A RTCP-based Retransmission Protocol
for Unicast RTP Streaming Multimedia

Abstract

This note defines a retransmission request protocol for use with unicast Real Time Protocol (RTP) [1] multimedia sessions. We refer to this protocol as RTPRx. The protocol is defined as a new RTCP type and is flexible enough to allow a variety of retransmission request and response schemes. This note describes the basic format for retransmission requests, and provides examples of three different retransmission request-response algorithms which use different methods to determine if retransmissions should be requested and whether the requests should be served.
1. Conventions and Acronyms

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [2].

2. Introduction

RTP was designed as a flexible protocol capable of transporting real-time data over multicast or unicast. It has been widely deployed and used extensively for transmitting real-time (or "near" real-time) multimedia signals, commonly called media streams. And although considerable effort went into the design of RTP so that it could support multi-party interactive conferencing over multicast, a common use of it today is for one-way non-interactive transmission of media streams. Because the communication is non-interactive, extra playback buffering and delay may be tolerable, which in turn enables receivers to request and receive retransmissions of lost RTP packets before their scheduled play-out times. For example, popular commercial products like RealNetworks’ G2 Player and Microsoft’s NetShow use RTP as the underlying transport protocol for their media streams, and use retransmissions in unicast delivery sessions.

However, as noted in [4], no standard exists for requesting the retransmission of streaming media. As a result, various independent and incompatible research and commercial schemes have been developed for retransmission of RTP streams [3, 6]. This purpose of this note is to define a standard retransmission request protocol for use with unicast RTP streams.

3. Control packets for unicast session

This note proposes extending the RTCP specification to accommodate retransmission requests. RTCP is a reasonable place for putting control packets into because it is already set up and no extra connection does not need to be established. However, the RTCP interval specified in [1] is too large for retransmission request. Thus, the limitation of RTCP interval should be relaxed and RTCP should be deployed for control packets for unicast session.

3.1. RTCP Transmission Bandwidth

The RTCP transmission interval specified in [1] is not frequent enough to allow efficient control of a unicast session needing retransmission requests of lost packets. The recommended value in the RFC for a minimum interval is 5 seconds between control packets. In order to make RTCP applicable for unicast retransmission, a packet-granularity RTCP interval is necessary.
The minimum interval restriction SHOULD be removed for unicast session, so that retransmission request is allowed to be sent immediately after loss detection. In order to prevent that RTCP packets cause network congestion, it is RECOMMENDED that the average bandwidth allocated to RTCP is fixed at 5% of session bandwidth similar to RFC 1889.

If the size of RTP payload packets is 1 KBytes and the size of RTCP packets is 40 bytes, the 5% limitation is large enough for a receiver to send an RTCP packet even at every packet reception. The interval must be frequent enough to allow timely retransmission requests.

In order to regulate the sending rate of RTCP packets within the RTCP bandwidth, the deployment of such a traffic shaping scheme as a token bucket is RECOMMENDED. The token rate should be set at \( \text{rtcp}_{\text{bw}} / \text{rtcp}_{\text{size}} \), where \( \text{rtcp}_{\text{bw}} \) is the RTCP bandwidth and \( \text{rtcp}_{\text{size}} \) is the size of RTCP packets. The bucket size should be determined according to the playback buffer size. As long as a token is in the bucket, the RTCP packet is sent immediately, and the long-term RTCP rate is regulated within the the RTCP bandwidth.

3.2. Packet format for requesting retransmission

This section defines a packet format for a multipurpose acknowledgment (MACK) packet. The MACK packet’s default use is as that of a negative acknowledgment (NACK) which is sent from the receiver to the sender and which asks for retransmission of one or more RTP packets. This default MACK interpretation may be changed by a retransmission protocol to allow for more general use (e.g., as a positive acknowledgment or selective acknowledgment). The format of the RTCP MACK packet is as follows:

```
0                   1                   2                   3
+-----------------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|   RXP   | PT=RTCP_MACK  |          length               |
+-----------------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             FSN               |            BLP                |
+================================+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
| protocol-specific extensions                      |
|                             ....                              |
```

The first 8 octets are present in every NACK packet. The various fields V, P, PT, and length are defined in the RTP specification [1]. The FSN and BLP fields have similar definitions to those given for the RTCP_NACK packet [3]. We summarize the meaning of all of the fields below:
version (V): 2 bits
   This field identifies the RTP version. The current version is 2.

padding (P): 1 bit
   If set the padding bit indicates that the RTCP MACK packet contains additional padding octets at the end which are not part of the retransmission control information but are included in the length field.

retransmission protocol (RXP): 5 bits
   This field identifies the specific retransmission protocol being used. The specification of retransmission protocols is outside the scope of this document; however, example protocols using this NACK format are given in Section 4. Note that a protocol specification MAY change the interpretation of the information presented by the FSN and BLP fields (see below). As part of its specification, a retransmission protocol SHOULD define who determines whether to send and/or respond to a retransmission request (the sender or receiver, or both). This allows clear identification of what steps, if any, are taken by the protocol to attempt to minimize the request of retransmissions that will not arrive in time for play out, or to suppress the response to requests that will arrive late. Additionally, a retransmission protocol SHOULD define protocol-specific extensions to the MACK if there is additional information to be communicated from receiver to sender for proper operation of the protocol.

packet type (PT): 8 bits
   This is the RTCP packet type which identifies the packet as being an RTCP MACK.

length: 16 bits
   The length of this RTCP MACK packet in 32-bit words minus one, including the header and any padding. This conforms with the definition of the length field used in RTCP sender and receiver reports [1].

first sequence number (FSN): 16 bits
   The FSN corresponds to an RTP sequence number of a media stream. Unless otherwise explicitly re-defined in the specification of a retransmission protocol, a MACK packet with a FSN field set to N is to be interpreted as a signal that the receiver has not received RTP data packet N, and that the receiver is requesting the retransmission of packet N. A retransmission protocol’s specification MAY redefine the interpretation of the FSN field (for example, it could become a positive acknowledgment). However, it is expected that this standard interpretation will be used by most retransmission protocols.
The BLP allows for the reporting of losses of the 16 RTP packets immediately following the RTP packet indicated by the FSN. Unless explicitly re-defined by a retransmission protocol specification, the BLP’s definition is identical to that given in [3]. Denoting the BLP's least significant bit as bit 1, and its most significant bit as bit 16, then bit i of the bitmask is set to 1 if the sender has not received RTP packet number FSN+i (modulo 2^16) and the receiver is asking is requesting its retransmission, and 0 otherwise. Thus, for a retransmission protocol using this default definition, the sender MUST NOT assume that a receiver has received a packet because its bitmask was set to 0. Thus, as specified by [3], the BLP is set to 0x00001 if the packet corresponding to the FSN and the following packet have been lost.

Note that the receiver may detect RTP packet loss via sequence number gaps, timer expirations, or other methods; however, details of the loss detection scheme are outside of the scope of this document but MAY be included as part of a retransmission protocol specification.

3.3. Other RTCP packets

For payload specific request packets (e.g. full INTRA-frame request for H.261 [3]), other RTCP packet types SHOULD be defined.

In order to realize sophisticated retransmission scheme which prevent packets which would arrive late from being retransmitted, extra control information is necessary (some examples are shown in the next section). New RTCP packet types SHOULD be defined to allow for communication of control information that cannot simply be included in the MACK’s protocol-specific field. Examples 4.2 and 4.3 illustrate this via use of a suggested "forward feedback" RTCP packet type.

The relaxation of the minimum RTCP interval limit enables fine granularity congestion control and loss recovery, similar to the level provided by TCP. For example, RTCP MACK packets could act as a positive acknowledgment signal, and a retransmission protocol may specify that bit 0 in the BLP field indicates which packets have arrived. Such a protocol might dictate that the sender uses gaps in the acknowledgment space or timers (similar to TCP) to decide which packets need retransmission, thus allowing the sender to perform TCP-like window based congestion control. Implementation of such a protocol would only require that a new RXP protocol type be defined for these SACK packets.
4. Examples

4.1 A simple retransmission request system

The simplest way to realize loss recovery using retransmission is that the receiver decides whether NACK packet should be sent or not and the sender replies all retransmission requests. When the receiver detects lost packets that should be in the play-out buffer, NACK packets for them is sent.

This scheme can be realized without an extra control packet between the sender and the receiver. However it could result in many late packets that arrive after play out time because it does not consider transmission delay.

4.2. Receiver-based decisions on requesting retransmissions

Before making a retransmission request at time $T_r$ of packet $N$ (scheduled for play-out at time $T_p[N]$), the receiver checks if

$$T_r + eRTT < T_p[N]$$

where $eRTT$ is an estimate of the round trip time (RTT). If true the receiver assumes the retransmission will arrive in time, and it sends the request for packet $N$. If false the receiver does not send a retransmission request. The above check can be made more general to account for receiver decoding delays and the sender’s request response time.

This scheme requires the receiver to maintain an estimate of the RTT. Although [RFC 1889] enables the sender to measure the RTT via the exchange of sender and receiver reports, no mechanism is currently defined which allows the receiver to measure the RTT. Thus a new RTCP packet type might be defined and used with this protocol which allows the sender to communicate its RTT estimate to the receiver. For example, the following RTCP packet could be used:

```
0                   1                   2                   3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P| CON=RTT | PT=RTCP_FF    |     maximum sequence number   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   sender’s RTT estimate                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where the RTCP packet type is RTCP_FF (for "forward feedback") and the RTCP_FF "CON" (content) field specifies that the packet contains...
the sender’s current RTT estimate. The maximum sequence number is the highest RTP sequence the sender has transmitted at the time it sends its RTT estimate; this allows the receiver to ensure that any estimate received is more current that its previously recorded estimate.

This protocol may require more frequent exchange of sender and receiver reports for timely update of the RTT. As an alternative the receiver could directly estimate the RTT itself. To enable this the retransmission protocol could instead specify a receiver timestamp be placed in the protocol-specific field of the MACK packet, and define a new content RTCP_FF type (e.g., a a timestamp echo "TSE" content) echoing the timestamp and containing a difference measurement.

4.3. Sender-based decisions on responding to retransmission requests

Sender-based means the sender determines if a retransmission will arrive in time, not that the sender chooses which packets need retransmission (i.e., via TCP-like positive ACKs). The sender has to estimate the deadline when the retransmission packet must be sent to arrive at the receiver before play out time. In order to estimate the deadline, the receiver has to send feedback information about time difference between the packet reception and its play out time. An example of the scheme is detailed in [5].

5. References


Support for Digital Audio and Video (NOSSDAV’97), Washington University in St. Louis, Missouri, May 1997.

AUTHORS’ ADDRESSES

Matthew Podolsky  
UC Berkeley Computer Science Dept.  
Phone: 510-642-8905  
Email: podolsky@eecs.berkeley.edu  
URL: http://www.eecs.berkeley.edu/~podolsky

Koichi Yano  
UC Berkeley Computer Science Dept.  
Phone: 510-642-9513  
Email: yano@cs.berkeley.edu  
URL: http://www.cs.berkeley.edu/~yano

Steven McCanne  
UC Berkeley Computer Science Dept.  
Phone: 510-642-0865  
Email: mccanne@cs.berkeley.edu  
URL: http://www.cs.berkeley.edu/~mccanne

This draft was created in October 22, 1999.  
It expires April 22, 2000.