CESNET technical report number 7/2003

Synchronizing RTP Packet Reflector

Tomáš Rebok, Petr Holub

2003-09-19

1 Abstract

In this technical report we describe an enhanced RTP/UDP packet reflector (known as `rum` [rum]) we have developed. This version of reflector is capable of listening to several separate incoming streams, buffering all incoming packets and synchronizing and reordering them before sending them back to the listening clients. We also describe an example application of such reflector.

2 Introduction

In contemporary Internet the most of the networks have consistent problems with reliable multicast support which is demanded not only by multimedia applications that use such infrastructure for efficient content distribution among clients. We have developed a tool that replicates (or reflects) RTP/UDP traffic from each connected client to all of them and thus it simulates multicast functionality on unicast network at cost of efficiency loss compared to true multicast. Original reflector implementation called `rum` by Julian Highfield [rum] has been further developed and enhanced by our group [TNC]. When using reflectors data may be sent over one link in multiple copies while in multicast data go over one link in maximum of one copy only. Advantage of reflector based networks lies in independence on underlying infrastructure capabilities and in possible implementation of many interesting features [ICN]. Efficiency drawbacks can be partially eliminated by creating overlay network using tunneling between reflectors. In this report we describe reflector implementation that performs synchronization of multiple reflected RTP packet streams for synchronized, timely, and in-order delivery to connected clients. Examples of applications for such reflector are described at the end of the report.

3 RTP packet format background

RTP packets include relative time-stamping information which may differ both in time base and time increment for different source streams coming even from several applications running on one client computer. Conversion between relative time and "real" absolute time can be performed using information sent in RTCP packet that are sent on lower frequency for each stream. RTCP packets contain both relative time-stamp and absolute time-stamp in NTP format. Therefore after receiving two RTCP packets it is possible to calculate both relative time base and increment (Fig 1). If source machines have their clocks synchronized e.g. using NTP protocol it is also possible to synchronize streams coming from different machines.

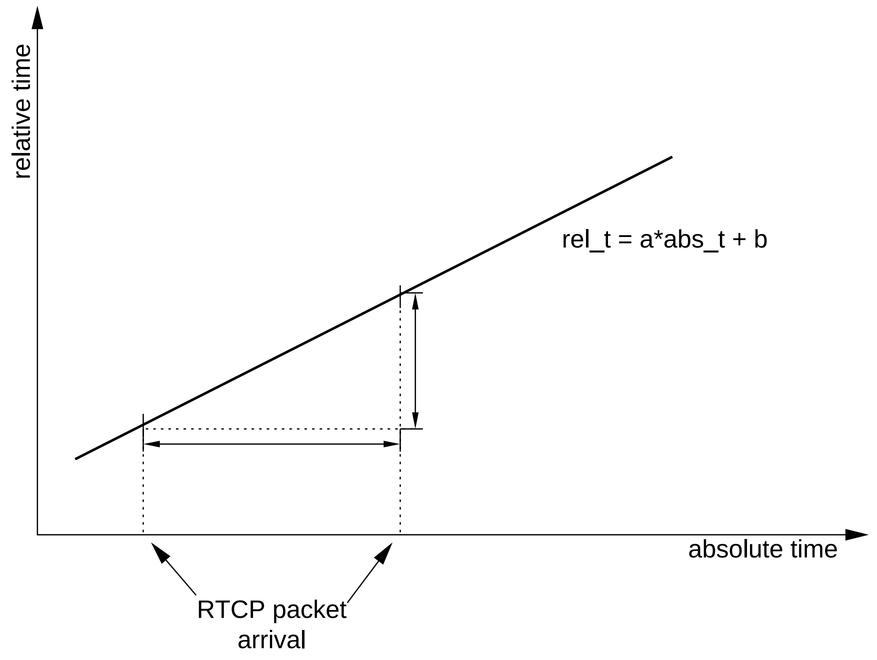

Figure 1: Conversion between "real" absolute time and relative RTP time using RTCP information.

Since UDP packets have no guarantees in terms of delivery it is necessary to perform two steps when synchronizing:

- reorder or discard out-of-order packets
- match time information for packets in different streams to send packets to clients synchronously

It is necessary to understand that due to packet processing the latency of transmission increases which is not desirable for interactive applications like videoconferencing where even small latencies in order of a few hundreds of milliseconds might induce communication problems (e.g. when one person tries to interrupt the other one to express his/her opinion).

Because of possible perturbations on the network the synchronizing reflector must be placed as close to the end-client as possible. Thus it is interesting to combine its features with tunneling to allow joining in network of ”normal” reflectors.

4 Implementation

The reflector implementation uses multi-threaded model in which several threads are used as network listeners that place packets coming from different streams into ordered buffers for each stream. Then the sending thread takes packets out of these queues and sends them to the connected clients in synchronized way if requested.

First `rum` analyzes specified command-line arguments. After initializing all mutexes guaranteeing mutual exclusion of shared variables, `rum` starts N separate threads where N is the number of ports placed as arguments. The main thread is now used as sender and the N threads are used as receiving listeners. Each receiving thread initializes particular socket the reflector is listening to (there are actually two sockets initialized for each RTP session - one for RTP and other one for RTCP packets).

When RTP packet arrives to the listener thread RTP header is extracted and parsed to obtain packet relative creation time which is in turn converted into absolute time. Then the packet is stored into time-sorted buffer - oldest packets are on the top and wait to be sent. Information on stored and dropped packets is kept for statistic purposes.

After receiving RTCP packet the data for conversion between relative RTP time and absolute time is updated taking into account previous conversion data and performing sliding average. We assume linear dependence of relative and absolute time (Fig 1). If abrupt change occurs program waits for at least three consecutive RTCP packets carrying consistent time information to achieve stability and avoid short time fluctuations. The RTCP packets are either saved into RTCP buffer (*-b* option) or sent immediately to the connected clients.

The main function of sending thread is to send packets which are saved on top of all the buffers (doing that synchronously if requested by *-s* option). When *-c* option is specified all the late arriving packets are dropped from the buffers. Sending is performed using Round Robin method for all buffers. Packets on the buffer top are sent to the connected clients when its time is earlier or equals to time of packets in other buffers). When all buffers run out, the sending process stops and reflector waits for incoming packets. Using *-t* command line option it is possible to make the reflector stop for random time period after the completion of each cycle. In this case demands of reflector on processor time are decreased. This option is ignored when some buffer is more than 50% full.

Command line syntax:
rum [-bcrsSt] port1 [port2 ... portN]

Description of command-line options:

- *-r*
  It is necessary to use this option when reflecting RTP data. In this mode `rum` uses specified ports for RTP packets while RTCP packets are listened to on port numbers incremented by one. When MBone tools are used specified port numbers must be even when using this option.

- *-s*
  This option makes `rum` synchronize packets when sending them from buffers. It means that `rum` sends simultaneously only packets created in the same time (even though they haven't come in the same moment). When packet created earlier is encountered, the behavior depends on *-c* option. *-s* option is available only when using RTP/RTCP packets (using *-r* option).

- *-b*
  Incoming RTCP packets are buffered before sending if this option is used. Otherwise packets are sent immediately after receiving. This option makes sense only when using RTP protocol (*-r* option).

- *-S <filename>*
  When using this option, `rum` calculates statistic information on running sessions (such as the number of all incoming packets, the number of missed packets, etc.) and saves this information into plain-text file named `<filename>`.

- *-c*
  This option results in dropping all late received packets (otherwise these are sent as soon as possible).

- *-t*
  If `rum` poses significant load on the processor you can use this option to decrease processor load. Sometimes it may result in increased number of missed packets (see above).

5 Applications

This synchronizing version of RTP packet reflector has been motivated by development of application for transmission of DV compressed 3D video encapsulated in RTP where streams for left and right eye are sent separately over the network. This application is currently under development and testing.

Another area where synchronized transmission of RTP packets might be interesting is multichannel audio with channels sent as separate streams. This may allow demanding computations performed on the audio stream in parallel on several independent computers synchronized using e.g. NTP protocol.

6 Conclusions

As the synchronizing reflector has been developed and tested on some dummy high-volume RTP streams we are currently proceeding with development the 3D video transmission application that uses separate streams for each eye. Further development of synchronizing reflector will go in direction of its reimplementation as a processor in new reflector architecture based on ideas of active routers [ICN].

References

[rum] Highfield J., *UDP Packet Reflector - rum*, `http://spirit.lboro.ac.uk/mug/mug.html`

[TNC] Hladká E., Holub P., Denemark J., *Teleconferencing support for small groups*, Terena Networking Conference 2002, Limerick, Ireland

[ICN] Hladká E., Holub P., Denemark J., *User Empowered Virtual Multicast for Multimedia Communication*, submitted to International Conference on Networking 2004.

[miro] CESNET Videoconferencing Server, `http://miro.cesnet.cz/`