Network Working Group
Internet Draft
<draft-roca-rmt-newfcast-02>
Intended status: Experimental
Expires: January 2009

V. Roca
INRIA
B. Adamson
Naval Research Laboratory
July 2008

FCAST: Scalable Object Delivery for the ALC and NORM Protocols
draft-roca-rmt-newfcast-02

Status of this Memo

Copyright Notice

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.

1. Introduction

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

- FCAST/ALC that relies on the Asynchronous Layer Coding (ALC) [2] and the Layered Coding Transport (LCT) [3] reliable multicast transport protocol, and
- FCAST/NORM that relies on the NACK-Oriented Reliable Multicast (NORM) [4] reliable multicast transport protocol.

Hereafter, the term FCAST denotes either FCAST/ALC or FCAST/NORM.

Depending on the target use case, the delivery service provided by FCAST is more or less reliable. For instance, with FCAST/ALC used in ON-DEMAND mode over a time period that largely exceeds the typical download time, the service can be considered as fully reliable. Similarly, when FCAST is used along with a session control application that collects reception information and takes appropriate corrective measures (e.g., a direct point-to-point retransmission of missing packets, or a new multicast recovery session, see Appendix A), then the service can be considered as fully reliable. On the opposite, if FCAST operates in PUSH mode, then the service is usually only partially reliable, and a receiver that is disconnected during a sufficient time will perhaps not have the possibility to download the object.

Depending on the target use case, the FCAST scalability is more or less important. 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 opposite, 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 limits 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 is out of the scope of the present document.

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 opposite, FCAST has some intrinsic limitations. From this point of view it differs from FLUTE [5] which favors flexibility at the expense of some additional complexity.

A good example of the design choices that are meant to favor simplicity, is the way FCAST manages the meta-data of an object: with FCAST, the meta-data are simply prepended to the object. This solution has many advantages in terms of simplicity as will be described later on. But it also has an intrinsic limitation since it does not enable a receiver to decide in advance, before beginning reception of the object, whether the object is of interest or not. Thus, if there is no out-of-band mechanism to enable receivers to obtain the meta-data (or a subset) in advance, then all the objects sent in the FCAST session should be of interest to all receivers. If this is not the case, a receiver will probably waste time and resources to receive and decode objects that will turn out to be useless to him.

1.1 Applicability

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 6).

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.

2. 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 [1].

3. Definitions, Notations and Abbreviations

3.1 Definitions

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 5.1, including any meta-data and the content of the original application object (e.g., a file);

Carousel denotes the compound object transmission system of an 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 Object (CIO) denotes a specific object that lists the compound objects that comprise a given 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;

The Transmission Object Identifier (TOI) refers the numeric identifier associated to a specific object by the underlying transport layer. In the case of ALC, this corresponds to the TOI described in that specification while for the NORM specification this corresponds to the NormObjectId described there.

3.2 Abbreviations

This document uses the following abbreviations:

Abbreviation	**Definition**
CIO	Carousel Instance Object
FEC OTI	FEC Object Transmission Information
TOI	Transmission Object Identifier

4. FCAST Principles

4.1 FCAST Content Delivery Service

The basic goal of FCAST is to transmit objects to a group of receivers in a reliable way. The receiver set MAY be restricted to a single receiver or MAY include possibly a very large number of receivers. FCAST is specified to support two forms of operation.

1. FCAST/ALC: where the FCAST application is meant to run on top of the ALC/LCT reliable multicast transport protocol, and
2. FCAST/NORM: where the FCAST application is meant to run on top of the NORM reliable multicast transport protocol.

This specification is designed such that both forms of operation share as much commonality as possible.

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. The transmission 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 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).

FCAST is designed to be as self-sufficient as possible, in particular in the way object meta-data is attached to object data content. However, for some uses, meta-data MAY also be communicated by an out-of-band mechanism that is out of the scope of the present document.

4.2 Meta-Data Transmission

FCAST usually carries meta-data elements by prepending them to the object it refers to. As a result, a compound object is created that is composed of a header followed by the original object content. 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.

Attaching the meta-data to the object is an efficient solution, since it guaranties that meta-data be received along with the associated object, and it allows the transport of the meta-data to benefit from any transport-layer FEC erasure protection of the compound object. However a limit of this scheme, as such, is that a client does not know the meta-data of an object before it begins receiving the object and perhaps not until decoding the object completely depending upon the transport protocol used and its particular FEC code type and parameters.

However, this solution can be associated to another in-band (e.g., via NORM INFO messages, Section 4.10) or out-of-band signaling mechanism (Appendix A) in order to carry the whole meta-data (or a subset of it) possibly ahead of time.

4.3 Meta-Data Content

The meta-data associated to an object can be composed of, but are not limited to:

- Content-Location: the URI of the object, which gives the name and location of the object;
- Content-Type: the MIME type of the object;
- Content-Length: the size of the initial object, before any content encoding (if any). Note that this content length does not include the meta-data nor the header of the compound object;
- Content-Encoding: the optional encoding of the object performed by FCAST;
- Content-MD5: the MD5 message digest of the object in order to check its integrity. Note that this digest is meant to protect from transmission and processing errors, not from deliberate attacks by an intelligent attacker. Note also that this digest only protects the object, not the header, and therefore not the meta-data;
- a digital signature for this object;

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. When a file is split into several objects by FCAST, the meta-data includes:

- Fcast_Obj_Slice_Nb: the total number of slices. A value strictly greater than 1 indicates that this object is the result of a split of the original object;
- Fcast_Obj_Slice_Idx: the slice index (in the [0 .. SliceNb[interval);
- Fcast_Obj_Slice_Offset: the offset at which this slice starts within the original object;

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

- TOI: the Transmission Object Identifier (TOI) of the object, in order to enable a receiver to easily associate the meta-data to the object for which he receives packets;
- FEC Object Transmission Information (FEC OTI). In this case the FCAST sender does not need to use the optional EXT_FTI mechanism of ALC or NORM protocols.

4.4 Carousel Transmission

A set of FCAST compound objects scheduled for transmission are considered a logical "Carousel". A single "Carousel Instance" is comprised of a fixed set of compound objects. Whenever the FCAST application needs to add new objects to or remove old objects from the transmission set, a new Carousel Instance is defined since the set of compound objects changes.

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 that others during a cycle, and/or benefit from higher FEC protection than others. This can be the case for instance of the CIO objects (Section 4.5). For some FCAST usage (e.g., a unidirectional "push" mode), a Carousel Instance may have only a single transmission cycle. In other cases there may be a large number of transmission cycles (e.g., such as an "on-demand" mode where objects are made available for download during a possibly very long period of time).

4.5 Carousel Instance Object

The FCAST sender MAY transmit an OPTIONAL Carousel Instance Object (CIO). The CIO carries a list of the compound objects that are part of a given Carousel Instance. The objects are listed using their respective Transmission Object Identifiers (TOI). There is no reserved TOI value for the CIO, since this object is regarded by ALC/LCT or NORM as a standard object. The nature of this object is indicated by means of a specific meta-data field so that it can be recognized and processed by the FCAST application as needed.

The CIO includes a Carousel Instance ID (CID) that identifies the Carousel Instance. The CIO includes a "Complete" flag that is used to indicate that no other modification to the enclosed list will be done in the future. However the CIO does not describe the objects themselves (i.e., there is no meta-data). Any objects that are not incuded in the CIO list MUST NOT be considered as part of the current Carousel Instance, even if they were part of any previous Carousel Instances.

Note use of a CIO is NOT mandatory. If it is not used, then the clients will progressively learn what files are part of the carousel instance by receiving ALC or NORM packets with new TOIs. However use of the CIO has several benefits:

- Receivers know when they can leave the session, i.e., when they have received all the objects that are part of the delivery session, thanks to the "Complete" flag;
- 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 CIO. This solution is more robust than the "Close Object flag (B)" of ALC/LCT since a client with an intermittent connectivity might loose 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 CIO provides an additional means for receivers to learn of objects for which it is futile to request repair

The decision of whether a CIO should be used, as well as how often and when it should be sent, is left to the sender and depends on many parameters, including the target use case and the session dynamics. In case of an FCAST session in a strictly unidirectional, proactive transmission mode (i.e., "push" mode), the CIO SHOULD be sent before the objects (and repeated periodically during the Carousel Instance transmission to enable late receivers to catch up, if this is desired). In case of a highly dynamic FCAST session, a CIO will probably be sent at the beginning of each new carousel instance, and then periodically. The period of CIO repetition depends on the desired maximum latency that could be experienced by late receivers who joined the FCAST session in the middle of a carousel instance transmission cycle, and therefore missed the initial CIO transmission. These operational aspects are out of the scope of the present document.

4.6 FCAST Sender Behavior

The following operations take place at a sender:

1. The user (or another application) selects a set of objects (e.g., files) to deliver and submits them to the FCAST application. The user also specifies how many times each object should be sent in this carousel instance. Said differently, if objects have similar lengths, assigning them a different number of transmissions leads to define different transmission priorities to each of them;
2. For each object, FCAST creates the compound object and registers this latter in the carousel instance.
3. The user then informs FCAST when all the objects of the set have been submitted. If no new object will be submitted later to FCAST (i.e., if the session's content is now complete), the user SHOULD also provide FCAST this information;
4. At this point, the FCAST application knows the full list of compound objects that are part of the carousel instance and can define a transmission schedule of these objects. This specification does not mandate any transmission schedule scheme. This is left to the developer within the provisions of the underlying ALC or NORM protocol used.
5. The FCAST application will create a CIO as needed. If no new object will be submitted, then the sender includes the "Complete" keyword in any CIO created to inform the receivers that no object in addition to the ones specified in this carousel instance will be sent. While this specification RECOMMENDS that the sender SHOULD send the CIO prior to the transmission of the associated objects, it does not mandate if or how the CIO transmission should be repeated during the associated carousel instance. This is left to the developer;
6. The FCAST application then starts the carousel transmission, for the number of cycles specified (which might be infinite), taking into account the possible transmission specificities of each object. The 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 opposite wants to remove old objects from the carousel. In that case a new carousel instance must be created.

   Then continue at Step 1 above.

   **** Editor's note: Question: should a sender use a CIO with an empty list of objects when he has reached the desired number of cycles? Do we say "SHOULD" or "MUST"? Possible wording (to discuss): When the desired number of transmission cycles has been reached, after a small duration during which the user did not submit any new object and did not tell FCAST to add some more transmission cycles, the sender SHOULD create and send a CIO with an empty list of objects. However, it should be noted that doing so is sub-optimal if some of the objects are to be sent once again latter on, since the receiver will destroy those objects that have not been totally decoded upon receiving this CIO.*

4.7 FCAST Receiver Behavior

The following operations take place at an FCAST receiver:

1. The receiver joins the session and collects symbols;

2. As the header portion of compound objects are received (which may be received before the entire object is received with some ALC/NORM transport configurations), the receiver processes the meta-data and may choose 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 CIO is received, which is indicated by the 'I' flag set in the compound object header, the receiver decodes the CIO, and retrieves the list of objects that are part of the current carousel instance. This list is used to remove objects sent in a previous carousel instance that might not have been totally decoded. This list is also used to set up a new filter, since all the received content for objects from the given sender that are not part of this list SHOULD be immediately discarded;
5. **** Editor's note: there is an exception: the TOI for the following CIO. This TOI is perhaps not yet known, but it MUST NOT be filtered. This is where having a floating TOI for CIOs makes things a bit more complex. How to address this problem is TBD.*
6. When a receiver has received a CIO 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;

4.8 FCAST Object Identification

FCAST objects are directly associated with the object-based transport service that the ALC and NORM protocols provide. In each of these protocols, messages containing transport object content are labeled with a numeric transport object identifier (i.e., the ALC TOI and the NORM *NormTransportId*). For purposes of this document, this identifier in either case (ALC or NORM) is referred to as the TOI. The FCAST Compound Object Header meta-data can include an attribute that identifies the given object's TOI. Additionally, the CIO lists objects for the applicable Carousel Instance by using the TOI.

In both NORM and ALC, it is possible that the transport identification space may eventually wrap for very long-lived sessions. 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. Thus, when an updated object set for a new Carousel Instance transport identifiers that exceed one-half of the TOI sequence space (or otherwise exceed the sender repair window capability in the case of NORM) it may be necessary to re-enqueue old objects within the Carousel with new TOI to stay within transport identifier limits. To allow receivers to properly combine new transport symbols for any olders objects with newly-assigned TOIs to achieve reliable transfer, a mechanism is required to equate the object(s) with new TOI with the older object TOI. *This mechanism is TBD.*

**** Editor's note: Perhaps a way to disambiguate possible wrapping of TOI is by concatenation of the Carousel Instance Id and TOI? And also provide a mechanism to equate an object with a new TOI in a new Carousel Instance with an older TOI in an older Carousel Instance if it represents the same content. This way the transport object id could "move forward" as needed and receivers could possibly combine symbols from the new transmission with the older content. Vincent had a scheme that partially addressed this notion in an email.*

4.9 FCAST/ALC Additional Specificities

There are no additional details or options for FCAST/ALC operation.

4.10 FCAST/NORM Additional Specificities

The NORM Protocol provides a few additional capabilities that can be used to specifically support FCAST operation:

1. The NORM_INFO message for conveying "out-of-band" content with respect to a given transport object MAY be used to provide the FCAST compound object header and meta-data to the receiver group. NORM's NACK-based repair request signaling allows for an object's NORM_INFO content to be requested separately and more quickly than the object's "in-band" data content that is typically encoded using FEC. However, the

limitation here is that the Compound Object Header and its meta-data MUST fit within the byte size limit defined by the NORM sender's configured "segment size" (typically a little less than the network MTU).

2. The NORM_CMD(SQUELCH) messages used by the NORM protocol sender to inform receivers of objects that have been canceled when receivers make repair requests for such invalid objects.
3. NORM also supports an optional positive acknowledgment mechanism that can be used for small-scale multicast receiver group sizes. Also, it may be possible in some cases for the sender to infer, after some period without receiving NACKs at the end of its transmission that the receiver set has fully received the transmitted content. In particular, if the sender completes its end-of-transmission series of NORM_CMD(FLUSH) messages without receiving repair requests from the group, it may have some assurance that the receiver set has received the content prior to that point.

Receivers automatically learn of the availability of NORM_INFO for a given object from a flag in the NORM_DATA message header. When NORM_INFO is used for FCAST/NORM operation, the NORM_INFO content MUST contain the FCAST Compound Object Header and meta-data for that object. In this case, the data content portion of the NORM transport object is the original application object. When NORM_INFO is not used for a given sender object (i.e., the corresponding NORM_DATA header flag is not set), the NORM transport object data content sent MUST contain the FCAST Compound Object Header unless this information is signaled by another means (out of scope of this document) prior to the carousel transmission.

It should also be noted that the NORM_INFO message header may carry the EXT_FTI extension. The reliable delivery of the NORM_INFO content allows the individual objects' FEC Transmission Information to be provided to the receivers without burdening every packet (i.e. NORM_DATA messages) with this additional, but important, content.

5. FCAST Specifications

This section details the technical aspects of FCAST.

5.1 Compound Object Header Format

In an FCAST session, its compound objects are constructed by prepending the Compound Object Header including any meta-data content as shown in Figure 1 before the original object data content.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|rsvd |I|MDE|MDF|             Header Length                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|     Object Meta-Data Content (optional, variable length)      |
|                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |      Padding (optional)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.      Object Data Content (optional, variable length)          .
.                                                               .
.                                                               .
```

Figure 1: Compound Object Header with Meta-Data

The Compound Object Header fields are:

Field	Description
I	1-bit field that, when set to 1, indicates the object is a Carousel Instance Object (CIO). When set to 0, this field indicates that the transported object is a standard object.
Meta-Data Encoding (MDEnc)	2-bit field that defines the optional encoding of the Object Meta-Data Content field (see Section 7). A plain text encoding is the default encoding and is associated value 0. A gzip encoding MAY be supported and is associated to value 1. Other encodings MAY be defined in the future.
Meta-Data Format (MDFmt)	2-bit field that defines the format of the object meta-data (see Section 7). An HTTP/1.1 metainformation format [8] MUST be supported and is associated to value 0. Other formats (e.g., XML) MAY be defined in the future.
Header Length	24-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 4 means that there is no meta-data included. When this size is not multiple to 32 bits words, padding is added. It should be noted that the meta-data field maximum size is equal to 2^24 - 4 bytes.
Object Meta-Data	Optional, variable length field that contains the meta-data associated to the object, either in plain text or encoded, as specified by the MDEnc field. The Meta-Data is NULL-terminated plain text of the "TYPE" ":" "VALUE" "<CR-LF>" format used in HTTP/1.1 for metainformation [8]. 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.
Padding	Optional, variable length field of zero-value bytes to align start of object data content to 32-bit boundary. Padding is only used when the header length value, in bytes, is not multiple of 4.
Object Data Content	Data content of original object represented by this Compound Object. Note that the length of this content is the transported object size minus the Compound Object Header Length

**** Editor's note: Should we add a checksum to protect the header itself? Since meta-data do not use an XML encoding, there is no way to digitally sign it to check its integrity. A checksum could offer some integrity guaranty (not security of course).*

5.2 Carousel Instance Object (CIO) Format

The format of the CIO, which is a particular compound object, is given in Figure 2. Because the CIO is transmitted as a special compound object, the following CIO-specific meta-data entry is defined:

- Fcast_CIO_complete: when set to 1, it indicates that no new objects in addition to the ones whose TOI are specified in this CIO, or the ones that have been specified in the previous CIO(s), will be generated;
- Fcast_CIO_ID: this value identifies the carousel instance. It starts from 0 and is incremented by 1 for each new carousel instance. This entry is not mandatory since the TOI numbering of the compound objects carrying a CIO can be used to identify the latest CIO instance. However, this value can be useful to detect possible gaps in the carousel instances, for instance caused by long disconnection periods. It can also be usefull to avoid problems when TOI wrapping to 0 takes place.

Additionaly, the following standard meta-data entries are often used:

- Content-Encoding: the optional encoding of the CIO object, by FCAST. For instance:

```
Content-Encoding: gzip
```

  indicates that the Object List field has been encoded with gzip [7]. When set to 0, this flag indicates the the Object List field is plain text. The support of gzip encoding is MANDATORY, both for an FCAST sender and for an FCAST receiver

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ^
|rsvd |1|MDE|MDF|              Header Length                    | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ h
|                                                               | d
|     Object Meta-Data Content (optional, variable length)      | r
|                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                               |      Padding (optional)       | v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ^
.                                                               . |
.                Object List (variable length)                  . O
.                                                               . b
.                                               +-+-+-+-+-+-+-+-+ j
.                                               |                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                 v
```

Figure 2: Carousel Instance Object Format

The CIO fields are:

Field	Description
Object List	List of TOIs included in the current carousel instance, in an exhaustive way. This list, whose format is defined below, can be either in plain text (if Z is not set) or gzip'ed (if Z is set). An empty list (0 length field) indicates that the current carousel instance does not include any object.

The non-encoded (i.e., plain text) Object List is a NULL-terminated, ASCII string containing the list of TOIs included in the current carousel instance, specified either as the individual TOIs of each object, or as TOI spans, or combinations of these. 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. We further require that TOI_a be strictly inferior to TOI_b. The ABNF specification is the following:

```
cio-list   =  *(list-elem *( "," list-elem))
list-elem  =  toi-value / toi-range
toi-value  =  1*DIGIT
toi-range  =  toi-value "-" toi-value
```

```
              ; additionally, the first toi-value MUST be
              ; strictly inferior to the second toi-value
DIGIT      =  %x30-39
              ; a digit between 0 and 9, inclusive
```

It is RECOMMENDED, for processing reasons, 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 RECOMMENDED, for processing reasons, that a given TOI value NOT be included mutiple times in the list.

6. Security Considerations

6.1 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 FCAST) or
- the content itself (e.g., to corrupt the objects 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 CIO, the object meta-data, 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.

6.2 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 object (e.g., in case of a non-free content) 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, which is a kind of Denial of Service (DoS)).

6.2.1 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 [13]). If confidentiality is a concern, it is RECOMMENDED that one of these solutions be used.

6.2.2 Object Corruption

Protection against corruptions (e.g., in case of 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:

- at the object level, the object can be digitally signed (with public key cryptography), for instance by using RSASSA-PKCS1-v1_5 [10]. 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;
- at the packet level, each packet can be digitally signed. A major limitation is the high computational and transmission overheads that this solution requires (unless perhaps if Elliptic Curve Cryptography (ECC) is used). 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;
- at the packet level, a Group Message Authentication Code (MAC) [9] scheme can be used, for instance by using HMAC-SHA-1 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, Timed Efficient Stream Loss-Tolerant Authentication (TESLA) [11] 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/AH [12] (possibly associated to IPSec/ ESP) can be used to protect all the packets being exchanged in a session.

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.

6.3 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 FCAST CIO,
- the attack can target the meta-data of an object,
- the attack can target the ALC/LCT parameters, carried within the LCT header or
- the attack can target the FCAST associated building blocks.

The latter one is particularly true with the multiple rate congestion control protocol which may be required.

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

6.3.1 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 archived is outside the scope of this document since this session description is usually carried out-of-band.

6.3.2 Attacks Against the FCAST CIO

Corrupting the FCAST CIO is one way to create a Denial of Service attack. For example, the attacker removes legitimate object TOIs from the list.

It is therefore RECOMMENDED that measures be taken to guarantee the integrity and to check the sender's identity of the CIO. To that purpose, one of the counter-measures mentioned above (Section 6.2.2) SHOULD be used. These measures will either be applied on a packet level, or globally over the whole CIO object. When there is no packet level integrity verification scheme, it is RECOMMENDED to digitally sign the CIO.

6.3.3 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 6.2.2) 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.

6.3.4 Attacks Against the ALC/LCT 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 one can lead the receiver to prematurely close the session. Similarly, sending forged ALC packets with the Close Object flag (B) set 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 each ALC packet received. To that purpose, one of the counter-measures mentioned above (Section 6.2.2) SHOULD be used.

6.3.5 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 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. 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 FCAST/ALC, it is also RECOMMENDED that a packet level authentication scheme be used, as explained in Section 6.2.2. Some of them, like TESLA, only provide a delayed authentication service, whereas congestion control requires a rapid reaction. It is therefore RECOMMENDED [2] 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 since no congestion actually occurred. 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 reduce the probability of this attack.

6.4 Other Security Considerations

Lastly, we note that the security considerations that apply to, and are described in, ALC [2], LCT [3] and FEC [4] 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.

7. IANA Considerations

This document requires a IANA registration for the following attributes:

Object meta-data format (MDFmt): All implementations MUST support format 0 (default).

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

Object Meta-Data Encoding (MDENC): All implementations MUST support value 0 (default).

Name	Value
plain text	0 (default)
gzip	1

8. Acknowledgments

The authors would like to thank Toni Paila and Rod Walsh for their challenging comments throughout the design of FCAST. The authors are grateful to the authors of [6] for specifying the first version of FCAST/ALC.

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] Luby, M., Watson, M., and L. Vicisano, "Asynchronous Layered Coding (ALC) Protocol Instantiation", Work in Progress, November 2007.

[3] Luby, M., Watson, M., and L. Vicisano, "Layered Coding Transport (LCT) Building Block", Work in Progress, February 2007.

[4] Adamson, B., Bormann, C., Handley, M., and J. Macker, "Negative-acknowledgment (NACK)-Oriented Reliable Multicast (NORM) Protocol", Work in Progress, May 2008.

[5] Paila, T., Walsh, R., Luby, M., Lehtonen, R., and V. Roca, "FLUTE - File Delivery over Unidirectional Transport", Work in Progress, October 2007.

9.2 Informative References

[6] Luby, M., Gemmell, G., Vicisano, L., Crowcroft, J., and B. Lueckenhoff, "Asynchronous Layered Coding: a Scalable Reliable Multicast Protocol", draft-ietf-rmt-pi-alc-00.txt, March 2000.

[7] Deutsch, P., Gailly, J-L., Adler, M., Deutsch, L.P., and G. Randers-Pehrson, "GZIP file format specification version 4.3", RFC 1952, May 1996.

[8] Fielding, R., Gettys, J., Mogul, J., Nielsen, H., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, January 1997.

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

[10] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

[11] Perrig, A., Song, D., Canetti, R., Tygar, J.D., and B. Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction", RFC 4082, June 2005.

[12] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

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

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/~roca/

Brian Adamson
Naval Research Laboratory
Washington, DC, 20375
USA
EMail: adamson@itd.nrl.navy.mil
URI: http://cs.itd.nrl.navy.mil

A. FCAST in practice

This section discusses how FCAST/ALC and FCAST/NORM can be used in practise.

Out-of-band transmission of the object meta-data: In some use-cases, the meta-data (or a subset of them) will be communicated to the receivers by means of an out-of-band mechanism. In some use-cases, this out-of-band mechanism can itself be a dedicated FCAST session. It is also possible that the TOI of each object be known in advance (e.g., the TOI can be reserved). When this is the case, sending this TOI along with the meta-data makes it possible for a receiver to know in advance the meta-data associated to each object, which enables the end-user (or the terminal when a set of preferences or selection criteria have been filled) to filter the incoming packets and discard those associated to a non-desired object.

SDP: The FCAST session parameters can be communicated in numerous ways. One of them consists in using the Session Description Protocol (SDP) [REF].

RoHC: In some situations, for instance in low-bandwidth wireless environments, it can be desirable to compress the various protocol headers (in our case IP, UDP, and possibly ALC/LCT) in a robust way. The Robust Header Compression (RoHC) family of compression schemes [REF] can be used to that purpose.

Object aggregation:

Session-level protocol: It is often desirable to use FCAST as a robust transport solution under the control of a session level protocol. This session level protocol can for instance have a certain knowledge of the set of receivers and perform receiver management operations. Examples of such operations include but are not limited to accepting new receivers in the group or performing cleaning operations after the departure of a receiver, managing security aspects like group keying or performing AAA. This session level protocol can also provide a higher level reliability framework, in order to make sure that each active receiver has received correctly a given object, and in case this is not the case, it can launch a recovery mechanism that might sometimes imply a direct point-to-point retransmission of missing symbols, or when the number of receivers concerned is higher than a certain threshold, a new multicast recovery session.

B. FCAST Examples

Figure 3 shows a compound object:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  0  |0| 0 | 0 |               37                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.       meta-data ASCII null terminated string (33 bytes)       .
.                                                               .
+               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               |                    padding                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                         Object data                           .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Compound Object Example

where the meta-data ASCII string, in HTTP/1.1 meta-information format 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 (Z=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.

Figure 4 shows a compound object without any meta-data:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  0  |0| 0 | 0 |               4                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                         Object data                           .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Compound Object Example with no Meta-Data.

The fact there is no meta-data is indicated by the value 3 of the Header Length field.

Figure 5 shows an example CIO object, in the case of a static FCAST session, i.e., a session where the set of objects is set once and for all.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  0  |1| 0 | 0 |               4                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                Object List string                             .
.                                                               .
.                                               +-+-+-+-+-+-+-+-+
.                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

The object list contains the following string:

```
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 CIO. The session being static the sender did not feel the necessity to carry a Carousel Instance ID meta-data.

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).

[1] mailto:ietf-ipr@ietf.org