

Reliable Multicast Transport (RMT)
Working Group
Internet-Draft
Expires: April 24, 2006

Luby
Watson
Digital Fountain
Gemmell
Microsoft Research
Vicisano
Cisco Systems, Inc.
Rizzo
Univ. di Pisa
Handley
University College London
Crowcroft
University of Cambridge
October 21, 2005

Layered Coding Transport (LCT) Building Block
draft-ietf-rmt-bb-lct-revised-01

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [Section 6 of BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on April 24, 2006.

Copyright Notice

Copyright (C) The Internet Society (2005).

Abstract

Layered Coding Transport (LCT) provides transport level support for reliable content delivery and stream delivery protocols. LCT is specifically designed to support protocols using IP multicast, but also provides support to protocols that use unicast. LCT is compatible with congestion control that provides multiple rate delivery to receivers and is also compatible with coding techniques that provide reliable delivery of content.

Table of Contents

1.	Introduction	3
2.	Rationale	4
3.	Functionality	6
4.	Applicability	9
4.1.	Environmental Requirements and Considerations	10
4.2.	Delivery service models	12
4.3.	Congestion Control	14
5.	Packet Header Fields	16
5.1.	LCT header format	16
5.2.	Header-Extension Fields	22
6.	Operations	25
6.1.	Sender Operation	25
6.2.	Receiver Operation	27
7.	Requirements from Other Building Blocks	29
8.	Security Considerations	31
9.	IANA Considerations	32
9.1.	Namespace declaration for LCT Header Extension Types	32
9.2.	LCT Header Extension Type registration	32
10.	Acknowledgments	33
11.	Change Log	34
11.1.	RFC3451 to draft-ietf-rmt-bb-lct-revised-00	34
11.2.	draft-ietf-rmt-bb-lct-revised-00 to -01	34
12.	References	35
12.1.	Normative References	35
12.2.	Informative References	35
	Authors' Addresses	38
	Intellectual Property and Copyright Statements	40

1. Introduction

Layered Coding Transport provides transport level support for reliable content delivery and stream delivery protocols. Layered Coding Transport is specifically designed to support protocols using IP multicast, but also provides support to protocols that use unicast. Layered Coding Transport is compatible with congestion control that provides multiple rate delivery to receivers and is also compatible with coding techniques that provide reliable delivery of content.

This document describes a building block as defined in [[RFC3048](#)]. This document is a product of the IETF RMT WG and follows the general guidelines provided in [[RFC3269](#)].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [[RFC2119](#)].

2. Rationale

LCT provides transport level support for massively scalable protocols using the IP multicast network service. The support that LCT provides is common to a variety of very important applications, including reliable content delivery and streaming applications.

An LCT session comprises multiple channels originating at a single sender that are used for some period of time to carry packets pertaining to the transmission of one or more objects that can be of interest to receivers. The logic behind defining a session as originating from a single sender is that this is the right granularity to regulate packet traffic via congestion control. One rationale for using multiple channels within the same session is that there are massively scalable congestion control protocols that use multiple channels per session. These congestion control protocols are considered to be layered because a receiver joins and leaves channels in a layered order during its participation in the session. The use of layered channels is also useful for streaming applications.

There are coding techniques that provide massively scalable reliability and asynchronous delivery which are compatible with both layered congestion control and with LCT. When all are combined the result is a massively scalable reliable asynchronous content delivery protocol that is network friendly. LCT also provides functionality that can be used for other applications as well, e.g., layered streaming applications.

LCT avoids providing functionality that is not massively scalable. For example, LCT does not provide any mechanisms for sending information from receivers to senders, although this does not rule out protocols that both use LCT and do require sending information from receivers to senders.

LCT includes general support for congestion control that must be used. It does not, however, specify which congestion control should be used. The rationale for this is that congestion control must be provided by any protocol that is network friendly, and yet the different applications that can use LCT will not have the same requirements for congestion control. For example, a content delivery protocol may strive to use all available bandwidth between receivers and the sender. It must, therefore, drastically back off its rate when there is competing traffic. On the other hand, a streaming delivery protocol may strive to maintain a constant rate instead of trying to use all available bandwidth, and it may not back off its rate as fast when there is competing traffic.

Beyond support for congestion control, LCT provides a number of fields and supports functionality commonly required by many protocols. For example, LCT provides a Transmission Session ID that can be used to identify which session each received packet belongs to. This is important because a receiver may be joined to many sessions concurrently, and thus it is very useful to be able to demultiplex packets as they arrive according to which session they belong to. As another example, LCT provides optional support for identifying which object each packet is carrying information about. Therefore, LCT provides many of the commonly used fields and support for functionality required by many protocols.

3. Functionality

An LCT session consists of a set of logically grouped LCT channels associated with a single sender carrying packets with LCT headers for one or more objects. An LCT channel is defined by the combination of a sender and an address associated with the channel by the sender. A receiver joins a channel to start receiving the data packets sent to the channel by the sender, and a receiver leaves a channel to stop receiving data packets from the channel.

LCT is meant to be combined with other building blocks so that the resulting overall protocol is massively scalable. Scalability refers to the behavior of the protocol in relation to the number of receivers and network paths, their heterogeneity, and the ability to accommodate dynamically variable sets of receivers. Scalability limitations can come from memory or processing requirements, or from the amount of feedback control and redundant data packet traffic generated by the protocol. In turn, such limitations may be a consequence of the features that a complete reliable content delivery or stream delivery protocol is expected to provide.

The LCT header provides a number of fields that are useful for conveying in-band session information to receivers. One of the required fields is the Transmission Session ID (TSI), which allows the receiver of a session to uniquely identify received packets as part of the session. Another required field is the Congestion Control Information (CCI), which allows the receiver to perform the required congestion control on the packets received within the session. Other LCT fields provide optional but often very useful additional information for the session. For example, the Transport Object Identifier (TOI) identifies which object the packet contains data for. As other examples, the Sender Current Time (SCT) conveys the time when the packet was sent from the sender to the receiver, the Expected Residual Time (ERT) conveys the amount of time the session will be continued for, flags for indicating the close of the session and the close of sending packets for an object, and header extensions for fields that for example can be used for packet authentication.

LCT provides support for congestion control. Congestion control **MUST** be used that conforms to [\[RFC2357\]](#) between receivers and the sender for each LCT session. Congestion control refers to the ability to adapt throughput to the available bandwidth on the path from the sender to a receiver, and to share bandwidth fairly with competing flows such as TCP. Thus, the total flow of packets flowing to each receiver participating in an LCT session **MUST NOT** compete unfairly with existing flow adaptive protocols such as TCP.

A multiple rate or a single rate congestion control protocol can be used with LCT. For multiple rate protocols, a session typically consists of more than one channel and the sender sends packets to the channels in the session at rates that do not depend on the receivers. Each receiver adjusts its reception rate during its participation in the session by joining and leaving channels dynamically depending on the available bandwidth to the sender independent of all other receivers. Thus, for multiple rate protocols, the reception rate of each receiver may vary dynamically independent of the other receivers.

For single rate protocols, a session typically consists of one channel and the sender sends packets to the channel at variable rates over time depending on feedback from receivers. Each receiver remains joined to the channel during its participation in the session. Thus, for single rate protocols, the reception rate of each receiver may vary dynamically but in coordination with all receivers.

Generally, a multiple rate protocol is preferable to a single rate protocol in a heterogeneous receiver environment, since generally it more easily achieves scalability to many receivers and provides higher throughput to each individual receiver. Some possible multiple rate congestion control protocols are described in [\[VIC1998\]](#), [\[BYE2000\]](#), and [\[LUB2002\]](#). A possible single rate congestion control protocol is described in [\[RIZ2000\]](#).

Layered coding refers to the ability to produce a coded stream of packets that can be partitioned into an ordered set of layers. The coding is meant to provide some form of reliability, and the layering is meant to allow the receiver experience (in terms of quality of playout, or overall transfer speed) to vary in a predictable way depending on how many consecutive layers of packets the receiver is receiving.

The concept of layered coding was first introduced with reference to audio and video streams. For example, the information associated with a TV broadcast could be partitioned into three layers, corresponding to black and white, color, and HDTV quality. Receivers can experience different quality without the need for the sender to replicate information in the different layers.

The concept of layered coding can be naturally extended to reliable content delivery protocols when Forward Error Correction (FEC) techniques are used for coding the data stream. Descriptions of this can be found in [\[RIZ1997a\]](#), [\[RIZ1997b\]](#), [\[GEM2000\]](#), [\[VIC1998\]](#) and [\[BYE1998\]](#). By using FEC, the data stream is transformed in such a way that reconstruction of a data object does not depend on the reception of specific data packets, but only on the number of

different packets received. As a result, by increasing the number of layers a receiver is receiving from, the receiver can reduce the transfer time accordingly. Using FEC to provide reliability can increase scalability dramatically in comparison to other methods for providing reliability. More details on the use of FEC for reliable content delivery can be found in [[RFC3453](#)].

Reliable protocols aim at giving guarantees on the reliable delivery of data from the sender to the intended recipients. Guarantees vary from simple packet data integrity to reliable delivery of a precise copy of an object to all intended recipients. Several reliable content delivery protocols have been built on top of IP multicast using methods other than FEC, but scalability was not the primary design goal for many of them.

Two of the key difficulties in scaling reliable content delivery using IP multicast are dealing with the amount of data that flows from receivers back to the sender, and the associated response (generally data retransmissions) from the sender. Protocols that avoid any such feedback, and minimize the amount of retransmissions, can be massively scalable. LCT can be used in conjunction with FEC codes or a layered codec to achieve reliability with little or no feedback.

Protocol instantiations MAY be built by combining the LCT framework with other components. A complete protocol instantiation that uses LCT MUST include a congestion control protocol that is compatible with LCT and that conforms to [[RFC2357](#)]. A complete protocol instantiation that uses LCT MAY include a scalable reliability protocol that is compatible with LCT, it MAY include an session control protocol that is compatible with LCT, and it MAY include other protocols such as security protocols.

4. Applicability

An LCT session comprises a logically related set of one or more LCT channels originating at a single sender. The channels are used for some period of time to carry packets containing LCT headers, and these headers pertain to the transmission of one or more objects that can be of interest to receivers.

LCT is most applicable for delivery of objects or streams in a session of substantial length, i.e., objects or streams that range in aggregate length from hundreds of kilobytes to many gigabytes, and where the duration of the session is on the order of tens of seconds or more.

As an example, an LCT session could be used to deliver a TV program using three LCT channels. Receiving packets from the first LCT channel could allow black and white reception. Receiving the first two LCT channels could also permit color reception. Receiving all three channels could allow HDTV quality reception. Objects in this example could correspond to individual TV programs being transmitted.

As another example, a reliable LCT session could be used to reliably deliver hourly-updated weather maps (objects) using ten LCT channels at different rates, using FEC coding. A receiver may join and concurrently receive packets from subsets of these channels, until it has enough packets in total to recover the object, then leave the session (or remain connected listening for session description information only) until it is time to receive the next object. In this case, the quality metric is the time required to receive each object.

Before joining a session, the receivers MUST obtain enough of the session description to start the session. This MUST include the relevant session parameters needed by a receiver to participate in the session, including all information relevant to congestion control. The session description is determined by the sender, and is typically communicated to the receivers out-of-band. In some cases, as described later, parts of the session description that are not required to initiate a session MAY be included in the LCT header or communicated to a receiver out-of-band after the receiver has joined the session.

An encoder MAY be used to generate the data that is placed in the packet payload in order to provide reliability. A suitable decoder is used to reproduce the original information from the packet payload. There MAY be a reliability header that follows the LCT header if such an encoder and decoder is used. The reliability header helps to describe the encoding data carried in the payload of

the packet. The format of the reliability header depends on the coding used, and this is negotiated out-of-band. As an example, one of the FEC headers described in [[I-D.ietf-rmt-fec-bb-revised](#)] could be used.

For LCT, when multiple rate congestion control is used, congestion control is achieved by sending packets associated with a given session to several LCT channels. Individual receivers dynamically join one or more of these channels, according to the network congestion as seen by the receiver. LCT headers include an opaque field which **MUST** be used to convey congestion control information to the receivers. The actual congestion control scheme to use with LCT is negotiated out-of-band. Some examples of congestion control protocols that may be suitable for content delivery are described in [[VIC1998](#)], [[BYE2000](#)], and [[LUB2002](#)]. Other congestion controls may be suitable when LCT is used for a streaming application.

This document does not specify and restrict the type of exchanges between LCT (or any PI built on top of LCT) and an upper application. Some upper APIs may use an object-oriented approach, where the only possible unit of data exchanged between LCT (or any PI built on top of LCT) and an application, either at a source or at a receiver, is an object. Other APIs may enable a sending or receiving application to exchange a subset of an object with LCT (or any PI built on top of LCT), or may even follow a streaming model. These considerations are outside the scope of this document.

4.1. Environmental Requirements and Considerations

LCT is intended for congestion controlled delivery of objects and streams (both reliable content delivery and streaming of multimedia information).

LCT can be used with both multicast and unicast delivery. LCT requires connectivity between a sender and receivers but does not require connectivity from receivers to a sender. LCT inherently works with all types of networks, including LANs, WANs, Intranets, the Internet, asymmetric networks, wireless networks, and satellite networks. Thus, the inherent raw scalability of LCT is unlimited. However, when other specific applications are built on top of LCT, then these applications by their very nature may limit scalability. For example, if an application requires receivers to retrieve out of band information in order to join a session, or an application allows receivers to send requests back to the sender to report reception statistics, then the scalability of the application is limited by the ability to send, receive, and process this additional data.

LCT requires receivers to be able to uniquely identify and

demultiplex packets associated with an LCT session. In particular, there MUST be a Transport Session Identifier (TSI) associated with each LCT session. The TSI is scoped by the IP address of the sender, and the IP address of the sender together with the TSI MUST uniquely identify the session. If the underlying transport is UDP as described in [\[RFC0768\]](#), then the 16 bit UDP source port number MAY serve as the TSI for the session. The TSI value MUST be the same in all places it occurs within a packet. If there is no underlying TSI provided by the network, transport or any other layer, then the TSI MUST be included in the LCT header.

LCT is presumed to be used with an underlying network or transport service that is a "best effort" service that does not guarantee packet reception or packet reception order, and which does not have any support for flow or congestion control. For example, the Any-Source Multicast (ASM) model of IP multicast as defined in [\[RFC1112\]](#) is such a "best effort" network service. While the basic service provided by [\[RFC1112\]](#) is largely scalable, providing congestion control or reliability should be done carefully to avoid severe scalability limitations, especially in presence of heterogeneous sets of receivers.

There are currently two models of multicast delivery, the Any-Source Multicast (ASM) model as defined in [\[RFC1112\]](#) and the Source-Specific Multicast (SSM) model as defined in [\[HOL2001\]](#). LCT works with both multicast models, but in a slightly different way with somewhat different environmental concerns. When using ASM, a sender S sends packets to a multicast group G, and the LCT channel address consists of the pair (S,G), where S is the IP address of the sender and G is a multicast group address. When using SSM, a sender S sends packets to an SSM channel (S,G), and the LCT channel address coincides with the SSM channel address.

A sender can locally allocate unique SSM channel addresses, and this makes allocation of LCT channel addresses easy with SSM. To allocate LCT channel addresses using ASM, the sender must uniquely chose the ASM multicast group address across the scope of the group, and this makes allocation of LCT channel addresses more difficult with ASM.

LCT channels and SSM channels coincide, and thus the receiver will only receive packets sent to the requested LCT channel. With ASM, the receiver joins an LCT channel by joining a multicast group G, and all packets sent to G, regardless of the sender, may be received by the receiver. Thus, SSM has compelling security advantages over ASM for prevention of denial of service attacks. In either case, receivers SHOULD use mechanisms to filter out packets from unwanted sources.

Some networks are not amenable to some congestion control protocols that could be used with LCT. In particular, for a satellite or wireless network, there may be no mechanism for receivers to effectively reduce their reception rate since there may be a fixed transmission rate allocated to the session.

LCT is compatible with either IPv4 or IPv6 as no part of the packet is IP version specific.

4.2. Delivery service models

LCT can support several different delivery service models. Two examples are briefly described here.

Push service model

One way a push service model can be used for reliable content delivery is to deliver a series of objects. For example, a receiver could join the session and dynamically adapt the number of LCT channels the receiver is joined to until enough packets have been received to reconstruct an object. After reconstructing the object the receiver may stay in the session and wait for the transmission of the next object.

The push model is particularly attractive in satellite networks and wireless networks. In these cases, a session may consist of one fixed rate LCT channel.

A push service model can be used for example for reliable delivery of a large object such as a 100 GB file. The sender could send a Session Description announcement to a control channel and receivers could monitor this channel and join a session whenever a Session Description of interest arrives. Upon receipt of the Session Description, each receiver could join the session to receive packets until enough packets have arrived to reconstruct the object, at which point the receiver could report back to the sender that its reception was completed successfully. The sender could decide to continue sending packets for the object to the session until all receivers have reported successful reconstruction or until some other condition has been satisfied.

There are several features ALC provides to support the push model. For example, the sender can optionally include an Expected Residual Time (ERT) in the packet header that indicates the expected remaining time of packet transmission for either the single object carried in the session or for the object identified by the Transmission Object Identifier (TOI) if there are multiple objects carried in the session. This can be used by receivers to determine if there is enough time remaining in the session to successfully receive enough additional packets to recover the object. If for example there is not enough time, then the push application may have receivers report back to the sender to extend the transmission of packets for the object for enough time to allow the receivers to obtain enough packets to reconstruct the object. The sender could then include an ERT based on the extended object transmission time in each subsequent packet header for the object. As other examples, the LCT header optionally can contain a Close Session flag that indicates when the sender is about to end sending packet to the session and a Close Object flag that indicates when the sender is about to end sending packets to the session for the object identified by the Transmission Object ID. However, these flags are not a completely reliable mechanism and thus the Close Session flag should only be used as a hint of when the session is about to close and the Close Object flag should only be used as a hint of when transmission of packets for the object is about to end.

On-demand content delivery model

For an on-demand content delivery service model, senders typically transmit for some given time period selected to be long enough to allow all the intended receivers to join the session and recover the object. For example a popular software update might be transmitted using LCT for several days, even though a receiver may be able to complete the download in one hour total of connection time, perhaps spread over several intervals of time. In this case the receivers join the session at any point in time when it is active. Receivers leave the session when they have received enough packets to recover the object. The receivers, for example, obtain a Session Description by contacting a web server.

In this case the receivers join the session, and dynamically adapt the number of LCT channels they subscribe to according to the available bandwidth. Receivers then drop from the session when they have received enough packets to recover the object.

As an example, assume that an object is 50 MB. The sender could send 1 KB packets to the first LCT channel at 50 packets per second, so that receivers using just this LCT channel could complete reception of the object in 1,000 seconds in absence of loss, and would be able to complete reception even in presence of some substantial amount of losses with the use of coding for reliability. Furthermore, the sender could use a number of LCT channels such that the aggregate rate of 1 KB packets to all LCT channels is 1,000 packets per second, so that a receiver could be able to complete reception of the object in as little 50 seconds (assuming no loss and that the congestion control mechanism immediately converges to the use of all LCT channels).

Other service models

There are many other delivery service models that LCT can be used for that are not covered above. As examples, a live streaming or an on-demand archival content streaming service model. A description of the many potential applications, the appropriate delivery service model, and the additional mechanisms to support such functionalities when combined with LCT is beyond the scope of this document. This document only attempts to describe the minimal common scalable elements to these diverse applications using LCT as the delivery transport.

4.3. Congestion Control

The specific congestion control protocol to be used for LCT sessions depends on the type of content to be delivered. While the general behavior of the congestion control protocol is to reduce the throughput in presence of congestion and gradually increase it in the absence of congestion, the actual dynamic behavior (e.g. response to single losses) can vary.

Some possible congestion control protocols for reliable content delivery using LCT are described in [[VIC1998](#)], [[BYE2000](#)], and

[[LUB2002](#)]. Different delivery service models might require different congestion control protocols.

5. Packet Header Fields

Packets sent to an LCT session **MUST** include an "LCT header". The LCT header format is described below.

Other building blocks **MAY** describe some of the same fields as described for the LCT header. It is **RECOMMENDED** that protocol instantiations using multiple building blocks include shared fields at most once in each packet. Thus, for example, if another building block is used with LCT that includes the optional Expected Residual Time field, then the Expected Residual Time field **SHOULD** be carried in each packet at most once.

The position of the LCT header within a packet **MUST** be specified by any protocol instantiation that uses LCT.

5.1. LCT header format

The LCT header is of variable size, which is specified by a length field in the third byte of the header. In the LCT header, all integer fields are carried in "big-endian" or "network order" format, that is, most significant byte (octet) first. Bits designated as "padding" or "reserved" (r) **MUST** be set to 0 by senders and ignored by receivers. Unless otherwise noted, numeric constants in this specification are in decimal (base 10).

The format of the default LCT header is depicted in Figure 1.

bits.

Transport Session Identifier flag (S): 1 bit

This is the number of full 32-bit words in the TSI field. The TSI field is $32*S + 16*H$ bits in length, i.e. the length is either 0 bits, 16 bits, 32 bits, or 48 bits.

Transport Object Identifier flag (O): 2 bits

This is the number of full 32-bit words in the TOI field. The TOI field is $32*O + 16*H$ bits in length, i.e., the length is either 0 bits, 16 bits, 32 bits, 48 bits, 64 bits, 80 bits, 96 bits, or 112 bits.

Half-word flag (H): 1 bit

The TSI and the TOI fields are both multiples of 32-bits plus $16*H$ bits in length. This allows the TSI and TOI field lengths to be multiples of a half-word (16 bits), while ensuring that the aggregate length of the TSI and TOI fields is a multiple of 32-bits.

Sender Current Time present flag (T): 1 bit

$T = 0$ indicates that the Sender Current Time (SCT) field is not present. $T = 1$ indicates that the SCT field is present. The SCT is inserted by senders to indicate to receivers how long the session has been in progress.

Expected Residual Time present flag (R): 1 bit

$R = 0$ indicates that the Expected Residual Time (ERT) field is not present. $R = 1$ indicates that the ERT field is present. The ERT is inserted by senders to indicate to receivers how much longer the session / object transmission will continue.

Senders MUST NOT set $R = 1$ when the ERT for the session is more than $2^{32}-1$ time units (approximately 49 days), where time is measured in units of milliseconds.

Close Session flag (A): 1 bit

Normally, A is set to 0. The sender MAY set A to 1 when termination of transmission of packets for the session is imminent. A MAY be set to 1 in just the last packet transmitted for the session, or A MAY be set to 1 in the last few seconds of packets transmitted for the session. Once the sender sets A to 1

in one packet, the sender SHOULD set A to 1 in all subsequent packets until termination of transmission of packets for the session. A received packet with A set to 1 indicates to a receiver that the sender will immediately stop sending packets for the session. When a receiver receives a packet with A set to 1 the receiver SHOULD assume that no more packets will be sent to the session.

Close Object flag (B): 1 bit

Normally, B is set to 0. The sender MAY set B to 1 when termination of transmission of packets for an object is imminent. If the TOI field is in use and B is set to 1 then termination of transmission for the object identified by the TOI field is imminent. If the TOI field is not in use and B is set to 1 then termination of transmission for the one object in the session identified by out-of-band information is imminent. B MAY be set to 1 in just the last packet transmitted for the object, or B MAY be set to 1 in the last few seconds packets transmitted for the object. Once the sender sets B to 1 in one packet for a particular object, the sender SHOULD set B to 1 in all subsequent packets for the object until termination of transmission of packets for the object. A received packet with B set to 1 indicates to a receiver that the sender will immediately stop sending packets for the object. When a receiver receives a packet with B set to 1 then it SHOULD assume that no more packets will be sent for the object to the session.

LCT header length (HDR_LEN): 8 bits

Total length of the LCT header in units of 32-bit words. The length of the LCT header MUST be a multiple of 32-bits. This field can be used to directly access the portion of the packet beyond the LCT header, i.e., to the first other header if it exists, or to the packet payload if it exists and there is no other header, or to the end of the packet if there are no other headers or packet payload.

Codepoint (CP): 8 bits

An opaque identifier which is passed to the packet payload decoder to convey information on the codec being used for the packet payload. The mapping between the codepoint and the actual codec is defined on a per session basis and communicated out-of-band as part of the session description information. The use of the CP field is similar to the Payload Type (PT) field in RTP headers as described in [[RFC1889](#)].

Congestion Control Information (CCI): 32, 64, 96 or 128 bits

Used to carry congestion control information. For example, the congestion control information could include layer numbers, logical channel numbers, and sequence numbers. This field is opaque for the purpose of this specification.

This field MUST be 32 bits if C=0.

This field MUST be 64 bits if C=1.

This field MUST be 96 bits if C=2.

This field MUST be 128 bits if C=3.

Transport Session Identifier (TSI): 0, 16, 32 or 48 bits

The TSI uniquely identifies a session among all sessions from a particular sender. The TSI is scoped by the IP address of the sender, and thus the IP address of the sender and the TSI together uniquely identify the session. Although a TSI in conjunction with the IP address of the sender always uniquely identifies a session, whether or not the TSI is included in the LCT header depends on what is used as the TSI value. If the underlying transport is UDP, then the 16 bit UDP source port number MAY serve as the TSI for the session. If the TSI value appears multiple times in a packet then all occurrences MUST be the same value. If there is no underlying TSI provided by the network, transport or any other layer, then the TSI MUST be included in the LCT header.

The TSI MUST be unique among all sessions served by the sender during the period when the session is active, and for a large period of time preceding and following when the session is active. A primary purpose of the TSI is to prevent receivers from inadvertently accepting packets from a sender that belong to sessions other than the sessions receivers are subscribed to. For example, suppose a session is deactivated and then another session is activated by a sender and the two sessions use an overlapping set of channels. A receiver that connects and remains connected to the first session during this sender activity could possibly accept packets from the second session as belonging to the first session if the TSI for the two sessions were identical. The mapping of TSI field values to sessions is outside the scope of this document and is to be done out-of-band.

The length of the TSI field is $32*S + 16*H$ bits. Note that the aggregate lengths of the TSI field plus the TOI field is a multiple of 32 bits.

Transport Object Identifier (TOI): 0, 16, 32, 48, 64, 80, 96 or 112 bits.

This field indicates which object within the session this packet pertains to. For example, a sender might send a number of files in the same session, using TOI=0 for the first file, TOI=1 for the second one, etc. As another example, the TOI may be a unique global identifier of the object that is being transmitted from several senders concurrently, and the TOI value may be the output of a hash function applied to the object. The mapping of TOI field values to objects is outside the scope of this document and is to be done out-of-band. The TOI field **MUST** be used in all packets if more than one object is to be transmitted in a session, i.e. the TOI field is either present in all the packets of a session or is never present.

The length of the TOI field is $32*O + 16*H$ bits. Note that the aggregate lengths of the TSI field plus the TOI field is a multiple of 32 bits.

Sender Current Time (SCT): 0 or 32 bits

This field represents the current clock at the sender and at the time this packet was transmitted, measured in units of 1ms and computed modulo 2^{32} units from the start of the session.

OR

This field represents the current clock at the sender expressed as the 32 most significant bits of a 64 bit Network Time Protocol (NTP) [[RFC1305](#)] time value. These 32 bits provide an unsigned integer representing the time in seconds relative to 0 hours 1 January 1900. Handling of wraparound of the 32 bit time is outside the scope of NTP and LCT.

This field **MUST NOT** be present if T=0 and **MUST** be present if T=1.

Expected Residual Time (ERT): 0 or 32 bits

This field represents the sender expected residual transmission time for the current session or for the transmission of the current object, measured in units of 1ms. If the packet containing the ERT field also contains the TOI field, then ERT refers to the object corresponding to the TOI field, otherwise it

refers to the session.

This field MUST NOT be present if $R=0$ and MUST be present if $R=1$.

5.2. Header-Extension Fields

Header Extensions are used in LCT to accommodate optional header fields that are not always used or have variable size. Examples of the use of Header Extensions include:

- o Extended-size versions of already existing header fields.
- o Sender and Receiver authentication information.

The presence of Header Extensions can be inferred by the LCT header length (HDR_LEN): if HDR_LEN is larger than the length of the standard header then the remaining header space is taken by Header Extension fields.

If present, Header Extensions MUST be processed to ensure that they are recognized before performing any congestion control procedure or otherwise accepting a packet. The default action for unrecognized header extensions is to ignore them. This allows the future introduction of backward-compatible enhancements to LCT without changing the LCT version number. Non backward-compatible header extensions CANNOT be introduced without changing the LCT version number.

There are two formats for Header Extension fields, as depicted in Figure 2. The first format is used for variable-length extensions, with Header Extension Type (HET) values between 0 and 127. The second format is used for fixed length (one 32-bit word) extensions, using HET values from 127 to 255.

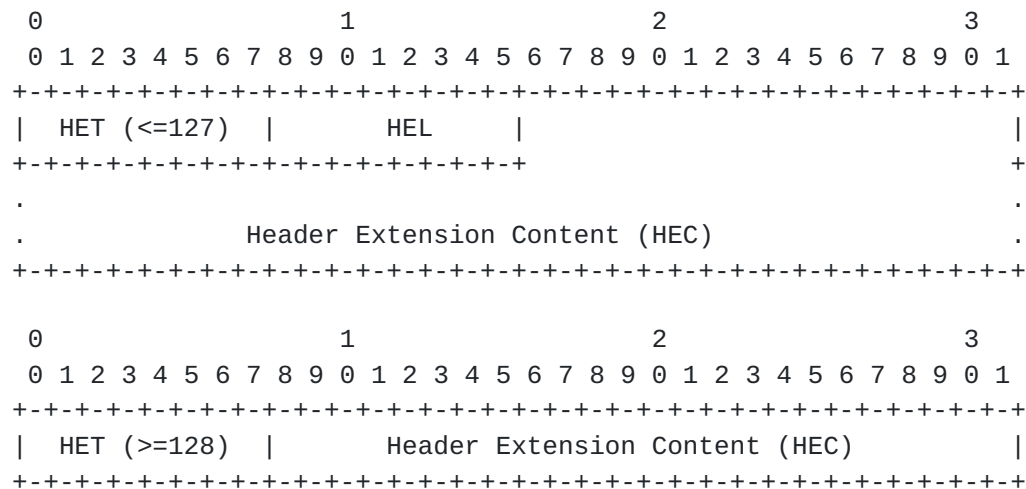


Figure 2: Format of additional headers

The explanation of each sub-field is the following:

Header Extension Type (HET): 8 bits

The type of the Header Extension. This document defines a number of possible types. Additional types may be defined in future versions of this specification. HET values from 0 to 127 are used for variable-length Header Extensions. HET values from 128 to 255 are used for fixed-length 32-bit Header Extensions.

Header Extension Length (HEL): 8 bits

The length of the whole Header Extension field, expressed in multiples of 32-bit words. This field **MUST** be present for variable-length extensions (HET between 0 and 127) and **MUST NOT** be present for fixed-length extensions (HET between 128 and 255).

Header Extension Content (HEC): variable length

The content of the Header Extension. The format of this sub-field depends on the Header Extension type. For fixed-length Header Extensions, the HEC is 24 bits. For variable-length Header Extensions, the HEC field has variable size, as specified by the HEL field. Note that the length of each Header Extension field **MUST** be a multiple of 32 bits. Also note that the total size of the LCT header, including all Header Extensions and all optional header fields, cannot exceed 255 32-bit words.

LCT Header Extensions with general applicability to any protocol which makes use of LCT **SHOULD** be defined in the ranges [0,63] or

[128,191] inclusive. LCT Header Extensions with narrower applicability (for example to a single Protocol Instantiation) SHOULD be defined in the ranges [64,127] or [191,255] inclusive.

The following LCT Header Extensions are defined by this specification:

EXT_NOP, HET=0 No-Operation extension. The information present in this extension field MUST be ignored by receivers.

EXT_AUTH, HET=1 Packet authentication extension Information used to authenticate the sender of the packet. The format of this Header Extension and its processing is outside the scope of this document and is to be communicated out-of-band as part of the session description.

It is RECOMMENDED that senders provide some form of packet authentication. If EXT_AUTH is present, whatever packet authentication checks that can be performed immediately upon reception of the packet SHOULD be performed before accepting the packet and performing any congestion control-related action on it.

Some packet authentication schemes impose a delay of several seconds between when a packet is received and when the packet is fully authenticated. Any congestion control related action that is appropriate MUST NOT be postponed by any such full packet authentication.

All senders and receivers implementing LCT MUST support the EXT_NOP Header Extension and MUST recognize EXT_AUTH, but MAY NOT be able to parse its content.

6. Operations

6.1. Sender Operation

Before joining an LCT session a receiver MUST obtain a session description. The session description MUST include:

- o The sender IP address;
- o The number of LCT channels;
- o The addresses and port numbers used for each LCT channel;
- o The Transport Session ID (TSI) to be used for the session;
- o Enough information to determine the congestion control protocol being used;
- o Enough information to determine the packet authentication scheme being used if it is being used.

The session description could also include, but is not limited to:

- o The data rates used for each LCT channel;
- o The length of the packet payload;
- o The mapping of TOI value(s) to objects for the session;
- o Any information that is relevant to each object being transported, such as when it will be available within the session, for how long, and the length of the object;

Protocol instantiations using LCT MAY place additional requirements on what must be included in the session description. For example, a protocol instantiation might require that the data rates for each channel, or the mapping of TOI value(s) to objects for the session, or other information related to other headers that might be required to be included in the session description.

The session description could be in a form such as SDP as defined in [RFC2327], or XML metadata as defined in [RFC3023], or HTTP/Mime headers as defined in [RFC2616], etc. It might be carried in a session announcement protocol such as SAP as defined in [RFC2974], obtained using a proprietary session control protocol, located on a Web page with scheduling information, or conveyed via E-mail or other out-of-band methods. Discussion of session description format, and distribution of session descriptions is beyond the scope of this

document.

Within an LCT session, a sender using LCT transmits a sequence of packets, each in the format defined above. Packets are sent from a sender using one or more LCT channels which together constitute a session. Transmission rates may be different in different channels and may vary over time. The specification of the other building block headers and the packet payload used by a complete protocol instantiation using LCT is beyond the scope of this document. This document does not specify the order in which packets are transmitted, nor the organization of a session into multiple channels. Although these issues affect the efficiency of the protocol, they do not affect the correctness nor the inter-operability of LCT between senders and receivers.

Several objects can be carried within the same LCT session. In this case, each object MUST be identified by a unique TOI. Objects MAY be transmitted sequentially, or they MAY be transmitted concurrently. It is good practice to only send objects concurrently in the same session if the receivers that participate in that portion of the session have interest in receiving all the objects. The reason for this is that it wastes bandwidth and networking resources to have receivers receive data for objects that they have no interest in.

Typically, the sender(s) continues to send packets in a session until the transmission is considered complete. The transmission may be considered complete when some time has expired, a certain number of packets have been sent, or some out-of-band signal (possibly from a higher level protocol) has indicated completion by a sufficient number of receivers.

For the reasons mentioned above, this document does not pose any restriction on packet sizes. However, network efficiency considerations recommend that the sender uses an as large as possible packet payload size, but in such a way that packets do not exceed the network's maximum transmission unit size (MTU), or when fragmentation coupled with packet loss might introduce severe inefficiency in the transmission.

It is recommended that all packets have the same or very similar sizes, as this can have a severe impact on the effectiveness of congestion control schemes such as the ones described in [[VIC1998](#)], [[BYE2000](#)], and [[LUB2002](#)]. A sender of packets using LCT MUST implement the sender-side part of one of the congestion control schemes that is in accordance with [[RFC2357](#)] using the Congestion Control Information field provided in the LCT header, and the corresponding receiver congestion control scheme is to be communicated out-of-band and MUST be implemented by any receivers

participating in the session.

6.2. Receiver Operation

Receivers can operate differently depending on the delivery service model. For example, for an on demand service model, receivers may join a session, obtain the necessary packets to reproduce the object, and then leave the session. As another example, for a streaming service model, a receiver may be continuously joined to a set of LCT channels to download all objects in a session.

To be able to participate in a session, a receiver **MUST** obtain the relevant session description information as listed in [Section 6.1](#).

If packet authentication information is present in an LCT header, it **SHOULD** be used as specified in [Section 5.2](#). To be able to be a receiver in a session, the receiver **MUST** be able to process the LCT header. The receiver **MUST** be able to discard, forward, store or process the other headers and the packet payload. If a receiver is not able to process a LCT header, it **MUST** drop from the session.

To be able to participate in a session, a receiver **MUST** implement the congestion control protocol specified in the session description using the Congestion Control Information field provided in the LCT header. If a receiver is not able to implement the congestion control protocol used in the session, it **MUST NOT** join the session. When the session is transmitted on multiple LCT channels, receivers **MUST** initially join channels according to the specified startup behavior of the congestion control protocol. For a multiple rate congestion control protocol that uses multiple channels, this typically means that a receiver will initially join only a minimal set of LCT channels, possibly a single one, that in aggregate are carrying packets at a low rate. This rule has the purpose of preventing receivers from starting at high data rates.

Several objects can be carried either sequentially or concurrently within the same LCT session. In this case, each object is identified by a unique TOI. Note that even if a server stops sending packets for an old object before starting to transmit packets for a new object, both the network and the underlying protocol layers can cause some reordering of packets, especially when sent over different LCT channels, and thus receivers **SHOULD NOT** assume that the reception of a packet for a new object means that there are no more packets in transit for the previous one, at least for some amount of time.

A receiver **MAY** be concurrently joined to multiple LCT sessions from one or more senders. The receiver **MUST** perform congestion control on each such LCT session. If the congestion control protocol allows the

receiver some flexibility in terms of its actions within a session then the receiver MAY make choices to optimize the packet flow performance across the multiple LCT sessions, as long as the receiver still adheres to the congestion control rules for each LCT session individually.

7. Requirements from Other Building Blocks

As described in [[RFC3048](#)], LCT is a building block that is intended to be used, in conjunction with other building blocks, to help specify a protocol instantiation. A congestion control building block that uses the Congestion Control information field within the LCT header **MUST** be used by any protocol instantiation that uses LCT, and other building blocks **MAY** also be used, such as a reliability building block.

The congestion control **MUST** be applied to the LCT session as an entity, i.e., over the aggregate of the traffic carried by all of the LCT channels associated with the LCT session. The Congestion Control Information field in the LCT header is an opaque field that is reserved to carry information related to congestion control. There **MAY** also be congestion control Header Extension fields that carry additional information related to congestion control.

The particular layered encoder and congestion control protocols used with LCT have an impact on the performance and applicability of LCT. For example, some layered encoders used for video and audio streams can produce a very limited number of layers, thus providing a very coarse control in the reception rate of packets by receivers in a session. When LCT is used for reliable data transfer, some FEC codecs are inherently limited in the size of the object they can encode, and for objects larger than this size the reception overhead on the receivers can grow substantially.

A more in-depth description of the use of FEC in Reliable Multicast Transport (RMT) protocols is given in [[RFC3453](#)]. Some of the FEC codecs that **MAY** be used in conjunction with LCT for reliable content delivery are specified in [[I-D.ietf-rmt-fec-bb-revised](#)]. The Codepoint field in the LCT header is an opaque field that can be used to carry information related to the encoding of the packet payload.

LCT also requires receivers to obtain a session description, as described in [Section 6.1](#). The session description could be in a form such as SDP as defined in [[RFC2327](#)], or XML metadata as defined in [[RFC3023](#)], or HTTP/Mime headers as defined in [[RFC2616](#)], and distributed with SAP as defined in [[RFC2974](#)], using HTTP, or in other ways. It is **RECOMMENDED** that an authentication protocol be used to deliver the session description to receivers to ensure the correct session description arrives.

It is **RECOMMENDED** that LCT implementors use some packet authentication scheme to protect the protocol from attacks. An example of a possibly suitable scheme is described in [[RIZ1997a](#)].

Some protocol instantiations that use LCT MAY use building blocks that require the generation of feedback from the receivers to the sender. However, the mechanism for doing this is outside the scope of LCT.

8. Security Considerations

LCT can be subject to denial-of-service attacks by attackers which try to confuse the congestion control mechanism, or send forged packets to the session which would prevent successful reconstruction or cause inaccurate reconstruction of large portions of an object by receivers. LCT is particularly affected by such an attack since many receivers may receive the same forged packet. It is therefore RECOMMENDED that an integrity check be made on received objects before delivery to an application, e.g., by appending an MD5 hash [[RFC1321](#)] to an object before it is sent and then computing the MD5 hash once the object is reconstructed to ensure it is the same as the sent object. Moreover, in order to obtain strong cryptographic integrity protection a digital signature verifiable by the receiver SHOULD be computed on top of such a hash value. It is also RECOMMENDED that protocol instantiations that use LCT implement some form of packet authentication such as TESLA [[PER2001](#)] to protect against such attacks. Finally, it is RECOMMENDED that Reverse Path Forwarding checks be enabled in all network routers and switches along the path from the sender to receivers to limit the possibility of a bad agent injecting forged packets into the multicast tree data path.

Another vulnerability of LCT is the potential of receivers obtaining an incorrect session description for the session. The consequences of this could be that legitimate receivers with the wrong session description are unable to correctly receive the session content, or that receivers inadvertently try to receive at a much higher rate than they are capable of, thereby disrupting traffic in portions of the network. To avoid these problems, it is RECOMMENDED that measures be taken to prevent receivers from accepting incorrect Session Descriptions, e.g., by using source authentication to ensure that receivers only accept legitimate Session Descriptions from authorized senders.

A receiver with an incorrect or corrupted implementation of the multiple rate congestion control building block may affect health of the network in the path between the sender and the receiver, and may also affect the reception rates of other receivers joined to the session. It is therefore RECOMMENDED that receivers be required to identify themselves as legitimate before they receive the Session Description needed to join the session. How receivers identify themselves as legitimate is outside the scope of this document.

9. IANA Considerations

9.1. Namespace declaration for LCT Header Extension Types

This document defines two name-spaces for registration of LCT Header Extensions Types named:

```
    ietf:rmt:lct:headerExtensionTypes:variableLength
and
    ietf:rmt:lct:headerExtensionTypes:fixedLength
```

The values that can be assigned within the "ietf:rmt:lct:headerExtensionTypes:variableLength" name-space are numeric indexes in the range [0, 127] inclusive. The values that can be assigned within the "ietf:rmt:lct:headerExtensionTypes:fixedLength" name-space are numeric indexes in the range [128, 255] inclusive. Assignment requests for both namespaces shall be granted on a "IETF Consensus" basis as defined in [[RFC2434](#)].

Note that the previous Experimental version of this specification reserved values in the ranges [64, 127] and [192, 255] for Protocol Instantiation-specific LCT Header Extensions. In the interests of simplification and since there were no overlapping allocations of these LCT Header Extension Type values by Protocol Inatntiations, this document specifies a single flat space for LCT Header Extension Types. Values in the range [0,63] and [128,191] SHOULD be used for Header Extensions which are expected to have broad applicability over all users of the LCT Building Block. Values outside this range SHOULD be used for Header Extensions with more limited applicability. However, these Header Extension Type values are global in scope and are NOT Protocol-Instantiation specific.

9.2. LCT Header Extension Type registration

This document registers two values in the namespace "ietf:rmt:lct:headerExtensionTypes:variableLength" as follows:

Value	Name	Reference
0	EXT_NOP	This specification
1	EXT_AUTH	This specification

10. Acknowledgments

Thanks to Vincent Roca and Roger Kermode for detailed comments and contributions to this document. Thanks also to Bruce Lueckenhoff, Hayder Radha and Justin Chapweske for detailed comments on this document.

11. Change Log

11.1. [RFC3451](#) to [draft-ietf-rmt-bb-lct-revised-00](#)

Update all references to the obsoleted [RFC 2068](#) to [RFC 2616](#)

Removed the 'Statement of Intent' from the introduction

The statement of intent was meant to clarify the "Experimental" status of [RFC3451](#). It does not apply to this draft that is intended for "Standard Track" submission.

11.2. [draft-ietf-rmt-bb-lct-revised-00](#) to -01

- o Inclusion of material from ALC which is applicable in the more general LCT context
- o Creation of an IANA registry for LCT Header Extension
- o Allocation of the 2 'reserved' bits in the LCT header as "Protocol Specific Indication" - usage to be defined by protocol instantiations
- o Proposal added to redefine the Sender Current Time as an NTP timestamp. Decision on this is tbd as it is non-backwards compatible.
- o Update references from Experimental RFCs to new drafts
- o Split references into normative and informative

12. References

12.1. Normative References

- [I-D.ietf-rmt-fec-bb-revised]
Watson, M., "Forward Error Correction (FEC) Building Block", [draft-ietf-rmt-fec-bb-revised-02](#) (work in progress), October 2005.
- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, [RFC 768](#), August 1980.
- [RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, [RFC 1112](#), August 1989.
- [RFC1305] Mills, D., "Network Time Protocol (Version 3) Specification, Implementation", [RFC 1305](#), March 1992.
- [RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", [BCP 9](#), [RFC 2026](#), October 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", [BCP 26](#), [RFC 2434](#), October 1998.

12.2. Informative References

- [BYE1998] Byers, J., Luby, M., Mitzenmacher, M., and A. Rege, "Fountain Approach to Reliable Distribution of Bulk Data", Proceedings ACM SIGCOMM'98, Vancouver, Canada , September 1998.
- [BYE2000] Byers, J., Frumin, M., Horn, G., Luby, M., Mitzenmacher, M., Rotter, A., and W. Shaver, "FLID-DL: Congestion Control for Layered Multicast", Proceedings of Second International Workshop on Networked Group Communications (NGC 2000), Palo Alto, CA , November 2000.
- [GEM2000] Gemmell, J., Schooler, E., and J. Gray, "Fcast Multicast File Distribution", IEEE Network, Vol. 14, No. 1, pp. 58-68 , January 2000.
- [HOL2001] Holbrook, H., "A Channel Model for Multicast", Ph.D. Dissertation, Stanford University, Department of Computer Science, Stanford, CA , August 2001.

- [LUB2002] Luby, M., Goyal, V., Skaria, S., and G. Horn, "Wave and Equation Based Rate Control using Multicast Round-trip Time", Proceedings of ACM SIGCOMM 2002, Pittsburgh PA , August 2002.
- [PER2001] Perrig, A., Canetti, R., Song, D., and J. Tygar, "Efficient and Secure Source Authentication for Multicast", Network and Distributed System Security Symposium, NDSS 2001, pp. 35-46 , February 2001.
- [RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", [RFC 1321](#), April 1992.
- [RFC1889] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", [RFC 1889](#), January 1996.
- [RFC2327] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", [RFC 2327](#), April 1998.
- [RFC2357] Mankin, A., Romanov, A., Bradner, S., and V. Paxson, "IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols", [RFC 2357](#), June 1998.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", [RFC 2616](#), June 1999.
- [RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", [RFC 2974](#), October 2000.
- [RFC3023] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", [RFC 3023](#), January 2001.
- [RFC3048] Whetten, B., Vicisano, L., Kermode, R., Handley, M., Floyd, S., and M. Luby, "Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer", [RFC 3048](#), January 2001.
- [RFC3269] Kermode, R. and L. Vicisano, "Author Guidelines for Reliable Multicast Transport (RMT) Building Blocks and Protocol Instantiation documents", [RFC 3269](#), April 2002.
- [RFC3453] Luby, M., Vicisano, L., Gemmell, J., Rizzo, L., Handley, M., and J. Crowcroft, "The Use of Forward Error Correction (FEC) in Reliable Multicast", [RFC 3453](#), December 2002.
- [RIZ1997] Rizzo, L., "Effective Erasure Codes for Reliable Computer

Communication Protocols", ACM SIGCOMM Computer
Communication Review, Vol.27, No.2, pp.24-36 , April 1997.

[RIZ1997a]

Rizzo, L., "Effective Erasure Codes for Reliable Computer
Communication Protocols", ACM SIGCOMM Computer
Communication Review, Vol.27, No.2, pp.24-36 , April 1997.

[RIZ1997b]

Rizzo, L. and L. Vicisano, "Reliable Multicast Data
Distribution protocol based on software FEC techniques",
Proceedings of the Fourth IEEE Workshop on the
Architecture and Implementation of High Performance
Communication Systems, HPCS'97, Chalkidiki Greece ,
June 1997.

[RIZ2000]

Rizzo, L., "PGMCC: A TCP-friendly single-rate multicast
congestion control scheme", Proceedings of SIGCOMM 2000,
Stockholm Sweden , August 2000.

[VIC1998]

Vicisano, L., Rizzo, L., and J. Crowcroft, "TCP-like
Congestion Control for Layered Multicast Data Transfer",
IEEE Infocom'98, San Francisco, CA , March 1998.

Authors' Addresses

Michael Luby
Digital Fountain
39141 Civic Center Dr.
Suite 300
Fremont, CA 94538
US

Email: luby@digitalfountain.com

Mark Watson
Digital Fountain
39141 Civic Center Dr.
Suite 300
Fremont, CA 94538
US

Email: mark@digitalfountain.com

Jim Gemmell
Microsoft Research
455 Market St. #1690
San Francisco, CA 94105
US

Email: jgemmell@microsoft.com

Lorenzo Vicisano
Cisco Systems, Inc.
510 McCarthy Blvd.
Milpitas, CA 95035
US

Email: lorenzo@cisco.com

Luigi Rizzo
Univ. di Pisa
Dip. Ing. dell'Informazione,
via Diotisalvi 2
Pisa, PI 56126
Italy

Email: luigi@iet.unipi.it

Mark Handley
University College London
Dept. of Computer Science
Gower Street
London, WC1E 6BT
United Kingdom

Email: M.Handley@cs.ucl.ac.uk

Jon Crowcroft
University of Cambridge
Computer Laboratory
William Gates Building
J J Thomson Avenue
Cambridge, CB3 0FD
United Kingdom

Email: Jon.Crowcroft@cl.cam.ac.uk

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2005). This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

Acknowledgment

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

