

RMT  
Internet-Draft  
Intended status: Standards Track  
Expires: April 21, 2012

V. Roca  
INRIA  
B. Adamson  
Naval Research Laboratory  
October 19, 2011

FCAST: Scalable Object Delivery for the ALC and NORM Protocols  
draft-ietf-rmt-fcast-05

Abstract

This document introduces the FCAST object (e.g., file) delivery application on top of the ALC and NORM reliable multicast protocols. FCAST is a highly scalable application that provides a reliable object delivery service.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on April 21, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	4
1.1. Requirements notation . . . . .	4
1.2. Definitions, Notations and Abbreviations . . . . .	5
2. FCAST Data Formats . . . . .	6
2.1. Compound Object Format . . . . .	6
2.2. Carousel Instance Descriptor Format . . . . .	8
3. FCAST Principles . . . . .	11
3.1. FCAST Content Delivery Service . . . . .	11
3.2. Meta-Data Transmission . . . . .	12
3.3. Meta-Data Content . . . . .	13
3.4. Carousel Transmission . . . . .	14
3.5. Carousel Instance Descriptor Special Object . . . . .	14
3.6. Compound Object Identification . . . . .	15
3.7. FCAST Sender Behavior . . . . .	17
3.8. FCAST Receiver Behavior . . . . .	18
4. Security Considerations . . . . .	18
4.1. Problem Statement . . . . .	18
4.2. Attacks Against the Data Flow . . . . .	19
4.2.1. Access to Confidential Objects . . . . .	19
4.2.2. Object Corruption . . . . .	19
4.3. Attacks Against the Session Control Parameters and Associated Building Blocks . . . . .	21
4.3.1. Attacks Against the Session Description . . . . .	21
4.3.2. Attacks Against the FCAST CID . . . . .	22
4.3.3. Attacks Against the Object Meta-Data . . . . .	22
4.3.4. Attacks Against the ALC/LCT and NORM Parameters . . . . .	22
4.3.5. Attacks Against the Associated Building Blocks . . . . .	23
4.4. Other Security Considerations . . . . .	23
4.5. Minimum Security Recommendations . . . . .	24
5. Requirements for Compliant Implementations . . . . .	24
5.1. Requirements Related to the Object Meta-Data . . . . .	24
5.2. Requirements Related to the Carousel Instance Descriptor (CID) Mechanism . . . . .	26
6. Operational Considerations . . . . .	26
7. IANA Considerations . . . . .	27
7.1. Namespace declaration for Object Meta-Data Format . . . . .	27
7.1.1. Object Meta-Data Format registration . . . . .	27
7.2. Namespace declaration for Object Meta-Data Encoding . . . . .	28
7.2.1. Object Meta-Data Encoding registration . . . . .	28
8. Acknowledgments . . . . .	28
9. References . . . . .	28
9.1. Normative References . . . . .	28
9.2. Informative References . . . . .	29
Appendix A. FCAST Examples . . . . .	30
A.1. Regular Compound Object Example . . . . .	30
A.2. Carousel Instance Descriptor Example . . . . .	31

Appendix B. Additional Meta-Data Transmission Mechanisms . . . .	32
B.1. Supporting Additional Mechanisms . . . . .	32
B.2. Using NORM_INFO Messages with FCAST/NORM . . . . .	33
B.2.1. Example . . . . .	33
Authors' Addresses . . . . .	35

## 1. Introduction

This document introduces the FCAST reliable and scalable object (e.g., file) delivery application. Two variants of FCAST exist:

- o FCAST/ALC that relies on the Asynchronous Layer Coding (ALC) [RFC5775] and the Layered Coding Transport (LCT) [RFC5651] reliable multicast transport protocol, and
- o FCAST/NORM that relies on the NACK-Oriented Reliable Multicast (NORM) [RFC5740] reliable multicast transport protocol.

Hereafter, the term FCAST denotes either FCAST/ALC or FCAST/NORM. FCAST is not a new protocol specification per se. Instead it is a set of data format specifications and instructions on how to use ALC and NORM to implement a file-casting service.

A design goal behind FCAST is to define a streamlined solution, in order to enable lightweight implementations of the protocol stack, and limit the operational processing and storage requirements. A consequence of this choice is that FCAST cannot be considered as a versatile application, capable of addressing all the possible use-cases. On the contrary, FCAST has some intrinsic limitations. From this point of view it differs from FLUTE [RMT-FLUTE] which favors flexibility at the expense of some additional complexity.

A good example of the design choices meant to favor simplicity is the way FCAST manages the object meta-data: by default, the meta-data and the object content are sent together, in a compound object. This solution has many advantages in terms of simplicity as will be described later on. However this solution has an intrinsic limitation since it does not enable a receiver to decide in advance, before beginning the reception of the compound object, whether the object is of interest or not, based on the information that may be provided in the meta-data. Therefore this document discusses additional techniques that may be used to mitigate this limitation. When use-cases require that each receiver download the whole set of objects sent in the session (e.g., with mirroring tools), this limitation is not considered a problem.

### 1.1. Requirements notation

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].

## 1.2. Definitions, Notations and Abbreviations

This document uses the following definitions:

FCAST/ALC: denotes the FCAST application running on top of the ALC/LCT reliable transport protocol;

FCAST/NORM: denotes the FCAST application running on top of the NORM reliable transport protocol;

FCAST: denotes either FCAST/ALC or FCAST/NORM;

Compound Object: denotes an ALC or NORM transport object composed of the Compound Object Header (Section 2.1), including any meta-data, and the content of the original application object (e.g., a file);

Carousel: denotes the process of sending Compound Objects implemented by a FCAST sender;

Carousel Instance: denotes a fixed set of registered Compound Objects that are sent by the carousel during a certain number of cycles. Whenever Compound Objects need to be added or removed, a new Carousel Instance is defined;

Carousel Instance Descriptor (CID): denotes a special object that lists the Compound Objects that comprise a given Carousel Instance;

Carousel Instance Identifier (CIID): numeric value that identifies a Carousel Instance.

Carousel Cycle: denotes a transmission round within which all the Compound Objects registered in the Carousel Instance are transmitted a certain number of times. By default, Compound Objects are transmitted once per cycle, but higher values are possible, that might differ on a per-object basis;

Transport Object Identifier (TOI): denotes the numeric identifier associated to a specific object by the underlying transport protocol. In the case of ALC, this corresponds to the TOI described in [RFC5651]. In the case of NORM, this corresponds to the NormTransportId described in [RFC5740].

FEC Object Transmission Information (FEC OTI): FEC information associated with an object and that is essential for the FEC decoder to decode a specific object.

## 2. FCAST Data Formats

This section details the various data formats used by FCAST.

### 2.1. Compound Object Format

In an FCAST session, Compound Objects are constructed by prepending the Compound Object Header (which contains the meta-data of the object) before the original object data content (see Section 3.2). Figure 1 illustrates the associated Compound Object header format.

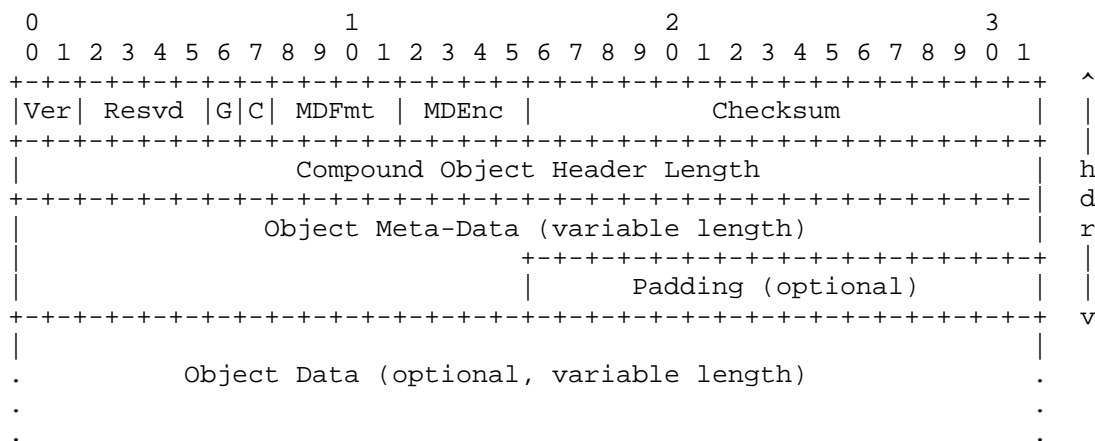


Figure 1: Compound Object Format.

The Compound Object Header fields are:

Field	Description
Version	2-bit field that MUST be set to 0 in this specification and indicates the protocol version number.
Reserved	4-bit field that MUST be set to 0 in this specification and is reserved for future use. Receivers MUST ignore this field.
G	1-bit field that, when set to 1, indicates that the checksum encompasses the whole Compound Object (Global checksum). When set to 0, this field indicates that the checksum encompasses only the Compound Object header.

	C	1-bit field that, when set to 1, indicates the object is a Carousel Instance Descriptor (CID). When set to 0, this field indicates that the transported object is a standard object.
Meta-Data Format (MDFmt)		4-bit field that defines the format of the object meta-data (see Section 7). An HTTP/1.1 metainformation format [RFC2616] MUST be supported and is associated to value 0. Other formats (e.g., XML) MAY be defined in the future.
Meta-Data Encoding (MDEnc)		4-bit field that defines the optional encoding of the Object Meta-Data field (see Section 7). By default, a plain text encoding is used and is associated to value 0. GZIP encoding MUST also be supported and is associated to value 1. Other encodings MAY be defined in the future.
Checksum		16-bit field that contains the checksum computed over either the whole Compound Object (when G is set to 1), or over the Compound Object header (when G is set to 0), using the Internet checksum algorithm specified in [RFC1071]. More precisely, the checksum field is the 16-bit one's complement of the one's complement sum of all 16-bit words to be considered. If a segment contains an odd number of octets to be checksummed, the last octet is padded on the right with zeros to form a 16-bit word for checksum purposes (this pad is not transmitted). While computing the checksum, the checksum field itself is set to zero.
Compound Object Header Length		32-bit field indicating total length (in bytes) of all fields of the Compound Object Header, except the optional padding. A header length field set to value 8 means that there is no meta-data included. When this size is not multiple to 32-bits words and when the Compound Object Header is followed by a non null Compound Object Data, padding MUST be added. It should be noted that the meta-data field maximum size is equal to $(2^{32} - 8)$ bytes.

Object Meta-Data	Variable length field that contains the meta-data associated to the object. The format and encoding of this field are defined respectively by the MDFmt and MDEnc fields. With the default HTTP/1.1 format and plain text encoding, the Meta-Data is NULL-terminated plain text that follows the "TYPE" ":" "VALUE" "<CR-LF>" format used in HTTP/1.1 for metainformation [RFC2616]. The various meta-data items can appear in any order. The associated string, when non empty, MUST be NULL-terminated. When no meta-data is communicated, this field MUST be empty and the Compound Object Header Length MUST be equal to 8.
Padding	Optional, variable length field of zero-value bytes to align the start of the Object Data to 32-bit boundary. Padding is only used when the Compound Object Header Length value, in bytes, is not multiple of 4 and when the Compound Object Header is followed by non null Compound Object Data.

The Compound Object Header is then followed by the Object Data, i.e., the original object possibly encoded by FCAST. Note that the length of this content is the transported object length (e.g., as specified by the FEC OTI) minus the Compound Object Header Length and optional padding if any.

## 2.2. Carousel Instance Descriptor Format

In an FCAST session, a Carousel Instance Descriptor (CID) MAY be sent in order to carry the list of Compound Objects that are part of a given Carousel Instance (see Section 3.5). The format of the CID, that is sent as a special Compound Object, is given in Figure 2. Being a special case of Compound Object, this format is in line with the format described in Section 2.1.



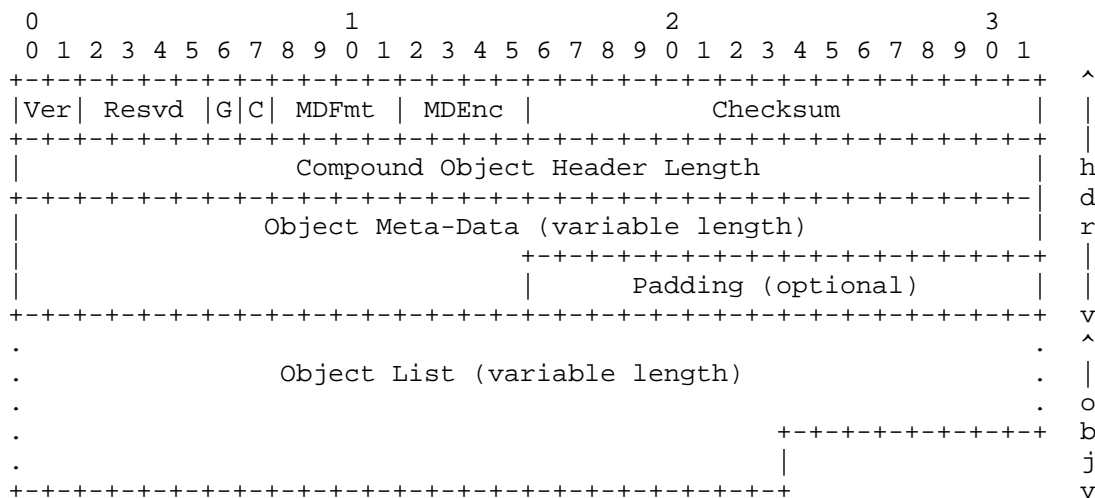


Figure 2: Carousel Instance Descriptor Format.

Because the CID is transmitted as a special Compound Object, the following CID-specific meta-data entries are defined:

- o Fcast-CID-Complete: when set to 1, it indicates that no new object in addition to the ones whose TOI are specified in this CID, or the ones that have been specified in the previous CID(s), will be sent in the future. Otherwise it MUST be set to 0. This entry is optional. If absent, a receiver MUST conclude that the session is complete.
- o Fcast-CID-ID: this entry contains the Carousel Instance Identifier, or CIID. It starts from 0 and is incremented by 1 for each new carousel instance. This entry is optional if the FCAST session consists of a single, complete, carousel instance. In all other cases, this entry MUST be defined. In particular, the CIID is used by the TOI equivalence mechanism thanks to which any object is uniquely identified, even if the TOI is updated (e.g., after re-enqueuing the object with NORM). The Fcast-CID-ID value can also be useful to detect possible gaps in the Carousel Instances, for instance caused by long disconnection periods. Finally, it can also be useful to avoid problems when TOI wrapping to 0 takes place to differentiate the various incarnations of the TOIs if need be.

The motivation for making the Fcast-CID-Complete and Fcast-CID-ID entries optional is to simplify the simple case of a session consisting of a single, complete, carousel instance, with an Object List given in plain text, without any content encoding. In that

case, the CID does not need to contain any meta-data entry.

Additionally, the following standard meta-data entries are often used (Section 3.3):

- o Content-Length: it specifies the size of the object list, before any content encoding (if any).
- o Content-Encoding: it specifies the optional encoding of the object list, performed by FCAST. For instance:  
Content-Encoding: gzip  
indicates that the Object List field has been encoded with GZIP [RFC1952]. If there is no Content-Encoding entry, the receiver MUST assume that the Object List field is plain text (default). The support of GZIP encoding, or any other solution, remains optional.

An empty Object List is valid and indicates that the current carousel instance does not include any object (Section 3.5). This can be specified by using the following meta-data entry:

Content-Length: 0  
or simply by leaving the Object List empty. In both cases, padding MUST NOT be used and consequently the transported object length (e.g., as specified by the FEC OTI) minus the Compound Object Header Length equals zero.

The non-encoded (i.e., plain text) Object List, when non empty, is a NULL-terminated ASCII string. It can contain two things:

- o a list of TOI values, and
- o a list of TOI equivalences;

First of all, this string can contain the list of TOIs included in the current carousel instance, specified either as the individual TOIs of each object, or as TOI intervals, or any combination. The format of the ASCII string is a comma-separated list of individual "TOI" values or "TOI\_a-TOI\_b" elements. This latter case means that all values between TOI\_a and TOI\_b, inclusive, are part of the list. In that case TOI\_a MUST be strictly inferior to TOI\_b. If a TOI wrapping to 0 occurs in an interval, then two TOI intervals MUST be specified, TOI\_a-MAX\_TOI and 0-TOI\_b.

This string can also contain the TOI equivalences, if any. The format is a comma-separated list of "(" newTOI "=" 1stTOI "/" 1stCIID ")" elements. Each element says that the new TOI, for the current Carousel Instance, is equivalent to (i.e., refers to the same object as) the provided identifier, 1stTOI, for the Carousel Instance of ID

1stCIID.

The ABNF specification is the following:

```
cid-list    = *(list-elem *( "," list-elem))
list-elem   = toi-elem / toieq-elem
toi-elem    = toi-value / toi-interval
toi-value   = 1*DIGIT
toi-interval = toi-value "-" toi-value
              ; additionally, the first toi-value MUST be
              ; strictly inferior to the second toi-value
toieq-elem  = "(" toi-value "=" toi-value "/" ciid-value ")"
ciid-value  = 1*DIGIT
DIGIT       = %x30-39
              ; a digit between 0 and 9, inclusive
```

For readability purposes, it is RECOMMENDED that all the TOI values in the list be given in increasing order. However a receiver MUST be able to handle non-monotonically increasing values. It is also RECOMMENDED to group the TOI equivalence elements together, at the end of the list, in increasing newTOI order. However a receiver MUST be able to handle lists of mixed TOI and TOI equivalence elements. Specifying a TOI equivalence for a given newTOI relieves the sender from specifying newTOI explicitly in the TOI list. However a receiver MUST be able to handle situations where the same TOI appears both in the TOI value and TOI equivalence lists. Finally, a given TOI value or TOI equivalence item MUST NOT be included multiple times in either list.

For instance, the following object list specifies that the current Carousel Instance is composed of 8 objects, and that TOIs 100 to 104 are equivalent to the TOIs 10 to 14 of Carousel Instance ID 2 and refer to the same objects:

```
97,98,99,(100=10/2),(101=11/2),(102=12/2),(103=13/2),(104=14/2)
```

or equivalently:

```
97-104,(100=10/2),(101=11/2),(102=12/2),(103=13/2),(104=14/2)
```

### 3. FCAST Principles

This section details the principles of FCAST.

#### 3.1. FCAST Content Delivery Service

The basic goal of FCAST is to transmit objects to a group of receivers in a reliable way, where the receiver set may be restricted

to a single receiver or may include possibly a very large number of receivers. FCAST supports two forms of operation:

1. FCAST/ALC, where the FCAST application works on top of the ALC/LCT reliable multicast transport protocol, without any feedback from the receivers, and
2. FCAST/NORM, where the FCAST application works on top of the NORM reliable multicast transport protocol, that requires positive/negative acknowledgements from the receivers.

This specification is designed such that both forms of operation share as much commonality as possible. Section 6 discusses some operational aspects and the content delivery service that is provided by FCAST for a given use-case.

### 3.2. Meta-Data Transmission

FCAST carries meta-data elements by prepending them to the object they refer to. As a result, a Compound Object is created that is composed of a header followed by the original object data. This header is itself composed of the meta-data as well as several fields, for instance to indicate the boundaries between the various parts of this Compound Object (Figure 3).

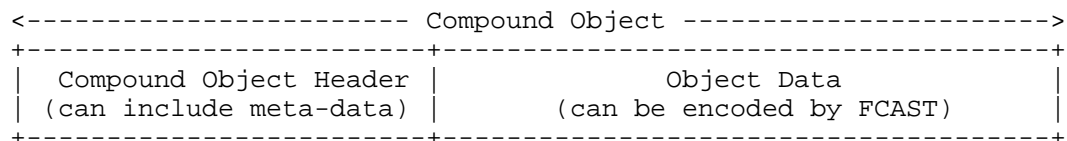


Figure 3: Compound Object composition.

Attaching the meta-data to the object is an efficient solution, since it guaranties that meta-data are received along with the associated object, and it allows the transport of the meta-data to benefit from any transport-layer erasure protection of the Compound Object (e.g., using FEC encoding and/or NACK-based repair). However a limit of this scheme is that a client does not know the meta-data of an object before beginning its reception, and in case of erasures affecting the meta-data, not until the object decoding is completed. The details of course depend upon the transport protocol and the FEC code used.

Appendix B describes extensions that provide additional means to carry meta-data, e.g., to communicate meta-data ahead of time.

### 3.3. Meta-Data Content

A compliant FCAST implementation MUST support at least the following items:

- o Content-Location: the URI of the object, which gives the name and location of the object;
- o Content-Type: a string that contains the MIME type of the object;
- o Content-Length: an unsigned 64-bit integer that contains the size of the initial object, before any content encoding (if any) and without considering the Compound Object header. Note that the use of certain FEC schemes MAY further limit the maximum value of the object;
- o Content-Encoding: a string that contains the optional encoding of the object performed by FCAST. If there is no Content-Encoding entry, the receiver MUST assume that the object is not encoded (default). The support of GZIP encoding, or any other solution, remains optional.
- o Content-MD5: a binary content coded as "base64" that contains the MD5 message digest of the object in order to check its integrity. This digest is meant to protect from transmission and processing errors, not from deliberate attacks by an intelligent attacker (see Section 4). This digest only protects the object, not the header, and therefore not the meta-data. A separate checksum is provided to that purpose (Section 2.1);

This list is not limited and new meta-data information can be added. For instance, when dealing with very large objects (e.g., that largely exceed the working memory of a receiver), it can be interesting to split this object into several sub-objects (or slices). When this happens, the meta-data associated to each sub-object MUST include the following entries:

- o Fcast-Obj-Slice-Nb: an unsigned 32-bit integer that contains the total number of slices. A value greater than 1 indicates that this object is the result of a split of the original object;
- o Fcast-Obj-Slice-Idx: an unsigned 32-bit integer that contains the slice index (in the {0 .. SliceNb - 1} interval);
- o Fcast-Obj-Slice-Offset: an unsigned 64-bit integer that contains the offset at which this slice starts within the original object;

### 3.4. Carousel Transmission

A set of FCAST Compound Objects scheduled for transmission are considered a logical "Carousel". A given "Carousel Instance" is comprised of a fixed set of Compound Objects. Whenever the FCAST application needs to add new Compound Objects to, or remove old Compound Objects from the transmission set, a new Carousel Instance is defined since the set of Compound Objects changes. Because of the native object multiplexing capability of both ALC and NORM, sender and receiver(s) are both capable to multiplex and demultiplex FCAST Compound Objects.

For a given Carousel Instance, one or more transmission cycles are possible. During each cycle, all of the Compound Objects comprising the Carousel are sent. By default, each object is transmitted once per cycle. However, in order to allow different levels of priority, some objects MAY be transmitted more often than others during a cycle, and/or benefit from higher FEC protection than others. This can be the case for instance for the CID objects (Section 3.5). For some FCAST usage (e.g., a unidirectional "push" mode), a Carousel Instance may be associated to a single transmission cycle. In other cases it may be associated to a large number of transmission cycles (e.g., in "on-demand" mode, where objects are made available for download during a long period of time).

### 3.5. Carousel Instance Descriptor Special Object

The FCAST sender CAN transmit an OPTIONAL Carousel Instance Descriptor (CID). The CID carries the list of the Compound Objects that are part of a given Carousel Instance, by specifying their respective Transmission Object Identifiers (TOI). However the CID does not describe the objects themselves (i.e., there is no meta-data). Additionally, the CID MAY include a "Complete" flag that is used to indicate that no further modification to the enclosed list will be done in the future. Finally, the CID MAY include a Carousel Instance ID that identifies the Carousel Instance it pertains to. These aspects are discussed in Section 2.2.

There is no reserved TOI value for the CID Compound Object itself, since this special object is regarded by ALC/LCT or NORM as a standard object. On the contrary, the nature of this object (CID) is indicated by means of a specific Compound Object header field (the "I" flag) so that it can be recognized and processed by the FCAST application as needed. A direct consequence is the following: since a receiver does not know in advance which TOI will be used for the following CID (in case of a dynamic session), he MUST NOT filter out packets that are not in the current CID's TOI list. Said differently, the goal of CID is not to setup ALC or NORM packet

filters (this mechanism would not be secure in any case).

The use of a CID remains OPTIONAL. If it is not used, then the clients progressively learn what files are part of the carousel instance by receiving ALC or NORM packets with new TOIs. However using a CID has several benefits:

- o When the "Complete" flag is set (if ever), the receivers know when they can leave the session, i.e., when they have received all the objects that are part of the last carousel instance of this delivery session;
- o In case of a session with a dynamic set of objects, the sender can reliably inform the receivers that some objects have been removed from the carousel with the CID. This solution is more robust than the "Close Object flag (B)" of ALC/LCT since a client with an intermittent connectivity might lose all the packets containing this B flag. And while NORM provides a robust object cancellation mechanism in the form of its NORM\_CMD(SQUELCH) message in response to receiver NACK repair requests, the use of the CID provides an additional means for receivers to learn of objects for which it is futile to request repair;
- o The TOI equivalence (Section 3.6) can be signaled with the CID. This is often preferable to the alternative solution where the equivalence is identified by examining the object meta-data, especially in case of erasures.

During idle periods, when the carousel instance does not contain any object, a CID with an empty TOI list MAY be transmitted. In that case, a new carousel instance ID MUST be used to differentiate this (empty) carousel instance from the other ones. This mechanism can be useful to inform the receivers that:

- o all the previously sent objects have been removed from the carousel. It therefore improves the FCAST robustness even during "idle" period;
- o the session is still active even if there is currently no content being sent. Said differently, it can be used as a heartbeat mechanism. If the "Complete" flag has not been set, it implicitly informs the receivers that new objects MAY be sent in the future;

### 3.6. Compound Object Identification

The FCAST Compound Objects are directly associated with the object-based transport service that the ALC and NORM protocols provide. In each of these protocols, the messages containing transport object

content are labeled with a numeric transport object identifier (i.e., the ALC TOI and the NORM NormTransportId). For the purposes of this document, this identifier in either case (ALC or NORM) is referred to as the TOI.

There are several differences between ALC and NORM:

- o the ALC use of TOI is rather flexible, since several TOI field sizes are possible (from 16 to 112 bits), since this size can be changed at any time, on a per-packet basis, and since the TOI management is totally free as long as each object is associated to a unique TOI (if no wraparound happened);
- o the NORM use of TOI is more directive, since the TOI field is 16 bit long and since TOIs MUST be managed sequentially;

In both NORM and ALC, it is possible that the transport identification space may eventually wrap for long-lived sessions (especially with NORM where this phenomenon is expected to happen more frequently). This can possibly introduce some ambiguity in FCAST object identification if a sender retains some older objects in newer Carousel Instances with updated object sets. To avoid ambiguity the active TOIs (i.e., the TOIs corresponding to objects being transmitted) can only occupy half of the TOI sequence space. If an old object, whose TOI has fallen outside the current window, needs to be transmitted again, a new TOI must be used for it. In case of NORM, this constraint limits to 32768 the maximum number of objects that can be part of any carousel instance. In order to allow receivers to properly combine the transport packets with a newly-assigned TOI to those of associated to the previously-assigned TOI, a mechanism is required to equate the objects with the new and the old TOIs.

The preferred mechanism consists in signaling, within the CID, that the newly assigned TOI, for the current Carousel Instance, is equivalent to the TOI used within a previous Carousel Instance. By convention, the reference tuple for any object is the {TOI; CI ID} tuple used for its first transmission within a Carousel Instance. This tuple MUST be used whenever a TOI equivalence is provided.

An alternative solution, when meta-data can be processed rapidly (e.g., by using NORM-INFO messages), consists for the receiver in identifying that both objects are the same, after examining the meta-data. The receiver can then take appropriate measures.



### 3.7. FCAST Sender Behavior

The following operations MAY take place at a sender:

1. The user (or another application) selects a set of objects (e.g., files) to deliver and submits them, along with their meta-data, to the FCAST application;
2. For each object, FCAST creates the Compound Object and registers this latter in the carousel instance;
3. The user then informs FCAST that all the objects of the set have been submitted. If the user knows that no new object will be submitted in the future (i.e., if the session's content is now complete), the user informs FCAST. Finally, the user specifies how many transmission cycles are desired (this number may be infinite);
4. At this point, the FCAST application knows the full list of Compound Objects that are part of the Carousel Instance and can create a CID if desired, possibly with the complete flag set;
5. The FCAST application can now define a transmission schedule of these Compound Objects, including the optional CID. This schedule defines in which order the packets of the various Compound Objects should be sent. This document does not specify any scheme. This is left to the developer within the provisions of the underlying ALC or NORM protocol used and the knowledge of the target use-case.
6. The FCAST application then starts the carousel transmission, for the number of cycles specified. Transmissions take place until:
  - \* the desired number of transmission cycles has been reached, or
  - \* the user wants to prematurely stop the transmissions, or
  - \* the user wants to add one or several new objects to the carousel, or on the contrary wants to remove old objects from the carousel. In that case a new carousel instance must be created.
7. If the session is not finished, then continue at Step 1 above;

### 3.8. FCAST Receiver Behavior

The following operations MAY take place at a receiver:

1. The receiver joins the session and collects incoming packets;
2. If the header portion of a Compound Object is entirely received (which may happen before receiving the entire object with some ALC/NORM configurations), or if the meta-data is sent by means of another mechanism prior to the object, the receiver processes the meta-data and chooses to continue to receive the object content or not;
3. When a Compound Object has been entirely received, the receiver processes the header, retrieves the object meta-data, perhaps decodes the meta-data, and processes the object accordingly;
4. When a CID is received, which is indicated by the 'C' flag set in the Compound Object header, the receiver decodes the CID, and retrieves the list of objects that are part of the current carousel instance. This list CAN be used to remove objects sent in a previous carousel instance that might not have been totally decoded and that are no longer part of the current carousel instance;
5. When a CID is received, the receiver also retrieves the list of TOI equivalences, if any, and takes appropriate measures, for instance by informing the transport layer;
6. When a receiver has received a CID with the "Complete" flag set, and has successfully received all the objects of the current carousel instance, it can safely exit from the current FCAST session;
7. Otherwise continue at Step 2 above.

## 4. Security Considerations

### 4.1. Problem Statement

A content delivery system is potentially subject to attacks. Attacks may target:

- o the network (to compromise the routing infrastructure, e.g., by creating congestion),

- o the Content Delivery Protocol (CDP) (e.g., to compromise the normal behavior of FCAST), or
- o the content itself (e.g., to corrupt the objects being transmitted).

These attacks can be launched either:

- o against the data flow itself (e.g., by sending forged packets),
- o against the session control parameters (e.g., by corrupting the session description, the CID, the object meta-data, or the ALC/LCT control parameters), that are sent either in-band or out-of-band, or
- o against some associated building blocks (e.g., the congestion control component).

In the following sections we provide more details on these possible attacks and sketch some possible counter-measures. We finally provide recommendations in Section 4.5.

#### 4.2. Attacks Against the Data Flow

Let us consider attacks against the data flow first. At least, the following types of attacks exist:

- o attacks that are meant to give access to a confidential object (e.g., in case of a non-free content) and
- o attacks that try to corrupt the object being transmitted (e.g., to inject malicious code within an object, or to prevent a receiver from using an object, which is a kind of Denial of Service (DoS)).

##### 4.2.1. Access to Confidential Objects

Access control to the object being transmitted is typically provided by means of encryption. This encryption can be done over the whole object (e.g., by the content provider, before submitting the object to FCAST), or be done on a packet per packet basis (e.g., when IPsec/ESP is used [RFC4303], see Section 4.5). If confidentiality is a concern, it is RECOMMENDED that one of these solutions be used.

##### 4.2.2. Object Corruption

Protection against corruptions (e.g., if an attacker sends forged packets) is achieved by means of a content integrity verification/sender authentication scheme. This service can be provided at the

object level, but in that case a receiver has no way to identify which symbol(s) is(are) corrupted if the object is detected as corrupted. This service can also be provided at the packet level. In this case, after removing all corrupted packets, the file may be in some cases recovered. Several techniques can provide this content integrity/sender authentication service:

- o at the object level, the object can be digitally signed, for instance by using RSASSA-PKCS1-v1\_5 [RFC3447]. This signature enables a receiver to check the object integrity, once this latter has been fully decoded. Even if digital signatures are computationally expensive, this calculation occurs only once per object, which is usually acceptable;
- o at the packet level, each packet can be digitally signed [RMT-SIMPLE-AUTH]. A major limitation is the high computational and transmission overheads that this solution requires. To avoid this problem, the signature may span a set of packets (instead of a single one) in order to amortize the signature calculation. But if a single packets is missing, the integrity of the whole set cannot be checked;
- o at the packet level, a Group Message Authentication Code (MAC) [RFC2104][RMT-SIMPLE-AUTH] scheme can be used, for instance by using HMAC-SHA-256 with a secret key shared by all the group members, senders and receivers. This technique creates a cryptographically secured digest of a packet that is sent along with the packet. The Group MAC scheme does not create prohibitive processing load nor transmission overhead, but it has a major limitation: it only provides a group authentication/integrity service since all group members share the same secret group key, which means that each member can send a forged packet. It is therefore restricted to situations where group members are fully trusted (or in association with another technique as a pre-check);
- o at the packet level, Timed Efficient Stream Loss-Tolerant Authentication (TESLA) [RFC4082][RFC5776] is an attractive solution that is robust to losses, provides a true authentication/integrity service, and does not create any prohibitive processing load or transmission overhead. Yet checking a packet requires a small delay (a second or more) after its reception;
- o at the packet level, IPsec/ESP [RFC4303] can be used to check the integrity and authenticate the sender of all the packets being exchanged in a session (see Section 4.5).

Techniques relying on public key cryptography (digital signatures and TESLA during the bootstrap process, when used) require that public

keys be securely associated to the entities. This can be achieved by a Public Key Infrastructure (PKI), or by a PGP Web of Trust, or by pre-distributing securely the public keys of each group member.

Techniques relying on symmetric key cryptography (Group MAC) require that a secret key be shared by all group members. This can be achieved by means of a group key management protocol, or simply by pre-distributing securely the secret key (but this manual solution has many limitations).

It is up to the developer and deployer, who know the security requirements and features of the target application area, to define which solution is the most appropriate. In any case, whenever there is any concern of the threat of file corruption, it is RECOMMENDED that at least one of these techniques be used.

#### 4.3. Attacks Against the Session Control Parameters and Associated Building Blocks

Let us now consider attacks against the session control parameters and the associated building blocks. The attacker has at least the following opportunities to launch an attack:

- o the attack can target the session description,
- o the attack can target the FCAST CID,
- o the attack can target the meta-data of an object,
- o the attack can target the ALC/LCT parameters, carried within the LCT header or
- o the attack can target the FCAST associated building blocks, for instance the multiple rate congestion control protocol.

The consequences of these attacks are potentially serious, since they can compromise the behavior of content delivery system or even compromise the network itself.

##### 4.3.1. Attacks Against the Session Description

An FCAST receiver may potentially obtain an incorrect Session Description for the session. The consequence of this is 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 possibly disrupting other traffic in the network.

To avoid these problems, it is RECOMMENDED that measures be taken to prevent receivers from accepting incorrect Session Descriptions. One such measure is the sender authentication to ensure that receivers only accept legitimate Session Descriptions from authorized senders. How these measures are achieved is outside the scope of this document since this session description is usually carried out-of-band.

#### 4.3.2. Attacks Against the FCAST CID

Corrupting the FCAST CID is one way to create a Denial of Service attack. For example, the attacker can set the "Complete" flag to make the receivers believe that no further modification will be done.

It is therefore RECOMMENDED that measures be taken to guarantee the integrity and to check the sender's identity of the CID. To that purpose, one of the counter-measures mentioned above (Section 4.2.2) SHOULD be used. These measures will either be applied on a packet level, or globally over the whole CID object. When there is no packet level integrity verification scheme, it is RECOMMENDED to digitally sign the CID.

#### 4.3.3. Attacks Against the Object Meta-Data

Corrupting the object meta-data is another way to create a Denial of Service attack. For example, the attacker changes the MD5 sum associated to a file. This possibly leads a receiver to reject the files received, no matter whether the files have been correctly received or not. When the meta-data are appended to the object, corrupting the meta-data means that the Compound Object will be corrupted.

It is therefore RECOMMENDED that measures be taken to guarantee the integrity and to check the sender's identity of the Compound Object. To that purpose, one of the counter-measures mentioned above (Section 4.2.2) SHOULD be used. These measures will either be applied on a packet level, or globally over the whole Compound Object. When there is no packet level integrity verification scheme, it is RECOMMENDED to digitally sign the Compound Object.

#### 4.3.4. Attacks Against the ALC/LCT and NORM Parameters

By corrupting the ALC/LCT header (or header extensions) one can execute attacks on the underlying ALC/LCT implementation. For example, sending forged ALC packets with the Close Session flag (A) set to one can lead the receiver to prematurely close the session. Similarly, sending forged ALC packets with the Close Object flag (B) set to one can lead the receiver to prematurely give up the reception of an object. The same comments can be made for NORM.

It is therefore RECOMMENDED that measures be taken to guarantee the integrity and to check the sender's identity of each ALC or NORM packet received. To that purpose, one of the counter-measures mentioned above (Section 4.2.2) SHOULD be used.

#### 4.3.5. Attacks Against the Associated Building Blocks

Let us first focus on the congestion control building block that may be used in an ALC or NORM session. A receiver with an incorrect or corrupted implementation of the multiple rate congestion control building block may affect the health of the network in the path between the sender and the receiver. That may also affect the reception rates of other receivers who joined the session.

When congestion control is applied with FCAST, it is therefore RECOMMENDED that receivers be required to identify themselves as legitimate before they receive the Session Description needed to join the session. If authenticating a receiver does not prevent this latter to launch an attack, it will enable the network operator to identify him and to take counter-measures. This authentication can be made either toward the network operator or the session sender (or a representative of the sender) in case of NORM. The details of how it is done are outside the scope of this document.

When congestion control is applied with FCAST, it is also RECOMMENDED that a packet level authentication scheme be used, as explained in Section 4.2.2. Some of them, like TESLA, only provide a delayed authentication service, whereas congestion control requires a rapid reaction. It is therefore RECOMMENDED [RFC5775] that a receiver using TESLA quickly reduces its subscription level when the receiver believes that a congestion did occur, even if the packet has not yet been authenticated. Therefore TESLA will not prevent DoS attacks where an attacker makes the receiver believe that a congestion occurred. This is an issue for the receiver, but this will not compromise the network. Other authentication methods that do not feature this delayed authentication could be preferred, or a group MAC scheme could be used in parallel to TESLA to prevent attacks launched from outside of the group.

#### 4.4. Other Security Considerations

Lastly, we note that the security considerations that apply to, and are described in, ALC [RFC5775], LCT [RFC5651], NORM [RFC5740] and FEC [RFC5052] also apply to FCAST as FCAST builds on those specifications. In addition, any security considerations that apply to any congestion control building block used in conjunction with FCAST also applies to FCAST. Finally, the security discussion of [RMT-SEC] also applies here.

#### 4.5. Minimum Security Recommendations

We now introduce a mandatory to implement but not necessarily to use security configuration, in the sense of [RFC3365]. Since FCAST/ALC relies on ALC/LCT, it inherits the "baseline secure ALC operation" of [RFC5775]. Similarly, since FCAST/NORM relies on NORM, it inherits the "baseline secure NORM operation" of [RFC5740]. More precisely, in both cases security is achieved by means of IPsec/ESP in transport mode. [RFC4303] explains that ESP can be used to potentially provide confidentiality, data origin authentication, content integrity, anti-replay and (limited) traffic flow confidentiality. [RFC5775] specifies that the data origin authentication, content integrity and anti-replay services SHALL be used, and that the confidentiality service is RECOMMENDED. If a short lived session MAY rely on manual keying, it is also RECOMMENDED that an automated key management scheme be used, especially in case of long lived sessions.

Therefore, the RECOMMENDED solution for FCAST provides per-packet security, with data origin authentication, integrity verification and anti-replay. This is sufficient to prevent most of the in-band attacks listed above. If confidentiality is required, a per-packet encryption SHOULD also be used.

#### 5. Requirements for Compliant Implementations

This section lists the features that any compliant FCAST/ALC or FCAST/NORM implementation MUST support, and those that remain OPTIONAL, e.g., in order to enable some optimizations for a given use-case, at a receiver.

##### 5.1. Requirements Related to the Object Meta-Data

Meta-data transmission mechanisms:

Feature	Status
meta-data transmission using FCAST's in-band mechanism	An FCAST sender MUST send meta-data with the in-band mechanism provided by FCAST, i.e., within the Compound Object header. All the FCAST receivers MUST be able to process meta-data sent with this FCAST in-band mechanism.



meta-data transmission using other mechanisms	In addition to the FCAST in-band transmission of meta-data, an FCAST sender MAY send a subset or all of the meta-data using another mechanism. Supporting this mechanism in a compliant FCAST receiver is OPTIONAL, and its use is OPTIONAL too. An FCAST receiver MAY support this mechanism and take advantage of the meta-data sent in this way. If it is not the case, the FCAST receiver will anyway receive and process meta-data sent in-bound. See Annex Appendix B.
---	--

## Meta-data format and encoding:

Feature	Status
Meta-Data Format (MDFmt field)	All FCAST implementations MUST support an HTTP/1.1 metainformation format [RFC2616]. Other formats (e.g., XML) MAY be defined in the future.
Meta-Data Encoding (MDEnc field)	All FCAST implementations MUST support both a plain text and a GZIP encoding [RFC1952] of the Object Meta-Data field. Other encodings MAY be defined in the future.

## Meta-data items (Section 3.3):

Feature	Status
Content-Location	MUST be supported
Content-Type	MUST be supported
Content-Length	MUST be supported
Content-Encoding	MUST be supported
Content-MD5	MUST be supported
Fcast-Obj-Slice-Nb	SHOULD be supported
Fcast-Obj-Slice-Idx	SHOULD be supported
Fcast-Obj-Slice-Offset	SHOULD be supported

## 5.2. Requirements Related to the Carousel Instance Descriptor (CID) Mechanism

Any compliant FCAST implementation MUST support the CID mechanism, in order to list the Compound Objects that are part of a given Carousel Instance. However its use is OPTIONAL.

## 6. Operational Considerations

FCAST is compatible with any congestion control protocol designed for ALC/LCT or NORM. However, depending on the use-case, the data flow generated by the FCAST application might not be constant, but instead be bursty in nature. Similarly, depending on the use-case, an FCAST session might be very short. Whether and how this will impact the congestion control protocol is out of the scope of the present document.

FCAST is compatible with any security mechanism designed for ALC/LCT or NORM. The use of a security scheme is strongly RECOMMENDED (see Section 4).

FCAST is compatible with any FEC scheme designed for ALC/LCT or NORM. Whether FEC is used or not, and the kind of FEC scheme used, is to some extent transparent to FCAST.

FCAST is compatible with both IPv4 and IPv6. Nothing in the FCAST specification has any implication on the source or destination IP address type.

The delivery service provided by FCAST might be fully reliable, or only partially reliable depending upon:

- o the way ALC or NORM is used (e.g., whether FEC encoding and/or NACK-based repair requests are used or not),
- o the way the FCAST carousel is used (e.g., whether the objects are made available for a long time span or not), and
- o the way in which FCAST itself is employed (e.g., whether there is a session control application that might automatically extend an existing FCAST session until all receivers have received the transmitted content).

The receiver set MAY be restricted to a single receiver or MAY include possibly a very large number of receivers. While the choice of the underlying transport protocol (i.e., ALC or NORM) and its parameters may limit the practical receiver group size, nothing in

FCAST itself limits it. For instance, if FCAST/ALC is used on top of purely unidirectional transport channels, with no feedback information at all, which is the default mode of operation, then the scalability is maximum since neither FCAST, nor ALC, UDP or IP generates any feedback message. On the contrary, the FCAST/NORM scalability is typically limited by NORM scalability itself. Similarly, if FCAST is used along with a session control application that collects reception information from the receivers, then this session control application may limit the scalability of the global object delivery system. This situation can of course be mitigated by using a hierarchy of feedback message aggregators or servers. The details of this are out of the scope of the present document.

The content of a carousel instance MAY be described by means of an OPTIONAL Carousel Instance Descriptor (CID) (Section 3.5). The decisions of whether a CID should be used or not, how often and when a CID should be sent, are left to the sender and depend on many parameters, including the target use case and the session dynamics. For instance it may be appropriate to send a CID at the beginning of each new carousel instance, and then periodically. These operational aspects are out of the scope of the present document.

## 7. IANA Considerations

### 7.1. Namespace declaration for Object Meta-Data Format

This document requires a IANA registration for the following namespace: "Object Meta-Data Format" (MDFmt). Values in this namespace are 4-bit positive integers between 0 and 15 inclusive and they define the format of the object meta-data ((see Section 2.1).

Initial values for the LCT Header Extension Type registry are defined in Section 7.1.1. Future assignments are to be made through Expert Review [RFC5226].

#### 7.1.1. Object Meta-Data Format registration

This document registers one value in the "Object Meta-Data Format" namespace as follows:

format name	Value
as per HTTP/1.1 metainformation format	0 (default)

## 7.2. Namespace declaration for Object Meta-Data Encoding

This document requires a IANA registration for the following namespace: "Object Meta-Data Encoding" (MDEnc). Values in this namespace are 4-bit positive integers between 0 and 15 inclusive and they define the optional encoding of the Object Meta-Data field (see Section 2.1).

Initial values for the LCT Header Extension Type registry are defined in Section 7.2.1. Future assignments are to be made through Expert Review [RFC5226].

### 7.2.1. Object Meta-Data Encoding registration

This document registers two values in the "Object Meta-Data Encoding" namespace as follows:

Name	Value
plain text	0 (default)
GZIP	1

## 8. Acknowledgments

The authors are grateful to the authors of [ALC-00] for specifying the first version of FCAST/ALC. The authors are also grateful to David Harrington, Gorrry Fairhurst and Lorenzo Vicisano for their valuable comments.

## 9. References

### 9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC1071] Braden, R., Borman, D., Partridge, C., and W. Plummer, "Computing the Internet checksum", RFC 1071, September 1988.

- [RFC5651] Luby, M., Watson, M., and L. Vicisano, "Layered Coding Transport (LCT) Building Block", RFC 5651, October 2009.
- [RFC5740] Adamson, B., Bormann, C., Handley, M., and J. Macker, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol", RFC 5740, November 2009.
- [RFC5775] Luby, M., Watson, M., and L. Vicisano, "Asynchronous Layered Coding (ALC) Protocol Instantiation", RFC 5775, April 2010.
- [RFC1952] Deutsch, P., Gailly, J-L., Adler, M., Deutsch, L., and G. Randers-Pehrson, "GZIP file format specification version 4.3", RFC 1952, May 1996.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

## 9.2. Informative References

- [ALC-00] Luby, M., Gemmell, G., Vicisano, L., Crowcroft, J., and B. Lueckenhoff, "Asynchronous Layered Coding: a Scalable Reliable Multicast Protocol", March 2000.
- [RMT-FLUTE] Paila, T., Walsh, R., Luby, M., Roca, V., and R. Lehtonen, "FLUTE - File Delivery over Unidirectional Transport", Work in Progress, February 2011.
- [RFC3365] Schiller, J., "Strong Security Requirements for Internet Engineering Task Force Standard Protocols", BCP 61, RFC 3365, August 2002.
- [RMT-SEC] Roca, V., Adamson, B., and H. Asaeda, "Security and Reliable Multicast Transport Protocols: Discussions and Guidelines", Work in progress, draft-ietf-rmt-sec-discussion-06.txt, March 2011.
- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.
- [RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.
- [RFC4082] Perrig, A., Song, D., Canetti, R., Tygar, J., and B.

Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction", RFC 4082, June 2005.

- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.
- [RFC5052] Watson, M., Luby, M., and L. Vicisano, "Forward Error Correction (FEC) Building Block", RFC 5052, August 2007.
- [RFC5510] Lacan, J., Roca, V., Peltotalo, J., and S. Peltotalo, "Reed-Solomon Forward Error Correction (FEC) Schemes", RFC 5510, April 2009.
- [RFC5776] Roca, V., Francillon, A., and S. Faurite, "Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Asynchronous Layered Coding (ALC) and NACK-Oriented Reliable Multicast (NORM) Protocols", RFC 5776, April 2010.
- [RMT-SIMPLE-AUTH] Roca, V., "Simple Authentication Schemes for the ALC and NORM Protocols", Work in progress draft-ietf-rmt-simple-auth-for-alc-norm-05.txt, September 2011.

## Appendix A. FCAST Examples

### A.1. Regular Compound Object Example

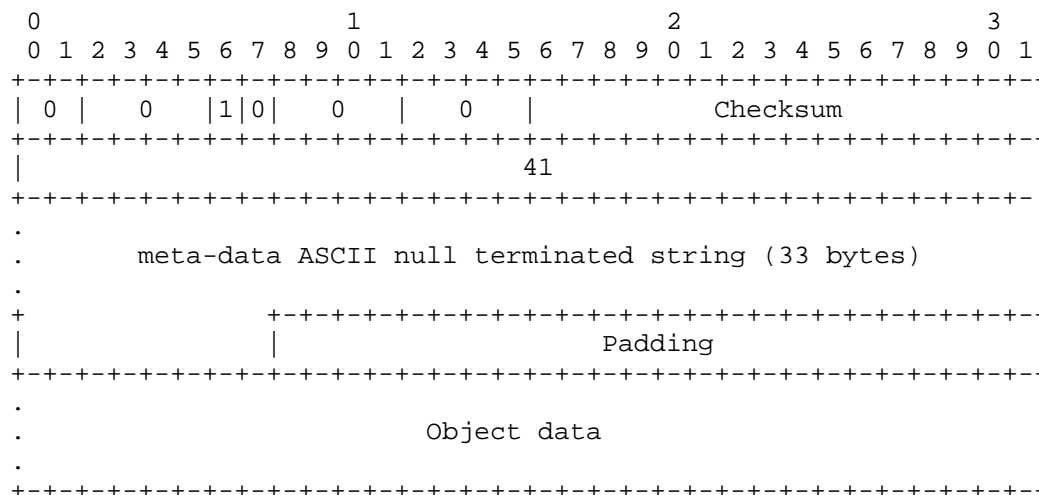


Figure 4: Compound Object Example.

Figure 4 shows a regular Compound Object where the meta-data ASCII string, in HTTP/1.1 meta-information format (MDFmt=0) contains:

```
Content-Location: example.txt <CR-LF>
```

This string is 33 bytes long, including the NULL-termination character. There is no GZIP encoding of the meta-data (MDEnc=0) and there is no Content-Length information either since this length can easily be calculated by the receiver as the FEC OTI transfer length minus the header length. Finally, the checksum encompasses the whole Compound Object (G=1).

#### A.2. Carousel Instance Descriptor Example

Figure 5 shows an example CID object, in the case of a static FCAST session, i.e., a session where the set of objects is set once and for all. There is no meta-data in this example since Fcast-CID-Complete and Fcast-CID-ID are both implicit.

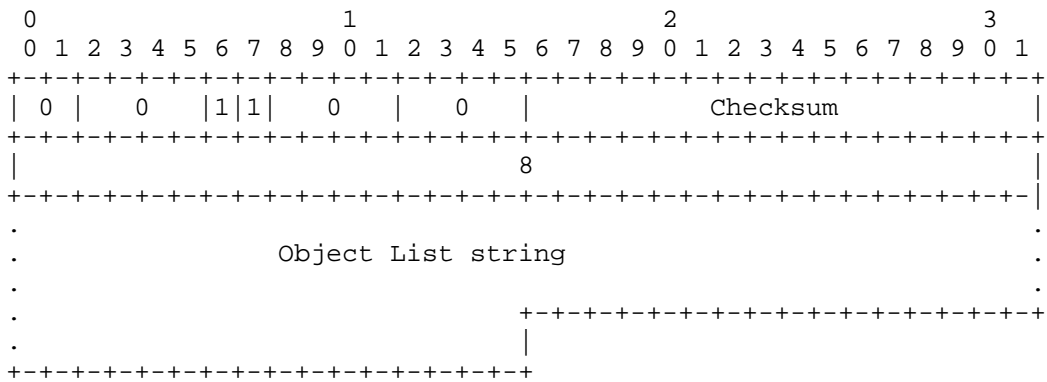


Figure 5: Example of CID, in case of a static session.

The object list contains the following 26 byte long string, including the NULL-termination character:

1,2,3,100-104,200-203,299

There are therefore a total of  $3+5+4+1 = 13$  objects in the carousel instance, and therefore in the FCAST session. There is no meta-data associated to this CID. The session being static and composed of a single Carousel Instance, the sender did not feel the necessity to carry a Carousel Instance ID meta-data.

Appendix B. Additional Meta-Data Transmission Mechanisms

B.1. Supporting Additional Mechanisms

In certain use-cases, FCAST MAY take advantage of another in-band (e.g., via NORM INFO messages Appendix B.2) or out-of-band signaling mechanism. This additional signaling scheme MAY be used to carry the whole meta-data, or a subset of it, ahead of time, before the associated compound object. Therefore a receiver may be able to decide in advance, before beginning the reception of the compound object, whether the object is of interest or not, based on the information retrieved by this way, which mitigates FCAST limitations. If out-of-band techniques are out of the scope of this document, we explain below how this may be achieved in case of FCAST/NORM.

Supporting one of these additional mechanisms is OPTIONAL in FCAST implementations. An FCAST/NORM sender MUST continue to send all the required meta-data in the compound object, even if the whole meta-data, or a subset of it, is sent by another mechanism (Section 5). Additionally, when meta-data is sent several times, there MUST NOT be



any contradiction in the information provided by the different mechanisms. In case a mismatch is detected, the meta-data contained in the Compound Object MUST be privileged.

When meta-data elements are communicated out-of-band, in advance of data transmission, the following piece of information MAY be useful:

- o TOI: a positive integer that contains the Transmission Object Identifier (TOI) of the object, in order to enable a receiver to easily associate the meta-data to the object. The valid range for TOI values is discussed in Section 3.6;

## B.2. Using NORM\_INFO Messages with FCAST/NORM

The NORM\_INFO message of NORM can convey "out-of-band" content with respect to a given transport object. With FCAST, this message MAY be used as an additional mechanism to transmit meta-data. In that case, the NORM\_INFO message carries a new Compound Object that contains all the meta-data of the original object, or a subset of it. The NORM\_INFO Compound Object MUST NOT contain any Object Data field (i.e., it is only composed of the header), it MUST feature a non global checksum, and it MUST NOT include any padding field. Finally, note that the availability of NORM\_INFO for a given object is signaled through the use of a dedicated flag in the NORM\_DATA message header. Along with NORM's NACK-based repair request signaling, it allows a receiver to quickly (and independently) request an object's NORM\_INFO content. However, a limitation here is that the NORM\_INFO Compound Object header MUST fit within the byte size limit defined by the NORM sender's configured "segment size" (typically a little less than the network MTU);

### B.2.1. Example

In case of FCAST/NORM, the FCAST Compound Object meta-data (or a subset of it) can be carried as part of a NORM\_INFO message, as a new Compound Object that does not contain any Compound Object Data. In the following example we assume that the whole meta-data is carried in such a message for a certain Compound Object. Figure 6 shows an example NORM\_INFO message that contains the FCAST Compound Object Header and meta-data as its payload. In this example, the first 16 bytes are the NORM\_INFO base header, the next 12 bytes are a NORM\_EXT\_FTI header extension containing the FEC Object Transport Information for the associated object, and the remaining bytes are the FCAST Compound Object Header and meta-data. Note that "padding" MUST NOT be used and that the FCAST checksum only encompasses the Compound Object Header (G=0).

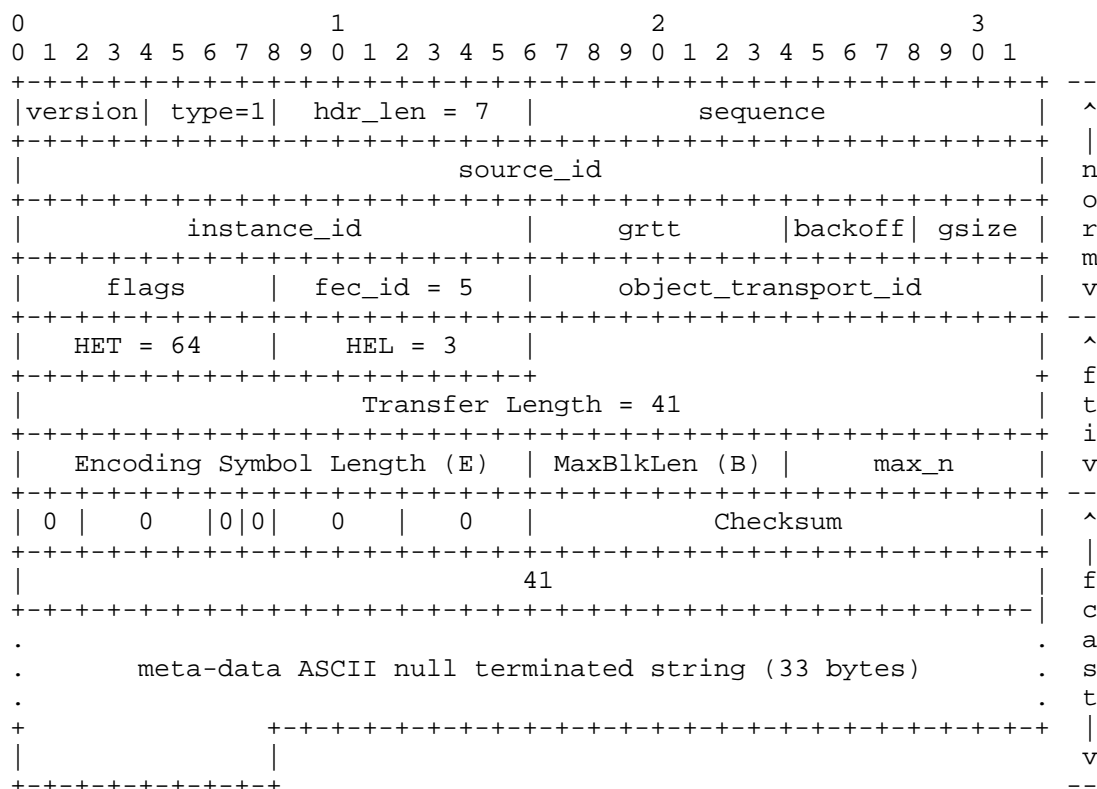


Figure 6: NORM\_INFO containing an EXT\_FTI header extension and an FCAST Compound Object Header

The NORM\_INFO message shown in Figure 6 contains the EXT\_FTI header extension to carry the FEC OTI. In this example, the FEC OTI format is that of the Reed-Solomon FEC coding scheme for fec\_id = 5 as described in [RFC5510]. Other alternatives for providing the FEC OTI would have been to either include it directly in the meta-data of the FCAST Compound Header, or to include an EXT\_FTI header extension to all NORM\_DATA packets (or a subset of them). Note that the NORM "Transfer\_Length" is the total length of the associated FCAST Compound Object, i.e., 41 bytes.

The FCAST Compound Object in this example does contain the same meta-data and is formatted as in the example of Figure 4. With the combination of the FEC\_OTI and the FCAST meta-data, the NORM protocol and FCAST application have all of the information needed to reliably receive and process the associated object. Indeed, the NORM protocol provides rapid (NORM\_INFO has precedence over the associated object content), reliable delivery of the NORM\_INFO message and its payload,

the FCAST Compound Object.

#### Authors' Addresses

Vincent Roca  
INRIA  
655, av. de l'Europe  
Inovallee; Montbonnot  
ST ISMIER cedex 38334  
France

Email: [vincent.roca@inria.fr](mailto:vincent.roca@inria.fr)  
URI: <http://planete.inrialpes.fr/people/roca/>

Brian Adamson  
Naval Research Laboratory  
Washington, DC 20375  
USA

Email: [adamson@itd.nrl.navy.mil](mailto:adamson@itd.nrl.navy.mil)  
URI: <http://cs.itd.nrl.navy.mil>



Reliable Multicast Transport (RMT)  
Internet-Draft  
Obsoletes: 3926 (if approved)  
Intended status: Standards Track  
Expires: September 13, 2012

T. Paila  
Nokia  
R. Walsh  
Tampere University of Technology  
M. Luby  
Qualcomm, Inc.  
V. Roca  
INRIA  
R. Lehtonen  
TeliaSonera  
March 12, 2012

FLUTE - File Delivery over Unidirectional Transport  
draft-ietf-rmt-flute-revised-14

Abstract

This document defines FLUTE, a protocol for the unidirectional delivery of files over the Internet, which is particularly suited to multicast networks. The specification builds on Asynchronous Layered Coding, the base protocol designed for massively scalable multicast distribution. This document obsoletes RFC3926.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on September 13, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

## Table of Contents

1. Introduction . . . . .	5
1.1. Applicability Statement . . . . .	6
1.1.1. The Target Application Space . . . . .	6
1.1.2. The Target Scale . . . . .	6
1.1.3. Intended Environments . . . . .	7
1.1.4. Weaknesses . . . . .	7
2. Conventions used in this Document . . . . .	8
3. File delivery . . . . .	8
3.1. File delivery session . . . . .	9
3.2. File Delivery Table . . . . .	11
3.3. Dynamics of FDT Instances within file delivery session . . . . .	13
3.4. Structure of FDT Instance packets . . . . .	16
3.4.1. Format of FDT Instance Header . . . . .	17
3.4.2. Syntax of FDT Instance . . . . .	18
3.4.3. Content Encoding of FDT Instance . . . . .	22
3.5. Multiplexing of files within a file delivery session . . . . .	23
4. Channels, congestion control and timing . . . . .	23
5. Delivering FEC Object Transmission Information . . . . .	25
6. Describing file delivery sessions . . . . .	26
7. Security Considerations . . . . .	28
7.1. Problem Statement . . . . .	28
7.2. Attacks against the data flow . . . . .	28
7.2.1. Access to confidential files . . . . .	28
7.2.2. File corruption . . . . .	29
7.3. Attacks against the session control parameters and associated Building Blocks . . . . .	30
7.3.1. Attacks against the Session Description . . . . .	31
7.3.2. Attacks against the FDT Instances . . . . .	31
7.3.3. Attacks against the ALC/LCT parameters . . . . .	32
7.3.4. Attacks against the associated Building Blocks . . . . .	32
7.4. Other Security Considerations . . . . .	33
7.5. Minimum Security Recommendations . . . . .	33
8. IANA Considerations . . . . .	33
8.1. Registration Request for XML Schema of FDT Instance . . . . .	34
8.2. Media-Type Registration Request for application/fdt+xml . . . . .	34
8.3. Content Encoding Algorithm Registration Request . . . . .	35
8.3.1. Explicit IANA Assignment Guidelines . . . . .	35
8.4. Registration of EXT_FDT LCT Header Extension Type . . . . .	36
8.5. Registration of EXT_CENC LCT Header Extension Type . . . . .	36
9. Acknowledgments . . . . .	36
10. Contributors . . . . .	36
11. Change Log . . . . .	37
11.1. RFC3926 to draft-ietf-rmt-flute-revised-12 . . . . .	37
12. References . . . . .	40
12.1. Normative references . . . . .	40
12.2. Informative references . . . . .	41

Appendix A. Receiver operation (informative) . . . . .	43
Appendix B. Example of FDT Instance (informative) . . . . .	44
Authors' Addresses . . . . .	44



## 1. Introduction

This document defines FLUTE version 2, a protocol for unidirectional delivery of files over the Internet. This specification is not backwards compatible with the previous experimental version defined in [RFC3926] (see Section 11 for details). The specification builds on Asynchronous Layered Coding (ALC), version 1 [RFC5775], the base protocol designed for massively scalable multicast distribution. ALC defines transport of arbitrary binary objects. For file delivery applications mere transport of objects is not enough, however. The end systems need to know what the objects actually represent. This document specifies a technique called FLUTE - a mechanism for signaling and mapping the properties of files to concepts of ALC in a way that allows receivers to assign those parameters for received objects. Consequently, throughout this document the term 'file' relates to an 'object' as discussed in ALC. Although this specification frequently makes use of multicast addressing as an example, the techniques are similarly applicable for use with unicast addressing.

This document defines a specific transport application of ALC, adding the following specifications:

- Definition of a file delivery session built on top of ALC, including transport details and timing constraints.
- In-band signaling of the transport parameters of the ALC session.
- In-band signaling of the properties of delivered files.
- Details associated with the multiplexing of multiple files within a session.

This specification is structured as follows. Section 3 begins by defining the concept of the file delivery session. Following that it introduces the File Delivery Table that forms the core part of this specification. Further, it discusses multiplexing issues of transmission objects within a file delivery session. Section 4 describes the use of congestion control and channels with FLUTE. Section 5 defines how the Forward Error Correction (FEC) Object Transmission Information is to be delivered within a file delivery session. Section 6 defines the required parameters for describing file delivery sessions in a general case. Section 7 outlines security considerations regarding file delivery with FLUTE. Last, there are two informative appendices. Appendix A describes an envisioned receiver operation for the receiver of the file delivery session. Readers who want to see a simple example of FLUTE in operation should refer to Appendix A right away. Appendix B gives an

example of a File Delivery Table.

This specification contains part of the definitions necessary to fully specify a Reliable Multicast Transport protocol in accordance with [RFC2357].

This document obsoletes [RFC3926] which contained a previous version of this specification and was published in the "Experimental" category. This Proposed Standard specification is thus based on [RFC3926] updated according to accumulated experience and growing protocol maturity since the publication of [RFC3926]. Said experience applies both to this specification itself and to congestion control strategies related to the use of this specification.

The differences between [RFC3926] and this document are listed in Section 11.

This document updates ALC [RFC5775] and Layered Coding Transport (LCT) [RFC5651] in the sense it defines two new header extensions, EXT\_FDT and EXT\_CENC.

## 1.1. Applicability Statement

### 1.1.1. The Target Application Space

FLUTE is applicable to the delivery of large and small files to many hosts, using delivery sessions of several seconds or more. For instance, FLUTE could be used for the delivery of large software updates to many hosts simultaneously. It could also be used for continuous, but segmented, data such as time-lined text for subtitling - potentially leveraging its layering inheritance from ALC and LCT to scale the richness of the session to the congestion status of the network. It is also suitable for the basic transport of metadata, for example SDP [RFC4566] files which enable user applications to access multimedia sessions.

### 1.1.2. The Target Scale

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

### 1.1.1.3. Intended Environments

All of the environmental requirements and considerations that apply to the RMT Building Blocks used by FLUTE shall also apply to FLUTE. These are the ALC protocol instantiation [RFC5775], the Layered Coding Transport (LCT) Building Block [RFC5651] and the FEC Building Block [RFC5052].

FLUTE can be used with both multicast and unicast delivery, but it's primary application is for unidirectional multicast file delivery. FLUTE requires connectivity between a sender and receivers but does not require connectivity from receivers to a sender. Because of its low expectations, FLUTE works with most types of networks, including LANs, WANs, Intranets, the Internet, asymmetric networks, wireless networks, and satellite networks.

FLUTE is compatible with both IPv4 or IPv6 as no part of the packet is IP version specific. FLUTE works with both multicast models: Any-Source Multicast (ASM) [RFC1112] and the Source-Specific Multicast (SSM) [PAPER.SSM].

FLUTE is applicable for both Internet use, with a suitable congestion control building block, and provisioned/controlled systems, such as delivery over wireless broadcast radio systems.

### 1.1.1.4. Weaknesses

FLUTE congestion control protocols depend on the ability of a receiver to change multicast subscriptions between multicast groups supporting different rates and/or layered codings. If the network does not support this, then the FLUTE congestion control protocols may not be amenable to these networks.

FLUTE can also be used for point-to-point (unicast) communications. At a minimum, implementations of ALC MUST support the Wave and Equation Based Rate Control (WEBRC) [RFC3738] multiple rate congestion control scheme [RFC5775]. However, since WEBRC has been designed for massively scalable multicast flows, it is not clear how appropriate it is to the particular case of unicast flows. Using a separate point-to-point congestion control scheme is another alternative. How to do that is outside the scope of the present document.

FLUTE provides reliability using the FEC building block. This will reduce the error rate as seen by applications. However, FLUTE does not provide a method for senders to verify the reception success of receivers, and the specification of such a method is outside the scope of this document.

## 2. Conventions used in this Document

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].

The terms "object" and "transmission object" are consistent with the definitions in ALC [RFC5775] and LCT [RFC5651]. The terms "file" and "source object" are pseudonyms for "object".

## 3. File delivery

Asynchronous Layered Coding [RFC5775] is a protocol designed for delivery of arbitrary binary objects. It is especially suitable for massively scalable, unidirectional, multicast distribution. ALC provides the basic transport for FLUTE, and thus FLUTE inherits the requirements of ALC.

This specification is designed for the delivery of files. The core of this specification is to define how the properties of the files are carried in-band together with the delivered files.

As an example, let us consider a 5200 byte file referred to by "http://www.example.com/docs/file.txt". Using the example, the following properties describe the properties that need to be conveyed by the file delivery protocol.

- \* Identifier of the file, expressed as a URI [RFC3986]. The identifier MAY provide a location for the file. In the above example: "http://www.example.com/docs/file.txt".
- \* File name (usually, this can be concluded from the URI). In the above example: "file.txt".
- \* File type, expressed as Internet Media Types (often referred to as "MIME Media Types"). In the above example: "text/plain".
- \* File size, expressed in octets. In the above example: "5200". If the file is content encoded then this is the file size before content encoding.
- \* Content encoding of the file, within transport. In the above example, the file could be encoded using ZLIB [RFC1950]. In this case the size of the transmission object carrying the file would probably differ from the file size. The transmission object size is delivered to receivers as part of the FLUTE protocol.

- \* Security properties of the file such as digital signatures, message digests, etc. For example, one could use S/MIME [RFC5751] as the content encoding type for files with this authentication wrapper, and one could use XML-DSIG [RFC3275] to digitally sign the file. XML-DSIG can also be used to provide tamper prevention e.g., on the Content-Location field. Content encoding is applied to file data before FEC protection.

For each unique file, FLUTE encodes the attributes listed above and other attributes as children of an XML file element. A table of XML file elements is transmitted as a special file called a 'File Delivery Table' (FDT) which is further described in the next subsection and in section 3.2

### 3.1. File delivery session

ALC is a protocol instantiation of Layered Coding Transport building block (LCT) [RFC5651]. Thus ALC inherits the session concept of LCT. In this document we will use the concept ALC/LCT session to collectively denote the interchangeable terms ALC session and LCT session.

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

One of the fields carried in the ALC/LCT header is the Transport Session Identifier (TSI), an integer carried in a field of size 16, 32, or 48 bits (note that the TSI may be carried by other means in which case it is absent from the LCT header [RFC5651]). The (source IP address, TSI) pair uniquely identifies a session. Note that the TSI is scoped by the IP address, so the same TSI may be used by several source IP addresses at once. Thus, the receiver uses the (source IP address, TSI) pair from each packet to uniquely identify the session sending each packet. When a session carries multiple objects, the Transmission Object Identifier (TOI) field within the ALC/LCT header names the object used to generate each packet. Note that each object is associated with a unique TOI within the scope of a session.

A FLUTE session consistent with this specification MUST use FLUTE version 2 as specified in this document. Thus, all sessions consistent with this specification MUST set the FLUTE version to 2. The FLUTE version is carried within the EXT\_FDT extension header

(defined in section 3.4.1) in the ALC/LCT layer. A FLUTE session consistent with this specification MUST use ALC version 1 as specified in [RFC5775], and LCT version 1 as specified in [RFC5651].

If multiple FLUTE sessions are sent to a channel then receivers MUST determine the FLUTE protocol version, based on version fields and the (source IP address, TSI) carried in the ALC/LCT header of the packet. Note that when a receiver first begins receiving packets, it might not know the FLUTE protocol version, as not every LCT packet carries the EXT\_FDT header (containing the FLUTE protocol version.) A new receiver MAY keep an open binding in the LCT protocol layer between the TSI and the FLUTE protocol version, until the EXT\_FDT header arrives. Alternately, a new receiver MAY discover a binding between TSI and FLUTE protocol version via a session discovery protocol that is out of scope in this document.

If the sender's IP address is not accessible to receivers, then packets that can be received by receivers contain an intermediate IP address. In this case the TSI is scoped by this intermediate IP address of the sender for the duration of the session. As an example, the sender may be behind a Network Address Translation (NAT) device that temporarily assigns an IP address for the sender. In this case the TSI is scoped by the intermediate IP address assigned by the NAT. As another example, the sender may send its original packets using IPv6, but some portions of the network may not be IPv6 capable. Thus, there may be an IPv6 to IPv4 translator that changes the IP address of the packets to a different IPv4 address. In this case, receivers in the IPv4 portion of the network will receive packets containing the IPv4 address, and thus the TSI for them is scoped by the IPv4 address. How the IP address of the sender to be used to scope the session by receivers is delivered to receivers, whether it is the sender's IP address or an intermediate IP address, is outside the scope of this document.

When FLUTE is used for file delivery over ALC the following rules apply:

- \* The ALC/LCT session is called a file delivery session.
- \* The ALC/LCT concept of 'object' denotes either a 'file' or a 'File Delivery Table Instance' (section 3.2)
- \* The TOI field MUST be included in ALC packets sent within a FLUTE session, with the exception that ALC packets sent in a FLUTE session with the Close Session (A) flag set to 1 (signaling the end of the session) and that contain no payload (carrying no information for any file or FDT) SHALL NOT carry the TOI. See section 5.1 of [RFC5651] for the LCT definition of the Close

Session flag, and see section 4.2 of [RFC5775] for an example of the use of a TOI within an ALC packet.

- \* The TOI value '0' is reserved for delivery of File Delivery Table Instances. Each non expired File Delivery Table Instance is uniquely identified by an FDT Instance ID within the EXT\_FDT header defined in section 3.4.1.
- \* Each file in a file delivery session MUST be associated with a TOI (>0) in the scope of that session.
- \* Information carried in the headers and the payload of a packet is scoped by the source IP address and the TSI. Information particular to the object carried in the headers and the payload of a packet is further scoped by the TOI for file objects, and is further scoped by both the TOI and the FDT Instance ID for FDT Instance objects.

### 3.2. File Delivery Table

The File Delivery Table (FDT) provides a means to describe various attributes associated with files that are to be delivered within the file delivery session. The following lists are examples of such attributes, and are not intended to be mutually exclusive nor exhaustive.

Attributes related to the delivery of file:

- TOI value that represents the file
- FEC Object Transmission Information (including the FEC Encoding ID and, if relevant, the FEC Instance ID)
- Size of the transmission object carrying the file
- Aggregate rate of sending packets to all channels

Attributes related to the file itself:

- Name, Identification and Location of file (specified by the URI)
- MIME media type of file
- Size of file

- Encoding of file
- Message digest of file

Some of these attributes **MUST** be included in the file description entry for a file, others are optional, as defined in section 3.4.2.

Logically, the FDT is a set of file description entries for files to be delivered in the session. Each file description entry **MUST** include the TOI for the file that it describes and the URI identifying the file. The TOI carried in each file description entry is how FLUTE names the ALC/LCT data packets used for delivery of the file. Each file description entry may also contain one or more descriptors that map the above-mentioned attributes to the file.

Each file delivery session **MUST** have an FDT that is local to the given session. The FDT **MUST** provide a file description entry mapped to a TOI for each file appearing within the session. An object that is delivered within the ALC session, but not described in the FDT, other than the FDT itself, is not considered a 'file' belonging to the file delivery session. This object received with an unmapped TOI (Non-zero TOI that is not resolved by the FDT) **SHOULD** in general be ignored by a FLUTE receiver. The details of how to do that is out of scope of this specification.

Note that a client that joins an active file delivery session **MAY** receive data packets for a TOI > 0 before receiving any FDT Instance (see Section 3.3 for recommendations on how to limit the probability this occurs). Even if the TOI is not mapped to any file description entry, this is hopefully a transient situation. When this happens, system performance might be improved by caching such packets within a reasonable time window and storage size. Such optimizations are use-case and implementation specific and further details are beyond the scope of this document.

Within the file delivery session the FDT is delivered as FDT Instances. An FDT Instance contains one or more file description entries of the FDT. Any FDT Instance can be equal to, a subset of, a superset of, overlap with or complement any other FDT Instance. A certain FDT Instance may be repeated multiple times during a session, even after subsequent FDT Instances (with higher FDT Instance ID numbers) have been transmitted. Each FDT Instance contains at least a single file description entry and at most the exhaustive set of file description entries of the files being delivered in the file delivery session.

A receiver of the file delivery session keeps an FDT database for received file description entries. The receiver maintains the



database, for example, upon reception of FDT Instances. Thus, at any given time the contents of the FDT database represent the receiver's current view of the FDT of the file delivery session. Since each receiver behaves independently of other receivers, it SHOULD NOT be assumed that the contents of the FDT database are the same for all the receivers of a given file delivery session.

Since the FDT database is an abstract concept, the structure and the maintenance of the FDT database are left to individual implementations and are thus out of scope of this specification.

### 3.3. Dynamics of FDT Instances within file delivery session

The following rules define the dynamics of the FDT Instances within a file delivery session:

- \* For every file delivered within a file delivery session there MUST be a file description entry included in at least one FDT Instance sent within the session. A file description entry contains at a minimum the mapping between the TOI and the URI.
- \* An FDT Instance MAY appear in any part of the file delivery session and packets for an FDT Instance MAY be interleaved with packets for other files or other FDT Instances within a session.
- \* The TOI value of '0' MUST be reserved for delivery of FDT Instances. The use of other TOI values (i.e., an integer > 0) for FDT Instances is outside the scope of this specification.
- \* The FDT Instance is identified by the use of a new fixed length LCT Header Extension EXT\_FDT (defined later in this section.) Each non expired FDT Instance is uniquely identified within the file delivery session by its FDT Instance ID, carried by the EXT\_FDT Header Extension. Any ALC/LCT packet carrying an FDT Instance MUST include EXT\_FDT.
- \* It is RECOMMENDED that an FDT Instance that contains the file description entry for a file is sent at least once before sending the described file within a file delivery session. This recommendation is intended to minimize the amount of file data which may be received by receivers in advance of the FDT Instance containing the entry for a file (such data must either be speculatively buffered or discarded). Note that this possibility cannot be completely eliminated since the first transmission of FDT data might be lost.

- \* Within a file delivery session, any TOI > 0 MAY be described more than once. An example: previous FDT Instance 0 describes TOI of value '3'. Now, subsequent FDT Instances can either keep TOI '3' unmodified on the table, not include it, or augment the description. However, subsequent FDT Instances MUST NOT change the parameters already described for a specific TOI.
- \* An FDT Instance is valid until its expiration time. The expiration time is expressed within the FDT Instance payload as an UTF-8 decimal representation of a 32 bit unsigned integer. The value of this integer represents the 32 most significant bits of a 64 bit Network Time Protocol (NTP) [RFC5905] time value. These 32 bits provide an unsigned integer representing the time in seconds relative to 0 hours 1 January 1900 in case of the prime epoch (era 0) [RFC5905]. The handling of time wraparound (to happen in 2036) requires to consider the associated epoch. In any case, both a sender and a receiver can determine to which (136 year) epoch the FDT Instance expiration time value pertains to by choosing the epoch for which the expiration time is closest in time to the current time.

Here is an example. Let us imagine a new FLUTE session is started on February 7th, 2036, 0h, i.e., at NTP time 4,294,944,000, a few hours before the end of epoch 0. In order to define an FDT Instance valid for the next 48 hours, The FLUTE sender sets an expiry time of 149,504. This FDT Instance will expire exactly on February 9th, 2036, 0h. A client that receives this FDT Instance on the 7th, 0h, just after it has been sent, immediately understands this value corresponds to epoch 1. A client that joins the session on February 8th, 0h, i.e., at NTP time 63,104, epoch 1, immediately understands that the 149,504 NTP timestamp corresponds to epoch 1.

- \* The space of FDT Instance IDs is limited by the associated field size (i.e., 20 bits) in the EXT\_FDT header extension (Section 3.4.1). Therefore senders should take care to always have a large enough supply of available FDT Instance IDs when specifying FDT expiration times.
- \* The receiver MUST NOT use a received FDT Instance to interpret packets received beyond the expiration time of the FDT Instance.
- \* A sender MUST use an expiration time in the future upon creation of an FDT Instance relative to its Sender Current Time (SCT).

- \* Any FEC Encoding ID MAY be used for the sending of FDT Instances. The default is to use the Compact No-code FEC Encoding ID 0 [RFC5445] for the sending of FDT Instances. (Note that since FEC Encoding ID 0 is the default for FLUTE, this implies that Source Block Number and Encoding Symbol ID lengths both default to 16 bits each.)
- \* If the receiver does not support the FEC scheme indicated by the FEC Encoding ID, the receiver MUST NOT decode the associated FDT.
- \* It is RECOMMENDED that the mechanisms used for file attribute delivery SHOULD achieve a delivery probability that is higher than the file recovery probability and the file attributes SHOULD be delivered at this higher priority before the delivery of the associated files begins.

Generally, a receiver needs to receive an FDT Instance describing a file before it is able to recover the file itself. In this sense FDT Instances are of higher priority than files. Additionally, a FLUTE sender SHOULD assume receivers will not receive all packets pertaining to FDT Instances. The way FDT Instances are transmitted has a large impact on satisfying the recommendation above. When there is a single file transmitted in the session, one way to satisfy the recommendation above is to repeatedly transmit on a regular enough basis FDT Instances describing the file while the file is being transmitted. If an FDT Instance is longer than one packet payload in length, it is RECOMMENDED that an FEC code that provides protection against loss be used for delivering this FDT Instance. When there are multiple files in a session concurrently being transmitted to receivers, the way the FDT Instances are structured and transmitted also has a large impact. As an example, a way to satisfy the recommendation above is to transmit an FDT Instance that describes all files currently being transmitted, and to transmit this FDT Instance reliably, using the same techniques as explained for the case when there is a single file transmitted in a session. If instead the concurrently transmitted files are described in separate FDT Instances, another way to satisfy this recommendation is to transmit all the relevant FDT Instances reliably, using the same techniques as explained for the case when there is a single file transmitted in a session.

In any case, how often the description of a file is sent in an FDT Instance, how often an FDT Instance is sent, and how much FEC protection is provided for an FDT Instance (if longer than one packet payload) are dependent on the particular application and are outside the scope of this document.

Sometimes the various attributes associated with files that are to be

delivered within the file delivery session are sent out-of-band (rather than in-band, within one or several FDT Instances). The details of how this is done are out of the scope of this document. However, it is still RECOMMENDED that any out-of-band transmission be managed in such a way that a receiver will be able to recover the attributes associated with a file with as much or greater reliability as the receiver is able to receive enough packets containing encoding symbols to recover the file. For example, the probability of a randomly chosen receiver being able to recover a given file can often be estimated based on a statistical model of reception conditions, the amount of data transmitted and the properties of any Forward Error Correction in use. The recommendation above suggests that mechanisms used for file attribute delivery should achieve higher a delivery probability than the file recovery probability.

#### 3.4. Structure of FDT Instance packets

FDT Instances are carried in ALC packets with TOI = 0 and with an additional REQUIRED LCT Header extension called the FDT Instance Header. The FDT Instance Header (EXT\_FDT) contains the FDT Instance ID that uniquely identifies FDT Instances within a file delivery session. The FDT Instance Header is placed in the same way as any other LCT extension header. There MAY be other LCT extension headers in use.

The FDT Instance is encoded for transmission, like any other object, using an FEC Scheme (which MAY be the Compact No-Code FEC Scheme). The LCT extension headers are followed by the FEC Payload ID, and finally the Encoding Symbols for the FDT Instance which contains one or more file description entries. A FDT Instance MAY span several ALC packets - the number of ALC packets is a function of the file attributes associated with the FDT Instance. The FDT Instance Header is carried in each ALC packet carrying the FDT Instance. The FDT Instance Header is identical for all ALC/LCT packets for a particular FDT Instance.

The overall format of ALC/LCT packets carrying an FDT Instance is depicted in the Figure 1 below. All integer fields are carried in "big-endian" or "network order" format, that is, most significant byte (octet) first. As defined in [RFC5775], all ALC/LCT packets are sent using UDP.

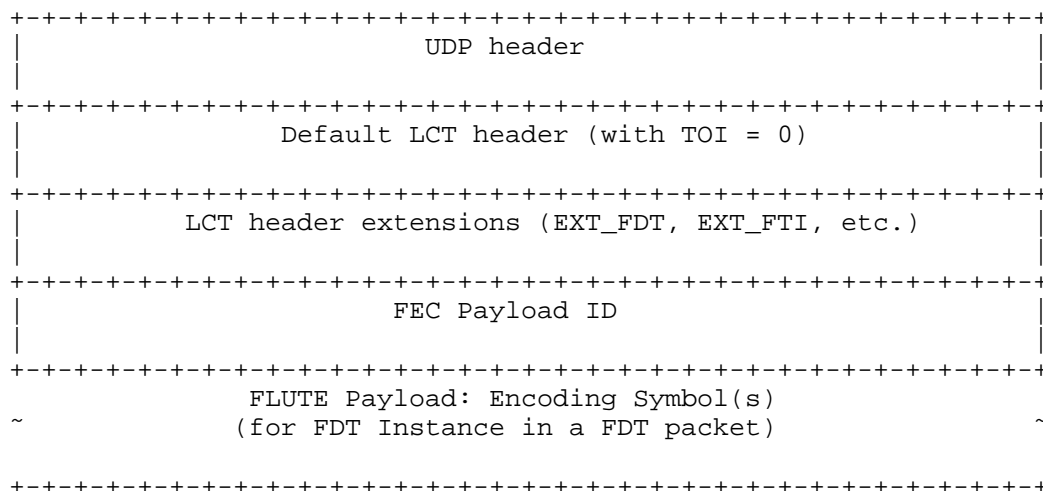


Figure 1: Overall FDT Packet

#### 3.4.1. Format of FDT Instance Header

The FDT Instance Header (EXT\_FDT) is a new fixed length, ALC PI specific LCT header extension [RFC5651]. The Header Extension Type (HET) for the extension is 192. Its format is defined below:

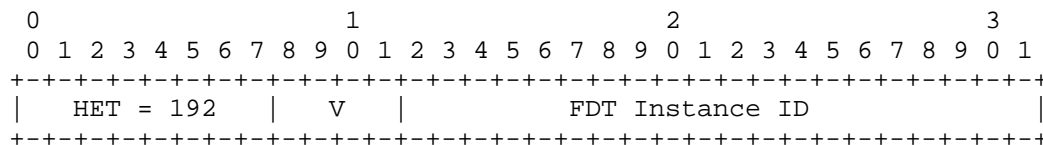


Figure 2

Version of FLUTE (V), 4 bits:

This document specifies FLUTE version 2. Hence in any ALC packet that carries FDT Instance and that belongs to the file delivery session as specified in this specification MUST set this field to '2'.

FDT Instance ID, 20 bits:

For each file delivery session the numbering of FDT Instances starts from '0' and is incremented by one for each subsequent FDT Instance. After reaching the maximum value ( $2^{20}-1$ ), the numbering starts from the smallest FDT Instance value assigned to an expired FDT Instance. When wraparound from a greater FDT Instance ID value to a smaller FDT

Instance ID value occurs, the smaller FDT Instance ID value is considered logically higher than the greater FDT Instance ID value. Senders **MUST NOT** re-use an FDT Instance ID value that is already in use for a non-expired FDT Instance. Sender behavior when all the FDT Instance IDs are used by non expired FEC Instances is outside the scope of this specification and left to individual implementations of FLUTE. Receipt of an FDT Instance that reuses an FDT Instance ID value that is currently used by a non expired FDT Instance **SHOULD** be considered as an error case. Receiver behavior in this case (e.g. leave the session or ignore the new FDT Instance) is outside the scope of this specification and left to individual implementations of FLUTE. Receivers **MUST** be ready to handle FDT Instance ID wraparound and situations where missing FDT Instance IDs result in increments larger than one.

#### 3.4.2. Syntax of FDT Instance

The FDT Instance contains file description entries that provide the mapping functionality described in 3.2 above.

The FDT Instance is an XML structure that has a single root element "FDT-Instance". The "FDT-Instance" element **MUST** contain "Expires" attribute, which tells the expiration time of the FDT Instance. In addition, the "FDT-Instance" element **MAY** contain the "Complete" attribute (boolean), which, when TRUE, signals that this "FDT Instance" includes the set of "File" entries that exhausts both the set of files delivered so far and also the set of files to be delivered in the session. This implies that no new data will be provided in future FDT Instances within this session (i.e., that either FDT Instances with higher ID numbers will not be used or if they are used, will only provide identical file parameters to those already given in this and previous FDT Instances). The "Complete" attribute is therefore used to provide a complete list of files in an entire FLUTE session (a "complete FDT").

The "FDT-Instance" element **MAY** contain attributes that give common parameters for all files of an FDT Instance. These attributes **MAY** also be provided for individual files in the "File" element. Where the same attribute appears in both the "FDT-Instance" and the "File" elements, the value of the attribute provided in the "File" element takes precedence.

For each file to be declared in the given FDT Instance there is a single file description entry in the FDT Instance. Each entry is represented by element "File" which is a child element of the FDT Instance structure.

The attributes of "File" element in the XML structure represent the

attributes given to the file that is delivered in the file delivery session. The value of the XML attribute name corresponds to MIME field name and the XML attribute value corresponds to the value of the MIME field body. Each "File" element MUST contain at least two attributes "TOI" and "Content-Location". "TOI" MUST be assigned a valid TOI value as described in section 3.3 above. "Content-Location" MUST be assigned a valid URI as defined in [RFC2616] which identifies the object to be delivered, for example a URI with the "http" or "file" URI scheme. The semantics for any two "File" elements declaring the same "Content-Location" but differing "TOI" is that the element appearing in the FDT Instance with the greater FDT Instance ID is considered to declare newer instance (e.g., version) of the same "File".

In addition to mandatory attributes, the "FDT-Instance" element and the "File" element MAY contain other attributes of which the following are specifically pointed out.

- \* The attribute "Content-Type" SHOULD be included and, when present, MUST be used for the purpose defined in [RFC2616].
- \* Where the length is described, the attribute "Content-Length" MUST be used for the purpose as defined in [RFC2616]. The transfer length is defined to be the length of the object transported in octets. It is often important to convey the transfer length to receivers, because the source block structure needs to be known for the FEC decoder to be applied to recover source blocks of the file, and the transfer length is often needed to properly determine the source block structure of the file. There generally will be a difference between the length of the original file and the transfer length if content encoding is applied to the file before transport, and thus the "Content-Encoding" attribute is used. If the file is not content encoded before transport (and thus the "Content-Encoding" attribute is not used) then the transfer length is the length of the original file, and in this case the "Content-Length" is also the transfer length. However, if the file is content encoded before transport (and thus the "Content-Encoding" attribute is used), e.g., if compression is applied before transport to reduce the number of octets that need to be transferred, then the transfer length is generally different than the length of the original file, and in this case the attribute "Transfer-Length" MAY be used to carry the transfer length.
- \* Whenever content encoding is applied the attribute "Content-Encoding" MUST be included. Whenever the attribute "Content-Encoding" is included it MUST be used as described in [RFC2616].

- \* Where the MD5 message digest is described, the attribute "Content-MD5" MUST be used for the purpose as defined in [RFC2616]. Note that the goal is to provide a decoded object integrity service in front of transmission and/or FLUTE/ALC processing errors (the collision probability is in that case negligible). It MUST NOT be regarded as a security mechanism (see Section 7 to that purpose).
- \* The FEC Object Transmission Information attributes as described in section 5.2.

The following specifies the XML Schema  
[XML-Schema-Part-1][XML-Schema-Part-2] for FDT Instance:

```
BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:ietf:params:xml:ns:fdt"
            xmlns:xs="http://www.w3.org/2001/XMLSchema"
            targetNamespace="urn:ietf:params:xml:ns:fdt"
            elementFormDefault="qualified">
  <xs:element name="FDT-Instance" type="FDT-InstanceType"/>
  <xs:complexType name="FDT-InstanceType">
    <xs:sequence>
      <xs:element name="File" type="FileType" maxOccurs="unbounded"/>
      <xs:any namespace="##other" processContents="skip"
              minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="Expires"
                  type="xs:string"
                  use="required"/>
    <xs:attribute name="Complete"
                  type="xs:boolean"
                  use="optional"/>
    <xs:attribute name="Content-Type"
                  type="xs:string"
                  use="optional"/>
    <xs:attribute name="Content-Encoding"
                  type="xs:string"
                  use="optional"/>
    <xs:attribute name="FEC-OTI-FEC-Encoding-ID"
                  type="xs:unsignedByte"
                  use="optional"/>
    <xs:attribute name="FEC-OTI-FEC-Instance-ID"
                  type="xs:unsignedLong"
                  use="optional"/>
    <xs:attribute name="FEC-OTI-Maximum-Source-Block-Length"
                  type="xs:unsignedLong"
                  use="optional"/>
    <xs:attribute name="FEC-OTI-Encoding-Symbol-Length"
```



```
        type="xs:unsignedLong"
        use="optional"/>
<xs:attribute name="FEC-OTI-Max-Number-of-Encoding-Symbols"
        type="xs:unsignedLong"
        use="optional"/>
<xs:attribute name="FEC-OTI-Scheme-Specific-Info"
        type="xs:base64Binary"
        use="optional"/>
<xs:anyAttribute processContents="skip"/>
</xs:complexType>
<xs:complexType name="FileType">
  <xs:sequence>
    <xs:any namespace="##other" processContents="skip"
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="Content-Location"
    type="xs:anyURI"
    use="required"/>
  <xs:attribute name="TOI"
    type="xs:positiveInteger"
    use="required"/>
  <xs:attribute name="Content-Length"
    type="xs:unsignedLong"
    use="optional"/>
  <xs:attribute name="Transfer-Length"
    type="xs:unsignedLong"
    use="optional"/>
  <xs:attribute name="Content-Type"
    type="xs:string"
    use="optional"/>
  <xs:attribute name="Content-Encoding"
    type="xs:string"
    use="optional"/>
  <xs:attribute name="Content-MD5"
    type="xs:base64Binary"
    use="optional"/>
  <xs:attribute name="FEC-OTI-FEC-Encoding-ID"
    type="xs:unsignedByte"
    use="optional"/>
  <xs:attribute name="FEC-OTI-FEC-Instance-ID"
    type="xs:unsignedLong"
    use="optional"/>
  <xs:attribute name="FEC-OTI-Maximum-Source-Block-Length"
    type="xs:unsignedLong"
    use="optional"/>
  <xs:attribute name="FEC-OTI-Encoding-Symbol-Length"
    type="xs:unsignedLong"
    use="optional"/>
```

```
<xs:attribute name="FEC-OTI-Max-Number-of-Encoding-Symbols"
              type="xs:unsignedLong"
              use="optional"/>
<xs:attribute name="FEC-OTI-Scheme-Specific-Info"
              type="xs:base64Binary"
              use="optional"/>
<xs:anyAttribute processContents="skip"/>
</xs:complexType>
</xs:schema>
END
```

Figure 3

Any valid FDT Instance MUST use the above XML Schema. This way FDT provides extensibility to support private attributes within the file description entries. Those could be, for example, the attributes related to the delivery of the file (timing, packet transmission rate, etc.). Unsupported private attributes SHOULD be silently ignored by a FLUTE receiver.

In case the basic FDT XML Schema is extended in terms of new descriptors (attributes or elements), for descriptors applying to a single file, those MUST be placed within the element "File". For descriptors applying to all files described by the current FDT Instance, those MUST be placed within the element "FDT-Instance". It is RECOMMENDED that the new attributes applied in the FDT are in the format of MIME fields and are either defined in the HTTP/1.1 specification [RFC2616], or another well-known specification, or in IANA registry [IANAMediatypes].

#### 3.4.3. Content Encoding of FDT Instance

The FDT Instance itself MAY be content encoded, for example compressed. This specification defines FDT Instance Content Encoding Header (EXT\_CENC). EXT\_CENC is a new fixed length LCT header extension [RFC5651]. The Header Extension Type (HET) for the extension is 193. If the FDT Instance is content encoded, the EXT\_CENC MUST be used to signal the content encoding type. In that case, EXT\_CENC header extension MUST be used in all ALC packets carrying the same FDT Instance ID. Consequently, when EXT\_CENC header is used, it MUST be used together with a proper FDT Instance Header (EXT\_FDT). Within a file delivery session, FDT Instances that are not content encoded and FDT Instances that are content encoded MAY both appear. If content encoding is not used for a given FDT Instance, the EXT\_CENC MUST NOT be used in any packet carrying the FDT Instance. The format of EXT\_CENC is defined below:

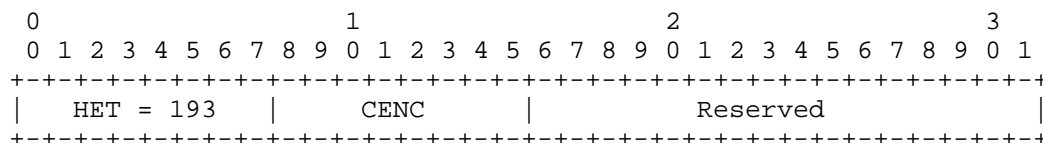


Figure 4

Content Encoding Algorithm (CENC), 8 bits:

This field signals the content encoding algorithm used in the FDT Instance payload. This subsection reserves the Content Encoding Algorithm values 0, 1, 2 and 3 for null, ZLIB [RFC1950], DEFLATE [RFC1951] and GZIP [RFC1952] respectively.

Reserved, 16 bits:

This field MUST be set to all '0'. This field MUST be ignored on reception.

### 3.5. Multiplexing of files within a file delivery session

The delivered files are carried as transmission objects (identified with TOIs) in the file delivery session. All these objects, including the FDT Instances, MAY be multiplexed in any order and in parallel with each other within a session, i.e., packets for one file may be interleaved with packets for other files or other FDT Instances within a session.

Multiple FDT Instances MAY be delivered in a single session using TOI = 0. In this case, it is RECOMMENDED that the sending of a previous FDT Instance SHOULD end before the sending of the next FDT Instance starts. However, due to unexpected network conditions, packets for the FDT Instances MAY be interleaved. A receiver can determine which FDT Instance a packet contains information about since the FDT Instances are uniquely identified by their FDT Instance ID carried in the EXT FDT headers.

#### 4. Channels, congestion control and timing

ALC/LCT has a concept of channels and congestion control. There are four scenarios in which FLUTE is envisioned to be applied.

- (a) Use of a single channel and a single-rate congestion control protocol.

- (b) Use of multiple channels and a multiple-rate congestion control protocol. In this case the FDT Instances MAY be delivered on more than one channel.
- (c) Use of a single channel without congestion control supplied by ALC, but only when in a controlled network environment where flow/congestion control is being provided by other means.
- (d) Use of multiple channels without congestion control supplied by ALC, but only when in a controlled network environment where flow/congestion control is being provided by other means. In this case the FDT Instances MAY be delivered on more than one channel.

When using just one channel for a file delivery session, as in (a) and (c), the notion of 'prior' and 'after' are intuitively defined for the delivery of objects with respect to their delivery times.

However, if multiple channels are used, as in (b) and (d), it is not straightforward to state that an object was delivered 'prior' to the other. An object may begin to be delivered on one or more of those channels before the delivery of a second object begins. However, the use of multiple channels/layers may complete the delivery of the second object before the first. This is not a problem when objects are delivered sequentially using a single channel. Thus, if the application of FLUTE has a mandatory or critical requirement that the first transmission object must complete 'prior' to the second one, it is RECOMMENDED that only a single channel is used for the file delivery session.

Furthermore, if multiple channels are used then a receiver joined to the session at a low reception rate will only be joined to the lower layers of the session. Thus, since the reception of FDT Instances is of higher priority than the reception of files (because the reception of files depends on the reception of an FDT Instance describing it), the following is RECOMMENDED:

1. The layers to which packets for FDT Instances are sent SHOULD NOT be biased towards those layers to which lower rate receivers are not joined. For example, it is okay to put all the packets for an FDT Instance into the lowest layer (if this layer carries enough packets to deliver the FDT to higher rate receivers in a reasonable amount of time), but it is not okay to put all the packets for an FDT Instance into the higher layers that only high rate receivers will receive.

2. If FDT Instances are generally longer than one Encoding Symbol in length and some packets for FDT Instances are sent to layers that lower rate receivers do not receive, an FEC Encoding other than Compact No-code FEC Encoding ID 0 [RFC5445] SHOULD be used to deliver FDT Instances. This is because in this case, even when there is no packet loss in the network, a lower rate receiver will not receive all packets sent for an FDT Instance.

## 5. Delivering FEC Object Transmission Information

FLUTE inherits the use of FEC building block [RFC5052] from ALC. When using FLUTE for file delivery over ALC the FEC Object Transmission Information MUST be delivered in-band within the file delivery session. There are two methods to achieve this: the use of ALC specific LCT extension header EXT\_FTI [RFC5775] and the use of FDT. The latter method is specified in this section. The use of EXT\_FTI requires repetition of the FEC Object Transmission Information to ensure reception (though not necessarily in every packet) and thus may entail higher overhead than the use of the FDT, but may also provide more timely delivery of the FEC Object Transmission Information.

The receiver of file delivery session MUST support delivery of FEC Object Transmission Information using the EXT\_FTI for the FDT Instances carried using TOI value 0. For the TOI values other than 0 the receiver MUST support both methods: the use of EXT\_FTI and the use of FDT.

The FEC Object Transmission Information that needs to be delivered to receivers MUST be exactly the same whether it is delivered using EXT\_FTI or using FDT (or both). The FEC Object Transmission Information that MUST be delivered to receivers is defined by the FEC Scheme. This section describes the delivery using FDT.

The FEC Object Transmission Information regarding a given TOI may be available from several sources. In this case, it is RECOMMENDED that the receiver of the file delivery session prioritize the sources in the following way (in the order of decreasing priority).

1. FEC Object Transmission Information that is available in EXT\_FTI.
2. FEC Object Transmission Information that is available in the FDT.

The FDT delivers FEC Object Transmission Information for each file using an appropriate attribute within the "FDT-Instance" or the "File" element of the FDT structure.

- \* "Transfer-Length" carries the Transfer-Length Object Transmission Information element defined in [RFC5052].
- \* "FEC-OTI-FEC-Encoding-ID" carries the "FEC Encoding ID" Object Transmission Information element defined in [RFC5052], as carried in the Codepoint field of the ALC/LCT header.
- \* "FEC-OTI-FEC-Instance-ID" carries the "FEC Instance ID" Object Transmission Information element defined in [RFC5052] for Under-specified FEC Schemes.
- \* "FEC-OTI-Maximum-Source-Block-Length" carries the "Maximum Source Block Length" Object Transmission Information element defined in [RFC5052], if required by the FEC Scheme.
- \* "FEC-OTI-Encoding-Symbol-Length" carries the "Encoding Symbol Length" Object Transmission Information element defined in [RFC5052], if required by the FEC Scheme.
- \* "FEC-OTI-Max-Number-of-Encoding-Symbols" carries the "Maximum Number of Encoding Symbols" Object Transmission Information element defined in [RFC5052], if required by the FEC Scheme.
- \* "FEC-OTI-Scheme-specific-information" carries the "encoded scheme-specific FEC Object Transmission Information" as defined in [RFC5052], if required by the FEC Scheme.

In FLUTE, the FEC Encoding ID (8 bits) for a given TOI MUST be carried in the Codepoint field of the ALC/LCT header. When the FEC Object Transmission Information for this TOI is delivered through the FDT, then the associated "FEC-OTI-FEC-Encoding-ID" attribute and the Codepoint field of all packets for this TOI MUST be the same.

## 6. Describing file delivery sessions

To start receiving a file delivery session, the receiver needs to know transport parameters associated with the session. Interpreting these parameters and starting the reception therefore represents the entry point from which thereafter the receiver operation falls into the scope of this specification. According to [RFC5775], the transport parameters of an ALC/LCT session that the receiver needs to know are:

- \* The source IP address;

- \* The number of channels in the session;
- \* The destination IP address and port number for each channel in the session;
- \* The Transport Session Identifier (TSI) of the session;
- \* An indication that the session is a FLUTE session. The need to demultiplex objects upon reception is implicit in any use of FLUTE, and this fulfills the ALC requirement of an indication of whether or not a session carries packets for more than one object (all FLUTE sessions carry packets for more than one object).

Optionally, the following parameters MAY be associated with the session (Note, the list is not exhaustive):

- \* The start time and end time of the session;
- \* FEC Encoding ID and FEC Instance ID when the default FEC Encoding ID 0 is not used for the delivery of FDT;
- \* Content Encoding format if optional content encoding of FDT Instance is used, e.g., compression;
- \* Some information that tells receiver, in the first place, that the session contains files that are of interest;
- \* Definition and configuration of congestion control mechanism for the session;
- \* Security parameters relevant for the session;
- \* FLUTE version number.

It is envisioned that these parameters would be described according to some session description syntax (such as SDP [RFC4566] or XML based) and held in a file which would be acquired by the receiver before the FLUTE session begins by means of some transport protocol (such as Session Announcement Protocol (SAP) [RFC2974], email, HTTP [RFC2616], SIP [RFC3261], manual pre-configuration, etc.) However, the way in which the receiver discovers the above-mentioned parameters is out of scope of this document, as it is for LCT and ALC. In particular, this specification does not mandate or exclude any mechanism.

## 7. Security Considerations

### 7.1. Problem Statement

A content delivery system is potentially subject to attacks. Attacks may target:

- \* the network (to compromise the routing infrastructure, e.g., by creating congestion),
- \* the Content Delivery Protocol (CDP) (e.g., to compromise the normal behavior of FLUTE), or
- \* the content itself (e.g., to corrupt the files being transmitted).

These attacks can be launched either:

- \* against the data flow itself (e.g., by sending forged packets),
- \* against the session control parameters (e.g., by corrupting the session description, the FDT Instances, or the ALC/LCT control parameters) that are sent either in-band or out-of-band, or
- \* against some associated building blocks (e.g., the congestion control component).

In the following sections we provide more details on these possible attacks and sketch some possible counter-measures. We provide recommendations in Section 7.5.

### 7.2. Attacks against the data flow

Let us consider attacks against the data flow first. At least, the following types of attacks exist:

- \* attacks that are meant to give access to a confidential file (e.g., in case of a non-free content) and
- \* attacks that try to corrupt the file being transmitted (e.g., to inject malicious code within a file, or to prevent a receiver from using a file, which is a kind of Denial of Service, DoS).

#### 7.2.1. Access to confidential files

Access control to the file being transmitted is typically provided by means of encryption. This encryption can be done over the whole file i.e., before applying FEC protection (e.g., by the content provider, before submitting the file to FLUTE), or be done on a packet per



packet basis (e.g., when IPsec/ESP is used [RFC4303], see Section 7.5). If confidentiality is a concern, it is RECOMMENDED that one of these solutions be used.

#### 7.2.2. File corruption

Protection against corruptions (e.g., if an attacker sends forged packets) is achieved by means of a content integrity verification/sender authentication scheme. This service can be provided at the file level i.e., before applying content encoding and forward error correction encoding. In that case a receiver has no way to identify which symbol(s) is(are) corrupted if the file is detected as corrupted. This service can also be provided at the packet level i.e., after applying content encoding and forward error correction encoding, on a packet by packet basis. In this case, after removing all corrupted packets, the file may be in some cases recovered from the remaining correct packets.

Integrity protection applied at the file level has the advantage of lower overhead since only relatively few bits are added to provide the integrity protection compared to the file size. However it has the disadvantage that it cannot distinguish between correct packets and corrupt packets and therefore correct packets, which may form the majority of packets received, may be unusable. Integrity protection applied at the packet level has the advantage that it can distinguish between correct and corrupt packets at the cost of additional per packet overhead.

Several techniques can provide this source authentication/content integrity service:

- \* at the file level, the file MAY be digitally signed, for instance by using RSASSA-PKCS1-v1\_5 [RFC3447]. This signature enables a receiver to check the file integrity, once this latter has been fully decoded. Even if digital signatures are computationally expensive, this calculation occurs only once per file, which is usually acceptable;
- \* at the packet level, each packet can be digitally signed [RMT-SIMPLE-AUTH]. A major limitation is the high computational and transmission overheads that this solution requires. To avoid this problem, the signature may span a set of symbols (instead of a single one) in order to amortize the signature calculation, but if a single symbol is missing, the integrity of the whole set cannot be checked;

- \* at the packet level, a Group Message Authentication Code (MAC) [RFC2104][RMT-SIMPLE-AUTH] scheme can be used, for instance by using HMAC-SHA-256 with a secret key shared by all the group members, senders and receivers. This technique creates a cryptographically secured digest of a packet that is sent along with the packet. The Group MAC scheme does not create prohibitive processing load nor transmission overhead, but it has a major limitation: it only provides a group authentication/integrity service since all group members share the same secret group key, which means that each member can send a forged packet. It is therefore restricted to situations where group members are fully trusted (or in association with another technique as a pre-check);
- \* at the packet level, TESLA [RFC4082][RFC5776] is an attractive solution that is robust to losses, provides a true authentication/integrity service, and does not create any prohibitive processing load or transmission overhead. Yet checking a packet requires a small delay (a second or more) after its reception;
- \* at the packet level, IPsec/ESP [RFC4303] can be used to check the integrity and authenticate the sender of all the packets being exchanged in a session (see Section 7.5).

Techniques relying on public key cryptography (digital signatures and TESLA during the bootstrap process, when used) require that public keys be securely associated to the entities. This can be achieved by a Public Key Infrastructure (PKI), or by a PGP Web of Trust, or by pre-distributing the public keys of each group member.

Techniques relying on symmetric key cryptography (Group MAC) require that a secret key be shared by all group members. This can be achieved by means of a group key management protocol, or simply by pre-distributing the secret key (but this manual solution has many limitations).

It is up to the developer and deployer, who know the security requirements and features of the target application area, to define which solution is the most appropriate. Nonetheless, in case there is any concern of the threat of file corruption, it is RECOMMENDED that at least one of these techniques be used.

### 7.3. Attacks against the session control parameters and associated Building Blocks

Let us now consider attacks against the session control parameters and the associated building blocks. The attacker has at least the following opportunities to launch an attack:

- \* the attack can target the session description,
- \* the attack can target the FDT Instances,
- \* the attack can target the ALC/LCT parameters, carried within the LCT header or
- \* the attack can target the FLUTE associated building blocks, for instance the multiple rate congestion control protocol.

The consequences of these attacks are potentially serious, since they might compromise the behavior of content delivery system itself.

#### 7.3.1. Attacks against the Session Description

A FLUTE receiver may potentially obtain an incorrect Session Description for the session. The consequence of this is 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 possibly disrupting other traffic in the network.

To avoid these problems, it is RECOMMENDED that measures be taken to prevent receivers from accepting incorrect Session Descriptions. One such measure is source authentication to ensure that receivers only accept legitimate Session Descriptions from authorized senders. How these measures are achieved is outside the scope of this document since this session description is usually carried out-of-band.

#### 7.3.2. Attacks against the FDT Instances

Corrupting the FDT Instances is one way to create a Denial of Service attack. For example, the attacker changes the MD5 sum associated to a file. This possibly leads a receiver to reject the files received, no matter whether the files have been correctly received or not.

Corrupting the FDT Instances is also a way to make the reception process more costly than it should be. This can be achieved by changing the FEC Object Transmission Information when the FEC Object Transmission Information is included in the FDT Instance. For example, an attacker may corrupt the FDT Instance in such a way that Reed-Solomon over  $GF(2^{16})$  be used instead of  $GF(2^8)$  with FEC Encoding ID 2. This may significantly increase the processing load while compromising FEC decoding.

It is therefore RECOMMENDED that measures be taken to guarantee the integrity and to check the sender's identity of the FDT Instances. To that purpose, one of the counter-measures mentioned above

(Section 7.2.2) SHOULD be used. These measures will either be applied on a packet level, or globally over the whole FDT Instance object. Additionally, XML digital signatures [RFC3275] are a way to protect the FDT Instance by digitally signing it. When there is no packet level integrity verification scheme, it is RECOMMENDED to rely on XML digital signatures of the FDT Instances.

#### 7.3.3. Attacks against the ALC/LCT parameters

By corrupting the ALC/LCT header (or header extensions) one can execute attacks on underlying ALC/LCT implementation. For example, sending forged ALC packets with the Close Session flag (A) set to one can lead the receiver to prematurely close the session. Similarly, sending forged ALC packets with the Close Object flag (B) set to one can lead the receiver to prematurely give up the reception of an object.

It is therefore RECOMMENDED that measures be taken to guarantee the integrity and to check the sender's identity of the ALC packets received. To that purpose, one of the counter-measures mentioned above (Section 7.2.2) SHOULD be used.

#### 7.3.4. Attacks against the associated Building Blocks

Let us first focus on the congestion control building block, that may be used in the ALC session. A receiver with an incorrect or corrupted implementation of the multiple rate congestion control building block may affect the health of the network in the path between the sender and the receiver. That may also affect the reception rates of other receivers who joined the session.

When congestion control building block is applied with FLUTE, it is therefore RECOMMENDED that receivers be required to identify themselves as legitimate before they receive the Session Description needed to join the session. How receivers identify themselves as legitimate is outside the scope of this document. If authenticating a receiver does not prevent this latter to launch an attack, it will enable the network operator to identify him and to take counter-measures.

When congestion control building block is applied with FLUTE, it is also RECOMMENDED that a packet level authentication scheme be used, as explained in Section 7.2.2. Some of them, like TESLA, only provide a delayed authentication service, whereas congestion control requires a rapid reaction. It is therefore RECOMMENDED [RFC5775] that a receiver using TESLA quickly reduces its subscription level when the receiver believes that a congestion did occur, even if the packet has not yet been authenticated. Therefore TESLA will not

prevent DoS attacks where an attacker makes the receiver believe that a congestion occurred. This is an issue for the receiver, but this will not compromise the network. Other authentication methods that do not feature this delayed authentication could be preferred, or a group MAC scheme could be used in parallel to TESLA to prevent attacks launched from outside of the group.

#### 7.4. Other Security Considerations

Lastly, we note that the security considerations that apply to, and are described in, ALC [RFC5775], LCT [RFC5651] and FEC [RFC5052] also apply to FLUTE as FLUTE builds on those specifications. In addition, any security considerations that apply to any congestion control building block used in conjunction with FLUTE also apply to FLUTE.

#### 7.5. Minimum Security Recommendations

We now introduce a mandatory to implement but not necessarily to use security configuration, in the sense of [RFC3365]. Since FLUTE relies on ALC/LCT, it inherits the "baseline secure ALC operation" of [RFC5775]. More precisely, security is achieved by means of IPsec/ESP in transport mode. [RFC4303] explains that ESP can be used to potentially provide confidentiality, data origin authentication, content integrity, anti-replay and (limited) traffic flow confidentiality. [RFC5775] specifies that the data origin authentication, content integrity and anti-replay services SHALL be supported, and that the confidentiality service is RECOMMENDED. If a short lived session MAY rely on manual keying, it is also RECOMMENDED that an automated key management scheme be used, especially in case of long lived sessions.

Therefore, the RECOMMENDED solution for FLUTE provides per-packet security, with data origin authentication, integrity verification and anti-replay. This is sufficient to prevent most of the in-band attacks listed above. If confidentiality is required, a per-packet encryption SHOULD also be used.

### 8. IANA Considerations

This specification contains five separate items for IANA Considerations:

1. Registration Request for XML Schema of FDT Instance.

2. Media-Type Registration Request for application/fdt+xml.
3. Content Encoding Algorithm Registration Request.
4. Registration of the EXT\_FDT LCT Header Extension Type
5. Registration of the EXT\_CENC LCT Header Extension Type

#### 8.1. Registration Request for XML Schema of FDT Instance

Document [RFC3688] defines an IANA maintained registry of XML documents used within IETF protocols. The following is the registration request for the FDT XML schema.

Registrant Contact: Toni Paila (toni.paila (at) nokia.com)

XML: The XML Schema specified in Section 3.4.2

#### 8.2. Media-Type Registration Request for application/fdt+xml

This section provides the registration request, as per [RFC4288], [RFC4289] and [RFC3023], to be submitted to IANA following IESG approval.

Type name: application

Subtype name: fdt+xml

Required parameters: none

Optional parameters: none

Encoding considerations: The fdt+xml type consists of UTF-8 ASCII characters [RFC3629] and must be well-formed XML.

Additional content and transfer encodings may be used with fdt+xml files, with the appropriate encoding for any specific file being entirely dependent upon the deployed application.

Restrictions on usage: Only for usage with FDT Instances which are valid according to the XML schema of section 3.4.2.

Security considerations: fdt+xml data is passive, and does not generally represent a unique or new security threat. However, there is some risk in sharing any kind of data, in that unintentional information may be exposed, and that risk applies to fdt+xml data as well.

Interoperability considerations: None

Published specification: The present document including section 3.4.2. The specified FDT Instance functions as an actual media format of use to the general Internet community and thus media type registration under the Standards Tree is appropriate to maximize interoperability.

Applications which use this media type: Not restricted to any particular application

Additional information:

    Magic number(s): none

    File extension(s): An FDT Instance may use the extension ".fdt" but this is not required.

    Macintosh File Type Code(s): none

Person and email address to contact for further information: Toni Paila (toni.paila (at) nokia.com)

Intended usage: Common

Author/Change controller: IETF

### 8.3. Content Encoding Algorithm Registration Request

Values of Content Encoding Algorithms are subject to IANA registration. The value of Content Encoding Algorithm is a numeric non-negative index. In this document, the range of values for Content Encoding Algorithms is 0 to 255. This specification already assigns the values 0, 1, 2 and 3 as described in section 3.4.3.

#### 8.3.1. Explicit IANA Assignment Guidelines

This document defines a name-space called "Content Encoding Algorithms".

IANA has established and manages the new registry for the "FLUTE Content Encoding Algorithm" name-space. The values that can be assigned within this name-space are numeric indexes in the range [0, 255], boundaries included. Assignment requests are granted on a "Specification Required" basis as defined in [RFC5226]. Note that the values 0, 1, 2 and 3 of this registry are already assigned by this document as described in section 3.4.3.

#### 8.4. Registration of EXT\_FDT LCT Header Extension Type

This document registers value 192 for the EXT\_FDT LCT Header Extension defined in Section 3.4.1.

#### 8.5. Registration of EXT\_CENC LCT Header Extension Type

This document registers value 193 for the EXT\_CENC LCT Header Extension defined in Section 3.4.3.

### 9. Acknowledgments

The following persons have contributed to this specification: Brian Adamson, Mark Handley, Esa Jalonen, Roger Kermode, Juha-Pekka Luoma, Topi Pohjolainen, Lorenzo Vicisano, Mark Watson, David Harrington, Ben Campbell, Stephen Farrell, Robert Sparks, Ronald Bonica and Francis Dupont. The authors would like to thank all the contributors for their valuable work in reviewing and providing feedback regarding this specification.

### 10. Contributors

Jani Peltotalo  
Tampere University of Technology  
P.O. Box 553 (Korkeakoulunkatu 1)  
Tampere FIN-33101  
Finland  
Email: jani.peltotalo (at) tut.fi

Sami Peltotalo  
Tampere University of Technology  
P.O. Box 553 (Korkeakoulunkatu 1)  
Tampere FIN-33101  
Finland  
Email: sami.peltotalo (at) tut.fi

Magnus Westerlund  
Ericsson Research  
Ericsson AB  
SE-164 80 Stockholm  
Sweden  
Email: magnus.westerlund (at) ericsson.com

Thorsten Lohmar  
Ericsson Research (EDD)  
Ericsson Allee 1



52134 Herzogenrath  
Germany  
EMail: thorsten.lohmar (at) ericsson.com

## 11. Change Log

### 11.1. RFC3926 to draft-ietf-rmt-flute-revised-12

Incremented FLUTE protocol version from 1 to 2, due to concerns about backwards compatibility. For instance, the LCT header changed between RFC 3451 and [RFC5651]. In RFC 3451, the T and R fields of the LCT header respectively indicate the presence of Sender Current Time and Expected Residual Time. In [RFC5651], these fields MUST be set to zero and MUST be ignored by receivers (instead the EXT\_TIME extension headers can convey this information if needed). Thus, [RFC5651] is not backwards compatible with RFC 3451, even though both have the same LCT version 1. FLUTE version 1 as specified in [RFC3926] MUST use RFC 3451. FLUTE version 2 as specified in this document MUST use [RFC5651]. Therefore an implementation that relies on [RFC3926] and RFC 3451 will not be backwards compatible with FLUTE as specified in this document.

Updated dependencies to other RFCs to revised versions, e.g., changed ALC reference from RFC 3450 to [RFC5775], changed LCT reference from RFC 3451 to [RFC5651], etc.

Two additional items are added in the IANA considerations section, specifically the registration of two values in the LCT Header Extension Types registry (192 for EXT\_FDT and 193 for EXT\_CENC).

Added clarification for the use of FLUTE for unicast communications in Section 1.1.4.

Clarified how to reliably deliver the FDT in Section 3.3 and the possibility of using an out-of-band delivery of FDT information.

Clarified how to address FDT Instance expiration time wraparound with the notion of "epoch" of NTPv4 in Section 3.3.

Clarified what should be considered as erroneous situations in Section 3.4.1 (definition of FDT Instance ID). In particular a receiver MUST be ready to handle FDT Instance ID wraparounds and missing FDT Instances.

Updated the security section to define IPsec/ESP as a mandatory to implement security solution in Section 7.5.

Removed the 'Statement of Intent' from the Section 1. The statement of intent was meant to clarify the "Experimental" status of [RFC3926]. It does not apply to this draft that is intended for "Standard Track" submission.

Added clarification on XML-DSIG in the end of Section 3.

Revised the use of word "complete" in the Section 3.2.

Clarified Figure 1 WRT "Encoding Symbol(s) for FDT Instance".

Clarified the FDT Instance ID wrap-around in the end of Section 3.4.1.

Clarification for "Complete FDT" in the Section 3.4.2.

Added semantics for the case two TOIs refer to same Content-Location. Now it is in line how 3GPP and DVB interpret the case.

In the Section 3.4.2 XML Schema of FDT instance is modified to various advices. For example, extension by element was missing but is now supported. Also namespace definition is changed to URN format.

Clarified FDT-schema extensibility in the end of Section 3.4.2.

The CENC value allocation is added in the end of Section 3.4.3.

Section 5 is modified so that EXT\_FTI and the FEC issues are replaced by a reference to LCT specification. We count on revised LCT specification to specify the EXT\_FTI.

Added a clarifying paragraph on the use of Codepoint in the very end of Section 5.

Reworked Section 8 - IANA Considerations. Now it contains three IANA registration requests:

- \* Registration Request for XML Schema of FDT Instance  
(urn:ietf:params:xml:schema:fdt)
- \* Media-Type Registration Request for application/fdt+xml
- \* Content Encoding Algorithm Registration Request (ietf:rmt:cenc)

Added Section 10 - Contributors.

Revised list of both Normative as well as Informative references.

Added a clarification that receiver should ignore reserved bits of Header Extension type 193 upon reception.

Minor changes to remove forward references (use before definition) or refer to forward reference sections.

Elaborate on just what kind of networks cannot support FLUTE congestion control (1.1.4)

In Section 3.2 revise "several" (meaning 3-n vs. "couple" = 2) to "multiple" (meaning 2-n)

Move Section 3.3 requirement to send FDT more reliably than files, to a bulleted RECOMMENDED requirement, making check-off easier for testers.

Sharpen Section 3.3 definition that future FDT file instances can "augment" (meaning enhance) rather than "complement" (sometimes meaning negate, which is not allowed) the file parameters.

Elaborate in Section 3.3 and Section 4 that FEC Encoding ID = 0 is Compact No-code FEC, so that the reader doesn't have to search other RFCs to understand these protocol constants used by FLUTE.

Require in Section 3.3 that FLUTE receivers SHALL NOT attempt to decode FDTs if they do not understand the FEC Encoding ID

Remove restriction of Section 3.3 in bullet #4 that TOI=0 for the FDT, to be consistent with Appendix, bullet 6, and elsewhere. An FDT is signaled by an FDT Instance ID, NOT only by TOI = 0.

Standardize on the term "expiration time" and avoid using the redundant but possibly confusing term "expiry time".

To interwork with experimental flute, stipulate in Section 3.1 that only 1 instantiation of all 3 protocols FLUTE, ALC, and LCT, can be associated with a session (source IP-Address, TSI) and mention in Section 6 that you may (optionally) derive the FLUTE version from the file delivery session description.

Use a software writing tool to lower reading grade level and simplify Section 3.1.

## 12. References

## 12.1. Normative references

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, BCP 14, March 1997.
- [RFC5775] Luby, M., Watson, M., and L. Vicisano, "Asynchronous Layered Coding (ALC) Protocol Instantiation", RFC 5775, April 2010.
- [RFC5651] Luby, M., Watson, M., and L. Vicisano, "Layered Coding Transport (LCT) Building Block", RFC 5651, October 2009.
- [RFC5052] Watson, M., Luby, M., and L. Vicisano, "Forward Error Correction (FEC) Building Block", RFC 5052, August 2007.
- [RFC5445] Watson, M., "Basic Forward Error Correction (FEC) Schemes", RFC 5445, March 2009.
- [RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.
- [XML-Schema-Part-1]  
Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition", W3C Recommendation, <http://www.w3.org/TR/xmlschema-1/>, October 2004.
- [XML-Schema-Part-2]  
Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", W3C Recommendation, <http://www.w3.org/TR/xmlschema-2/>, October 2004.
- [RFC3023] Murata, M., St.Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, January 2001.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 3629, November 2003.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, May 2008.
- [RFC3738] Luby, M. and V. Goyal, "Wave and Equation Based Rate Control (WEBRC) Building Block", RFC 3738, Note: this

reference is to a target document of a lower maturity level and some caution should be used since it may be less stable than the present document. , April 2004.

- [RFC4303] Kent, S., "Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

## 12.2. Informative references

- [RFC3926] Paila, T., Luby, M., Lehtonen, R., Roca, V., and R. Walsh, "FLUTE - File Delivery over Unidirectional Transport", RFC 3926, October 2004.
- [RFC2357] Mankin, A., Romanov, A., Bradner, S., and V. Paxson, "IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols", RFC 2357, June 1998.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC1950] Deutsch, P. and J-L. Gailly, "ZLIB Compressed Data Format Specification version 3.3", RFC 1950, May 1996.
- [RFC1951] Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, May 1996.
- [RFC1952] Deutsch, P., "GZIP file format specification version 4.3", RFC 1952, May 1996.
- [IANAmmediatypes]  
IANA, "IANA Media Types registry",  
URL: <http://www.iana.org/assignments/media-types/>.
- [RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "Session Description Protocol", RFC 4566, July 2006.
- [RFC1112] Deering, S., "Host Extensions for IP Multicasting", RFC 1112, STD 5, August 1989.
- [PAPER.SSM]  
Holbrook, H., "A Channel Model for Multicast, Ph.D. Dissertation, Stanford University, Department of Computer Science, Stanford, California", August 2001.

- [RFC3365] Schiller, J., "Strong Security Requirements for Internet Engineering Task Force Standard Protocols", BCP 61, RFC 3365, August 2002.
- [RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010.
- [RFC3275] Eastlake, D., Reagle, J., and D. Solo, "(Extensible Markup Language) XML-Signature Syntax and Processing", RFC 3275, March 2002.
- [RFC4288] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", RFC 4288, December 2005.
- [RFC4289] Freed, N. and J. Klensin, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", RFC 4289, December 2005.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: session initiation protocol", RFC 3261, June 2002.
- [RFC3688] Mealling, M., "The IETF XML Registry", RFC 3688, January 2004.
- [RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.
- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.
- [RFC4082] Perrig, A., Canetti, R., Tygar, J D., and B. Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction", RFC 4082, June 2005.
- [RFC5776] Roca, V., Francillon, A., and S. Faurite, "Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Asynchronous Layered Coding (ALC) and NACK-Oriented Reliable Multicast (NORM) Protocols", RFC 5776, April 2010.
- [RMT-SIMPLE-AUTH]  
Roca, V., "Simple Authentication Schemes for the ALC and

NORM Protocols",  
draft-ietf-rmt-simple-auth-for-alc-norm-06.txt (work in  
progress), December 2011.

#### Appendix A. Receiver operation (informative)

This section gives an example how the receiver of the file delivery session may operate. Instead of a detailed state-by-state specification the following should be interpreted as a rough sequence of an envisioned file delivery receiver.

1. The receiver obtains the description of the file delivery session identified by the pair: (source IP address, Transport Session Identifier). The receiver also obtains the destination IP addresses and respective ports associated with the file delivery session.
2. The receiver joins the channels in order to receive packets associated with the file delivery session. The receiver may schedule this join operation utilizing the timing information contained in a possible description of the file delivery session.
3. The receiver receives ALC/LCT packets associated with the file delivery session. The receiver checks that the packets match the declared Transport Session Identifier. If not, packets are silently discarded.
4. While receiving, the receiver demultiplexes packets based on their TOI and stores the relevant packet information in an appropriate area for recovery of the corresponding file. Multiple files can be reconstructed concurrently.
5. Receiver recovers an object. An object can be recovered when an appropriate set of packets containing Encoding Symbols for the transmission object have been received. An appropriate set of packets is dependent on the properties of the FEC Encoding ID and FEC Instance ID, and on other information contained in the FEC Object Transmission Information.
6. Objects with TOI = 0 are reserved for FDT Instances. All FDT Instances are signaled by including an EXT\_FDT header extension in the LCT header. The EXT\_FDT header contains an FDT Instance ID (i.e., an FDT version number.) If the object has an FDT Instance ID 'N', the receiver parses the payload of the instance 'N' of FDT and updates its FDT database accordingly.

7. If the object recovered is not an FDT Instance but a file, the receiver looks up its FDT database to get the properties described in the database, and assigns the file the given properties. The receiver also checks that the received content length matches with the description in the database. Optionally, if MD5 checksum has been used, the receiver checks that the calculated MD5 matches the description in the FDT database.
8. The actions the receiver takes with imperfectly received files (missing data, mismatching digestive, etc.) is outside the scope of this specification. When a file is recovered before the associated file description entry is available, a possible behavior is to wait until an FDT Instance is received that includes the missing properties.
9. If the file delivery session end time has not been reached go back to 3. Otherwise end.

#### Appendix B. Example of FDT Instance (informative)

```
<?xml version="1.0" encoding="UTF-8"?>
<FDT-Instance xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:fdt
    ietf-flute-fdt.xsd"
  Expires="2890842807">
  <File
    Content-Location="http://www.example.com/menu/tracklist.html"
    TOI="1"
    Content-Type="text/html"/>
  <File
    Content-Location="http://www.example.com/tracks/track1.mp3"
    TOI="2"
    Content-Length="6100"
    Content-Type="audio/mp3"
    Content-Encoding="gzip"
    Content-MD5="+VP5IrWploFkZWclliLDdA=="
    Some-Private-Extension-Tag="abc123"/>
</FDT-Instance>
```



## Authors' Addresses

Toni Paila  
Nokia  
Itamerenkatu 11-13  
Helsinki 00180  
Finland

Email: [toni.paila@nokia.com](mailto:toni.paila@nokia.com)

Rod Walsh  
Tampere University of Technology  
P.O. Box 553 (Korkeakoulunkatu 1)  
Tampere FI-33101  
Finland

Email: [roderick.walsh@tut.fi](mailto:roderick.walsh@tut.fi)

Michael Luby  
Qualcomm, Inc.  
3165 Kifer Rd.  
Santa Clara, CA 95051  
USA

Email: [luby@qualcomm.com](mailto:luby@qualcomm.com)

Vincent Roca  
INRIA  
655, av. de l'Europe  
Inovallee; Montbonnot  
ST ISMIER cedex 38334  
France

Email: [vincent.roca@inria.fr](mailto:vincent.roca@inria.fr)

Rami Lehtonen  
TeliaSonera  
Hatanpaan valtatie 18  
Tampere FIN-33100  
Finland

Email: [rami.lehtonen@teliasonera.com](mailto:rami.lehtonen@teliasonera.com)



RMT  
Internet-Draft  
Intended status: Standards Track  
Expires: September 13, 2012

I. Curcio  
Nokia Research Center  
R. Walsh  
J. Peltotalo  
S. Peltotalo  
Tampere University of Technology  
H. Mehta  
March 12, 2012

SDP Descriptors for FLUTE  
draft-ietf-rmt-flute-sdp-02

Abstract

This document specifies the use of SDP to describe the parameters required to begin, join, receive data from, and/or end FLUTE sessions. It also provides a Composite Session SDP media grouping semantic for grouping media streams into protocol-specific sessions, such as multiple-channel FLUTE sessions.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on September 13, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

## Table of Contents

1. Introduction . . . . .	4
2. Conventions Used in This Document . . . . .	5
2.1. New Terms . . . . .	5
3. FLUTE Descriptors . . . . .	6
3.1. FLUTE Protocol Identifier . . . . .	7
3.2. Composite Session Semantics . . . . .	8
3.2.1. Composite Session Semantics for FLUTE Sessions . . . . .	9
3.2.2. Composite Session Semantics for Protocols other than FLUTE . . . . .	10
3.3. Source IP Address . . . . .	10
3.4. Transport Session Identifier . . . . .	11
3.5. Session Timing Parameters . . . . .	12
3.6. Channelization Descriptors . . . . .	12
3.6.1. Number of Channels . . . . .	12
3.6.2. Destination IP Address and Port Number for Channels . . . . .	13
3.7. FEC Scheme . . . . .	15
3.8. Content Description Pointer . . . . .	16
3.9. Security Parameters . . . . .	17
3.10. Bandwidth Specification . . . . .	17
3.10.1. Bandwidth Specification for Composite Sessions . . . . .	18
3.11. SDP Specific Parameters . . . . .	18
4. SDP Syntax Examples . . . . .	19
5. Security Considerations . . . . .	22
6. IANA Considerations . . . . .	23
6.1. Transport Protocol . . . . .	23
6.1.1. Media formats ("fmt") . . . . .	23
6.2. Attribute Names . . . . .	23
6.3. Composite Session Token to Differentiate FLUTE Sessions . . . . .	25
7. Acknowledgements . . . . .	26
8. Contributors . . . . .	27
9. Change Log . . . . .	28
9.1. From draft-ietf-rmt-flute-sdp-01 to draft-ietf-rmt-flute-sdp-02 . . . . .	28
9.2. From draft-ietf-rmt-flute-sdp-00 to draft-ietf-rmt-flute-sdp-01 . . . . .	28
9.3. From draft-mehta-rmt-flute-sdp-06 to draft-ietf-rmt-flute-sdp-00 . . . . .	28
10. References . . . . .	30
10.1. Normative References . . . . .	30
10.2. Informative References . . . . .	31
Appendix A. Use of FEC attributes with RTP sessions (informative) . . . . .	33
Appendix B. Further Design Logic for FEC-OTI Descriptors . . . . .	34
Authors' Addresses . . . . .	35

## 1. Introduction

The Session Description Protocol (SDP) [RFC4566] provides a general-purpose format for describing multimedia sessions in announcements or invitations. SDP uses an entirely textual data format (the US-ASCII subset of UTF-8 [RFC3629]) to maximize portability among transports. SDP does not define a protocol, but only the syntax to describe a multimedia session with sufficient information to participate in that session. Session descriptions may be sent using arbitrary existing application protocols for transport (e.g. FLUTE [I-D.ietf-rmt-flute-revised], SAP [RFC2974], SIP [RFC3261], RTSP [RFC2326], HTTP [RFC2616], email etc.).

SDP defines two protocol identifiers that represent unreliable connectionless protocols. These are RTP/AVP and UDP. These are appropriate choices for multimedia streams. [RFC4145] defines protocol identifiers for connection-oriented reliable transports: TCP and TCP/TLS.

This document defines two new protocol identifiers for File Delivery over Unidirectional Transport (FLUTE) protocol [I-D.ietf-rmt-flute-revised] and other required SDP attributes for initiating a FLUTE session. The formal ABNF syntax [RFC5234] is used for the attributes. This SDP syntax is independent of whether Any Source Multicast (ASM) or Source Specific Multicast (SSM) is used to route the media.

Note, this document may also be used to describe sessions of the experimental FLUTE specification [RFC3926].

## 2. Conventions Used in This Document

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

### 2.1. New Terms

SDP instance: A syntatically complete SDP description of an SDP session, possibly stored in a single computer file.

Composite Session: An SDP mechanism that enables the grouping of media lines in to distinct sessions, so that a single SDP instance can describe multiple protocol-specific sessions.

### 3. FLUTE Descriptors

The FLUTE specification [I-D.ietf-rmt-flute-revised] describes the optional and required parameters for a FLUTE session. This document specifies the SDP parameters for FLUTE sessions that can be used for the discovery of FLUTE download and/or service announcement sessions. Listed below are the required and optional SDP parameters for FLUTE sessions (the parameters introduced, or made mandatory, by this specification but not inherited from the FLUTE specification are marked with an asterisk "\*").

The required parameters are:

- o The source IP address;
- o The number of channels in the session;
- o The destination IP address and port number for each channel in the session;
- o The Transport Session Identifier (TSI) of the session;
- o An indication that the session is a FLUTE session;
- \* The start time and end time of the session.

The optional parameters are:

- o FEC scheme (a subset of FEC Object Transmission Information);
- o Some information that tells receiver in the first place, that the session contains files that are of interest;
- o Security parameters relevant for the session;
- \* Bandwidth specification.

(Note, the best practise to provide parameters for FLUTE's optional content encoding of FDT Instances is in-band within FLUTE sessions and is therefore not specified using SDP.)

(Note, out-of-band FEC Object Transmission Information useful for FLUTE sessions is limited to SDP signalling of capabilities requirements describing FEC Encoding ID(s) and FEC Instance ID(s) as FLUTE provides header fields for machine configuration for object reception. This specification also provides a "fec-oti-extension", as an informative appendix, so that the same SDP syntax can be used to describe sessions using protocols other than FLUTE that do not



have an in-band mechanism for FEC machine configuration.)

(Note, description of congestion control parameters are not in scope of this document.)

The semantics of a FLUTE session within an SDP description differ slightly from that of the well-establish RTP session descriptions. A FLUTE session includes one or more FLUTE channels which are each a distinct media stream. (Note, the SDP specification [RFC4566] use of the term "media stream" is semantically equivalent to the FLUTE specification use of the term "channel".) Generally, each RTP media is recognised as a distinct RTP media session. Hence, to preserve harmony with RTP media sessions within SDP descriptions, the optional Composite Session mechanism is specified in this document, using the SDP Grouping Framework [RFC5888].

The description of these parameters in SDP is presented in the following sections.

### 3.1. FLUTE Protocol Identifier

The following is the ABNF syntax for an "m=" line, as specified by RFC4566 [RFC4566]:

```
media-field = "m=" media SP port [ "/" integer ] SP
              proto 1*(SP fmt) CRLF
```

We define two new values for the "proto" sub-field: FLUTE/UDP and FLUTE/UDP/ESP. The FLUTE/UDP protocol identifier specifies that the session being described will use the FLUTE [I-D.ietf-rmt-flute-revised] protocol on top of a UDP connection. The FLUTE/UDP/ESP protocol identifier specifies that the session being described will use the FLUTE [I-D.ietf-rmt-flute-revised] protocol on top of a UDP connection and session security is achieved by means of IPsec/ESP in transport mode [RFC4303].

As described below, more than one FLUTE session may be described by a single SDP instance using the Composite Session mechanism.

The fmt (format) list may be ignored for FLUTE. The fmt list of FLUTE "m=" lines MAY contain a single "\*" character to indicate that miscellaneous and unspecified MIME types (file formats) are contained in the FLUTE session. Use of any other values (MIME types) in a FLUTE fmt list is out of scope of this specification. "0" is known to be used in the fmt list to represent the same semantic as "\*", in a non-standardized way, and so implementers may take this into account. An example of a FLUTE/UDP protocol identifier is shown in Section 4.

FLUTE is a general file delivery protocol and so it is not considered necessary to identify a list of media types per FLUTE session or channel in this session description specification.

### 3.2. Composite Session Semantics

The Composite Session mechanism enables the grouping of media lines in to distinct sessions. The complete Composite Session semantics are protocol-specific - as determined by the protocol id of the grouped media lines. This section defines the Composite Session semantic generically and protocol-id-independently. Subsection 3.2.1. defines the FLUTE/UDP and FLUTE/UDP/ESP protocol identifier specific semantic.

This mechanism is useful where multiple FLUTE sessions are described as part of a larger service or application, and so where maintaining and delivering session descriptions together (with a shared delivery fate) is good practice. It may also improve bandwidth efficiency by eliminating repetition of redundant descriptors that would be necessary with multiple discrete SDP instances.

The Composite Session mechanism inherits the "group" and "mid" attributes from the SDP grouping framework [RFC5888] and introduces the "CS" (Composite Session) token as a "semantics-extension".

When the Composite Session mechanism is used: the SDP grouping framework [RFC5888] MUST be used (and requirements from that are inherited); and the "CS" token MUST be used with the "group" attribute to indicate a Composite Session grouping. The SDP grouping framework declares groups at session-level and labels media (with the "mid" attribute) at media-level. Hence, all media given "mid" values that are identified in an "a=group:CS" line belong to the same Composite Session group and inherit the grouping specified for these mid values at session-level.

The first (leftmost and uppermost) mid value declared for a Composite Session group is the Primary Media. Just as session-level attributes are inherited to media-level declarations (unless specifically overwritten by an additional media-level attribute), Primary Media attributes SHALL be inherited to all media of a particular Composite Session group. These primary media attributes (i.e. Composite Session default attributes) SHALL be overwritten at media level (for the specific media) where an attribute's syntax mandates this behaviour for media-level overwriting of SDP session-level attributes; and media-level attribute overwriting of session level attribute inheritance shall not be allowed otherwise.

### 3.2.1. Composite Session Semantics for FLUTE Sessions

When an SDP instance specifies only one FLUTE session, using the Composite Session mechanism is OPTIONAL. When an SDP instance specifies more than one FLUTE session, using the Composite Session mechanism is REQUIRED.

The Composite Session provides an unambiguous way to define multiple FLUTE sessions as distinct from multiple the media-sessions semantics of RTP. It is used for describing more than one FLUTE session in an SDP instance and so its general use and support in SDP are OPTIONAL. For SDP instances which describe multiple FLUTE sessions, the Composite Session semantics MUST be used. Whenever an SDP describes just one FLUTE session with more than a single media stream of one FLUTE protocol identifier (i.e. a FLUTE channel), use of the Composite Session semantics is RECOMMENDED.

To support simple applications, as well as ensure harmony with FLUTE SDP standards outside of the IETF [3GPP.26.346], when the Composite Session mechanism is not used for media of the FLUTE/UDP or FLUTE/UDP/ESP protocol, exactly one FLUTE session is specified within the SDP instance and all FLUTE/UDP or FLUTE/UDP/ESP media of that SDP instance belong to the same FLUTE session (this is known as the Restricted Behaviour).

The Composite Session mechanism SHOULD NOT be used where the target clients are expected to include simpler FLUTE SDP parsers, such as in some releases of 3GPP MBMS [3GPP.26.346]. In this Restricted Behaviour only one media protocol SHALL be described in one SDP instance (i.e. only FLUTE/UDP or only FLUTE/UDP/ESP or neither).

A partial example of using the Composite Session mechanism for FLUTE is shown below.

```
<other session-level attributes>
a=group:CS 1 2
a=group:CS 3
m=application 12345 FLUTE/UDP *
a=mid:1
<other media-level attributes>
m=application 12346 FLUTE/UDP *
a=mid:2
<other media-level attributes>
m=application 56789 FLUTE/UDP *
a=mid:3
<other media-level attributes>
```

The example shows two groups with the 1st and 3rd media ("m=") lines

(mid values 1 and 3) being the Primary Media for each group respectively. In the example, the media with mid value "2" inherits attributes of the media with mid value "1".) Each of these groups identifies a separate FLUTE Session. Several of the attributes subsequently specified in this document use this feature of Primary Media inheritance to all media of a Composite Session.

### 3.2.2. Composite Session Semantics for Protocols other than FLUTE

The Composite Session mechanism solves the problem of describing multiple FLUTE sessions in a single SDP instance. However, this does not place any restrictions on the use of the Composite Session mechanism with transport protocols other than FLUTE/UDP or FLUTE/UDP/ESP, nor on whether a complete SDP would include media of other transport protocols too. Specification of Composite Session semantics beyond the use of FLUTE sessions is outside the scope of this document.

### 3.3. Source IP Address

The Asynchronous Layered Coding (ALC) [RFC5775] and the Layered Coding Transport (LCT) [RFC5651] specifications require that all the channels of a single ALC/LCT session are from the same source IP address. Hence, there MUST be exactly one source IP address per FLUTE session, and therefore one source IP address per description of a FLUTE session description. Restricted behaviour is one source IP address per SDP instance. Where multiple FLUTE sessions are described within one SDP instance this means one source IP address per Composite Session.

The source IP address MUST be defined according to the source-filter attribute ("a=source-filter") [RFC4570], with the following exceptions:

- o The source-filter attribute MUST be included in any SDP instance describing FLUTE sessions and per FLUTE session described.
- o The number of source-filter attributes in any SDP describing FLUTE sessions must be exactly equal to the number of FLUTE sessions described in that SDP.
- o In the restricted behaviour of only one FLUTE session description in an SDP and no use of the Composite Session mechanism: The source-filter attribute MUST be in the session part of the session description and MUST NOT be given per media. Note, the requirement that there must not be more than a single source-filter attribute in the session part is inherited from the SDP Source Filter specification [RFC4570].

- o Where the Composite Session mechanism is used: The source-filter attribute MUST be in the media part of Primary Media of each distinct FLUTE session, and MUST NOT be given in other media declarations but these, nor in the session-level part of the SDP.
- o Exactly one source address is specified by any instance of this attribute. Exactly one source address MUST be given in an inclusive-mode "src-list". Exclusive-mode MUST NOT be used.
- o The "\*" value MUST be used for the "dest-address" sub-field, even when the FLUTE session employs only a single channel (e.g. a multicast group).

An example of the use of this attribute is:

```
a=source-filter: incl IN IP6 * 2001:0DB8:1:2:240:96FF:FE25:8EC9
```

This example uses the source-filter attribute to describe an IPv6 source address.

### 3.4. Transport Session Identifier

The combination of the TSI and the source IP address identifies a FLUTE session. Each TSI MUST uniquely identify a FLUTE session for a given source IP address during the time that the session is active and also for a large time before and after the active session time. This requirement is inherited from LCT [RFC5651]. Note, the SDP specification [RFC4566] advises that sessions expire 30 minutes after the session-end time given in the t-field. This should be considered an absolute minimum interpretation of a "large time". TSI reuse is NOT RECOMMENDED whenever possible (thus, making "large time" unbounded regarding TSI reuse).

The TSI MUST be described by the "flute-tsi" attribute.

There MUST be exactly one occurrence of the "flute-tsi" attribute per FLUTE session description of an SDP instance.

- o The number of "flute-tsi" attributes in any SDP describing FLUTE sessions must be exactly equal to the number of FLUTE sessions described in that SDP.
- o In the restricted behaviour of only one FLUTE session description in an SDP and no use of the Composite Session mechanism: The "flute-tsi" attribute MUST be in the session part of the session description and MUST NOT be given per media. A "flute-tsi" attribute in the session-part SHALL be used to identify restricted behaviour.

- o Where the Composite Session mechanism is used: The "flute-tsi" attribute MUST be in the media part of Primary Media of each distinct FLUTE session, and MUST NOT be given in other media declarations but these, and MUST NOT be given in the session-level part of the SDP.

The syntax for the attribute in ABNF is given below:

```
flute-tsi-line = "a=flute-tsi:" tsi CRLF
tsi = 1*DIGIT
```

Note, the range of values a TSI can adopt depends on the bitlength of the TSI for a session as defined by RFC5651 [RFC5651].

### 3.5. Session Timing Parameters

The SDP timing field "t=" [RFC4566] MUST be used to indicate the FLUTE session start and end times. This value applies to all FLUTE and transport sessions defined in a single SDP instance and, thus, FLUTE sessions of different timing values need to be declared in different SDP instances.

Note, implementers may assume reasonable clock synchronisation between SDP description, receiver wall clock and sender wall clock (within 60 seconds) unless specified otherwise for a specific deployment. The method to achieve this is beyond the scope of the current specifications, but may use well known and mature approaches such as SNTP [RFC5905].

### 3.6. Channelization Descriptors

This section specifies the description of the channel(s) used within a FLUTE session. The required parameters for channelization description are:

- o Number of channels
- o Destination IP address and port number for channels

#### 3.6.1. Number of Channels

The FLUTE specification allows the use of multiple channels (e.g. multicast groups) to transport the files of a single FLUTE session. This is referred to as FLUTE session channelization in this document. A FLUTE channel is equivalent to an ALC/LCT channel. An ALC/LCT channel is defined by the combination of a sender and an address associated with the channel by the sender. Details of each channel are defined by SDP media-level information also described in this

document. The number of channels is calculated by summing the number of unique destination IP address and port number pairs for a certain FLUTE session (assignment of media to FLUTE sessions is done with presence of absence of the Composite Session grouping).

The OPTIONAL "flute-ch" attribute describes the number of channels used by the source to transmit the FLUTE session. When present, it is used to validate the channel number calculation based on the number of destination address/port pairs, and it is expected to be used where SDP proxies and other automatic and manual editing that introduces errors would cause bad failure conditions at the client.

When the "flute-ch" attribute is used:

- o The number of "flute-ch" attributes in any SDP describing FLUTE sessions MUST be exactly equal to the number of FLUTE sessions described in that SDP. A client SHOULD discard all of an SDP instance if this condition is not met. Alternative behaviour, such as retries at delivery, error reporting and partial use of SDP instances known to include errors, are beyond the scope of this document.
- o In the restricted behaviour of only one FLUTE session description in an SDP and no use of the Composite Session mechanism: Any "flute-ch" attribute MUST be in the session part of the session description and MUST NOT be given per media.
- o Where the Composite Session mechanism is used: The "flute-ch" attribute MUST be in the media part of Primary Media of each distinct FLUTE session, and MUST NOT be given in other media declarations but these, nor in the session-level part of the SDP.

The syntax for the attribute in ABNF is given below:

```
flute-channel-line = "a=flute-ch:" ch CRLF
ch = integer
;integer is as defined in [RFC4566], and its value is the number of
;channels used by the source to transmit data in a FLUTE session.
```

### 3.6.2. Destination IP Address and Port Number for Channels

SDP media-level information describes one or more channels. The channel parameters MUST be given per channel and are:

- o Destination IP address
- o Destination port number

The destination IP address MUST be defined according to the connection data field ("c=") of SDP [RFC4566]. The destination port number MUST be defined according to the "port" sub-field of the media description field ("m=") of SDP [RFC4566].

A "c=" line can describe multiple addresses by using "number of addresses" sub-field, and also an "m=" line can describe multiple ports by using "number of ports" sub-field. So multiple channels can be described by using one "c=" line and one "m=" line (called "slash notation").

When more than one channel is used in a multicast FLUTE session, it is RECOMMENDED that the channels are differentiated based on destination IP address, and channels are not differentiated based on destination port (although those ports could be same or different for each of the channels). Whenever destination port number is used to differentiate between FLUTE channels, the same destination IP address MUST be used for all channels in that FLUTE session. Note, when more than one channel is used in a unicast FLUTE session, the channels have to be differentiated based on destination ports, as only one destination IP address could be used.

In the case (always with a unicast session) where the same destination IP address is used for all the channels of the session and only the destination port number differentiates channels, the destination IP address MAY be given by the connection data field at session-level for all channels (if so, the connection data field MUST NOT be used at media-level).

In the case where each channel has a different destination IP address, the destination IP addresses MUST be given at media-level, i.e. following an "m=" line.

Some applications of FLUTE and FLUTE SDP benefit from identifying an ordinal sequence of FLUTE channels in a FLUTE session. In this case, the sequence of multiple channels MUST be determined by the order in which their media descriptions are defined in the session description (i.e. the first media description gives the first channel in the sequence). This applies individually to each FLUTE session of an SDP whether one or more FLUTE sessions are described. In the case of the slash notation usage for specifying multiple destination addresses or ports, the order of the channel sequence MUST be lowest value first and highest last. Note, slash notation for both destination IP address and port would be incompatible with requirement to not use both destination IP address and port to differentiate channels in a FLUTE session and thus slash notation for both destination IP address and port is not allowed for a single FLUTE session - i.e. for a single composite session (when the SDP describes multiple FLUTE



sessions) or for a single SDP instance (when only one FLUTE session is described).

Also we need to indicate the presence of a FLUTE session on a certain channel. This is done by using the "m=" line in the SDP description as shown in the following example:

```
m=application 12345 FLUTE/UDP *  
c=IN IP6 FF33::8000:1
```

In the above SDP attributes, the "m=" line indicates the media used and the "c=" line together with "m=" line's "port" sub-field indicates the corresponding channel's address and port respectively. Thus, in the above example, the media is transported on a channel that uses FLUTE over UDP. Further, the "c=" line indicates the channel's address, which, in this case, is an IPv6 address, and "m=" line indicates the channel's port (12345).

Note, the value of the destination IP address can indicate whether a multicast media belongs to an ASM or a SSM group as described by [RFC4607].

### 3.7. FEC Scheme

An SDP description for a FLUTE session MAY include information for one or more FEC schemes (a subset of FEC Object Transmission Information (FEC-OTI) [RFC5052] that is useful for reception capability evaluation at the receiver). FEC parameters can be placed either at session-level or at media-level, although it is RECOMMENDED to place them at session-level. Furthermore, if FEC parameters are placed at media level (contrary to the recommendation) and the Composite Session mechanism is used, they SHOULD only be placed in the Primary Media for any FLUTE session description. If FEC declarations on both session and media level use the same reference number (fec-ref) then the media level declaration takes precedence for that media component. FEC parameters include:

- o FEC Encoding ID
- o FEC Instance ID (for FEC Encoding IDs 128-255)

Where any FEC scheme is given, FEC parameters MUST be described in a "FEC-declaration" attribute. Multiple instances of this attribute MAY exist both at session-level and media-level. If an instance exists at session-level (or in a Primary Media), a reference to it MAY be used at media-level, and an attribute "FEC" MUST be defined for this purpose.

The syntax for the attributes in ABNF is given below:

```
fec-declaration-line = "a=FEC-declaration:" fec-ref SP
    fec-enc-id [ ";" SP fec-inst-id ] CRLF
fec-ref = 1*3DIGIT
;value is the SDP-internal identifier for FEC-declaration
```

```
fec-enc-id = "encoding-id=" enc-id
enc-id = 1*DIGIT
;value is the FEC Encoding ID used
```

```
fec-inst-id = "instance-id=" inst-id
inst-id = 1*DIGIT
;value is the FEC Instance ID used
```

```
fec-line = "a=FEC:" fec-ref CRLF
```

Examples of FEC scheme information are shown in Section 4.

The FEC parameters are for capabilities description for the session. These parameters do not mandate a certain machine configuration but instead indicate which capabilities might be needed for successful reception of objects from specific channels. (Note, any "FDT-like" fuller description of files in the session could give the FEC parameters per file). FLUTE's FDT syntax (and codepoint header field usage) allows complete specification of these FEC parameters in-band with FLUTE (per file). Thus machine configuration can be performed using FLUTE alone.

A more complete list of notes on the design logic for the FEC-OTI descriptors is provided as an appendix to this document.

### 3.8. Content Description Pointer

The syntax of the information that tells receiver, in the first place, that the session contains files that are of interest is out of scope of this document. However, the SDP MAY include a content description pointer at the session-level and/or media-level (including Primary Media of Composite Sessions) to enable efficient linkage to such information.

The content description pointer attribute describes to the receiver(s) the URI where the content description is stored. The content description pointer MUST be defined according to the "content-desc" attribute.

The syntax for the attribute in ABNF is given below:

```
content-desc-line = "a=content-desc:" URI-reference CRLF
;URI-reference is as defined in [RFC3986].
```

An example of content description pointer is shown in Section 4.

### 3.9. Security Parameters

To support the baseline secure ALC operation [RFC5775], subsection 3.2.1. defines the FLUTE/UDP/ESP protocol identifier for a FLUTE session where the session security is achieved using IPsec/ESP in a transport mode.

This document describes two informative ways which can be used to deliver the needed security parameters for IPsec/ESP Security Association (SA). Firstly, the "key-mgmt" attribute defined in [RFC4567] can be used to carry messages, as specified by a key management protocol, in order to secure the media. In [RFC4567] the usage of the Multimedia Internet KEYing (MIKEY) key management protocol [RFC3830] is defined and guidelines for the registration of additional key management protocols is also provided.

Secondly, the "crypto" attribute defined in [RFC4568] provides cryptographic key distribution capabilities, but it is intended for usage when keying material is protected along with the signalling (i.e. when the session description itself is retrieved using a secure method). [RFC4568] describes this attribute primarily for a unicast offer/answer models, but also for "offer only" mode with multicast delivery, and sets a policy by mandating that "the sender MUST include exactly one crypto attribute" (implying "one crypto attribute per media", although [RFC4568] is slightly ambiguous on this). The definition of composite session adds a case to the usage of the crypto attribute and so, a crypto attribute definition in a Composite Session Primary Media SHALL BE inherited to all media of that particular Composite Session group, except where individual child media overwrite this with their own crypto attribute.

### 3.10. Bandwidth Specification

The specification of bandwidth (data rate) is OPTIONAL and where included in the SDP it SHALL adhere to the following rules.

The maximum bit-rate required by a particular FLUTE media line (one or more FLUTE channels, depending on the usage or IP address and port ranges) MAY be specified. In this case it is RECOMMENDED to use the TIAS bandwidth modifier [RFC3890] on media-level, although the AS bandwidth modifier [RFC4566] MAY be used on media-level.

The session bit-rate MAY also be specified. In this case it is

RECOMMENDED to use the TIAS bandwidth modifier and the "a=maxprate" attribute for the session, and again AS is optional but not recommended.

TIAS is generally preferred as it allows the calculation of the bit-rate in environments with translation of IP version or transport protocol, where as AS does not and thus adds significant complexity in such environments.

Any Transport Independent (TIAS) bandwidth SHALL be the largest sum of the sizes of all FLUTE/UDP or FLUTE/UDP/ESP packets transmitted during any one second long period of the FLUTE session, depending on which level it is being used, expressed as kilobits. The size of the packet SHALL include all FLUTE, ALC, LCT and any extensions headers and payload. IP, UDP and ESP headers are excluded from the TIAS bit-rate calculation. Any Application Specific (AS) bandwidth SHALL be the largest sum of the sizes of all FLUTE/UDP or FLUTE/UDP/ESP packets transmitted during any one second long period for the related media line(s), expressed as kilobits. The size of the packet SHALL be the complete packet, i.e. IP, UDP, ESP and FLUTE headers, and the data payload.

#### 3.10.1. Bandwidth Specification for Composite Sessions

Where the multimedia session bit-rate is specified (at SDP session level) this applies to all media, irrespective of whether the Composite Session mechanism is used to describe multiple sessions (e.g. multiple FLUTE sessions). So if multiple Composite Sessions are described in a single SDP instance and SDP session-level bit-rate is described, this session-level bit-rate would not relate to any single Composite Session.

A normal TIAS or AS bit-rate declaration at the Primary Media level is to be interpreted as media-specific and not imply any inheritance to other media of the same Composite Session. It is RECOMMENDED that aggregate Composite Session bandwidth is calculated as the sum of all constituent media bit-rate declarations. Specification of a descriptor specifically for aggregate Composite Session bandwidth is beyond the scope of this document.

#### 3.11. SDP Specific Parameters

SDP [RFC4566] also mandates three parameters ("v=", "o=" and "s=") that would be present in every FLUTE SDP instance regardless of their usefulness to the FLUTE session description.

#### 4. SDP Syntax Examples

This section gives examples of the use of SDP attributes to describe a FLUTE session.

```
v=0
o=user123 2890844526 2890842807 IN IP6 2001:0DB8::112E:144A:1E24
s=File delivery session example
i=More information
t=2873397496 2873404696
a=source-filter: incl IN IP6 * 2001:0DB8:1:2:240:96FF:FE25:8EC9
a=flute-tsi:3
a=flute-ch:2
a=FEC-declaration:0 encoding-id=0
a=FEC-declaration:1 encoding-id=129; instance-id=0
a=content-desc:http://www.example.com/flute-sessions/session001
m=application 12345 FLUTE/UDP *
c=IN IP6 FF33::8000:1
a=FEC:0
m=application 12346 FLUTE/UDP *
c=IN IP6 FF33::8000:2
a=FEC:1
```

Figure 1: An SDP Instance for a FLUTE Session with Two Channels

Figure 1 shows an example SDP instance for a FLUTE session with two channels.

The attribute defined in the line "a=source-filter: incl IN IP6 \* 2001:0DB8:1:2:240:96FF:FE25:8EC9" describes a source filter. In this example the source indicates that the receiver(s) should include the given IP address (2001:0DB8:1:2:240:96FF:FE25:8EC9) into the session. It should be noted that although other possibilities may be used, in this case only the incl and \* attributes may be used in the above attribute.

The attribute defined in the line "a=flute-tsi:3" describes the Transport Session Identifier for the session. The pair made of the source IP address and the TSI together uniquely identifies a FLUTE session.

The source indicates in the above example that it will transmit data in the FLUTE session on two channels (a=flute-ch:2). The source then specifies the channels.

The "a=FEC-declaration" lines describes two different FEC schemes used in the FLUTE session.

The "a=content-desc" line describes the URI where the content description is stored.

The line "m=application 12345 FLUTE/UDP \*" indicates the media used for the channel. In this example, there are two "m=" lines for the two channels described.

The destination addresses for the channels are given in the "c=" lines. These also show to the receiver(s) that the channels are two (maybe more in other cases) consecutive channels of an ordinal sequence of channels.

The "a=FEC" lines at media-level reference FEC declarations at session-level ("a=FEC-declaration").

```
v=0
o=user123 2890844526 2890842807 IN IP6 2001:0DB8::112E:144A:1E24
s=File delivery session example
i=More information
t=2873397496 2873404696
a=source-filter: incl IN IP6 * 2001:0DB8:1:2:240:96FF:FE25:8EC9
a=flute-tsi:2
a=flute-ch:1
m=application 12345 FLUTE/UDP *
c=IN IP6 FF33::8000:1
a=FEC-declaration:0 encoding-id=129; instance-id=0
```

Figure 2: An SDP Instance for a FLUTE Session with One Channel

Figure 2 shows an example SDP instance for a FLUTE session with one channel.

```
v=0
o=user123 2890844526 2890842807 IN IP6 2001:0DB8::112E:144A:1E24
s=File delivery session example
i=More information
t=2873397496 2873404696
a=source-filter: incl IN IP6 * 2001:0DB8:1:2:240:96FF:FE25:8EC9
a=FEC-declaration:0 encoding-id=0
a=FEC-declaration:1 encoding-id=129; instance-id=0'
a=group:CS 1 2
a=group:CS 3 4
m=application 12345 FLUTE/UDP *
c=IN IP6 FF33::8000:1
a=flute-tsi:1
a=FEC:0
a=mid:1
m=application 12346 FLUTE/UDP *
c=IN IP6 FF33::8000:2
a=mid:2
m=application 12347 FLUTE/UDP *
c=IN IP6 FF33::8000:3
a=flute-tsi:2
a=FEC:1
a=mid:3
m=application 12348 FLUTE/UDP *
c=IN IP6 FF33::8000:4
a=mid:4
```

Figure 3: An SDP Instance for Multiple FLUTE Sessions using the Composite Session Mechanism

Figure 3 shows an example SDP instance for multiple FLUTE sessions using the Composite Session mechanism.

## 5. Security Considerations

See [RFC4566] for security considerations specific to the Session Description Protocol in general. See also [RFC4570] for security consideration related to source address filters.

[I-D.ietf-rmt-flute-revised] provides security consideration regarding FLUTE sessions, from which Section 7.3.1 specifically addresses attacks against the session description. The current document does not introduce additional security considerations beyond these prior specifications.



## 6. IANA Considerations

### 6.1. Transport Protocol

The "proto" sub-field of the media description field ("m=") describes the transport protocol used. This document registers two values: "FLUTE/UDP" is a reference to FLUTE [I-D.ietf-rmt-flute-revised] running over UDP/IP. "FLUTE/UDP/ESP" is a reference to FLUTE [I-D.ietf-rmt-flute-revised] running over UDP/IP and the session security is achieved by means of IPsec/ESP in a transport mode [RFC4303]

#### 6.1.1. Media formats ("fmt")

FLUTE media using the "FLUTE/UDP" or "FLUTE/UDP/ESP" proto value may use the character "\*" as their "fmt" value. The "\*" character represents a wild card which indicates that miscellaneous and unspecified MIME types are contained in the FLUTE session. Alternatively a list of MIME types (file formats) may be given in the "fmt" list. These formats SHOULD be registered. Use of an existing MIME subtype for the format is encouraged. If no MIME subtype exists, it is RECOMMENDED that a suitable one is registered through the IETF process as described in RFC4289 [RFC4289].

### 6.2. Attribute Names

As recommended by [RFC4566], the new attribute names "flute-tsi", "flute-ch", "FEC-declaration", "FEC", "FEC-OTI-extension" and "content-desc" should be registered with IANA, as follows:

The following contact information shall be used for all registrations included here:

Contact: Rod Walsh  
EMail: rod.walsh (at) nokia.com

SDP Attribute ("att-field"):  
Attribute name: flute-tsi  
Long form: FLUTE Transport Session Identifier  
Type of name: att-field  
Type of attribute: Session level or media level  
Subject to charset: No  
Purpose: See this document  
Reference: This document  
Values: See this document

SDP Attribute ("att-field"):  
Attribute name: flute-ch

Long form: Number of Channels in a FLUTE Session  
Type of name: att-field  
Type of attribute: Session level or media level  
Subject to charset: No  
Purpose: See this document  
Reference: This document  
Values: See this document

SDP Attribute ("att-field"):  
Attribute name: FEC-declaration  
Long form: Forward Error Correction Declaration  
Type of name: att-field  
Type of attribute: Session level or media level  
Subject to charset: No  
Purpose: See this document  
Reference: This document  
Values: See this document

SDP Attribute ("att-field"):  
Attribute name: FEC  
Long form: A Reference to FEC Declaration  
Type of name: att-field  
Type of attribute: Media level  
Subject to charset: No  
Purpose: See this document  
Reference: This document  
Values: See this document

SDP Attribute ("att-field"):  
Attribute name: FEC-OTI-extension  
Long form: FEC Object Transmission Information extension  
Type of name: att-field  
Type of attribute: Session level or media level  
Subject to charset: No  
Purpose: See this document  
Reference: This document  
Values: See this document

SDP Attribute ("att-field"):  
Attribute name: content-desc  
Long form: Content Description Pointer  
Type of name: att-field  
Type of attribute: Session level or media level  
Subject to charset: No  
Purpose: See this document  
Reference: This document  
Values: See this document

### 6.3. Composite Session Token to Differentiate FLUTE Sessions

IANA needs to register the following new 'semantics' attribute for the SDP grouping framework [RFC5888]:

Semantics	Token	Reference
-----	----	-----
Composite Session	CS	This document

It should be registered in the SDP parameters registry (<http://www.iana.org/assignments/sdp-parameters>) under Semantics for the "group" SDP Attribute.

## 7. Acknowledgements

The authors would like to thank all the people who gave feedback on this document.

## 8. Contributors

Magnus Westerlund  
Ericsson Research  
Ericsson AB  
SE-164 80 Stockholm  
Sweden  
EMail: Magnus.Westerlund (at) ericsson.com

Joerg Ott  
Aalto University  
Otakaari 5A  
FI-02150 Espoo  
Finland  
EMail: jo (at) netlab.tkk.fi

## 9. Change Log

### 9.1. From draft-ietf-rmt-flute-sdp-01 to draft-ietf-rmt-flute-sdp-02

#### Author affiliation update

Changed the term "FEC OTI" to "FEC scheme" where appropriate to avoid misunderstandings between the current specification's use for OTI capability requirements description and the wider application of OTI for receiver FEC configuration for LCT/FLUTE objects (the latter being beyond the scope of the present specification).

More specific reference to Section 7.3.1 in [I-D.ietf-rmt-flute-revised] for attacks against session descriptions is added to Section 5.

Moved the note on out of scope congestion control to a more sensible place.

### 9.2. From draft-ietf-rmt-flute-sdp-00 to draft-ietf-rmt-flute-sdp-01

New value for the "proto" sub-field, i.e. FLUTE/UDP/ESP defined.

New section "Security Parameters" added. This section defines two possibilities which can be used to set up the needed security parameters for ESP.

Clarifications for: media-level attribute overwriting of Composite Session Primary Media attributes; media protocol usage for Restricted Behaviour only one; crypto attribute usage.

### 9.3. From draft-mehta-rmt-flute-sdp-06 to draft-ietf-rmt-flute-sdp-00

Document name changed to reflect the status as an official working group draft.

All editorial notes removed, except those related to open CC and SEC discussion.

Clarified Composite Session Semantics regarding primary media. "The first media line declared..." changed to "The first (leftmost) mid value declared...".

Clarifying note added to disambiguate the apparent difference between LCT and SDP "guard interval" for session expiry and session id reuse. This include the non mandatory recommendation to avoid TSI reuse whenever possible (e.g. for a 48bit TSI and non-extremely-high-frequency session changes from a specific sender, this would often be

the case).

Documented the previously implicit assumption regarding wall clock synchronization.

RFC5445 reference corrected (now in the informative references section).

## 10. References

### 10.1. Normative References

- [I-D.ietf-rmt-flute-revised]  
Paila, T., Walsh, R., Luby, M., Roca, V., and R. Lehtonen,  
"FLUTE - File Delivery over Unidirectional Transport",  
draft-ietf-rmt-flute-revised-13 (work in progress),  
January 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate  
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5775] Luby, M., Watson, M., and L. Vicisano, "Asynchronous  
Layered Coding (ALC) Protocol Instantiation", RFC 5775,  
April 2010.
- [RFC5651] Luby, M., Watson, M., and L. Vicisano, "Layered Coding  
Transport (LCT) Building Block", RFC 5651, October 2009.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax  
Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session  
Description Protocol", RFC 4566, July 2006.
- [RFC4570] Quinn, B. and R. Finlayson, "Session Description Protocol  
(SDP) Source Filters", RFC 4570, July 2006.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO  
10646", STD 63, RFC 3629, November 2003.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform  
Resource Identifier (URI): Generic Syntax", STD 66,  
RFC 3986, January 2005.
- [RFC5052] Watson, M., Luby, M., and L. Vicisano, "Forward Error  
Correction (FEC) Building Block", RFC 5052, August 2007.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description  
Protocol (SDP) Grouping Framework", RFC 5888, June 2010.
- [RFC3890] Westerlund, M., "A Transport Independent Bandwidth  
Modifier for the Session Description Protocol (SDP)",  
RFC 3890, September 2004.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)",  
RFC 4303, December 2005.



- [RFC4567] Arkko, J., Lindholm, F., Naslund, M., Norrman, K., and E. Carrara, "Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP)", RFC 4567, July 2006.
- [RFC4568] Andreassen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, July 2006.
- [RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, August 2004.

## 10.2. Informative References

- [RFC3926] Paila, T., Luby, M., Lehtonen, R., Roca, V., and R. Walsh, "FLUTE - File Delivery over Unidirectional Transport", RFC 3926, October 2004.
- [RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.
- [RFC4145] Yon, D. and G. Camarillo, "TCP-Based Media Transport in the Session Description Protocol (SDP)", RFC 4145, September 2005.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.
- [RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.
- [RFC4289] Freed, N. and J. Klensin, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", BCP 13, RFC 4289, December 2005.
- [RFC5445] Watson, M., "Basic Forward Error Correction (FEC) Schemes", RFC 5445, March 2009.

- [RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.
- [3GPP.26.346]  
3GPP, "Multimedia Broadcast/Multicast Service (MBMS);  
Protocols and codecs", 3GPP TS 26.346 10.2.0,  
December 2011.

## Appendix A. Use of FEC attributes with RTP sessions (informative)

The "FEC-declaration" and "FEC" attributes provide general FEC-OTI information in FEC Encoding ID and FEC Instance ID values. These may also be used for RTP sessions employing same FEC Building Block (e.g. as is done for 3GPP MBMS [3GPP.26.346]). However, semantics of RTP are different from FLUTE (FEC is per session not per object) and RTP does not have in-band mechanism to signal FEC OTI extensions. Thus, RTP FEC declarations are expected to be used for machine configuration as well as capability requirements specification (for FLUTE it is generally only the latter).

Hence, the FLUTE SDP, defined in this document, may be extended using a "FEC-OTI-extension" attribute, depending on the configuration needs of the FEC decoder used and the lack of an alternative means to signal the extended FEC-OTI information. The purpose of extended FEC-OTI information is to define FEC code/instance-specific OTI required for receiver FEC payload configuration. The contents of such an extension would be FEC code-specific and exact specification, beyond adherence to the ABNF below, needs to be specified by any FEC code using this attribute, and hence is outside the scope of this Appendix.

A "FEC-OTI-extension" attribute must be immediately preceded by its associated "FEC-declaration" attribute and so the full FEC-OTI, including extension, will be found in two neighbouring attribute lines. The fec-ref value binds a "FEC-OTI-extension" and "FEC-declaration attribute" pair.

The syntax for the attribute in ABNF is given below:

```
fec-oti-extension-line = "a=FEC-OTI-extension:" fec-ref SP
                        oti-extension CRLF
oti-extension = base64
base64 = *base64-unit [base64-pad]
base64-unit = 4base64-char
base64-pad = 2base64-char "==" / 3base64-char "="
base64-char = ALPHA / DIGIT / "+" / "/"
```

## Appendix B. Further Design Logic for FEC-OTI Descriptors

There are several reasons that the FEC Encoding and Instance IDs are optional capabilities descriptions:

1. It is not always necessary to explicitly describe the FEC capabilities in advance of the session - e.g. for simple (and short) sessions it can be more elegant to discover this from the session (FDT) itself (even when some mechanism for machine-readable session parameters, such as IP addresses and ports, is wanted in advance).
2. There may be some other out-of-band discovery of FEC capability requirements (e.g. well known-FEC/standardised capabilities for a certain application, verbal agreement between a group, etc.) that provides the FEC capability information. This document does not want to prevent this, and in this case repeating the information in SDP instance would be unnecessary and wasteful (and probably result in implementations not following the flute-sdp specification).
3. FLUTE defaults to Compact No-Code FEC [RFC5445] and support for this is mandatory for FLUTE anyway so it is a given (capability requirement) which does not need to be described by the SDP instance. In cases where only Compact No-Code FEC is required, there is no use in specifying any FEC Encoding (and Instance) IDs in the SDP instance(though it is allowed).
4. In cases where a FLUTE session description (SDP instance) is not defined once for all time, it is possible that the FEC usage is not known in advance and the FEC capabilities would only be added to the SDP in a later version of that SDP instance when the FEC codes have been selected (e.g. a larger audience may suggest stronger FEC to make FLUTE delivery more reliable, whereas additional bi-directional messages may be scalable for smaller groups).
5. Also, in cases where a FLUTE session description (SDP instance) is very static (e.g. once for all time for that session), it is possible that the FEC usage is not known in advance and it needs to be left to some other mechanism (e.g. FDT) to discover any FEC capability requirements set closer to the session transmission - with the same examples as mentioned above.

Also, in a complex case of very many FEC codes being used in the session giving a full list in SDP instance is not seen as being reasonable (but this is likely to be a rare case anyway).

## Authors' Addresses

Igor D.D. Curcio  
Nokia Research Center  
P.O. Box 88 (Tieteenkatu 1)  
Tampere FI-33721  
Finland

Email: igor.curcio (at) nokia.com

Rod Walsh  
Tampere University of Technology  
P.O. Box 553 (Korkeakoulunkatu 1)  
Tampere FI-33101  
Finland

Email: roderick.walsh (at) tut.fi

Jani Peltotalo  
Tampere University of Technology  
P.O. Box 553 (Korkeakoulunkatu 1)  
Tampere FI-33101  
Finland

Email: jani.peltotalo (at) ieee.org

Sami Peltotalo  
Tampere University of Technology  
P.O. Box 553 (Korkeakoulunkatu 1)  
Tampere FI-33101  
Finland

Email: sami.peltotalo (at) tut.fi

Harsh Mehta

Email: harsh.mehta (at) gmail.com



RMT  
Internet-Draft  
Intended status: Informational  
Expires: September 15, 2011

B. Adamson  
Naval Research Laboratory  
V. Roca  
INRIA  
H. Asaeda  
Keio University  
March 14, 2011

Security and Reliable Multicast Transport Protocols: Discussions and  
Guidelines  
draft-ietf-rmt-sec-discussion-06

Abstract

This document describes general security considerations for the Reliable Multicast Transport (RMT) Working Group set of building blocks and protocols. An emphasis is placed on risks that might be resolved in the scope of transport protocol design. However, relevant security issues related to IP Multicast control-plane and other concerns not strictly within the scope of reliable transport protocol design are also discussed. The document also begins an exploration of approaches that could be embraced to mitigate these risks. The purpose of this document is to provide a consolidated security discussion and provide a basis for further discussions and potential resolution of any significant security issues that may exist in the current set of RMT standards.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on September 15, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the

document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.



## Table of Contents

1. Introduction . . . . .	5
1.1. Conventions Used in this Document . . . . .	6
2. Quick Introduction to RMT Protocols and their Use . . . . .	6
2.1. The Two Families of CDP . . . . .	6
2.2. RMT Protocol Characteristics . . . . .	7
2.3. Target Use Case Characteristics . . . . .	7
3. Some Security Threats . . . . .	8
3.1. Control-Plane Attacks . . . . .	9
3.1.1. Control Plane Monitoring . . . . .	9
3.1.2. Unauthorized (or Malicious) Group Membership . . . . .	10
3.2. Data-Plane Attacks . . . . .	10
3.2.1. Rogue Traffic Generation . . . . .	11
3.2.2. Sender Message Spoofing . . . . .	11
3.2.3. Receiver Message Spoofing . . . . .	12
3.2.4. Replay Attacks . . . . .	12
4. General Security Goals . . . . .	13
4.1. Network Protection . . . . .	14
4.2. Protocol Protection . . . . .	14
4.3. Content Protection . . . . .	14
4.4. Privacy . . . . .	15
5. Elementary Security Techniques . . . . .	15
6. Technological Building Blocks . . . . .	17
6.1. IPsec . . . . .	17
6.1.1. Benefits . . . . .	17
6.1.2. Requirements . . . . .	18
6.1.3. Limitations . . . . .	18
6.2. Group MAC . . . . .	19
6.2.1. Benefits . . . . .	19
6.2.2. Requirements . . . . .	19
6.2.3. Limitations . . . . .	19
6.3. Digital Signatures . . . . .	19
6.3.1. Benefits . . . . .	19
6.3.2. Requirements . . . . .	20
6.3.3. Limitations . . . . .	20
6.4. TESLA . . . . .	20
6.4.1. Benefits . . . . .	20
6.4.2. Requirements . . . . .	21
6.4.3. Limitations . . . . .	21
6.5. Source-Specific Multicast . . . . .	21
6.5.1. Requirements . . . . .	22
6.5.2. Limitations . . . . .	22
6.5.3. Source-Based and Receiver-Based Attacks . . . . .	22
6.6. Summary . . . . .	23
7. Security Infrastructure . . . . .	23
8. New Threats Introduced by the Security Scheme Itself . . . . .	24
9. Consequences for the RMT and MSEC Working Group . . . . .	24

9.1. RMT Transport Message Security Encapsulation Header . . .	24
10. IANA Considerations . . . . .	25
11. Security Considerations . . . . .	25
12. Acknowledgments . . . . .	25
13. References . . . . .	25
13.1. Normative References . . . . .	25
13.2. Informative References . . . . .	27
Authors' Addresses . . . . .	27

## 1. Introduction

The Reliable Multicast Transport (RMT) Working Group has produced a set of building block (BB) and protocol instantiation (PI) specifications for reliable multicast data transport. Some present PIs defined within the scope of RMT include Asynchronous Layered Coding (ALC) [RFC5775], NACK-Oriented Reliable Multicast (NORM) [RFC5740], and the File Delivery over Unidirectional Transport (FLUTE) [I-D.ietf-rmt-flute-revised] application that is built on top of ALC. These can be considered "Content Delivery Protocols" (CDP) as described in [Neumann05]. In this document, the term CDP will refer indifferently to either ALC or NORM, with their associated BBs.

The use of these BBs and PIs raises some new security risks. For instance, these protocols share a novel set of Forward Error Correction (FEC) and congestion control building blocks that present some new capabilities for Internet transport, but may also pose some new security risks. Yet some security risks are not related to the particular BBs used by the PIs, but are more general. Reliable multicast transport sessions are expected to involve at least one sender and multiple receivers. Thus, the risk of and avenues to attack are implicitly greater than that of point-to-point (unicast) transport sessions. Also the nature of IP multicast can expose other coexistent network flows and services to risk if malicious users exploit it. The classic Any-Source Multicast (ASM) [RFC1112] model of multicast routing allows any host to join an IP multicast group and send traffic to that group. This poses many potential security challenges. And, while the emerging Source-Specific Multicast (SSM) [RFC3569], [RFC4607] model that enables users to receive multicast data sent only from specified sender(s) simplifies some challenges, there are still specific issues. For instance, possible areas of attack include those against the control plane where malicious hosts join IP multicast groups to cause multicast traffic to be directed to parts of the network where it is not needed or desired. This may indirectly cause denial-of-service (DoS) to other network flows. Also, attackers may transmit erroneous or corrupt messages to the group or employ strategies such as replay attack within the "data plane" of protocol operation.

The goals of this document are therefore to:

1. Define the possible general security goals: protecting the network infrastructure, and/or the protocol, and/or the content, and/or the user (e.g., its privacy);
2. List the possible elementary security services that will make it possible to fulfill the general security goals. Some of these services are generic (e.g., object and/or packet integrity),

while others are specific to RMT protocols (e.g., congestion control specific security schemes);

3. List some technological building blocks and solutions that can provide the desired security services;
4. Highlight the CDP and the use-case specificities that will impact security. Indeed, the set of solutions proposed to fulfill the security goals will greatly be impacted by these considerations;

In some cases, the existing RMT documents already discuss the risks and outline approaches to solve them, at least partially. The purpose of this document is to consolidate this content and provide a basis for further discussion and potential resolution of any significant security issues that may exist.

#### 1.1. Conventions Used in this Document

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. Quick Introduction to RMT Protocols and their Use

#### 2.1. The Two Families of CDP

The ALC and NORM classes of CDP are designed to reliably deliver content to a group of multicast receivers. However, ALC and NORM have a different set of features and limitations. ALC supports a unidirectional delivery model where there is no feedback from the receivers to senders. Reliability is achieved by the joint use of carousel-based transmission techniques associated to FEC encoding to recover from missing (erased) packets.

On the opposite, NORM achieves reliability by means of FEC encoding (as with ALC) and feedback from the receivers. More specifically, NORM leverages Negative Acknowledgement techniques to control the senders' transmission of content. The advantage is that the sender need not transmit any more information than necessary to satisfy the receivers' need for fully reliable transfers. However, while NORM specifies feedback control techniques to allow it to scale to large group sizes, it is not as massively scalable as ALC. Additionally, the NORM feedback control mechanisms add some header content and protocol implementation complexity.

The appropriate choice of a CDP depends upon application needs, deployment constraints, and network connectivity considerations. And

while there are many common security considerations for these two classes of CDP, there are also some unique considerations for each.

## 2.2. RMT Protocol Characteristics

This section focuses on the RMT protocol characteristics that impact the choice of the technological building blocks, and the way they can be applied. Both ALC and NORM have been designed with receiver group size scalability. While ALC targets massively scalable sessions (e.g., millions of receivers), NORM is less ambitious, essentially because of the use of feedback messages.

The ALC and NORM protocols also differ in the communication paths:

- o sender to receivers: ALC and NORM, for bulk data transfer and signaling messages;
- o receivers to sender: NORM only, for feedback messages;
- o receivers to receivers: NORM only for control messages;

Note that the fact ALC is capable of working on top of purely unidirectional networks does not mean that no back-channel is available (Section 2.3).

The NORM and ALC protocols support a variety of content delivery models where transport may be carefully coordinated among the sender and receivers or with looser coordination and interaction. This leads to a number of different use cases for these protocols.

## 2.3. Target Use Case Characteristics

This section focuses on the target use cases and their special characteristics. These details will impact both the choice of the technological building blocks and the way they can be applied. One can distinguish the following use case features:

- o Purely unidirectional transport versus symmetric bidirectional transport versus asymmetric bidirectional transport. Most of the time, the amount of traffic flowing to the source is limited, and one can overlook whether the transport channel is symmetric or not. The nature of the underlying transport channel is of paramount importance, since many security building blocks will require a bidirectional communication;
- o Massively scalable versus moderately scalable session. Here we do not define precisely what the terms "massively scalable" and "moderately scalable" mean.

- o Known set of receivers versus unknown set of receivers: I.e., does the source know at any point of time the set of receivers or not? Of course, knowing the set of receivers is usually not compatible with massively scalable sessions;
- o Dynamic set of receivers versus fixed set of receivers: I.e., does the source know at some point of time the maximum set of receivers or will it evolve dynamically?
- o High rate data flow versus small rate data flow: Some security building blocks are CPU-intensive and are therefore incompatible with high data rate sessions (e.g., solutions that digitally sign all packets sent).
- o Protocol stack available at both ends: A solution that requires some unusual features within the protocol stack will not always be usable. Some target environments (e.g., embedded systems) provide a minimum set of features and extending them (e.g., to add IPsec) is not necessarily realistic;
- o Multicast routing and other layer-3 protocols in use: E.g., SSM routing is often seen as one of the key service to improve the security within multicast sessions, and some security building blocks require specialized versions of layer-3 protocols (e.g., IGMP/MLD with security extensions). In some cases these assumptions might not be realistic.

Depending on the target goal and the associated security building block used, other features might be of importance. For instance TESLA requires a loose time synchronization between the source and the receivers. Several possible techniques are available to provide this, but some of them may be feasible only if the target use case has the appropriate characteristics.

### 3. Some Security Threats

The IP architecture provides common access to notional control and data planes to both end and intermediate systems. For the purposes of discussion here, the "control plane" mechanisms are considered those with message exchanges between end systems (typically computers) and intermediate systems (typically routers) (or among intermediate systems) while the "data plane" encompasses messages exchanged among end systems, usually pertaining to the transfer of application data. The security threats described here are introduced within the taxonomy of control plane and data plane IP mechanisms.

### 3.1. Control-Plane Attacks

In this discussion, "control-plane" in the context of Internet Protocol systems refers to signaling among end systems and intermediate systems to facilitate routing and forwarding of packets. For IP multicast, this notably includes Internet Group Management Protocol (IGMP), Multicast Listener Discovery protocol (MLD), and multicast routing protocol messaging. While control-plane attacks may be considered outside of the scope of the transport protocol specifications discussed here, it is important to understand the potential impact of such attacks with respect to the deployment and operation of these protocols. For example, awareness of possible IP Multicast control-plane manipulation that can lead to unauthorized (or unexpected) monitoring of data plane traffic by malicious users may lead a transport application or protocol implementation to support encryption to ensure data confidentiality and/or privacy. Also, these types of attack also have bearing on assessing the real risks of potentially more complex attacks against the transport mechanisms themselves. In some cases, the solutions to these control-plane risk areas may reduce the impact or possibility of some data-plane attacks that are discussed in this document.

The presence of these types of attack may necessitate that policy-based controls be embedded in routers to limit the distribution (including transmission and reception) of multicast traffic (on a group-wise and/or traffic volume basis) to different parts of the network. Such policy-based controls are beyond the scope of the RMT protocol specifications. However, such network protection mechanisms may reduce the opportunities for or effectiveness of some of the data-plane attacks discussed later. For example, reverse-path checks can significantly limit opportunities for attackers to conduct replay attacks when hosts actually do use IPsec. Also, future IP Multicast control protocols may wish to consider providing security mechanism to prevent unauthorized monitoring or manipulation of messages related to group membership, routing, and activity. The sections below describe some variants of control-plane attacks.

#### 3.1.1. Control Plane Monitoring

While this may not be a direct attack on the transport system, it may be possible for an attacker to gain useful information in advancing attack goals by monitoring IP Multicast control plane traffic including group membership and multicast routing information. Identification of hosts and/or routers participating in specific multicast groups may readily identify systems vulnerable to protocol-specific exploitation. And, with regards to user privacy concerns, such "side information" may be relevant to this emerging aspect of network security as described in Section 4.4.

### 3.1.2. Unauthorized (or Malicious) Group Membership

One of the simplest attacks is that where a malicious host joins an IP multicast group so that potentially unwanted traffic is routed to the host's network interface. This type of attack can turn a legitimate source of IP traffic into a "attacker" without requiring any access privileges to the source host or routers involved. This type of attack can be used for denial-of-service purposes or for the real attacker (the malicious joiner) to gain access to the information content being sent. Similarly, some routing protocols may permit any sender (whether joined to the specific group or not) to transmit messages to a multicast group.

It is possible that malicious hosts could also spoof IGMP/MLD messages, joining groups posing as legitimate hosts (or spoof source traffic from legitimate hosts). This may be done at intermediate locations in the network or by hosts co-resident with the authorized hosts on local area networks. Such spoofing could be done by raw packet generation or with replay of previously-recorded control messages.

For the sake of completeness, it should be noted that multicast routing protocol control messaging may be subject to similar threats if sufficient protocol security mechanisms are not enabled in the routing infrastructure. [RFC4609] describes security threats to the PIM-SM multicast routing infrastructures.

### 3.2. Data-Plane Attacks

This section discusses some types of active attacks that might be conducted "in-band" with respect to the reliable multicast transport protocol operating within the data plane of network data transfer. I.e., the "data-plane" here refers to IP packets containing end-to-end transport content to support the reliable multicast transfer. The passive attack of unauthorized data-plan monitoring is discussed above since such activity might be made possible by the vulnerabilities of the IP Multicast control plane. To cover the two classes of RMT protocols, the active data-plane attacks are categorized as 1) those where the attacker generates messages posing as a data sender, and 2) those where the attacker generates messages posing as a receiver providing feedback to the sender(s) or group. Additionally, a common threat to protocol operation is that of brute-force, rogue packet generation. This is discussed briefly below, but the more subtle attacks that might be conducted are given more attention as those fall within the scope of the RMT transport protocol design. Additionally, special consideration is given to that of the "replay attack" [see Section 3.2.4], as it can be applied across these different categories.



### 3.2.1. Rogue Traffic Generation

If an attacker is able to successfully inject packets into the multicast distribution tree, one obvious denial-of-service attack is for the attacker to generate a large volume of apparently authenticate traffic (and if authentication mechanisms are used, a "replay" attack strategy might be used). The impact of this type of attack can be significant since the potential for routers to relay the traffic to multiple portions of a networks (as compared to a single unicast routing path). However, other than the amplified negative impact to the network, this type of attack is no different than what is possible with rogue unicast packet generation and similar measures used to protect the network from such attacks could be used to contain this type of brute-force attack. Of course, the pragmatic question of whether current implementations of such protection mechanisms support IP Multicast SHOULD be considered.

### 3.2.2. Sender Message Spoofing

Sender message spoofing attacks are applicable to both CDP: ALC (sender-only transmission) and NORM (sender-receiver exchanges). Without an authentication mechanism, an attacker can easily generate sender messages that could disrupt a reliable multicast transfer session. And with FEC-based transport mechanisms, a single packet with an apparently-correct FEC payload identifier[RFC5052] but a corrupted FEC payload could potentially render an entire block of transported data invalid. Thus, a modest injection rate of corrupt traffic could cause severe impairment of data transport. Additionally, such invalid sender packets could convey out-of-bound indices (e.g., bad symbol or block identifiers) that can lead to buffer overflow exploits or similar issues in implementations that insufficiently check for invalid data.

An indirect use of sender message spoofing would be to generate messages that would cause receivers to take inappropriate congestion-control action. In the case of the layered congestion control mechanisms proposed for ALC use, this could lead to the receivers erroneously leaving groups associated with higher bandwidth transport layers and suffering unnecessarily low transport rates. Similarly, receivers may be misled to join inappropriate groups directing unwanted traffic to their part of the network. Attacks with similar effect could be conducted against the TCP-Friendly Multicast Congestion Control (TFMCC) [RFC4654] approach proposed for NORM operation with spoofing of sender messages carrying congestion control state to receivers.

### 3.2.3. Receiver Message Spoofing

These attacks are limited to CDP that use feedback from receivers in the group to influence sender and other receiver operation. In the NORM protocol, this includes negative-acknowledgement (NACK) messages fed back to the sender to achieve reliable transfer, congestion control feedback content, and the optional positive acknowledgement features of the specification. It is also important to note that for ASM operation, NORM receivers pay attention to the messages of other receivers for the purpose of suppression to avoid feedback implosion as group size grows large.

An attacker that can generate false feedback can manipulate the NORM sender to unnecessarily transmit repair information and reduce the goodput of the reliable transfer regardless of the sender's transmit rate. Contrived congestion control feedback could also cause the sender to transmit at an unfairly low rate.

As mentioned, spoofed receiver messaging may not be directed only at senders, but also at receivers participating in the session. For example, an attacker may direct phony receiver feedback messages to selected receivers in the group causing those receivers to suppress feedback that might have otherwise been transmitted. This attack could compromise the ability of those receivers to achieve reliable transfer. Also, suppressed congestion control feedback could cause the sender to transmit at a rate unfair to those attacked receivers if their fair congestion control rate were lower.

### 3.2.4. Replay Attacks

The infamous "replay attack" (injection of a previously transmitted packet to one or more participants) is given special attention here because of the special consequences it can have on RMT protocol operation. Without specific protection mechanisms against replay (e.g., duplicate message detection), it is possible for these attacks to be successful even when security mechanisms such as packet authentication and/or encryption are employed.

#### 3.2.4.1. Replay of Sender Messages

Generally, replay of recent protocol messages from the sender will not harm transport, and could potentially assist it, unless it is of sufficient volume to result in the same type of impact as the "rogue traffic generation" described above. However, it is possible that replay of sufficiently old messages may cause receivers to think they are "out of sync" with the sender and reset state, compromising the transfer. Also, if sender transport data identifiers are reused (object identifiers, FEC payload identifiers, etc), it is possible

that replay of old messages could corrupt data of a current transfer.

#### 3.2.4.2. Replay of Receiver Messages

Replay of receiver messages are problematic for the NORM protocol, because replay of NACK messages could cause the sender to unnecessarily transmit repair information for an FEC coding block. Similarly, the sender transmission rate might be manipulated by replay of congestion control feedback messages from receivers in the group. And the way that NORM senders estimate group round-trip timing (GRTT) could allow a replay attack to manipulate the senders' GRTT estimate to an unnecessarily large value, adding latency to the reliable transport process.

### 4. General Security Goals

The term "security" is extremely vast and encompasses many different meanings. The goal of this section is to clarify what "security" means when considering the CDP defined in the IETF RMT working group. However, the scope can also encompass additional applications, like streaming applications. This section only focuses on the desired general goals. The following sections will then discuss the possible elementary services that will be required to fulfill these general goals, as well as the underlying technological building blocks.

The possible final goals include, in decreasing order of importance:

- o network protection: the goal is to protect the network from attacks, no matter whether these attacks are voluntary (i.e., launched by one or several attackers) or non voluntary (i.e., caused by a misbehaving system, where "system" can designate a building block, a protocol, an application, or a user);
- o protocol protection: the goal is to protect the RMT protocol itself, e.g., to avoid that a misbehaving receiver prevents other receivers to get the content, no matter whether this is done intentionally or not;
- o content protection: the goal is to protect the content itself, for instance to guaranty the integrity of the content, or to make sure that only authorized clients can access the content;
- o and user protection: the goal is often to protect the user privacy.

#### 4.1. Network Protection

Protecting the network is of course of primary importance. An attacker should not be able to damage the whole infrastructure by exploiting some features of the RMT protocol. Unfortunately, recent past has shown that the multicast routing infrastructure is relatively fragile, as well as the applications built on top of it. Since the RMT protocols may use congestion control mechanisms to regulate sender transmission rate, the protocol security features should ensure that the sender may not be manipulated to transmit at incorrect rates (most importantly not at an excessive rate) to any parts of the receiver group. In the case of NORM, the security mechanisms should ensure that the feedback suppression mechanisms are protected to prevent badly-behaving network nodes from purposefully causing feedback implosion. In the case of ALC, where layered congestion control may be used via dynamic group/layer membership, this extends to considerations of excessive manipulation of the multicast router control plane.

#### 4.2. Protocol Protection

Protecting the protocols is also of importance, since the higher the number of clients, the more serious the consequences of an attack. This is all the more true as scalability is often one of the desired goals of CDP. Ideally, receivers should be sufficiently isolated from one another, so that a single misbehaving receiver does not affect others. Similarly, an external attacker should not be able to break the system, i.e., resulting in unreliable operation or delivery of incorrect content.

#### 4.3. Content Protection

The content itself should be protected when meaningful. This level of security is often the concern of the content provider (and its responsibility). For instance, in case of confidential (or non-free) content, the typical solution consists in encrypting the content. It can be done within the upper application, i.e., above the RMT protocol, or within the transport system.

But other requirements may exist, like verifying the integrity of a received object, or authenticating the sender of the received packets. To that goal, one can rely on the use of building blocks integrated within, or above, or beneath the RMT protocol.

One may also consider that offering the packet sender authentication and content integrity services are basic requirements that should fulfill any RMT system that operates within an open network, where any attacker can easily inject spurious traffic in an ongoing NORM or

ALC session. In that case this goal is not the responsibility of the content provider but the responsibility of the administrator who deploys the RMT system itself.

#### 4.4. Privacy

Finally the user should be protected, and more specifically its privacy. In general, there is no privacy issue for data sender: the data sender's address is announced to all prospective receivers prior to their joins. Moreover receivers need to specify the source address(es) as well as the IP multicast address in SSM communication upon their subscription. The situation is different if we consider receivers since their address should not be disclosed publicly.

Data receivers use IGMP or MLD protocols to notify their upstream routers to join or leave IP multicast session. The recent IGMPv3 [RFC3376] and MLDv2 [RFC3810] do not adopt the "report suppression mechanism". Report suppression makes the receiver host withdraw its own report when the host hears a report scheduled to be sent from other host joining the same group. Eliminating the report suppression mechanism does not contribute to minimizing the number of responses, but enables the router to keep track of host membership status on a link. Due to this specification, operators who maintain upstream routers that attach multicast data receiver can recognize data receivers' addresses by tracing IGMP/MLD report messages. Although such traced data may be useful for capacity planning or accounting from operator's perspective, the detail information including receivers' IP addresses should be carefully treated.

As described in Section 3.1.2, unauthorized users may spoof IGMP/MLD query messages and trace receivers' addresses on the same LAN. Currently, IGMP/MLD protocols do not protect this attack. It is desired for these protocols to ignore invalid query messages and provide receiver's privacy by some means.

#### 5. Elementary Security Techniques

The goals defined in Section 4 will be fulfilled by means of underlying security techniques, provided by one or several technological building blocks. This section only focuses on these elementary security techniques. Some general techniques traditionally available are:

Technique	Goal
packet integrity	Enable session participants to verify that a packet has not been inappropriately modified in transit.
packet source authentication	Enable a receiver to verify the source of a packet.
packet group authentication	Enable a receiver to verify that a packet originated or was modified only within the group and has not been modified by nonmembers in transit; Additionally, if attribution of any modifications by the group is required, certain group authentication mechanisms may provide this capability.
packet non-repudiation	Enable any third party to verify the source of a packet such that the source cannot repudiate having sent the packet.
packet anti-replay	Enable a receiver to detect that a packet is the same as a previously-received packet
object integrity	Enable a receiver to verify the integrity of a whole object. Such object integrity verification should be possible for any singular object or any composition of sub-objects which together constitute a larger object structure.
object source authentication	Enable a receiver to verify the source of an object.
object confidentiality	Enable a source to guarantee that only authorized receivers can access the object data.

### General Security Techniques

Some additional techniques are specific to the RMT protocols:

Technique	Goal
congestion control security	Prevent an attacker from modifying the congestion control protocol normal behavior (e.g., by reducing the transmission (NORM) or reception (ALC) rate, or on the opposite increasing this rate up to a point where congestion occurs)
group management	Ensure that only authorized receivers (as defined by a certain group management policy) join the RMT session and possibly inform the source

backward group secrecy	Prevent a new group member to access the information in clear sent to the group before he joined the group
forward group secrecy	Prevent a former group member to access the information in clear sent to the group after he left the group

#### RMT-Specific Security Techniques

These techniques are usually achieved by means of one or several technological building blocks. The target use case where the RMT system will be deployed will greatly impact the choice of the technological building block(s) used to provide these services, as explained in Section 2.3.

## 6. Technological Building Blocks

Here is a list of techniques and building blocks that are likely to fulfill one or several of the goals listed above:

- o IPsec;
- o Group MAC;
- o Digital signatures;
- o TESLA;
- o SSM communication model;

Each of them is now quickly discussed. In particular we identify what service it can offer, its limitations, and its field of application (adequacy with respect to the CDP and the target use case).

### 6.1. IPsec

#### 6.1.1. Benefits

One direct approach using existing standards is to apply IPsec [RFC4301] to achieve the following properties:

- o source authentication and packet integrity (IPsec AH or ESP)
- o confidentiality by means of encryption (IPsec ESP)

### 6.1.2. Requirements

It is expected that the approach to apply IPsec for reliable multicast transport sessions is similar to that described for OSPFv3 security[RFC4552]. The following list proposes the IPsec capabilities needed to support a similar approach to RMT protocol security:

- o Mode - Transport mode IPsec security is required;
- o Selectors - source and destination addresses and ports, protocol.
- o For some uses, preplaced, manual key support may be required to support application deployment and operation. For automated key management for group communication the Group Secure Association Key Management Protocol (GSAKMP) described in [RFC4535] may be used to emplace the keys for IPsec operation.

Note that a periodic rekeying procedure similar to that described in RFC 4552 can also be applied with the additional benefit that the reliable transport aspects of the CDP provide robustness to any message loss that might occur due to ANY timing discrepancies among the participants in the reliable multicast session.

### 6.1.3. Limitations

It should be noted that current IPsec implementations may not provide the capability for anti-replay protection for multicast operation. In the case of the NORM protocol, a sequence number is provided for packet loss measurement to support congestion control operation. This sequence number can also be used within a NORM implementation for detecting duplicate (replayed) messages from sources (senders or receivers) within the transport session group. In this way, protection against replay attack can be achieved in conjunction with the authentication and possibly confidentiality properties provided by an IPsec encapsulation of NORM messages. NORM receivers generate a very low volume of feedback traffic and it is expected that the 16-bit sequence space provided by NORM will be sufficient for replay attack protection. When a NORM session is long-lived, the limits of the sender repair window are expected to provide protection from replayed NACKs as they would typically be outside of the sender's current repair window. It is suggested that IPsec implementations that can provide anti-replay protection for IP Multicast traffic, even when there are multiple senders within a group, be adopted. The GSAKMP document has some discussion in this area.



## 6.2. Group MAC

### 6.2.1. Benefits

The use of Group MAC (Message Authentication Codes) within the CDP Simple Authentication Schemes for the ALC and NORM Protocols [I-D.ietf-rmt-simple-auth-for-alc-norm] is a simple solution to provide a loss tolerant group authentication/integrity service for all the packets exchanged within a session (i.e., the packets generated by the session's sender and the session's receivers). This scheme is easy to deploy since it only requires that all the group members share a common secret key. Because Group MAC heavily relies on fast symmetric cryptographic building blocks, CPU processing remains limited both at the sender and receiver sides, which makes it suitable for high data rate transmissions, and/or lightweight terminals. Finally, the transmission overhead remains limited.

### 6.2.2. Requirements

This scheme only requires that all the group members share a common secret key, possibly associated to a re-keying mechanism (e.g., each time the group membership changes, or on a periodic basis).

### 6.2.3. Limitations

This scheme cannot protect against attacks coming from inside the group, where a group member impersonates the sender and sends forged messages to other receivers. It only provides a group-level authentication/integrity service, unlike the TESLA and Digital Signature schemes. Note that the Group MAC and Digital Signature schemes can be advantageously used together, as explained in Simple Authentication Schemes for the ALC and NORM Protocols [I-D.ietf-rmt-simple-auth-for-alc-norm].

## 6.3. Digital Signatures

### 6.3.1. Benefits

The use of Digital Signatures within the CDP Simple Authentication Schemes for the ALC and NORM Protocols [I-D.ietf-rmt-simple-auth-for-alc-norm] is a simple solution to provide a loss-tolerant authentication/integrity service for all the packets exchanged within a session (i.e., the packets generated by the session's sender and the session's receivers). This scheme is easy to deploy since it only requires that the participants know the packet sender's public key, which can be achieved with either Public Key Infrastructure (PKI) or by preplacement of these keys.

### 6.3.2. Requirements

This scheme is easy to deploy since it requires only that the participants know the packet sender's public key, which can be achieved with either PKI or by preplacement of these keys.

### 6.3.3. Limitations

When RSA [RsaPaper] asymmetric cryptography is used, the digital signatures approach has two major shortcomings:

- o it is limited by high computational costs, especially at the sender, and
- o it is limited by high transmission overheads.

This scheme is well suited to low data rate flows, when transmission overheads are not a major issue. For instance it can be used as a complement to TESLA for the feedback traffic coming from the session's receivers. The use of ECC ("Elliptic Curve Cryptography") significantly relaxes these constraints, especially when seeking for higher security levels. For instance, the following key size provide equivalent security:

Symmetric Key Size	RSA Key Size	ECC Key Size
80 bits	1024 bits	160 bits
112 bits	2048 bits	224 bits

However in some cases, the Intellectual Property Rights (IPR) considerations for ECC may limit its use, so the other techniques are presented here as well. Note that the Group MAC and Digital Signature schemes can be advantageously used together, as explained in Simple Authentication Schemes for the ALC and NORM Protocols [I-D.ietf-rmt-simple-auth-for-alc-norm].

## 6.4. TESLA

### 6.4.1. Benefits

The use of TESLA [RFC5776] within the CDP offers a loss tolerant, lightweight, authentication/integrity service for the packets generated by the session's sender. Depending on the time synchronization and bootstrap methods used, TESLA can be compatible with massively scalable sessions. Because TESLA heavily relies on fast symmetric cryptographic building blocks, CPU processing remains

limited both at the sender and receiver sides, which makes it suitable for high data rate transmissions, and/or lightweight terminals. Finally, the transmission overhead remains limited.

#### 6.4.2. Requirements

The security offered by TESLA relies heavily on time. Therefore the session's sender and each receiver need to be loosely synchronized in a secure way. To that purpose, several methods exist, depending on the use case: direct time synchronization (which requires a bidirectional transport channel), using a secure Network Time Protocol (NTP) [RFC5905] infrastructure (which also requires a bidirectional transport channel), or a Global Positioning System (GPS) device, or a clock with a time-drift that is negligible in front of the TESLA time accuracy requirements.

The various bootstrap parameters must also be communicated to the receivers, using either an in-band or out-of-band mechanism, sometimes requiring bidirectional communications. So, depending on the time synchronization scheme and the bootstrap mechanism method, TESLA can be used with either bidirectional or unidirectional transport channels.

#### 6.4.3. Limitations

One limitation is that TESLA does not protect the packets that are generated by receivers, for instance the feedback packets of NORM. These packets must be protected by other means.

Another limitation is that TESLA requires some buffering capabilities at the receivers in order to enable the delayed authentication feature. This is not considered though as a major issue in the general case (e.g., FEC decoding of objects within an ALC session already requires some buffering capabilities, that often exceed that of TESLA), but it might be one in case of embedded environments.

#### 6.5. Source-Specific Multicast

Source-Specific Multicast (SSM) [RFC3569], [RFC4607] amends the classical Any-Source Multicast (ASM) model by creating logical IP multicast "channels" that are defined by the multicast destination address and the specific source address(es). Thus for a given "channel", only the specific source(s) can inject packets that are distributed to the receivers. This form of multicast has group management benefits since a source can independently control the "channels" it creates.

#### 6.5.1. Requirements

Use of SSM requires that the network intermediate systems explicitly support it. Additionally, hosts operating systems are required to support the IGMPv3/MLDv2 extensions for SSM, and the CDP implementations need to support the IGMPv3/MLDv2 API, including management of the <srcAddr; dstMcastAddr> "channel" identifiers.

#### 6.5.2. Limitations

CDP such as NORM that use signaling from receivers to multicast senders will need to use unicast addressing for feedback messages. In the case of NORM, its timer-based feedback suppression requires support of the sender NORM\_CMD(REPAIR\_ADV) message to control receiver feedback. In some topologies, use of unicast feedback may require some additional latency (increased backoff factor) for safe operation. The security of the unicast feedback from the receivers to sender will need to be addressed separately since the IP multicast model, including SSM, does not provide the sender knowledge of authorized group members.

#### 6.5.3. Source-Based and Receiver-Based Attacks

The security threats are categorized into "source-based" and "receiver-based" attacks [RFC4609]. In short, the former is a DoS attack against the multicast networks, in which data is sent to numerous and random group addresses, and the latter is a DoS attack against multicast routers, in which innumerable IGMP/MLD joins are sent from a client.

Regarding source-based attack, there are some security benefits in SSM. Since data-plane traffic for an SSM "channel" is limited to that of a single, specific source address, it is possible that network intermediate systems may impose mechanism that prevent injection of traffic to the group from inappropriate (perhaps malicious) nodes. This can reduce the risk for denial-of-service and some of the other attacks described in this document. While SSM alone is not a complete security solution, it can simplify secure RMT operation.

On the contrary, SSM is not robust against receiver-based attack. An SSM capable router constructs a Shortest-Path Tree (SPT) with no shared tree coordination. Therefore, even if a host triggers invalid or unavailable channel subscriptions, the upstream router starts establishing all SPTs with no intellectual decision. What is worse is that these multicast routers cannot recognize the original router that is attacked and cannot stop the attack itself.

## 6.6. Summary

The following table summarizes the pros/cons of each authentication/integrity scheme used at application/transport level (where "-" means bad, "0" means neutral, and "+" means good):

	RSA Digital Signature	ECC Digital Signature	Group MAC	TESLA
True authentication and integrity	Yes	Yes	No (group security)	Yes
Immediate authentication	Yes	Yes	Yes	No
Processing load	-	0	+	+
Transmission overhead	-	0	+	+
Complexity	+	+	+	-

## 7. Security Infrastructure

Deploying the elementary technological building blocks often requires that a security infrastructure exists. Such security infrastructure can provide:

- o Public Key Infrastructure (PKI) for trusted third party vetting of, and vouching for, user identities. PKI also allows the binding of public keys to users, usually by means of certificates.
- o Group Key Management with rekeying schemes that are either periodic or triggered by some higher level event. It is required in particular when the group is dynamic and forward/backward secrecy are important. This is also required to improve the scalability of the CDP (since key management is done automatically, using a key server topology), or the security provided by the CDP (since the underlying cryptographic keys will be changed frequently)

It is expected that some CDP deployments may use existing client-server security infrastructure models so that receivers may acquire any necessary security material and be authenticated or validated as needed for group participation. Then, the reliable delivery of session data content will be provided via the applicable RMT protocols. Note that in this case the security infrastructure itself may limit the scalability of the group size or other aspects of

reliable multicast transfer. The IETF Multicast Security (MSEC) Working Group has developed some protocols that can be applied to achieve more scalable and effective group communication security infrastructure[RFC4046]. It is encouraged that these mechanisms be considered in the development of security for CDP.

## 8. New Threats Introduced by the Security Scheme Itself

Introducing a security scheme, as a side effect, can sometimes introduce new security threats. For instance, signing all packets with asymmetric cryptographic schemes (to provide a source authentication/content integrity/anti-replay service) opens the door to DoS attacks. Indeed, verifying asymmetric-based cryptographic signatures is a CPU intensive task. Therefore an attacker can easily overload a receiver (or a sender in case of NORM) by injecting a significant number of faked packets.

## 9. Consequences for the RMT and MSEC Working Group

To meet the goals outlined in this document, it is expected that the RMT and MSEC Working Groups may need to develop some supporting protocol security mechanisms. It is also possible to cooperate with the Multicast Backbone (MBONE) Deployment (MBONED) Working Group for defining operational considerations.

### 9.1. RMT Transport Message Security Encapsulation Header

An alternative approach to using IPsec to provide the necessary properties to protect RMT protocol operation from the application attacks described earlier, is to extend the RMT protocol message set to include a message encapsulation option. This encapsulation header could be used to provide authentication, confidentiality, and anti-replay protection as needed. Since this would be independent of the IP layer, the header might need to provide a source identifier to be used as a "selector" for recalling security state (including authentication certificate(s), sequence state, etc) for a given message. In the case of the NORM protocol, a "NormNodeId" field exists that could be used for this purpose. In the case of ALC, the security encapsulation mechanism would need to add this function. The security encapsulation mechanism, although resident "above" the IP layer, could use GSAKMP [RFC4535] or a similar approach for automated key management.

## 10. IANA Considerations

This document has no actions for IANA.

## 11. Security Considerations

This document is a general discussion of security for the RMT protocol family. But specific security considerations are not applicable as this document does not introduce any new techniques.

## 12. Acknowledgments

The authors would like acknowledge Magnus Westerlund for stimulating the working group activity in this area. Additionally George Gross and Ran Atkinson contributed many ideas to the discussion here.

## 13. References

### 13.1. Normative References

- [I-D.ietf-rmt-flute-revised]  
Paila, T., Walsh, R., Luby, M., Roca, V., and R. Lehtonen,  
"FLUTE - File Delivery over Unidirectional Transport",  
draft-ietf-rmt-flute-revised-12 (work in progress),  
February 2011.
- [I-D.ietf-rmt-simple-auth-for-alc-norm]  
Roca, V., "Simple Authentication Schemes for the ALC and  
NORM Protocols",  
draft-ietf-rmt-simple-auth-for-alc-norm-03 (work in  
progress), July 2010.
- [RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5,  
RFC 1112, August 1989.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate  
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A.  
Thyagarajan, "Internet Group Management Protocol, Version  
3", RFC 3376, October 2002.
- [RFC3569] Bhattacharyya, S., "An Overview of Source-Specific  
Multicast (SSM)", RFC 3569, July 2003.

- [RFC3810] Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004.
- [RFC4046] Baugher, M., Canetti, R., Dondeti, L., and F. Lindholm, "Multicast Security (MSEC) Group Key Management Architecture", RFC 4046, April 2005.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.
- [RFC4535] Harney, H., Meth, U., Colegrove, A., and G. Gross, "GSAKMP: Group Secure Association Key Management Protocol", RFC 4535, June 2006.
- [RFC4552] Gupta, M. and N. Melam, "Authentication/Confidentiality for OSPFv3", RFC 4552, June 2006.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.
- [RFC4609] Savola, P., Lehtonen, R., and D. Meyer, "Protocol Independent Multicast - Sparse Mode (PIM-SM) Multicast Routing Security Issues and Enhancements", RFC 4609, October 2006.
- [RFC4654] Widmer, J. and M. Handley, "TCP-Friendly Multicast Congestion Control (TFMCC): Protocol Specification", RFC 4654, August 2006.
- [RFC5052] Watson, M., Luby, M., and L. Vicisano, "Forward Error Correction (FEC) Building Block", RFC 5052, August 2007.
- [RFC5740] Adamson, B., Bormann, C., Handley, M., and J. Macker, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol", RFC 5740, November 2009.
- [RFC5775] Luby, M., Watson, M., and L. Vicisano, "Asynchronous Layered Coding (ALC) Protocol Instantiation", RFC 5775, April 2010.
- [RFC5776] Roca, V., Francillon, A., and S. Faurite, "Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Asynchronous Layered Coding (ALC) and NACK-Oriented Reliable Multicast (NORM) Protocols", RFC 5776, April 2010.



## 13.2. Informative References

[Neumann05]

Neumann, C., Roca, V., and R. Walsh, "Large Scale Content Distribution Protocols", ACM Computer Communications Review (CCR) Vol. 35 No. 5, October 2005.

[RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

[RsaPaper]

Rivest, R., Shamir, A., and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM 21, pp. 120-126, 1978.

## Authors' Addresses

Brian Adamson  
Naval Research Laboratory  
Washington, DC 20375  
USA

Email: [adamson@itd.nrl.navy.mil](mailto:adamson@itd.nrl.navy.mil)  
URI: <http://cs.itd.nrl.navy.mil>

Vincent Roca  
INRIA  
Montbonnot 38334  
France

Email: [vincent.roca@inria.fr](mailto:vincent.roca@inria.fr)  
URI: <http://planete.inrialpes.fr/~roca/>

Hitoshi Asaeda  
Keio University  
5322 Endo  
Fujisawa, Kanagawa 252-8520  
Japan

Email: [asaeda@wide.ad.jp](mailto:asaeda@wide.ad.jp)  
URI: <http://www.sfc.wide.ad.jp/~asaeda/>



Reliable Multicast Transport  
Internet Draft

E. Stauffer  
Broadcom  
B. Shen  
Broadcom  
S. Chakraborty  
Broadcom  
D Tujkovic  
Broadcom  
Jing Huang  
Broadcom  
K. Rath  
Broadcom  
May 13, 2012

Intended status: Standards Track  
Expires: November 2012

Supercharged Codes  
draft-stauffer-rmt-bb-fec-supercharged-00.txt

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

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

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

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

This Internet-Draft will expire on November 13, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Abstract

This document describes a fully-specified FEC scheme for the Supercharged forward error correction code. Supercharged codes are designed for use on the erasure channel. Coding for the erasure channel commonly arises for data transmission over the internet, where lower layers either successfully deliver packets or fail to deliver them. Coding is required to insure that data is not lost, even if packets are lost at the lower layers. Error free reception is important for multimedia applications, such as streaming, where it may not be possible to correct an error in time by any other means. Coding insures that lost packets can be recovered.

## Table of Contents

1. Introduction.....	3
2. Supercharged Code.....	3
2.1.1. Definitions.....	3
2.2. Overview.....	4
2.3. Matrix Representation.....	5
2.4. Systematic Encoding.....	6
2.5. Erasure Channel.....	6
2.6. Decoding.....	6
2.7. Matrix P Construction.....	7
2.7.1. Function Prototypes.....	7
2.7.2. Parallel Filter Code T Construction.....	8
2.7.3. Repetition Code R Construction.....	10
2.7.4. Block Code B_1 Construction.....	11
2.7.5. Block Code B_2 and B_3 Construction.....	11
2.7.6. SC_Parameters.....	13
2.7.7. K Table.....	13
2.7.8. Random Number Generator.....	18
2.7.9. Random Permutation.....	22
2.7.10. RS Generator.....	23
2.7.11. Systematic RS.....	24
2.7.12. SC_Filter_Data.....	24

2.7.13. GF(256) Operations.....	25
3. FEC Packets.....	25
3.1. Segmentation.....	25
3.1.1. Transmit Blocks.....	25
3.1.2. Working Blocks.....	26
3.1.3. Padding.....	26
4. Parameter Selection.....	26
5. Protocol IEs.....	27
5.1. FEC Payload IEs.....	27
5.2. Common.....	27
5.3. Scheme Specific.....	28
6. Conventions used in this document.....	29
7. Security Considerations.....	29
8. IANA Considerations.....	29
9. References.....	29
9.1. Normative References.....	29
9.2. Informative References.....	30
10. Acknowledgments.....	30

## 1. Introduction

This document describes a fully-specified FEC scheme for the Supercharged forward error correction code. The Supercharged code is designed for the erasure channel with performance very close to the ideal Maximum Distance Separable(MDS) code and with very low complexity. Section 2 describes the architecture of the code and defines the generator matrices used by the code. Section 3 describes how to construct FEC packets. Section 4 discusses code parameter selection for a particular usage context. Section 5 defines the protocol information elements. Section 6 considers security. Section 7 considers IANA.

## 2. Supercharged Code

### 2.1.1. Definitions

ceil(a): rounds a to the nearest integer towards infinity

floor(a): rounds a to the nearest integer towards minus infinity

min(a,b): returns the minimum of a and b

max(a,b): returns the maximum of a and b

a % b: is a modulo b

a + b: is a plus b

`a * b`: `a` multiplied by `b`.

`a ^ b`: the bitwise XOR of `a` and `b`

`a ^^ b`: raises `a` to the `b` power

`I_a`: the `a` x `a` identity matrix

`zeros(a,b)`: the `a` x `b` zero matrix

## 2.2. Overview

Figure 1 shows a general block diagram of the supercharged code. It consists of a network of codes including block codes, repetition codes, and parallel filter codes. Block code 1 consists of a Vandermonde matrix in GF(256), a non-systematic Reed Solomon code. Block code 2 and 3 consist of binary block codes.

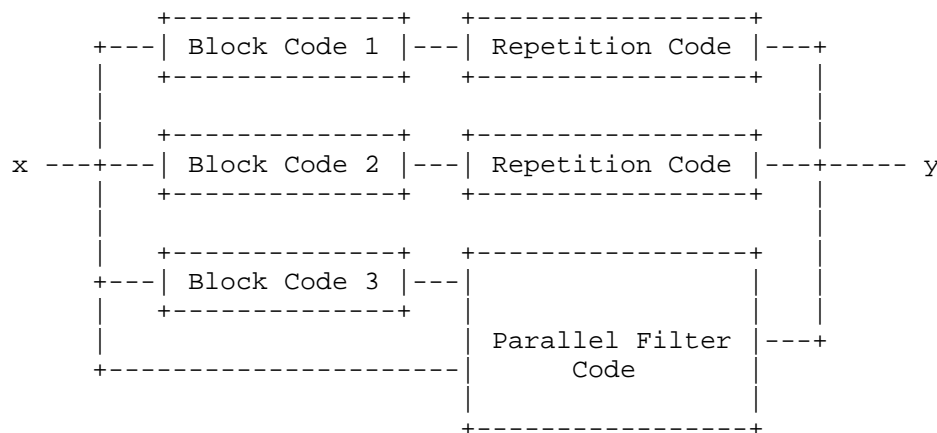


Figure 1 Block Diagram of the SC Code

The parallel filter code of Figure 1 is detailed in Figure 2. It consists of interleavers, tailbiting FIR filters, and a multiplexer to select the output of the filters.

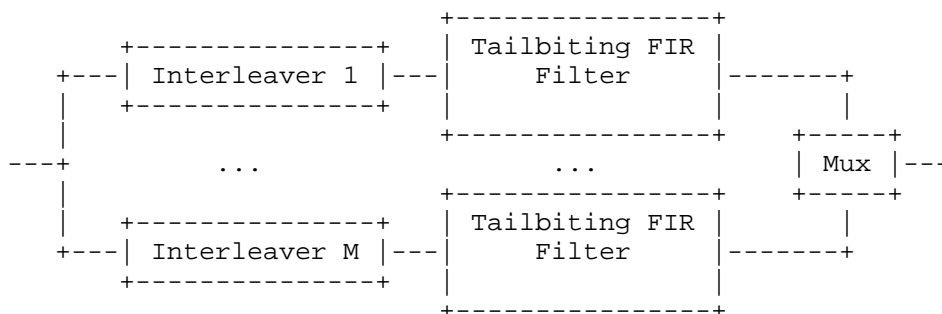


Figure 2 An example parallel filter code showing individual data interleavers and tailbiting FIR filters as coding components.

An example of one of the tailbiting FIR filters is illustrated in Figure 3, where the state of the filter is initialized with the final state to make it tailbiting.

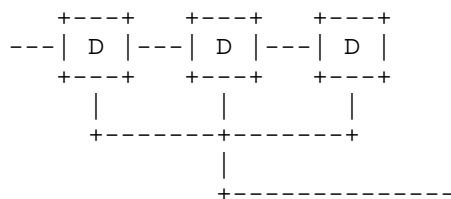


Figure 3 An example 3 tap FIR filter that can be used for the tailbiting FIR filter coding component. An XOR operation is applied at the output of the delay elements to produce the final output.

Optionally, if the number of transmit symbols  $N$  is signaled to be limited such that  $N \leq 256$ , then the code can achieve ideal performance by utilizing a Reed Solomon code.

### 2.3. Matrix Representation

Since supercharged codes are linear, an output codeword can be expressed as a matrix multiplied by an input vector. Given  $K \times 1$  encoding state vector  $x$ , consisting of binary transmit symbols, the output  $N \times 1$  codeword,  $y$ , can be written as

$$y = (T*[I_K; B_3] + R_1*B_1 + R_2*B_2)*x \quad (1)$$

where  $T$  is the  $N \times (K + \text{Num\_B\_3})$  generator matrix for the FIR structure,  $B_1$  is the  $\text{Num\_V\_RS} \times K$  generator matrix for the first block code,  $B_2$  is the  $\text{Num\_B\_2} \times K$  generator for the second block code,  $B_3$  is the  $\text{Num\_B\_3} \times K$  generator matrix of the third block code, and  $R_1$  is a  $N \times \text{Num\_V\_RS}$  stack and  $R_2$  is a  $N \times \text{Num\_B\_2}$  stack of identity matrices which facilitates repetition. For example, matrix  $R_1$  would consist of  $\text{floor}(N/\text{Num\_V\_RS})$  copies of the identity matrix stacked vertically, with a fractional identity matrix below consisting of  $N \bmod \text{Num\_V\_RS}$  rows. The "+" operator indicates the bitwise XOR operation. For convenience, denote the generator matrix  $P = (T * [I_K; B_3] + R_1 * B_1 + R_2 * B_2)$ , such that  $y = Px$ .

#### 2.4. Systematic Encoding

Supercharged codes are not inherently systematic codes. Non-systematic codes are commonly transformed into an effective systematic code by pre-processing the input data before using it as the input to the encoder,  $y = Px$ . The encoder input is calculated by decoding the desired input data and running the decoder to determine the encoder input vector  $x$ . Let matrix  $P_{\text{enc}}$  be the  $K \times K$  generating matrix corresponding to the first  $K$  elements of  $y$ , the encoder input  $x$  can be computed using the following

$$x = P_{\text{enc}}^{(-1)} * d.$$

Now,  $x$  can be used to encode using equation (1) to generate  $y$ . The first  $K$  elements of vector  $y$  will be equal to  $d$ .

#### 2.5. Erasure Channel

After encoding, the  $N$  transmit symbols of codeword vector  $y$  are transmitted on the channel. Some of these transmit symbols are erased by the channel. Suppose that the  $N \times r$  matrix  $E$  represents the erasure pattern of the channel in that it selects out the  $r$  received transmit symbols  $y_r$  from the transmitted symbols  $y$ . If the  $i$ th received symbol is the  $j$ th transmit symbol, then  $E(i, j) = 1$ . This results in

$$y_r = E * y.$$

At the decoder, the effective generator matrix at the receiver is  $P_r = E * P$ .

#### 2.6. Decoding

Decoding is the process of determining  $x$  given  $y_r$  and  $P_r$ . Decoding can be implemented in several different ways, but each are equivalent



to solving the least squares problem  $x = (P_r^{TT}P_r)^{-1} * P_r^{TT} * y_r$ . Modern sparse matrix factorization techniques can take advantage of the sparse structure imposed by the parallel filter structure if (1) is rewritten in the following equivalent form

$$z = Gw, \quad (2)$$

with augmented generator matrix  $G$  defined as

$$G = [ [B_3; B_2; B_1] \ I_L; \ T \ R_2 \ R_1 ]$$

and where the augmented output vector  $z=[zeros(L,1); y]$ , the augmented input vector  $w=[x; B_3*x; B_2*x; B_1*x]$ , and where  $L=Num\_V\_RS+Num\_B\_2+Num\_B\_3$ . The bottom  $L$  elements of vector  $w$  contain the outputs, before repetition, of the block codes. These  $L$  values are appended to vector  $x$  to form the augmented input vector  $w$ . The first  $L$  rows of  $G$  implement the block code and XOR the block code output with itself to generate the  $L$  zeros at the top of the  $z$  vector. The subsequent  $N$  rows of  $G$  implement the FIR structure and XOR the output with the output of the block codes.

This problem can be efficiently solved using direct sparse matrix factorization techniques described in [3-8]. It is RECOMMENDED that the Dulmage-Mendelsohn based solver in chapter 8 of [5] be used with addition, multiplication, and division updated to support a finite field. This algorithm utilizes pivoting based on node degrees in the equivalent graph to minimize fill-in. The solution is completed by performing forward and backward substitutions. Iterative solvers are also possible.

Once the encoder state vector  $x$ , or equivalently the augmented encoder state vector  $w$ , has been determined, the task remains to determine the data vector  $d$ . For any elements of  $d$  that are missing, then can be recovered by using appropriate rows of (1) or (2).

## 2.7. Matrix P Construction

### 2.7.1. Function Prototypes

The following functions are utilized to construction the Supercharged code.

```
[K_eff, Num_V_RS, Num_B_2, Num_B_3] = SC_Parameters(K, N)
```

```
K_eff=SC_K_table(K)
```

```
b=RNG(a)
a=RNG_2(a,b)
[permutation,the_seed]= Generate_Permutation(a,b)
G_V_RS = RS_gen(K,N)
[filter_data, filter_N]=SC_filter_data(z)
b=GF_exp(a)
C=GF_Multiply(A,B)
```

#### 2.7.2. Parallel Filter Code T Construction

The parallel filter code matrix T can be generated using the following pseudo code. The code generates multiple random interleavers and selects which output of which interleaver depending on the SID, where the SID is defined in section 3. Note that at the receiver, only filter outputs corresponding to the received SID's are required. The following code generates filter outputs for SIDs 0 to N-1. Determination of the filter output is a function of the SID only, not any other filter output, making it simple to generate only the filter outputs needed at encoding or decoding. The Generate\_Permutation function is defined in section 2.7.9. , the SC\_filter\_data function is defined in section 2.7.12. , and the RNG function is defined in section 2.7.8.

```
seed1 = 758492
seed2 = ( ((K_eff*874) % (2^32)) ^ (seed1) )
seed3 = 23091
base_permutation = Generate_Permutation(K_eff+Num_B_3,seed2)
filter_data = SC_filter_data(K_eff+NUM_B_3)

T = zeros(N,K_eff+NUM_B_3)
for SID=0:N-1
```

```
%Determine which filter to select
rn1 = RNG(15*loop+2*seed3)
index = 0
while(rn1>(filter_data[index+1]))
    index = index+1
end

%Determine which interleaver to select
rn2 = RNG(2*K_eff+3*SID)
interleaver_number = ( (rn2) % (K_eff+NUM_B_3) )

%Determine which part of the interleaver to select
rn3 = RNG(98573+2*SID+rn3)
interleaver_part = ((rn3) % (K_eff+NUM_B_3))

for tap_loop=0:tdeg-1
    filter_tap = (tap_loop+interleaver_part) %
(K_eff+NUM_B_3)
    tap_location = (base_permutation[filter_tap] +
base_permutation[interleaver_number]) % (K_eff+NUM_B_3)
    T[Num_V_RS+Num_Rep+SID,tap_location] = 1
end
end
```

### 2.7.3. Repetition Code R Construction

The repetition code matrix  $R_1$  and  $R_2$  can be constructed via the following pseudo code. Note that at the receiver, only filter outputs corresponding to the received SID's are required. The following code generates filter outputs for SIDs 0 to N-1 for  $R_1$ .

```
R_1 = zeros(N,Num_V_RS)

for SID = 0:N-1

    for k = 0:Num_V_RS-1

        if( ((SID-k) % (Num_V_RS)) == 0 )

            R_1[SID,k] = 1

        end

    end

end
```

The following code generates filter outputs for SIDs 0 to N-1 for  $R_2$ .

```
R_2 = zeros(N, Num_B_2)

for SID = 0:N-1

    for k = 0: Num_B_2-1

        if( ((SID-k) % (Num_B_2)) == 0 )

            R_2[SID,k] = 1

        end

    end

end
```

#### 2.7.4. Block Code B<sub>1</sub> Construction

The Vandermonde matrix of block code B<sub>1</sub> can be constructed via the following pseudo code. The GF\_exp function is defined in section 2.7.13.

```
B_1 = zeros(Num_V_RS,K_eff)
for i = 0:Num_V_RS-1
    for k = 0:K_eff-1
        B_1[i+1,k+1] = GF_exp( ((i+1)*k) % (2^^8-1) )
    end
end
```

#### 2.7.5. Block Code B<sub>2</sub> and B<sub>3</sub> Construction

The block code B<sub>2</sub> and B<sub>3</sub> can be constructed jointly via the following pseudo code, where B<sub>23</sub>=[B<sub>3</sub>; B<sub>2</sub>].

```
B_23 = zeros(Num_B_2 + Num_B_3,K_eff)
for i = 0:K_eff-1
    for k = 0: Num_B_2 + Num_B_3 - 1
        if( ( (k-i) % (Num_B_2 + Num_B_3) ) == 0)
            B_23[k,i] = 1
        end
    end
end
```

```
m=1

for i = 0:K_eff-1

    for k = 0: Num_B_2 + Num_B_3 - 1

        if( ( (k-i-2*floor(m/( Num_B_2 + Num_B_3))) % (Num_B_2 +
Num_B_3) ) == 0)

            B_23[k,i] = 1

        end

        m = m+1

    end

end

m=2

for i = 0:K_eff-1

    for k = 0: Num_B_2 + Num_B_3 - 1

        if( ( (k-i-3*floor(m/( Num_B_2 + Num_B_3))) % (Num_B_2 +
Num_B_3) ) == 0)

            B_23[k,i] = 1

        end

        m = m+1

    end

end

end
```

### 2.7.6. SC\_Parameters

The following pseudo code determines a set of parameters needed for matrix construction. The SC\_K\_table is defined in section 2.7.7.

```
function [K_eff, Num_V_RS, Num_B_2, Num_B_3] = SC_Parameters(K, N)

    K_eff = SC_K_table(K)

    Num_V_RS = 16 + floor(K_eff/10000)

    Num_B = floor(K_eff^(0.62)) + 3

    if( K_eff >= 17376 )

        Num_B = ceil( K_eff*0.0152 + 163 )

    end

    Num_B_3 = ceil(0.75*( Num_B ))

    Num_B_2 = Num_B - Num_B_3
```

### 2.7.7. K Table

The function `K_eff=SC_K_table(K)` is implemented based on the following table, by returning the smallest `K_eff` such that `K_eff>=K`.

```
10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,
33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,
56,57,58,59,60,61,62,63,64,65,66,67,69,70,71,72,73,74,75,76,77,78,79,
80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,101,1
02,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,11
9,120,121,122,123,124,125,126,127,128,129,130,131,133,134,135,136,137
,138,139,140,141,142,143,144,145,146,147,148,149,150,151,152,153,154,
155,156,157,158,159,160,161,162,163,164,165,166,167,168,169,170,171,1
72,173,174,175,176,177,178,179,180,181,182,183,184,185,186,187,188,18
9,190,191,192,193,194,195,196,197,198,199,200,201,202,203,204,205,206
,207,208,209,210,211,212,213,214,215,216,217,218,219,220,221,222,223,
```

224,225,226,227,228,229,230,231,232,233,234,235,236,237,238,239,240,241,242,243,244,245,246,247,248,249,250,251,252,253,254,255,256,257,258,259,260,261,262,263,264,265,266,267,268,269,270,271,272,273,274,275,276,277,278,279,280,281,282,283,284,285,286,287,288,289,290,291,292,293,294,295,296,297,298,299,300,302,303,304,305,306,307,308,309,310,311,312,314,315,316,320,321,324,328,329,335,337,338,340,341,344,347,349,352,355,357,358,360,362,364,366,368,372,377,380,381,382,384,385,388,389,393,394,395,397,399,405,408,409,410,411,416,418,424,426,428,431,432,434,438,443,447,448,451,452,453,457,460,465,466,467,469,473,476,477,478,482,483,484,485,486,490,491,492,493,494,496,497,498,500,501,502,503,504,506,507,508,509,510,511,512,513,514,515,516,517,518,519,520,521,522,524,526,527,528,529,530,532,533,534,535,536,537,539,541,542,543,545,546,549,551,552,553,554,555,557,558,559,561,562,563,564,566,569,571,572,573,574,576,577,578,579,580,582,583,585,586,587,588,589,590,592,593,594,597,598,599,600,602,603,606,607,608,609,610,612,614,615,616,617,619,620,622,625,626,627,628,629,630,631,633,635,636,637,638,640,643,645,648,650,652,653,654,655,656,659,660,661,662,664,666,667,668,669,672,673,674,675,677,687,688,691,692,693,694,695,696,698,699,700,701,703,710,711,712,715,716,717,718,726,727,730,731,734,736,737,741,744,747,748,751,752,753,757,759,760,762,764,766,769,771,772,773,774,775,777,778,779,786,788,790,792,793,794,795,797,798,799,800,801,802,804,805,810,811,812,813,815,820,821,822,823,825,827,829,830,831,834,835,837,838,839,840,843,844,845,846,848,849,851,852,853,854,857,858,860,863,864,866,868,869,870,875,877,879,883,886,887,890,891,894,897,898,899,900,902,903,904,905,906,907,909,912,913,914,917,922,926,927,928,931,934,938,940,942,944,945,948,950,953,954,960,961,963,967,968,970,971,972,974,977,979,980,981,985,987,989,990,995,996,1000,1002,1003,1005,1006,1007,1009,1010,1015,1020,1021,1022,1024,1025,1027,1032,1033,1034,1035,1037,1041,1042,1043,1046,1048,1050,1051,1054,1056,1057,1059,1060,1062,1065,1069,1070,1071,1074,1076,1078,1079,1082,1083,1085,1086,1087,1088,1089,1095,1098,1099,1106,1110,1111,1118,1120,1123,1124,1125,1131,1132,1134,1136,1139,1140,1142,1144,1150,1152,1157,1161,1162,1165,1169,1173,1175,1176,1179,1181,1182,1183,1194,1200,1201,1204,1205,1206,1208,1209,1212,1213,1214,1218,1219,1220,1222,1225,1227,1228,1229,1232,1236,1238,1240,1242,1243,1245,1248,1250,1252,1253,1255,1258,1261,1269,1273,1278,1279,1280,1283,1284,1292,1293,1302,1303,1306,1310,1311,1315,1318,1319,1321,1325,1330,1331,1342,1343,1347,1348,1352,1357,1359,1361,1365,1374,1380,1382,1384,1388,1389,1390,1391,1392,1395,1397,1403,1404,1407,1413,1417,1418,1420,1425,1429,1431,1435,1436,1437,1447,1450,1461,1462,1464,1473,1474,1475,1477,1485,1490,1494,1496,1497,1502,1503,1507,1513,1514,1516,1521,1522,1526,1530,1534,1539,1541,1549,1552,1554,1555,1561,1564,1569,1572,1579,1585,1586,1590,1591,1593,1595,1596,1597,1598,1600,1604,1608,1610,1611,1612,1616,1617,1624,1631,1633,1636,1641,1646,1649,1650,1658,1660,1665,1667,1671,1673,1679,1683,1689,1692,1696,1698,1703,1705,1707,1708,1713,1716,1722,1728,1733,1734,1739,1740,1742,1744,1745,1756,1759,1760,1764,1768,1771,1776,1777,1780,1782,1787,1800,1807,1814,1824,1826,1827,1842,1844,1854,1857,1863,1867,1873



,1874,1878,1881,1883,1887,1889,1890,1891,1892,1894,1896,1903,1905,1906,1910,1919,1924,1926,1931,1933,1943,1944,1948,1952,1954,1967,1971,1973,1976,1979,1985,1986,1987,1989,1992,1994,1995,1998,2000,2005,2006,2018,2019,2030,2040,2043,2048,2054,2055,2057,2061,2070,2071,2074,2077,2082,2084,2087,2089,2093,2096,2098,2103,2104,2107,2111,2120,2122,2125,2128,2138,2150,2152,2155,2160,2175,2177,2182,2189,2195,2200,2201,2203,2217,2219,2225,2226,2231,2234,2235,2236,2237,2245,2247,2274,2276,2278,2280,2282,2283,2286,2292,2303,2304,2306,2310,2315,2316,2319,2320,2321,2330,2333,2336,2339,2343,2344,2345,2351,2367,2368,2371,2374,2382,2389,2392,2395,2396,2400,2402,2407,2410,2412,2416,2421,2422,2434,2442,2446,2447,2462,2473,2477,2478,2481,2486,2490,2492,2495,2502,2505,2507,2509,2512,2513,2522,2525,2527,2528,2536,2543,2549,2556,2559,2561,2563,2565,2583,2587,2590,2592,2596,2598,2601,2603,2604,2606,2617,2622,2625,2626,2636,2638,2640,2643,2654,2660,2668,2673,2677,2679,2688,2695,2699,2701,2713,2714,2723,2737,2741,2747,2753,2762,2764,2769,2772,2775,2776,2785,2796,2802,2805,2808,2826,2828,2830,2831,2834,2836,2853,2875,2877,2878,2884,2906,2938,2945,2948,2950,2961,2964,2966,2968,2979,2980,2985,2989,2998,3008,3011,3015,3018,3022,3027,3048,3049,3051,3053,3056,3062,3071,3075,3080,3093,3094,3095,3097,3101,3107,3109,3119,3122,3128,3149,3150,3151,3158,3166,3167,3173,3178,3180,3181,3182,3186,3190,3195,3200,3201,3203,3204,3205,3208,3216,3217,3223,3224,3232,3236,3240,3248,3251,3253,3269,3276,3278,3279,3286,3292,3299,3306,3309,3336,3340,3342,3344,3351,3352,3356,3357,3371,3375,3380,3387,3396,3404,3407,3410,3423,3430,3445,3451,3463,3466,3471,3478,3479,3502,3513,3520,3528,3531,3534,3539,3540,3546,3551,3565,3577,3579,3603,3606,3608,3612,3614,3616,3620,3647,3650,3653,3658,3664,3677,3682,3686,3694,3697,3705,3707,3724,3728,3744,3749,3751,3754,3761,3765,3776,3778,3781,3792,3797,3799,3801,3834,3840,3841,3848,3861,3863,3883,3901,3903,3919,3924,3941,3943,3960,3965,3970,3971,3989,3992,4007,4013,4015,4037,4039,4045,4050,4055,4069,4072,4073,4091,4096,4106,4112,4124,4129,4133,4140,4146,4156,4165,4188,4207,4209,4210,4215,4221,4236,4237,4247,4252,4253,4257,4261,4266,4270,4318,4330,4341,4346,4359,4363,4365,4366,4388,4415,4418,4436,4438,4453,4468,4474,4477,4503,4512,4513,4519,4522,4538,4548,4567,4575,4576,4577,4583,4590,4621,4639,4651,4659,4681,4693,4698,4700,4702,4729,4731,4739,4741,4742,4748,4749,4758,4764,4765,4771,4772,4780,4785,4803,4804,4838,4840,4843,4868,4871,4878,4885,4898,4901,4918,4924,4933,4939,4954,4959,4979,4982,4988,4991,4999,5000,5008,5021,5023,5030,5039,5060,5062,5063,5096,5116,5137,5143,5145,5162,5163,5167,5172,5186,5218,5225,5238,5240,5252,5260,5279,5285,5295,5301,5310,5314,5317,5331,5332,5334,5348,5353,5354,5390,5391,5392,5405,5407,5432,5449,5451,5453,5460,5464,5466,5471,5473,5477,5492,5506,5508,5537,5540,5543,5554,5561,5566,5570,5576,5579,5587,5616,5637,5672,5674,5676,5684,5694,5716,5732,5774,5792,5798,5800,5808,5823,5838,5844,5863,5896,5897,5899,5900,5916,5921,5930,5960,5975,6039,6055,6057,6059,6067,6068,6078,6092,6099,6102,6107,6136,6151,6169,6189,6191,6218,6233,6249,6271,6274,6296,6318,6352,6363,6376,6407,6430,6435,6441,6463,6486,6491,6502,6512,6518,6520,6534,6542,6549,6553,6589,6590,6593,6599,6614,6625,6634,6643,6655,66

70,6680,6684,6691,6692,6701,6708,6711,6724,6730,6732,6752,6799,6803,6809,6812,6834,6849,6855,6877,6878,6879,6899,6907,6919,6936,6945,6946,6954,6955,6956,6958,6981,7000,7011,7030,7032,7033,7108,7111,7127,7164,7171,7175,7179,7181,7185,7225,7226,7281,7288,7295,7307,7325,7359,7360,7390,7392,7411,7476,7520,7535,7548,7552,7558,7567,7589,7596,7616,7645,7675,7679,7714,7726,7747,7770,7780,7785,7805,7818,7855,7870,7883,7923,7935,7936,7953,7974,7999,8028,8030,8069,8074,8093,8104,8111,8122,8150,8154,8172,8173,8189,8192,8193,8194,8223,8236,8290,8304,8377,8425,8438,8439,8464,8481,8492,8521,8556,8559,8575,8582,8595,8602,8606,8624,8628,8648,8654,8666,8672,8689,8738,8739,8744,8775,8787,8837,8841,8842,8860,8928,8929,8970,8977,8993,9009,9019,9020,9029,9041,9051,9087,9111,9151,9195,9208,9298,9303,9327,9344,9352,9360,9364,9388,9400,9402,9446,9448,9449,9461,9462,9470,9485,9497,9512,9539,9546,9560,9572,9601,9612,9642,9649,9653,9677,9689,9692,9704,9708,9758,9765,9794,9813,9860,9916,9922,9927,9949,9971,9978,9981,9986,9987,10017,10040,10065,10073,10084,10097,10105,10120,10124,10134,10166,10187,10197,10202,10204,10241,10242,10279,10308,10324,10336,10351,10361,10458,10460,10567,10643,10676,10705,10712,10717,10759,10786,10787,10857,10883,10899,10911,10933,10944,10958,10963,11011,11015,11024,11036,11039,11049,11060,11119,11130,11146,11172,11203,11210,11216,11219,11230,11245,11316,11358,11371,11376,11423,11475,11534,11590,11649,11653,11677,11686,11707,11711,11740,11748,11751,11780,11823,11829,11843,11890,11896,11919,11947,11956,11976,12026,12037,12045,12072,12087,12108,12119,12154,12160,12208,12215,12216,12228,12229,12235,12247,12294,12333,12400,12437,12455,12458,12460,12469,12471,12510,12528,12567,12569,12593,12685,12694,12704,12721,12726,12754,12790,12817,12857,12914,12928,12936,12956,13002,13012,13026,13030,13035,13038,13057,13067,13082,13114,13143,13159,13193,13204,13214,13270,13278,13284,13326,13335,13417,13421,13423,13460,13479,13558,13607,13695,13696,13742,13764,13816,13827,13833,13837,13874,13879,13974,13987,14022,14100,14115,14140,14202,14272,14342,14350,14370,14376,14385,14393,14408,14409,14415,14417,14442,14486,14509,14560,14565,14713,14729,14743,14755,14798,14862,14874,14913,14934,14990,15007,15011,15120,15170,15194,15217,15227,15235,15285,15314,15321,15325,15332,15438,15499,15573,15611,15651,15668,15732,15735,15741,15757,15780,15808,15813,15847,15870,15941,15953,15977,16002,16017,16060,16108,16161,16286,16287,16304,16336,16374,16377,16384,16414,16505,16563,16623,16665,16670,16674,16689,16691,16710,16727,16743,16794,16828,16851,16900,16974,17005,17024,17029,17038,17039,17051,17086,17098,17148,17151,17195,17206,17266,17316,17323,17326,17331,17357,17376,17466,17489,17531,17559,17642,17681,17791,17868,17926,17929,17988,17991,18009,18026,18027,18056,18116,18168,18232,18307,18309,18438,18503,18504,18511,18590,18628,18629,18630,18636,18647,18672,18691,18694,18719,18909,18988,19023,19036,19096,19126,19132,19139,19193,19204,19210,19277,19304,19314,19325,19539,19544,19547,19631,19632,19635,19675,19700,19705,19740,19748,19921,19939,19951,19972,19985,20042,20052,20133,20141,20152,20173,20230,20245,20269,20287,20335,20355,20396,20407,20455,20501,20564,20580,20583,20664,20683,20710,20768,20776,20778,20789,20794,2098

8,21058,21087,21141,21143,21151,21186,21199,21216,21224,21385,21412,2  
1468,21475,21478,21479,21486,21487,21515,21569,21616,21629,21673,2170  
2,21729,21737,21747,21852,21927,21969,22060,22062,22068,22073,22114,2  
2131,22244,22301,22320,22366,22433,22450,22482,22490,22498,22536,2272  
7,22787,22947,22994,23010,23026,23063,23084,23135,23158,23180,23252,2  
3392,23457,23491,23500,23568,23607,23721,23730,23787,23935,23971,2399  
1,24023,24185,24215,24232,24398,24406,24476,24548,24550,24555,24562,2  
4566,24591,24592,24616,24633,24673,24721,24735,24743,24761,24832,2489  
1,24967,24976,25062,25080,25230,25391,25407,25433,25463,25493,25543,2  
5613,25668,25756,25919,26022,26048,26050,26092,26291,26297,26329,2634  
2,26371,26535,26566,26582,26676,26741,26838,26908,26910,26973,26984,2  
7111,27119,27163,27256,27296,27353,27392,27428,27492,27594,27644,2766  
6,27682,27771,27885,27895,27959,27987,28088,28116,28134,28137,28248,2  
8263,28365,28466,28548,28549,28787,28816,28845,28966,29002,29042,2905  
4,29072,29127,29138,29265,29326,29345,29434,29481,29487,29500,29588,2  
9731,29816,29827,29868,29905,29964,30037,30097,30153,30169,30280,3034  
6,30405,30433,30461,30493,30513,30550,30583,30646,30654,30909,30915,3  
0921,30930,30974,30997,31052,31056,31142,31199,31283,31285,31303,3150  
5,31578,31605,31948,31957,31997,32124,32139,32142,32272,32403,32555,3  
2601,32630,32631,32648,32699,32768,32807,32849,32912,32932,32961,3296  
5,33129,33171,33200,33282,33334,33623,34258,34302,34654,34708,35024,3  
5031,35388,35395,35462,35488,35586,35600,35747,35750,35774,35802,3607  
1,36112,36189,36252,36254,36294,36328,36357,36448,36476,36477,36479,3  
6485,36637,36749,36849,36874,36894,37170,37185,37187,37227,37612,3769  
5,37701,37767,37793,37805,37815,37826,37906,37992,38008,38010,38046,3  
8080,38130,38236,38385,38763,38787,39166,39176,39201,39237,39288,3939  
8,39482,39643,39786,39831,39960,39980,40089,40105,40140,40152,40192,4  
0220,40274,40293,40303,40398,40549,40604,40625,40666,40690,40816,4084  
3,40847,40894,40896,40962,40969,41003,41087,41107,41132,41216,41226,4  
1265,41314,41321,41357,41367,41539,41576,41641,41717,41820,42033,4206  
7,42172,42490,42662,42795,42813,42916,43339,43351,43388,43482,43498,4  
3691,43840,43905,43924,43932,44033,44129,44279,44821,44883,44945,4495  
1,45097,45162,45359,45389,45557,45582,45638,45813,45830,45919,45960,4  
6038,46086,46104,46187,46281,46428,46463,46481,46574,47047,47324,4741  
8,47523,47717,48007,48264,48334,48489,48501,48702,48788,48976,48994,4  
9504,49550,49703,49711,49978,49995,50006,50338,50511,50799,50946,5094  
7,50951,50980,51017,51150,51244,51530,51616,51977,52007,52062,52364,5  
2441,52586,52598,52768,52883,52978,53047,53064,53114,53127,54024,5454  
6,54578,54735,54803,55123,55289,55510,55661,55744,55843,55885,55921,5  
6297,56403,56696,57113,57424,57614,57779,58294,58326,58721,58908,5934  
6,59541,59651,59882,60076,60164,60250,60618,60799,61144,61208,61217,6  
1617

#### 2.7.8. Random Number Generator

The SC code utilizes two random number generators. The first uses the second. The first is described by the following pseudo code:

```
function b=RNG(a)
for i = 0:7
    a = RNG_2( a, ( (a) % (89) ) + 1 )
    b = (b) % (a)
end
```

The second random number generator uses a selectable set of feedback taps. The second is described by the following pseudo code:

```
function a=RNG_2(a,b)
tap_list=[32, 31, 30, 10
32, 31, 29, 1
32, 31, 26, 18
32, 31, 26, 9
32, 31, 26, 7
32, 31, 23, 10
32, 31, 22, 17
32, 31, 21, 16
32, 31, 21, 5
32, 31, 18, 10
32, 31, 16, 2
32, 31, 15, 10
32, 31, 14, 4
```

32, 31, 13, 8  
32, 31, 9, 7  
32, 31, 5, 4  
32, 30, 29, 23  
32, 30, 29, 20  
32, 30, 29, 16  
32, 30, 29, 15  
32, 30, 27, 24  
32, 30, 27, 21  
32, 30, 27, 12  
32, 30, 27, 8  
32, 30, 26, 25  
32, 30, 26, 13  
32, 30, 25, 16  
32, 30, 23, 16  
32, 30, 23, 14  
32, 30, 23, 4  
32, 30, 21, 14  
32, 30, 19, 8  
32, 30, 19, 4  
32, 30, 17, 3  
32, 30, 15, 6  
32, 30, 11, 8  
32, 30, 11, 5

32, 30, 8, 3  
32, 30, 7, 4  
32, 29, 28, 19  
32, 29, 27, 23  
32, 29, 27, 21  
32, 29, 27, 6  
32, 29, 26, 6  
32, 29, 25, 6  
32, 29, 22, 18  
32, 29, 19, 16  
32, 29, 17, 15  
32, 29, 15, 8  
32, 29, 6, 5  
32, 29, 6, 4  
32, 28, 25, 15  
32, 28, 25, 11  
32, 28, 25, 6  
32, 28, 23, 6  
32, 28, 15, 13  
32, 28, 9, 7  
32, 27, 26, 14  
32, 27, 25, 20  
32, 27, 25, 19  
32, 27, 25, 17

32, 27, 25, 7  
32, 27, 25, 5  
32, 27, 23, 6  
32, 27, 21, 6  
32, 27, 20, 18  
32, 27, 18, 14  
32, 27, 15, 14  
32, 27, 14, 12  
32, 27, 14, 9  
32, 27, 8, 6  
32, 26, 25, 10  
32, 26, 23, 12  
32, 26, 22, 7  
32, 26, 20, 11  
32, 26, 19, 9  
32, 26, 19, 7  
32, 26, 18, 13  
32, 26, 15, 7  
32, 25, 24, 7  
32, 25, 22, 15  
32, 25, 17, 7  
32, 25, 14, 13  
32, 24, 22, 13  
32, 23, 21, 16

```
32, 23, 18, 14
32, 21, 20, 19
32, 20, 17, 15
32, 19, 18, 13]
taps[0]=tap_list[b,0]
taps[1]=tap_list[b,1]
taps[2]=tap_list[b,2]
taps[3]=tap_list[b,3]

feedback=2.^(32-taps[0]) + 2.^(32-taps[1]) + 2.^(32-taps[2]) +
2.^(32-taps[3])

if( (a) & (1) )
    a = (a) ^ (feedback)
    a = (a) >> (1)
    a = (2^31) || (a)
else
    a = (a) >> (1)
end
```

#### 2.7.9. Random Permutation

The SC code utilizes a random permutation of length K to facilitate the construction of the random interleavers needed for the parallel filter codes. The random permutation is given by the following pseduocode. The RNG\_2 function is defined in section 2.7.8.

```
function [permutation,the_seed]= Generate_Permutation(a,b)

for i=0:a-1
    permutation[i] = i + 1
```



```
end

for i=0:a-1
    c = RNG_2(b,1)
    b = ( (c) % (a-(i-1)) ) + i - 1
    d = permutation[i]
    permutation[i] = permutation[b]
    permutation[b] = d
end
```

#### 2.7.10. RS Generator

A Reed Solomon code is utilized in the construction of the SC code. Its construction is described by the following pseudo code. The GF\_exp and the GF\_Multiply functions are defined in section 2.7.13.

```
function G_V_RS = RS_gen(K,N)
Gt=zeros[N,K]
for i=0:N-1
    for k=0:K-1
        a = ((i+1)*k) % (2^8-1)
        Gt[i,k]=GF_exp(a)
    end
end

G1=Gt[1:K,1:K]
G2=Gt[K+1:N,1:K]
```

```
G_V_RS = GF_Multiply(G2,G1^^-1)
```

GF\_Multiply implements  $G2 \cdot G1_{inv}$  where the multiplication and addition are performed in the GF field. The matrix inverse  $G1^{-1}$  can be easily implemented using Gaussian Elimination for the small matrix  $G1$ .

#### 2.7.11. RS Code

If the number of transmit symbols  $N$  is optionally limited to  $N \leq 256$  and signaled using  $Max\_N=0$ , then the following pseudo code is used to generate matrix  $P$ . The  $RS\_gen$  function is defined in section 2.7.10.

```
Num_V_RS = N - K

B_1 = RS_gen(K,K+Num_V_RS)

P = [I[K]

      B_1 ]

Num_B = 0

K_eff = K
```

#### 2.7.12. SC\_Filter\_Data

```
[filter_data, filter_N]=SC_filter_data(z)

Filter_data=[0,2147483648,2863311531,3221225472,3435973837,3579139413
,3681400539,3758096384,3817748708,3865470566,3904515724,3937053355,39
64585196,3988183918,4008636143,4026531840,4042322161,4056358002,40689
16386,4080218931,4090445044,4099741510,4108229587,4116010325,41231686
04,4129776246,4135894433,4141575607,4146864975,4151801719,4156419964,
4160749568,4164816772,4168644728,4172253945,4175662649,4178887099,418
1941841,4184839929,4187593114,4190211996,4192706170,4195084336,419735
4403,4199523578,4201598442,4203585013,4205488811,4207314902,420906795
0,4210752251,4212371771,4213930177,4215430865,4216876982,4218271451,4
219616993,4220916136,4222171240,4223384508,4224557996,4225693630,4226
793212,4227858432,4228890876,4229892034,4230863307,4231806012,4232721
393,4233610620,4234474799,4235314972,4236132128,4236927197,4237701065
```

```
,4238454568,4239188500,4239903613,4240600621,4241280205,4241943008,4242589646,4243220702,4243836733,4244438269,4245025816,4245599856,4246160849,4246709236,4247245437,4247769853,4248282869,4248784852,4249276155,4249757114,4250228053,4250689283,4251141099,4251583788,4294967295]
```

```
filter_N=min(100,z)
```

```
Filter_data[Filter_N-1]=4294967295
```

### 2.7.13. GF(256) Operations

The SC code utilizes Galois field arithmetic in GF(256). The primitive polynomial is  $D^8 + D^4 + D^3 + D^2 + 1$ . The `b=GF_exp(a)` function raises the primitive element to the supplied power, `a`. The function `C=GF_Multiply(A,B)` multiplies two matrices in the Galois field.

## 3. FEC Packets

Encoded packets are constructed using a 4 byte FEC Payload IE followed by transmit symbols. The Source ID field (SID) of the FEC Payload IE identifies the Source ID of the first transmit symbol in the packet. Subsequent transmit symbols have sequential increasing SIDs. If the last transmit symbol of a packet contains source padding, these padding bytes may be excluded from the packet. Otherwise, packets must contain only whole transmit symbols.

It is RECOMMENDED that each packet include exactly one transmit symbol. Multiple transmit symbols per packet SHALL also be supported.

### 3.1. Segmentation

In order to encode large files within the working memory constraint, the source file may need to be segmented into transmit blocks and working blocks.

#### 3.1.1. Transmit Blocks

Given a source file of size `F` bytes and a transmit symbol size of `T` bytes, the file can be divided into  $\text{ceil}(F/T)$  transmit symbols. A source transmit block is a collection of `KL` or `KS` of these transmit symbols. `KL` and `KS` may be different if the total number of source transmit blocks does not evenly divide the number of transmit symbols required to represent the file. The number of source transmit blocks with `KL` transmit symbols and the number of source transmit blocks

with KS transmit symbols are communicated to the decoder using parameters ZL and ZS, respectively. After encoding, a transmit block consists of a source transmit block and a repair transmit block.

The transmit blocks are ordered such that the first ZL transmit block are encoded from source transmit blocks of size KL transmit symbols. The remaining ZS transmit blocks are encoded from source transmit blocks of size KS transmit symbols. The parameters KS and KL are related to ZS and ZL via  $KS = \text{ceil}((F/T - ZL) / (ZL + ZS))$  and  $KL = KS + 1$ . Working Blocks

In order to satisfy the working memory requirement, the transmit symbols can be further subdivided into working symbols of size TW bytes. A transmit symbol therefore consists of  $T/TW$  working symbols. A source working block is then a collection of working symbols selected such that an entire source working block can fit into the

workin

g memory. The  $i$ th source working block consists of the  $i$ th working symbol of transmit symbols of a source transmit block. Additionally, a source working block is to be sized such the size of the working symbols remain a multiple of the byte alignment factor, AL. The KL (or KS) transmit symbols of a source transmit block can be viewed as a collection of working symbols or equivalently as a collection of source working blocks.

After encoding, a working block consists of a source working block and a repair working block. The receiver attempts to decode on a subset of the source and repair working symbols in a working block.

th The worki  
ng blocks are ordered in a packet such that the  $i$ th working symbol of TW bytes  
th corresponds to the  $i$ th working block. The  $i$ th packet contains  $i$ th working symbol for each of the working blocks.

### 3.1.2. Padding

In cases where effective number of transmit symbols used by the encoder and decoder,  $K_{\text{eff}}$ , is  $K_{\text{eff}} > K$ , then  $K_{\text{eff}} - K$  transmit symbols must be padded (with 0) to the data before encoding. These padded symbols do not need to be transmitted, as the decoder is aware that they are padding. (SIDs  $K$  to  $K_{\text{eff}} - 1$  MAY be transmitted, but it is RECOMMENDED that they are not.)

## 4. Parameter Selection

The code requires  $F$ ,  $T$ ,  $ZS$ , and  $TW$ .  $F$  is the total file size in Bytes.  $T$  is the transmit symbol size in bytes, and it is RECOMMENDED

that it be equal to the packet payload size. The number of transmit blocks  $ZS$  with  $KS$  transmit symbols (the number of working blocks with  $KS$  working symbols) MUST be chosen such that  $KL \leq K_{max}$ , where  $KL$  is computed in section 3.1.1.  $K_{max}$  is the maximum value in section 2.7.7.

The working symbols size in bytes,  $TW$ , MUST be chosen small enough such that  $K_L * TW$  is less than or equal to the working memory requirement. Additionally,  $TW$  MUST be chosen to be a multiple of the byte alignment factor  $AL$  and  $TW$  MUST be a divisor of  $T$ . The byte alignment,  $AL$ , is to be chosen based on the protocol and the typical machine architectures, a value of 4 (bytes) is RECOMMENDED.

## 5. Protocol IEs

This section describes IEs that are used by the FEC. All fields are big-endian.

The value of the FEC Encoding ID MUST be 7, as assigned by IANA (see Section 8).

### 5.1. FEC Payload IEs

The FEC payload ID is a 4-byte field defined as follows:

[0:7] TBN, (8 bits, unsigned integer): A non-negative integer identifier indicating the transmit block number.

[8:31] SID, (24 bits, unsigned integer): A non-negative integer identifier indicating the transmit symbols in the packet. SID 0 to  $K-1$  indicate systematic symbols.

The FEC Payload ID is shown in Figure 4.

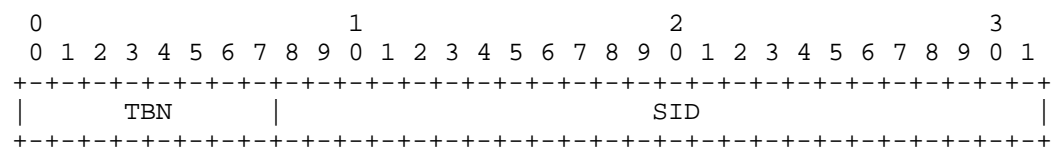


Figure 4 FEC Payload ID format

### 5.2. Common

The Common FEC Object Transmission Information elements used by this FEC Scheme are:

[0:39] Transfer Length (F), (40 bits, unsigned integer): A non-negative integer. This is the transfer length of the object in bytes.

[40:47] are reserved.

[48:63] Transmit Symbol Size (T), (16 bits, unsigned integer): A positive integer that is less than  $2^{16}$ . This is the size of a transmit symbol in units of bytes.

The encoded Common FEC Object Transmission Information format is shown in Figure 5.

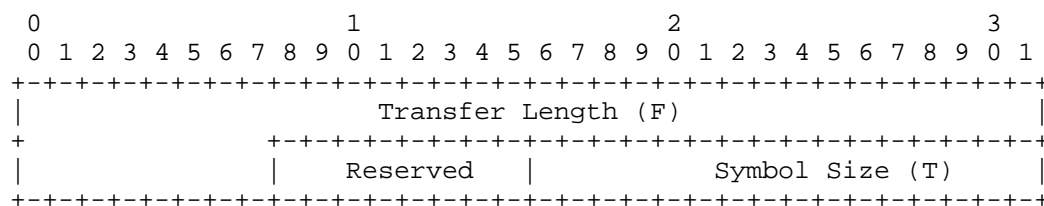


Figure 5 Encoded Common FEC IE for Supercharged FEC Scheme

### 5.3. Scheme Specific

The following parameters are carried in the Scheme-Specific FEC Object Transmission Information element for this FEC Scheme:

[0:7] ZL: The number of transmit blocks with KL working symbols (and the number of working blocks with KL working symbols). (8 bits, unsigned integer)

[8:15] ZS: The number of transmit blocks with KS working symbols (and the number of working blocks with KS working symbols). (8 bits, unsigned integer)

[16:30] TW: The working symbol size in bytes (15 bits, unsigned integer)

[31] Max\_N: 0: OPTIONALLY Indicates that the maximum value of N satisfies  $N \leq 256$ . 1: Otherwise or if unknown (1 bit, boolean).

The encoded Specific FEC Object Transmission Information format is shown in Figure 5.

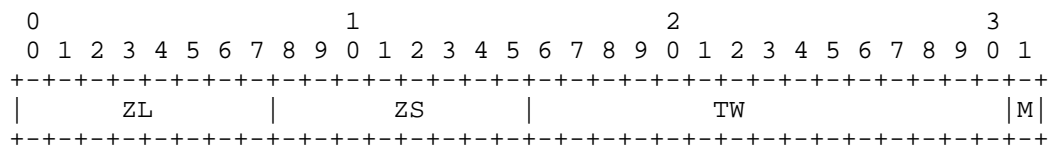


Figure 6 FEC Payload ID format

## 6. Conventions used in this document

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 RFC-2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

## 7. Security Considerations

Users could potentially be subject to a denial of service attack if a single erroneous packet is injected into the delivery stream. Therefore, it is RECOMMENDED that source authentication and integrity checking are applied to the file or data object before delivering decoded data to applications. The hashing methodology of SHA-256 is an example [2].

## 8. IANA Considerations

Values of FEC Encoding IDs and FEC Instance IDs are subject to IANA registration. For general guidelines on IANA considerations as they apply to this document, see [RFC5052]. IANA is requested to assign a value under the ietf:rmt:fec:encoding name-space to "Supercharged Code" as the FEC Encoding ID value associated with this specification, preferably the value 7.

## 9. References

### 9.1. Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [2] "Secure Hash Standard", National Institute of Standards and Technology FIPS PUB 180-3, October 2008.

## 9.2. Informative References

- [3] Timothy Vismor, "Matrix Algorithms."
- [4] Sergio Pissanetzky, "Sparse Matrix Technology," Academic Press, London (1984).
- [5] Timothy A. Davis, "Direct Methods for Sparse Linear Systems" SIAM, Philadelphia, Pa (2006)
- [6] Yousef Saad, "Iterative Methods for Sparse Linear Systems" 2nd Ed. SIAM, Philadelphia, Pa (2003)
- [7] I.S. Duff, A.M. Erisman, and J. K. Reid, "Direct Methods for Sparse Matrices" (2008) (ISBN: 978-0198534082)
- [8] John K. Reid, "Solution of linear systems of equations: Direct methods" (1977)
- [9] Golub, G.H. "Numerical methods for solving linear least-squares problems" Numerische Mathematik Volumn 7, Number 3 (1965) pp 206-216

## 10. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

## Authors' Addresses

Erik Stauffer  
Broadcom  
190 Mathilda Place  
Sunnyvale, Ca 94086  
  
Email: [eriks@broadcom.com](mailto:eriks@broadcom.com)



BZ Shen  
Broadcom  
5300 California Avenue  
Irvine, CA 92617

Email: bzshen@broadcom.com

Soumen Chakraborty  
Broadcom  
RMZ Ecospace  
Bellandur  
Bangalore 560037, India

Email: soumen@broadcom.com

Djordje Tujkovic  
Broadcom  
190 Mathilda Place  
Sunnyvale, Ca 94086

Email: djordje@broadcom.com

Jing Huang  
Broadcom  
190 Mathilda Place  
Sunnyvale, Ca 94086

Email: jingh@broadcom.com

Kamlesh Rath  
Broadcom  
190 Mathilda Place  
Sunnyvale, Ca 94086

Email: krath@broadcom.com

