Internet Engineering Task Force
INTERNET-DRAFT
draft-ietf-rmt-pi-alc-05.ps

RMT WG
M.Luby/Digital Fountain
J.Gemmell/Microsoft
L.Vicisano/Cisco
L.Rizzo/ACIRI and Univ. Pisa
J. Crowcroft/UCL

8 February 2002
Expires: August 2002

Asynchronous Layered Coding protocol instantiation

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

This document describes the Asynchronous Layered Coding protocol, a massively scalable reliable content delivery protocol, hereafter referred to as ALC. ALC combines the LCT [11], WEBRC [12] and FEC [10] building blocks to provide congestion controlled reliable asynchronous delivery of content to an unlimited number of concurrent receivers from a single sender.

Table of Contents

1. Applicability Statement

This document describes a massively scalable reliable content delivery protocol, Asynchronous Layered Coding (ALC), for multiple rate congestion controlled reliable content delivery. The protocol is specifically designed to provide massive scalability using IP multicast as the underlying network service. Massive scalability in this context means the number of concurrent receivers for an object is potentially in the millions, the size of an object to be delivered ranges from hundreds of kilobytes to hundreds of gigabytes, each receiver can initiate reception of an object asynchronously, the reception rate of each receiver in the session is the maximum fair bandwidth available between that receiver and the sender, and all of this can be supported using a single sender.

Because ALC is focused on reliable content delivery, the goal is to delivery an object as quickly as possible to each receiver while at the same time remaining network friendly to competing traffic. Thus, the congestion control used strives to maximize use of available bandwidth between receivers and the sender while at the same time backing off aggressively in the face of competing traffic.

The sender side of ALC consists of generating packets based on objects to be delivered within the session and sending the appropriately formatted packets at the appropriate rates to the channels associated with the session. The receiver side of ALC consists of joining appropriate channels associated with the session, performing congestion and flow control by adjusting the set of joined channels associated with the session in response to detected congestion, and using the packets to reliably reconstruct objects. All information flow in an ALC session is in the form of data packets sent by a single sender to channels that receivers join to receive data.

ALC does specify the session description needed by receivers before they join a session, but the mechanisms by which receivers obtain this required information is outside the scope of ALC. An application that uses ALC may require that receivers report statistics on their reception experience back to the sender, but the mechanisms by which receivers report back statistics is outside the scope of ALC. In general, ALC is designed to be a minimal protcol instantiation that provides reliable content delivery without unnecessary limitations to the scalability of the basic protocol.

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].

1.1. Delivery service models

ALC can support several different reliable content delivery service models. Some examples are briefly described here.

Push service model.

A push model is a sender initiated concurrent delivery of objects to a selected set of receivers. 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 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 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. In this example, the sender uses ALC to generate packets based on the object and send packets to channels associated with the session, and the receivers use ALC to receive packets from the session and reconstruct the object.

There are several features ALC provides to support the push model. For example, the sender can optionally include an Expected Residual Time (ERT) for the session in each packet header. This can be used by receivers to determine if there is enough time remaining in the session to successfully receive enough 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 session 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 session time in each subsequent packet header. 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.

The push model is particularly attractive in satellite networks and wireless networks. In these environments a session may include one channel and a sender may send packets at a fixed rate to this channel, but sending at a fixed rate without congestion control is outside the scope of ALC.

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 a single object. For example a popular software update might be transmitted using ALC 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 obtain a session description for example by contacting a web server.

Other service models.

There may be other reliable content delivery service models that can be supported by ALC. The description of the potential applications, the appropriate delivery service model, and the additional mechanisms to support such functionalities when combined with ALC is beyond the scope of this document.

1.2. Scalability

Massive scalability is a primary design goal for ALC. IP multicast is inherently massively scalable, but the best effort service that it provides does not provide session management functionality, congestion control or reliability. ALC provides all of this on top of IP multicast without sacrificing any of the inherent scalability of IP multicast. ALC has the following properties:

- o To each receiver, it appears as if though there is a dedicated session from the sender to the receiver, where the reception rate adjusts to congestion along the path from sender to receiver.
- o To the sender, there is no difference in load or outgoing rate if one receiver is joined to the session or a million (or any number of) receivers are joined to the session, independent of when the receivers join and leave.
- o No feedback packets are required from receivers to the sender.
- o Almost all packets in the session that pass through a bottleneck link are utilized by downstream receivers, and the session shares the link with competing flows fairly in proportion to their utility.

Thus, ALC provides a massively scalable content delivery transport that is network friendly.

ALC intentionally omits any application specific features that could potentially limit its scalability. By doing so, ALC provides a minimal protocol that is massively scalable. Applications may be built on top of ALC to provide additional features that may limit the scalability of the application. Such applications are outside the scope of ALC.

1.3. Environmental Requirements and Considerations

All of the environmental requirements and considerations that apply to the LCT [11], FEC [10], and WEBRC [12] building blocks and to any additional building blocks that ALC uses also apply to ALC.

ALC requires connectivity between a sender and receivers, but does not require connectivity from receivers to a sender. ALC 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 ALC is unlimited. However, ALC requires receivers to obtain the session description out-of-band before joining a session and some implementations of this may limit scalability.

If a receiver is joined to multiple ALC sessions then the receiver MUST be able to uniquely identify and demultiplex packets to the correct receiver. The TSI is scoped by the IP address of the sender, and the IP address of the sender together with the TSI uniquely identify the session. Thus, the demultiplexing MUST be done on the basis of the IP address of the sender and the TSI of the session from that sender.

ALC is presumed to be used with an underlying IP multicast 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. 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 []. ALC 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 an ALC 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 an ALC channel address coincides with the SSM channel address.

A sender can locally allocate unique SSM channel addresses, and this makes allocation of ALC channel addresses easy with SSM. To allocate ALC 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 ALC channel addresses more difficult with ASM.

ALC channels and SSM channels coincide, and thus the receiver will only receive packets sent to the requested ALC channel. With ASM, the receiver joins an ALC 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.

Other issues specific to ALC with respect to ASM is the way WEBRC interacts with ASM. WEBRC uses the measured difference in time between when a join to a channel is sent and when the first packet from the channel arrives in determing the receiver reception rate. WEBRC also uses packet sequence numbers per channel to measure losses, and this is also used to determine the receiver reception rate. These features raise two concerns with respect ASM: The time difference between when the join to a channel is sent and when the first packet arrives can be signifcant due to the use of Rendezvous Points (RPs) and the MSDP protocol, and packets can be lost in the switch over from the (*,G) join to the RP and the (S,G) join directly to the sender. Both of these issues could potentially substantially degrade the reception rate of receivers. To ameliorate these concerns, it is RECOMMENDED that the RP be as close to the sender as possible. SSM does not share these same concerns.

Some networks are not amenable to some congestion control protocols that could be used with ALC. 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.

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

2. Architecture Definition

ALC uses the LCT building block [11] to provide in-band session management functionality. ALC uses the WEBRC building block [12] to provide multiple rate congestion control that is feedback free. Receivers adjust their reception rates individually by joining and leaving channels associated with the session. ALC uses the FEC building block [10] to provide reliability. The sender generates encoding symbols based on the object to be delivered using FEC codes and sends them in packets to channels associated with the session. Receivers simply wait for enough packets to arrive in order to reliably reconstruct the object. Thus, there is no request for retransmission of individual packets from receivers that miss packets in order to assure reliable reception of an object, and the packets and their rate of transmission out of the sender can be independent of the number and the individual reception experiences of the receivers.

The definition of a session for ALC is the same as it is for LCT. An ALC 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. WEBRC congestion control is performed over the aggregate of packets sent to channels belonging to a session.

ALC is a protocol instantiation as defined in RFC3048 [18]. This document describes version 1 of ALC which MUST use version 1 of LCT described in [11]. Like LCT, ALC is designed to be used with the IP multicast network service. ALC could be used as the basis for designing a protocol that uses a different underlying network service such as unicast UDP, but the design of such a protocol is outside the scope of this document. This specification defines ALC as payload of the UDP transport protocol [16] that supports IP multicast delivery of packets. Future versions of this specification, or companion documents may extend ALC to use the IP network layer service directly.

An ALC packet header immediately follows the UDP header and consists of the default LCT header that is described in [11] followed by the FEC Payload ID that is described in [10]. The Congestion Control Information field within the LCT header carries the required WEBRC Congestion Control Information that is described in [12]. The packet payload that follows the ALC packet header consists of encoding symbols that are identified by the FEC Payload ID as described in [10].

The out-of-band information required by each receiver before participating in an ALC session consists of a session description that includes all the out-of-band information required for the LCT, FEC and WEBRC building blocks. The means for acquiring this out-of-band information is outside the scope of ALC.

2.1. LCT building block

LCT requires receivers to be able to uniquely identify and demultiplex packets associated with an LCT session, and ALC inherits and strengthens this requirement. A Transport Session Identifier (TSI) MUST be associated with each session and MUST be carried in the LCT header of each ALC packet. The TSI is scoped by the sender IP address, and the (sender IP address, TSI) pair MUST uniquely identify the session.

The LCT header contains a Congestion Control Information (CCI) field that MUST be used to carry the WEBRC Congestion Control Information. There is a field in the LCT header that specifies the length of the CCI field, and as described in WEBRC this length uniquely determines the format of the CCI field.

The LCT header contains a Codepoint field that MUST be used to carry the FEC Encoding ID required by the FEC building block to identify the format of the FEC Payload ID.

Because the FEC Encoding ID carried in the Codepoint field of the LCT header specifies the format of the FEC Payload ID, and because and the length of the CCI field determines its format and because the remainder of the LCT header self-describes its own format, the overall ALC packet header format is self-describing.

If more than one object is to be delivered within a session then the Transmission Object ID (TOI) in the LCT header MUST be used to identify which packets are to be associated with which objects. Each TOI MUST be unique within the session and SHOULD be globally unique across all sessions.

The default LCT header from version 1 of the LCT building block [11] MUST be used.

2.2. WEBRC building block

Implementors of ALC MUST implement WEBRC [12], which is a multiple rate feedback-free congestion control building block that is in accordance to RFC2357 [13]. Congestion control MUST be applied to all packets within a session independently of which information about which object is carried in each packet. WEBRC is chosen because of its suitability as a massively scalable congestion control protocol for reliable content delivery. The WEBRC Congestion Control Information MUST be carried in the Congestion Control Information (CCI) field of the LCT header. The length of the CCI field determines its format as described in WEBRC.

When using WEBRC a sender sends packets in the session to several channels at potentially different rates. Then, individual receivers adjust their reception rate within a session by adjusting which set of channels they are joined to at each point in time depending on the available bandwidth between the receiver and the sender, but independent of other receivers. WEBRC is described in detail in [12].

2.3. FEC building block

The FEC building block [10] provides reliable object delivery within an ALC session. Each object sent in the session is independently encoded using FEC codes as described in [9], which provide a more in-depth description of the use of FEC codes in reliable content delivery protocols. All packets in an ALC session MUST contain an FEC Payload ID in a format that is compliant with the FEC building block [10]. The FEC Payload ID uniquely identifies the encoding symbols that constitute the payload of each packet, and the receiver MUST use the FEC Payload ID to determine how the encoding symbols carried in the payload of the packet were generated from the object as described in the FEC building block. The FEC Encoding ID that specifies the FEC Payload ID format MUST be carried in the Codepoint field of the LCT header.

If more than one object is to be delivered in a session then each LCT header contains a TOI that identifies which object within the session each packet contains encoding symbols for. In this case the receiver MUST use the TOI to associate received encoding symbols with objects,

2.4. Session description

The session description that a receiver is REQUIRED to obtain before joining an ALC session MUST contain the following information:

- o The sender IP address;
- o The number of channels in the session;
- o The addresses and port numbers used for each channel in the session;
- o The Transport Session ID (TSI) to be used for the session;
- o An indication of whether or not the session carries packets for more than one object;
- o If the session carries packets for more than one object, the set of Transport Object IDs (TOIs) for the objects in the session.
- o The total length of the objects in the session in bytes.
- o The FEC Encoding ID.
- o If an Under-Specified FEC Encoding ID is used then the FEC Encoding Name associated with the FEC Encoding ID.

- o The additional required FEC Object Transmission Information for the FEC Encoding ID as prescribed in the FEC building block [10]. For example, when the FEC Encoding ID is 128, the required FEC Object Transmission Information is the number of source blocks that the object is partitioned into and the length of each source block in bytes.
- o Enough information to determine the packet authentication scheme being used if it is being used.

How this out-of-band information is communicated is outside the scope of this document and in particular some of it MAY be implicit based on the implementation. As an example the source block lengths may be derived by a fixed algorithm from the object length. As another example, it may be that all source blocks are the same length and this is what is passed out-of-band to the receiver. As another example, it could be that the full sized source block length is provided and this is the length used for all but the last source block, which is calculated based on the full source block length and the object length. As another example, it could be that the same FEC Encoding ID and FEC Encoding Name is always used for a particular application and thus the FEC Encoding ID and FEC Encoding Name is implicitly defined.

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

- o The data rates used for each channel;
- o The length of the packet payload;
- 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;

The session description could be in a form such as SDP as defined in RFC2327 [4], or XML metadata as defined in RFC3023 [14], or HTTP/Mime headers as defined in RFC2068 [3], etc. It might be carried in a session announcement protocol such as SAP as defined in RFC2974 [5], 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.

If multiple objects are carried in the same session, then the mapping between the objects and the TOIs MUST be provided as part of the session description. This mapping MAY be implicit, for example it could be agreed out-of-band that the objects carried within the session are to be numbered consecutively.

2.5. Packet authentication building block

It is RECOMMENDED that implementors of ALC use some packet authentication scheme to protect the protocol from attacks. An example of a possibly suitable scheme is described in [15]. Packet authentication in ALC, if used, is to be integrated through the header extension support for packet authentication provided in the LCT building block.

3. Conformance Statement

This Protocol Instantiation document, in conjunction with the LCT [11], FEC [10] and WEBRC [12] building blocks completely specifies a working reliable multicast transport protocol that conforms to the requirements described in RFC2357 [13].

4. Functionality Definition

This section describes the format and functionality of the data packets carried in an ALC session as well as the sender and receiver operations for a session.

4.1. Packet format used by ALC

The packet format used by ALC is the UDP header followed by the default LCT header followed by the FEC Payload ID followed by the packet payload. The default LCT header is described in the LCT building block [11] and the FEC Payload ID is described in the FEC building block [10]. The Congestion Control Information field in the LCT header contains the REQUIRED WEBRC Congestion Control Information described in [12]. The packet payload contains encoding symbols generated from an object. If more than one object is carried in the session then the Transmission Object ID (TOI) within the LCT header MUST be used to identify which object the encoding symbols are generated from. Within the scope of an object, encoding symbols carried in the payload of the packet are identified by the FEC Payload ID as described in the FEC building block.

The version number of ALC specified in this document is 1. This coincides with version 1 of the LCT building block [11] used in this specification. The LCT version number field should be interpreted as the ALC version number field.

The overall ALC packet format is depicted in Fig. 1. The packet is an IP packet, either IPv4 or IPv6, and the IP header precedes the UDP header. The ALC packet format has no dependencies on the IP version number. The default LCT header MUST be used by ALC and this default is described in detail in the LCT building block [11].

```
  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
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                         UDP header                            |
 |                                                               |
 +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 |                     Default LCT header                        |
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                       FEC Payload ID                          |
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                     Encoding Symbol(s)                        |
 |                           ...                                 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fig. 1 - Overall ALC packet format

In some special cases an ALC sender may need to produce ALC packets that do not contain any payload. This may be required, for example, to signal the end of a session or to convey congestion control information. These data-less packets do not contain the FEC Payload ID either, but only the LCT header fields. The total datagram length, conveyed by outer protocol headers (e.g. the IP or UDP header), enables receivers to detect the absence of the ALC payload and FEC Payload ID.

A detailed example of an ALC packet starting with the LCT header is shown in Fig. 2. In the example, the LCT header is the first 5 32-bit words, the FEC Payload ID is the next 2 32-bit words, and the remainder of the packet is the payload.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   1   | 0 | 0 |1| 1 |0|1|0|0|0|       5       |      128      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      CTSI     | Channel Number|    Packet Sequence Number     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Transport Session Identifier (TSI, length = 32 bits)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Transport Object Identifier (TOI, length = 32 bits)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Sender Current Time                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Source Block Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Encoding Symbol ID                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Encoding Symbol(s)                         |
|                          ...                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fig. 2 - Detailed ALC packet format

The LCT portion of the overall ALC packet header is of variable size, which is specified by a length field in the third byte of the 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 function and length and particular setting of the value in the example of each field in the header is the following, described in the order of their appearance in the header.

ALC version number (V): 4 bits

Indicates the ALC version number.
The ALC version number for this specification is 1 as shown. This is also the LCT version number.

Congestion control flag (C): 2 bits

Two Congestion Control Information (CCI) field formats are provided by WEBRC, a 32-bit format and a 64-bit format.
C=0 indicates the 32-bit CCI field format is to be used.
C=1 indicates the 64-bit CCI field format is to be used.
In the example C=0 indicates that the 32-bit format is to be used.

Reserved (r): 2 bits

Reserved for future use. A sender MUST set these bits to zero and a receiver MUST ignore these bits.
As required, these bits are set to 0 in the example.

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. For ALC the length of the TSI field is REQUIRED to be non-zero. This implies that the setting S=0 and H=0 MUST NOT be used.
In the example S=1 and H=0, and thus the TSI is 32-bits in length.

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. If more than one object is to be delivered in the session then the TOI MUST be used, in which case the setting O=0 and H=0 MUST NOT be used.
In the example O=1 and H=0, and thus the TOI is 32-bits in length.

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.
In the example H=0 which indicates that both TSI and TOI are both multiples of 32-bits in length.

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.
In the example T=1, which indicates that the SCT is carried in this packet.

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.
In the example R=0, which indicates that the ERT is not carried in this packet.

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.
In the example A=0, and thus this packet does not indicate the close of 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.
In the example B=0, and thus this packet does not indicate the end of sending data packets for the object.

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.
In the example HDR_LEN=5 to indicate that the length of the LCT header portion of the overall ALC is 5 32-bit words.

Codepoint (CP): 8 bits

This field is used by ALC to carry the 8-bit FEC Encoding ID described in the FEC building block. The FEC encoding ID specifies the format and the length of the FEC Payload ID.
In the example CP=128, which is the FEC Encoding ID for the ‘‘Small Block, Large Block and Expandable FEC Codes" as described in the FEC building block [10]. The FEC Payload ID associated with this FEC Encoding ID is 64-bits in length.

Congestion Control Information (CCI): 32 or 64 bits

This is field contains the WEBRC Congestion Control Information. There are two formats provided by WEBRC for the CCI field, a 32-bit format and a 64-bit format. The value of the C field determines which format is used.
This field MUST be 32 bits if C=0.
This field MUST be 64 bits if C=1.
In the example, since C=0, the CCI is 32-bits in length. The corresponding WEBRC CCI field is partitioned into the following three fields and are to be used as described in WEBRC [12] to provide congestion control over all packets sent to an ALC session.

Current Time Slot Index (CTSI): 8 bits

CTSI indicates the index of the current time slot. The Current Time Slot Index increases by one modulo T each TSD seconds at the sender, where T is the number of time slots associated with the session and TSD is the time slot duration.

Channel Number (CN): 8 bits

CN is the channel number that this packet belongs to. CN for the base channel is T, where T is the total number of channels in the session. The CNs for the wave channels are 0 through T-1.

Packet Sequence Number (PSN): 16 bits

The PSN of each packet is scoped by its CN value. The sequence numbers of consecutive packets sent to the base channel are numbered consecutively decreasing modulo 2^16. The same sequence of PSNs are used for each wave channel in each cycle. The sequence numbers of consecutive packets sent to a wave channel are numbered consecutively decreasing modulo 2^16 within each cycle, ending with the last packet sent to the channel before the channel goes quiescent with PSN = 0.

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

The TSI uniquely identifies a session among all sessions from a particular sender. The TSI is scoped by the sender IP address, and thus the (sender IP address, TSI) pair uniquely identify the session. For ALC, 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.
In the example the TSI is 32 bits in length.

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.
In the example the TOI is 32 bits in length.

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.
In this example the SCT is present.

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.
In this example the ERT is not present.

FEC Payload ID: X bits

The length and format of the FEC Payload ID depends on the FEC Encoding ID as described in the FEC building block [10]. The example packet format corresponds to the format for ‘‘Small Block, Large Block and Expandable FEC Codes" as described in the FEC building block, for which the associated FEC Encoding ID 128. Note that the FEC Payload ID format is determined by the FEC Encoding ID carried in the Codepoint field of the LCT header as described above. For FEC Encoding ID 128, the FEC Payload ID consists of the following two fields that in total are X = 64 bits in length:

Source Block Number (SBN): 32 bits

The Source Block Number identifies from which source block of the object the encoding symbol(s) in the payload are generated. These blocks are numbered consecutively from 0 to N-1, where N is the number of source blocks in the object.

Encoding Symbol ID (ESI): 32 bits

The Encoding Symbol ID identifies which specific encoding symbol(s) generated from the source block are carried in the packet payload. The exact details of the correspondence between Encoding Symbol IDs and the encoding symbol(s) in the packet payload are dependent on the particular encoding algorithm used as identified by the Fec Encoding ID and by the FEC Encoding Name.

Encoding Symbol(s): Y bits

The encoding symbols are what the receiver uses to reconstruct an object. The total length Y of the encoding symbol(s) in the packet can be determined by the receiver of the packet by computing the total length of the received packet and subtracting off the length of the headers.

4.2. Header-Extension Fields

Header Extensions can be used to extend the LCT header portion of the ALC header to accommodate optional header fields that are not always used or have variable size. Header Extensions are not used in the example ALC packet format shown in the previous subsection. 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 ALC without changing the ALC version number. Non backward-compatible header extensions CANNOT be introduced without changing the ALC version number.

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)          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fig. 3 - 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. For this version of ALC, there are no PI-specific extensions.

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 ALC MUST support the EXT_NOP Header Extension and MUST recognize EXT_AUTH, but MAY NOT be able to parse its content.

4.3. Sender Operation

The sender operation when using ALC includes all the points made about the sender operation when using the LCT [11], WEBRC [12] and FEC [10] building blocks.

A sender using ALC MUST make available the required session description as described in Section 2.4.

Within a session a sender transmits a sequence of packets to the channels associated with the session. The ALC sender MUST obey the rules for filling in the CCI field in the packet headers and MUST send packets at the appropriate rates to the channels associated with the session as dictated by WEBRC [12].

The ALC sender MUST use the same TSI for all packets in the session. Several objects MAY be delivered within the same ALC session. If more than one object is to be delivered within a session then the sender MUST use the TOI field and each object MUST be identified by a unique TOI within the session, and the sender MUST use corresponding TOI for all packets pertaining to the same object. The FEC Payload ID MUST correspond to the encoding symbol(s) for the object carried in the payload of the packet.

Objects MAY be transmitted sequentially within a session, and they MAY be transmitted concurrently. However, 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. However, there are no rules with respect to mixing packets for different objects carried within the session. Although this issue affects the efficiency of the protocol, it does not affect the correctness nor the inter-operability of ALC between senders and receivers.

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.

It is RECOMMENDED that packet authentication be used. If packet authentication is used then the Header Extensions described in Section 4.2 MUST be used to carry the authentication.

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 WEBRC.

4.4. Receiver Operation

The receiver operation when using ALC includes all the points made about the receiver operation when using the LCT [11], WEBRC [12] and FEC [10] building blocks.

To be able to participate in a session, a receiver MUST obtain the REQUIRED session description as listed in Section 2.4. How receivers obtain a session description is outside the scope of ALC.

To be able to be a receiver in a session, the receiver MUST be able to process the ALC 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 the ALC header, it MUST drop from the session.

To be able to participate in a session, a receiver MUST implement the WEBRC building block using the Congestion Control Information field provided in the LCT header. If a receiver is not able to implement WEBRC it MUST NOT join the session.

Several objects can be carried either sequentially or concurrently within the same session. In this case, each object is identified by a unique TOI. Note that even if a sender 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 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 ALC sessions from one or more senders. The receiver MUST perform congestion control on each such session. The receiver MAY make choices to optimize the packet flow performance across multiple sessions, as long as the receiver still adheres to WEBRC for each session individually.

Upon receipt of each packet the receiver proceeds with the following steps in the order listed.

(1) The receiver MUST parse the packet header and verify that it is a valid header. If it is not valid then the packet MUST be discarded without further processing. If multiple packets are received that cannot be parsed then the receiver SHOULD leave the session.

(2) The receiver MUST verify that the sender IP address together with the TSI carried in the header matches one of the (sender IP address, TSI) pairs that was received in a session description and that the receiver is currently joined to. If there is not a match then the packet MUST be discarded without further processing. If multiple packets are received with non-matching (sender IP address, TSI) values then the receiver SHOULD leave the session. If the receiver is joined to multiple ALC sessions then the remainder of the steps are performed within the scope of the (sender IP address, TSI) session of the received packet.

(3) The receiver MUST process and act on the CCI field in accordance with WEBRC.

(4) If more than one object is carried in the session, the receiver MUST verify that the TOI carried in the LCT header matches one of the TOIs computed from the session description. If there is not a match, the packet MUST be discarded without further processing. If multiple packets are received with non-matching TOI values then the receiver SHOULD leave the session.

(5) The receiver SHOULD process the remainder of the packet, including interpreting the other header fields appropriately, and using the FEC Payload ID and the encoding symbol(s) in the payload to reconstruct the corresponding object.

It is RECOMMENDED that packet authentication be used. If packet authentication is used then it is RECOMMENDED that the receiver immediately check the authenticity of a packet before proceeding with step (1) above. If immediate checking is possible and if the packet fails the check then the receiver MUST discard the packet and reduce its reception rate to a minimum.

Some packet authentication schemes such as TESLA [15] do not allow an immediate authenticity check. In this case the receiver SHOULD check the authenticity of a packet as soon as possible, and if the packet fails the check then it MUST be discarded before step (5) above and reduce its reception rate to a minimum.

If multiple packets are received that fail the authentication check then the receiver SHOULD leave the session.

5. Security Considerations

The same security consideration that apply to the LCT, FEC and WEBRC building blocks also apply to ALC.

Because of the use of FEC, ALC is especially vulnerable to denial-of-service attacks by attackers that try to send forged packets to the session which would prevent successful reconstruction or cause inaccurate reconstruction of large portions of the object by receivers. ALC is also particularly affected by such an attack because 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 [17] 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.

WEBRC can be subject to denial-of-service attacks by attackers which try to confuse the congestion control mechanism for receivers by injecting forged packets into the multicast stream. This attack most adversely affects network elements and receivers downstream of the attack, and much less significantly the rest of the network and other receivers. Because of this and because of the potential attacks due to the use of FEC described above, it is RECOMMENDED that some form of packet authentication such as TESLA [15] be used to protect against such attacks and 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.

A receiver with an incorrect or corrupted implementation of WEBRC 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.

Another vulnerability of ALC 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) [8] or AH [7] protocols of IPSEC [6] to ensure the authenticity of the session description.

6. IANA Considerations

No information in this specification is directly subject to IANA registration. However, building blocks components used by ALC may introduce additional IANA considerations. In particular, the FEC building block used by ALC does require IANA registration of the FEC codecs used.

7. Intellectual Property Issues

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

ALC may be used with congestion control building blocks and other building blocks which contain proprietary elements, or have pending or granted patents.

8. Acknowledgments

Thanks to Vincent Roca, Justin Chapweske and Roger Kermode for their detailed comments on this draft.

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

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

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

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

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

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

[9] 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, February 2002, a work in progress.

[10] 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, February 2002.

[11] Luby, M., Gemmell, J., Vicisano, L., Rizzo, L., Handley, M., Crowcroft, J., "Layered Coding Transport building block", Internet Draft draft-ietf-rmt-bb-lct-04.txt, February 2002.

[12] Luby, M., Goyal, V., Skaria, S., "Wave and Equaltion Based Rate Control building block", Internet Draft draft-ietf-rmt-bb-webrc-01.txt, February 2002.

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

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

[15] 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.

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

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

[18] 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.

10. 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
Dip. Ing. dell'Informazione,

Univ. di Pisa
via Diotisalvi 2, 56126 Pisa, Italy

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

11. Full Copyright Statement