

# THE MULTICAST DISSEMINATION PROTOCOL (MDP) TOOLKIT

Joseph P. Macker  
Information Technology Division  
Naval Research Laboratory

R. Brian Adamson  
Newlink Global Engineering

## ABSTRACT

*The Multicast Dissemination Protocol (MDP) provides reliable multicast file and data delivery on top of the generic UDP/IP multicast transport. Early work on MDP was deployed across the global Internet Multicast Backbone (Mbone) as part of the publicly available Image Multicaster (IMM) application. This paper describes more recent work on MDP resulting in the MDP version 2 (MDPv2) toolkit. This recent effort has significantly modified and generalized the MDP protocol and the associated software interface. Enhancements made to the protocol are suitable for a wide range of network environments. Additionally, integrated erasure-based repairing improves reliable multicast efficiency and robustness. We briefly discuss the protocol design, general performance characteristics, and ongoing MDP work including: initial rate-based congestion control design and results, network simulation and modeling, and asymmetric satellite operation.*

## BACKGROUND

A wide variety of applications can benefit from the basic Internet Protocol (IP) suite multicast model [Deering89] that provides network layer multipoint delivery of group-addressed packets. The IP multicast model uses generic IP datagrams as its raw service and does not provide inherent reliability of data delivery. IP multicast applications requiring guaranteed delivery need reliable multicast transport mechanisms. There exists a large body of past and present ongoing work in network transport methods for reliable multicast. Some past Naval Research Laboratory (NRL) work in reliable multicasting included co-development of version 1 of the Multicast Dissemination Protocol (MDP). This earlier protocol was framework was part of the freely available Image Multicaster (IMM) software deployed and tested over the Internet Multicast Backbone (Mbone) since 1993 [Macker96a]. This document describes the Multicast Dissemination Protocol version 2 (MDPv2), research enhancements, and the associated software toolkit. The authors intend MDPv2 to replace previous MDP work and this paper will herein use the term MDP in reference to MDPv2.

## OVERVIEW

A primary design goal of MDP was to develop a reliable, scalable, and efficient multicast transport protocol for use under a variety of heterogeneous network architectures. A secondary design goal was the seamless integration of novel erasure-based parity repairing techniques with selective multicast packet retransmission. Targeted operational factors that were key considerations throughout the protocol design process included:

- Use in heterogeneous, WAN infrastructures
- Use in mobile and wireless network conditions
- Operation in asymmetric delivery scenarios

- Support for small to large group sizes
- Support for group dynamics

MDP mainly uses selective negative acknowledgement (NACK) of missing data by receivers (clients) to enforce reliability. The NACK approach used is similar to that of earlier MDP work with the intended purpose of maintaining multicast protocol efficiency and scalability. Previous work has shown that reliable, selective NACK-based multicast protocols are more scalable than those based on positive acknowledgement (ACK) of received data [Pingali93]. Furthermore, it is possible to suppress transmission of redundant NACK transmissions among a group of receivers using probabilistic techniques as originally designed into MDP or as done in other reliable multicast protocol work (e.g., SRM [Floyd95]).

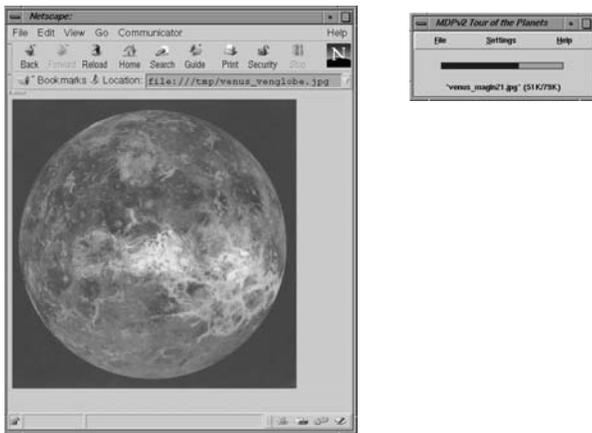
In conjunction with selective NACK, the current MDP protocol uses a parity-based repairing mechanism based upon packet-level forward error correction coding concepts [Macker97]. The use of parity-based erasure repairing for multicast selective retransmission offers significant performance advantages (such as in error-prone wireless environments or across scaled WAN sessions). In MDP, encoded parity repair packets are normally sent only in response to repair requests by receivers so the algorithm adds no additional protocol overhead above pure selective retransmission methods. However, the protocol may be optionally configured to transmit “proactive” repair packets as part of the original data transmission block. Protocol performance features of this nature and the resultant effect on delay and throughput performance were briefly studied and investigated in [Gossink98].

The MDP design supports distributed multicast session participation with little coordination among senders and receivers. The protocol allows senders and receivers to dynamically join and leave multicast sessions with a minimal amount of required overhead for control information and timing synchronization among participants. As a result of this requirement, MDP protocol message headers contain some common information allowing receivers to easily synchronize to sources on a dynamic, *ad hoc* basis. In its common mode of operation, the MDP protocol uses multicast delivery mechanisms for both source and receiver transmissions, but the protocol permits optional unicast-based client feedback to MDP data sources. Optional unicast feedback may be suitable for use in asymmetric networks or in networks where only unidirectional multicast routing/delivery service exists.

### *Application Issues*

Reliable multicast transport protocol design is a rich and growing area of technology development and a variety of solutions and performance issues have been widely discussed elsewhere in the literature [Levine96, Macker96b]. In general, there is no “one size fits all” solution to reliable multicasting across the complete set of

application semantics and networking architectures foreseen. MDP has been initially designed as a reliable multicast bulk transfer protocol for use in heterogeneous network environments, but contains enhancements to support the reliable transport of small, self-contained data messages as well. The generic MDP protocol toolkit assumes no underlying structure in the network architecture and typically works in a completely end-to-end fashion. This does not preclude the adaptation of the protocol to more structured operation (e.g., reliable multicast tree hierarchy, etc), but more direct application areas include mobile wireless, asymmetric satellite, and dynamic, heterogeneous WAN conditions. MDP has been designed to be tolerant of poor timing estimations that might occur in dynamic conditions (e.g., mobile and wireless networks). In addition, the protocol design is robust under severe packet loss, reordering, and large queuing or transmission delays.



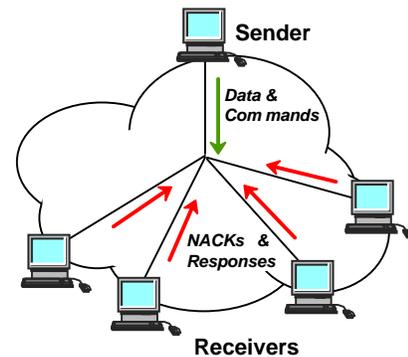
**Figure 1: Screenshot of MDP-based browser client used in Worldwide Internet Mbone Test**

As mentioned in the introduction, MDP has undergone extensive use on the worldwide Internet MBONE. Higher layer applications demonstrated with the protocol toolkit have included web content multicasting, imagery dissemination, directory replication, generic multicast file transfer, and the integration into distributed situational awareness applications. The use of the MDP Application Programming Interface (API) allows applications to use and control protocol features. The present software includes an operational, self-contained basic multicast file transfer application (e.g., tkMDP and/or winMDP) that works with standard web browser and uses the MDP API. Software designers can use the API interface for fine control of the protocol metrics, status, and operation from upper layer applications. The software has been developed to work under Win32 Operating Systems (e.g., Win95, Win98, WinNT) and a variety of UNIX platforms (e.g., Linux, Solaris, NetBSD, IRIX, HPUX). Figure 1 is a screenshot of MDP providing web-based hypertext content and imagery during worldwide Internet Mbone testing using tkMDP client and Netscape as a content viewer (winMDP operates with Netscape or Internet Explorer).

## MDP PROTOCOL OPERATION

This section provides a brief overview of MDP operation. Due to space limitations, detailed protocol discussion will be sparse and the interested reader is referred to additional documentation and updates [see <http://manimac.itd.nrl.navy.mil/MDP>].

We begin by describing the makeup of an MDP reliable multicast session. At a basic level, participants exchange User Datagram Protocol (UDP) packets over an Internet Protocol (IP) network on a common, pre-determined address and port number, and this defines an MDP protocol session. Generally, MDP session participants exchange packets on a common IP “multicast” group address, but point-to-point unicast address transport sessions are also possible as is the use of optional unicast feedback. Multiple MDP sessions may simultaneously exist on different multicast addresses and/or port numbers. While the MDP implementation allows for multiple senders within the context of a single MDP session (i.e. many-to-many operation), a single sender with multiple receivers will be assumed in the remaining discussion here. Figure 2 depicts the general relationship of the sender and receivers and summarizes the types of messages generated by each.



**Figure 2: MDP Group Members and Messages**

An MDP sender primarily generates messages of type MDP\_DATA and MDP\_PARITY to carry data content and related parity-based repair information for the bulk data (or file) objects being transferred. MDP\_PARITY information is usually sent only on explicit repair demand, but the protocol implementation also supports optional “proactive” MDP\_PARITY, which is sent with the initial data transmission. Another sender message of type MDP\_INFO is also defined and used to carry optional context information for a given transport object. A sender generates messages of type MDP\_CMD to perform certain protocol operations such as end-of-transmission indication, object flushing, round trip time estimation, optional positive acknowledgement requests, and session squelch commands.

An MDP receiver generates messages of type MDP\_NACK or optionally MDP\_ACK in response to transmission periods and/or commands from a sender. The MDP\_NACK messages are generated in response to detected data transmission losses. Sequencing information is used to detect loss and is embedded in the all transmitted packets from the sender. As mentioned, MDP does not require the use of positive acknowledgements, but MDP\_ACK messages are defined and optionally used to meet potential application requirements for object receipt bookkeeping and active congestion feedback in future algorithms.

### Server Transmission

In the current MDP implementation, protocol activity within a session is initiated by the transmission of data by a source node. Session data is comprised of serialized *segments* of *objects*

enqueued for transmission by the source application. The transmission rate and the format of data content of an MDP *object* is determined by several source-based protocol parameters. The *transmit\_rate* parameter governs the peak rate data is transmitted by a session participant in units of bits per second. The multicast transmission rate by the source, including the data, repairs, and commands, is upper bounded by this parameter. Dynamically adjusting this parameter supports implementation of dynamic, rate-based congestion control as will be described later in this paper. The *segment\_size* parameter determines the maximum message payload size the source uses for packet transmissions. Data object fragmentation and re-assembly is provided based upon this parameter.

The MDP source application has the option of setting and advertising a small amount of "out-of-band" information (*info*) for each object enqueued for transmission. For an example of how this may be used, consider a file transfer application where MIME-type information and/or name identification for file content might be embedded in the *info* portion of an MDP transport object. At present, the amount of *info* may be up to *segment\_size* bytes according to the server settings. Thus the *info* associated with an *object* is transmitted as a single MDP message. Thus MDP\_INFO packets provide an alternative means to provide reliable multicast data that is not of a bulk nature (e.g., small data objects that fit into a single segment size packet).

The *block\_size* and *max\_parity* parameters used during a session affect how the server calculates, maintains, and transmits parity-based repair messages. The MDP software release presently uses shortened, Reed-Solomon encoding to construct repair segments based on a block of data segments. The *block\_size* parameter corresponds to the number of MDP\_DATA messages per Reed-Solomon encoding block while the *max\_parity* parameter corresponds to the number of repair (MDP\_PARITY) segments the source calculates and maintains per block.

#### **Client Reception and Synchronization**

Upon reception of MDP\_INFO or MDP\_DATA messages from a new session source, an MDP client will "synchronize" with a source by beginning to maintain state on the source using the object segmentation and encoding parameters and current transmission sequencing information embedded in the received messages. For this reason, certain information is embedded in all MDP\_INFO, MDP\_DATA, and MDP\_PARITY messages transmitted by the source.

#### **Client NACK Process**

Once a client "synchronizes" with a server, it begins tracking the sequence of transmission using the *object\_id* and *offset* fields contained in the data and commands sent by the server. If the client detects missing data from the server at the end of an encoding block, end of an object transmission, or upon receipt of an MDP\_CMD\_FLUSH command, it initiates a process to request repairs from the server. Note that the end-of-block or end-of-object boundaries are detected either explicitly by presence of the MDP\_FLAG\_BLOCK\_END flag in a received message or implicitly by the receipt of a message for an object or encoding block beyond the last incompletely received block.

To initiate the repair request process and to facilitate the suppression of redundant NACK transmission, clients use a random hold-off timer to delay repair requests. Thus, if another

NACK for the same (or more) repair information (or the repair information itself) arrives before the timeout ends, the client suppresses its transmission of an MDP\_NACK message. Note that after transmission or suppression of the NACK occurs, an additional timeout period is used before reinitiating the same repair request attempt (this additional timeout allows the source time to provide a group repair response to the previous request).

#### **Server NACK Aggregation and Repair**

Upon receipt of an MDP\_NACK from a client, the server parses and tracks the Repair Request data and begins a hold-off timeout period before responding to pending repair requests. This allows the source to receive and aggregate multiple Repair Requests from other clients. Note that during this repair response hold-off time, the server continues to transmit data for new or other pending objects. The exception to the hold-off timeout is that retransmission of MDP\_INFO messages occurs immediately upon receipt of the MDP\_NACK message. Note, however, that repeat retransmission of duplicate MDP\_INFO is restricted to a maximum of once per timeout period. The reason a server retransmits the MDP\_INFO repair quickly is because there is no need to aggregate multiple repair requests for maximum efficiency as when handling bulk data control. This added responsiveness to the repair cycle for MDP\_INFO messages is potentially useful in the context of reliable multicast session control or for certain types of multicast application data (e.g., situational awareness).

The MDP protocol attempts to maximize its use of parity segments it has calculated for repair transmissions. For example, if during an initial repair cycle for an object, receivers request a portion of the available parity segments, the server uses parity segments from any remaining unused portion for subsequent repair cycles in the same encoding block. There is a significant gain to this approach as the client parity decoding process fills missing data segments (erasures) with any combination of the same number of, but different, parity segments. Thus, when reliability requires multiple repair cycles (more than one repair attempt) to complete an encoded object block, receivers are freed from the difficulty of tracking and requesting explicit sequences of parity segments. Given a sufficient number of parity segments calculated by a source and nominal packet loss, sources generally never need to send the same segment twice, thus maximizing the use of the parity information and minimizing the maintenance of redundant data requests amongst receivers.

#### **Positive Object Acknowledgement Process**

In addition to its pure NACK-based mode of operation, MDP provides an optional mode for a source to request positive acknowledgment (ACK) or receipt of individual transport objects from a specific set of receivers in the group. The list of receivers providing receipt acknowledgement is determined by the server application with *a priori* knowledge of participating nodes and/or by receivers who indicate an ACKing status with a flag in their MDP\_REPORT messages. Positive acknowledgment can be requested for all transport objects sent by the server or may be applied at certain "watermark" progress points in the course of transmission of a series (stream) of transport objects.

### *Silent Receiver Operation*

MDP provides some useful protocol features for emission controlled (EMCON), or "silent" client operation. When it is not possible or desirable for the client nodes to transmit, by any means, messages back to the server node or if delivery delay is of paramount concern the protocol can provide proactive parity data for more effective unidirectional repairing. This is an optional feature the protocol provides through the *auto\_parity* parameter. The *auto\_parity* feature of the server transmission sequence provides a percentage of parity repairing packets with the forward multicast data stream. Additionally, clients can combine the information from multiple repeat transmissions of transport object data into a complete object. The MDP API provides support for basic EMCON modes of operation. The MDP Toolkit is currently being refined in this area and is anticipated to provide support for different, application-defined EMCON concepts of operation in the future.

### **SOME TEST RESULTS AND DISCUSSIONS**

Numerous live tests have been conducted and empirical evidence has been gathered regarding various aspects of MDP performance over the last few years. In this section, we discuss a few experiments and summarize results.

An earlier paper [Macker97a] presented analytical results predicting significant protocol improvements in efficiency for NACK-based reliable multicast when using proposed parity packet repairing methods. MDP now provides an implemented model of that proposal and an example empirical study was performed using protocol parameters similar to those used in the earlier analytical study. The test included one multicast source and twenty independent multicast receivers. The source used a 20 packet per block coding boundary and a ten-percent independent packet loss rate was generated at the receivers. Payloads were 1024 bytes with a constant transmit rate of 9600 bits per second was used. In conclusion, 122 files of around 62 Kbytes were reliably transported with results given in Table 1.

**Table 1: MDP Parity Repairing Experiment**

<b>MDP Trial Summary</b>	<b>No Parity Used</b>	<b>Parity Used</b>
Session Transfer Time	3 hours, 44 minutes, 38 seconds	2 hours, 16 minutes, 54 seconds
Total Data/Repair Transmissions	7447 Data + 8045 Repair = 15492	7447 Data + 1982 Repair = 9429
Total Client NACK Transmissions	6111	1726
Total Server Data Transmitted	16,083,804 bytes	9,719,414 bytes

The results in Table 1 illustrate several things. First, total required session transfer time is significantly reduced (completed in 61% the time of using no parity repairing). Second, overall protocol message transmission requirements are significantly reduced. Message reduction occurs on both the server (75% fewer repairs) and the receiver side (72% fewer NACKs).

MDP was also demonstrated within an asymmetric, high-speed Urban Warrior experiment during 1998. MDP was shown to perform with high robustness and throughout the test and results were reported in a MILCOM 98 paper [Krout98]. These tests are continuing and further enhancements to MDP for use in asymmetric network architectures are being considered.

### **RELATED WORK AND ONGOING ISSUES**

This section describes ongoing MDP related research and design efforts. While MDP is presently an available reliable multicast software toolkit, past and ongoing research has been conducted using MDP as a framework.

#### *UC Berkeley/VINT ns Network Simulator Model*

A complete model of the MDP protocol has been developed in the ns simulation environment and is supporting ongoing studies of MDP performance and continuing research. The following areas are presently being studied and evaluated: MDP congestion control enhancements, TCP friendliness issues, wireless and mobile network performance, reliable multicast and traffic management [Macker97c], general protocol scalability and performance under various networking architectures.

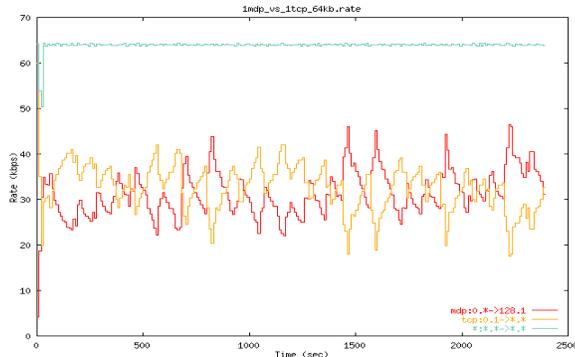
#### *Rate-based Congestion Control*

At NRL, the ns simulation model is being used to research and develop new rate-based congestion control enhancements to MDP. This general area of rate-based congestion control for reliable multicast presents numerous technical challenges [Whetten98]. TCP friendliness, of some form, should be demonstrated before reliable multicast solutions are considered for general Internet protocol standardization [Mankin98]. Our initial approach is to investigate end-to-end extensions of the MDP protocol engine to provide congestion sensitive rate-control based loosely upon a steady state TCP model reported in [Padhye98]. If successful, this approach gives MDP a basis for TCP friendliness across a variety of conditions. At the time of writing, we have an initial integrated algorithm working and are actively testing stability, robustness, and TCP friendliness. Our initial design uses multicast group loss and round trip time (RTT) estimation schemes to estimate worst path model friendliness [Whetten98] among the receiver group. Linear increase and exponential decrease is used to adapt the rate dynamically according to a goal rate calculated from the steady state TCP rate model.

Previous work at USC and HRL [DeLucia98] used MDP as a framework to demonstrate rate-based congestion control and TCP friendliness using the concept of representatives. In this work, representatives among the multicast receiver set were dynamically elected based upon loss reporting and these designated receivers would provide more proactive feedback to assist in a rate-based congestion control algorithm. In our present design, we have retained a concept of "designated receivers" to improve scalability but we deploy algorithms using the worst path estimation based upon a steady state TCP transfer function.

Figure 3 provides a performance snapshot of TCP and MDP dynamically sharing a 64 kbps bottleneck link. These results were obtained by simulating TCP unicast and MDP multicast sessions sourced from separate 10 Mbps links (e.g., Ethernet LAN) jointly traversing a bottleneck link of 64 kbps (e.g., satellite link). The y-axis shows the throughput from each transport flow across the

bottleneck link over the total 40-minute test period. The initial results here show that MDP exhibits reasonable dynamic fairness with a simultaneous TCP session over the bottleneck link. Other recent tests of alternate scenarios (e.g., bandwidth, dynamics) show similar behavior of relatively fair dynamic sharing and good stability. While the work is ongoing to refine our initial dynamic algorithm and evaluate more complex scenarios, these initial results are highly encouraging.



**Figure 3: Example End-to-end Rate-Based Congestion Control Results and TCP friendliness**

The challenge of developing an MDP rate-based congestion control approach has been to retain much of the scalability and efficiency of an existing NACK-based reliable multicast design while obtaining reasonable feedback and group estimates of loss and RTT for use in the TCP rate friendly calculations. Future work is planned to investigate the use of router assisted messaging (e.g., early congestion notification (ECN)) to further ease the burden of end-to-end multicast congestion control implementation and performance. The use of enhanced traffic management can also provide additional support to congestion control approaches and support more robust operation in wireless network scenarios. While there are many ongoing research aspects to reliable multicast congestion control, we are encouraged by the initial results achieved with a working dynamic end-to-end implementation based upon MDP.

Congestion control enhancements are being developed and integrated into the existing MDP implementation as options. It is envisioned that many scenarios may continue to use rate-controlled MDP functionality without dynamic congestion control enhancements. Future releases of the protocol toolkit will offer the option to run with or without dynamic congestion control features.

## SUMMARY

In conclusion, we have presented an overview of the MDP reliable multicast protocol and toolkit. We explained some of the history and design approach taken in MDP, including the use and integration of parity-based repairing schemes. A brief overview of MDP message types, modes of operation, and protocol behavior was provided. After the protocol description, we presented empirical results demonstrating overhead reduction and delay improvements resulting from MDP multicast parity-based repairing. These results compare well with previous analytical projections [Macker97a]. Finally, early NRL research and results regarding rate-based multicast congestion control were discussed. End-to-end “TCP friendly” behavior has been demonstrated and initial results from simulation studies are encouraging.

## REFERENCES

- [Clark90] D. Clark, D. Tennenhouse, “Architectural Considerations for a New Generation of Protocols”. In Proc. ACM SIGCOMM, pages 201--208, September 1990.
- [Deering89] S. Deering. “Host Extensions for IP Multicasting”. Internet RFC 1112, August 1989.
- [DeLucia98] D. DeLucia, K. Obraczka, "Congestion Control Performance of a Reliable Multicast Protocol", Proc. IEEE ICNP'98, August 1998.
- [Floyd95] S. Floyd, V. Jacobson, S. McCanne, C. Liu, and L. Zhang. “A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing”, Proc. ACM SIGCOMM, August 1995.
- [Gossink98] D. Gossink, J. Macker, "Reliable Multicast and Integrated Parity Retransmission with Channel Estimation", IEEE GLOBECOM 98'.
- [Kruus98] P. Kruus, J. Macker, "*Techniques and Issues in Multicast Security*", Proc. IEEE MILCOM 98.
- [Macker96a] J. Macker, W. Dang, "*The Multicast Dissemination Protocol version 1 (mdpv1) Framework*", Technical White Paper, 1996.
- [Macker96b] Macker, J.P., M.S. Corson, E. Klinker, “Reliable Multicast Data Delivery for Military Internetworking”, October 1996, Proc. IEEE MILCOM 96’.
- [Macker97a] J. Macker, "*Reliable Multicast Transport and Integrated Erasure-based Forward Error Correction*", Proc. IEEE MILCOM 97, Oct 1997.
- [Macker97b] J. Macker, "*Integrated Erasure-Based Coding for Reliable Multicast Transmission*", IRTF Meeting presentation, March 1997.
- [Macker97c] J. Macker, M.S. Corson, "*Controlled Link Sharing and Reliable Multicast for Asymmetric Networks* ", Pacific Telecommunications Conference, Jan 1997.
- [Metzner84] J. Metzner, “*An Improved Broadcast Retransmission Protocol*”, IEEE Transactions on Communications, Vol. Com-32, No.6, June 1984.
- [Padhye98] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, “Modeling TCP Throughput: A Simple Model and its Empirical Validation”. Univ of Mass, Technical Report CMPCSI TR 98-008.
- [Pingali93] S. Pingali, D. Towsley, J. Kurose. “A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols”. In Proc. INFOCOM, San Francisco, CA, October 1993.
- [Whetton98] B. Whetton, J. Conlan. “A Rate Based Congestion Control Scheme for Reliable Multicast”, Technical White Paper, Oct 1998.
- [Mankin98] A. Mankin, A. Romanow, S. Bradner, V. Paxson, “[IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols](#)”, RFC 2357, June 1998.
- [Levine 96] B.N. Levine, J.J. Garcia-Luna-Aceves. "A Comparison of Known Classes of Reliable Multicast Protocols", Proc. International Conference on Network Protocols (ICNP-96), Columbus, Ohio, Oct 29--Nov 1, 1996