Multimedia Protocol Decoding

Dragos Ilie

January 1999

Contents

Chapter 1

Overview

This is a Master's of Science thesis for the Telecommunication/Signal Processing program at the University of Karlskrona/Ronneby[1]. It's major subject is *multimedia protocol decoding*.

The project "Multimedia Protocol Decoding" is part of a larger project "Distributed Network Education Systems Enhancement through Measurements and Analysis (DNSM)" within the Sweden-Silicon Valley Link SSVL effort. One of the main topics of study for the DNSM project is on the provisioning of QoS guarantees when delivering audio and video over IP based networks.

This chapter will present the background for the thesis and will give some introductory information about multimedia network communications. For an introduction and further details on this topic please refer to [CWH+96].

1.1 Abstract

Early trials on delivering high-quality media over "long-haul" or WAN links have shown that delivering QoS for multimedia applications on IP networks is not just a matter of interconnecting all the "high-tech" pieces. In fact, each of the pieces must be individually optimized, not individually as isolated components, but as integral part of the whole. Furthermore, the picture is even more complicated because of many specific problems related to this application, e.g. "live" classrooms, large virtual class sizes, need for the ability of "continuous" feedback between the lecturer and the virtual class rooms, need for off-line access, etc.

This thesis will focus on measurements at the application level and show how the network characteristics affects the performance of multimedia applications. For example, an application transmitting MPEG video can have extremely poor perceived performance despite a relatively low packet (or cell) loose rate if the packets or cells lost are those containing the MPEG I-frames. Since other frames use the I-frames as predictors, the effects of a single lost packet can persist through several video frames. As another example, the perceived performance of a web browser depends on the total time to load an entire page including all inline images. Even though the transmission delay through the network of any

[1] *http://www.hk-r.se*

given packet might be acceptable, the total time required to complete all the HTTP transactions could be unacceptable.

We will analyze MBone multimedia traffic generated by network distributed education. To achieve this goal software will be programmed to decode MBone traffic, QoS metrics for each session type will be identified and the metrics will be graphically displayed in real time.

More specifically, a Java applet will integrate the user interface into the web browser window. The applet will acquire a set of required parameters and then pass them to a CGI script on the server side. The CGI script will use the parameters to construct calls to a network filter. The filter will pass back to the CGI script the results of the filtering operation, that is detailed information about the ongoing sessions. The information will then be listed in the browser window. By clicking on one of the list entries, the user will be able to start playback of a particular session. During playback, bandwidth utilization for different media, video and/or sound will be shown in real-time.

1.2 Background

For the last few years the number of hosts on the Internet has been growing at an exponential rate. Thousands of new users all over the world get connected to the Internet every week pushing this technology to its limits and sometimes beyond that.

The first applications of Internet were for textual transmission of electronic mail messages and for exchanging small amounts of information between academic sites. Soon graphical capabilities were added and the World Wide Web was born. Featuring a friendly user interface, at the beginning it was used as a means of conveying information between scientists at CERN. But to use the Web people didn't need to be scientists. The World Wide Web (WWW), allows the user to obtain information by clicking on hyperlinks of interest. A hyperlink contains necessary data for the Web browser to read documents located anywhere on the Internet. The information on a Web page consists of text, images and hyperlinks. While the text and images are the actual information parts of the Web page, the hyperlinks give the user the possibility to access other information, located outside the current page. This fast and easy access to information resulted into a explosion in the numbers of users who wanted Web access.

Today, the Web is slowly becoming a major information source in all domains and in all the parts of the world. HTML[2] has gained the ability to define dynamical Web pages that change each time one access them. There is a large amount of plug-ins[3] capable of presenting a multitude of information, from showing LaTeX pages embedded into HTML documents to acting as live radio receivers.

With Internet distances shrink significantly. People begin to weigh their options. Sometimes they find out that instead of flying several hours to meetings

[2]HyperText Markup Language - a collection of textual commands describing the web page appearance.

[3]A plug-in is a small program hooked up to the browser's API in the intent of extending the browser capabilities.

far away where trivial matures are discussed, they can do the same thing by the way of video conferencing.

But this it is not all. A lot of todays common media is also making its way to the Internet. People want fast access to information and they want to share it with others. Full-motion video on-demand and IP[4] telephony are seen as the drivers for several communication companies around the world.

As the user demands are increasing, so is the strain the data networks are experiencing. New methods of traffic shaping and traffic control are required in order to avoid network congestion followed by poor QoS and perhaps, total network collapse.

Only recently scientists have shown that data traffic is different than voice traffic (i.e. telephony). Voice traffic is a relatively well studied type of traffic, easily modeled by Markov processes. Theory has been developed, both for analysis and control. Data traffic on the other hand, at first may *look* similar to the voice traffic, but in-depth analysis reveals a very complex fractal behavior. As a result many of the traffic control systems today, systems assuming short-range traffic distribution, are not properly engineered and are unable to offer good QoS in the real-world [Pru95].

One goal for this thesis is to develop software as an aid in doing measurements on the Quality of Service (QoS) for MBone sessions. QoS is a set of parameters describing the quality of network service experienced by a end-user. The software that is to be developed will decode some of the commonly used audio/video protocols (RTP, RTCP, SDP, etc.) and display the application-level characteristics of the data flow. Some of the characteristics are bandwidth utilization, packet loss, packet delay etc.

1.3 Multimedia networks

According to PC Webopedia[5] multimedia is "The use of computers to present text, graphics, video, animation and sound in an integrated way. (. . .)".

A big challenge when designing a multimedia system is getting it accepted by the targeted user group. A required step in this this direction is making sure your system is using some standardized way of creating, collecting, accessing and distributing data. This thesis is concerned more with the data transport (access and distribution) across computer networks and much less with data creation and collection. Accordingly, from now on when discussing multimedia systems, the text refers to data transport unless otherwise specified.

Two of today's most established standards for carrying multimedia throughout the network are MBone and H.323. Both of them rely on similar methods to get the job done. We will only discuss details about MBone in this thesis.

MBone is an acronym for *Multimedia Backbone*. It's purpose is to define common means for distributing multimedia using IP Multicast as the carrier. For information about IP Multicast see [Ste94]. MBone has been allocated the IP multicasting addresses in the range 224.2.xxx.xxx [Kum97].

There can be many MBone sessions simultaneously active around the world. Details about each session is multicast between MBone servers, along with information about which server to connect and participate in the session. The

[4] Internet Protocol - the *de-facto* standard for transporting data over Internet.
[5] *http://webopedia.internet.com*

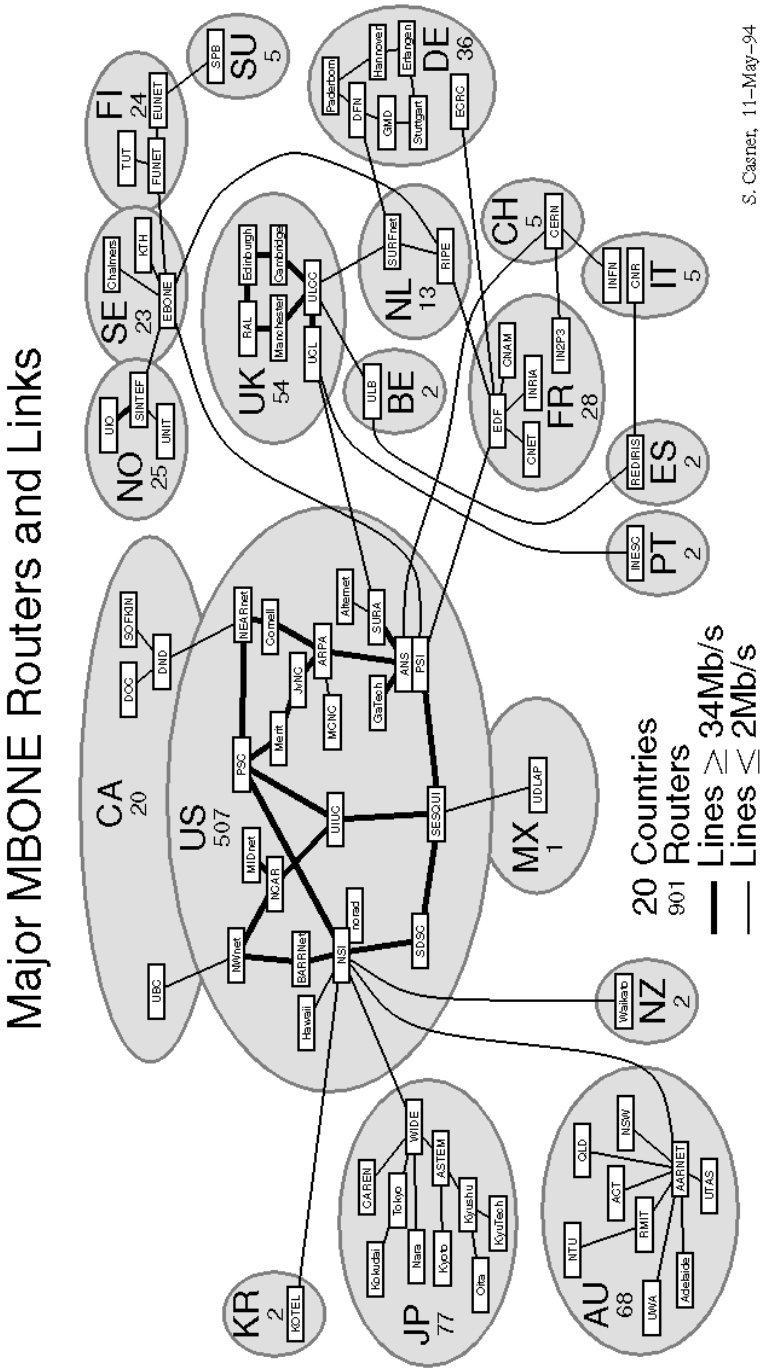

Figure 1.1: MBone Topology [Kum97]

information about each session can be obtained by any user with programs similar to SDR[6].

Session Directory (SDR) lists all the sessions announced by the server together with the time and date when they will be active. In addition to that, the SDR GUI allows the user to easily join or leave the session.

MBone acts as the transportation entity for a given data set. What data is transported and how the data is interpreted is not MBone's concern. The software running on top of MBone has the responsibility to fill enough information inside the session announcement in such a manner that a receiving part can identify the data flows. Based on this identification, SDR can automatically launch specific applications that interpret the data flows and present the interpretation to the user.

Presently, there are three common types of data carried by MBone. First, there is audio, where users communicate using microphones or similar devices that transforms voice and sound into digital data. Secondly, there is video, generated usually by live video cameras. Other means of generating video data can also be used, most notably, playback devices such video and DVD players. Third, there are whiteboard applications. Whiteboard applications allow users to work simultaneously and interactively on the same document, in real-time.

When the user has joined a session, SDR will collect the information about the type of available data streams (i.e. video, audio, whiteboard) and automatically launch the necessary applications to process these streams. Common applications include Video Conference (VIC), VAT, Robust Audio Tool (RAT) and WhiteBoard (WB). The users have the freedom to choose the type of applications they would like to use and also, what type of data streams they want to receive. An example showing when this type of freedom is useful is when a low-bandwidth user will choose to receive only the audio part of a session. This allows him to participate in the session without forcing the high-bandwidth users to give up the video transmissions.

For MBone, the data streams are carried by Real-Time Transport Protocol (RTP) packets. Control information about the RTP packets, such as QoS and number of users in each session, is supplied by the RTP Control Protocol (RTCP). Details of RTP/RTCP standard can be obtained from [SCFJ96].

1.4 Multimedia tools

Today, there is a large variety of software aimed at generating multimedia over IP. Unfortunately, almost every software package uses some specific protocol not known by the others. MBone and H.323 are attempts to create a standard for multimedia transport.

This section will first present some of todays most popular tools used to access multimedia over Internet. Secondly, it will present the available means of generating artificial multimedia traffic.

Multimedia tools can be roughly divided into two main categories: audio capable only and both audio and video capable tools[7]. Also, they can be interactive or non-interactive. Interactive tools rely heavily on the user input as

[6] MBone related software can be found at *http://www-mice.cs.ucl.ac.uk/multimedia-/software/*

[7] Note that there are other tools such as whiteboard etc. which we will not elaborate upon.

INTERNET PHONE	Interactive	*http://www.vocaltec.com*
Very advanced, H.323 capable software for IP-telephony from VocalTec It can do conferencing from IP to PSTN by using gateways. It handles video and whiteboard also. Some quick testing shows that high-quality video demands 100-120 kbps. Recently VocalTec and Cisco made an agreement to develop interoperability between their IP telephony solutions based on H.323 Version 2 international standard. Also, Telia Light has chosen the VocalTec Ensemble Architecture as the foundation for their IP telephony services		
NETMEETING	Interactive	*http://www.microsoft.com*
Microsoft Netmeeting is used for multimedia conferences. Uses the H.323 protocol. Measurements at Qmetrix Inc, using Intel Connection Advisor, show that a audio+video session uses 80-120kbps bandwidth.		
VAT	Interactive	*http://www-nrg.ee.lbl.gov/vat/*
Offers same services as RAT		
RAT	Interactive	*http://www-mice.cs.ucl.ac.uk /multimedia/software*
Robust Audio Tool (RAT) is the audio tool often used for MBone conferences. Data is encoded with 64kbps PCM. Measurements at Qmetrix International Inc. show a variation between 60-70 kbps in the bandwidth usage.		
REAL PLAYER	Non-Interactive	*http://www.real.com*
Real Player is one of the most popular tools today for network radio Typical bandwidth usage is 16-56kbps but has the capability to go up to 10 Mbps in LAN environments. The server is the entity which controls the bandwidth usage. Real Player sessions use RTSP on top of TCP, UDP or HTTP and PNA. RealPlayer also handles audio and video sessions well.		
VIC	Interactive	*http://www-mice.cs.ucl.ac.uk multimedia/software/*
Video Conferencing Tool (VIC) is used for receiving video from MBone sessions. Measurements at Qmetrix International Inc. on typical network education session show bandwidth usage of 120-700 kbps.		
VIDEO PHONE	Interactive	*http://support.intel.com/support videophone/trial21*
Similar to INTERNET PHONE. Is H.323 capable and can do video. Can even do H.323 through a firewall.		

Table 1.1: Multimedia tools

opposed to non-interactive which do not require any user input.

Table 1.1 lists some of the most common tools used to generate and transport multimedia over the Internet.

For a network designer it is important to test the effects of various network traffic on the designed network. For this purpose the designer would like to use artificial traffic generators, that is software capable of creating network traffic with user set properties. Two important traffic classes one would like to test are Constant Bit Rate (CBR) and Variable Bit Rate (VBR). VBR traffic can be used to model video while CBR traffic is useful for audio traffic modeling.

Unfortunately, there is a shortage on true multimedia traffic generators. Some traffic generators claim they can simulate real-time VBR traffic and that this can be used to model video traffic. In other words they treat multimedia as a special case of VBR and it is the user's job to find a good VBR model for the multimedia traffic.

A research team at the Ohio State University has modeled MPEG2 compressed video as piecewise CBR, long-range dependent rate. They are combining fractional Gaussian noise (FGN) sequences generated using fast Fourier transforms to produce long-range dependent traffic which models multiplex MPEG-2 video over VBR. Their results can be found at http://www.cis.ohio-state.edu-/~jain/atmf/a97-0177.htm.

The "Pseudo Random Traffic Generator" at http://www.caip.rutgers.edu-/~jpanchal/comm_proj_rpt/index.html says it simulates Poisson and TES (Transfer Expand Sample).

At Dublin City University, Ireland the video source was modeled as a discrete state Markov chain. Details at http://www.eeng.dcu.ie/~murphyj/publ/video-/video.html

1.5 Goals to achieve

The first goal is to create software able to discover MBone sessions in the traffic flow. This is also a requirement for fulfilling the other goals.

The software will interact with the *Niksun Traffic Recorder*. The traffic recorder is connected to all the interfaces in a local area network (LAN) and can "see" all the data traffic in that LAN. The observed data traffic can be recorded for later reference on a storage device such as hardisk or tape. The recorder's API[8] allows to retrieve stored data traffic according to certain criteria (e.g. time interval, packet type, IP address, port number etc.)

Using the *Niksun Traffic Recorder* we want to extract MBone session information (i.e. session announcements) and to playback the MBone data flows for a specific session.

Before we can playback the session, we must first identify the packets (audio and/or video) belonging to that session. This means we must find a set of qualifiers unique for those packets. The qualifiers can be constructed from the session announcement which contains all information required to access the server which transmits the data.

The next step is to apply the qualifiers to the traffic recorder and direct the traffic flow from the recorder to the playback client applications, VIC and VAT

[8] Application Programming Interface - a set of function calls to the application. It gives the possibility to control the application or add new functionality

or RAT.

Ranking the QoS by only looking at the playback video or listening to the playback sound is a very subjective measurement. Having statistical data as support for user observations will give a much more fair picture of the QoS. Not only that, but it will also help the designer to find the weak points in the system.

Finding the weak points in the system is the first step towards identifying the QoS parameters. These parameters are the factors affecting the quality of the session and are the key factor in obtaining QoS metrics.

Shortly, the goals to achieve are the following:

- Software to discover MBone sessions
- Software to playback MBone sessions
- Software to extract statistics
- Statistical analysis and QoS parameters identification
- QoS metrics identification

Chapter 2

Stream identification and data extraction

In this chapter we will present a fairly detailed description of the protocols used for MBone communications. Key fields in the packet headers will be examined along with the entity interaction. Also, we will present the MBone related software developed as part of the thesis.

2.1 MBone

MBone communications is a three step processes:

1. Identify the desired among all the sessions announced simultaneously
2. Interpret the information in the session announcement
3. Use the interpreted information to connect to the media streams

There are two ways to announce MBone sessions. One is the *Session Announcement Protocol* (SAP) and the other is the *Session Invitation Protocol* (SIP). The main difference between the two is that while SAP happily lists all the announced sessions to everybody who wants to read them, SIP will send this information to a specific set of users only. This thesis will focus on SAP. Details about SAP and SIP can be found in [KMKW98, HJ98, CWH+96].

Each session announcement contains important information, such as a session name, session owner, time and date when the session is active, and most important where to connect in order to receive the media streams. All this information is encoded using the *Session Description Protocol* (SDP).

The media streams use the RTP/RTCP protocol to carry data to the receiver. The RTP stream is the actual multimedia carrier, while the RTCP stream carries receiver reports to the sender.

2.1.1 SAP

The Session Announcement Protocol multicasts announcements periodically on a well-known address and port combination. The time period between two

announcements of the same session depends on the *Time To Live* (TTL) of the session and the total number of sessions announced [CWH+96]. The announcements are SAP packets containing SDP encoded information. The SAP header[KMKW98] is shown in Figure 2.1.

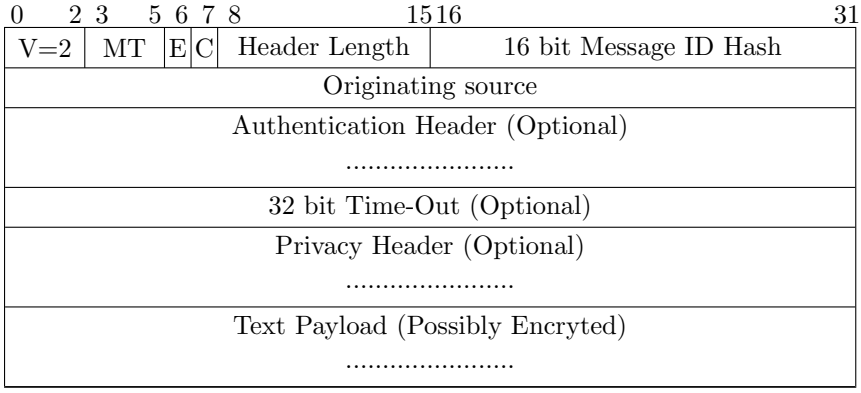

Figure 2.1: SAP Packet Format

- Version Number (V), bit 0–2, carries the SAP version used for the packet. This text describes SAP version 2, abbreviated as SAPv2.
- Message Type (MT), bit 3–5, describes the packet payload. Currently, there are only two possible types of packets: Session Descriptor Announcement Packet and Session Description Deletion Packet. As names implies the first one will announce a session, while the second one informs that the session no longer exists.
- Encryption Bit (E), bit 6, is set if the text payload has been encrypted.
- Compression Bit (B), bit 7, is set if the text payload has been compressed using the gzip[1] utility.
- Header Length, bit 8–15, is defined by [KMKW98] to be the number of 32 bit words following the main SAP header that contains authentication data. It is set to zero if no Authentication Header is present. In [CWH+96], where SAPv1 is described, it is called *auth len*.
- Message Identifier Hash, bit 16–31, is used as a globally unique id identifying the precise version of this announcement. The value in this field changes if any field of the session description changes. The message should always be parsed if this field is zero.
- Originating Source is a 32-bit field containing the IP address of the original source of the message. This field can some times be zero for backward compatibility with SAPv0 clients.

[1] gzip is described in RFC 1952

The optional fields are used when the text payload is authenticated or encrypted. Because there is no way to extract the text payload if the packet has been encrypted, the text will not discuss these type of packets. In other words, only packets containing the first two 32-bit fields and plain text payload are considered. This is the case when both (E) and (C) fields are both set to zero.

2.1.2 SDP

SDP is designed to convey conference setup information and transports this information using SAP, SIP, RTSP[2], HTTP or electronic mail with MIME extensions. When SAP is used, only one session description is allowed per packet [HJ98].

A media session, in the SDP context, is defined as a set of media streams existing for some duration of time. Media streams can be unicast or multicast. SDP provides for each stream information about media type (video, audio, whiteboard, etc.), format or encoding protocol (H.261[3], PCM[4], etc.) used and transport protocol. For IP, the transport protocol is generally composed of IP address and port [HJ98]. For encoded sessions, the media stream information is encoded also, and one will need to have the appropriate cryptographic key to access it. An encrypted session has the (E) bit in the SAP header set to one.

Each line in the SDP packet text payload describes one attribute of the session or media type used in the session. The line format is <*type*>=<*value*>, where <*type*> is exactly one character and is case-significant. The <*value*> field is a structured text string whose value depends on <*type*>. There is no space between text on either side of the the = sign and the sign itself [HJ98].

A session description consists of a session-level description and zero or more media-level descriptions. The session-level description starts with a *v*= line and continues to first media-level description which starts with *m*=.

The <*type*> descriptors used at session level and at media level are presented in Table 2.1 and Table 2.2 respectively.

The *c*= descriptor at session-level becomes optional if included at media-level for all media types. At media-level, the *c*= descriptor becomes optional if already used at session-level and its use will override the session-level connection information. For additional information regarding session descriptors see [HJ98].

2.1.3 RTP/RTCP

The Real-time Transport Protocol (RTP) "provides end-to-end delivery services for data with real-time characteristics, such as interactive audio and video. Those services include payload type identification, sequence numbering, time stamping and delivery monitoring" [SCFJ96].

RTP expects the lower layers to provide all QoS guarantees and does not provide any itself. RTCP provides QoS of monitoring to RTP and information about participants in an on-going session.

The RTP specification is left intentionally incomplete to allow applications to tailor it to their needs. A complete specification requires a profile specification document and one or more payload format specification documents. The profile

[2] Real Time Streaming Protocol is used to maintain real-time sessions
[3] H.261 is a ITU-T standard used for video encoding
[4] Pulse Code Modulation (PCM) is a method used to sample analog signals (e.g audio)

v=	required	SDP version number
o=	required	owner/creator and session identifier
s=	required	session name
i=	optional	session information
u=	optional	URI of the description
e=	optional	e-mail address
p=	optional	phone number
c=	required	connection information if not included at media level
b=	optional	bandwidth information
z=	optional	time zone adjustments
k=	optional	encryption key
a=	optional	zero or more session attribute lines
t=	optional	time the session is active
r=	optional	zero or more repeat times

Table 2.1: SDP session-level descriptors

m=	required	media name and transport address
i=	optional	media title
c=	required	connection information if not included at session level
b=	optional	bandwidth information
k=	optional	encryption key
a=	optional	zero or more session attribute lines

Table 2.2: SDP media-level descriptors

specification defines the mapping from a set of payload codes to payload formats. A payload format specification defines how one payload is to be carried in RTP [SCFJ96].

The RTP header is shown in Figure 2.2.

- Version Number (V), bit 0–1, carries the RTP version number. This text describes RTP version 2, RTPv2.
- Padding (P), bit 2, is set if the packet contains one or more padding octets at the end which are not part of the payload.
- Extension (X), bit 3, the packet contains a header extension.
- CSRC count (CC), bit 4–7, contains the number of CSRC identifiers that follow the fixed header. A CSRC is a source of a stream of RTP packets that has contributed to the combined stream generated by an RTP mixer[5] [SCFJ96].
- Marker (M), bit 8, is profile dependent. It can be used to mark significant events in the packet stream.
- Payload type (PT), bit 9–15, carries the payload code for a payload format.

[5]The RTP mixer combines together packets from different sources

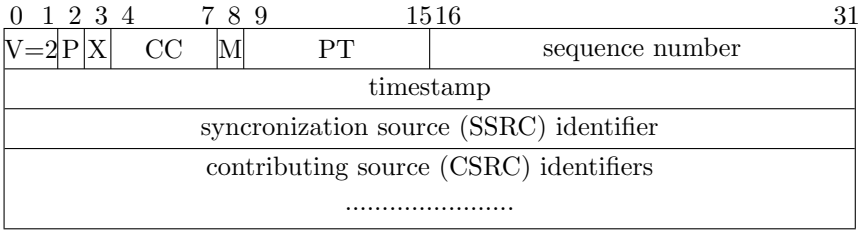

Figure 2.2: RTP Header Format

- Sequence number, bit 16–31, is incremented by one for each RTP packet sent. The receiver can use this information to detect packet loss.
- Timestamp, 32 bits, contains the sampling instant of the first octet in the payload.
- Synchronization source identifier (SSRC), is a 32-bit number identifying the source of a RTP packet stream.
- Contributing sources list (CSRC), contains up to 15 contributing sources. Additional sources do not have place in the header and are lost.

RTCP periodically sends control packets to all participants. There are five types of RTCP packets: sender report (SR), receiver report (RR), source description items (SDES), end of participation notification (BYE) and application specific functions (APP). The rest of the section will present the SR packet. For additional information see [SCFJ96].

The RTCP packet, Figure 2.3, consists of three sections, namely the header, the sender info and zero or more reception report blocks. The sections are separated in Figure 2.3 by thick lines. A fourth section, the profile specific extensions, might exist if the profile defines it.

- Version Number (V), bit 0–1, is set to 2 for the version described here, RTCPv2.
- Padding (P), bit 2, is set if the packet contains one or more padding octets at the end which are not part of the payload.
- Reception report count (RC), bit 3–7, carries the number of reception blocks contained in this packet. A value of zero is valid.
- Packet type (PT), bit 8–15, contains the packet type code. It is set to 200 for SR packets.
- Length, bit 16–31, contains the length of the packet counted as the number of 32-bit words minus one.
- Synchronization source (SSRC), 32 bits, is the identifier for the sender of this packet.
- NTP timestamp, 64 bits, carries the NTP timestamp when the report was sent. Along with timestamps from other receivers it can be used to measure round-trip propagation to those receivers.

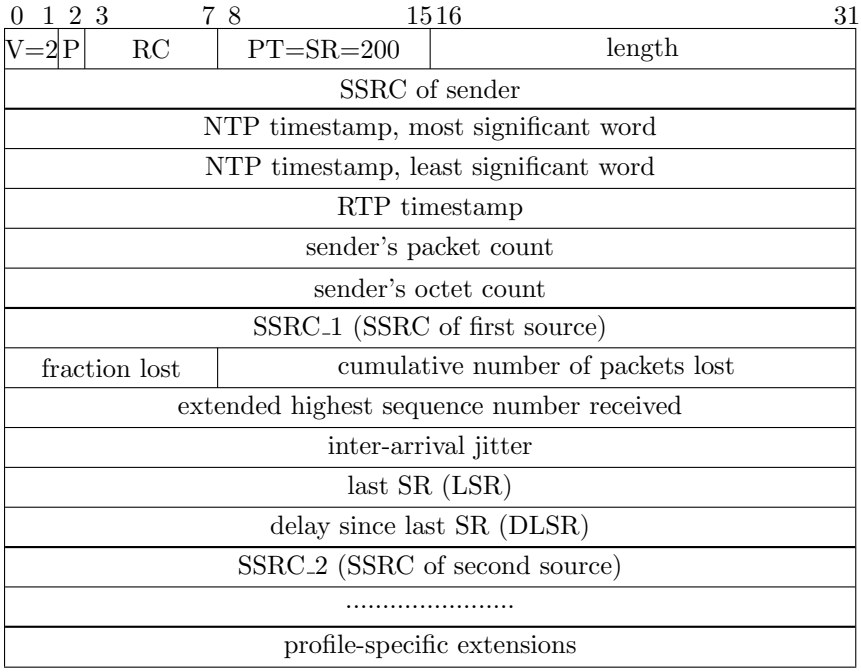

Figure 2.3: Sender report: RTCP Packet Format

- RTP timestamp, 32 bits, is the same as the NTP time except for the time units and the random offset. They are the same as those used in the RTP timestamps in data packets.
- Sender's packet count, 32 bits, carries the total number of RTP data packets transmitted by the sender from the beginning of the transmission up to the time when the current SR packet was generated. The count is reset when the sender SSRC changes.
- Sender's octet count, 32 bits, is the total number of payload octets in data packets transmitted by the sender from the beginning of the transmission up to the time when the current packet was generated. The count is reset when the the sender SSRC changes.
- Synchronization source n (SSRC_n), 32 bits, marks the start of the reception block for the SSRC_n.
- Fraction lost, bit 0–7, is the fraction of packets sent by SSRC_n, lost on the way to receiver since the previous SR or RR was sent.
- Cumulative number of packets lost, bit 8–31, is the number of RTP data packets from SSRC_n lost since the beginning of reception.
- Extended highest sequence number received, 32 bits, carries in the low 16 bits the highest sequence number received from SSRC_n. The high 16 bits contain the count of sequence number cycles which may be maintained, as explained in [SCFJ96], Appendix A.1.

- Inter-arrival jitter, 32 bits, is the smoothed absolute value of the difference in packet spacing at the receiver compared to the sender for two consecutively received packets. The packets may be duplicates or arrive out of sequence at the receiver.
- Last SR timestamp (LSR), 32 bits, carries the middle 32 bits of the NTP timestamp belonging to the most recent RTCP SR packet from SSRC_*n*.
- Delay since last SR (DLSR), 32 bits, is the delay, expressed in units of 1/65536 seconds, between the last SR packet received from SSRC_*n* and the time this packet was sent. This can be used to calculate round propagation delay to the receiver.

2.2 The session master—*sdrdump*

One of the goals for this thesis was aimed at developing software that would work similar to SDR. The software would work as a plug-in for the Niksun[6] network monitor. The monitor collects and records information from all network layers. The interface is Web-based, making the monitored information available to any connected host with a Java capable Web browser.

The first step was to implement the SAP and SDP protocols. The program would collect information about all MBone sessions announced during a time interval. This information would be dumped on the standard output from where it could be redirected to a file. Because this functionality is somehow similar to that of the *tcpdump*, this software was named *sdrdump*.

Mr. Andrew Heybey, one of the Niksun monitor creators, developed initial source code for a program called *play_mbone*. This program, given the source IP address and port number and a time interval, would playback the traffic qualified by these variables. Because this was initial alpha code[7] the user was required to know all details about how to playback the stream. The objective was to automatize *play_mbone* to such a degree that the user will just click on the desired session and the playback will automatically start.

SAP is multicasting session announcements on IP multicast address 224.2.127.254 and UDP port 9875. Therefore, *sdrdump* interfaces with the monitor by retrieving all data packets sent from the IP address and port stated above during a user specified time interval. If no packets are found, *sdrdump* exits gracefully.

If data packets are found, *sdrdump* first checks these packets to make sure they are SAP packets and if true interprets the SDP coded data. The results are presented in a human readable format on the standard output, usually the screen console. The output can be redirected to a file for later viewing.

The command line for *sdrdump* is: `sdrdump [-B begin time] [-E end time] [-i interface] [-r recorder] [playback host] [playback port] [data qualifiers]`

`begin time`: can be expressed as the absolute time and date as in `'Thu Jul 28 1998 23:17:00'`, or as a subset of that `'Jul 28 23:17'` or relative to *now*

[6] http://www.niksun.com

[7] Alpha code is a label used by programmers to state that the software is not tested, not all features are fully implemented and that there is a high probability to encounter bugs. This type of software is usually released only for testing purpose.

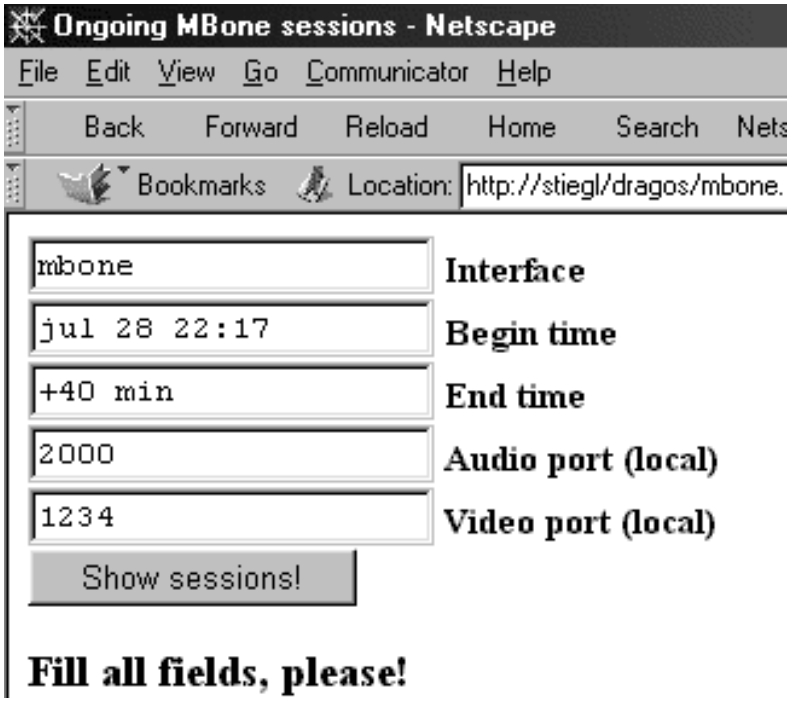

Figure 2.4: SDR form

as in `'+10 min'` or `'-1 day'`. The *now* variable is equal to the time of the query. Observe that the time and date are enclosed in simple quotes to escape the spaces between the strings. This prevents the shell from using the spaces inside the quotes as argument separators.

`end time`: similar to `begin time` (e.g. `'+3 hours'`).

`interface`: the name of the interface which records or has recorded the data. Usually the same name as the network interface (e.g. `/dev/eth0` would become interface `eth0`), but exceptions from the rule are allowed.

`recorder`: the network name of the monitor that records or has recorded the data (e.g `nm.hk-r.se`).

`playback host`: kept for historical reasons. Must be filled in (i.e. use a dummy) but will not be used by current *sdrdump* implementation.

`playback port`: kept for historical reasons. Must be filled in (i.e. use a dummy) but will not be used by current *sdrdump* implementation.

`data qualifiers`: define the filter used by the monitor to find the desired data packets. For *sdrdump* it is: `host 224.2.127.254 and port 9875`.

A usage example where the results are redirected to a file called `sessions.txt` is presented below.

```
sdrdump -B 'Jul 28 1998 23:17' -E '+10 min' -i eth0 -r nw.hk-r.se
myhost 0 host 224.2.127.254 and port 9875 > sessions.txt
```

A CGI script is used to present the results to a Web browser. The script, called *sdr* presents a web form to the user, see Figure 2.4.

The form fields, with the exception of the last two, contain the command

line arguments discussed previously. The last two fields are used to pass information about where the audio and video part of the session are to be played. It is assumed that VIC and RAT or VAT are listening on the video and audio playback port respectively.

sdr, the CGI script, reads the output from *sdrdump* and finds out among other things what type of media each session supports. When all the output has been processed, *sdr* lists all the sessions together with the relevant information. The output from a *sdr* listing is presented in Figure 2.5.

The name of each session is actually a hyperlink. If the user clicks on it, and client applications are listening on the playback ports then the MBone session will be played. Figure 2.6 shows an example of session playback.

2.3 Session playback—*play_mbone*

Having all the sessions listed and information about how to connect to each session, the next step is to actually playback the desired session.

The *play_mbone* software used is the original program designed at Niksun with small enhancements such as statistics computing.

The solution used here makes use of a Java applet, the *Spy applet*. This Java applet connects to *play_mbone* and opens a communication channel. This channel allows passing certain parameters to *sdr*, the CGI script, and also to send more information to the user, for instance real-time statistics. Currently, the *Spy applet* can display real-time bandwidth usage for audio and video.

Spy applet starts when the user clicks on a session name. It retrieves the user host IP address and dynamically allocates a TCP port, the *Spy server* port. Then it calls a CGI script, called *play_mbone.pl* and passes it all parameters needed: host IP address, TCP port, session IP multicast address and a lot of other information. This is the same information as presented by *sdr*.

The reason for using the CGI script as an intermediary step in starting playback is that Java applets, with very small exceptions, are not allowed to start processes. This is a security measure to prevent malicious applets from destroying and modifying data without having the user's consent.

Play_mbone.pl uses the parameters it received from *Spy applet* and constructs calls for the *play_mbone* application. For each media, audio and/or video, the script will launch two *play_mbone* processes, one for the RTP connection and one for the RTCP.

When *play_mbone* starts it checks first the command line. The command line is similar to the one used for *sdrdump*: `play_mbone [-B begin time] [-E end time] [-c] [-i interface] [-r recorder] [playback host] [playback port] [spy port] [data qualifiers]`

All parameters are identical as those used for *sdrdump* with the exception `[-c]` and the `[spy port]`.

`-c`: appears on the command line if the *play_mbone* will playback the packets from a RTCP flow. The switch, among other things prevents *play_mbone* from computing statistics meant for the RTP data packets and communicating with the applet too often. One of the reasons for doing this is to prevent the *Spy applet* from providing poor performance to the user.

`spy port`: is the communication port used by the *Spy applet*.

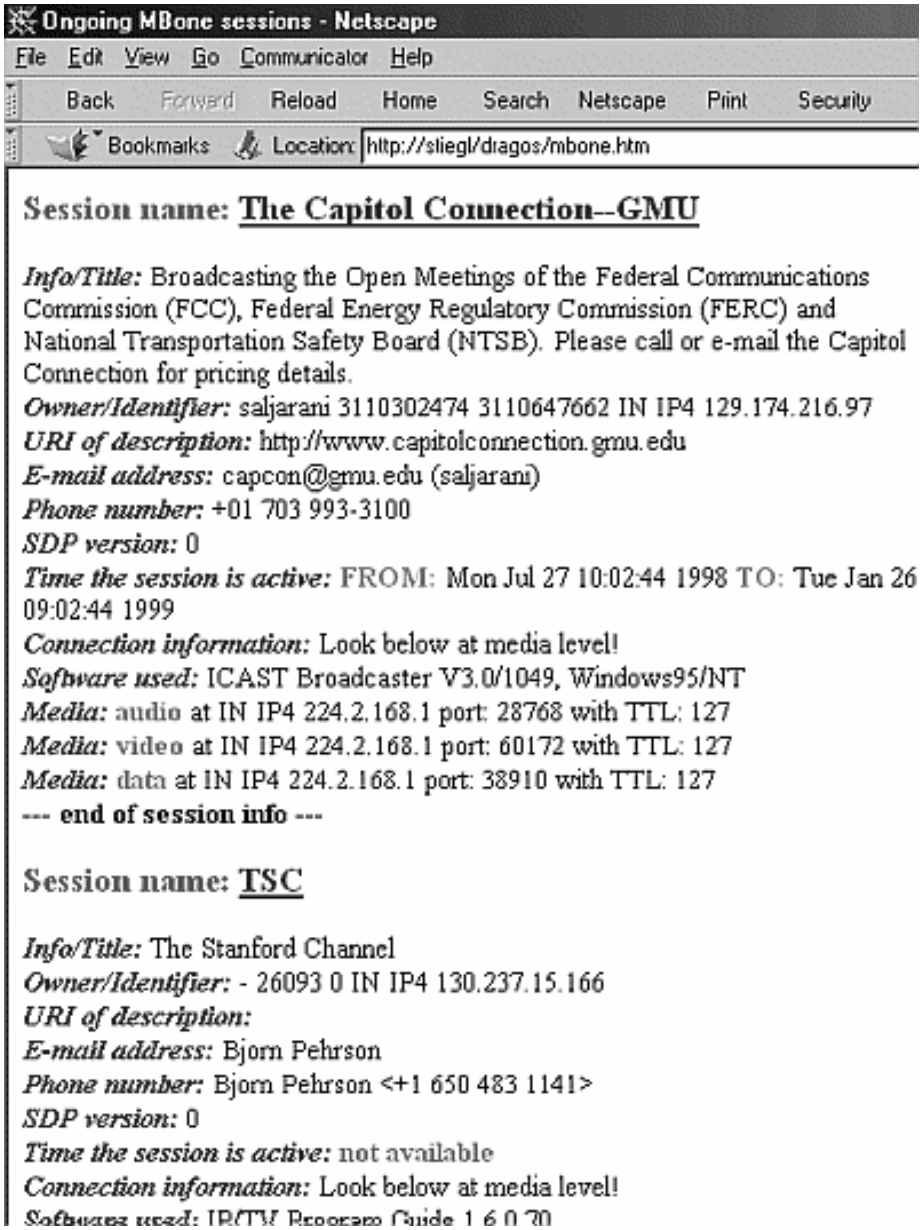

Figure 2.5: SDR list

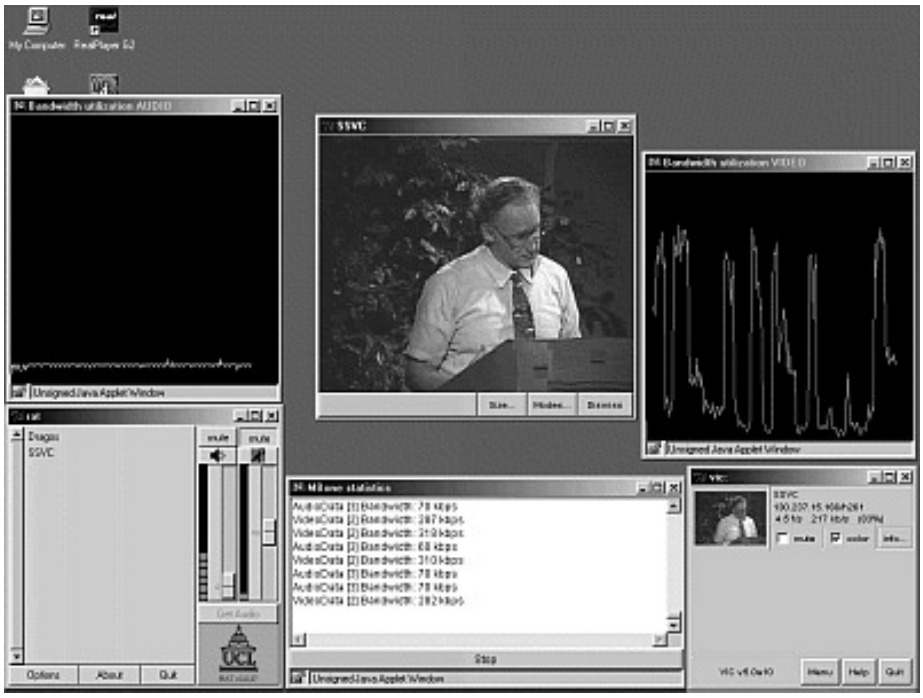

Figure 2.6: MBone playback

After the command line has been processed *play_mbone* starts to set up the communication sockets. It tries to connect one socket to the client application (i.e. VIC or RAT) and another to *Spy applet*. If any of these fail, *play_mbone* will abort, otherwise it uses the data qualifiers to build a filter. The filter is passed to the monitor for further processing. The monitor tries to find a data set matching the filter description and if successful uses a callback function passed by *play_mbone* to send data packets to it.

For each packet *play_mbone* examines the header inserted by the monitor and reads the timestamp. Having this information, it computes the original inter-arrival delay from this information. Then, before sending the packets to the playback host, it delay them by the same amount of time. This way it recreates the the same QoS as experienced at the time the MBone session was monitored.

Chapter 3

Quality of Service

Quality of Service (QoS) is defined to be a measure of the network performance perceived by a end-user. Despite the simplicity of the this definition, QoS is very difficult to measure. One problem arise when deciding what actually is bad, or good, performance.

Most of the QoS measurements executed in the past focused on traffic at the data link layer or at the network layer. More specific these measurements reflected the number of packets lost and the average delay experienced by traffic on the layers we mentioned. The statistics achieved were used to predict future traffic performance on those layers.

It is our intention to accomplish a different type of measurements. We believe that traffic measurements at the low layers do not give an accurate picture of the QoS experienced by an end-user. Instead we will measure application layer performance directly. After all, the user works at the application layer and any performance problems are observed there.

This chapter will present QoS measurements performed at the application layer with the MBone playback system. We have chosen to perform these measurements on one video/audio session containing a forty minutes long seminar. In our opinion this is a typical session for network distributed education.

3.1 Traffic analysis

Traffic analysis is a key factor in network design and maintenance. Not only does it help with the identification of problems within the network, but it can also provide actual traffic data to verify traffic theories and to develop more accurate traffic models. The models can be used to improve network control mechanisms, to test new theories and methods and to predict future traffic.

This part of the thesis will concentrate on traffic analysis of multimedia. It has been difficult to provide good QoS for multimedia traffic over packet or cell networks in a cost effective manner. The problem is that multimedia sources generate large amounts of data in bursts. For example, variable rate (VBR) video uses a special compression algorithm. The algorithm sends only the portion of the image frame that have changed since the previous frame. Unless the video camera is recording a totally static environment, any movements in the image focus will generate packet bursts. The size of the packets is variable

because large image changes generate large amounts of data while small changes generate small amounts of data.

The multimedia packets are both loss and delay sensitive. While video might still be able to present a frame with acceptable quality if no vital packets have been lost, audio will most likely break up or generate disturbing "pops" at the receiver if packets are lost. This is also true for delays except that here even video will rapidly lose image quality. This a result of something called the *playback point*. Multimedia network applications are aware that packets inside the network experience a certain amount of delay. Consider a packet carrying 100ms of speech. If the packet is delayed by approximately 300ms then the application will most likely discard this packet and play the ones situated in time closest to present. To avoid these situations the application keeps a buffer where it stores a certain amount of data before starting to play. This way, even if packets are delayed in the network, there are still packets in the buffer that can be played. The buffer size, counted in units of time, is called the playback point. While the playback point can be a solution to the delay problem it introduces another problem: what value should the playback point have? Delays in excess of the playback point would most likely result in degraded quality. Not even the video compression algorithm can cope with this. As stated before, the algorithm optimizes the network traffic by sending only the change between previous and current frame. It cannot do much about delays. The decoder will most likely wait passively for the next packet carrying frame information and keep the display in the same state until it gets it. This will result in jerky movements.

For a network designer wishing to setup an average or large size network ensuring that all the constraints are satisfied for an end-to-end good QoS of multimedia traffic can be a nightmare. Due to the bursty nature of multimedia traffic, the network resources can be best utilized by using the silence periods between bursts to send data. Unfortunately, the behavior of multimedia traffic makes it very hard to actually discover these silence periods. There are some traffic models based on fractal mathematics that better characterize multimedia traffic flows than the conventional methods but there remains a lot of work to do before a workable solution can be obtained.

In our experiments we wanted to see how packet loss and packet delays affects the QoS from a user's subjective point of view. Also, we wanted to discover the importance of bandwidth size and if there is a correlation between any of these factors and the perceived performance. To do this a small scale QoS test has been conducted as part of this thesis. One person listened to a MBone session of 40 minutes length and marked the time intervals when the sound quality was bad. The test was conducted first using 60 seconds time intervals and than stepped down to 10 seconds intervals. More information about this test is provided in the following sections.

Even if the test is subject to the person ranking the audio quality, it can be used as a crude QoS measurement at application level before better QoS metrics are found.

3.2 Bandwidth measurements

In this section will present the bandwidth measurements obtained for a sample session with *play_mbone* and *Spy applet*. For more information about these

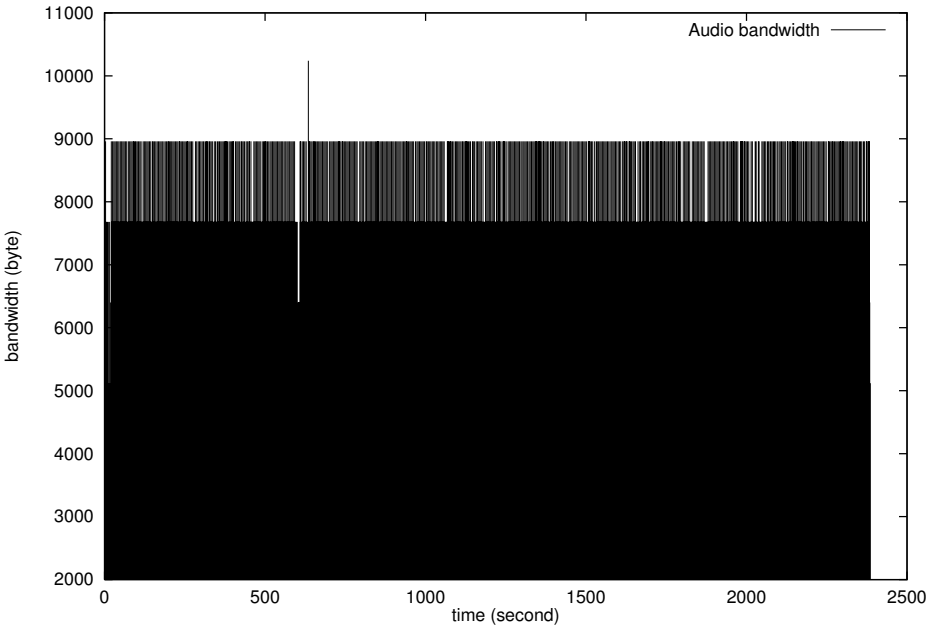

Figure 3.1: Audio bandwidth

applications please refer to Section 2.3.

Niksun Inc. had recorded previously a MBone video session on a T1 link. They kindly let me use their raw data. From this data the MBone traffic stream was extracted. The application used for this purpose was *play_mbone*. *Play_mbone* is computing statistics every time a new packet is retrieved from the monitor. The bandwidth computation is carried out by first acquiring the packet size at IP layer and then subtracting the IP, UDP and RTP header. This computation is executed only for RTP data packets to save processing power because we must run simultaneously between two and four instances of *play_mbone*. There will be one instance for each RTP or RTCP we receive.This is where the `-c` switch (see Section 2.3) comes to use.

The *Spy applet* and *play_mbone* share a TCP channel. *play_mbone* uses the channel to send bandwidth measurements to *Spy applet*. The applet presents the measurement graphically in real-time, as seen in Figure 2.6, in the top left and top right window.

A more detailed graph of the audio bandwidth usage is shown in Figure 3.1. On the horizontal axis we have the time scale in units of a second. The vertical axis designates the bandwidth usage in bytes. We can see that it has a rather flat amplitude, situated near 9000 bytes/second suggesting constant bit rate (CBR) behavior. The application used to receive the audio was RAT, using PCM μ-law encoding.

A T1 link, as the one where the MBone session was recorded, has a data rate of 1.536 Mbps. The average link utilization for the audio stream is then close to 4.7%.

Figure 3.3 is showing the video bandwidth usage. Video was encoded using

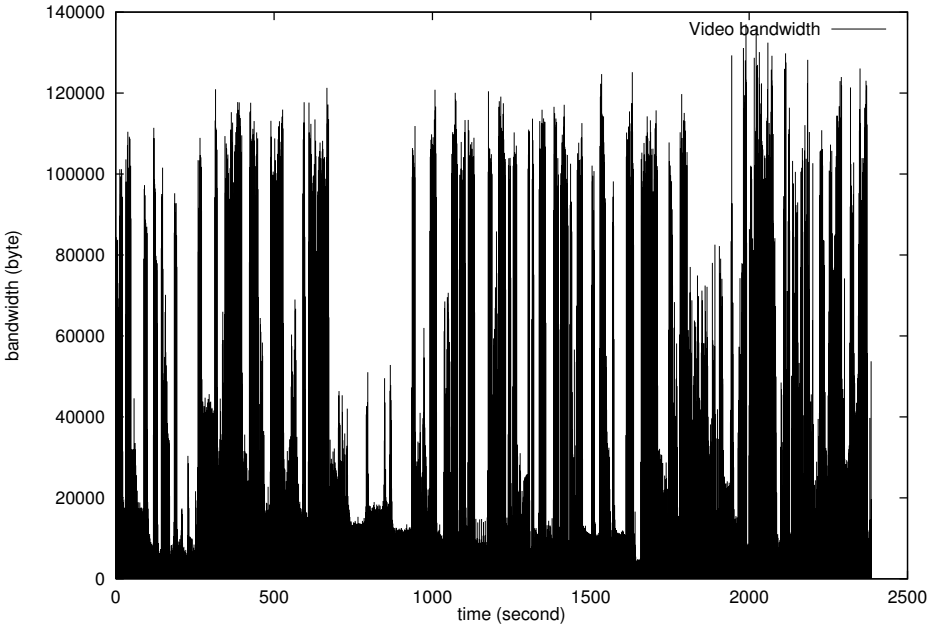

Figure 3.2: Video bandwidth

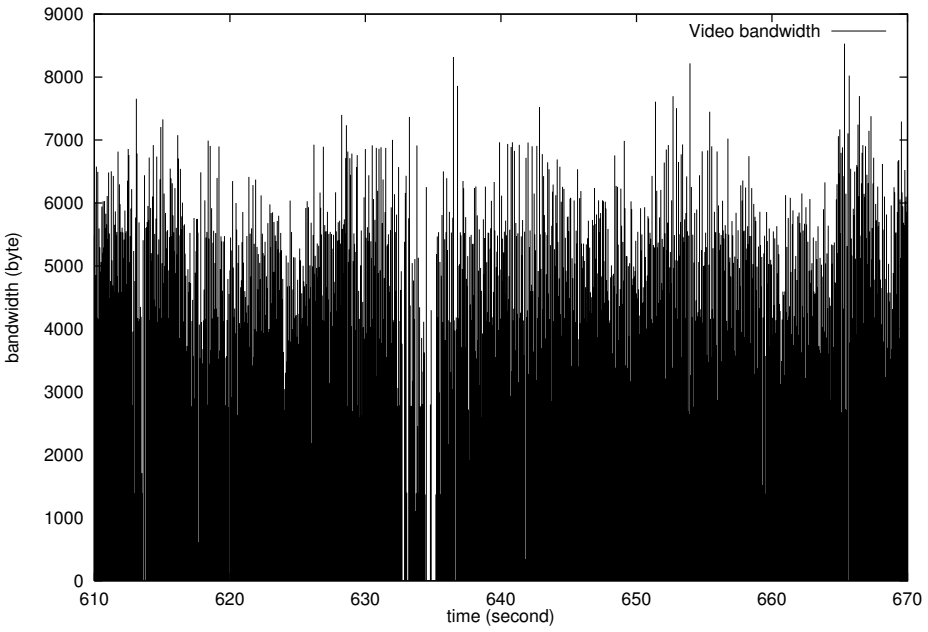

Figure 3.3: Zoomed in portion of the video bandwidth (50ms aggregation)

H.261. The figure shows a bursty behavior, high peaks and low averages. The minimum required bandwidth is about 10 kbytes/second which is 5.2% link utilization in our case. The maximum bandwidth is close to 140 kbyte/second (i.e. 67.7% link utilization). An average bandwidth requirement is 54 kbyte/s (i.e. 28.1% link utilization). These facts makes it very difficult to dimension the link for such traffic. Assuming low bandwidth usage will result in clipped peaks while the opposite, high bandwidth usage assumption, will give poor utilization during the idle period between two peaks.

3.3 Long-range dependency testing

For a network analyst or network designer it is useful to have detailed information about the statistical properties of the data stream passing through the network. One important issue is to find out what type of dependence the data exhibits. More exactly the stochastic process representing the data in the networks can be categorized as long-range or short-range dependent.

Dependence in this context is closely related to variance and correlation. According to [Ber94] a stationary stochastic process having slowly decaying correlations, such that they are not summable, is a process with long-memory or equivalently long-range dependence. Mathematically this can be expressed as

$$\sum_{k=-\infty}^{\infty} \rho(k) = \infty \tag{3.1}$$

where $\rho(k)$ is the autocorrelation function. In contrast a process with summable correlations, that is

$$\sum_{k=-\infty}^{\infty} \rho(k) < \infty \tag{3.2}$$

is called a short-memory or short-range dependent process.

A subset of long-range dependency is called self-similarity. Self-similar stochastic processes have similar statistical properties on all time scales. For instance it has been found that Ethernet traffic has the same statistical properties when the traffic was observed during milliseconds, seconds, hours or days.

A property of self-similarity, besides (3.1), is the slowly decaying variance property. [Ber94] shows that the sample mean for a self-similar stochastic process Y_t the sample mean can be written as

$$\overline{X} = n^{-1}(Y_n - Y_0) =_d n^{-1} n^H (Y_1 - Y_0) \tag{3.3}$$

where $=_d$ is equality in distribution and H is the self-similarity parameter. The variance $var(\overline{X})$ is then

$$var(\overline{X}) = \sigma^2 n^{2H-2} \tag{3.4}$$

The H-parameter can assume values $0.5 \leq H \leq 1$. In general the value of H is proportional to the degree of long-range dependency. Short-range dependent processes typically would have an H value of 0.5 while self-similar processes would have H close to 1.

This means that for short-range dependent processes

$$var(\overline{X}) = \frac{\sigma^2}{n} \tag{3.5}$$

which is a well-known result from statistics, and for self-similarity

$$var(\overline{X}) \to \sigma^2 \tag{3.6}$$

A more general form of 3.4 that is valid for the whole superset of long-range dependency is [Ber94]

$$\lim_{n\to\infty} \frac{var(\sum_{i=1}^{n} X_i)}{c_\gamma n^{2H}} = \frac{1}{H(2H-1)} \tag{3.7}$$

where we assume X_t be a stationary process with long-range dependence and c_γ is a constant.

There are several methods to test for long-range dependency, most notably Variance-Time plots, Index of Dispersion for Counts (IDC), Rescaled Range Analysis (R/S) and the Whittle estimator.

The Whittle estimator is an accurate tool used in estimating long-dependency. However, it assumes the data has a Gaussian distribution. It can be used on non-Gaussian data provided that the samples are of such nature that the central limit theorem can be considered valid. The central limit theorem states that given an infinite sequence of independent random variables from the same distribution, where the distribution has limited mean and variance, the sum minus mean times n divided by the square of the product of variance and n is *approximately* normal that is [LM86],

$$\lim_{n\to\infty} P\left(c < \frac{Y_1 + \ldots + Y_n - n\mu}{\sqrt{n}\sigma} < d\right) = \frac{1}{\sqrt{2\pi}} \int_c^d e^{-(1/2)y^2}\, dy \tag{3.8}$$

In (3.8), μ is the mean, σ the variance and c and d are any two numbers.

We want it to test if the MBone video stream is long-range dependent or, maybe, even self-similar. The video stream is only forty minutes long and counts a little more than 127000 packets. The data trace, containing the size of each packet, was tested to see whether or not it is normally (Gaussian) distributed. For this purpose we used a quantile-quantile plot. As the name implies the method plots the quantiles of the normal distribution against the quantiles of the video stream. If the plot shows a straight line than the video stream has a Gaussian distribution.

As we can see in Figure 3.4 the video stream is not normally distributed meaning we could not use the Whittle estimator to check for long-range dependency. Instead, we decided to use the variance-time plot method.

The variance-time plot method is not very accurate. In fact, all it does is to compute an approximate value of the H-parameter by doing linear regression on a log-log plot of the data. It's strength is the fact it does not make any assumptions on the the distribution of the data.

[Ber94] explains the variance-time plot as following. The variance from (3.7) can be approximated as cn^{2H-2},

$$var(\overline{X}_n) \approx cn^{2H-2} \tag{3.9}$$

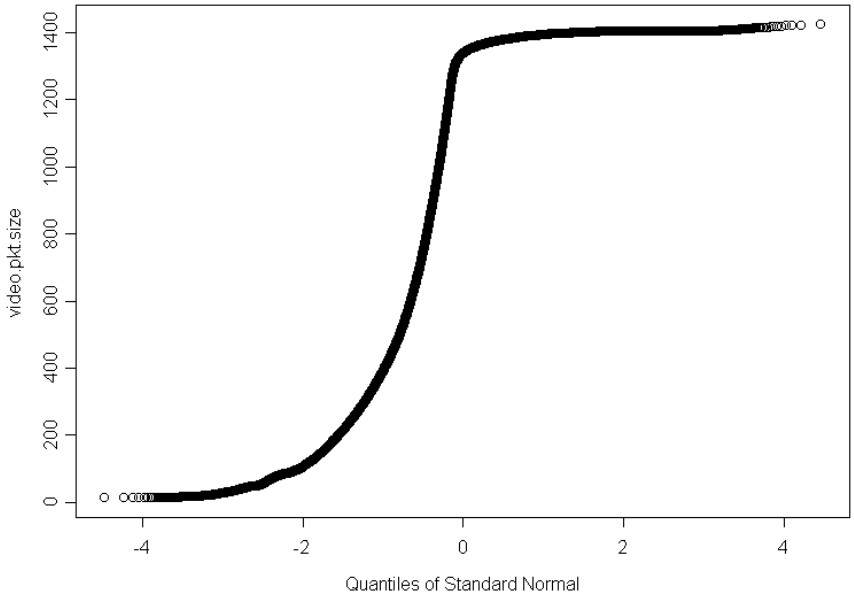

Figure 3.4: Quantile-Quantile plot for the video stream

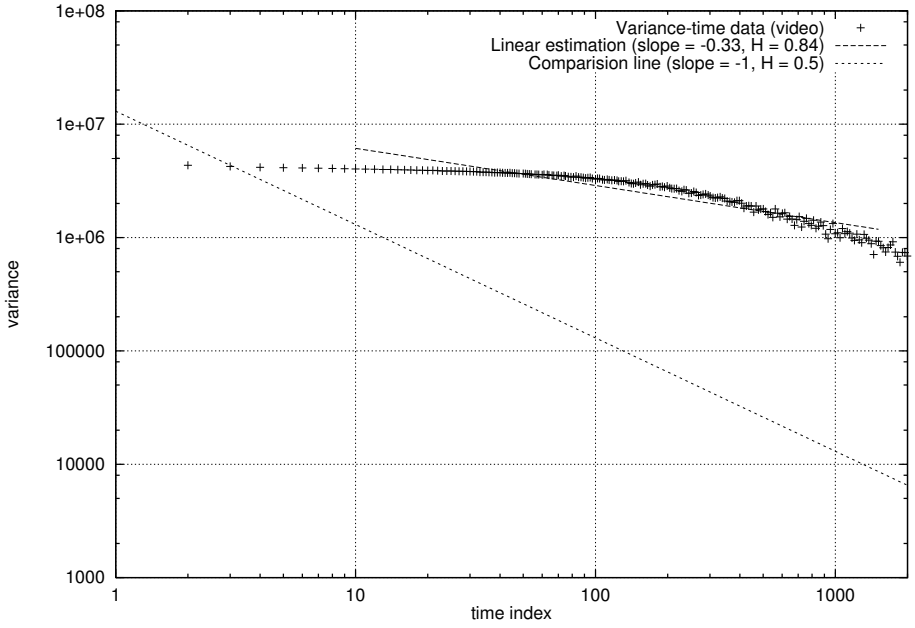

Figure 3.5: Variance-time plot for video stream

where $c > 0$. The series X_n are divided into m_k sub-series of length k, where $2 \leq k \leq n/2$. This is done for different integers k. Then the arithmetic variance $s^2(k)$ is calculated for each k. Finally $\log s^2(k)$ is plotted against $\log k$.

To understand what it happens we take the logarithm on both sides of (3.9).

$$\log var(\overline{H}_n) \approx \log(cn^{2H-2}) \Leftrightarrow \frac{var(\overline{H}_n)}{cn} \approx 2H - 2 \tag{3.10}$$

This tells us that the points on the plot can be approximated with a straight line having a slope of $2H - 2$. For short-range dependent data the line will have a slope of -1. Equation 3.10 gives also a rough approximation for the H-parameter. If we denote the slope of the line by β, then

$$H = 1 + \frac{\beta}{2} \tag{3.11}$$

We applied the variance-time plot to the video stream data and the result is shown in Figure 3.5. The line approximating the data has a slope of $\beta = -0.33$. The H-parameter is 0.84. This is an indication for having long-range dependent data. There is however no sign of self-similarity, because generally self-similar data has the H-parameter ≥ 0.9. The result we obtained can be trusted. The absence of self-similarity is explained by the absence of aggregation. We are in fact looking to only one video stream.

This concludes our long-range dependency testing. As far as audio is concerned it makes no sense to check for long-range dependency. Audio, in our case has a constant bit rate (CBR). A CBR source has zero variance and long-range dependency relays heavily on decaying variance.

3.4 Interarrival delay measurements

End-to-end delay is defined as the time it takes one packet to travel from the originating point to the destination point.

Inter-arrival delay is the delay between two consecutive packets arriving at one point in the network. This delay is made of two parts: the delay inserted by the source and the delay over the network path. Large inter-arrival delays can be accounted by lost packets or silence suppression mechanisms at the sender. Silence suppression mechanisms suppress packet transmission during silence periods in order to preserve bandwidth for other applications.

Delays are extremely important inside a multimedia network. As it has been explained in Section 3.1 they have a direct impact on the experienced QoS.

Knowing the importance of delays, we added functionality to *play_mbone* so we could measure them. The MBone session was recorded only at one end point, at the sender. As such we do not have data about the arriving times at the destination. This is the reason why we are not making end-to-end delay measurements and instead are looking to the inter-arrival delays only. For the rest of this section the word delay refers to inter-arrival delay unless otherwise specified.

Also, we listened to the session. The session time was divided into a number of segments of equal length, at first ten seconds and later sixty seconds. We marked each segment as good or bad according to the quality of the received audio. For this experiment bad audio quality meant that the received audio broke up more the three times per segment.

We did not look at the video quality. It is much harder to judge the QoS for the video part of the session. Some users would say blurry frames mean bad video quality to them while others consider the frame rate more important. Given this situation we preferred to concentrate on audio.

The inter-arrival delays were averaged over each time segment, plotted in a graph and finally the observed audio QoS is superimposed on the graph as big black dots.

Figure 3.6 and Figure 3.7 show an overall picture for the inter-arrival delays experienced in this session. Both figures show large variation in the delay. While this is normal for VBR video it is quite strange for CBR audio. One would have expected to see a flat plot for audio delays. A possible explanation is that silence-suppression mechanisms and lost packets are the reasons behind the delay variance. The lost packets can be a result of high bandwidth utilization or network congestion.

Figure 3.8 shows the inter-arrival delay for the audio stream averaged over one minute. Figure 3.9 zooms inside minute 9–12 of the audio stream where the delays are averaged over 10 seconds. Each sample in the zoomed graph covers the time segment starting ten seconds back in time end ending where the sample is. That is the sample at second 600 covers the time segment 591–600.

At a first glance one might expect the zoomed region to be all bad. But we can see that only 5 samples out of 24 are marked as bad. The plot reveals that a bad period of 10 seconds, the one situated at 520 seconds on the horizontal axis, accounted for a whole bad minute, minute 9 on the plot at the top of the page. Minute 10 (second 541–600) it is all good with the exception of the last sample. Minute 11 on the other hand (second 601–660) it is marked as "Bad QoS" on both plots. This shows an interesting thing: inside of what appears to

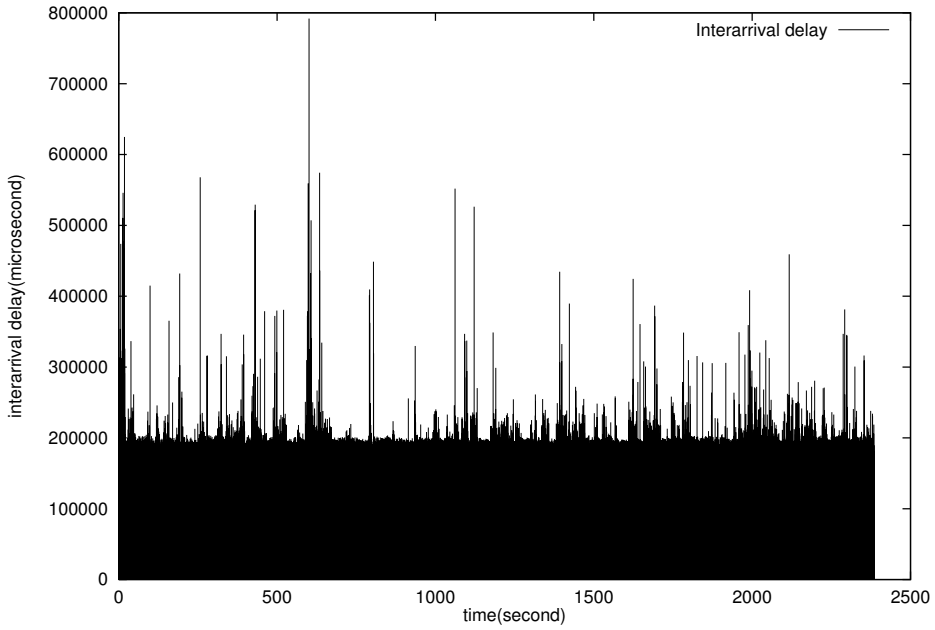

Figure 3.6: Audio packets inter-arrival delay

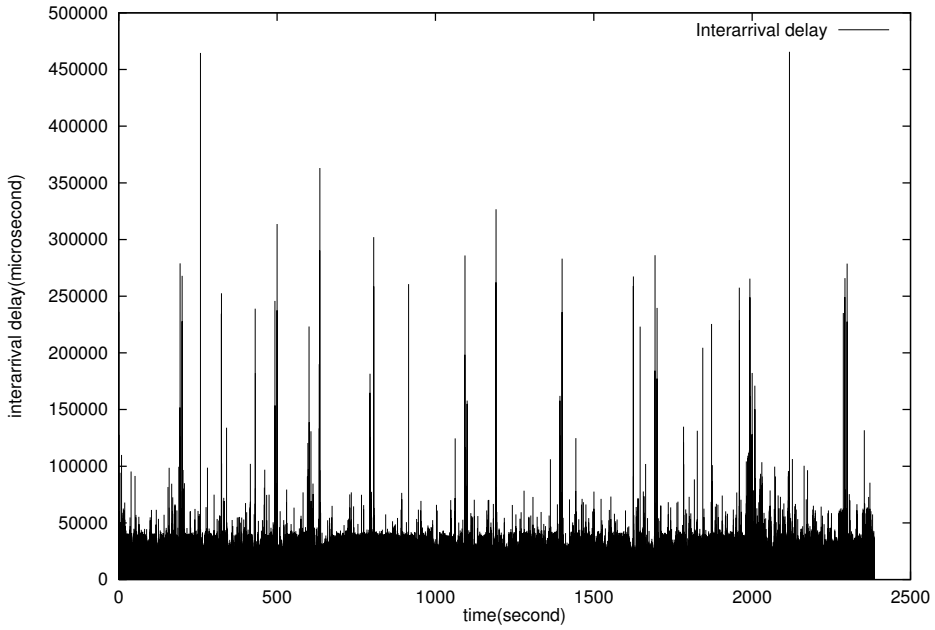

Figure 3.7: Video packets inter-arrival delay

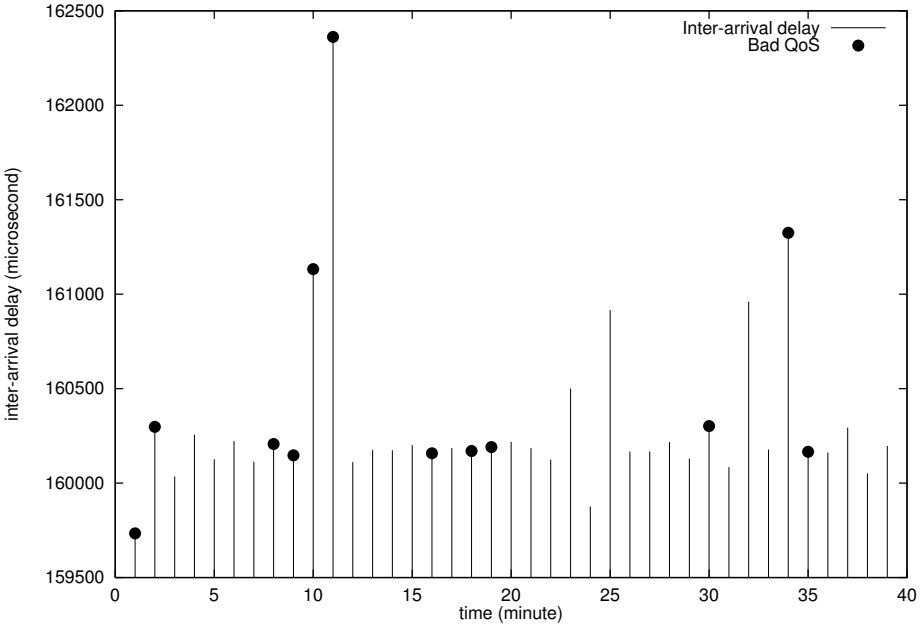

Figure 3.8: Audio packets average inter-arrival delay per minute

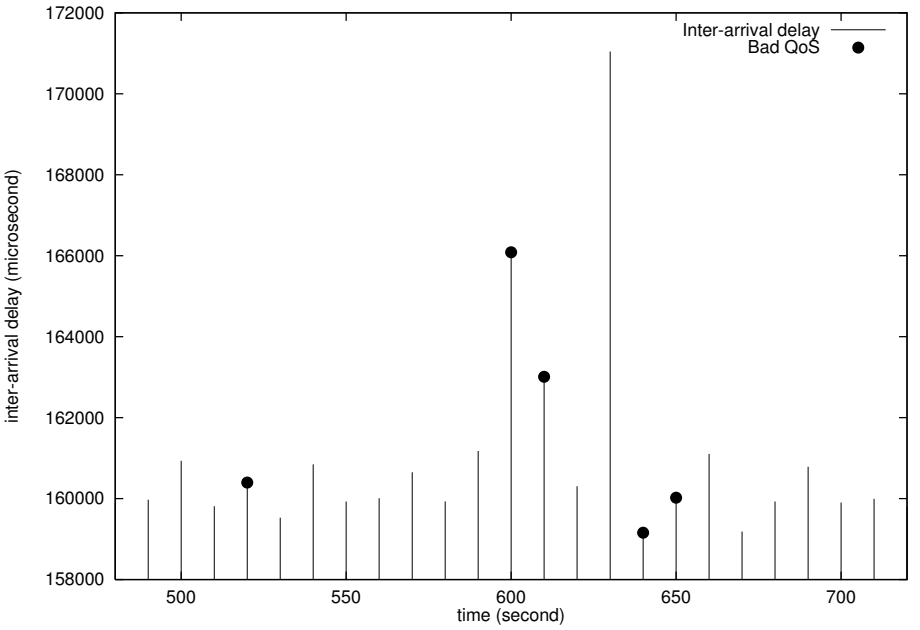

Figure 3.9: Audio packetes average inter-arrival delay minute 9–12

be a bad segment, one can see good and bad segments if he or she looks at a finer time scale. This is fractal like.

Both the audio and video codecs must be trained at the beginning. The software on top of the codec is receiving RTCP messages containing reports from various receivers about different aspects of the reception (see Section 2.1.3 for details over RTCP). The software examines this data and the result is passed to the codec. The codec is using this information to optimize its transmission. The first two samples marked as "Bad QoS" on the top plot (minute 1 and 2) are most likely results of codec training.

The average audio inter-arrival delay is a little over 160ms. The maximum inter-arrival delay is found at minute 11. There, the top plot shows a delay over 162ms and the plot below shows three spikes over 162ms with one of them going as far as 170ms.

For CBR audio the inter-arrival delays should be constant because by definition a CBR source sends data at a constant bit rate. According to [MM98] end-to-end delays of 200ms have shown to be commercially acceptable. The goal is to limit end-to-end delays to the range 100–200ms. Our measurements show that delays experienced by the session are in range but we have to keep in mind that the measurements were executed at the originating point. At far away destination, where many hops are involved, the inter-arrival delay and delays in general, will increase. This happens because, often, at each hop the packet is placed in a queue. The routing software at that particular hop has to look inside the packet header in order to know what the next hop the current packet must be sent to. These operations insert delays.

We want it to find out what whether or not inter-arrival delays were one of the causes for bad QoS in our session and tried to see if there were any excesive delays. Figure 3.10 shows the maximum inter-arrival delay for each minute. What we did was to look inside each segment of one minute length and extract the largest inter-arrival delay experienced in that segment. Figure 3.11 is the zoomed in portion of the top plot, using ten seconds long time segments. The largest spike, almost 800ms, is at minute 10.

Such long delays are either the result of silence suppression mechanisms or lost packets. If silence suppression mechanisms were working during minute 10 it is unlikely the listener would have noticed any audio quality degradation. We have to assume that lost packets account for this large delay. Section 3.5 will show this assumption is correct.

We will now perform a similar analysis on the video stream. An important thing to keep in mind is that the marks for "Bad QoS" belong to the audio stream. We are looking to see if there is any connection between bad audio quality and the behavior of the video stream.

As before, the top plot contains the whole sequence with averages over one minute while the plot at the bottom zooms in over minute 9-12 at a finer resolution (10ms). The inter-arrival delay varies quite a lot. This is common for a VBR video source. The medium delay is around 22ms, almost eight times smaller than the medium delay for audio. The bottom plot, Figure 3.13, has some averages going over 30ms.

The interesting thing to observe is that large inter-arrival delays seem to be good for the video stream. Most errors actually occur when the inter-arrival delay is below 18ms. One explanation can be that when packets come to fast some of them are dropped and, of course, if audio packets happen to arrive at

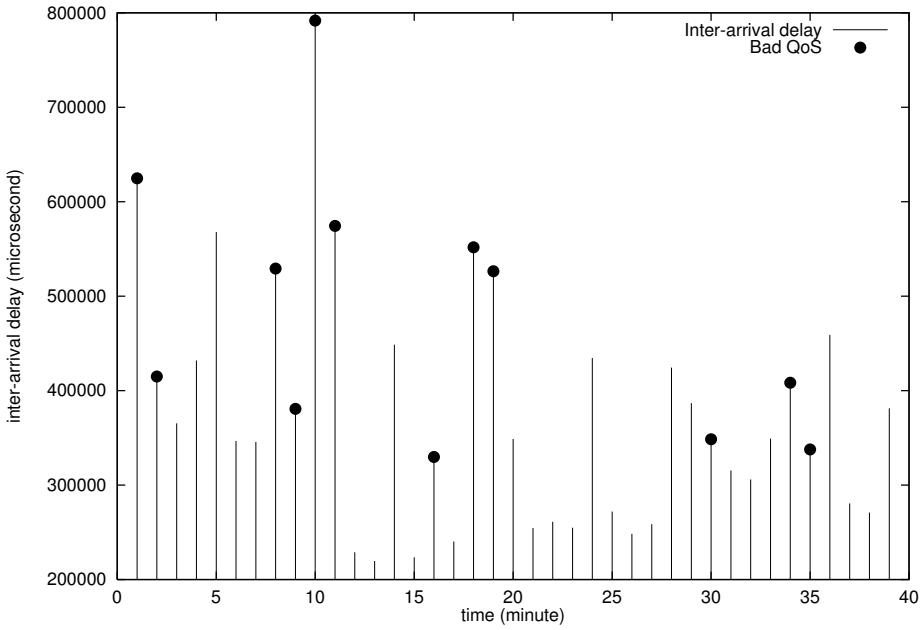

Figure 3.10: Audio packets maximum inter-arrival delay per minute

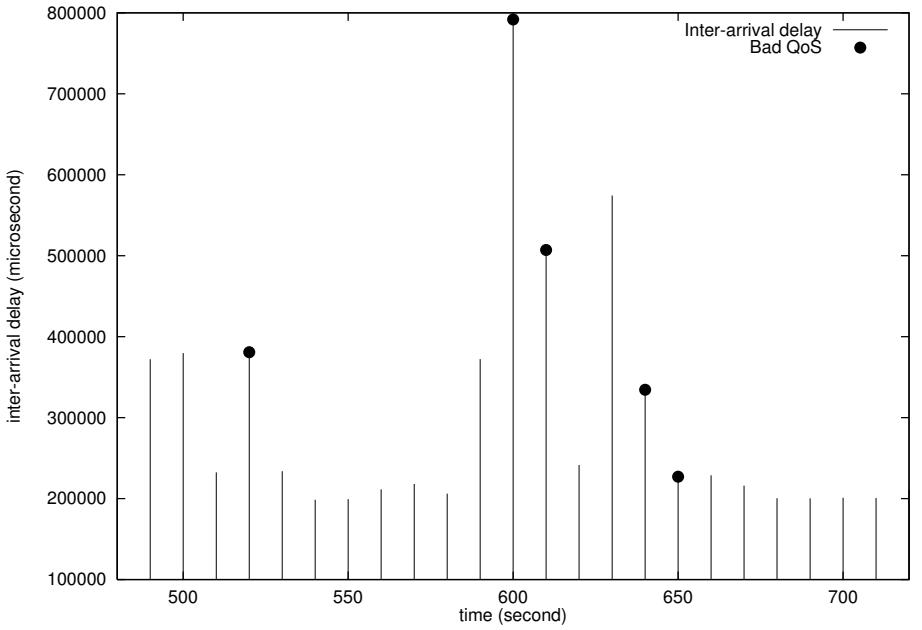

Figure 3.11: Audio packets maximum inter-arrival delay minute 9–12

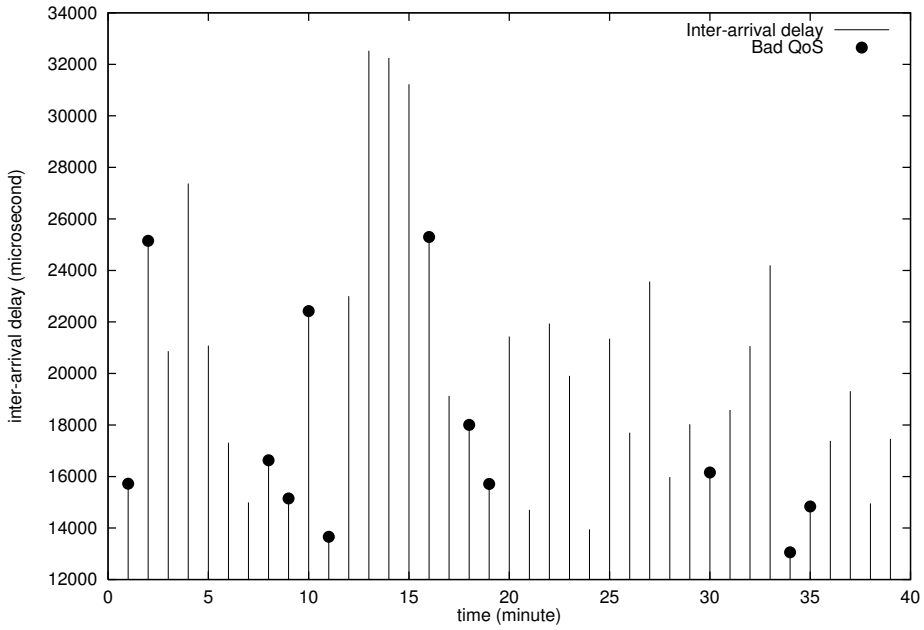

Figure 3.12: Video packets average inter-arrival delay per minute

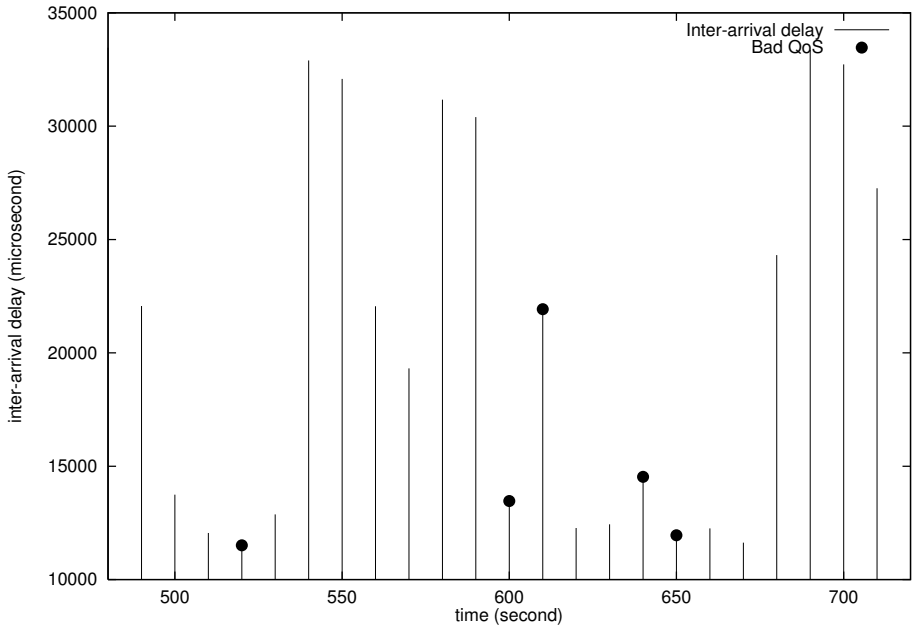

Figure 3.13: Video packets average inter-arrival delay minute 9–12

the same time the situation becomes worse.

A first look at the maximum inter-arrival delays, Figure 3.14 and Figure 3.15 seem to contradict our previous statement. More then fifty percent "Bad QoS" marks are found on delay samples larger then the average. We must consider this however, a worst case scenario and agree that the maximal delay found inside a time segment is not necessary representative for the whole segment. Also, given the number of "Bad QoS" samples additional testing on longer multimedia session is required before we can come to a solid conclusion.

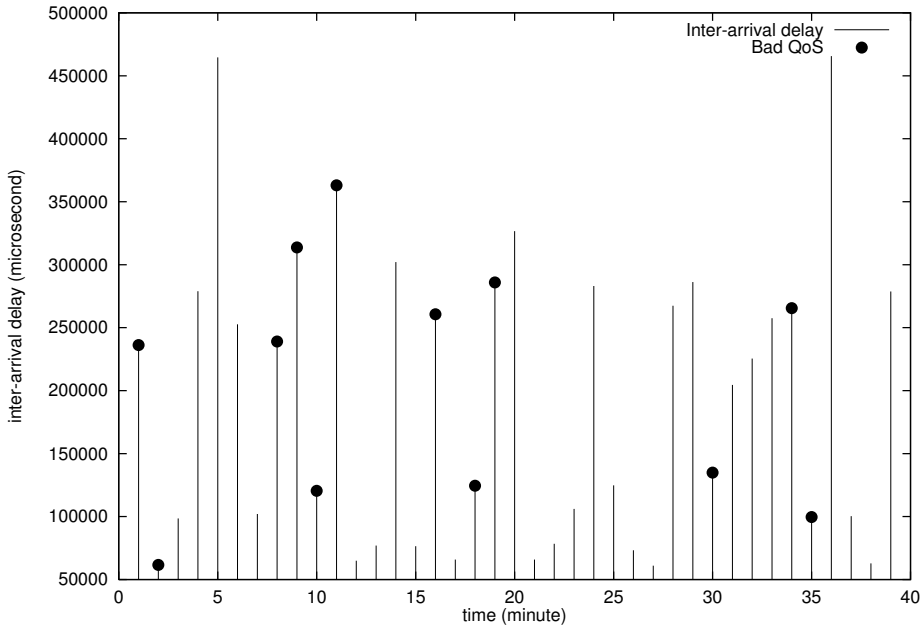

Figure 3.14: Video packets max. inter-arrival delay per minute

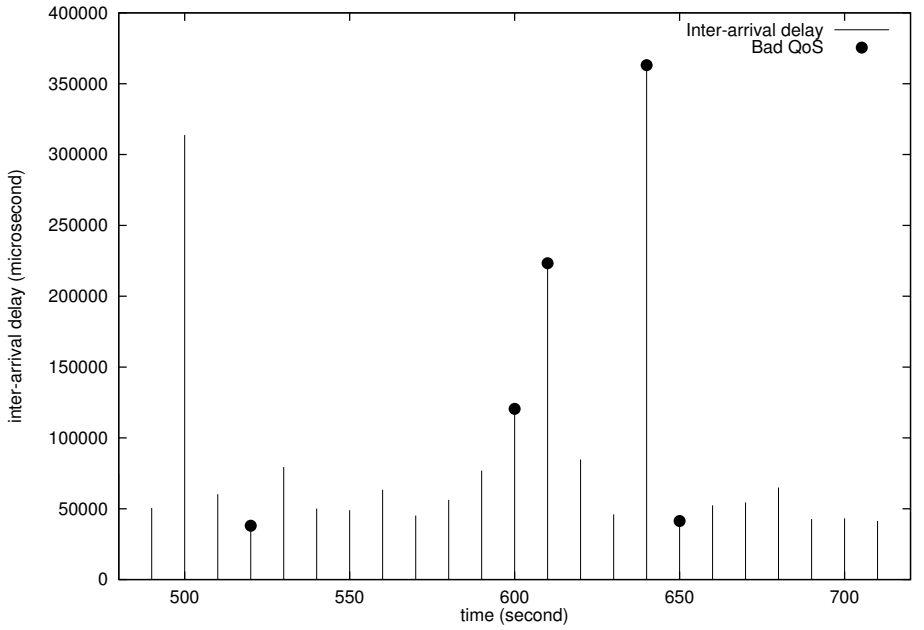

Figure 3.15: Video packets max. inter-arrival delay minute 9–12

3.5 Packet loss

Packet loss is one contributing factor to bad QoS. It usually occurs when the bandwidth utilization is very high and the network becomes congested. Also, in real-time applications, packets arriving to late are dropped by the application and become this way lost packets.

In this section we are going to examine the impact of packet loss on the audio QoS of the MBone session. We are using the same type of QoS measurement described in the previous section. In fact, we superimpose the same "Bad QoS" marks we used before on the plots in the current section.

Play_mbone measures packet loss by looking inside the RTP header and reading the sequence number field. The sequence number is increased by one for each packet sent by the source. This makes it easy to discover lost packets.

Figure 3.16 and Figure 3.17 show an overall picture over the packet loss. We can see that the average packet loss is around 4% for the audio stream and about 2% for the video stream. At the beginning we can see a high percentage of the packets were lost, over 45% for audio and more than 14% for video. Also, during minute 10 a similar amount of packets, 45% for audio and 12% for video, were lost.

We turn now our attention to the audio averages per minute in Figure 3.16. A very interesting observation is that all minutes marked "Bad QoS" show a packet loss greater than 0.5%. The audio codec sends on average 375 packets per minute. We can state that *for this MBone session any audio packet loss greater than 0.5% per minute (i.e. more than 2 packets per minute) will result in bad QoS.*

Figure 3.17 zooms into the time frame between minute 9 and 12. We see that our assumption in Section 3.4 concerning the packet loss being the main reason behind excessive inter-arrival delays was correct. Second 541-600 (minute 10) shows a packet loss of over 45% concentrated on a ten seconds segment (i.e. second 591-600).

We see a "Bad QoS" mark at second 650 where there is no packet loss at all. This can explained by the previous "Bad QoS" mark. On average, 63 audio packets are sent during 10 seconds. It is possible that the 4% loss rate at second 640 includes some packet sent from the end of the time segment. The effect of the lost packet was then observed in the next time segment.

Figure 3.20 and Figure 3.21 focus on the video packets loss. Video loses on average per minute about 0.2% of its packets. Because video transmission is using a variable bit rate it is hard to express the percentage as a general number of packets lost.

The first minute shows the greatest packet loss for the whole session. One contributing factor is again the codec. It simply needs a certain amount of time before it adjusts to the network impulse response. At the same time the audio codec is also doing the same thing and it is possible both of them are using a large amount of bandwidth, probably more than it is available. This results in packet loss. The bandwidth utilization problem can also explain the packet loss inside time segment 9–12.

The video top plot does not offer more information, at least about the audio QoS. Also, the zoomed in plot contains all to little information for any other time segment than 590–610.

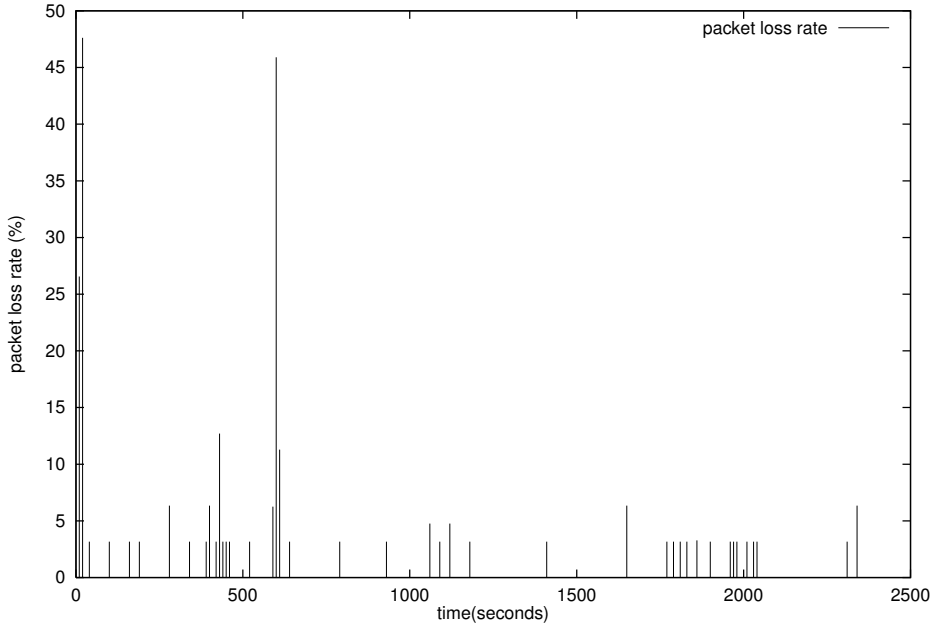

Figure 3.16: Lost audio packets

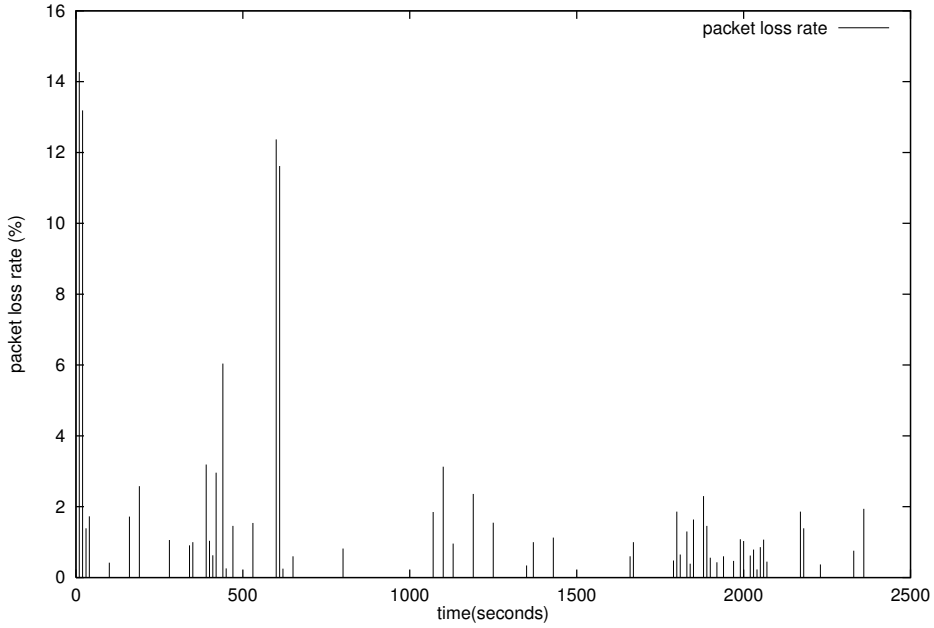

Figure 3.17: Lost video packets

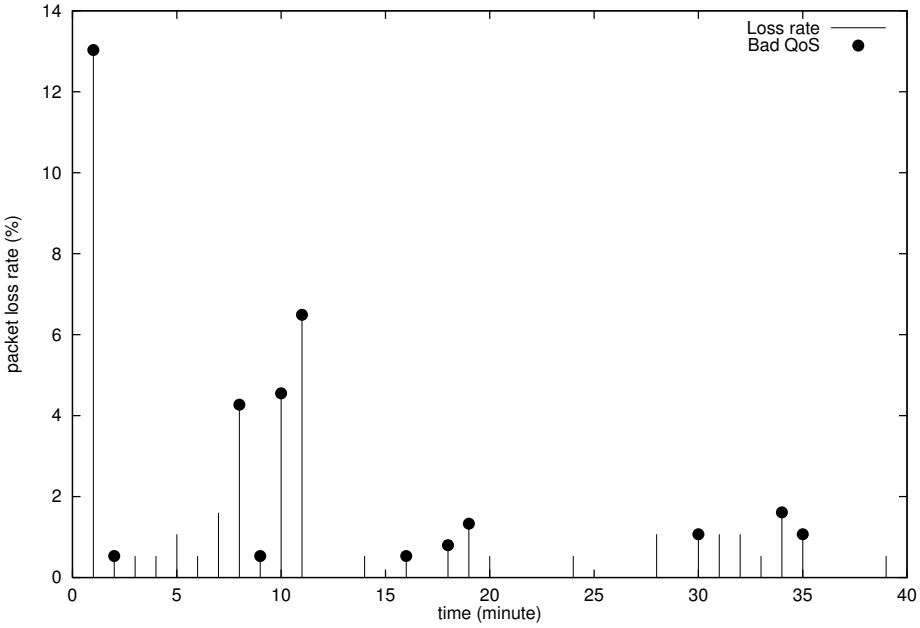

Figure 3.18: Audio packets loss rate

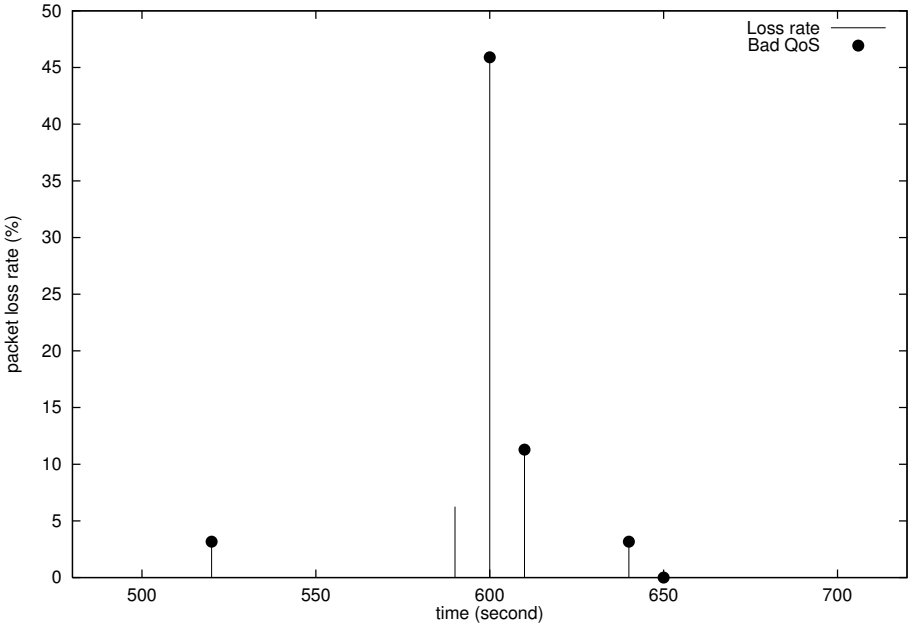

Figure 3.19: Audio packets loss rate minute 9–12

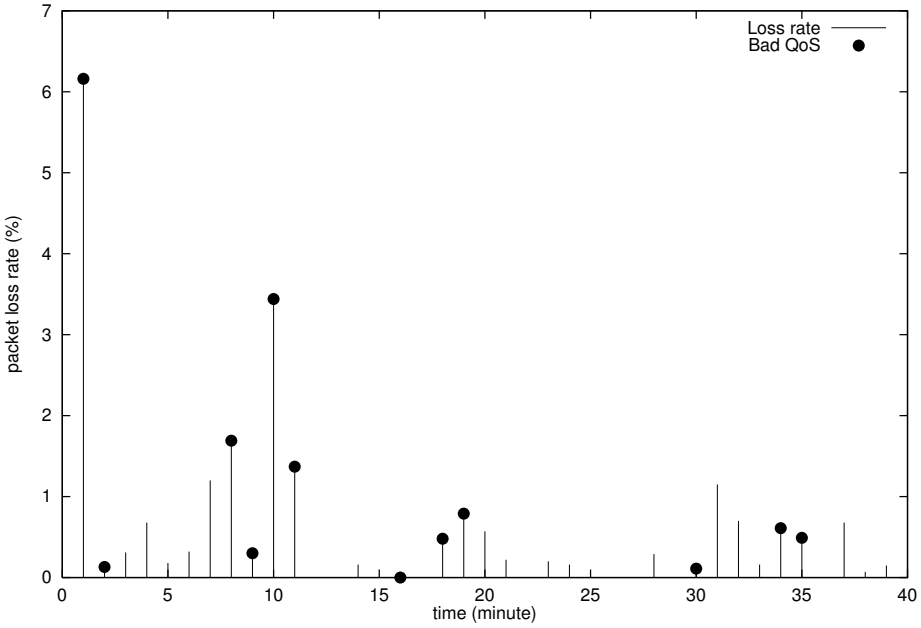

Figure 3.20: Video packets loss rate

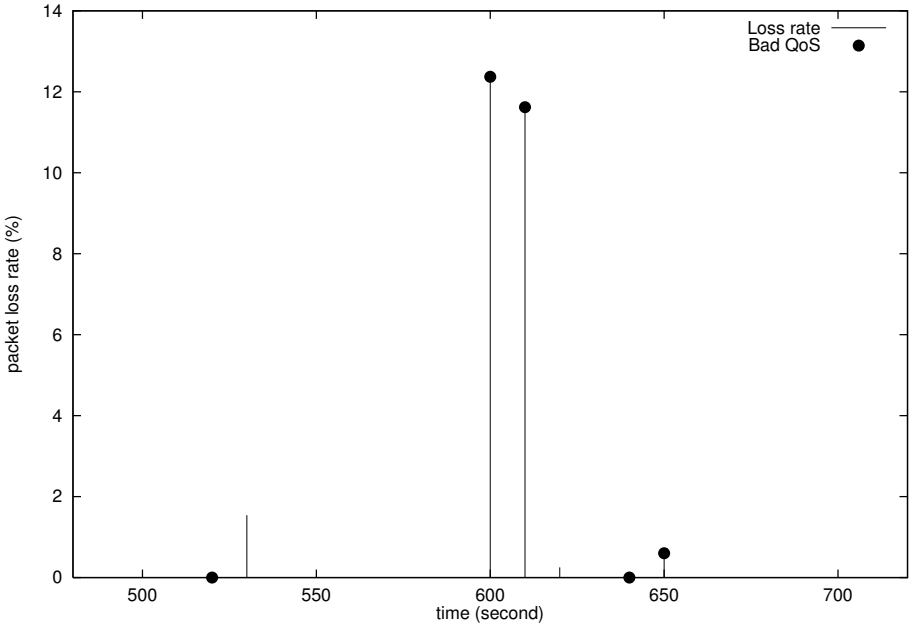

Figure 3.21: Video packets loss rate minute 9–12

Chapter 4

Summary

MBone multimedia traffic gives a rough idea about the kind of network traffic we should expect in the near future. Multimedia data has a different behavior than plain data. Todays networks are not fully fit to handle large quantities of this kind of traffic and provide good QoS at the same time.

QoS itself is a term difficult to define for certain type of data, video for instance. There are many factors that can affect the QoS and often it is the agreggated effect of these factors that has a noticeable impact on the quality the user experiences.

4.1 Goals achieved

The primary goal was to create software for MBone data extraction and playback. The software we created, *play_mbone*, *sdrdump* and *Spy applet* act together to achieve this goal. By using them one can re-experience the same QoS as it was when the MBone session was recorded.

The software is also computing statistics in real-time. The results are stored in plain text log files that can easily be loaded into mathematical software for additional processing.

We examined the statistics for a typical MBone session and discovered that QoS is a very complex issue. Video packets (with H.261 encoding) are non-normally distributed and exhibit long-range dependency which makes it harder to predict future traffic.

Inter-arrival delay is close related to packet loss. Short delays increase the bandwidth utilization and generate packet loss. Long delays on the other hand generate noticeable stops in the media flow and that translates into bad QoS. There must be a balance for what inter-arrival delay is optimal for a specific media but more research is required before a general solution can be found.

One QoS metric, that holds at least for the audio part of this session, is the packet loss. Whenever the number of lost packets per minute is greater than (i.e. $\approx$2% loss rate) the user is likely to notice bad QoS.

There were a lot of lessons I learned. First of all, one must first think carefully if the assumptions made are the correct ones. It makes no sense for instance to apply non-normally distributed data to a method that assumes the input data is normally distributed. Nor it makes any sense to apply methods relying

on decaying variance to data with zero variance. Also, when programming one should make maximal use of the existing software and not "reinvent the wheel". Generally, one should think ahead but be flexible and ready to adjust.

4.2 Future work and acknowledgements

The work done here is by far not over. Existing software can be improved and additional functionality added. For example, private MBone sessions make use of Session Invitation Protocol (SIP). No MBone implementation can be considered fully complete if it does not support SIP.

More theoretical questions must be answered. Do all MBone sessions traffic behave as the one we analyzed? To answer this more data must collected and analyzed. This way it is also possible to find a reliable mechanism to ensure good QoS.

We must not forget that H.323 lies "around the corner". This is the ITU-T's powerful generic standard for the next generation of telephony and multimedia transport. What will it happen when thousands of people will start to use it? What has to be done to make sure tommorow's networks and internetworks can cope with the expected load?

The conclusion is that multimedia communications is a very new and broad field. There is a considerable amount of work left to before multimedia networks are ready for their task.

Working with my Master's Thesis has been a great experience. I had the opportunity to do very interesting work and learn a lot from my colleagues in United States. I would like thank the former University of Karlskrona/Ronneby in SWEDEN (I believe it is called now Blekinge Institute of Technology) and my telecommunications professor in Sweden, Dr. Adrian Popescu, for giving me the chance to work on this thesis.

I would like to express my gratitude for the people in US who have been very helpful and made me feel like home. Particularly, I would like to thank Dr. Parag Pruthi who has been my advisor in US and who encouraged and inspired me all time during my presence there. It is his advice that often helped me make the right decisions. Dr. Varugis Kurien deserves all the credit for introducing me to the heavy artillery of statistics. Discussions with him always made things look more clear.

Mr. Andrew Heybey was always ready to answer my questions in the context of UNIX programming and Niksun Network Recorder. We had some interesting discussions about operating systems performance.

Also, I would like to thank Dr. Ajay Singh for the talks we had over the topic corporate networks and the effect of full-scale multimedia communications.

My roommate, Mr. Ajit K. Jena, diserves acknowledgment. We had a lot of interesting discussions about ATM and network simulations.

Last but not least, I would like to thank my parents and friends in Sweden for their support in this endeavor. You have been far away but yet so close!

`East Brunswick NJ, 27 January 1999`

Bibliography

[Ber94] Jan Beran. *Statistics for Long-Memory Processes*. Monographs on Statistics and Applied Probability. Chapman & Hall, One Penn Plaza, New York, NY 10119, USA, 1994.

[CWH+96] Jon Crowcroft, Ian Wakeman, Mark Handley, Stuart Clayman, and Paul White. *Internetworking Multimedia*. UCL Press, 1996. http://ee.mokwon.ac.kr/~music/tutorials/mmbook/book.html.

[HJ98] M. Handley and V. Jacobson. SDP: Session Description Protocol. Standards track, Internet Engineering Task Force, April 1998.

[KMKW98] P. Kirstein, G. Montaser-Kohsari, and E. Whelan. SAP Security Using Public Key Algorithms. Internet draft, Internet Engineering Task Force, September 1998. draft-ietf-mmusic-sap-sec-04.txt.

[Kum97] Vinay Kumar. What is the mbone (or ip multicast). WWW, 1997. http://www.mbone.com.

[LM86] Richard J. Larsen and Morris L. Marx. *An Introduction to Mathematical Statistics and Applications*. Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 1986.

[MM98] Daniel Minoli and Emma Minoli. *Delivering Voice over IP Networks*. John Wiley & Sons, USA, 1998.

[Pru95] Parag Pruthi. *An Application of Chaotic Maps to Packet Traffic Modeling*. Royal Institute of Technology, Department of Teleinformatics, Royal Institute of Technology, Stockholm, SWEDEN, October 1995.

[SCFJ96] H. Schulzrine, S. Casner, R. Frederik, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. Standards Track RFC 1889, Network Working Group, January 1996.

[Ste94] W. Richard Stevens. *The Protocols*, volume 1 of *TCP/IP Illustrated*. Addison-Wesley, USA, 1994.

Index