Internet Engineering Task Force
INTERNET-DRAFT
draft-ietf-rmt-pi-alc-07.ps

RMT WG
M.Luby/Digital Fountain
J.Gemmell/Microsoft
L.Vicisano/Cisco
L.Rizzo/ACIRI and Univ. Pisa
J. Crowcroft/UCL
28 March 2002
Expires: September 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 building block [13], a multiple rate congestion control building block that is in compliance with RFC2357 [14] and the FEC building block [12] 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 deliver 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 in conjunction with ALC should strive 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 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 protocol 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) in the packet header that indicates the expected remaining time of packet transmission for either the single object carried in the session or for the object identified by the Transmission Object Identifier (TOI) if there are multiple objects carried in the session. This can be used by receivers to determine if there is enough time remaining in the session to successfully receive enough additional packets to recover the object. If for example there is not enough time, then the push application may have receivers report back to the sender to extend the transmission of packets for the object for enough time to allow the receivers to obtain enough packets to reconstruct the object. The sender could then include an ERT based on the extended object transmission time in each subsequent packet header for the object. As other examples, the LCT header optionally can contain a Close Session flag that indicates when the sender is about to end sending packet to the session and a Close Object flag that indicates when the sender is about to end sending packets to the session for the object identified by the Transmission Object ID. However, these flags are not a completely reliable mechanism and thus the Close Session flag should only be used as a hint of when the session is about to close and the Close Object flag should only be used as a hint of when transmission of packets for the object is about to end.

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 this document.

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 this document.

1.3. Environmental Requirements and Considerations

All of the environmental requirements and considerations that apply to the LCT building block [13], the FEC building block [12], the multiple rate congestion control building block 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 session. The Transmission Session Identifier (TSI) that MUST appear in each packet header is used for this purpose. 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 [3] and the

Source-Specific Multicast (SSM) model as defined in [7]. 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 the multiple rate congestion control building block interacts with ASM. The congestion control building block may use 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. The congestion control building block may 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. For a fuller consideration of these issues, consult the multiple rate congestion control building block.

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 [13] to provide in-band session management functionality. ALC uses a multiple rate congestion control building block that is compliant with RFC2357 [14] to provide 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 [12] 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. Congestion control is performed over the aggregate of packets sent to channels belonging to a session. The fact that an ALC session is restricted to a single sender does not preclude the possibility of receiving packets for the same objects from multiple senders. However, each sender would be sending packets to a a different session to which congestion control is individually applied. Although receiving concurrently from multiple sessions is allowed, how this is done at the application level is outside the scope of this document.

ALC is a protocol instantiation as defined in RFC3048 [19]. This document describes version 1 of ALC which MUST use version 1 of LCT described in [13]. Like LCT, ALC is designed to be used with the IP multicast network service. This specification defines ALC as payload of the UDP transport protocol [17] 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. 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.

An ALC packet header immediately follows the UDP header and consists of the default LCT header that is described in [13] followed by the FEC Payload ID that is described in [12]. The Congestion Control Information field within the LCT header carries the required Congestion Control Information that is described in the multiple rate congestion control building block specified that is compliant with RFC2357 [14]. The packet payload that follows the ALC packet header consists of encoding symbols that are identified by the FEC Payload ID as described in [12].

Each receiver is required to obtain a Session Description before joining an ALC session. As described later, the Session Description includes out-of-band information required for the LCT, FEC and the multiple rate congestion control building blocks. The FEC Object Transmission Information specified in the FEC building block [12] required for each object to be received by a receiver can be communicated to a receiver either out-of-band or in-band using a Header Extension. The means for communicating the Session Description and the FEC Object Transmission Information to a receiver is outside the scope of this document.

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 Congestion Control Information from the specified multiple rate congestion control protocol. There is a field in the LCT header that specifies the length of the CCI field, and the multiple rate congestion control building block MUST uniquely identify a format of the CCI field that corresponds to this length.

The LCT header contains a Codepoint field that MAY be used to communicate to a receiver the settings for information that may vary during a session. If used, the mapping between settings and Codepoint values is to be communicated in the Session Description, and this mapping is outside the scope of this document. For example, the FEC Encoding ID that is part of the FEC Object Transmission Information as specified in the FEC building block [12] could vary for each object carried in the session, and the Codepoint value could be used to communicate the FEC Encoding ID to be used for each object. The mapping between FEC Encoding IDs and Codepoints could be for example the identity mapping.

If more than one object is to be carried within a session then the Transmission Object Identifier (TOI) MUST be used in the LCT header to identify which packets are to be associated with which objects. In this case the receiver MUST use the TOI to associate received packets with objects. The TOI is scoped by the IP address of the sender and the TSI, i.e., the TOI is scoped by the session. The TOI for each object is REQUIRED to be unique within a session, but MAY NOT be unique across sessions. Furthermore, the same object MAY have a different TOI in different sessions. The mapping between TOIs and objects carried in a session is outside the scope of this document.

If only one object is carried within a session then the TOI MAY be omitted from the LCT header.

The default LCT header from version 1 of the LCT building block [13] MUST be used.

2.2. Multiple rate congestion control building block

Implementors of ALC MUST implement a multiple rate feedback-free congestion control building block that is in accordance to RFC2357 [14]. Congestion control MUST be applied to all packets within a session independently of which information about which object is carried in each packet. Multiple rate congestion control is specified because of its suitability to scale massively and because of its suitability for reliable content delivery. The multiple rate congestion control building block MUST specify in-band Congestion Control Information (CCI) that MUST be carried in the CCI field of the LCT header. The multiple rate congestion control building block MAY specify more than one format, but it MUST specify at most one format for each of the possible lengths 32, 64, 96 or 128 bits. The value of C in the LCT header that determines the length of the CCI field MUST correspond to one of the lengths for the CCI defined in the multiple rate congestion control building block, this length MUST be the same for all packets sent to a session, and the CCI format that corresponds to the length as specified in the multiple rate congestion control building block MUST be the format used for the CCI field in the LCT header.

When using a multiple rate congestion control building block 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.

2.3. FEC building block

The FEC building block [12] provides reliable object delivery within an ALC session. Each object sent in the session is independently encoded using FEC codes as described in [11], 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 [12]. 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.

As described in [12], a receiver is REQUIRED to obtain the FEC Object Transmission Information for each object for which data packets are received from the session. The FEC Object Transmission Information includes:

- 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 For each object in the session, the length of the object in bytes.
- o The additional required FEC Object Transmission Information for the FEC Encoding ID as prescribed in the FEC building block [12]. 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.

Some of the FEC Object Transmission Information MAY be implicit based on the implementation. As an example, 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 are always used for a particular application and thus the FEC Encoding ID and FEC Encoding Name are implicitly defined.

Sometimes the objects that will be sent in a session are completely known before the receiver joins the session, in which case the FEC Object Transmission Information for all objects in the session can be communicated to receivers before they join the session. Other times the objects may not known when the session begins, or receivers may join a session in progress and may not be interested in some objects for which transmission has finished, or receivers may leave a session before some objects are even available within the session. In these cases, the FEC Object Transmission Information for each object may be dynamically communicated to receivers at or before the time packets for the object are received from the session. This may be accomplished using either an out-of-band mechanism, in-band using the Codepoint field or a Header Extension, or any combination of these methods. How the FEC Object Transmission Information is communicated to receivers is outside the scope of this document.

If packets for more than one object are transmitted within a session then a Transmission Object Identifier (TOI) that uniquely identifies objects within a session MUST appear in each packet header. Portions of the FEC Object Transmission Information could be the same for all objects in the session, in which case these portions can be communicated to the receiver with an indication that this applies to all objects in the session. These portions may be implicitly determined based on the application, e.g., an application may use the same FEC Encoding ID for all objects in all sessions. If there is a portion of the FEC Object Transmission Information that may vary from object to object and if this FEC Object Transmission Information is communicated to a receiver out-of-band then the TOI for the object MUST also be communicated to the receiver together with the corresponding FEC Object Transmission Information, and the receiver MUST use the corresponding FEC Object Transmission Information for all packets received with that TOI. How the TOI and corresponding FEC Object Transmission Information is communicated out-of-band to receivers is outside the scope of this document.

It is also possible that there is a portion of the FEC Object Transmission Information that may vary from object to object that is carried in-band, for example in the CodePoint field or in Header Extensions. How this is done is outside the scope of this document. In this case the FEC Object Transmission Information is associated with the object identified by the TOI carried in the packet.

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 multiple rate congestion control building block to be used for the session;
- o The sender IP address;
- o The number of channels in the session;
- o The address and port number 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 Header Extensions are to be used, the format of these Header Extensions.
- o Enough information to determine the packet authentication scheme being used, if it is being used.

How Session Description is communicated to receivers is outside the scope of this document.

The Codepoint field within the LCT portion of the header CAN be used to communicate in-band some of the dynamically changing information within a session. To do this, a mapping between Codepoint values and the different dynamic settings MUST be included within the Session Description, and then settings to be used are communicated via the Codepoint value placed into each packet. For example, it is possible that multiple objects are delivered within the same session and that a different FEC encoding algorithm is used for different types of objects. Then the Session Description could contain the mapping between Codepoint values and FEC Encoding IDs. As another example, it is possible that a different packet authentication scheme is used for different packets sent to the session. In this case, the mapping between the packet authentication scheme and Codepoint values could be provided in the session description. Combinations of settings can be mapped to Codepoint values as well. For example, a particular combination of a FEC Encoding ID and a packet authentication scheme could be associated with a Codepoint value.

The Session Description could also include, but is not limited to:

- o The mappings between combinations of settings and Codepoint values;
- 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 the Object Transmission Information for each object, when the object will be available within the session and for how long.

The Session Description could be in a form such as SDP as defined in RFC2327 [5], or XML metadata as defined in RFC3023 [15], or HTTP/Mime headers as defined in RFC2068 [4], etc. It might be carried in a session announcement protocol such as SAP as defined in RFC2974 [6], 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 formats and methods for communication of Session Descriptions to receivers is beyond the scope of this document.

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 [16]. 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 building block [13], the FEC building block [12] and with a multiple rate congestion control building block completely specifies a working reliable multicast transport protocol that conforms to the requirements described in RFC2357 [14].

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 [13] and the FEC Payload ID is described in the FEC building block [12]. The Congestion Control Information field in the LCT header contains the REQUIRED Congestion Control Information that is described in the multiple rate congestion control building block used. 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 [13] 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 [13].

```
  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.

4.2. Detailed Example of Packet format used by ALC

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      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Congestion Control Information (CCI, length = 32 bits)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  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 - A detailed example of the 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 for each field in this detailed example of 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

The Congestion Control Information (CCI) field specified by the multiple rate congestion control building block is a multiple of 32-bits in length. The multiple rate congestion control building block MUST specify a format for the CCI. The congestion control building block MAY specify formats for different CCI lengths, where the set of possible lengths is 32, 64, 96 or 128 bits. The value of C MUST match the length of exactly one of the possible formats for the congestion control building block, and this format MUST be used for the CCI field. The value of C MUST be the same for all packets sent to a session.
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.

C=2 indicates the 96-bit CCI field format is to be used.
C=3 indicates the 128-bit CCI field format is to be used.
In the example C=0 indicates that a 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 packets will be sent to the session for either the single object carried in the session or for the object identified by the TOI if there are multiple objects carried in the session.
Senders MUST NOT set R = 1 when the ERT for the object 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 mapping that identifies settings for portions of the Session Description that can change within the session. The mapping between Codepoint values and the settings for portions of the Session Description is to be communicated out-of-band.
In the example the portion of the Session Description that can change within the session is the FEC Encoding ID, and the identity mapping is used between Codepoint values and FEC Encoding IDs. Thus, CP=128 identifies FEC Encoding ID 128, the ‘‘Small Block, Large Block and Expandable FEC Codes" as described in the FEC building block [12]. The FEC Payload ID associated with FEC Encoding ID 128 is 64-bits in length.

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

This is field contains the Congestion Control Information as defined by the specified multiple rate congestion control building block. The format of this field is determined by the multiple rate congestion control building block.
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.
In the example, the CCI is 32-bits in length. The format of the CCI field for the example MUST correspond to the format for the 32-bit version of the CCI specified in the multiple rate congestion control building block.

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 of packets for either the single object carried in the session or for the object identified by the TOI if there are multiple objects carried in 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 [12]. The FEC Payload ID format is determined by the FEC Encoding ID that MUST be communicated in the Session Description. The Session Description MAY specify that more than one FEC Encoding ID is used in the session, in which case the Session Description MUST contain a mapping that identifies which Codepoint values correspond to which FEC Encoding IDs. This mapping, if used, is outside the scope of this document.
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. 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.

4.3. 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.

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.

For this version of ALC, the following PI-specific extension is defined:

EXT_FTI=64 FEC Object Transmission Information extension
The purpose of this extension is to carry in-band the FEC Object Transmission

Information for an object. 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.

4.4. Sender Operation

The sender operation when using ALC includes all the points made about the sender operation when using the LCT building block [13], the FEC building block [12] and the multiple rate congestion control building block.

A sender using ALC MUST make available the required Session Description as described in Section 2.4. A sender also MUST make available the required FEC Object Transmission Information as described in Section 2.3.

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 the multiple rate congestion control building block.

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 4.3 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 the multiple rate congestion control building block.

4.5. Receiver Operation

The receiver operation when using ALC includes all the points made about the receiver operation when using the LCT building block [13], the FEC building block [12] and the multiple rate congestion control building block.

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 this document.

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 multiple rate congestion control building block using the Congestion Control Information field provided in the LCT header. If a receiver is not able to implement the multiple rate congestion control building block 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.

As described in Section 2.3, a receiver MUST obtain the required FEC Object Transmission Information for each object for which the receiver receives and processes packets.

A receiver MAY concurrently join 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 the multiple rate congestion control building block 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 the multiple rate congestion control building block.

(4) If more than one object is carried in the session, the receiver MUST verify that the TOI carried in the LCT header is valid. If the TOI is not valid, the packet MUST be discarded without further processing.

(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 (3) 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 before continuing to regulate its reception rate using the multiple rate congestion control.

Some packet authentication schemes such as TESLA [16] 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 before continuing to regulate its reception rate using the multiple rate congestion control.

5. Security Considerations

The same security consideration that apply to the LCT, FEC and the multiple rate congestion control 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 application built on top of ALC make an integrity check on received content, e.g., by appending an MD5 hash [18] 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.

The multiple rate congestion control building block 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 [16] 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 the multiple rate congestion control building block may affect health of the network in the path between the sender and the receiver, and may also affect the reception rates of other receivers joined to the session. It is therefore RECOMMENDED that receivers be required to identify themselves as legitimate before they receive the Session Description needed to join the session.

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) [10] or AH [9] protocols of IPSEC [8] 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] Deering, S., "Host Extensions for IP Multicasting", RFC1112, August 1989.

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

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

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

[7] Holbrook, H. W., "A Channel Model for Multicast", Ph.D. Dissertation, Stanford University,

Department of Computer Science, Stanford, California, August 2001.

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

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

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

[11] 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.

[12] 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.

[13] 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.

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

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

[16] 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.

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

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

[19] 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