RMFP: A Reliable Multicast Framing Protocol

Status of this Memo

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

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

To learn the current status of any Internet-Draft, please check the '1id-abstracts.txt' listing contained in the Internet-Drafts Shadow Directories on ds.internic.net (US East Coast), nic.nordu.net (Europe), ftp.isi.edu (US West Coast), or munnari.oz.au (Pacific Rim).

The distribution of this memo is unlimited. It is filed as <draft-crowcroft-rmfp-01.txt>, and expires 15 Sept, 1997. Please send comments to the authors.

1. Introduction

There has been considerable interest in reliable multicast, and a number of reliable multicast transport applications and systems have been built in the past years.

Reliable multicast transport is considerably more complex than reliable unicast. It is generally difficult to build a generic reliable transport protocol for multicast, much as TCP is a generic transport protocol for unicast, since different applications often have very different reliability and latency requirements, and modes of operation.

In this document we propose a framing protocol for reliable multicast application design - Reliable Multicast Framing Protocol (RMFP). The purpose of RMFP is to provide a common framework upon which a set of reliable multicast applications can be built and share common functionalities where exist.

RMFP follows the principles of application level framing and integrated layer processing []. It is designed to be integrated into an application rather than being implemented as a separate layer. RMFP includes extension mechanisms to allow modifications and additions required by individual applications. A complete specification of RMFP for a particular application will require a profile document which specifies extensions and modifications for the application.

RMFP runs over UDP and itself does not provide any reliability (or functionality in a larger extend). Reliability and other protocol functionalities will be defined in specific profiles and implemented as part of the reliable multicast applications.

The philosophy of RMFP is in many respects similar to the one
of RTP. However, we believe that using RTP for reliable multicast is not a right approach and will not lead to a clean application design.

2. RMFP Data Packet Format

RMFT data packet has the following header format:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| V |P|R|F|S|E|X| PAYLOAD TYPE |             LENGTH            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          SOURCE ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        SEQUENCE NUMBER        |           OBJECT ID           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
|                             OFFSET                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  PROFILE-SPECIFIC EXTENSIONS                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Version (V): 2 bits
This field identifies the version of RMFP.

Padding (P): 1 bit
If the padding bit is set, the packet contains one or more additional padding octets at the end which are not part of the payload. The last octet of the padding contains a count of how many octets should be ignored.

Retransmission (R): 1 bit
This bit, when set, indicates that the data is being retransmitted.

Forward Error Correction (F): 1 bit
This bit, when set, indicates that FEC is used. The exact format of FEC is determined by the payload type and its profile.

Start of Object (S): 1 bit
This bit, when set, indicates that the data in the packet is the start of an object.

End of Object (E): 1 bit
This bit, when set, indicates that the data in the packet is the end of an object.

Exceptional Handling (X): 1 bit
This bit, when set, indicates that the packet may be handled in an unusual way. The use of the bit is determined by the payload type and its profile.

Payload Type: 8 bits
This field identifies the format of the payload and determines its interpretation of profile-specific fields. A profile will be defined for each payload type.

Length: 16 bits
This field identifies the length of the packet in 32 bits minus one, including the header and any padding. It can be used to combine several RMFP packets into one UDP packet.
When several RMFP packets (data packets + retransmission packets) are concatenated within one UDP packet, the RMFP data packets (R=0) should all be placed at the beginning of the UDP packet so that receivers that do not encounter losses can just drop the tail of the retransmitted packets without processing it.

Source ID: 32 bits
This field identifies the source. It is generated randomly similar to the SSRC field in RTP.

Sequence Number: 16 bits
The sequence number is a packet count. It increments it by one for each data packet sent. It can be used to detect packet losses (including both data packet and retransmitted packets), and calculate loss rates.

Object ID: 16 bits
This field identifies the object carried in the packet. Object ID provides a handler to the internal structure of the data, and it can be used to multiplex several objects into a single session. For example, when a sender transmits multiple HTML files to a group of receivers, the receivers can use the Object ID to find out which object a packet belongs to and to start processing immediately. Without the Object ID, the receivers may have to wait until all previous packet are received in order.

For some applications that do not have multiple objects in one session such as "mcast talk", the Object ID may be used for some logical division. For example, a multicast talk may divide the session into a number of intervals. When a new member joins in, it will only get the historical data in the previous one (or several) intervals. For small objects (e.g., one per packet), one may use the Object ID for retransmission request instead of byte range.

Offset: 64 bits
This field identifies the byte position of the data relative to the first byte of the session. A sender may not send data in the order that an application wants to receive. For example, a file transfer program may divide a file into blocks and then send the blocks in a random order. The offset field can be used to assemble the received data into right order.

Profile-Specific Extensions: Variable
The fixed data header covers most of the functions in common across all applications. However, profile-specific modifications and extensions can be defined in the profile for each payload type.

3. RMFP Control Packet Format
RMFP control packets include sender’s report packets and receiver’s report packets.

Sender’s Report Packet
Sender’s report is sent periodically by the sender about the data transmitted in the session.

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
+-------------------------------+-----------------------------+
| V |P| SR TYPE | PAYLOAD TYPE | LENGTH |
+-------------------------------+-----------------------------+
| SOURCE ID                      |
Version(V): 2 bits
This field identifies the version number.

Padding(P): 1 bit
If the padding bit is set, the packet contains one or more additional padding octets at the end which are not part of the payload. The last octet of the padding contains a count of how many octets should be ignored.

SR Type: 5 bits
The SR Type field determines the interpretation of the profile-specific extensions.

Payload Type: 8 bits
This field is set to xxx for Sender’s Report Packets

Header Length: 16 bits
This field specifies the length of the header.

Source ID: 32 bits
This field identifies the source of the sender.

Object ID: 16 bits
The Object ID that is associated with the sequence number field.

Highest Sequence Number: 16 bits
This field indicates the data sent at the time the report is sent.

Receiver’s Report Packet
Receiver’s report is periodically sent by the receivers to give feedback on congestion and packet losses.

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 2
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| V |P| RR TYPE | PAYLOAD TYPE | LENGTH |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| SOURCE ID |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| FRACTION LOST | RESERVED | HIGHEST SEQUENCE NUMBER |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| PROFILE-SPECIFIC EXTENSIONS |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

Version(V): 2 bits
This field identifies the version number.

Padding(P): 1 bit
If the padding bit is set, the packet contains one or more additional padding octets at the end which are not part of the payload. The last octet of the padding contains a count of how many octets should be ignored.
RR Type: 5 bits
   The RR Type field determines the interpretation of the
   profile-specific extensions.

Payload Type: 8 bits
   This field is set to xxx for Receiver’s Report Packets

Header Length: 16 bits
   This field specifies the length of the header.

Source ID: 32 bits
   This field identifies the source of the report

Fraction Lost: 8 bits
   The fraction of packets lost since last Sender’s report,
   expressed as a fixed point number with the binary
   point at the left edge of the field. Fraction lost is the
   loss rate seen by the receiver. The information may be
   used for congestion control, error recovery purpose by the
   sender.

Highest Sequence Number: 16 bits
   This field indicates the highest sequence number received so far.

4. Authors’s Addresses

J Crowcroft, Zheng Wang
(j.crowcroft, z.wang)@cs.ucl.ac.uk
Department of Computer Science
University College London
Gower Street
London
WC1E 6BT

Atanu Ghosh
atanu@socs.uts.edu.au
School of Computing Sciences
University of Technology
Sydney
PO Box 123, Broadway
NSW 2007
Australia

Christophe Diot
Christophe.Diot@sophia.inria.fr
INRIA
Sophia Antipolis, 2004
route des Lucioles
BP93 06902
France