RMT Working Group
Internet Engineering Task Force
INTERNET-DRAFT
draft-ietf-rmt-buildingblocks-02.txt
**10 March 2000**
Expires 10 September 2000

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

# Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer

<draft-ietf-rmt-buildingblocks-02.txt>

## Status of this Memo

## Abstract

This document describes a framework for the standardization of bulk-data reliable multicast transport. It builds upon the experience gained during the deployment of several classes of contemporary reliable multicast transport, and attempts to pull out the commonalities between these classes of protocols into a number of building blocks. To that end, this document recommends that certain components that are common to multiple protocol classes be standardized as "building blocks." The remaining parts of the protocols, consisting of highly protocol specific, tightly intertwined functions, shall be designated as "protocol cores." Thus, each protocol can then be constructed by merging a "protocol core" with a number of "building blocks" which can be re-used across multiple protocols.

Copyright Notice

## 1. Introduction

RFC2357 lays out the requirements for reliable multicast protocols that are to be considered for standardization by the IETF. They include:

- Congestion Control. The protocol must be safe to deploy in the widespread Internet. Specifically, it must adhere to three mandates: a) it must achieve good throughput (i.e. it must not consistently overload links with excess data or repair traffic), b) it must achieve good link utilization, and c) it must not starve competing flows.

- Scalability. The protocol should be able to work under a variety of conditions that include multiple network topologies, link speeds, and the receiver set size. It is more important to have a good understanding of how and when a protocol breaks than when it works.

- Security. The protocol must be analyzed to show what is necessary to allow it to cope with security and privacy issues. This includes understanding the role of the protocol in data confidentiality and sender authentication, as well as how the protocol will provide defenses against denial of service attacks.

These requirements are primarily directed towards making sure that any standards will be safe for widespread Internet deployment. The advancing maturity of current work on reliable multicast congestion control (RMCC) [HFW99] in the IRTF Reliable Multicast Research Group (RMRG) has been one of the events that has allowed the IETF to charter the RMT working group. RMCC only addresses a subset of the design space for reliable multicast. Fortuitously, the requirements it addresses are also the most pressing application and market requirements.

A protocol's ability to meet the requirements of congestion control, scalability, and security is affected by a number of secondary requirements that are described in a separate document [HWKFV99]. In summary, these are:

- Ordering Guarantees. A protocol must offer at least one of either source ordered or unordered delivery guarantees. Support for total ordering across multiple senders is not recommended, as it makes it more difficult to scale the protocol, and can more easily be implemented at a higher level.

- Receiver Scalability. A protocol should be able to support a "large" number of simultaneous receivers per transport group. A typical receiver set could be on the order of at least 1,000 - 10,000 simultaneous receivers per group, or could even eventually scale up to millions of receivers in the large Internet.

- Real-Time Feedback. Some versions of RMCC may require soft real-time feedback, so a protocol may provide some means for this information to be measured and returned to the sender. While this does not require that a protocol deliver data in soft real-time, it is an important application requirement that can be provided easily given real-time feedback.

- Delivery Guarantees. 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 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 must provide 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 may optionally provide delivery confirmation, 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. Packet level confirmation may also aid in application data unit confirmation.

- Network Topologies. A protocol must not break the network when deployed in the full Internet. However, we recognize that intranets will be where the first wave of deployments happen, and so are also very important to support. Thus, support for satellite networks (including those with terrestrial return paths or no return paths at all) is encouraged, but not required.

- Group Membership. The group membership algorithms must be scalable. Membership can be anonymous (where the sender does not know the list of receivers), or fully distributed (where the sender receives a count of the number of receivers, and optionally a list of failures).

- Example Applications. Some of the applications that a RM protocol could be designed to support include multimedia broadcasts, real time financial market data distribution, multicast file transfer, and server replication.

In the rest of this document the following terms will be used with a specific connotation: "protocol family", "protocol component", "building block", "protocol core", and "protocol instantiation". A "protocol family" is a broad class of RM protocols which share a common characteristic. In our classification, this characteristic is the mechanism used to achieve reliability. A "protocol component" is a logical part of the protocol that addresses a particular functionality. A "building block" is a constituent of a protocol that implements one, more than one or a part of a component. A "protocol core" is the set of functionality required for the instantiation of a complete protocol, that is not specified by any building block. Finally a "protocol instantiation" is an actual RM protocol defined in term of building blocks and a protocol core.

### 1.1. Protocol Families

The design-space document [HWKFV99] also provides a taxonomy of the most popular approaches that have been proposed over the last ten years. After congestion control, the primary challenge has been that of meeting the requirement for ensuring good throughput in a way that scales to a large number of receivers. For protocols that include a back-channel for recovery of lost packets, the ability to take advantage of support of elements in the network has been found to be very beneficial for supporting good throughput for a large numbers of receivers. Other protocols have found it very beneficial to transmit coded data to achieve good throughput for large numbers of receivers.

This taxonomy breaks proposed protocols into four families. Some protocols in the family provide packet level delivery confirmation that may be useful to the transport level. All protocols in all families can be supplemented with higher level protocols that provide delivery confirmation of application data units.

1 **NACK only.** Protocols such as SRM [FJM95] and MDP2 [MA99] attempt to limit traffic by only using NACKs for requesting packet retransmission. They do not require network infrastructure.

2 **Tree based ACK.** Protocols such as RMTP [LP96, PSLM97], RMTP-II [WBPM98] and TRAM [KCW98], use positive acknowledgments (ACKs). ACK based protocols reduce the need for supplementary protocols that provide delivery confirmation, as the ACKS can be used for this purpose. In order to avoid ACK implosion in scaled up deployments, the protocol can use servers placed in the network.

3 **Open-Loop delivery.** These protocols (examples include [RV97] and [BLMR98]) use sender-based Forward Error Correction (FEC) methods with no feedback from receivers or the network to ensure good throughput.

4 **Router assist.** Like SRM, protocols such as PGM [FLST98] and [LG97] also use negative acknowledgments for packet recovery. These protocols take advantage of new router software to do constrained negative acknowledgments and retransmissions. Router assist protocols can also provide other functionality more efficiently than end to end protocols. For example, [LVS99] shows how router assist can provide fine grained congestion control to open-loop delivery protocols. Router assist protocols can be designed to complement all protocol families described above.

Note that the distinction in protocol families in not necessarily precise and mutually exclusive. Actual protocols may use a combination of the mechanisms belonging to different classes. For example, hybrid NACK/ACK based protocols (such as [WBPM98]) are possible. Other examples are protocols belonging to class 1 through 3 that take advantage of router support.

## 2. Building Blocks Rationale

As specified in RFC2357 [MRBP98], no single reliable multicast protocol will likely meet the needs of all applications. Therefore, the IETF expects to standardize a number of protocols that are tailored to application and network specific needs. This document concentrates on the requirements for "one-to-many bulk-data transfer", but in the future, additional protocols and building-blocks are expected that will address the needs of other types of applications, including "many-to-many" applications. Note that bulk-data transfer does not refer to the timeliness of the data, rather it states that there is a large amount of data to be transferred in a session. The scope and approach taken for the development of protocols for these additional scenarios will depend upon large part on the success of the "building-block" approach put forward in this document.

## 2.1. Building Blocks Advantages

Building a large piece of software out of smaller modular components is a well understood technique of software engineering. Some of the advantages that can come from this include:

- Specification Reuse. Modules can be used in multiple protocols, which reduces the amount of development time required.

- Reduced Complexity. To the extent that each module can be easily defined with a simple API, breaking a large protocol in to smaller pieces typically reduces the total complexity of the system.

- Reduced Verification and Debugging Time. Reduced complexity results in reduced time to debug the modules. It is also usually faster to verify a set of smaller modules than a single larger module.

- Easier Future Upgrades. There is still ongoing research in reliable multicast, and we expect the state of the art to continue to evolve. Building protocols with smaller modules allows them to be more easily upgraded to reflect future research.

- Common Diagnostics. To the extent that multiple protocols share common packet headers, packet analyzers and other diagnostic tools can be built which work with multiple protocols.

- Reduces Effort for New Protocols. As new application requirements drive the need for new standards, some existing modules may be reused in these protocols.

- Parallelism of Development. If the APIs are defined clearly, the development of each module can proceed in parallel.

## 2.2. Building Block Risks

Like most software specification, this technique of breaking down a protocol in to smaller components also brings tradeoffs. After a certain point, the disadvantages outweigh the advantages, and it is not worthwhile to further subdivide a problem. These risks include:

- Delaying Development. Defining the API for how each two modules inter-operate takes time and effort. As the number of modules increases, the number of APIs can increase at more than a linear rate. The more tightly coupled and complex a component is, the more difficult it is to define a simple API, and the less opportunity there is for reuse. In particular, the problem of how to build and standardize fine grained building blocks for a transport protocol is a difficult one, and in some cases requires fundamental research.

- Increased Complexity. If there are too many modules, the total complexity of the system actually increases, due to the preponderance of interfaces between modules.

- Reduced Performance. Each extra API adds some level of processing overhead. If an API is inserted in to the "common case" of packet processing, this risks degrading total protocol performance.

- Abandoning Prior Work. The development of robust transport protocols is a long and time intensive process, which is heavily dependent on feedback from real deployments. A great deal of work has been done over the past five years on components of protocols such as RMTP-II, SRM, and PGM. Attempting to dramatically re-engineer these components risks losing the benefit of this prior work.

## 2.3. Building Block Requirements

Given these tradeoffs, we propose that a building block must meet the following requirements:

- Wide Applicability. In order to have confidence that the component can be reused, it should apply across multiple protocol families and allow for the component's evolution.

- Simplicity. In order to have confidence that the specification of the component APIs will not dramatically slow down the standards process, APIs must be simple and straight forward to define. No new fundamental research should be done in defining these APIs.

- o Performance. To the extent possible, the building blocks should attempt to avoid breaking up the "fast track", or common case packet processing.

## 3. Protocol Components

This section proposes a functional decomposition of RM bulk-data protocols from the perspective of the functional components provided to an application by a transport protocol. It also covers some components that while not necessarily part of the transport protocol, are directly impacted by the specific requirements of a reliable multicast transport. The next section then specifies recommended building blocks that can implement these components.

Although this list tries to cover all the most common transport-related needs of one-to-many bulk-data transfer applications, new application requirements may arise during the process of standardization, hence this list must not be interpreted as a statement of what the transport layer should provide and what it should not. Nevertheless, it must be pointed out that some functional components have been deliberately omitted since they have been deemed irrelevant to the type of application considered (i.e. one-to-many bulk-data transfer). Among these are advanced message ordering (i.e. those which cannot be implemented through a simple sequence number) and atomic delivery.

It is also worth mentioning that some of the functional components listed below may be required by other functional components and not directly by the application (e.g. membership knowledge is usually required to implement ACK-based reliability).

The following list covers various transport functional components and splits them in sub-components.

```
  o Data Reliability (ensuring good throughput)    |
                        | - Loss Detection/Notification
                        | - Loss Recovery
                        | - Loss Protection



  o Congestion Control  |
                        | - Congestion Feedback
                        | - Rate Regulation
                        | - Receiver Controls

  o Security

  o Group membership    |
                        | - Membership Notification
                        | - Membership Management

  o Session Management  |
                        | - Group Membership Tracking
                        | - Session Advertisement
                        | - Session Start/Stop
                        | - Session Configuration/Monitoring

  o Tree Configuration
```

Note that not all components are required by all protocols, depending upon the fully defined service that is being provided by the protocol. In particular, some minimal service models do not require many of these functions, including loss notification, loss recovery, and group membership.

### 3.1. Sub-Components Definition

Loss Detection/Notification. This includes how missing packets are detected during transmission and how knowledge of these events are propagated to one or more agents which are designated to recover from the transmission error. This task raises major scalability issues and can lead to feedback implosion and poor throughput if not properly handled. Mechanisms based on TRACKs (tree-based positive acknowledgements) or NACKs (negative acknowledgements) are the most widely used to perform this function. Mechanisms based on a combination of TRACKs and NACKs are also possible.

Loss Recovery. This function responds to loss notification events through the transmission of additional packets, either in the form of copies of those packets lost or in the form of FEC packets. The manner in which this function is implemented can significantly affect the scalability of a protocol.

Loss Protection. This function attempts to mask packet-losses so that they don't become Loss Notification events. This function can be realized through the pro-active transmission of FEC packets. Each FEC packet is created from an entire application data unit [LMSSS97] or a portion of an application data unit [RV97], [BKKKLZ95], a fact that allows a receiver to recover from some packet loss without further retransmissions. The number of losses that can be recovered from without requiring retransmission depends on the amount of FEC packets sent in the first place. Loss protection can also be pushed to the extreme when good throughput is achieved without any Loss Detection/Notification and Loss Recovery functionality, as in the open loop family of protocols defined above.

Congestion Feedback. For sender driven congestion control protocols, the receiver must provide some type of feedback on congestion to the sender. This typically involves loss rate and round trip time measurements.

Rate Regulation. Given the congestion feedback, the sender then must adjust its rate in a way that is fair to the network. One proposal that defines this notion of fairness and other congestion control requirements is [Whetten99].

Receiver Controls. In order to avoid allowing a receiver that has an extremely slow connection to the sender to stop all progress within single rate schemes, a congestion control algorithm will often reuire receivers to leave groups. For multiple rate approaches, receivers of all connection speeds can have data delivered to them according to the rate of their connection without slowing down other receivers.

Security. Security for reliable multicast contains a number of complex and tricky issues that stem in large part from the IP multicast service model. In this service model, hosts do not send traffic to another host, but instead elect to receive traffic from a multicast group. This means that any host may join a group and receive its traffic. Conversely, hosts may also leave a group at any time. Therefore, the protocol must address how it impacts the following security issues:

o Sender Authentication (since any host can send to a group),

o Data Encryption (since any host can join a group)

o Transport Protection (denial of service attacks, through corruption of transport state, or requests for unauthorized resources)

- Group Key Management (since hosts may join and leave a group at any time) [WHA98]

In particular, a transport protocol needs to pay particular attention to how it protects itself from denial of service attacks, through mechanisms such as lightweight authentication of control packets. [HW99]

Sender Authentication, Data Encryption, and Group Key Management. While these functions are not typically part of the transport layer per se, a protocol needs to understand what ramifications it has on data security, and may need to have special interfaces to the security layer in order to accommodate these ramifications.

Transport Protection. The primary security task for a transport layer is that of protecting the transport layer itself from attack. The most important function for this is typically lightweight authentication of control packets in order to prevent corruption of state and other denial of service attacks.

Membership Notification. This is the function through which the data source--or upper level agent in a possible hierarchical organization--learns about the identity and/or number of receivers or lower level agents. To be scaleable, this typically will not provide total knowledge of the identity of each receiver.

Membership Management. This implements the mechanisms for members to join and leave the group, to accept/refuse new members, or to terminate the membership of existing members.

Group Membership Tracking. As an optional feature, a protocol may interface with a component which tracks the identity of each receiver in a large group. If so, this feature will typically be implemented out of band, and may be implemented by an upper level protocol. This may be useful for services that require tracking of usage of the system, billing, and usage reports.

Session Advertisement. This publishes the session name/contents and the parameters needed for its reception. This function is usually performed by an upper layer protocol (e.g. [HPW99] and [HJ98]).

Session Start/Stop. These functions determine the start/stop time of sender and/or receivers. In many cases this is implicit or performed by an upper level application or protocol. In some protocols, however, this is a task best performed by the transport layer due to scalability requirements.

Session Configuration/Monitoring. Due to the potentially far reaching scope of a multicast session, it is particularly important for a protocol to include tools for configuring and monitoring the protocol's operation.

Tree Configuration. For protocols which include hierarchical elements (such as PGM and RMTP-II), it is important to configure these elements in a way that has approximate congruence with the multicast routing topology. While tree configuration could be included as part of the session configuration tools, it is clearly better if this configuration can be made automatic.

## 4. Building Block Recommendations

The families of protocols introduced in section 1.1 generally use different mechanisms to implement the protocol functional components described in section 3. This section tries to group these mechanisms in macro components that define protocol building blocks.

A building block is defined as

> "a logical protocol component that results in explicit APIs for use by other building blocks or by the protocol client."

Building blocks are generally specified in terms of the set of algorithms and packet formats that implement protocol functional components. A building block may also have API's through which it communicates to applications and/or other building blocks. Most building blocks should also have a management API, through which it communicates to SNMP and/or other management protocols.

In the following section we will list a number of building blocks which, at this stage, seem to cover most of the functional components needed to implement the protocol families presented in section 1.1. Nevertheless this list represents the "best current guess", and as such it is not meant to be exhaustive. The actual building block decomposition, i.e. the division of functional components into building blocks, may also have to be revised in the future.

### 4.1. NACK-based Reliability

This building block defines NACK-based loss detection/notification and recovery. The major issues it addresses are implosion prevention (suppression) and NACK semantics (i.e. how packets to be retransmitted should be specified, both in the case of selective and FEC loss repair).

Suppression mechanisms to be considered are:

- o Multicast NACKs
- o Unicast NACKs and Multicast confirmation

These suppression mechanisms primarily need to both minimize delay while also minimizing redundant messages. They may also need to have special weighting to work with Congestion Feedback.

### 4.2. FEC Repair

This building block is concerned with packet level FEC repair. It specifies the FEC codec selection and the FEC packet naming (indexing) for both on-demand FEC and pro-active FEC.

### 4.3. Congestion Control

There will likely be multiple versions of this building block, corresponding to different design policies in addressing congestion control. Two main approaches are considered for the time being: a source-based rate regulation with a single rate provided to all the receivers in the session, and a multiple rate receiver-driven approach with different receivers receiving at different rates in the same session. The multiple rate approach may use multiple layers of multicast traffic [VRC98] or router filtering of a single layer [LVS99].

Both approaches are still in the phase of study, however the first seems to be mature enough [HFW99] to allow the standardization process to begin.

At the time of writing this document, a third class of congestion control algorithm based on router support is beginning to emerge in the IRTF RMRG [LVS99]. This work may lead to the future standardization of one or more additional building blocks for congestion control.

### 4.4. Generic Router Support

The task of designing RM protocols can be made much easier by the presence of some specific support in routers. In some application-specific cases, the increased benefits afforded by the addition of special router support can justify the resulting additional complexity and expense [FLST98].

Functional components which can take advantage of router support include feedback aggregation/suppression (both for loss notification and congestion control) and constrained retransmission of repair packets. Another component that can take advantage of router support is intentional packet filtering to provide different rates of delivery of packets to different receivers from the same multicast packet stream. This could be most advantageous when combined with open-loop delivery protocols [LVS99].

The process of designing and deploying these mechanisms inside routers can be much slower than the one required for end-host protocol mechanisms. Therefore, it would be highly advantageous to define these mechanisms in a generic way that multiple protocols can use if it is available, but do not necessarily need to depend on.

This component has two halves, a signaling protocol and actual router algorithms. The signaling protocol allows the transport protocol to request from the router the functions that it wishes to perform, and the router algorithms actually perform these functions. It is more urgent to define the signaling protocol, since it will likely impact the common case protocol headers.

An important component of the signaling protocol is some level of commonality between the packet headers of multiple protocols, which allows the router to recognize and interpret the headers.

### 4.5. Tree Configuration

It has been shown that the scalability of RM protocols can be greatly enhanced by the insertion of some kind of retransmission or feedback aggregation agents between the source and receivers. These agents are then used to form a tree with the source at (or near) the root, the receivers at the leaves of the tree, and the aggregation/local repair nodes in the middle. The internal nodes can either be dedicated software for this task, or they may be receivers that are performing dual duty.

The effectiveness of these agents to assist in the delivery of data is highly dependent upon how well the logical tree they use to communicate matches the underlying routing topology. The purpose of this building block would be to construct and manage the logical tree connecting the agents. Ideally, this building block would perform these functions in a manner that adapts to changes in session membership, routing topology, and network availability.

### 4.6. Security

At the time of writing, the security issues are the subject of research within the IRTF Secure Multicast Group (SMuG). Solutions for these requirements will be standardized within the IETF when ready.

### 4.7. Common Headers

As pointed out in the generic router support section, it is important to have some level of commonality across packet headers. It may also be useful to have common data header formats for other reasons. This building block would consist of recommendations on fields in their packet headers that protocols should make common across themselves.

### 4.8. Protocol Cores

The above building blocks consist of the functional components listed in section 3 that appear to meet the requirements for being implemented as building blocks presented in section 2.

The other functions from section 3, which are not covered above, should be implemented as part of "protocol cores", specific to each protocol standardized.

## 5. Conclusions

In this document, we briefly described a number of building blocks that may be used for the generation of reliable multicast protocols to be used in the application space of one-to-many reliable bulk-data transfer. The list of building blocks presented was derived from considering the functions that a protocol in this space must perform and how these functions should be grouped. This list is not intended to be all-inclusive but instead to act as guide as to which building blocks are considered during the standardization process within the Reliable Multicast Transport WG.

## 6. Acknowledgements

This document represents an overview of a number of building blocks for one to many bulk data transfer that may be ready for standardization within the RMT working group.  The ideas presented are not those of the authors, rather they are a summarization of many years of research into multicast transport combined with 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.

## 7. References

[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.

[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.

[FLST98]
D. Farinacci, S. Lin, T. Speakman, and A. Tweedly, "PGM reliable transport protocol specification," Internet Draft, Internet Engineering Task Force, Aug. 1998. Work in progress.

[HFW99]
M. Handley, S. Floyd, B. Whetten, "Strawman Specification for TCP Friendly (Reliable) Multicast Congestion Control (TFMCC)," work in progress.

[HJ98]
M. Handley, V. Jacobson, "SDP: Session Description Protocol," RFC2327, April 1998.

[HPW99]
M. Handley, C. Perkins, E. Whelan, "Session Announcement Protocol," Internet Draft, work in progress, June 1999.

[HW99]
T. Hardjorno, B. Whetten, "Security Requirements for RMTP-II," Internet Draft, work in progress, June 1999.

[HWKFV99]
M. Handley, B.Whetten, R. Kermode, S.Floyd, and L. Vicisano, "The Reliable Multicast Design Space for Bulk Data Transfer," Internet Draft, Internet Engineering Task Force, June 1999.

[KCW98]
M. Kadansky, D. Chiu, and J. Wesley, `"Tree-based reliable multicast (TRAM),'" Internet Draft, Internet Engineering Task Force, Nov. 1998. Work in progress.

[Kermode98]
R. Kermode, "Scoped Hybrid Automatic Repeat Request with Forward Error Correction," Proc ACM SIGCOMM 98, Sept 1998.

[LDW98]
M. Lucas, B. Dempsey, A. Weaver, "MESH: Distributed Error Recovery for Multimedia Streams in Wide-Area Multicast Networks"

[LESZ97]
C-G. Liu, D. Estrin, S. Shenkar, L. Zhang, "Local Error Recovery in SRM: Comparison of Two Approaches," USC Technical Report 97- 648, Jan 1997.

[LG97]
B.N. Levine, J.J. Garcua-Luna-Aceves, "Improving Internet Multicast Routing with Routing Labels," IEEE International Conference on Network Protocols (ICNP-97), Oct 28-31, 1997, p.241-250.

[LP96]
K. Lin and S. Paul. "RMTP: A Reliable Multicast Transport Protocol," IEEE INFOCOMM 1996, March 1996, pp. 1414-1424.

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

[LVS99]
M. Luby, L. Vicisano, T. Speakman. ``Heterogeneous multicast congestion control based on router packet filtering, RMT working group, June 1999, Pisa, Italy.

[MA99]
J. Macker, B. Adamson. "Multicast Dissemination Protocol version 2 (MDPv2)," work in progress, http://manimac.itd.nrl.navy.mil/MDP.

[MRBP98]
A. Mankin, A. Romanow, S.Brander, V.Paxson, "IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols," RFC2357, June 1998.

[OXB99]
O. Ozkasap, Z. Xiao, K. Birman. "Scalability of Two Reliable Multicast Protocols," Work in progress, May 1999.

[PSLB97]
"Reliable Multicast Transport Protocol (RMTP)," S. Paul, K. K. Sabnani, J. C. Lin, and S. Bhattacharyya, IEEE Journal on Selected Areas in Communications, Vol. 15, No. 3, April 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.

[VRC98]
L. Vicisano, L. Rizzo, J. Crowcroft, "TCP-Like Congestion Control for Layered Multicast Data Transfer", Proc. of IEEE Infocom'98, March 1998.

[WBPM98]
B. Whetten, M. Basavaiah, S. Paul, T. Montgomery, N. Rastogi, J. Conlan, and T. Yeh, "THE RMTP-II PROTOCOL," Internet Draft, Internet Engineering Task Force, Apr. 1998. Work in progress

[WHA98]
D. Wallner, E. Hardler, R. Agee, `"Key Management for Multicast: Issues and Architectures," Internet Draft, work in progress, Sept 1998.

[Whetten99]
B. Whetten, "A Proposal for Reliable Multicast Congestion Control Requirements," work in progress. http://www.talarian.com/rmtp-ii/overview.htm

## 8. Authors' Addresses

Brian Whetten
Talarian Corporation,
333 **Distel Circle,**
Los Altos, CA 94022, USA
whetten@talarian.com

Lorenzo Vicisano
Cisco Systems,
170 **West Tasman Dr.**
San Jose, CA 95134, USA
lorenzo@cisco.com

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

Mark Handley, Sally Floyd
ATT Center for Internet Research at ICSI,
International Computer Science Institute,
1947 **Center Street, Suite 600,**
Berkeley, CA 94704, USA
mjh@aciri.org, floyd@aciri.org

Michael Luby
Digital Fountain, Inc. and ICSI
luby@dfountain.com, luby@icsi.berkeley.edu

## 9. Full Copyright Statement

Table of Contents

1 Introduction .................................................... 2
1.1 Protocol Families ............................................. 4
2 Building Blocks Rationale ....................................... 5
2.1 Building Blocks Advantages .................................... 6
2.2 Building Block Risks .......................................... 7
2.3 Building Block Requirements ................................... 7
3 Protocol Components ............................................. 8
3.1 Sub-Components Definition ..................................... 9
4 Building Block Recommendations .................................. 12
4.1 NACK-based Reliability ........................................ 12
4.2 FEC Repair .................................................... 13
4.3 Congestion Control ............................................ 13
4.4 Generic Router Support ........................................ 13
4.5 Tree Configuration ............................................ 14

4.6 Security .................................................... 15
4.7 Common Headers .............................................. 15
4.8 Protocol Cores .............................................. 15
5 Conclusions ................................................... 15
6 Acknowledgements .............................................. 16
7 References .................................................... 16
8 Authors' Addresses ............................................ 19
9 Full Copyright Statement ...................................... 19