Did everyone receive the data?

In many applications a logically defined unit or units of data is to be delivered to multiple clients, e.g., a file or a set of files, a software package, a stock quote or package of stock quotes, an event notification, a set of slides, a frame or block from a video. An application data unit (ADU) is defined to be a logically separable unit of data that is useful to the application. In some cases, an application data unit may be short enough to fit into a single packet (e.g., an event notification or a stock quote), whereas in other cases an application data unit may be much longer than a packet (e.g., a software package).

A protocol may optionally provide delivery confirmation to ensure reliable delivery, i.e., a mechanism for receivers to inform the sender when data has been delivered. There are two types of confirmation, at the application data unit level and at the packet level. Application data unit confirmation is useful at the application level, e.g., to inform the application about receiver progress and to decide when to stop sending packets about a particular application data unit. Packet confirmation is useful at the transport level, e.g., to inform the transport level when it can release buffer space being used for storing packets for which delivery has been confirmed.

Some applications have a strong requirement for confirmation that all the receivers got an ADU, or if not, to be informed of which specific receivers failed to receive the entire ADU. Examples include applications where receivers pay for data, and reliable file-system replication. Other applications do not have such a requirement. An example is the distribution of free software.

If the application does need to know that every receiver got the ADU, then a positive acknowledgment must be received from every receiver, although it may be possible to aggregate these acknowledgments. If the application needs to know precisely which receivers failed to get the ADU, additional constraints are placed on acknowledgment aggregation.

It should be noted that different mechanisms can be used for ADU-level confirmation and packet-level confirmation in the same application. For example, an ADU-level confirmation mechanism using

positive acknowledgments may sit on top of a packet-level NACK or FEC-based transport. Typically this only makes sense when ADUs are significantly larger than a single packet.

2.2. Constraining differences

Some applications need to constrain differences between receivers so that the data reception characteristics for all receivers falls within some range. An example is a stock price feed, where it is unacceptable for a receiver to suffer delivery that is delayed significantly more than any other receiver.

This requirement is difficult to satisfy without harming performance. Typically solutions involve not sending more than a limited amount of new data until positive acknowledgments have been received from all the receivers. Such a solution does not cope with network and end-system failures well.

2.3. Receiver Set Scaling

There are many applications for RM that do not need to scale to large numbers of receivers. For such applications, a range of solutions may be available that are not available for applications where scaling to large receiver sets is a requirement.

A protocol must achieve good throughput of application data units to receivers. This means that most data that is delivered to receivers is useful in recovering the application data unit that they are trying to receive. A protocol must also provide good congestion control to fairly share the available network resources between all applications. Receiver set scaling is one of the most important constraints in meeting these requirements, because it strictly limits the mechanisms that can be used to achieve these requirements to those that will efficiently scale to a large receiver population. Acknowledgement packets have been employed by many systems to achieve these goals, but it is important to understand the strength and limitations of different ways of using such packets.

In a very small system, it may be acceptable to have the receivers acknowledge every packet. This approach provides the sender with the maximum amount of information about reception conditions at all the receivers, information that can be used both to achieve good throughput and to achieve congestion control.

For larger systems, such "flat ACK" schemes cause acknowledge implosions at the sender. Attempts have been made to reduce this problem by sending aggregate ACKs infrequently [RMWT98, BC94], but it is very difficult to incorporate effective congestion control into such protocols because of the spareceness of feedback.

Using negative acknowledgments (NACKs) instead of ACKs reduces this problem to one of NACK implosion (only from the receivers missing the packets), and because the sender really only needs to know that at least one receiver is missing data in order to achieve good throughput, various NACK suppression mechanisms can be applied.

An alternative to NACKs is ACK aggregation, which can be done by arranging the receivers into a logical tree, so that each leaf sends ACKs to its parent which aggregates them, and passes them on up the tree. Tree-based protocols scale well, but tree formation can be problematic.

Other ACK topologies such as rings are also possible, but are often more difficult to form and maintain than trees are. An alternative strategy is to add mechanisms to routers so that they can help out in achieving good throughput or in reducing the cost of achieving good throughput.

All these solutions improve receiver set scaling, but they all have limits of one form or another. One class of solutions scales to an infinite number of receivers by having no feedback channel whatsoever in order to achieve good throughput. These open-loop solutions take the initial data and encode it using an FEC-style mechanism. This encoded data is transmitted in a continuous stream. Receivers then join the session and receive packets until they have sufficient packets to decode the original data, at which point they leave the session.

Thus, it is clear that the intended scale of the session constrains the possible solutions. All solutions will work for very small sessions, but as the intended receive set increases, the range of possible solutions that can be deployed safely decreases.

It should also be noted that hybrids of these mechanisms are possible, and that using one mechanism at the packet-level and a different (typically higher overhead) solution at the ADU level may also scale reasonably if the ADUs are large compared to packets.

2.4. Total vs Semi-reliable

Many applications require delivery of application data units to be totally reliable; if any of the application data unit is missing, none of the received portion of the application data unit is useful. File transfer applications are a good example of applications requiring total reliability.

However, some applications do not need total reliability. An example is audio broadcasting, where missing packets reduce the quality of the received audio but do not render it unusable. Such applications can sometimes get by without any additional reliability over native IP reliability, but often having a semi-reliable multicast protocol is desirable.

2.5. Time-bounded Delivery

Many applications just require data to be delivered to the receivers as fast as possible. They have no absolute deadline for delivery.

However, some applications have hard delivery constraints - if the data does not arrive at the receiver by a certain time, there is no point in delivering it at all. Such time-boundedness may be as a result of real-time constraints such as with audio or video streaming, or as the result of new data superseding old data. In both cases, the requirement is for the application to have a greater degree of control over precisely what the application sends at which time than might be required with applications such as file transfer.

Time-bounded delivery usually also implies a semi-reliable protocol, but the converse does not necessarily hold.

3. Network Constraints

The properties of the network in which the application is being deployed may themselves constrain the reliable multicast design space.

3.1. Internet vs Intranet

In principle the Internet and intranets are the same. In practice however, the fact that an intranet is under one administration might allow for solutions to be configured that can not easily be done in the public Internet. Thus, if the data is of very high value, it might be appropriate to enhance the routers to provide assistance to a reliable multicast transport protocol. In the public Internet, it is less likely that the additional expense required to support this state in the routers would be acceptable.

3.2. Return Path

In principle, when feedback is required from receivers, this feedback can be multicast or unicast. Multicast feedback has advantages, especially in NACK-based protocols where it is valuable for NACK suppression. However, it is not clear at this time whether all ISPs will allow all members of a session to send to that session. If multicast feedback is not allowed, then unicast feedback can almost always be substituted, although often at the expense of additional messages and mechanisms.

Some networks may not allow any form of feedback however. The primary example of this occurs with satellite broadcasts where the back channel may be very narrow or even non-existent. For such networks the solution space is very constrained - only FEC-based encodings have any real chance of working. If the receivers are direct satellite receivers, then no congestion control is needed, but it is dangerous to make such assumptions because it is possible for a satellite hop to feed downstream networks. Thus, congestion control still needs to be considered with solutions that do not have a return path.

3.3. Network Assistance

A reliable multicast protocol must involve mechanisms running in end hosts, and must involve routers forwarding multicast packets. However under some circumstances, it is possible to rely on some additional degree of assistance from network elements. Broadly speaking we can cluster RM protocols into four classes depending on the degree of support received from other network elements.

No Additional Support
: The routers merely forward packets, and only the sender and receivers have any reliable multicast protocol state.

Layered Approaches
: Data is split across multiple multicast groups. Receivers join appropriate groups to receive only the traffic they require. This may in some cases require fast join or leave functionality from the routers, and may require more forwarding state in the routers.

Server-based Approaches
: Additional nodes are used to assist with data delivery or feedback aggregation. These additional nodes might not be normal senders or receivers, and may be present on the distribution or feedback tree only to provide assistance to the reliable multicast protocol. They would not otherwise receive the multicast traffic.

Router-based Approaches

With router-based approaches, routers on the normal data distribution tree from the sender to the receivers assist in the delivery of data or feedback aggregation or suppression. As routers can directly influence multicast routing, they have more control over which traffic goes to which group members than server-based approaches. However routers do not normally have a large amount of spare memory or processing power, which restricts how much functionality can be placed in the routers. In addition, router code is normally more difficult to upgrade than application code, so router-based approaches need to be very general as they are more difficult to deploy and to change.

4. Good Throughput Mechanisms

Two main concerns that a RM protocol must address are congestion control and good throughput. Packet loss plays a major role with respect to both concerns. The primary symptom of congestion in many networks is packet loss. The primary obstacle that must be overcome to achieve good throughput is packet loss. Thus, measuring and reacting to packet loss is crucial to address both concerns. RM solutions that address these concerns can be roughly categorized as using one or more of the following techniques:

- Data packet acknowledgment.
- Negative acknowledgment of missing data packets.
- Redundancy allowing not all packets to be received.

These techniques themselves can be usefully subdivided, so that we can examine the parts of the requirement space in which each mechanism can be deployed. In this section, we focus on using these mechanisms for achieving good throughput, and in the next section we focus on using these mechanisms for congestion control.

4.1. ACK-based Mechanisms

The simplest ACK-based mechanism involves every receiver sending an ACK packet for every data packet it receives and resending packets that are lost by any receiver. Such mechanisms are limited to very small receiver groups by the implosion of ACKs received at the sender, and for this reason they are impractical for most applications.

Putting multiple ACKs into a single data packet [RMWT98] reduces the implosion problem by a constant amount, allowing slightly larger receiver groups. However a limit is soon reached whereby feedback to the sender is too infrequent for sender-based congestion control mechanisms to work reliably.

Arranging the receivers into a ring [WKM94] whereby an "ACK-token" is passed around the ring prevents the implosion problem for data. However ring creation and maintenance may itself be problematic. Also if ring creation does not take into account network topology (something which is difficult to achieve in practice), then the number of ACK packets crossing the network backbone for each data packet sent may increase O(n) with the number of receivers.

4.1.1. Tree-based ACK Mechanisms

Arranging the receivers into a tree [MWB+98, KCW98] whereby receivers generate ACKs to a parent node, which aggregates those ACKs to its parent in turn, is both more robust and more easily configured than a ring. The ACK-tree is typically only used for ACK-aggregation - data packets are multicast from the sender to the receivers as normal. Trees are easier to construct than rings because more local information can be used in their construction. Also they can be more fault tolerant than rings because node failures only affect a subset of receivers, each of which can easily and locally decide to by-pass its parent and report directly to the node one level higher in the tree. With good ACK-tree formation, tree-based ACK mechanisms have the potential to be one of the most scalable RM solutions.

To be simple to deploy, tree-based protocols must be self-organizing - the receivers must form the tree themselves using local information in a scalable manner. Such mechanisms are possible, but are not trivial. The main scaling limitations of tree-based protocols therefore come from the tree formation and maintenance mechanisms rather than from the use of ACKs. Without such a scalable and automatic tree-formation mechanism, tree-based protocols must rely on manual configuration, which significantly limits their applicability (often to intranets) and (due to the complexity of configuration) their scalability.

Orthogonal to the issue of tree formation is the issue of subtree retransmission. With appropriate router mechanisms, or the use of multiple multicast groups, it is possible to allow the intermediate tree nodes to retransmit missing data to the nodes below them in the tree rather than relying on the original sender to retransmit the data. This relies on there being a good correlation at the point of the intermediate node between the ACK tree and the actual data tree, as well as there being a mechanism to constrain the retransmission to

the subtree. A good automatic tree formation mechanism combined with the use of administrative scoped multicast groups might provide such a solution. Without such tree formation mechanisms, subtree retransmission is difficult to deploy in large groups in the public internet. This could also be solved by the use of transport-level router mechanisms to assist or perform retransmission, although existing router mechanisms [FLST98] support NACK-based rather than ACK-based protocols.

Another important issue is the nature of the aggregation performed at interior nodes on the ACK-tree. Such nodes could:

1. aggregate ACKs by sending a single ACK when all their children have ACKed,

2. aggregate ACKs by listing all the children that have ACKed,

3. send an aggregated ACK with a NACK-like exception list.

For data packets, 1. is clearly more scalable, and should be preferred. However if the sender needs to know exactly which receivers received the data, 2. and 3. provide this information. Fortunately, there is usually no need to do this on a per-packet basis, but rather on a per-ADU basis. Doing 1. on a per packet basis, and 3. on a per ADU basis is the most scalable solution for applications that need this information, and suffers virtually no disadvantage compared to the other solutions used on a per-packet basis.

4.2. NACK-based mechanisms

Instead of sending an ACK for every data packet received, receivers can send a negative acknowledgment (NACK) for every data packet they discover they did not receive. This has a number of advantages over ACK-based mechanisms:

- The sender no longer needs to know exactly how many receivers there are. This removes the topology-building phase needed for ring- or tree-style ACK-based algorithms.

- Fault-tolerance is made somewhat simpler by making receivers responsible for reliability.

- Sender state can be significantly reduced because the sender does not need to keep track of the receivers state.

- Only a single NACK is needed from any receiver to indicate a packet that is missing by any number of receivers. Thus NACK suppression is possible.

The disadvantages are that it is more difficult for the sender to know that it can free transmission buffers, and that additional session level mechanisms are needed if the sender really needs to know if a particular receiver actually received all the data. However for many applications, neither of these is an issue.

4.2.1. NACK Suppression

The key differences between NACK-based protocols is in how NACK-suppression is performed. The goal is for only one NACK to reach the sender (or a node that can resend the missing data) as soon as possible after the loss is first noticed, and for only one copy of the missing data to be received by those nodes needing retransmission.

Different mechanisms come close to satisfying these goals in different ways.

- SRM [FJM95] uses random timers weighted by the round trip time between the sender and each node missing the data. This is effective, but requires computing the RTT to each receiver before suppression works properly.

- NTE [HC97] uses a sender-triggered mechanism based on random keys and sliding masks. This does not require random timers, and works for very large sessions, but makes it difficult to provide the constant low-level stream of feedback needed to perform congestion control.

- AAP [Ha99] uses exponentially distributed random timers and is effective for large sessions without needing to compute the RTT to each receiver.

- PGM [FLST98] and LMS [PPV98] use additional mechanisms in routers to suppress duplicate NACKs. In the case of PGM, router assistance suppliments SRM-stype random timers and localizes the suppression so that the whole group does not need suppressing.

The most general of these mechanisms is probably exponentially weighted random timers. Although SRM style timers can reduce feedback delay, they are harder to use correctly in situations where all the RTTs are not known, or where the number of respondees is

unknown. In contrast, exponentially weighted random timers work well across a large range of session sizes with good worst case delay characteristics.

Either form of random timer based mechanism can be supplemented by router-support where it is available. Sender triggered NACK mechanisms (e.g. [HC97]) are more difficult to integrate with router-based support mechanisms.

4.3. Replication

Some RM protocols can be designed so as to not need explicit reliability mechanisms except in comparatively rare cases. An example is in a multicast game, where the position of a moving object is continuously multicast. This positional stream does not require additional reliability because a new position superseding the old one will be sent before any retransmission could take place. However, when the moving object interacts with other objects or stops moving, then an explicit reliability mechanism is required to reliably send the interaction information or last position.

It is not just games that can be built in this manner - the NTE shared text editor[HC97] uses just such a mechanism with changes to a line of text. For every change the whole line is sent, and so long as the user keeps typing no explicit reliability mechanism is needed. The major advantage of replication is that it is not susceptible to spatially uncorrelated packet loss. With a traditional ACK or NACK based protocol, the probability of any particular packet being received by all the receivers in a large group can be very low. This leads to high retransmission rates. In contrast, replicated streams do not suffer as the size of the receiver group increases - different receivers lose different packets, but this does not increase network traffic.

4.4. Packet-level Forward Error Correction

Forward Error Correction (FEC) is a well known technique for protecting data against corruption. For reliable multicast it is most useful in the form of erasure codes.

The simplest form of packet-level FEC is to take a group of packets that is to be sent, and to XOR the packets together to form a newpacket which is also sent. If there were three original packets plus the XOR packet sent, then if a receiver is missing any one of the original data packets, but receives the XOR packet, then it can reproduce the missing original packet.

More general erasure codes exist [BKKKLZ95], [Ri97], [LMSSS97] that allow the generation of n encoding packets from k original data packets. In such cases, so long as at least k of the n encoding packets are received, then the k original data packets can be reproduced.

To apply FEC the sender groups data packets into rounds, and encoding packets are produced based on all the data packets in a round. A round may consist of all data packets in an entire application data unit in some cases, whereas in other cases it may consist of a group of data packets that make up only a small portion of an application data unit.

Using erasure codes to repair packet loss is a significant improvement over simple retransmission because the dependency on which packets have been lost is removed. Thus, the amount of repair traffic required to repair spatially uncorrelated packet loss is considerably lessened.

We can divide packet-level FEC schemes into two categories: proactive FEC and reactive FEC. The difference between the two is that for proactive FEC the sender decides a priori how many encoding packets to send for each round of data packets, whereas for reactive FEC the sender initially transmits only the original data packets for each round. Then, the sender uses feedback from the receivers to compute how many packets were lost by the receiver that experienced the most loss in each round, and then only that number of additional encoding packets are sent for that round. These encoding packets will then also serve to repair loss at the other receivers that are missing fewer packets. The receivers report via ACKs or NACKs how many packets are missing from each round. With NACKs, only the receiver missing the most packets need send a NACK for this round, so this is used to weight the random timers in the NACK calculation.

Proactive and reactive FEC can be combined, e.g., a certain amount of proactive FEC can be sent for each round and if there are receivers that experience more loss than can be overcome by this for some rounds then they can request and receive additional encoding packets for these rounds.

FEC is very effective at reducing the repair traffic for packet loss. However, it requires that the data to be sent to be grouped into rounds, which can add to end-to-end latency. For bulk-data applications this is typically not a problem, but this may be an issue for interactive applications where replication may be a better solution.

4.5. Layered FEC

An alternative use of packet level FEC is possible when data is spread across several multicast groups [RVC98], [BLMR98]. In such cases, the original k data packets are used to generate n encoding packets, where n is much larger than k. The n encoded packets are then striped across multiple multicast groups. When a receiver wishes to receive the original data it joins one or more of the multicast groups, and receives the encoding packets. Once it has received k different encoding packets, the receiver can then leave all the multicast groups and reconstruct the original data.

The primary importance of such a layering is that it allows different receivers to be able to receive the traffic at different rates according to the available capacity. Such schemes do not require any form of feedback from the receivers to the sender to ensure good throughput, and therefore the need for good throughput does not constrain the size of the receiver set. However, to perform adequate network congestion control using receiver joins and leaves in this manner may require coordination between members that are behind the same congested link from the sender. As described in the next section, [RVC98] suggests such a layered congestion control scheme.

5. Congestion Control Mechanisms

The basic delivery model of the Internet is best-effort service. No guarantees are given as to throughput, delay or packet loss. End-systems are expected to be adaptive, and to reduce their transmission rate to a level appropriate for the congestion state of the network. Although increasingly the Internet will start to support reserved bandwidth and differentiated service classes for specialist applications, unless an end-system knows explicitly that it has reserved bandwidth, it must still perform congestion control.

Broadly speaking, there are five classes of single-sender multicast congestion control solution:

- Sender-controlled, one group.

  A single multicast group is used for data distribution. Feedback from the group members is used to control the rate of this group. The goal is to transmit at a rate dictated by the slowest receiver.

- Sender-controlled, multiple groups.

  One initial multicast group is adaptively subdivided into multiple subgroups with subdivisions centered on congestion points in the network. Application-level relays buffer data from a group nearer the original sender, and retransmit it at a slower rate into a group further from the original sender. In this way, different receivers can receiver the data at different rates. Sender-based congestion control takes place between the members of a subgroup and their relay.

- Receiver-controlled, one group.

  A single multicast group is used for data distribution. The receivers determine if the sender is transmitting too rapidly for the current congestion state of the network, and they leave the group if this is the case.

- Receiver-controlled, layered organization.

  A layered approach for how to combine this scheme with a congestion control protocol that requires no receiver feedback is described in [RVC98]. The sender stripes data across multiple multicast groups simultaneously. Receivers join and leave these layered groups depending on their measurements of the congestion state of the network, so that the amount of data being received is always appropriate. However, this scheme relies on receivers to join and leave the different multicast groups in a coordinated fashion behind a bottleneck link, and it has not yet been completely confirmed that this approach will scale in practice to the Internet. As a result, more work on this congestion control mechanism would be beneficial.

- Router-based congestion control.

  It is possible to add additional mechanisms to multicast routers to assist in multicast congestion control. Such mechanisms could include:

  - Conditional joins (a multicast join that specifies a loss rate above which it is acceptable for the router to reject the join).

  - Router filtering of traffic that exceeds a reasonable rate. This may include mechanisms for filtering traffic at different points in the network at different rates depending on local congestion conditions [LVS99].

- o Fair queuing schemes combined with end-to-end adaptation.

  Router-based schemes generally require more state in network routers than has traditionally been acceptable for backbone routers. Thus, in the near-term, such schemes are only likely to be applicable for intranet solutions.

For reliable multicast protocols, it is important to consider congestion control at the same time as reliability is being considered. The same mechanisms that are used to provide reliability will sometimes be used to provide congestion control.

In the case of receiver-based congestion control, open-loop delivery using FEC is the likely choice for achieving good throughput for bulk- data transfer. This is because open-loop delivery requires no feedback from receivers, and thus it is a perfect match with a receiver-based congestion-control mechanism that operates without feedback from receivers.

6. Security Considerations

Generally speaking, security considerations have relatively little effect on constraining the design space for reliable multicast protocols. The primary issues constraining the design space are all related to receiver-set scaling. For authentication of the source and of data integrity, receiver-set scaling is not a significant issue. However, for data encryption, key distribution and particularly re-keying may be significantly affected by receiver-set scaling. Tree and graph based re-keying solutions[WHA98,WGL97] would appear to be appropriate solutions to these problems. It is not clear however that such re-keying solutions need to directly affect the design of the data distribution part of a reliable multicast protocol.

The primary question to consider for the security of reliable multicast protocols is the role of third-parties. If nodes other than the original source of the data are allowed to send or resend data packets, then the security model for the protocol must take this into account. In particular, it must be clear whether such third parties are trusted or untrusted. A requirement for trusted third parties can make protocols difficult to deploy on the Internet.

Untrusted third parties (such as receivers that retransmit the data) may be used so long as the data authentication mechanisms take this into account. Typically this means that the original sender digitally signs and timestamps the data, and that the third parties resend this signed timestamped payload unmodified.

Unlike unicast protocols, denial-of-service attacks on multicast transport state are easy if the protocol design does not take such attacks into account. This is because any receiver can join the session, and can then produce feedback that influences the progress of a session involving many other receivers. Hence protection against denial-of-service attacks on reliable multicast protocols must be carefully considered. A receiver that requests retransmission of every packet, or that refuses to acknowledge packets in an ACK-based protocol can potentially bring a reliable multicast session to a standstill. Senders must have appropriate policy to deal with such conditions, and if necessary, evict the receiver from the group. A single receiver masquerading as a large number of receivers may still be an issue under such circumstances with protocols that support NACK-like functionality. Providing unique "keys" to each NACKer when they first NACK using a unicast response might potentially prevent such attacks.

Denial-of-service attacks caused by traffic flooding are however somewhat easier to protect against than with unicast. Unwanted senders can simply be pruned from the distribution tree using the mechanisms implemented in IGMP v3[CDT99].

7. Conclusions

In this document we present an overview of the design space for reliable multicast within the context of one-to-many bulk-data transfer. Other flavors of multicast application are not considered in this document, and hence the overview given should not be considered inclusive of the design space for protocols that fall outside the context of one-to-many bulk-data transfer. During the course of this overview, we have reaffirmed the notion that the process of reliable multicast protocol design is affected by a number of factors that render the generation of a "one size fits all solution" moot. These factors are then described to show how an application’s needs serve to constrain the set of available techniques that may be used to create a reliable multicast protocol. We examined a number of basic techniques and to show how well they can meet the needs of certain types of applications.

This document is intended to provide guidance to the IETF community regarding the standardization of reliable multicast protocols for bulk-data transfer. Given the degree to which application requirements constrain reliable multicast solutions, and the diverse set of applications that need to be supported, it should be clear that any standardization work should take great pains to be future-proof. This would seem to imply not standardizing complete reliable multicast transport protocols in one pass, but rather examining the degree to which such protocols are separable into functional building

blocks, and standardizing these blocks separately to the maximum degree that makes sense. Such an approach allows for protocol evolution, and allows applications with new constraints to be supported with maximal reuse of existing and tested mechanisms.

8. Acknowledgments

This document represents an overview of the reliable multicast design space. The ideas presented are not those of the authors, but are collected from the varied presentations and discussions in the IRTF Reliable Multicast Research Group. Although they are too numerous to list here, we thank everyone who has participated in these discussions for their contributions.

9. Authors' Addresses

Mark Handley
ATT Center for Internet Research at ICSI,
International Computer Science Institute,
1947 Center Street, Suite 600,
Berkeley, CA 94704, USA

EMail: mjh@aciri.org

Sally Floyd
ATT Center for Internet Research at ICSI,
International Computer Science Institute,
1947 Center Street, Suite 600,
Berkeley, CA 94704, USA

EMail: floyd@aciri.org

Brian Whetten
Talarian Corporation,
333 Distel Circle,
Los Altos, CA 94022, USA

EMail: whetten@talarian.com

Roger Kermode
Motorola Australian Research Centre
Level 3, 12 Lord St,
Botany NSW 2019,
Australia

EMail: Roger.Kermode@motorola.com

Lorenzo Vicisano
Cisco Systems,
170 West Tasman Dr.
San Jose, CA 95134, USA

EMail: lorenzo@cisco.com

Michael Luby
Digital Fountain, Inc.
600 Alabama Street
San Francisco, CA 94110

EMail: luby@digitalfountain.com

10. References

[BC94] K. Birman, T. Clark. "Performance of the Isis Distributed Computing Toolkit." Technical Report TR-94-1432, Dept. of Computer Science, Cornell University.

[BKKKLZ95] J. Bloemer, M. Kalfane, M. Karpinski, R. Karp, M. Luby, D. Zuckerman, "An XOR-based Erasure Resilient Coding Scheme", ICSI Technical Report No. TR-95-048, August 1995.

[BLMR98] J. Byers, M. Luby, M. Mitzenmacher, A. Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data", Proc ACM SIGCOMM 98.

[CDT99] Cain, B., Deering, S., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", Work in Progress.

[FLST98] Farinacci, D., Lin, S., Speakman, T. and A. Tweedly, "PGM reliable transport protocol specification", Work in Progress.

[FJM95] S. Floyd, V. Jacobson, S. McCanne, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", Proc ACM SIGCOMM 95, Aug 1995 pp. 342-356.

[Ha99] Handley, M., "Multicast address allocation protocol (AAP)", Work in Progress.

[HC97] M. Handley and J. Crowcroft, "Network text editor (NTE) a scalable shared text editor for MBone," ACM Computer Communication Review, vol. 27, pp. 197-208, Oct. 1997. ACM SIGCOMM'97, Sept. 1997.

[KCW98] Kadansky, M., Chiu, D. and J. Wesley, "Tree-based reliable multicast (TRAM)", Work in Progress.

[LMSSS97] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, V. Stemann, "Practical Loss-Resilient Codes", Proc ACM Symposium on Theory of Computing, 1997.

[MWB+98] Montgomery, T., Whetten, B., Basavaiah, M., Paul, S., Rastogi, N., Conlan, J. and T. Yeh, "THE RMTP-II PROTOCOL", Work in Progress.

[PPV98] C. Papadopoulos, G. Parulkar, and G. Varghese, "An error control scheme for large-scale multicast applications," in Proceedings of the Conference on Computer Communications (IEEE Infocom), (San Francisco, California), p. 1188, March/April 1998.

[Ri97] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," ACM Computer Communication Review, vol. 27, pp. 24-36, Apr. 1997.

[RV97] L. Rizzo, L. Vicisano, "A Reliable Multicast data Distribution Protocol based on software FEC techniques", Proc. of The Fourth IEEE Workshop on the Architecture and Implementation of High Performance Communication Systems (HPCS'97), Sani Beach, Chalkidiki, Greece June 23-25, 1997.

[RVC98] L. Rizzo, L. Vicisano, J. Crowcroft, "The RLC multicast congestion control algorithm", submitted to IEEE Network - special issue multicast.

[RMWT98] Robertson, K., Miller, K., White, M. and A. Tweedly, "StarBurst multicast file transfer protocol (MFTP) specification", Work in Progress.

[WHA98] Wallner, D., Hardler, E. and R. Agee, "Key Management for Multicast: Issues and Architectures", RFC 2627, June 1999.

[WKM94] Brian Whetten, Simon Kaplan, and Todd Montgomery, "A high performance totally ordered multicast protocol," research memorandum, Aug. 1994.

[WGL97] C.K. Wong, M. Gouda, S. Lam, "Secure Group Communications Using Key Graphs," Technical Report TR 97-23, Department of Computer Sciences, The University of Texas at Austin, July 1997.

11. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.