Internet Engineering Task Force
INTERNET-DRAFT
draft-ietf-rmt-bb-lct-04.ps

RMT WG
M.Luby/Digital Fountain
J.Gemmell/Microsoft
L.Vicisano/Cisco
L.Rizzo/ACIRI and Univ. Pisa
M.Handley/ACIRI
J. Crowcroft/UCL
8 February 2002
Expires: August 2002

Layered Coding Transport building block

Status of this Document

This document is a product of the IETF RMT WG. Comments should be addressed to the authors, or the WG's mailing list at rmt@lbl.gov.

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 muliple rate delivery to receivers and is also compatible with coding techniques that provide reliable delivery of content.

Table of Contents

1. Introduction

This document describes a massively scalable reliable content delivery and stream delivery transport level support, Layered Coding Transport (LCT), for multiple rate content delivery primarily designed to be used with the IP multicast network service, but may also be used with other basic underlying network or transport services such as unicast UDP. LCT supports both reliable and unreliable delivery.

This document describes a building block as defined in RFC3048 [26]. 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].

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 but does not 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, and thus it must 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 thus 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. Thus, 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 other 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 continue 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 [16] 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 protcol 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 [25] and [3]. A possible single rate congestion control protocol is described in [22].

Layered coding refers to the ability to produce a coded stream of packets that can be partioned 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.

Layered coding can be naturally combined with multiple rate congestion control. For example, the sender could send the packets for each layer to a separate channel associated with the session, and then receivers dynamically join and leave channels to adjust their reception rate according to the multiple rate congestion control protocol.

Layered coding can also be combined with single rate congestion control. For example, the sender could dynamically vary how many layers are sent to the channel associated with the session as the rate of transmission varies according to the single rate congestion control protocol.

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 [23], [21], [7], [25] and [4]. 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 [14].

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 [16]. 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 that are used for some period of time to carry packets containing LCT headers pertaining 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 of substantial length, i.e., objects or streams that range in length from hundreds of kilobytes to many gigabytes, and whose transfer time is in 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, whereas 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 [15] 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 [25] and [3]. 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 RFC768 [19], 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, 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 [5] 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 [5] and the Source-Specific Multicast (SSM) model as defined in [10]. 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.

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.

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, and dynamically adapt the number of LCT channels they subscribe to (and the reception quality) 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. The 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 [25] and [3]. 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 described below is the default format, and this is the format that is recommended for use by protocol instantiations to ensure a uniform format across different protocol instantiations. Other LCT header formats MAY be used by protocol instantiations, but if the default LCT header format is not used by a protocol instantiation that uses LCT, then the protocol instantiation MUST specify the lengths and positions within the LCT header it uses of all fields described in the default LCT header.

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. Default LCT header format

The default 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 by 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.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   V   | C | r |S| O |H|T|R|A|B|   HDR_LEN     | Codepoint (CP)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Congestion Control Information (CCI, length = 32*(C+1) bits)  |
|                           ...                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Transport Session Identifier (TSI, length = 32*S+16*H bits)  |
|                           ...                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Transport Object Identifier (TOI, length = 32*O+16*H bits)  |
|                           ...                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Sender Current Time (SCT, if T = 1)             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Expected Residual Time (ERT, if R = 1)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Header Extensions (if applicable)              |
|                           ...                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1 - Default LCT header format

The function and length of each field in the default LCT header is the following. Fields marked as "1" mean that the corresponding bits MUST be set to "1" by the sender. Fields marked as "r" or "0" mean that the corresponding bits MUST be set to "0" by the sender.

LCT version number (V): 4 bits

Indicates the LCT version number. The LCT version number for this specification is 1.

Congestion control flag (C): 2 bits

C=0 indicates the Congestion Control Information (CCI) field is 32-bits in length.
C=1 indicates the CCI field is 64-bits in length.
C=2 indicates the CCI field is 96-bits in length.
C=3 indicates the CCI field is 128-bits in length.

Reserved (r): 2 bits

Reserved for future use. A sender MUST set these bits to zero and a receiver MUST ignore these 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 [24].

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 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 ouptut 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 at the time this packet was transmitted, measured in units of 1ms and computed modulo 2^32 units from the start of the session. 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.

Protocol instantiation MAY override this default behavior for PI-specific extensions (see below).

There are two formats for Header Extension fields, as depicted below. 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.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  HET (<=127)  |       HEL     |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
.                                                               .
.              Header Extension Content (HEC)                   .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  HET (>=128)  |       Header Extension Content (HEC)          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5 - 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.

Header Extensions are further divided between general LCT extensions and Protocol Instantiation specific extensions (PI-specific). General LCT extensions have HET in the ranges 0:63 and 128:191 inclusive. PI-specific extensions have HET in the ranges 64:127 and 192:255 inclusive.

General LCT extensions are intended to allow the 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.

PI-specific extensions are reserved for PI-specific use with semantic and default parsing actions defined by the PI.

The following general LCT Header Extension types are defined:

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

EXT_AUTH=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:

- The sender IP address;
- The number of LCT channels;
- The addresses and port numbers used for each LCT channel;
- 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 [8], or XML metadata as defined in RFC3023 [17], or HTTP/Mime headers as defined in RFC2068 [6], etc. It might be carried in a session announcement protocol such as SAP as defined in RFC2974 [9], 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 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 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 [25] and [3]. 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 [16] 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 encoding symbols 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 [26], 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. Some possible schemes are specified in [25] and [3]. 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 [14]. Some of the FEC codecs that MAY be used in conjunction with LCT for reliable content delivery are specified in [15]. 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 [8], or XML metadata as defined in RFC3023 [17], or HTTP/Mime headers as defined in RFC2068 [6], and distributed with SAP as defined in RFC2974 [9], using HTTP, or in other ways. It is RECOMMENDED that an authentication protocol such as IPSEC [11] 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 [18].

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 the 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 content before delivery to an application, e.g., by appending an MD5 hash [20] to the content before it is sent and then computing the MD5 hash once the content is reconstructed to ensure it is the same as the sent content. 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 [18] to protect against such attacks. Furthermore, 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 the receiver authenticate the session description, for example by using either the ESP (with enabled authentication service) [13] or AH [12] protocols of IPSEC [11] to ensure the authenticity of the session description.

A receiver with an incorrect or corrupted implementation of the 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 identify themselves as legitimate before they receive the session description needed to join the session.

9. IANA Considerations

No information in this specification is subject to IANA registration.

Building blocks used in conjunction with LCT MAY introduce additional IANA considerations.

10. Intellectual Property Issues

No specific reliability protocol or congestion control protocol is specified or referenced as mandatory in this document.

LCT MAY be used with congestion control protocols and other protocols, such as reliability protocols, which are proprietary or have pending or granted patents.

11. 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.

12. References

[1] Bradner, S., "The Internet Standards Process -- Revision 3", RFC2026, October 1996.

[2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC2119, March 1997.

[3] Byers, J.W., Frumin, M., Horn, G., Luby, M., Mitzenmacher, M., Roetter, A., and Shaver, W., "FLID-DL: Congestion Control for Layered Multicast", Proceedings of Second International Workshop on Networked Group Communications (NGC 2000), Palo Alto, CA, November 2000.

[4] Byers, J.W., Luby, M., Mitzenmacher, M., and Rege, A., "A Digital Fountain Approach to Reliable Distribution of Bulk Data", Proceedings ACM SIGCOMM '98, Vancouver, Canada, September 1998.

[5] Deering, S., "Host Extensions for IP Multicasting", RFC1112, August 1989.

[6] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Berners-Lee, T., "Hypertext Transfer Protocol -- HTTP/1.1", RFC2068, January 1997.

[7] Gemmell, J., Schooler, E., and Gray, J., "Fcast Multicast File Distribution", IEEE Network, Vol. 14, No. 1, pp. 58-68, January 2000.

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

[9] Handley, M., Perkins, C., Whelan, E., "Session Announcement Protocol", RFC2974, October 2000.

[10] Holbrook, H. W., "A Channel Model for Multicast", Ph.D. Dissertation, Stanford University, Department of Computer Science, Stanford, California, August 2001.

[11] Kent, S., Atkinson, R., "Security Architecture for the Internet Protocol", RFC2401, November 1998.

[12] Kent, S., Atkinson, R., "IP Authentication Header", RFC2406, November 1998.

[13] Kent, S., Atkinson, R., "IP Encapsulating Security Payload (ESP)", RFC2406, November 1998.

[14] Luby, M., Gemmell, Vicisano, L., J., Rizzo, L., Handley, M., Crowcroft, J., "The use of Forward Error Correction in Reliable Multicast", Internet Draft draft-ietf-rmt-info-fec-02.txt, January 2002, a work in progress.

[15] Luby, M., Gemmell, J., Vicisano, L., Rizzo, L., Handley, M., Crowcroft, J., "Forward Error Correction building block", Internet Draft draft-ietf-rmt-bb-fec-06.txt, January 2002.

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

[17] Murata, M., St.Laurent, S., Kohn, D., "XML Media Types", RFC3023, January 2001.

[18] Perrig, A., Canetti, R., Song, D., Tygar, J.D., "Efficient and Secure Source Authentication for Multicast", Network and Distributed System Security Symposium, NDSS 2001, pp. 35-46, February 2001.

[19] Postel, J., "User Datagram Protocol", RFC768, August 1980.

[20] Rivest, R., "The MD5 Message-Digest Algorithm", RFC1321, April 1992.

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

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

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

[24] Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V., "RTP: A Transport Protocol for Real-Time Applications", RFC1889, January 1996.

[25] Vicisano, L., Rizzo, L., Crowcroft, J., "TCP-like Congestion Control for Layered Multicast Data Transfer", IEEE Infocom '98, San Francisco, CA, Mar.28-Apr.1 1998.

[26] Whetten, B., Vicisano, L., Kermode, R., Handley, M., Floyd, S., Luby, M., "Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer", RFC3048, January 2001.

13. Authors' Addresses

Michael Luby
luby@digitalfountain.com
Digital Fountain
39141 Civic Center Drive
Suite 300
Fremont, CA, USA, 94538

Jim Gemmell
jgemmell@microsoft.com
Microsoft Research
301 Howard St., #830
San Francisco, CA, USA, 94105

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

Luigi Rizzo
luigi@iet.unipi.it
ACIRI/ICSI,

1947 Center St, Berkeley, CA, USA, 94704
and
Dip. Ing. dell'Informazione,
Univ. di Pisa
via Diotisalvi 2, 56126 Pisa, Italy

Mark Handley
mjh@aciri.org
ACIRI,
1947 Center St,
Berkeley, CA, USA, 94704

Jon Crowcroft
J.Crowcroft@cs.ucl.ac.uk
Department of Computer Science
University College London
Gower Street,
London WC1E 6BT, UK

14. Full Copyright Statement