INTERNET-DRAFT 28 February 2002
Internet Engineering Task Force Expires: 28 August 2002
Audio/Video Transport Working Group

Timur Friedman, LiP6
Ramon Caceres, ShieldIP
Kevin Almeroth, UCSB
Kamil Sarac, UCSB

RTCP Reporting Extensions

draft-friedman-avt-rtcp-report-extns-02.txt

Status of this Memo

Copyright Notice

Abstract

This document defines the XR (extended report) RTCP packet type, for carrying information beyond that which is contained in the SR (sender report) and RR (receiver report) packets that are defined in the RTP specification. Within their "reception report blocks", SR and RR packets are limited to reporting six specified statistics on any given data source. This document describes how other information can be reported in "extended report blocks" that are contained within an

XR packet. Some specific block formats are provided here. For other formats that may be defined as the need arises, this document specifies a simple framework that they must adhere to.

1. Introduction

This document defines the XR (extended report) RTCP packet type for RTCP, the control portion of RTP [7]. The definition consists of three parts. First, Section 2 of this document defines a general packet framework capable of including a number of different "extended report blocks." Second, Section 3 defines the general format for such blocks. Third, Section 4 defines a number of such blocks.

The extended report blocks convey information beyond that which is already contained in the reception report blocks of RTCP's SR or RR packets. For example, while a reception report block contains an average loss rate field, an application might opt to use an extended report block that details exactly which packets were received and which were lost.

The framework for these blocks is minimal: only a type field and a length field are required. The purpose is to maintain flexibility and to keep overhead low. While some specific block formats are provided here, others may be defined as the need arises.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [3] and indicate requirement levels for compliant RTP implementations.

2. XR Packet Format

The XR packet consists of a header of two 32-bit words, followed by a number, possibly zero, of extended report blocks. This packet format has been deployed, as described in [4] and [1], as an RTCP APP (application-specific) packet. The XR packet header is identical to that of the APP packet, with the name field removed and the subtype field cleared.

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|reserved |   PT=XP=205   |             length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           SSRC/CSRC                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                         report blocks                         :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

version (V): 2 bits

Identifies the version of RTP. This specification applies to RTP ver¡sion two (2).

padding (P): 1 bit

If the padding bit is set, this individual RTCP packet contains some additional padding octets at the end that are not part of the control information but are included in the length field. The last octet of the padding is a count of how many padding octets should be ignored, including itself (it will be a multiple of four). A full description of padding in RTCP packets may be found in the RTP specification.

reserved: 5 bits

This field is reserved for future definition. The bits in this field MUST be set to zero unless otherwise defined.

packet type (PT): 8 bits

Contains the constant 205 to identify this as an RTCP XR packet. This is a proposed value, pending assignment of a number by the Internet Assigned Numbers Authority [6].

length: 16 bits

The length of this RTCP packet in 32-bit words minus one, including the header and any padding. (The offset of one makes zero a valid length and avoids a possible infinite loop in scanning a compound RTCP packet, while counting 32-bit words avoids a validity check for a multiple of 4.)

SSRC: 32 bits

The synchronization source identifier for the originator of this XR packet.

report blocks: variable length.

Zero or more extended report blocks. The blocks MUST be a multiple of 32 bits long. They MAY be zero bits long.

3. Extended Report Block Framework

Extended report blocks MUST be stacked, one after the other, at the end of an XR packet. An individual block's length MUST be a multiple of 4 octets. The XR header's length field MUST describe the total length of the packet, including these extended report blocks.

Each block has block type and length fields that facilitate parsing. A receiving application can demultiplex the blocks based upon their type, and can use the length information to locate each successive block, even in the presence of block types it does not recognize.

An extended report block has the following format:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      BT       | type-specific |             length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                      type-specific data                       :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

block type (BT): 8 bits
   Identifies the specific block format.

type-specific: 8 bits
   The use of these bits is defined by the particular block type.

length: 16 bits
   The length of this report block in 32-bit words minus one, including the header.

type-specific data: variable length
   This MUST be a multiple of 32 bits long. It MAY be zero bits long.

4. Specific Extended Report Blocks

This section defines five extended report blocks: an experimental block type and block types for losses, duplicates, timestamps, and detailed statistics. Other block types MAY be defined in the future. Any such definition MUST include block type numbers assigned by the Internet Assigned Numbers Authority [6].

4.1 Experimental Block

This type MUST be used for extended report block types that have not been standardized. In addition to the standard type and length fields, it includes a 32 bit name field that serves to distinguish one experimental block type from another.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      BT=0     |  app-specific |             length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          name (ASCII)                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                  application-specific data                    :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

block type (BT): 8 bits
   Block type 0 identifies this as an experimental block.

app-specific: 8 bits
   The use of these bits is defined by the application that uses this block.

length: 16 bits
   The length of this report block in 32-bit words minus one, including the header.

name: 4 octets
   A name chosen by the person definining the experimental block to be unique with respect to other experimental blocks the application might receive.

application-specific data: variable length.
   This MUST be a multiple of 32 bits long. It MAY be zero bits long.

4.2 Loss RLE Block

With this block type, a boolean trace of lost and received packets can be conveyed in compressed form using run length encoding. This block type has been deployed on the internet, as part of an RTCP APP (application-specific) packet, as described in [4] and [1].

Caution SHOULD be used in sending such blocks because, even with compression, they can easily consume bandwidth out of proportion with normal RTCP packets.

Each block reports on a single source, identified by its SSRC. The receiver that is supplying the report is identified in the header of the RTCP packet.

The beginning and ending sequence numbers for the trace are specified in the block, the ending sequence number being the last sequence number in the trace plus one. The last sequence number in the trace MAY or may not be the sequence number reported on accompanying SR or RR packets, depending on the needs of the application.

The encoding itself consists of a series of 16 bit chunks. Each chunk either specifies a run length or a bit vector, or, if the trace otherwise encodes into an odd number of chunks, MUST be a terminating null chunk used to round out the block to a 32 bit word boundary.

The mapping from a sequence of lost and received packets into a sequence of chunks is not unique and is left to the application. A run length chunk can describe runs of between 1 and 16,383 packet losses or receipts whereas a bit vector chunk can describe a sequence of 15 packet losses and receipts. It is RECOMMENDED that the description of run lengths of 14 or shorter be subsumed into bit vector chunks, for purposes of brevity.

A bit vector chunk MAY purport to contain information on packets at or beyond the ending sequence number. Any such purported information MUST be ignored.

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     BT=17     | rsvd. |   T   |         block length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         SSRC of source                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          begin_seq            |             end_seq           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          chunk 1              |             chunk 2           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          chunk n-1            |             chunk n           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

block type (BT): 8 bits
   A Loss RLE block is identified by the constant 17 = 0x11.

rsvd.: 4 bits
This field is reserved for future definition. The bits in this field MUST be set to zero unless otherwise defined.

thinning (T): 4 bits
The amount of thinning performed on the sequence space. Only those packets with sequence numbers 0 mod 2^T are reported on by this block. A value of 0 indicates that there is no thinning, and all packets are reported on. The maximum thinning is one packet in every 32,768 (amounting to two packets within each 16-bit sequence space).

length: 16 bits
The length of this report block in 32-bit words minus one, including the header.

begin_seq: 16 bits
The first sequence number that this block reports on.

end_seq: 16 bits
The last sequence number that this block reports on plus one.

chunk i: 16 bits
There are three chunk types: run length, bit vector, and terminating null. If the chunk is all zeroes then it is a terminating null chunk. Otherwise, the leftmost bit of the chunk determines its type: 0 for run length and 1 for bit vector.

4.2.1 Run-Length Chunk

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|C|R|        run length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

chunk type (C): 1 bit
A zero identifies this as a runlength chunk.

run type (R): 1 bit
Zero indicates a run of losses. One indicates a run of received packets.

run length: 14 bits
A value between 1 and 16,383. The value MUST not be zero (zeroes in both the run type and run length fields would make the chunk a

terminating null chunk). Run lengths of 15 or less MAY be described with a run length chunk despite the fact that they could also be described as part of a bit vector chunk.

4.2.2 Bit Vector Chunk

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|C|        bit vector           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

chunk type (C): 1 bit
   A one identifies this as a bit vector chunk.

bit vector: 15 bits
   In the bit vector, as in the run length chunk, a zero indicates a loss and a one indicates a received packet.

4.2.3 Terminating Null Chunk

This chunk is all zeroes.

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

4.3 Duplicate RLE Block

This block is identical in format to the Loss RLE Block type but car¡ ries information about individual or runs of duplicate packets. A zero indicates the presence of duplicate packets for a given sequence number, whereas a one indicates that no duplicates were received. Note that a packet loss is encoded as a one in this case.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    BT=33      |   reserved    |             length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        SSRC of source                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          begin_seq                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          end_seq                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          chunk 1              |             chunk 2           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          chunk n-1            |             chunk n           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

block type (BT): 8 bits
A Duplicate RLE block is identified by the constant 33 = 0x21.

reserved: 8 bits
This field is reserved for future definition All of the bits in this field MUST be set to zero unless otherwise defined.

length: 16 bits
The length of this report block in 32-bit words minus one, including the header.

begin_seq: 32 bits
The first sequence number that this block reports on.

end_seq: 32 bits
The last sequence number that this block reports on plus one.

chunk i: 16 bits
There are three chunk types: run length, bit vector, and terminating null. All zeroes indicates a terminating null. Otherwise, the left¡most bit of the chunk determines its type: 0 for run length and 1 for bit vector. See the descriptions of these block types in the section on the Loss RLE Block, above, for details.

4.4 Timestamp Report Block

This block carries RTCP-style timestamps for each packet in the range of packet sequence numbers. A similar caution, but more emphatic, is

made for timestamp report blocks as was made for Loss RLE Block packets. For each packet in the sequence number range, a 32 bit value MUST be recorded and sent. This could easily consume significant bandwidth. Care SHOULD be taken in the size of the sequence space over which to monitor timestamps.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    BT=48      |   reserved    |             length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        SSRC of source                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          begin_seq                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          end_seq                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         RTP timestamp (pkt n)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

block type (BT): 8 bits

A Timestamp block is identified by the constant 48 = 0x30.

reserved: 8 bits

This field is reserved for future definition. All bits in this field MUST be set to zero unless otherwise defined.

length: 16 bits

The length of this report block in 32-bit words minus one, including the header.

begin_seq: 32 bits

The first sequence number that this block reports on.

end_seq: 32 bits

The last sequence number that this block reports on plus one.

RTP timestamp: 32 bits

Corresponds to the same units as the RTP timestamp in RTP data packets. The timestamp is established upon packet arrival. It can be used to measure partial path characteristics and to model distributions for packet jitter.

4.4.1 Statistics Summary Block

This block reports detailed statistics above and beyond the informa¡ tion carried in the standard RTCP packet format. Information is recorded about lost packets, duplicate packets, jitter measurements, and TTL values. The packet contents are dependent upon a bit vector carried in the first part of the header. Not all values need to be carried in each packet. Header fields for values not carried are not included in the packet.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     BT=1      |L|D|J|T|resvd. |             length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        SSRC of source                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          begin_seq                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          end_seq                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        lost_packets                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        dup_packets                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         min_jitter                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         max_jitter                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         avg_jitter                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         dev_jitter                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   min_ttl     |   max_ttl     |   avg_ttl     |     dev_ttl   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

block type (BT): 8 bits
A Statistics Summary block is identified by the constant 1 = 0x01.

content bits (L,D,J,T): 4 bits
Bit set to 1 if packet contains (L)oss, (D)uplicate, (J)itter, and/or (T)TL report.

resvd.: 4 bits
This field is reserved for future definition. All bits in this field MUST be set to zero unless otherwise defined.

length: 16 bits
The length of this report block in 32-bit words minus one, including the header.

begin_seq: 32 bits
The first sequence number that this block reports on.

end_seq: 32 bits
The last sequence number that this block reports on plus one.

lost_packets: 32 bits
Number of lost packets in the above sequence number interval.

dup_packets: 32 bits
Number of duplicate packets in the above sequence number interval.

min_jitter: 32 bits
The minimum relative transit time between two packets in the above sequence number interval. All jitter values are measured as the difference between a packet's RTP timestamp and the reporter's clock at the time of arrival, measured in the same units.

max_jitter: 32 bits
The maximum relative transit time between two packets in the above sequence number interval.

avg_jitter: 32 bits
The average relative transit time between each two packet series in the above sequence number interval.

dev_jitter: 32 bits
The standard deviation of the relative transit time between each two packet series in the above sequence number interval.

min_ttl: 8 bits
The minimum TTL value of data packets in sequence number range.

max_ttl: 8 bits
The maximum TTL value of data packets in sequence number range.

avg_ttl: 8 bits
The average TTL value of data packets in sequence number range.

dev_ttl: 8 bits
The standard deviation of TTL values of data packets in sequence number range.

4.4.2 Receiver Timestamp Report Block

This block extends RTCP's timestamp reporting so that non-senders may also send timestamps. It recapitulates the NTP timestamp fields from the RTCP Sender Report [7, Sec. 6.3.1]. A non-sender may estimate its RTT to other participants, as proposed in [8], by sending this report block and receiving DLRR report blocks (see next section) in reply.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     BT=2      |                   reserved                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              NTP timestamp, most significant word             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             NTP timestamp, least significant word             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

block type (BT): 8 bits
A Receiver Timestamp block is identified by the constant 2 = 0x02.

reserved: 24 bits
This field is reserved for future definition. The bits in this field MUST be set to zero unless otherwise defined.

NTP timestamp: 64 bits
Indicates the wallclock time when this block was sent so that it may be used in combination with timestamps returned in DLRR report blocks from other receivers to measure round-trip propagation to those receivers. Receivers should expect that the measurement accuracy of the timestamp may be limited to far less than the resolution of the NTP timestamp. The measurement uncertainty of the timestamp is not indicated as it may not be known. A report block sender that can keep track of elapsed time but has no notion of wallclock time may use the elapsed time since joining the session instead. This is assumed to be less than 68 years, so the high bit will be zero. It is permissible to use the sampling clock to estimate elapsed wallclock time. A report sender that has no notion of wallclock or elapsed time may set the NTP timestamp to zero.

4.4.3 DLRR Report Block

This block extends RTCP's DLSR mechanism [7, Sec. 6.3.1] so that non-senders may also calculate round trip times, as proposed in [8]. It

is termed DLRR for Delay since Last Receiver Report, and may be sent in response to a Receiver Timestamp report block (see previous section) from a receiver to allow that receiver to calculate its round trip time to the respondant. The report consists of one or more 3 word sub-blocks: one sub-block per receiver report.

```
  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
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     BT=3      |   reserved    |             length            |
 +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 |                 SSRC_1 (SSRC of first receiver)               | sub-
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ block
 |                         last RR (LRR)                         |   1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                   delay since last RR (DLRR)                  |
 +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 |                 SSRC_2 (SSRC of second receiver)              | sub-
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ block
 :                               ...                             :   2
 +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

block type (BT): 8 bits
A DLRR block is identified by the constant 3 = 0x03.

reserved: 8 bits
This field is reserved for future definition. All bits in this field MUST be set to zero unless otherwise defined.

length: 16 bits
The length of this report block in 32-bit words minus one, including the header. The number of sub-blocks is length divided by three (3).

last RR timestamp (LRR): 32 bits
The middle 32 bits out of 64 in the NTP timestamp (as explained in the previous section) received as part of a Receiver Timestamp report block from participant SSRC_n. If no such block has been received, the field is set to zero.

delay since last RR (DLRR): 32 bits
The delay, expressed in units of 1/65536 seconds, between receiving the last Receiver Timestamp report block from participant SSRC_n and sending this DLRR report block. If no Receiver Timestamp report block has been received yet from SSRC_n, the DLRR field is set to zero (or the DLRR is omitted entirely). Let SSRC_r denote the receiver issuing this DLRR report block. Participant SSRC_n can

compute the round-trip propagation delay to SSRC_r by recording the time A when this Receiver Timestamp report block is received. It calculates the total round-trip time A-LSR using the last SR times¡ tamp (LSR) field, and then subtracting this field to leave the round-trip propagation delay as (A- LSR - DLSR). This is illustrated in [7, Fig. 2].

5. Acknowledgements

We thank the following people: Colin Perkins, Steve Casner, and Hen¡ ning Schulzrinne for their considered guidance; Nick Duffield for extensive ongoing contributions; Sue Moon for helping foster collabo¡ ration between the authors of this document; and Mounir Benzaid for drawing our attention to the reporting needs of MLDA.

6. Intellectual Property

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to per¡ tain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP 11 [5]. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such propri¡ etary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

7. References

[1] A. Adams, T. Bu, R. Cßceres, N.G. Duffield, T. Friedman, J. Horowitz, F. Lo Presti, S.B. Moon, V. Paxson, and D. Towsley, "The Use of End-to-End Multicast Measurements for Characterizing Internal Network Behavior," IEEE Communications Magazine, May 2000.

[2] S. Bradner, "The Internet Standards Process -- Revision 3," BCP

9, RFC 2026, IETF, October 1996.

[3] S. Bradner, "Key words for use in RFCs to indicate requirement levels," BCP 14, RFC 2119, IETF, March 1997.

[4] R. Caceres, N.G. Duffield, and T. Friedman, "Impromptu measurement infrastructures using RTP," Proc. IEEE Infocom 2002, New York, 23-27 June 2002, to appear.

[5] R. Hovey and S. Bradner, "The Organizations Involved in the IETF Standards Process," BCP 11, RFC 2028, IETF, October 1996.

[6] J. Reynolds and J. Postel, "Assigned Numbers," STD 2, RFC 1700, IETF, October 1994.

[7] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," RFC 1889, IETF, February 1996.

[8] D. Sisalem and A. Wolisz, "MLDA: A TCP-friendly Congestion Control Framework for Heterogeneous Multicast Environments", Eighth International Workshop on Quality of Service (IWQoS 2000), Pittsburgh, 5-7 June 2000.

8. Full Copyright Statement

BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MER¡ CHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

9. Authors' Addresses

Timur Friedman <timur.friedman@lip6.fr>
Laboratoire d'informatique de Paris 6
8, rue du Capitaine Scott
75015 PARIS, FRANCE

Ramon Caceres <ramon@shieldip.com>
ShieldIP, Inc.
11 West 42nd Street, 31st Floor
New York, NY 10036, USA

Kevin Almeroth <almeroth@cs.ucsb.edu>
Department of Computer Science
University of California
Santa Barbara, CA 93106, USA

Kamil Sarac <ksarac@cs.uscb.edu>
Department of Computer Science
University of California
Santa Barbara, CA 93106, USA

10. Expiry

This draft expires 28 August 2002.