Internet Engineering Task Force
INTERNET-DRAFT
draft-ietf-avt-X11-new-00.txt

Audio-Video Transport WG
L. Gannoun
EURECOM

March 11, 1998
Expires: September 11, 1998

RTP Payload Format for X Protocol Media Streams

1. 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 ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

Distribution of this document is unlimited.

2. Abstract

This document specifies the payload format for encapsulating X-protocol streams in the Realtime Transport Protocol (RTP). This specification is intended for X-protocol media streams that are not already handled by other RTP payload specifications. This specification gives details of the payload format of X-protocol data requests that are carried over RTP/UDP/IP in a shared window session. Multiple X protocol requests can be carried over an RTP packet as well as one X protocol request can be carried over multiple RTP packets. a shared window header is defined within the RTP payload to carry X protocol media requests. An RTCP join and RTP reply packets are specified to allow a latecomer to join an on-going session. This specification is intended for streaming stored X Window protocol data as well as live X protocol data requests.

3. Introduction and Motivation

In a conference, information must be distributed to all the conference participants. Early conferencing systems multiplex all of the data streams, so that, they use a unicast connection between each pair of participants which means that each information must cross some networks more than once [7]. However, the Internet architecture provides a more efficient approach of multicasting the information to all participants. Multicast transmissions save the network bandwidth compared to unicast transmissions.

Application sharing systems have become a popular vehicle for supporting distributed, synchronous collaboration. However, existing window shared systems, use unicast channels to multiplex X protocol data streams to all participants of the conference session [8],[9]. This is not optimal, since the source that multiplexes the X protocol data streams will be heavily loaded when the number of participants becomes very large. Moreover with multicast transmissions the network bandwidth is reduced comparing to unicast transmissions.

Furthermore most multiparty conferencing systems over the Internet re being implemented over IP multicast which is provided by the MBone [7]. Due, to the limitations of the traditional application sharing systems, new approach that is based on multicast delivery of X protocol data streams is

required. Thus, a new shared window system is designed for high requirements of scalability to very large numbers of participants, TCP-like congestion control and high efficiency. This system is intended to be used in a distance learning environment for large groups.

This specification proposes a sheme to carry X-protocol data requests over RTP. In addition, it specifies some RTCP packets that are required for the Latejoin-session protocol. The sheme specified here handles the X-protocol requests which are loss-intolerent media. This specification is intended for streaming stored X-window sessions as well as live X-window shared session.

4. X-window System Overview

The X Window System is both hardware and operating system independent, written application software will compile and run on any system that supports X. The X-window system is a server that accepts requests to manipulate the display on the computer's console while reading input from the console's keyboard and mouse devices [3]. The client is a program displaying on the screen and taking input from the keyboard and the mouse.

A client sends drawing requests and information requests to the server. The server, sends back to the client user input, reply to information requests, and error reports [3]. The client may run on the same machine as the server or on a different machine over the network. Communication between the server and clients is managed by the X-protocol messages.

There are four classes of X-protocol messages that can be transferred over the network which are the requests, replys, events and errors [3]. For the purpose of this document we will be concerned with multicasting X protocol data requests using RTP. An X-protocol data request can carry a wide variety of information, such as a specification for drawing a line or changing the color value in a cell in a colormap or an inquiry for the current size of a window. An X-protocol request can be any multiple of 4 bytes in length.

X-Protocol data requests that are multicasted are those that do not need replies from the X window server. However, some of the X protocol requests that need replies (Roud-trip requests)

such as, AllocColor, AllocNamedCollor, etc...are not filtered and are multicasted to all receivers. These requests are normally handled by the shared application. However, in the receivers side they are handled by the shared window system application level that adapts them to remote X servers.

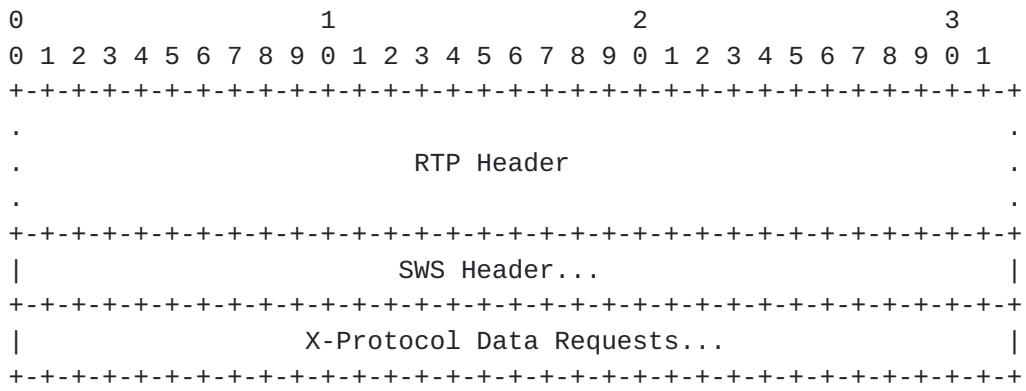
X protocol data requests are loss-intolerent media data. Then, these requests are carried over RTP/UDP using a reliable multicast protocol such as the Scalable Reliable Multicast protocol (SRM) [5].

5. Payload Format Specification

The encapsulation scheme described here specifies that each X-request media stream associated with a single shared X-application is sent over the same RTP session.

The X-request media stream is carried as payload data within the RTP protocol. There is a fixed shared window (X-request media) header immediatly following the RTP header. The X-protocol media stream is packetized and placed in the RTP packet following the shared window header.

The RTP packet is formatted sa follows:



5.1. RTP Header

The format and general usage of RTP header fileds are described in [1]. The following fileds of the RTP header will be used as specified bellow:

- The payload type should specify the static payload type of X-protocol media streams (if assigned) or one of the

dynamic payload types. (The need of a static payload type can be discussed in the IETF AVT working group).

- The RTP timestamp is based on the time capture of the first X-protocol data request if we have more than one X-protocol request. The timestamp encodes the capturing instant of the first X-protocol media request contained in the RTP packet. Multiple requests can be encoded in the same RTP packet or a single X-protocol request may require multiple RTP packets. Packetization rules are defined in a subsequent section. If an X-protocol request occupies more than one packet the timestamp will be the same for all the packets. An RTP packet containing multiple X-protocol requests do not require different timestamps. In fact these requests are captured at different instants but the diffrences of their timestamps are not significantly and hence are grouped to form the same group. The timestamp of this group is the timestamp of the first captured X-request of this group.

- The marker bit of the RTP header is set to one in the last packet of a segmented X-request and otherwise, must be zero. This marker bit significance can be also foundd in [2]. If one or more requests are fully contained in an RTP packet the marker bit must be set to one. Thus, it is possible to detect that a complete request has been arrived and can be decoded and presented (translated and after lanched to the X-server).

5.2. X-protocol Data header (SWS Header)

The Shared Window header is defined as follows:

The fields in the Shared Window header has the following meanings:

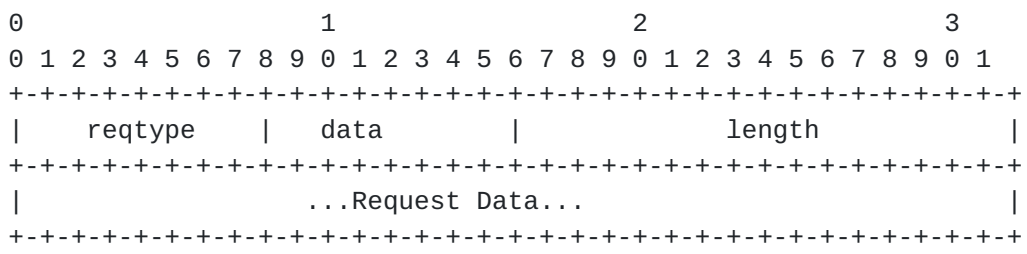
VER: 4 bits
A version field must be set to zero by transmitters implementing this specification.

J: 1bit
This field identifies this RTP packet as a Join/Reply packet. This helps to distinguish these Join packets from normal RTP packets that can be carried by the same channel-group. This field should be set to zero if the packets are not Join/packets and otherwise to 1. Later in a subsequent section we will explicite how RTP join packets can be sent on a separate channel-group from the normal RTP packets group that carries X-protocol data requests.

AID: 12 bits
This filed corresponds to the Application IDentifier that is being shared. In fact this identifier is relative to an X-client which is shared. Since, an X-application can be composed of several X-clients, then we should associate the X-protocol data requests to the corresponding shared X-client. Furthermore, different applications can be independently shared and hence the corresponding X-protocol data streams can be multiplexed in the same RTP/UDP/IP connection.

RC: 16 bits
The Request Count (RC) field represents the number of requests that are encapsulated in the RTP packet. If the RTP packet carries only a portion of a segmented request, this field should contain the length of the request segment that immediatly follows this header. The X-protocol request data starts at the RTP data offset plus the 4 bytes of the fixed Shared Window header.

In the case of a grouped requests, it's imperative to have a length field for each request in order to extract correctly each request. However, in the X-request protocol header there is a length field that represents the length of the X-request in 4-bytes quantities of the hole request. Then we can reuse this field as a request length field. The X-protocol request header is defined sa follows in the X11-protocol data requests specifications [3].



The fields meanings are as follows:

reqtype: 8 bits
This field represents the request type. there is 128 different request in the X-protocol request specifications.

data: 8 bits
This field has a meaning relative to the request type.
length: 16 bits
This field contains the number in 4-bytes quantities of the hole request including the 4 bytes of the request header.

Thus we can reuse the original X-request header to extract properly the length of each request encapsulated in the RTP packet. This helps saving the bandwidth with 2 bytes for each request transmitted in the RTP Packet.

6. Join Request/Reply Packet Formats

This section explains how we can use RTP/RTCP packets to allow a latecommer to join in a consistent way a teleconference based on a Shared Window System. Since a Shared Window System multicasts X-protocol Data Streams which are medias that do not support losses and require a reliable multicast transport protocol. A such protocol that can handle X-protocol data streams is the Scalable Reliable Multicast protocol (SRM). An other feature of the Shared Window System is that a latecommer should request a special context of the shared application. This context is defined as follows:

A consistant context at an instant t of the space time of the shared window application is:

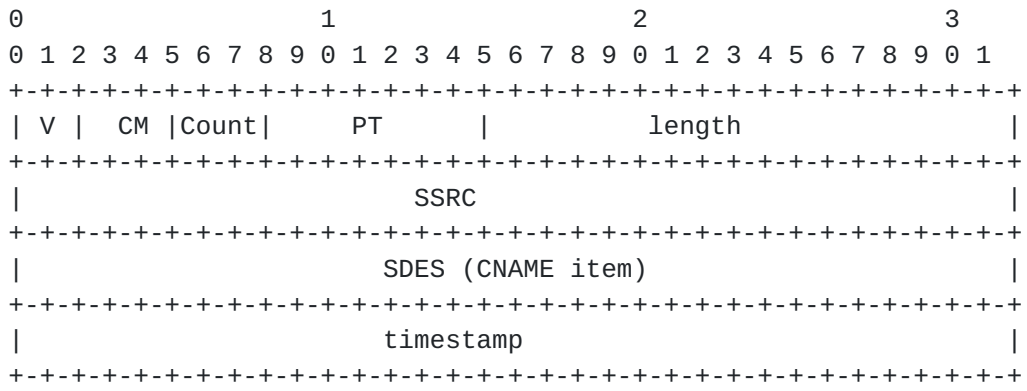
- The state of allocated ressources in the X-window server relative to the shared application at this instant t.

- The state of the Graphical User Interface (GUI) of the shared application at this instant t.

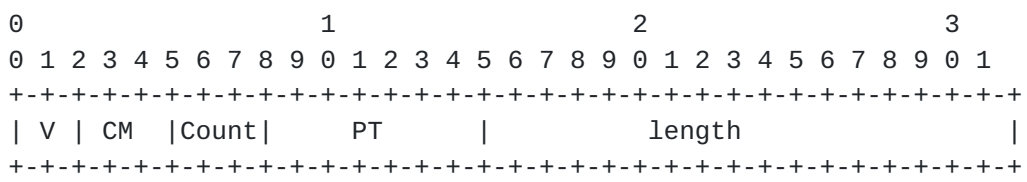
A context can be provided by either the source or any member of the teleconferencing group (the nearest member).

In order to request a consistant context of an application being shared in a teleconferencing environement we define an RTCP packet that request a special context of a shared application. This RTCP packet can be a variant packet from RTCP-SRM [4] packets that are defined in order to implement SRM with the RTP/RTCP.

The format of this packet is as follows:



In this format we distinguish an RTCP header that is defined as:



V: 2 bit
The version of the RTP equals to 2 (RTP version).

CM: 3 bits
A novel type of a command that defines a join request command. This field is defined in [4] as 2 bits field and hence can not handle the novel Join/Request command. This field is extended to a 3 bis field to incorporate the Join/Request command and should be set to 4.

Count: 3 bits
Here is not used. PT: 8 bits
This Payload Type field contains 205 to identify this RTCP packet as an SRM packet as defined in [5].

length: 16 bits
Defines the length of the hole RTCP packet in 32-bits mines one including the header. Following the RTCP header in the Join/request packet we find the following fields:

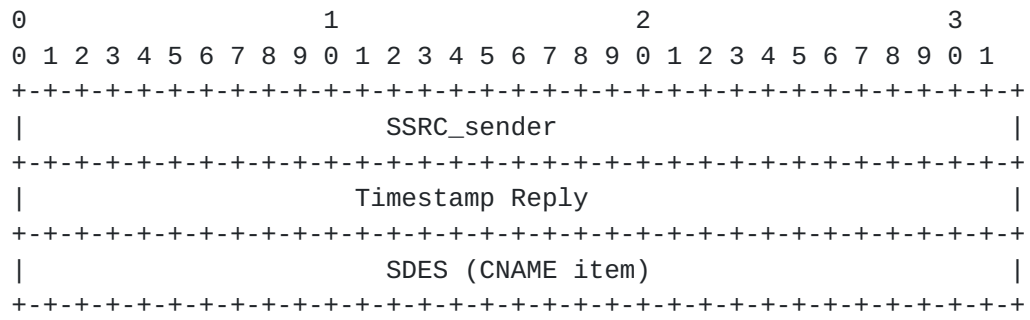
SSRC: 32 bits
It identifies the Source of the sender and

SDES (CNAME item): 32 bits
The canonical end-point identifier is a source description (SDES) item defined in the RTP protocol[1]. This name is a unique identifier that identifies the sender (the latecommer user) of the Join-request message. By this address the latecommer will be contacted by potential providers of the shared window context. After an electing protocol the sender of the context transmits the shared window context on a special multicast channel to all the latejoiners.

This allows providing the shared application context to a multiple and large number of latecommers. This follows a special Join-Session protocol that is described in [6].

Timestamp: 32 bits
This timestamp serves as a to compute later an RTT between the the latecommer and potential providers of the shared window context.

The Reply is only received by the transmitter of the Join-Request message on a unicast channel. Its format is as follows:



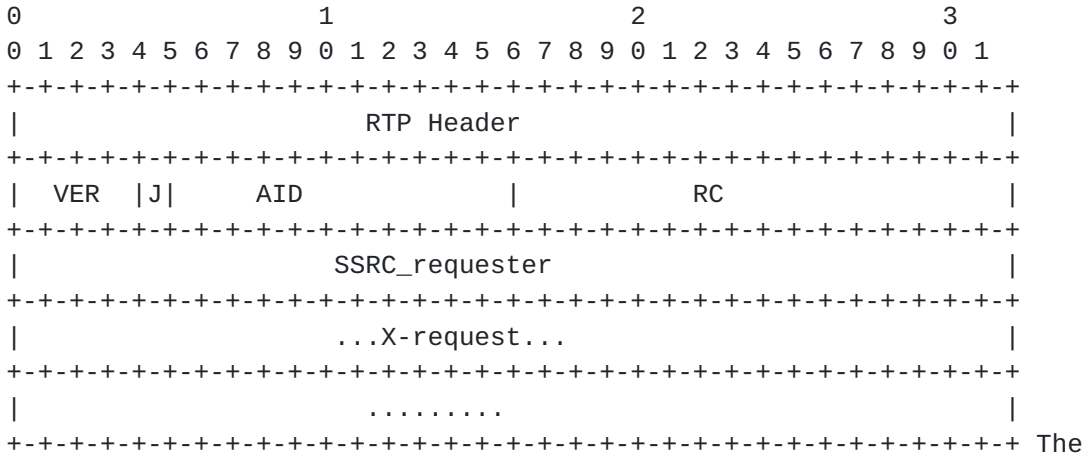
This packet may not be an RTP packet, since it's intended to be carried by a unicast channel as a response from the potential providers of the shared window context. The fields are:

SSRC_sender: 32 bits
Identifies the sender of the Join-reply as a potential sender of the shared context.

Timestamp Reply: 2 bytes
It's composed of two fields, the Last Timestamp Query (LTQ) and Duration from LTQ (DLTQ). These fileds are computed as defined in [5]. These fields help computing the Round Trip Time (RTTs) between the potential providers of the shared window context and the the requester of this context.

SDES (CNAME item): 32 bits
The canonical end-point identifier serves to compute an IP adress of the provider by which the latecommer can confirm to this provider his election. On receiving the confirm message, it then transmit the shared context on the join-multicast channel.

The RTP data packets for providing the shared window context are defined as follows:



The RTP header contains the SSRC identifier of the site that transmits the Join-packets.

SSRC_requester: 32 bits
Its an identifier of the member participant that requests the shared window context.

The data format of a Join-packet is similar to that of an RTP payload data format described in section 2.3.

Following immediatly the Shared Window Header which is of 4 bytes length, we find a sequence of requests. These requests are generated from the sender to create the current shared window context on the remote latecommer X-server.

7. Media Data Packetization

The following definitions are required to make paketization decisions:

- MDR: Maximum X protocol Data Requests that can be accommodated in a single RTP packet.

- DRS: All Data Request Size of a group of requests

- SDR: Specific Data Request Size

Sheme I: (DRS <= MDR) This sheme defines a packetization method of multiple X protocol requests that can be carried with a single RTP packet. The RTP header M-bit is set to one on all RTP packets.

Sheme II: SDR > MDR This sheme is specific to a segmentation of a request that will be carried over multiple RTP packets. The RTP header M-bit is set to one in the last packet and otherwise is set to zero.

8. Summary

Two problems will be treated in the near future and which are dealing with user interactions with the shared application and appropriate floor control policy that defines which users can interact with the shared application. This policy should be applied in a context of large groups. Besides, two other features will be supported and related informations will be carried over RTP packets. The first feature is Application information such as, the name of the shared application and the second one is related to the multicast messages for all the participants.

- User interactions: A user interaction at a special receiver generate a flow of X protocol events from the

user X server. Since these events are only handled by the shared application, then they are transmitted on a unicast channel to the shared window system at the shared application site. At this level these events are translated to the convenient ones and after lanched to the shared application. In response, the shared application generates X protocol requests that are multicasted to all participants.

- Floor control policy: A control policy is required to allow a participant, among all attached participants, to input events to a shared application. We can use here a policy based on a floor token that is requested, passed, and released from a specific user. A conference chairman can explicitly grant a floor or revoke a floor from a specific user as proposed in [9].

- Application Information: Information of the current shared applications can be obtained from the shared window system. Information can be requested by the following requests; ListApplications, GetApplicationName, GetApplicationOwner. These information can be directly requested from the control agent of the shared application site.

9. Open Issues

The following open issues need to be resolved:

- How to compress X protocol requests ? There are many techniques for compressing X Window System Protocol that contribute to bandwidth elimination which enables X applications to run over very slow connections such as wireless modems or on very small platforms such as hand-held notebooks. Compression sheme such as X remote [10] or Fast Higher bandwidth X, FHBX [11] which are a compressed transformation of the X protocol and can be used to compress X protocol data requests that are sent over a multicast channel-group. This reduces the bandwidth requirements of an X application when sharing it over IP multicast.

- What is the convenient Forward Error Correction sheme ? The strict delay requirements of real-time multimedia and

especially the shared window system in a wide area network environement, may usually eliminate the possibility of retransmissions. It's for this reason that forward error correction (FEC) has been proposed to enhance transmission reliability in the Internet [12]. In particular should we use a traditional forward error correction based on parity operations or should we adopt a sheme that sends redundant application data (the most recent X request groups sent) which more reflects the shared window application context and has a less bandwidth requirement.

Addresses of Authors

Lassaad Gannoun
Institut EURECOM
2229, route des Cretes BP. 193 06904 Sophia Antipolis
FRANCE
electronic mail: gannoun@eurecom.fr

References

[1] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

[2] A. Jones, A. Periyannan, and D. Singer, "RTP Payload Format for QuicTime Media Streams", INTERNET-DRAFT, July 1997.

[3] A. Nye,"X Protocol Reference Manual", O'Reilly & Associates, 1992.

[4] P. Parnes,"RTP extension for Scalable Relible Multicast", INTERNET-DRAFT. available at: http://www.cdt.luth.se/~peppar.

[5] Floyd/Jacobson/McCanne/Liu/Zhang,"A reliable Multicast Framework for Light-weight sessions and Application Level Framing". In Proceedings of the ACM/Sigcomm'95.

[6] N. Ben Ali,"Systeme de partage d'applications base sur le multicast", Master's Thesis, Institut Eurecom, June 1997.

[7] Handley/Crowcroft/Bormann/Ott,"The Internet Multimedia Conferencing Architecture", INTERNET-DRAFT, July 1998. available at: http://north.east.isi.edu/~mjh/papers.html

[8] H. M. Abdel-Wahab, and M. A. Feit,"XTV: A framework for sharing X-window clients in remote synchronous collaboration", In Proceedings of the IEEE Conference on Communications Software: Communications for Distributed Applications and Systems", 1991.

[9] T. Gutekunst, D. Bauer, G. Caronni, Hasan and B. Plattner, "A distributed and policy-free general-purpose shared window system", IEEE/ACM Transactions on Networking, February, 1995.

[10] D. Cornelius," XRemote: a serial line protocol for X", sixth Annual X Technical Conference, Boston, MA, 1992.

[11] J.M. Danskin, Q. Zhang and D.M. Abrahams-Gessel,"Fast higher bandwith X", Proceedings of the Multimedia Networking, IEEE Computer Society Press, pp. 192-199. Aizu, Japan.1995

[12] J. -C. Bolot and A. Garcia,"The case for FEC-based error control for packet audio in the Internet, Multimedia Systems, 1997.

Table of Contents