Internet Engineering Task Force (IETF)
Request for Comments: 6968
Category: Experimental
ISSN: 2070-1721

V. Roca
INRIA
B. Adamson
Naval Research Laboratory

July 2013

FCAST: Object Delivery for the Asynchronous Layered Coding (ALC) and NACK-Oriented Reliable Multicast (NORM) Protocols

Abstract

This document introduces the FCAST reliable object (e.g., file) delivery application. It is designed to operate either on top of the underlying Asynchronous Layered Coding (ALC) / Layered Coding Transport (LCT) reliable multicast transport protocol or the NACK-Oriented Reliable Multicast (NORM) transport protocol.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6968.

Table of Contents

1. Introduction

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

- FCAST/ALC, which relies on the Asynchronous Layered Coding (ALC) [RFC5775] and Layered Coding Transport (LCT) [RFC5651] reliable multicast transport protocol, and

- FCAST/NORM, which relies on the NACK-Oriented Reliable Multicast (NORM) [RFC5740] 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.

FCAST is expected to work in many different environments and is designed to be flexible. The service provided by FCAST can differ according to the exact conditions under which FCAST is used. For instance, the delivery service provided by FCAST might be fully reliable, or only partially reliable, depending upon the exact way FCAST is used. Indeed, if FCAST/ALC is used for a finite duration over purely unidirectional networks (where no feedback is possible), a fully reliable service may not be possible in practice. This is different with NORM, which can collect reception and loss feedback from receivers. This is discussed in Section 6.

The delivery service provided by FCAST might also differ in terms of scalability with respect to the number of receivers. The FCAST/ALC service is naturally massively scalable, since neither FCAST nor ALC limits the number of receivers (there is no feedback message at all). Conversely, the scalability of FCAST/NORM is typically limited by NORM itself, as NORM relies on feedback messages from the receivers. However, NORM is designed in such a way to offer a reasonably scalable service (e.g., through the use of proactive Forward Error Correction (FEC) codes [RFC6363]), and so does the service provided by FCAST/NORM. This aspect is also discussed in Section 6.

A design goal behind FCAST is to define a streamlined solution, in order to enable lightweight implementations of the protocol stack and to limit the operational processing and storage requirements. A consequence of this choice is that FCAST cannot be considered 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 the File Delivery over Unidirectional Transport (FLUTE) [RFC6726], 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 metadata: by default, the metadata 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 metadata. 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.

Finally, Section 4 provides guidance for compliant implementation of the specification and identifies those features that are optional.

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 transport protocol.

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

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

Compound Object: denotes an ALC or NORM transport object composed of the FCAST Header and the Object Data (some Compound Objects may not include any Object Data).

FCAST Header: denotes the header prepended to the Object Data, which together form the Compound Object. This FCAST Header usually contains the object metadata, among other things.

Object Data: denotes the original object (e.g., a file) that forms the payload of the Compound Object.

Carousel: denotes the building block that enables an FCAST sender to transmit Compound Objects in a cyclic manner.

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, which might differ on a per-object basis, are possible.

Transport Object Identifier (TOI): denotes the numeric identifier associated with 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 FCAST Header (which usually contains the metadata of the object) to the Object Data (see Section 3.2). Figure 1 illustrates the associated format. All multi-byte fields MUST be in network (Big Endian) byte order.

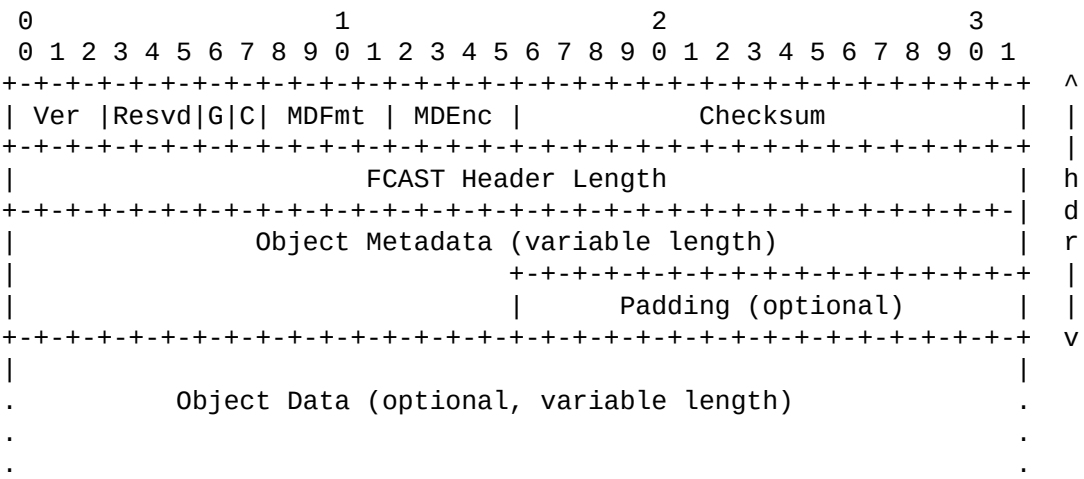

Figure 1: Compound Object Format

The FCAST Header fields are:

Field	Description
Version	3-bit field that MUST be set to 0 in this specification and that indicates the FCAST protocol version number.
Reserved	3-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 FCAST Header.
C	1-bit field that, when set to 1, indicates that the object is a CID. When set to 0, this field indicates that the transported object is a standard object.
Metadata Format (MDFmt)	4-bit field that defines the format of the Object Metadata field (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.
Metadata Encoding (MDEnc)	4-bit field that defines the optional encoding of the Object Metadata field (see Section 7). Two values are currently defined. A value of 0 indicates that the field contains UTF-8 encoded [RFC3629] text. A value of 1 indicates that the field contains GZIP [RFC1952] compressed UTF-8 encoded text.

Checksum	16-bit field that contains the checksum computed over either the whole Compound Object (when G is set to 1) or over the FCAST 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 MUST be set to zero.
FCAST Header Length	32-bit field indicating total length (in bytes) of all fields of the FCAST Header, except the optional padding. An FCAST Header Length field set to value 8 means that there is no metadata included. When this size is not a multiple of 32-bit words and when the FCAST Header is followed by non-null Object Data, padding MUST be added. It should be noted that the Object Metadata field maximum size is equal to (2^32 - 8) bytes.
Object Metadata	Variable-length field that contains the metadata associated to the object. The format and encoding of this field are defined by the MDFmt and MDEnc fields, respectively. With the default format and encoding, the Object Metadata field, if not empty, MUST contain UTF-8 encoded text that follows the "TYPE" ":" "VALUE" "<CR-LF>" format used in HTTP/1.1 for metainformation [RFC2616]. The various metadata items can appear in any order. The receiver MUST NOT assume that this string is NULL-terminated. When no metadata is communicated, this field MUST be empty and the FCAST 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 a 32-bit boundary. Padding is only used when the FCAST Header Length value, in bytes, is not a multiple of 4 and when the FCAST Header is followed by non-null Object Data.

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

2.2. Carousel Instance Descriptor Format

In an FCAST session, a 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.

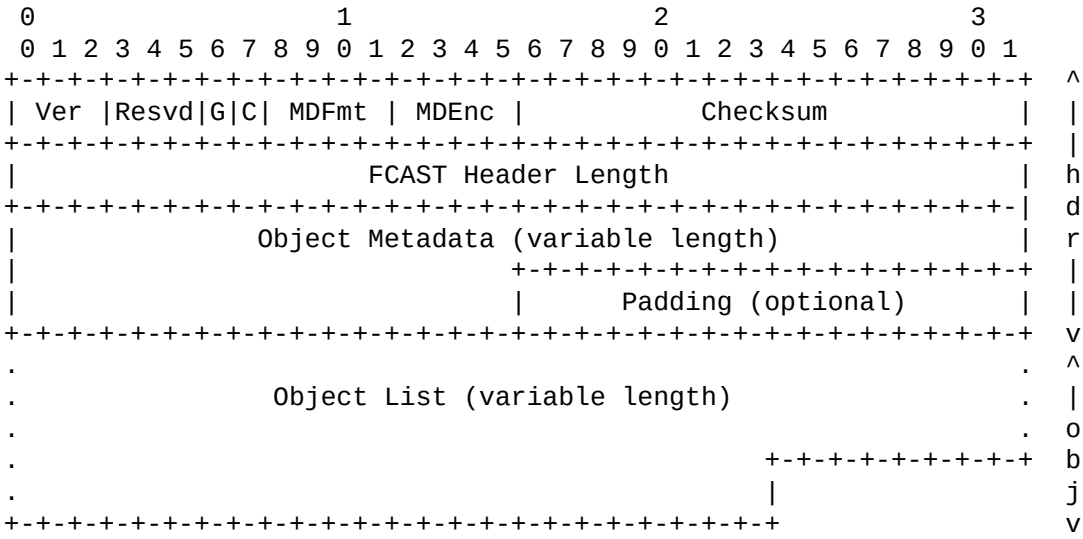

Figure 2: Carousel Instance Descriptor Format

Because the CID is transmitted as a special Compound Object, the following CID-specific metadata entries are defined and MUST be supported:

- Fcast-CID-Complete: this is an optional entry that, when set to "Fcast-CID-Complete: 1", indicates no new object (if we ignore CID Compound Objects) in addition to the ones whose TOIs are listed in this CID or the ones that have been listed in the previous CID(s), will be sent in the future. Conversely, if it is set to "Fcast-CID-Complete: 0", or if this entry is absent, it indicates that the session is not complete. An FCAST sender MUST NOT use any other value for this entry.

- Fcast-CID-ID: this entry contains the Carousel Instance IDentifier, or CIID. It starts from 0 upon FCAST session creation 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 (no need for the FCAST sender to specify it and for the FCAST receiver to process it). 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 for detecting possible gaps in the Carousel Instances, for instance, gaps caused by long disconnection periods. Finally, it can also be useful for avoiding problems when TOI wrapping to 0 takes place to differentiate the various incarnations of the TOIs if need be.

The following standard metadata entry types are also used (Section 3.3):

- Content-Length: specifies the size in bytes of the Object List, before any content encoding (if any).

- Content-Encoding: specifies the optional encoding of the Object List, performed by FCAST.

An empty Object List is valid and indicates that the current Carousel Instance does not include any objects (Section 3.5). This can be specified by using the following metadata entry:

```
Content-Length: 0
```

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

The Object List, when non-empty and with MDEnc=0, is UTF-8-encoded text that is not necessarily NULL-terminated. It can contain two things:

- a list of TOI values, and

- a list of TOI equivalences.

A list of TOIs included in the current Carousel Instance is specified as an ASCII string containing comma-delimited individual TOI values and/or TOI intervals. Individual TOIs consist of a single integer value, while TOI intervals are a hyphen-delimited pair of TOI values

to indicate an inclusive range of TOI values (e.g., "1,2,4-6" would indicate the list of TOI values of 1, 2, 4, 5, and 6). For a TOI interval indicated by "TOI_a-TOI_b", the "TOI_a" value MUST be strictly inferior to the "TOI_b" value. 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 equivalence TOI value pairs with a delimiting equals sign '=' to indicate the equivalence assignment (e.g., " newTOI "=" 1stTOI "/" 1stCIID "). Each equivalence indicates 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. In the case of the NORM protocol, where NormTransportId values need to monotonically increase for NACK-based protocol operation, this allows an object from a prior Carousel Instance to be relisted in a subsequent Carousel Instance with the receiver set informed of the equivalence so that unnecessary retransmission requests can be avoided.

The ABNF [RFC5234] is as follows:

```
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 and to simplify processing, the TOI values in the list MUST be given in increasing order, handling wrap of the TOI space appropriately. TOI equivalence elements MUST be grouped together at the end of the list in increasing newTOI order. Specifying a TOI equivalence for a given newTOI relieves the sender from specifying newTOI explicitly in the TOI list. 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 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 transport protocol, which requires positive/negative acknowledgments 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. Compound Object and Metadata Transmission

FCAST carries metadata elements by prepending them to the object they refer to. As a result, a Compound Object is created that is composed of an FCAST Header followed by the Object Data (Figure 3). This header is itself composed of the object metadata (if any) as well as several fields (e.g., to indicate format, encoding, or boundaries (Section 2.1)).

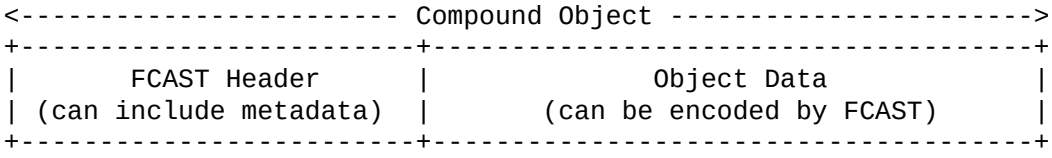

Figure 3: Compound Object Composition

Attaching the metadata to the object is an efficient solution, since it guarantees that metadata are received along with the associated object, and it allows the transport of the metadata 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 metadata of an object before beginning its reception, and in the case of erasures affecting the metadata, 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 metadata, e.g., to communicate metadata ahead of time.

3.3. Metadata Content

The following metadata types are defined in [RFC2616]:

- Content-Location: the URI of the object, which gives the name and location of the object.

- Content-Type: a string that contains the MIME type of the object.

- Content-Length: an unsigned 64-bit integer that contains the size in bytes of the initial object, before any content encoding (if any) and without considering the FCAST Header. Note that the use of certain FEC schemes MAY further limit the maximum value of the object.

- Content-Encoding: a string that contains the optional encoding of the object performed by FCAST. For instance:

  Content-Encoding: gzip

  indicates that the object has been encoded with GZIP [RFC1952]. If there is no Content-Encoding entry, the receiver MUST assume that FCAST did not modify the original encoding of the object (default).

The following additional metadata types are defined to check object integrity:

- Fcast-Obj-Digest-SHA256: a string that contains the "base64" [RFC4648] encoding of the SHA-256 message digest of the object [RFC3174] [RFC6234], before any content encoding is applied (if any) and without considering the FCAST Header. This digest is meant to protect from transmission and processing errors, not from deliberate attacks by an intelligent attacker (see Section 5). This digest only protects the object, not the header, and therefore not the metadata. A separate checksum is provided for that purpose (Section 2.1).

- Fcast-Obj-Digest-SHA1: similar to Fcast-Obj-Digest-SHA256, except that SHA-256 is replaced by SHA-1. An FCAST sender MAY include both an Fcast-Obj-Digest-SHA1 and an Fcast-Obj-Digest-SHA256 message digest in the metadata, in order to let a receiver select the most appropriate algorithm (e.g., depending on local processing power).

The following additional metadata types are used for dealing with very large objects (e.g., objects that largely exceed the working memory of a receiver). When this happens, the metadata associated to each sub-object MUST include the following entries:

- 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.

- Fcast-Obj-Slice-Idx: an unsigned 32-bit integer that contains the slice index (in the {0 .. SliceNb - 1} interval).

- Fcast-Obj-Slice-Offset: an unsigned 64-bit integer that contains the offset at which this slice starts within the original object.

Future IANA assignments to extend the set of metadata types supported by FCAST are to be made through Expert Review [RFC5226].

3.4. Carousel Transmission

A set of FCAST Compound Objects scheduled for transmission is 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, a sender and receiver(s) are both capable of multiplexing and demultiplexing 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. For example, this can be the case for the CID objects (Section 3.5), where extra protection can benefit overall carousel integrity. For some FCAST usage (e.g., a unidirectional "push" mode), a Carousel Instance may be sent in a single transmission cycle. In other cases, it may be conveyed in 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 CID. The CID carries the list of the Compound Objects that are part of a given Carousel Instance by specifying their respective Transport Object Identifiers (TOIs). However, the CID does not describe the objects themselves (i.e., there is no metadata). Additionally, the CID MAY include an "Fcast-CID-Complete: 1" metadata entry 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 (CIID) 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 FCAST Header field (the "C" flag from Figure 1) so that it can be recognized and processed by the FCAST application as needed. A direct consequence is that since a receiver does not know in advance which TOI will be used for the following CID (in the case of a dynamic session), it MUST NOT filter

out packets that are not in the current CID's TOI list. Said differently, the goal of the CID is not to set up 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:

- When an "Fcast-CID-Complete" metadata entry set to "Fcast-CID-Complete: 1" is included, 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.

- In the 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 "B" flag of ALC/LCT, since a client with 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.

- The TOI equivalence (Section 3.6) is signaled within the CID.

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:

- all the previously sent objects have been removed from the carousel. This therefore improves the robustness of FCAST even during "idle" periods.

- 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 no "Fcast-CID-Complete" metadata entry is included (or if set to "Fcast-CID-Complete: 0"), it informs the receivers that the Carousel Instance may be modified and that new objects could 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 protocol, the packets containing transport object content are labeled with a numeric transport object identifier: the TOI with ALC, and the NormTransportId with NORM. 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:

- ALC's use of the 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 management of the TOI is totally free as long as each object is associated to a unique TOI (if no wraparound occurred).

- NORM's use of the TOI serves a more "directive" purpose, since the TOI field is 16 bits long and since TOIs MUST be managed sequentially.

In both NORM and ALC, it is possible that the transport identification space eventually wraps 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 the 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 associated to the previously assigned TOI, a mechanism is required to equate the objects with the new and the old TOIs. This mechanism consists of 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; CIID} tuple used for its first transmission within a Carousel Instance. This tuple MUST be used whenever a TOI equivalence is provided. Section 2.2 details how to describe these TOI equivalences.

3.7. FCAST Sender Behavior

This section provides an informative description of expected FCAST sender behavior. The following operations can 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 metadata, to the FCAST application.

2. For each object, FCAST creates the Compound Object and registers it 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 "Fcast-CID-Complete: 1" if no new objects will be sent in the future.

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

This section provides an informative description of expected FCAST receiver behavior. The following operations can 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 metadata is sent by means of another mechanism prior to the object, the receiver processes the metadata and chooses whether or not to continue to receive the object content.

3. When a Compound Object has been entirely received, the receiver processes the header, retrieves the object metadata, perhaps decodes the metadata, and processes the object accordingly.

4. When a CID is received, as indicated by the "C" flag set in the FCAST 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 receives a CID with an "Fcast-CID-Complete" metadata entry set to "Fcast-CID-Complete: 1" 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. 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.

4.1. Requirements Related to the Object Metadata

Metadata transmission mechanisms:

Feature	Status
metadata transmission using FCAST's in-band mechanism	An FCAST sender MUST send metadata with the in-band mechanism provided by FCAST, i.e., within the FCAST Header. All the FCAST receivers MUST be able to process metadata sent with this FCAST in-band mechanism.
metadata transmission using other mechanisms	In addition to the FCAST in-band transmission of metadata, an FCAST sender MAY send a subset or all of the metadata 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 metadata sent in this way. If that is not the case, the FCAST receiver will receive and process metadata sent in-band anyway. See Appendix B.

Metadata format and encoding:

Feature	Status
Metadata Format (MDFmt field)	All FCAST implementations MUST support an HTTP/1.1 metainformation format [RFC2616].
Metadata Encoding (MDEnc field)	All FCAST implementations MUST support both UTF-8 encoded text and GZIP compressed [RFC1952] UTF-8 encoded text for the Object Metadata field.

Metadata 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. All FCAST implementations MUST support GZIP encoding [RFC1952].
Fcast-Obj-Digest-SHA1	MUST be supported.
Fcast-Obj-Digest-SHA256	MUST be supported.
Fcast-Obj-Slice-Nb	MUST be supported.
Fcast-Obj-Slice-Idx	MUST be supported.
Fcast-Obj-Slice-Offset	MUST be supported.

4.2. Requirements Related to the Carousel Instance Descriptor

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.

CID-specific Metadata items (Section 2.2):

Feature	Status
Fcast-CID-Complete	MUST be supported.
Fcast-CID-ID	MUST be supported.

5. Security Considerations

5.1. Problem Statement

A content delivery system may be subject to attacks that target:

- the network, to compromise the delivery infrastructure (e.g., by creating congestion),
- the Content Delivery Protocol (CDP), to compromise the delivery mechanism (i.e., FCAST in this case), or
- the content itself, to corrupt the objects being transmitted.

These attacks can be launched against all or any subset of the following:

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

More details on these possible attacks are provided in the following sections, along with possible countermeasures. Recommendations are made in Section 5.5.

5.2. Attacks against the Data Flow

The following types of attacks against the data flow exist:

- attacks that are meant to gain unauthorized access to a confidential object (e.g., obtaining non-free content without purchasing it) and
- 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; this would be a denial-of-service (DoS) attack).

5.2.1. Attacks Meant to Gain Access to Confidential Objects

Modern cryptographic mechanisms can provide access control to transmitted objects. One way to do this is by encrypting the entire object prior to transmission, knowing that authenticated receivers have the cryptographic mechanisms to decrypt the content. Another way is to encrypt individual packets using IPsec/ESP [RFC4303] (see also Section 5.5). These two techniques can also provide confidentiality to the objects being transferred.

If access control and/or confidentiality services are desired, one of these mechanisms is RECOMMENDED and SHOULD be deployed.

5.2.2. Attacks Meant to Corrupt Objects

Protection against attacks on the data integrity of the object may be achieved by a mechanism agreed upon between the sender and receiver that features sender authentication and a method to verify that the object integrity has remained intact during transmission. This service can be provided at the object level, but in that case a receiver has no way to identify what symbols are corrupted if the object is detected as corrupted. This service can also be provided at the packet level. In some cases, after removing all corrupted packets, the object may be recovered. Several techniques can provide data integrity and sender authentication services:

- 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. Even if digital signatures are computationally expensive, this calculation occurs only once per object, which is usually acceptable.

- At the packet level, each packet can be digitally signed [RFC6584]. A major limitation is the high computational and transmission overheads that this solution requires.

- At the packet level, a Group-keyed Message Authentication Code (MAC) [RFC2104] [RFC6584] 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 itself. The Group-keyed MAC scheme does not create prohibitive processing loads or 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; this 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 preliminary check to quickly detect attacks initiated by non-group members and to discard their packets.

- At the packet level, Timed Efficient Stream Loss-Tolerant Authentication (TESLA) [RFC4082] [RFC5776] is an attractive solution that is robust to losses, provides an authentication and integrity verification service, and does not create any prohibitive processing load or transmission overhead. Yet, a delay is incurred in checking a TESLA authenticated packet; this delay may be more than what is desired in some use-cases.

- 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 5.5).

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 via a Public Key Infrastructure (PKI), a Pretty Good Privacy (PGP) Web of Trust, or by securely preplacing the public keys of each group member.

Techniques relying on symmetric key cryptography (Group-keyed 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 securely preplacing 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 a threat of object corruption, it is RECOMMENDED that at least one of these techniques be used. Section 5.5 defines minimum security recommendations that can be used to provide such services.

5.3. 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 can target, among other things, the following:

- the session description,

- the FCAST CID,

- the metadata of an object,

- the ALC/LCT parameters, carried within the LCT header, or
- 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 the content delivery system or even compromise the network itself.

5.3.1. Attacks against the Session Description

An FCAST receiver may potentially obtain an incorrect session description for the session. The consequence is that legitimate receivers with the wrong session description will be unable to correctly receive the session content or will 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 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.

5.3.2. Attacks against the FCAST CID

Corrupting the FCAST CID is one way to create a DoS attack. For example, the attacker can insert an "Fcast-CID-Complete: 1" metadata entry 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 countermeasures mentioned above (Section 5.2.2) SHOULD be used. These measures will either be applied at the 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.

5.3.3. Attacks against the Object Metadata

Modifying the object metadata is another way to launch an attack. For example, the attacker may change the message digest associated to an object, leading a receiver to reject an object even if it has been correctly received. More generally, a receiver SHOULD be very careful during metadata processing. For instance, a receiver SHOULD NOT try to follow links (e.g., the URI contained in the

Content-Location metadata). As another example, malformed HTTP content can be used as an attack vector, and a receiver should take great care with such content.

It is therefore RECOMMENDED that measures be taken to guarantee the integrity and to check the identity of the sender of the Compound Object. To that purpose, one of the countermeasures mentioned above (Section 5.2.2) SHOULD be used. These measures will either be applied at the 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.

5.3.4. 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 "A" flag set to 1 can lead the receiver to prematurely close the session. Similarly, sending forged ALC packets with the Close Object "B" flag set to 1 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 in each ALC or NORM packet received. To that purpose, one of the countermeasures mentioned above (Section 5.2.2) SHOULD be used.

5.3.5. 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 and 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 authenticated as legitimate receivers before they can join the session. If authenticating a receiver does not prevent that receiver from launching an attack, the network operator will still be able to easily identify the receiver that launched the attack and take countermeasures. The details of how this 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 5.2.2. 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 reduce its subscription level when the receiver believes that 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 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 might be preferable, or a Group-keyed MAC scheme could be used in parallel with TESLA to prevent attacks launched from outside of the group.

5.4. 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 apply to FCAST. Finally, the security discussion of [RMT-SEC] also applies here.

5.5. Minimum Security Recommendations

We now introduce a security configuration that is mandatory to implement but not necessarily mandatory to use, 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]. Therefore, IPsec/ESP in transport mode MUST be implemented, but not necessarily used, in accordance with [RFC5775] and [RFC5740]. [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 the 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.

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 might 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 5).

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, are 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:

- the way ALC or NORM is used (e.g., whether FEC encoding and/or NACK-based repair requests are used or not),
- the way the FCAST carousel is used (e.g., whether the objects are made available for a long time span or not), and
- the way in which FCAST itself is deployed (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 can be restricted to a single receiver or 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 scalability is at a maximum, since neither FCAST, ALC, UDP, nor IP generates any feedback message. On the contrary, the scalability of FCAST/NORM is typically limited by the scalability of NORM itself. For example, NORM can be configured to operate using proactive FEC without feedback, similar to ALC, with receivers configured to provide NACK and, optionally, ACK feedback,

or a hybrid combination of these. 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 servers or feedback message aggregation. 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 CID (Section 3.5). The decision of whether the CID mechanism should be used or not is left to the sender. When it is used, the question of how often and when a CID should be sent is also left to the sender. These considerations 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

Per this specification, IANA has created three new registries.

7.1. Creation of the FCAST Object Metadata Format Registry

IANA has created a new registry, "FCAST Object Metadata Format" (MDFmt), with a reference to this document. The registry entries consist of a numeric value from 0 to 15, inclusive (i.e., they are 4-bit positive integers), that defines the format of the object metadata (see Section 2.1).

The initial value for this registry is defined below. Future assignments are to be made through Expert Review with Specification Required [RFC5226].

Value	Format Name	Format Reference	Reference
0 (default)	HTTP/1.1 metainformation format	[RFC2616], Section 7.1	This specification

7.2. Creation of the FCAST Object Metadata Encoding Registry

IANA has created a new registry, "FCAST Object Metadata Encoding" (MDEnc), with a reference to this document. The registry entries consist of a numeric value from 0 to 15, inclusive (i.e., they are 4-bit positive integers), that defines the encoding of the Object Metadata field (see Section 2.1).

The initial values for this registry are defined below. Future assignments are to be made through Expert Review [RFC5226].

Value	Encoding Name	Encoding Reference	Reference
0	UTF-8 encoded text	[RFC3629]	This specification
1	GZIP'ed UTF-8 encoded text	[RFC1952], [RFC3629]	This specification

7.3. Creation of the FCAST Object Metadata Types Registry

IANA has created a new registry, "FCAST Object Metadata Types" (MDType), with a reference to this document. The registry entries consist of additional text metadata type identifiers and descriptions for metadata item types that are specific to FCAST operation and not previously defined in [RFC2616]. The initial values are those described in Section 3.3 of this specification. This table summarizes those initial registry entries. Future assignments are to be made through Expert Review [RFC5226].

Metadata Type	Description	Reference
Fcast-Obj-Digest-SHA1	A string that contains the "base64" encoding of the SHA-1 message digest of the object before any content encoding is applied (if any) and without considering the FCAST Header	This specification
Fcast-Obj-Digest-SHA256	A string that contains the "base64" encoding of the SHA-256 message digest of the object before any content encoding is applied (if any) and without considering the FCAST Header	This specification
Fcast-Obj-Slice-Nb	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	This specification
Fcast-Obj-Slice-Idx	Unsigned 32-bit integer that contains the slice index (in the {0 .. SliceNb - 1} interval)	This specification
Fcast-Obj-Slice-Offset	Unsigned 64-bit integer that contains the byte offset at which this slice starts within the original object	This specification

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, Gorry Fairhurst, and Lorenzo Vicisano for their valuable comments. The authors are also grateful to Jari Arkko, Ralph Droms, Wesley Eddy, Roni Even, Stephen Farrell, Russ Housley, Chris Lonvick, Pete Resnick, Joseph Yee, and Martin Stiemerling.

9. References

9.1. Normative References

[RFC1071] Braden, R., Borman, D., Partridge, C., and W. Plummer, "Computing the Internet checksum", RFC 1071, September 1988.

[RFC1952] Deutsch, P., "GZIP file format specification version 4.3", RFC 1952, May 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[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.

[RFC3174] Eastlake, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, September 2001.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[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.

[RFC6234] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, May 2011.

9.2. Informative References

[ALC-00] Luby, M., Gemmell, J., Vicisano, L., Rizzo, L., Crowcroft, J., and B. Lueckenhoff, "Asynchronous Layered Coding: A scalable reliable multicast protocol", Work in Progress, March 2000.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC3365] Schiller, J., "Strong Security Requirements for Internet Engineering Task Force Standard Protocols", BCP 61, RFC 3365, August 2002.

[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.

[RFC6363] Watson, M., Begen, A., and V. Roca, "Forward Error Correction (FEC) Framework", RFC 6363, October 2011.

[RFC6584] Roca, V., "Simple Authentication Schemes for the Asynchronous Layered Coding (ALC) and NACK-Oriented Reliable Multicast (NORM) Protocols", RFC 6584, April 2012.

[RFC6726] Paila, T., Walsh, R., Luby, M., Roca, V., and R. Lehtonen, "FLUTE - File Delivery over Unidirectional Transport", RFC 6726, November 2012.

[RMT-SEC] Adamson, B., Roca, V., and H. Asaeda, "Security and Reliable Multicast Transport Protocols: Discussions and Guidelines", Work in Progress, May 2013.

Appendix A. FCAST Examples

This appendix provides informative examples of FCAST Compound Objects and Carousel Instance Descriptor formats.

A.1. Simple Compound Object Example

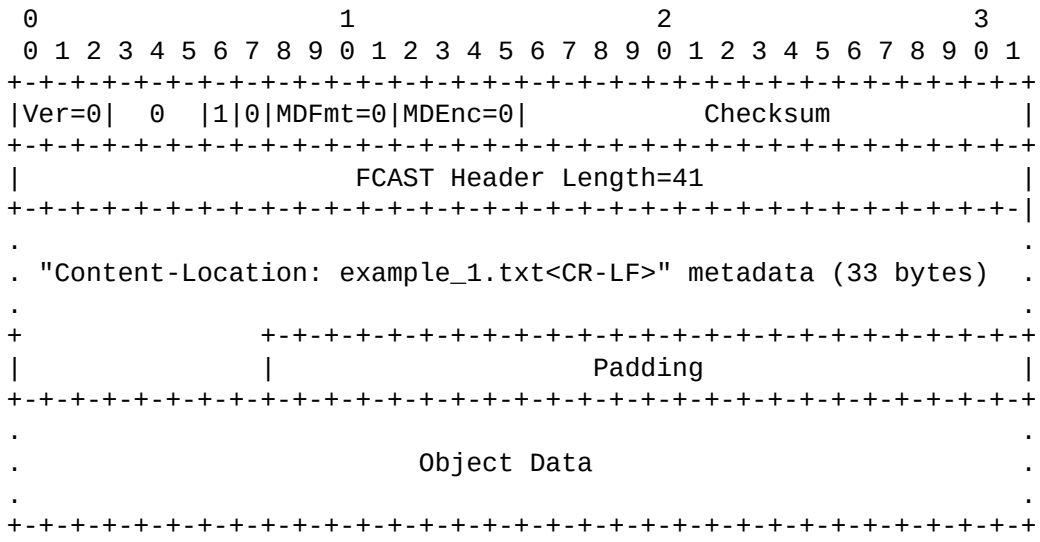

Figure 4: Simple Compound Object Example

Figure 4 shows a simple Compound Object where the metadata string, in HTTP/1.1 metainformation format (MDFmt=0), contains:

Content-Location: example_1.txt<CR-LF>

This UTF-8 encoded text (since MDEnc=0) is 33 bytes long (there is no final '\0' character). Therefore, 3 padding bytes are added. There is no Content-Length metadata entry for the object transported (without FCAST additional encoding) in the Object Data field, 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

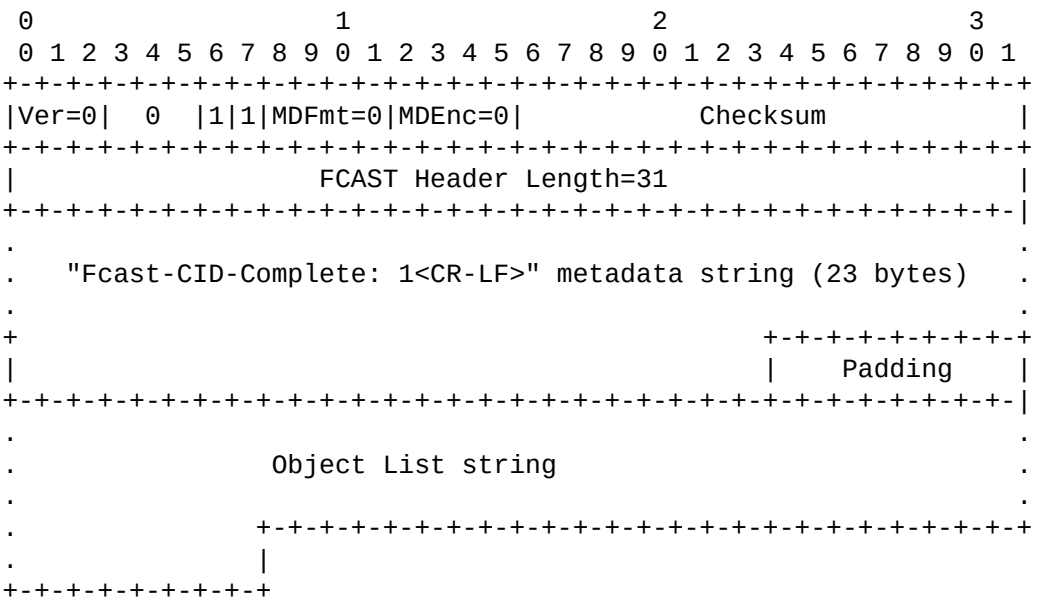

Figure 5: CID Object Example: Static Session

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. The metadata UTF-8 encoded text only contains the following entry, since Fcast-CID-ID is implicit:

```
Fcast-CID-Complete: 1<CR-LF>
```

This UTF-8 encoded text (since MDEnc=0) is 23 bytes long (there is no final '\0' character). Therefore, 1 padding byte is added.

The Object List contains the following 25-byte-long string (there is no final '\0' 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 metadata associated to this CID. As the session is static and composed of a single Carousel Instance, the sender did not feel the necessity to carry a Carousel Instance ID metadata.

Appendix B. Additional Metadata Transmission Mechanisms

B.1. Supporting Additional Mechanisms

In certain use-cases, FCAST can take advantage of another in-band (e.g., via NORM_INFO messages (Appendix B.2)) or out-of-band signaling mechanism. This section provides an overview of how other signaling mechanisms could be employed and a normative specification for how FCAST information is embedded when NORM_INFO messages are used for carrying FCAST Headers. Such additional signaling schemes can be used to carry the whole metadata, or a subset of it, ahead of time, before the associated Compound Object. Therefore, based on the information retrieved in this way, a receiver could decide in advance (i.e., before beginning the reception of the compound object) whether the object is of interest or not; this would mitigate the limitations of FCAST. While out-of-band techniques are out of the scope of this document, we explain below how this can be achieved in the case of FCAST/NORM.

Supporting additional mechanisms is OPTIONAL in FCAST implementations. In any case, an FCAST sender MUST continue to send all the required metadata in the Compound Object, even if the whole metadata, or a subset of it, is sent by another mechanism (Section 4). Additionally, when metadata is sent several times, there MUST NOT be any contradiction in the information provided by the different mechanisms. If a mismatch is detected, the metadata contained in the Compound Object MUST be used as the definitive source.

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

- TOI: a positive integer that contains the Transport Object Identifier (TOI) of the object, in order to enable a receiver to easily associate the metadata 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 metadata. In that case, the NORM_INFO message carries a new Compound Object that contains all the metadata 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 a 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 FCAST 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 the case of FCAST/NORM, the object metadata (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 Object Data. In the following informative example, we assume that the whole metadata is carried in such a message. Figure 6 shows an example NORM_INFO message that contains the FCAST Header, including metadata. 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 Header, including metadata. Note that "padding" MUST NOT be used and that the FCAST checksum only encompasses the Compound Object Header (G=0).

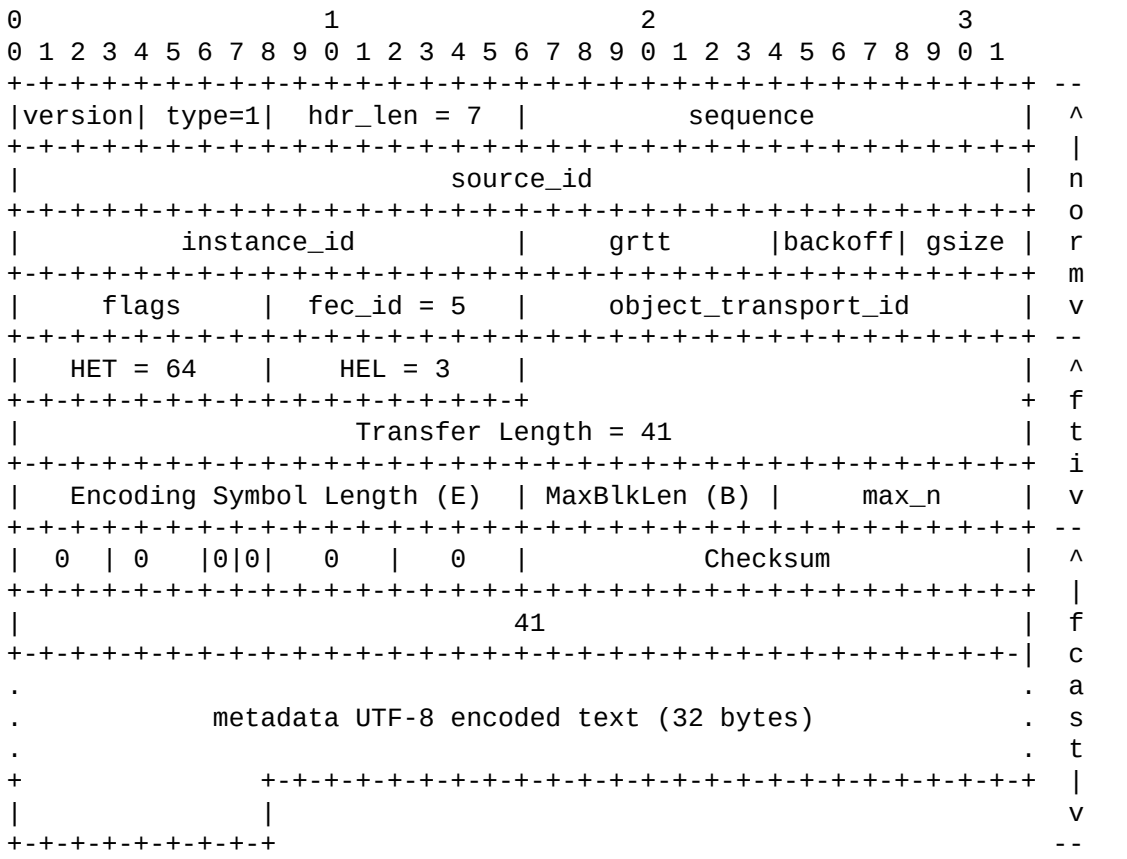

Figure 6: NORM_INFO Message Containing an EXT_FTI Header Extension and an FCAST Header

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 metadata of the FCAST 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 Compound Object, i.e., 41 bytes.

The Compound Object in this example does contain the same metadata and is formatted as in the example of Figure 4. With the combination of the FEC_OTI and the FCAST metadata, 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
URI: http://planete.inrialpes.fr/people/roca/

Brian Adamson
Naval Research Laboratory
Washington, DC 20375
USA

EMail: adamson@itd.nrl.navy.mil
URI: http://cs.itd.nrl.navy.mil