Notes on the use of RTP for shared workspace applications

Colin Perkins
Department of Computer Science
University College London
Gower Street
London WC1E 6BT
c.perkins@cs.ucl.ac.uk

Jon Crowcroft
Department of Computer Science
University College London
Gower Street
London WC1E 6BT
jon@cs.ucl.ac.uk

ABSTRACT
The Real-time Transport Protocol, RTP, has become the dominant protocol for streaming audio and video in IP-based environments. A number of proposals have been made which attempt to build on this success and apply RTP for shared workspace applications. We discuss the needs of such applications and the features provided by RTP, with an aim to showing why RTP is not appropriate for such uses.

1. INTRODUCTION
In recent years the Real-time Transport Protocol, RTP [15], has become the protocol of choice for audio/video transport in the Internet. One of the reasons for this success is the flexibility of RTP, which was designed as a protocol framework, rather than a monolithic protocol, allowing it to be tailored to different applications and media formats.

This flexibility has led a number of authors to consider the use of RTP for other scenarios, not necessarily related to audio/video transport. For example, it has recently been suggested that RTP may provide the base for a protocol for the transport of interactive media such as shared whiteboards [11]. In this paper we provide a critical evaluation of the applicability of RTP for these applications, noting those areas where RTP is not a good fit and suggesting alternative solutions.

This paper is structured as follows: section 2 presents an overview of interactive media applications, highlighting their key functional and protocol requirements. This is followed, in sections 3 and 4, by a discussion of RTP and its suitability for these applications. In section 5 we discuss other possible approaches. Finally we present our conclusions in section 6.

2. INTERACTIVE MEDIA
Informally, an interactive media application is one where the state of the system changes primarily in response to user interaction. Examples include a shared whiteboard or text editor, a distributed presentation tool, or a networked multiplayer game. It does not include audio or video, since those media are primarily time related, rather than responding to explicit user events.

A more formal classification of media flows is presented in [11]. That work defines interactive media to be those which can be influenced by events external to the media, and splits this definition into two: discrete interactive media are only influenced by external events, continuous interactive media also change their state over time without external interaction. In contrast, non-interactive media change their state over time irrespective of external events.

This leads to an important observation: interactive media applications have a multi-dimensional state space based on both time and the set of external events which may affect the system, whereas the state space for non-interactive media is linear.

Consider, for example, a shared presentation tool. This may have time-based state (e.g. an animation, or an audio/video clip), ephemeral state based on user interaction (e.g. a telepointer) and persistent state (e.g. the text and diagrams comprising the presentation). This is in contrast to a non-interactive media application such as streaming video, where media frames are presented in time order with no other interaction.

There are a number of consequensesdue to these observations. Firstly, we note that a single linear namespace is not sufficient to identify state updates in an interactive media application, since these updates may be triggered by a combination of factors.

Loss detection and reliability becomes more complex in interactive media applications, since there is no longer a single linear namespace for objects and since some objects are persistent. In addition, an interactive media application must be aware of the possibility of conflicting updates, since the state of the system may be affected by multiple external events simultaneously, complicating any reliability mechanism.

Handling late joiners becomes more complex, since the state of the system no longer depends solely on when they joined, but also upon the shared application state at that time.

Finally, we note that interactive media have partial rather than strict data ordering requirements, with late data being useful, and browsing being common.

These observations rapidly lead to the conclusion that a single protocol is unlikely to be sufficient for the full range of interactive media applications. Indeed, many authors [3; 4; 5; 6] have concluded that application level framing [2] is a requirement for such applications.

There are, however, a number of features in common across a broad range of interactive media applications, and those features can potentially be abstracted into a common protocol framework. In particular, it has been observed that many applications need structured application data unit (ADU) names [12], a simple mechanism for the detection of packet loss, a means of distinguishing different types of data, a means of identifying participants, and a timestamping mechanism.

It has also been observed that it is *not* necessary to include common reliability and error recovery schemes, or mechanisms for handling late joiners, since applications have a wide and disjoint range of requirements in these areas.

3. OVERVIEW OF RTP

RTP is a protocol framework designed to provide end-to-end delivery services for data with real time characteristics, including payload type identification, timestamping, sequence numbers, source identification and reception quality feedback. It draws heavily on the notion of application level framing [2] to provide mutability to different application scenarios, and efficient adaptation to different network conditions. These characteristics have lead a number of authors to consider the use of RTP for interactive media applications. In this section we review RTP in some detail, this will be followed in section 4 by a critique of the use of RTP for interactive media.

An application using RTP as its transport is expected to provide a single media stream with real time characteristics. Such a media stream comprises a number of application data units, which

- are ephemeral, with no long lived name
- are timed, and require timely delivery
- are ordered
- may come in a range of different types
- are associated with a named source
- are useful if delivered unreliably, but some feedback on reception quality is needed

These features are provided by a fixed packet header which is applied to each data packet. In addition, the RTP control protocol provides approximate membership information and reception quality feedback.

An RTP data packet comprises a fixed header, followed by an optional header extension and the application data. The format of the fixed header is shown in figure 1.

RTP provides a single sequence number space, used to detect packet loss and misordering. The sequence number increases by one for each packet sent, regardless of the contents of the packet and hence does not provide structured naming of application data units.

The RTP timestamp reflects the sampling instant of the first octet in the payload. It is a free running clock at a rate natural to the media format; for example the audio sampling rate or the 90kHz video

```
 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|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        contributing source (CSRC) identifiers (if any)        |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   optional header extension                   |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         payload data                          |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: An RTP data packet header

clock chosen to be a common multiple of PAL and NTSC framerate. It must be of sufficient resolution to achieve accurate synchronisation and to measure network jitter.

The RTP timestamp denotes the playout order for application data units. If those ADUs are resequenced before transmission it will not be monotonically increasing in the packets as received.

Since the RTP timestamp is derived from the media clock, and each media stream is transported in a separate RTP session, it is not possible to synchronize media streams from the information contained in the data packets alone. The mapping from RTP timestamp to real-time is obtained via sender report packets, described later.

The combination of sequence number and timestamp is sufficient to uniquely identify each application data unit. Due to the ephemeral nature of ADUs, there is no external mapping onto long lived names, and no feedback on the reception of individual ADUs is provided.

Each participant in an RTP session is identified by a 32 bit synchronisation source (SSRC) identifier included in each data packet. The SSRC is chosen randomly when joining a session, and a mechanism for resolving collisions is included. It should be noted that the SSRC is an ephemeral identifier, which is not suitable for long-term data naming. A mapping from SSRC to a canonical name is provided by source description control packets, described later.

RTP supports the concept of mixers, which combine data from multiple sources. Each data packet may include a list of contributing sources (CSRCs) for this purpose.

The RTP data packet header also includes a payload type identifier field, to differentiate payload formats. The mapping from payload format to payload type number is provided by either a static profile definition (e.g. [14]) or non-RTP signalling (e.g. via SDP [7] carried in SIP [9]).

Finally, a marker is included as a hint for the receiver to indicate that this ADU denotes a significant event in the media stream. The significance of a packet with the marker present is unknown to an application which is not familiar with the ADU semantics.

In addition to media transport, RTP provides a separate reporting and control protocol, RTCP. This control protocol works by periodic rate controlled multicast from each participant, such that the average aggregate control traffic rate is 5% of the data rate. The interval between transmissions of control packets from each participant is at minimum once every 5 seconds, randomised up to 50% in either direction. This reporting interval increases as the number of members in an RTP session increases, such that RTCP cannot be considered a timely protocol.

There are four types of RTCP packet: sender report, receiver report, source description and BYE. Multiple control packets may be transported in a single compound packet. Control packets are associated with their sender by SSRC.

An RTCP sender report has two functions: it maps between the RTP media clock and an NTP format timestamp giving wallclock time, and it provides a packet and octet count for the media stream (allowing the data rate to be estimated). The mapping between RTP and NTP time allows receivers to perform inter-media synchronisation (e.g. for lipsync [10]) and in conjunction with information from receiver reports allows the sender to estimate the round trip time to all receivers. As their name implies, sender report packets are only sent by those members who are actively transmitting data.

Receiver reports are sent by all members who receive media data during a reporting interval. They contain a report block for each active source, that report block contains

- the fraction of packets lost this interval
- the cumulative number of packets lost
- the extended highest sequence number received
- an estimate of the network induced timing jitter
- the timestamp from the last sender report (LSR)
- the delay since receiving the last sender report (DLSR)

The LSR and DLSR fields are included so that senders may estimate the round trip time to each receiver.

The estimate of the network timing jitter assumes that packets are evenly spaced, for example as is the case with most network audio applications. If packets are not evenly spaced in time, the estimate of the jitter will be incorrect.

Source description packets provide a mapping from SSRC to a canonical name for each participant. The canonical name comprises the participant's username and network address (e.g. colin@128.16.32.159), and so is consistent across restarts of a media tool, for example, but is not persistent or long-term consistent. In addition, other personal details such as participant names, email addresses and location may optionally be transported. Finally, BYE packets inform others participants that a member of the session is about to leave.

As discussed earlier, RTP is a protocol framework rather than a complete protocol specification. In order to be complete, RTP needs one or more payload format specifications and profiles.

Payload format specifications detail how particular media formats, such as audio and video encodings, are transported in RTP. They define the mapping from the codec output bitstream to an RTP packet stream, including any payload specific header information to be carried at the beginning of the payload section of an RTP data packet. More detail on the design of RTP payload format specifications can be found in [8].

An RTP profile defines a mapping from payload formats to payload type numbers, and may redefine some features of the base RTP header. The RTP specification [15] has this to say on the subject:

> "The existing RTP data packet header is believed to be complete for the set of functions required in common across all the application classes that RTP might support. However, in keeping with the ALF design principle, the header may be tailored through modifications or additions defined in a profile specification while still allowing profile-independent monitoring and recording tools to function."

These modifications and additions include the ability to change the size of the payload type field to include additional marker bits, add additional fixed headers to the data packet header, define new RTCP packet types and extend sender/receiver reports with additional fixed header fields.

It should be noted that profiles are not permitted to redefine the existing header fields, since that would affect the operation of profile independent monitoring and recording tools.

4. RTP FOR INTERACTIVE MEDIA?

We have presented the requirements of interactive media applications in section 2, and the features afforded by RTP in section 3. We now address the issue of how well RTP fits the needs of interactive media applications.

The features of RTP data packets which are directly relevent to interactive media applications include the payload type, synchronisation source, marker and padding bits.

The sequence number performs two roles in RTP: detection of packet loss and ADU naming. Interactive media applications may still benefit from detection of packet loss via the sequence number, but it is likely to be insufficient to identify ADUs in an interactive media application due to the non-linear namespace in such applications [12].

The semantics of the RTP timestamp are different to those expected by most interactive media applications, being based on a free running linear clock of relevence to the media. Whilst most interactive media applications can use a timestamp, they have no concept of a media clock and a better fit can be made by using wallclock time as their timestamp (this also implies that the mapping from the media clock to wallclock time via sender reports is not useful for these applications, although a clock synchronization protocol may be required to align the participants' ideas of wallclock time).

It should further be noted that many interactive media applications employ some form of reliability, whereby data packets are retransmitted to repair lost data. The use of a media timestamp to determine playout of such packets, as would be done following the RTP model, is clearly dubious. Figure 2 shows the RTP adaptive playout model, where media samples are buffered based on the observed jitter in the network and RTP timestamps, to regenerate the inter-ADU timing necessary for correct playout of the media (see, for example, [13] for a more detailed discussion of this process). An interactive media application typically does not have to regenerate the timing of a media stream before playout, and can often play ADUs as soon as they are received, or use late ADUs for repair when they would

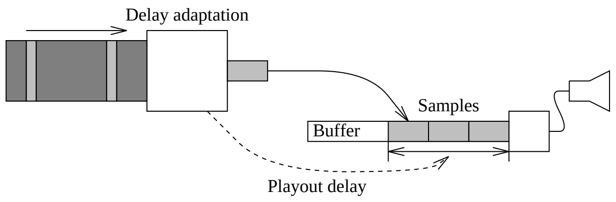

Figure 2: Adaptive media playout in an RTP application

have to be discarded by an application following the standard RTP playout model.

Of the four types of RTCP packet (sender report, receiver report, source description and bye), we have previously noted that RTCP sender report packets are of little use to interactive media applications, since their primary purpose is to convey the mapping between the RTP media clock and the real-time NTP clock. Since interactive media applications do not have a comparable media clock, this mapping is redundant.

This does not necessarily mean that the concept of sender reports is of no use to interactive media applications. It may be possible to use some form of sender report to inform receivers of the progress of a transmission, helping with the detection of tail-loss, or to allow estimation of the round trip time between receivers. Such sender report packets will, however, differ from those used by RTCP.

The concept of reception reports is valid for interactive media applications, but the details of the report are likely to differ from those used by RTP.

The statistics relating to loss fraction and cumulative number of packets lost are meaningful only if packets are transmitted with a unique sequence number per packet, and retransmissions are not used (or if a retransmitted data packet has a different sequence number from the original). This is not to say that analogous measurements cannot be derived for interactive media applications, but unless these fields are calculated in exactly the same way as per [15] those applications which are using the original definition will be confused by reception reports using a different definition for these fields.

The extended sequence number field in a reception report is not useful for those interactive media applications which use structured sequence numbers, since it relates to the linear RTP sequence number space. If a linear sequence number space is applied to data packets, with structured sequence numbers being carried within the payload, then the extended sequence number will be meaningful, but perhaps not useful.

The interarrival jitter field in a reception report gives meaningful results only if data packets are sent regularly. Since interactive media applications typically have non-regular data transmission patterns, this field will often be valid but meaningless.

The LSR and DLSR fields are useful if the application uses sender reports.

Source description and BYE packets may be used unchanged, although since sender and receiver report packets are not appropriate in their existing form, it may be necessary to change the packing rules for compound RTCP packets.

It is important to note that RTP explicitly allows for profile independent media applications, and it is not legal to redefine the calculation of the existing header fields, since this would confuse those applications expecting standard RTP (for example, a profile independent reception quality monitor or recording tool). An extension mechanism allows additional information to be carried in RTP/RTCP packets, allowing applications which require extra functionality.

The rules for when to transmit RTCP packets may not be appropriate for an interactive media application. The transmission rate was designed to scale to very large groups by dynamically adjusting the interval such that a constant rate of 5% of the data bandwidth is assigned to RTCP. The result is that an RTP application cannot send an RTCP packet at an arbitrary time, and must obey fixed transmission rules. Violating these will affect the operation of other (profile independent) applications which may be using the standard RTCP transmission rules.

For example, it is not appropriate to use RTCP as a mechanism to request retransmission of data packets since that would limit the chances for retransmission requests to those when the rate-limited RTCP packets can be sent.

Another example is the use of RTCP to convey data (as a signalling protocol) since this will affect the rate at which we can convey signalling messages. An application which relys on this signalling may be performance limited by the low rate at which RTCP operates.

To summarise, we believe that the needs of interactive media applications are not well met by RTP.

5. RELATED WORK

The primary motivation for this work was the RTP/I protocol [11], an attempt to define an RTP profile suitable for use by interactive media applications. That work provides an interesting and useful taxonomy of media applications (interactive vs non-interactive and discrete vs continuous), and addresses a number of important problems in the transport of interactive media – application level naming of ADUs and reliability, synchronisation with RTP media streams and recording of interactive media applications.

The major flaw of RTP/I, in our opinion, is the decision to make the protocol an RTP profile, rather than using RTP as a useful point in the design space from which to start work, and discard those aspects which are not useful. This shows itself in the design of an RTP/I data packet, where the header is twice the size of an RTP header due to the inclusion of a sub-component sequence number and identifier, in addition to the sequence number in the RTP header. In addition, the mapping from sub-component identifier to an application meaningful name is done via a new RTCP packet type, as are state queries. As previously mentioned, the rate limits for RTCP will seriously impact the performance of the system if they are followed, and if not could impact the performance of other RTP applications monitoring/recording an RTP/I session.

The RTP/I work does not deal with these issues, nor of the applicability or otherwise of the RTP timestamp, sequence number, receiver report statistics and use of sender reports.

As an alternative the Reliable Multicast Framing Protocol [3] and the Reliable Multicast Framework [4] both derive protocol frameworks for reliable multicast, with requirements similar to those of many interactive media applications.

The Reliable Multicast Framing Protocol [3] was designed as a generic protocol framework for reliable multicast; hence it is directly applicable to interactive media applications. It shares much of the same philosophy as RTP, but has a number of significant differences resulting from the different focus adopoted.

Like RTP, the RMFP specifies the format of an ADU header and a number of session packets (e.g. reception report). Those packets, whilst similar in concept have somewhat different structure and semantics to those of the corresponding RTP headers. Consider, for example, the RMFP data packet header illustrated in figure 3.

```
 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 |P|R|F|S|E|X| PAYLOAD TYPE  |            LENGTH             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           SOURCE ID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       SEQUENCE NUMBER         |          OBJECT ID            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
|   NAME LENGTH |                   ADU NAME                    :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
:                            ADU NAME                           :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: An RMFP data packet header

The RMFP data packet header includes a sequence number, object identifer and ADU name. However, RMFP does not specify the semantics of these fields - they must be defined by a profile specification for a particular class of applications (for example [1]). This allows considerable flexibility, and removes many of the problems associated with the use of RTP for interactive media: the semantics of the sequence number can now fit the application, and structured ADU names can be directly used.

RMFP ADUs also include bits in the header to indicate if this packet is a retransmission or contains forward error correction (FEC) information. The explicit inclusion of these bits allows for applications which can use jitter information to calculate this, without having to include such retransmitted data which adversly affects these statistics in RTP.

RMFP also does not include a timestamp. This removes another problem with the use of RTP, since there is no implicit or explicit media playout model in RMFP and applications can use ADUs in any desired order. However, it introduces a problem - it is not possible to directly synchronise RMFP data with an RTP flow. An RMFP profile can specify a timestamp and playout model if it is desired for a particular class of applications.

Like RTP, the RMFP has separate flows for data packets and session messages. It also introduces a third flow for control packets. This allows control packets to be sent at any desired rate, rather than being limited to the session data rate as happens with RTP control packets.

All in all, we consider RMFP to be a much better fit for interactive media applications than is RTP, although both derive from the same philosophy of protocol design.

The Reliable Multicast Framework, RMF [4] attempts to solve a similar problem to RMFP. Like RMFP it defines common formats for data, control and session packets, but rather than defining a common core with a series of profile extensions, RMF focuses on the development of a ‘universal receiver’ which can speak any desired reliable multicast protcol. Quoting from [4]:

> “A receiver’s actions are specialized to conform to the requirements of a specific reliable multicast protocol as a consequence of the sender setting various fields in data packets appropriately and as a consequence of the session-level control protocols setting state information appropriately.”

As a consequence of this, it is difficult to categorize RMF since the behaviour can vary significantly. It does, however, have a number of features which are useful for interactive media applications and avoids many of the problems of using RTP in this area.

For example RMF includes bits in the header to indicate that this packet is a retransmission (to avoid confusion of reception quality statistics) and to allow for segmentation and reassembly of ADUs. It also provides for flexible reception quality feedback, avoiding the limitations of RTCP RR packets for this use, and for flexible repair strategies. Unfortunately, it still provides only a single sequence number and no explicit naming of ADUs, and no timestamp is included.

The opinion of the authors is that the RMF model is overcomplex for the gain provided, but it does explore a number of interesting concepts. Many, but not all, of the pitfalls of the use of RTP for interactive media are avoided.

6. CONCLUSIONS

We have presented a critical overview of the use of RTP for interactive media applications such as shared workspaces, networked games, or distributed presentation tools. Our conclusion is that RTP is not the correct protocol for most such applications. Rather, RTP may form one part for the needed protocol suite, supplemented by other protocols more tuned for the needs of different parts of an interactive media application.

Many applications are best designed around the use of multiple protocols: it may be that non-interactive media are transported by RTP, interactive media by another protocol, and bulk transfer of data is handled by some form of reliable multicast. This presents no problem to a well designed system, and allows for considerable flexibility, for example by transporting the different media on different transport addresses they can be assigned to a different QoS categories. It also allows for each media to be transported by a protocol optimised for that class of application, in accordance with the principles of application level framing [2].

Finally, it is our belief that the development of a common framing protocol for interactive media is of importance. There exists a great deal of common functionality between different application classes, and it is desirable to leverage that into a single framework if possible.

7. ACKNOWLEDGMENTS

This work grew of out of discussion within the European Commission Telematics for Research Programme, project RE4007. In particular, we wish to thank Martin Mauve for many stimulating conversations on this subject. We also acknowledge the careful review provided by Orion Hodson.

8. REFERENCES

[1] C. Castelluccia, C.-J. Villanueva, and T. Turletti. RMFP Profile for SRM. Work in progress (Internet draft), March

1996.

[2] D. D. Clark and D. L. Tennenhouse. Architectural considerations for a new generation of protocols. In *Proceedings ACM SIGCOMM'90*, Philadelphia, September 1990.

[3] J. Crowcroft, L. Vicisano, Z. Wang, A. Ghosh, M. Fuchs, C. Diot, and T. Turletti. RMFP: A reliable multicast framing protocol, March 1998. Work in progress (Internet draft).

[4] B. DeCleene, S. Bhattacharaya, T. Friedman, M. Keaton, J. Kurose, D. Rubenstein, and D. Towsley. Reliable multicast framework (RMF): A white paper, March 1997.

[5] S. Floyd, V. Jacobson, S. McCanne, C.-G. Liu, and L. Zhang. A reliable multicast framework for light-weight sessions and applications level framing. *IEEE/ACM Transactions on Networking*, December 1997.

[6] M. Handley and J. Crowcroft. Network text editor (NTE): A scalable shared text editor for the Mbone. In *Proceedings ACM SIGCOMM'97*, Cannes, France, September 1997.

[7] M. Handley and V. Jacobson. SDP: Session Description Protocol. IETF Network Working Group, April 1998. RFC2327.

[8] M. Handley and C. Perkins. Guidelines for writers of RTP payload format specifications. IETF Network Working Group, December 1999. RFC2736.

[9] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. SIP: Session Initiation Protocol. IETF Network Working Group, March 1999. RFC2543.

[10] I. Kouvelas, V. Hardman, and A. Watson. Lip synchronisation for use over the Internet: Analysis and implementation. In *Proceedings IEEE Globecom'96*, London, UK, November 1996.

[11] M. Mauve, V. Hilt, C. Kuhmünch, and W. Effelsberg. RTP/I - An Application-Layer Protocol for the Transmission of Interactive Media with Real-Time Characteristics. In *Proceedings IEEE Multimedia Systems*, 1999.

[12] S. Raman and S. McCanne. Scalable data naming for application level framing in reliable multicast. In *Proceedings ACM Multimedia'98*, Bristol, UK, September 1998.

[13] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *Proceedings IEEE INFOCOM*, Toronto, Canada, June 1994.

[14] H. Schulzrinne. RTP profile for audio and video conferences with minimal control. IETF Network Working Group, January 1996. RFC1889.

[15] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. IETF Network Working Group, January 1996. RFC1889.