PREDICTIVE QUALITY OF SERVICE FOR VIDEOCONFERENCING

by

Martin Bateman

A thesis submitted for the degree of

Doctor of Philosophy

University of St Andrews

February 2005

ABSTRACT

Real-time videoconferencing across the Internet is a potentially very useful facility. Unfortunately it demands relatively high levels of Quality of Service (QoS) from the network infrastructure: low levels of delay, jitter and packet loss coupled with a high bit-rate. The current Internet cannot provide these QoS parameters as infrastructure options.

This thesis presents a framework and development test bed which aims to address these QoS problems for videoconferencing on the Internet. The framework, named the Conference Controller Architecture (CCA), performs passive and active network measurements of videoconferences. These traffic measurements are used to make predictions of network behaviour for future videoconferences. The predictions are used as the basis for the session configuration. The choice of CODECs, encoding parameters, packet loss repair mechanisms and smoothing buffers are all based on the expected state of the network paths to be involved.

A test bed was created which is able to create virtual network topologies which can be used to aid development and testing of multi-user networked application. Not only are network technologies emulated via the use of traffic shaping but users are also emulated. This gives the ability for user behaviour to be taken into account during the development and testing process.

Finally, the policies supported by the CCA take participants' perceptions and "fair" use of Internet bandwidth into account. In particular, it is important to achieve consistency rather

than optimality during a session, and it is important to be a good network citizen and not hog resources.

I, Martin Bateman, hereby certify that this thesis, which is approximately 60000 words in length, has been written by me, that it is the record of work carried out by me, and it has not been submitted in any previous application for a higher degree.

date 14th February 2005 *signature of candidate*

I was admitted as a research student in September 2000 and as a candidate for the degree of Doctor of Philosophy in September 2001; the higher study for which this is a record was carried out in the University of St Andrews between 2000 and 2005.

date 14th February 2005 *signature of candidate*

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of Doctor of Philosophy in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree

date 14th February 2005 *signature of supervisor*

In submitting this thesis to the University of St Andrews I understand that I am giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. I also understand that the title and abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research worker.

date 14th February 2005 *signature of candidate*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank the people who have made this thesis possible. I would like to thank my supervisor Dr. Colin Allison for the help and guidance throughout the entire project, without him this thesis would not have been possible. My thanks go to the members of the Distributed Systems and Networking group at St Andrews for providing a sounding board for my thought and ideas, to the School of Computer Science for their understanding when it became time to submit.

My thanks go to my wife, San, who has had to put up with me throughout the writing of this thesis.

Chapter 1

Introduction

1. Introduction

Audio and Video[1] conferencing allows for multipoint communication in a way which is not possible with traditional means. Multiparty videoconferences can be setup, joined and torn down in a much shorter amount of time than is required to setup a conference using more traditional means such as face-to-face meetings.

Videoconferencing systems have been around for a long time. The earliest verifiable reference is to a system called Picturephone [1, 2] created by AT&T in 1956. The Picturephone was demonstrated at the World's Fair in New York in 1964 and later offered in 1970 as a commercial product for $160 a month. In 1992 the first worldwide audio [3] broadcasts were performed using the Mbone [4]. In 1994 CuSeeMe [5] was released for Macintosh which provided videoconferencing over IP for the desktop. By that time the Mbone was also being used for videoconferencing.

Videoconferencing is becoming more common. As increases in computing power and network resources allow the average user to take advantage of videoconferencing from their desktop, it is expected to become a major part of the desktop environment in coming years. Videoconferencing provides a simple and cost effective way of providing a communication facility by leveraging an existing infrastructure, an IP network, to provide extra services. The availability of ever increasing bandwidth on the network as network paths are upgraded appears to be part of the solution to the videoconferencing problem but this is accompanied by an increase in applications competing for a finite resource. The phenomenon of competing applications vying for resource means that the Quality of Service (QoS) which can be provided by the network changes from moment to moment.

Videoconferencing is expected to become a major part of communicating in the future [6, 7]. As such the amount of traffic which is used by videoconferencing will increase as a percentage of the total traffic on the Internet. The current situation is that TCP based traffic makes up around 90%[2] of Internet traffic but as videoconferencing becomes more common this will change and UDP traffic will become more prevalent. As a consequence of this it is possible that a situation similar to the congestion collapse of the late 1980's will arise with unregulated UDP applications attempting to send large bit-rates in order to gain higher quality for both video and audio applications.

1.1. Why use videoconferencing?

The benefits of face-to-face contact are well known and documented in the physiology domain [8, 9] when involved in human communication. Mehrabain is quoted by Pease [10] as saying that 'the total impact of a message is about 7 per cent verbal (words only) and 38 per cent vocal (including tone of voice, inflection and other sounds) and 55 per cent is non-verbal'. It is clear from this that the majority of communications come from the non-verbal domain. There are two forms of non-verbal cues; paralinguistic and extralinguistic. Paralinguistic cues are voice signals which accompany speech. These could include intonation, for example 'question intonation' where the pitch is raised at the end of a statement in order to make it a question. Tempo and word stress also play a part. Extralinguistic cues are ones which are not related to the voice but are more to do with the body. These include gaze direction and eye direction, as well as gestures and body language. Paralinguistic cues can be provided by audio only conferencing. Extralinguistic cues require some form of visual link in order to communicate them. Currently videoconferencing is used

[1] Hereto referred to as videoconferencing

[2] See Chapter 2 for a breakdown of Internet traffic

to provide the benefits of paralinguistic and extralinguistic communication to people who do not have the luxury of being spatially co-located.

Meetings are a normal part of business with some sources [11] stating that up to 33% of employee time is spent in meetings in the UK. The number rises to 37% in the US [12]. The average cost of a six person face-to-face meeting, with five people having to travel 50 miles or more, in the UK is £1,645. The cost is closer to £3,000 when plane travel is involved. Cost savings of up to 92% have been achieved by using audio or videoconferencing in the US. However, not only are there cost benefits associated with not having to travel there are also benefits of time saved. As an example MCI showed that for a five person meeting the total time involved is 53 hours 24 minutes, 16 hours 51 minutes and 16 hours 29 minutes for a meeting attended in person, attended via audioconference and attended via videoconference. This leads to a reduction in stress since there is no need to organise travelling, spend time away from family, or to claim back expenses from the trip. The reduction in time spent travelling can be used to make more efficient use of the working day.

In general, there are three types of meeting

1. Problem Solving meetings are formed in order to resolve a problem.
2. Brainstorming meetings are formed in order provide ideas about a specific topic.
3. Informational meetings are formed in order to disseminate or receive information on a specific topic or idea.

All three types of meetings require some preparation. It can be argued that there is a fourth type of meeting, impromptu. These are generally not thought of as meetings but are more

akin to two or more people running into each other in a corridor, or perhaps one person going to another's office for an informal chat.

There are several different types of videoconference

1. *Videophony* – Biparty videoconferencing usually has low to medium frame rates and small to medium frame size which show a talking head. This form of videoconferencing is the most often used and commercial offerings include CuSeeMe and Microsoft MSN.

2. *Video seminar distribution* – This generally consists of a speaker and her notes, with many receivers. There is a video stream which shows the speaker and possibly some lecture notes as well as the audio stream.

3. *Media Space* – A media space is a computer controlled network of audio and video equipment for collaboration between groups of distributed people. A media space is an always on service which is available all the time. They provide shared public spaces to spatially dispersed groups of people.

4. *Video-wall* – A large screen is placed on the walls of common areas at different sites, each of which presents audio and video from the other site. The idea is that as people use the common area they are able to see and talk to the people at the other site.

1.2. What is meant by application quality?

Quality of service is quite often thought of in terms of some absolute qualities. In the case of networks which can support QoS, specific network parameter values are either requested or negotiated at connection setup time. For the Internet, this can mean providing the best available values at any given time. This could translate to using all available bandwidth on a

network path for a video conference and lead to large variations in the quality received over the course of a session as the available bandwidth fluctuates with the demand placed on a network path. The notion of QoS used in this thesis takes the participants' perspective into account and does not adopt either of the above approaches.

1.3. The problems of videoconferencing

Videoconferencing is sensitive to delay, loss and bandwidth restrictions. With this in mind several systems which provide videoconferencing services have been created using transport mechanisms such as ISDN and ATM, both of which create virtual circuits, thus allowing quality of service parameter values to be allocated at circuit setup time.

This resource reservation approach, in which the resources that are needed for the videoconference are reserved throughout the network, is not possible on the general purpose Internet since it is impossible to reserve resources at each point along the traffic route.

Another method is to use real-time feedback from the transmission stream in order to alter the transmitters' bandwidths to suit the available bandwidths on the network paths. Generally, these systems repeatedly increase the bandwidth used until packet loss is detected and then throttle back. This leads to a '*buzzsaw*' effect of bandwidth usage which can cause varying absolute quality of service and is distracting for the users.

1.4. Thesis question

Videoconferences are extremely sensitive to delay, jitter and packet loss in that human perception systems have not evolved to cope well with these conditions. Unfortunately these are exactly the conditions which are created by the use of a best effort packet delivery network such as the Internet.

This thesis takes the stance that quality of service is unlikely to be provided to Internet applications by resource reservation or admission control in the short or medium term. Instead this thesis investigates the use of *predictive* quality of service, based on the use of network monitoring, and statistical analysis, of network paths. So, is it possible to predict QoS, and is it feasible for an application to exploit this knowledge to adapt its own network usage to that which the network is able to provide?

1.5. The original contributions of this dissertation

The major original contribution of this thesis is the exploitation of predictive quality of service for participant-oriented QoS requirements. Using past measurements of the network and the knowledge that network traffic patterns have a temporal aspect to them allows a system to make a prediction on how the network will behave. A proof of concept system has been designed and built which is able to make this form of prediction and then configure a multiparty videoconference for consistency of quality.

As part of the testing of the system it was necessary to host a large number of video conferences. People who were willing and able to participate in repeated videoconferences for the purposes of testing and development were in short supply so a test bed had to be created. The test bed, a scenario driven network emulator, emulates both the network conditions and the users of a real-time groupware application. At the highest level it emulates user behaviour, in the case of videoconferencing this is an emulation of speech patterns. At the lowest level there is a traffic shaper which emulates network paths, it was necessary to build a new traffic shaper as currently available traffic shapers did not provide all the required functionality. On top of the traffic shaper there is a configuration manager which when given a network topology configures a series of nodes to reflect that topology.

Finally this thesis presents a series of configuration suggestions for group videoconferences which take into account the network conditions (bandwidth, packet loss, jitter and delay) and provide configuration information for CODECs to use bit-rates, colour depths, packet repair techniques, sample rates and (in the case of video) spatial resolution.

1.6. Thesis organisation

In Chapter 2 the background to the thesis is presented. This starts with the physiological requirements, which provides grounding on how people perceive audio and video. Next, the requirements that audio and video media make on a network are discussed before describing, the Internet network conditions in which videoconferencing applications must operate. This includes the traffic patterns for global and local level networks.

Chapter 3 is concerned with related work, introducing a selection of multimedia toolkits which can be used to construct audio and videoconferencing applications. The current state of the art in quality of service for multi-user videoconferencing is presented. Complete frameworks as well as multimedia transport protocols are discussed.

Chapter 4 fully describes the design and implementation of a Conference Controller Architecture (CCA), while Chapter 5 describes the Scenario Driven Network Emulation test bed, a system built in order to test and develop the CCA. Chapter 6 reports on the testing and evaluation of the CCA, and Chapter 7 presents the conclusions and future work.

Chapter 2

Technical Context

2. Technical Context

This chapter introduces the required background knowledge which is needed in order to understand this thesis. This chapter is split into three parts. The first section - "Taking the Participant into Account" - describes the physiological requirements placed upon audio and video by the human perception system. This allows us to make decisions that are more informed when we are talking about the quality of the audio and video. This also means that we are better informed when considering how to configure video conferences in order to create the best quality of service for the users. The second section describes the various demands made upon a network by the transport of audio and video across it. With this information, we can make better informed decisions on the audio and video transport. The third section identifies and describes network traffic characteristics. This in turn is used to demonstrate that there are patterns within network traffic which can be exploited when configuring conference sessions.

2.1. Taking the participant into account

There are various taxonomies which provide valuable information on acceptable audio and video qualities.

There have been various studies on acceptable frame-rate for use within a video conference. Isaacs et al [13] report that 4 frames per second is distracting while Watson and Sasse [14] support this by saying that perception of the audio-video synchronisation was impaired below 5 frames per second. Tang and Isaacs [15] report that users rated video at 5 frames per second as '*acceptable*'.

The task the user is engaged in plays a part in the quality of the videoconference that is deemed tolerable. Both Anderson et al [16] and Matarazzo et al [17] showed that when using both videoconferencing and teledata the impact of low quality of service for the videoconferencing is reduced.

Patrick [18] identifies mode, media and task as their top level parameters, with various sublevel aspects which should be taken into account, see Figure 1.

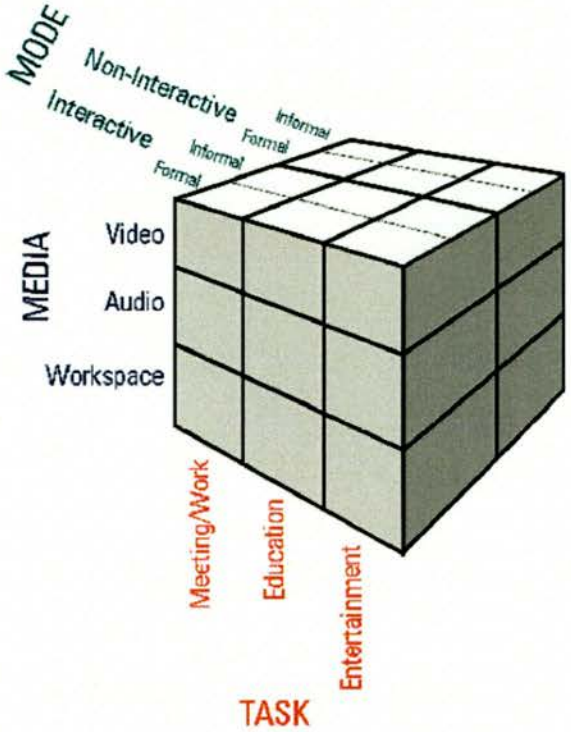


Figure 1 Schematic representation of videoconference taxonomies, reproduced from [18].

Mode indicates how the user will be interacting within the conference. Mode subdivides into interactive and non-interactive; each of those situations has both formal and informal settings. Media divides into audio, video and workspace. Finally task divides into meeting/work, education and entertainment.

Mullin et al [19] identifies the interactions of task, user and situation as the top level in their taxonomy. Task can be subdivided into telepresense and teledata and into foreground and background. User requirement subdivide into hearing, sign and no impairment. Finally the situations include items such as background noise, background movement and the size of the group.

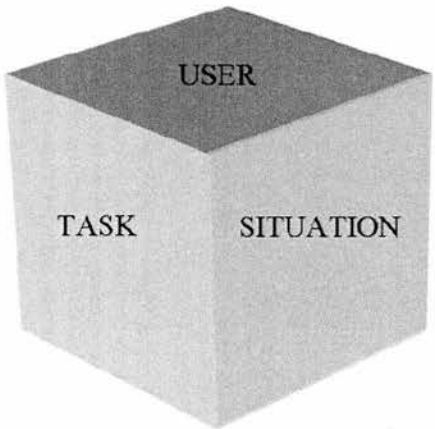


Figure 2 The taxonomic Application Cube, reproduced from [19]

2.1.1. Scenarios

Ochsman and Chapainis [20] found that adding a video channel had no significant effect on communication time or communication behaviour while Gale [21] found that there was no difference when a video channel was added in the quality or time taken to complete a task. These studies were focussed on the outcome of the collaboration whereas Tang and Isaacs [15] was focused on the process of collaboration. It was found that videoconferencing aids in the process of interaction. Rudman et al [22] and Isaacs et al [23] support this, their findings are summed up by Carles as *'people like to see each other when they interact, especially when they do not know each other well'* [24].

Buxton [25] divides tasks into foreground and background tasks, an implication of this is that the quality requirements for a background task, where users pay less attention may be lower than for a foreground task [26].

There are three scenarios which will be examined. These feature on the scale of interactivity and number of participants. The three scenarios are meeting, tutorial and lecture.

2.1.1.1. Meeting

Rettinger [27] describes a collaboration as:

> '*Collaboration mode is typically a many-to-many interaction. Each participant is a peer who participates in the conference equally, although in some cases there may be a facilitator who manages the conference and keeps the agenda flowing smoothly. Collaboration mode utilizes symmetric communication among the participants.*' [27].

Within a meeting there are usually a small number of participants. This may not always be the case and meetings with large numbers of participants do happen. The level of interactivity between the participants of a meeting depends on the familiarity of the participants [28] as well as the number of participants [19] within the meeting.

In the case of a low number of participants the level of interactivity between the participants will be high and is more likely to use a social floor control mechanism for normal conversations. When the group is large or has unfamiliar participants it is likely to use a more formal floor control mechanism.

For the majority of participants a meeting in this sense fits into the ETNA taxonomy [19] as foreground telepresence. The size and makeup of the group of participants determines if the meeting is interactive or non-interactive. For the purposes of this thesis a meeting is deemed

to have a low number of participants and to adopt an interactive style. So its categorisation in terms of the ETNA taxonomy would be as an interactive foreground telepresence situation.

In Foreground telepresence the '*speech channel is the primary means of communication making audio particularly important*' [19].

Low quality can impair speech perception with an '*increase in mishearing for certain syllables at low frame rates (less than 5 frames per second) caused by a mismatch between visual and auditory cues*' [29].

Meetings are highly interactive and as such they require low levels of end-to-end delay and jitter. Miras [30] suggests that the end-to-end delay should be no more than 100 ms to 150 ms, 40 ms to 75 ms is deemed noticeable but tolerable and jitter below 40 ms not noticeable within the meeting.

Massaro et al [31] showed that the perception of audio and video is dependent upon whether the audio lags the video, or vice versa. If the audio lags the video it was found that delays of up to 200 ms are not noticed, whereas when the video lags the audio delays up to 100 ms are not noticeable. The delay between the audio and video stream should be less than 50 ms – 100 ms for VCR quality audio and video. Low bandwidth audio and video have much rougher synchronisation requirements at around 400 ms.

2.1.1.2. Tutorial

There is a single person who drives the tutorial (the tutor) who is able to pass on and answer any question that the tutees may have. The tutor will take up most of the time speaking but the tutees will also have a more interactive role since they will question the tutor in order to clarify any points. The tutor may also question the tutees in order to assess their understanding and provide more focused help.

With this description a tutorial is more interactive than a lecture but has more of a rigid structure than a meeting. There is typically a single person who dominates the tutorial (the tutor). This is the main time that tutees are able to solidify the understanding they have from lectures. Therefore it is important that they are able to interactive with and understand the tutor.

Visual aids are sometimes used in order to help the understanding that students have, for example a shared whiteboard application [32-34] may be used.

A tutorial in this sense fits into the ETNA taxonomy [19] as interactive Foreground telepresence. In Foreground telepresence the '*speech channel is the primary means of communication making audio particularly important*' [19].

2.1.1.3. Lecture

Rettinger [27] describes a lecture as:

> '*Lecture mode is typically a one-to-many interaction. There are distinct and unequal roles of the participants. There is typically one lecturer and multiple students. The lecturer is in control of the conference. The lecturer may ask for interaction from the students in the form of questions or discussion. Students may indicate their desire for interaction by raising their hand. Lecture mode utilizes asymmetric communication among the participants.* [27]

A lecture typically consists of one person (the lecturer) presenting information to a group (the students). The group size can vary widely from single digits into the hundreds. The variation in the group size does not normally alter the level of interaction which is present. The implication for lectures is that they are typically low or non-interactive entities. There

is normally a set of visual aids[3] which accompanies a lecture. This means that a lecturing situation would for the majority of the interaction fit into the ETNA taxonomy as foreground teledata. There are situations where interpersonal communications between the lecturer and the students is made, such as a question and answer session at the end. These situations class as interactive foreground telepresence, with the lecturer responding directly to the questions of the students.

Kies et al [35] examined distance learning and found that students had difficulty maintaining attention when the video quality was low.

Kies et al [36] performed detailed studies in distance education and found that video quality did not effect understanding in distance education but they recommended a resolution of 320x240 with a frame-rate of 6 Hz. Bruce [37] suggests a frame-rate of 17 Hz in order to provide access to facial cues such as lip movements with Kies et al [38] suggesting a lower limit of 6 frames per second for distance education situations.

Since a lecture is less interactive than either a meeting or a tutorial it is possible to use higher quality CODECs at the expense of end-to-end delay.

2.1.2. Summary

Audio is the most important aspect of a videoconference with the majority of information being transmitted via speech. The case for video is less clear cut. It has been shown that the addition of video does not improve the performance of a task within a videoconference but it does aid the interpersonal interactions of the participants especially if the participants do not know each other. This has the implication that the quality of the audio should be maintained at an acceptable level at the expense of the video quality.

[3] These are normally in the form of a number of slides which are presented while the speaker is lecturing.

For all situations there is a minimum acceptable frame-rate (around 5 Hz) with Bruce [37] suggesting 17 Hz in order to access facial cues. The minimum suggested spatial resolution is 320x240 pixels although many applications use 176x144.

2.2. Participants expectations

Bouch and Sasse [39] performed experiments on the expectation of audio quality and how it affects the assessment of the quality received. They concluded that *'an objective level of quality will be judged as being of a comparatively higher subjective quality if it is stable'.*

Hands and Avon [40] showed that the subjective ratings of digital images are dependent on the order in which the images are presented. Users react negatively when poorer quality images are shown after higher quality images.

2.3. Network characteristics of audio and video

This section described the requirements made upon the network when using it for video and audio conferencing. Knowledge of the requirements on the network allows us to have an informed decision when configuring videoconferencing sessions. It shows how the network conditions affect the media streams as they are transported across the network.

2.3.1. Network QoS parameter

The quality of media which has been transported across a network depends on the underlying network used. The most significant network based factors that influence the end-to-end quality of an application are bandwidth, delay, jitter and loss.

- *Bandwidth* – This is the amount of data per time slice which can be transported end-to-end within the application. Lower bandwidths mean a reduced amount of data that can be transported from end-to-end within a videoconferencing application,

meaning that information is lost and leading to a reduced quality of experience within the session.

- *Delay* – This is the time that it takes data to be transferred from one multimedia end-point to another within the applications. The multimedia end-points are normally desktop computers being used as a videoconferencing station. There are two types of delay which are of importance. The first is the one-way delay from one videoconferencing station to another. The second type of delay, commonly known as the round trip time (RTT), is the time that it takes for a packet transmitted to reach an end-point and an acknowledgement received at the origin. The RTT is not necessarily the end to end delay multiplied by two due to the non-symmetrical nature of IP routing.

- *Jitter* – Jitter is usually caused by the buffers built up on routers during periods of increased traffic and less often by changes in routing due to failures. Jitter within an audio or video stream leads to difficulty in understanding the participants. Jitter can be smoothed out at the expense of delay using a play-out buffer but large amounts of jitter being smoothed results in an unacceptable delay.

- *Packet Loss* – Packet loss is typically the result of congestion within the network. Packet loss is defined as the fraction of IP data packets, out of the total number of packet transmitted, which are lost along the path from the sender to the destination.

- *Packet Loss Pattern* – The pattern of the packet loss is also important when dealing with the transport of audio and video over an IP network. The most useful observation about the state of packet loss patterns within the Internet is that packet loss usually occurs in short bursts; see section 2.4 for details. This information is

useful in the design of efficient audio and video CODECs which are able to encode media for short periods of continuous loss.

2.3.2. Network requirements of digital video transmission

Many different video CODECs have been designed for many different purposes. Some of these CODECs are not designed for transport over a '*lossy*' network and instead are designed for storage of multimedia on other forms of storage media and react badly to packet loss or low network bandwidth. Table 1 shows the bandwidth requirements of typical video CODECs. Most of the CODECs shown are not suitable for network transfer due to either high bandwidth requirements as in the base of HDTV through to MPEG-1 or the CODEC reacts badly to packet loss.

Video Format	Bit-rate	Encoding Delay (ms)
Uncompressed HDTV	1.5 Gbps	0
MPEG-2 Standard Definition TV (SDTV)	6 Mbps	40 – 500 [41]
MPEG-1	1.5 Mbps	40 – 500 [42]
H.263	28 Kbps – 1 Mbps	60 - 200 [43]
H.263+	28 Kbps – 1 Mbps	60 - 200 [44]
H.261	28 Kbps – 1 Mbps	20 [45]

Table 1 Bandwidth requirements of video formats.

From the table, above, it can be seen that only two of the CODECs have typical bandwidth requirements that match up with those provided by typical edge network technologies (modem, ISDN, ADSL)

2.3.3. Network requirements of audio transmission

Some typical parameters for network audio are shown in Table 2. The end-to-end delay of the CODEC shows how long it takes to record/encode then decode/play back the audio. There is a trade off; typically CODECs which require more time to encode the audio will also have lower bit-rates and vice versa.

CODEC	Bit-rate (Kbps)	Encoding Delay (ms)
G.711	48, 56, 64	1 [46]
G.722	48, 56, 64	2 [47]
G.723.1	5.3, 6.3	> 40 [48]
G.726	32	20 [49]
G.728	16	1-15 [50]
G.729	8	25 - 35 [51]
G.729 annex A	8	25 – 35 [51]
GSM-EFR	12.2	40 [51]
MP3	64 - 384	59 – 240 [52]

Table 2 Network requirements of network audio

Audio CODECs have different subjective quality levels. Nortel Networks [49] provides Mean Objective Scores (MOS) for five of the above CODECs. Both G.771 and GSM-EFR score well but G.711 has a high bit-rate requirement (48 – 64 Kbps) while GSM-EFR has a high delay cost. For a situation where low delay and low bit-rate is required absolute quality in terms of MOS score will have to be sacrificed and perhaps G.726 or G.729 employed instead.

2.3.4. Synchronisation

The proper play-out of the audio and the video depends on the audio and video being synchronised. Audio/Video streams are distracting if lip-synchronisation is not maintained. Finally there is a possible issue of face expression not matching the spoken word. This could cause problems since facial expressions are used to alter the meaning of the spoken word. The mismatch of the spoken word and the facial expression could alter the perceived meaning of the spoken word. This could lead to misunderstanding and would be detrimental to the conference.

Massaro et al [31] showed that the perception of audio and video is dependent upon whether the audio lags the video or vice versa. If the audio lags the video it was found that delays of up to 200 ms are not noticed, whereas when the video lags the audio delays up to 100 ms are not noticeable. The delay between the audio and video stream should be less than 50 – 100 ms for VCR quality audio and video. Low bandwidth audio and video have much rougher synchronisation requirements at around 400 ms.

2.3.5. Summary

The ideal conditions for networked audio and video are high bandwidth, low delay and negligible jitter, but this situation is rarely realized on the Internet. Networked video (using H.261) bandwidth requirements range from 28 Kbps up to 1 Mbps. The lower bit-rates provide limited support for the interactive nature of video conferencing, due to the reduced frame-rates. The bandwidth requirements for audio are significantly lower at a minimum of 5.3 Kbps but it does has a significant amount of delay at 67 ms. There is a trade off to be made for both the audio and video transport. In order to provide the maximum amount of compression for the media stream extra time must be spent processing the data, this adds extra delay at decoding but mainly during the encoding process. The requirements that the

chosen audio or video CODEC place upon the system should be factored into the process which configures the videoconferences.

2.4. IP traffic characteristics

In this section evidence is provided for the patterns exhibited on the Internet, in both the wider backbone and the traffic patterns which can be expected at the network edges. First we shall be looking at various experiments conducted in order to give an insight into traffic patterns. Then we shall examine traffic patterns on the Internet backbone and those at the end-points of the network.

In 1997 Thompson et al [53] reported upon the wide area traffic patterns as measured from two locations within internetMCI's[4] backbone. Their analysis shows that TCP based traffic is 95% of traffic by bytes; 90% of traffic in terms of number of packets and at least 75% of all the flows on the link. UDP accounts for the second highest usage at 5% of bytes, 15% of packets and 20% of the flows.

Of IP traffic, TCP accounts for 95% of the data by bytes, 85% - 95% of the packets, and 75-85% of the flows. TCP flows average fewer than 20 packets, about 7 kilobytes and fewer than 20 seconds in duration. UDP makes up most of the remaining IP traffic and ICMP packets account for less than 1% of all packets.

Traffic between the US and the UK is dominated by the US-to-UK direction in packet and bytes, except for bytes transferred in the evening hours US Eastern Daylight Time (EDT). There is also a 15% to 25% drop in the total traffic over the weekends.

[4] http://www.internetmci.com/

In 1997 web traffic dominated as the single largest Internet application. Web traffic accounted for more than half the bytes (65% to 80%), packets (55% to 75%) and flows (65% to 75%). Web server traffic averages 10 kilobytes per flow, 15 packets per flow and 13 seconds per flow. The web client on the other hand averages 1 kilobyte per flow with 15 packets per flow and with a flow lasting about 13 seconds. This is due to the asynchronous nature of the request and reply message sizes typically found in HTTP traffic.

Other TCP applications such as FTP and NNTP increase during the evening and night hours; however they rarely exceed 10% of the overall traffic.

Of UDP applications, DNS traffic accounts for a third to half of all UDP traffic, depending upon the time of day. The majority of this traffic is limited to the backbone and is server-to-server communications.

2.4.1. Packet loss

Numerous researchers [54-58] have studied packet loss on the Internet, but due to the enormous diversity of the Internet few studies agree on an average packet loss and an average loss burst length. According to Bolot [54] loss varies between 11% and 23% dependant upon the inter-packet spacing. Borella [57] reports a loss range of 0.36% to 3.54% depending upon the path. Yajnik et al [59] report packet loss between 1.38% and 11% depending on the location of the Mbone receiver. Paxson [60] reports packet loss in the range of 2.7% and 5.2% depending upon the year of the experiment. Balakrishnan et al [61] reported packet loss at an average of 0.49%.

A common observation of various studies [54, 58, 62, 63] was that packet loss events are correlated for up to 200 ms while [58] reports correlation for up to one second.

The results from Loguinov [64] are the most interesting as they show packet loss variations depending upon the time of day. Their results show that packet loss rates increase at 9 am and remain high until 6 pm[5] when it lowers slightly, but is still higher than the early morning rate. This is consistent with people using the Internet in the evening after they have come home from work, or perhaps working in the evening. The lower rates indicate a reduced usage during the evening.

Paxson [56] reports that less than 1% of packets are delivered out of order while the experiments performed by Moon et al [62] report that out of order packets are less than 0.1% of the total number of packets in their experiments. With a time sensitive application, such as videoconferencing, packet reordering may have to be treated as packet loss. This is because the reordered packet may arrive too late for use in the playback of the media stream.

2.4.2. Round trip time

Loguinov [64] presents the mean round trip times over the course of 24 hours with a 3 hour granularity. They show a pattern where the round trip time increases during working hours (9 am to 5 pm) with a lower level outside of working hours.

2.4.3. Bandwidth

Both [53] and [65] provide evidence for temporal aspect of Internet traffic in relation to the bandwidth which is being used.

Figure 3 shows the traffic patterns found on a domestic link in the US in 1997. The left hand graph shows the pattern for a single day; whereas the right hand graph is for a week. As can be seen in the left-hand graph, the traffic hits a high around midday and remains that way until around 6pm at which time it gradually tails off to a low at 6am from which it increases.

[5] The results were gather from 1188 US cities and are reported in Eastern Daylight time (EDT).

The right hand graph shows clearly the patterns over the course of the week starting on Sunday and ending on Saturday.

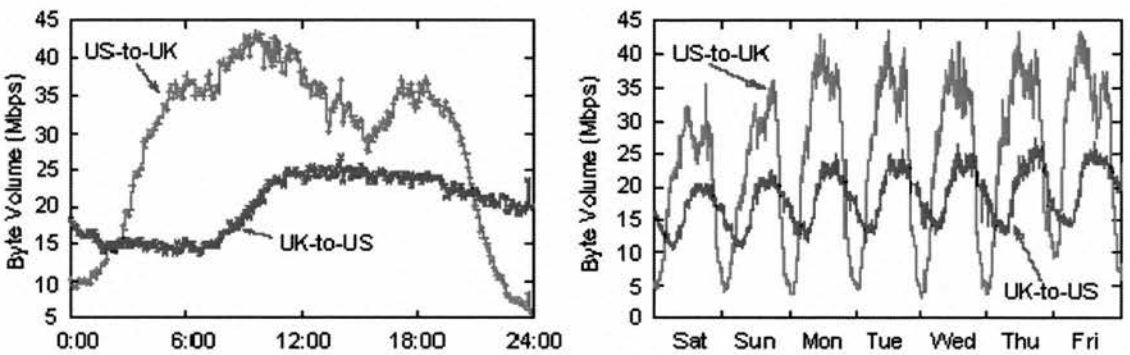


Figure 3 Domestic link traffic patterns. Reproduced from [53]

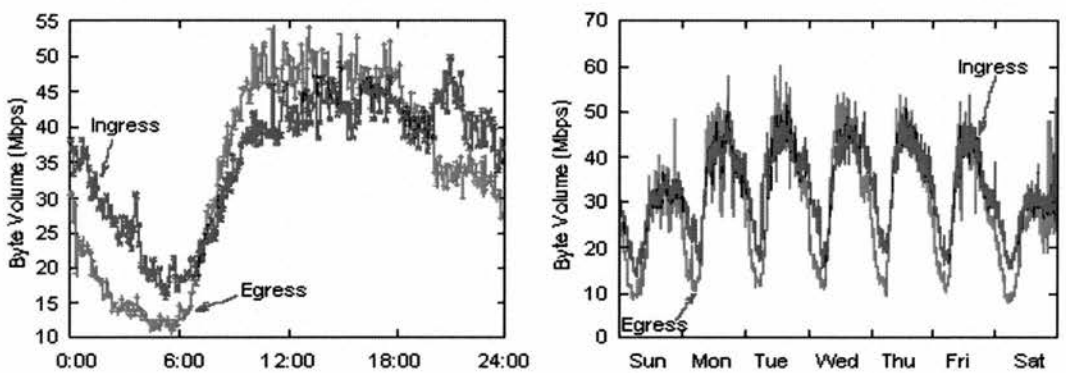


Figure 4 Byte volume on the UK to US link. 24 hour period. Right 7 day period. Reproduced from [53]

Figure 4 shows the traffic patterns of the transatlantic link, between the US and the UK in 1997. The times shown are for the East Coast of the US[6]. The left graph being the pattern for a twenty-four hour period and the right hand-side shows the traffic pattern for a seven day period. There is a pattern of higher traffic usage during the day, with low usage during the night. The much higher bandwidth usage of the US-to-UK link can be explained since a large number of commonly used websites are located in the US, and therefore few requests

[6] Which is GMT – 5 or GMT – 6 depending upon the time of year.

will be made from the US side of the link for web pages on the UK whereas at the UK side many requests are made to the US for web pages. The pattern of traffic usage shown on the US-to-UK link is echoed in the UK-to-US traffic pattern which is time shifted by 5 to 6 hours. Thomson et al [53] ascribe this shift to the time difference between the US East Coast and the UK/Europe.

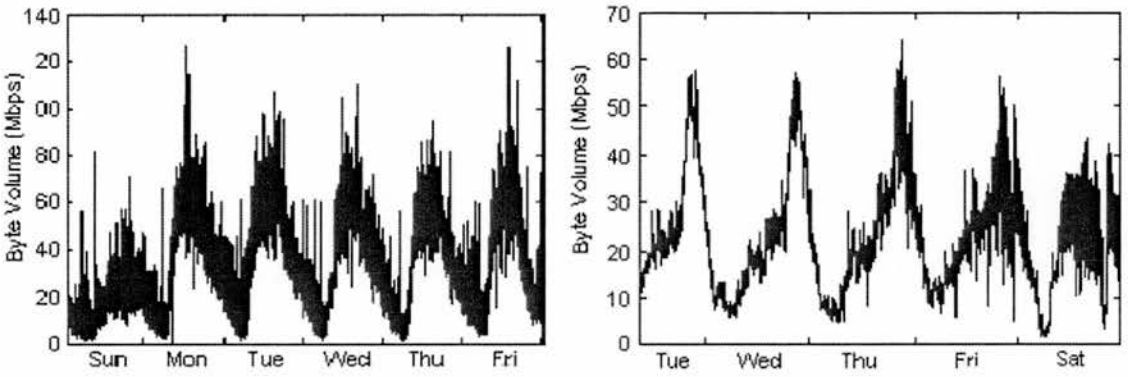


Figure 5 Traffic Patterns; left research IPs; right dial-up IPs reproduced from [65]

Figure 5 shows two traces of Internet traffic during the day from 2001, this is several years after the adoption of P2P networking in the form of Napster [66]. The network traffic still shows the daily pattern as was evident from the 1997 study.

The left hand graph shows traffic measurements from research institutions, these having large pipes to the Internet which are being used mainly during the day. The graph clearly shows the large jump in traffic usage during working hours[7] which tails off into the evening. The right hand graph shows the network usage for dial-up IPs which are most likely to be used from an individual's home. This graph takes the opposite traffic organisation, so that

[7] Between 9 am and 5 pm.

the peak usage is during the evening with minimal usage during the day, presumably when most people are at work which accounts for the increase in traffic observed.

CacheLogic [67] performed network measurements from January 2004 to June 2004. They showed that although the makeup of the network traffic has changed, (HTTP is no longer the dominant protocol on the Internet in terms of bytes) the temporal nature of network usage remains. Peer-to-peer traffic bandwidth usage is reasonably constant over time while the non peer-to-peer traffic varies, creating the temporal patterns which have been seen.

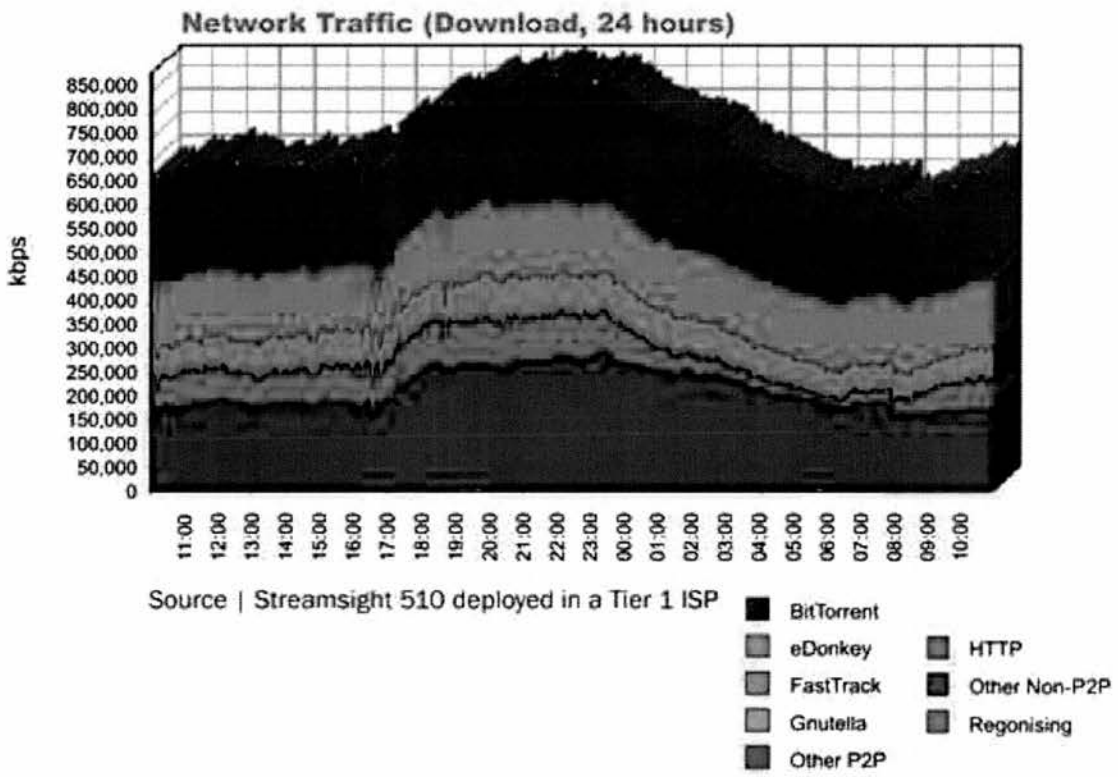


Figure 6 Bandwidth trace from first half 2004, reproduced from [67]

2.4.4. Mbone traffic behaviour

The multicast backbone (multicast backbone) [4] is an overlay network on top of IPv4 [68]. It adds multicast capabilities between different sites on the Internet even when they are not

connected by multicast capable routers. It provides a way of interconnecting distinct sites on the Internet so that multicast traffic is able to traverse the Internet from one site to another.

Yajnik et al [59] performed a series of experiments on collecting packet loss information over the Mbone at 12 distinct locations spread over Europe and the USA. The primary goal of their research was to investigate the temporal and spatial correlation in packet loss amongst participants in a multicast session.

The measurements were performed by monitoring and recording the received multicast packets during an audio multicast session. There were three different audio sources, "World Radio Network" (WRN) transmitted from Washington DC; the "UC Berkeley Multimedia Seminar" (UCB) transmitted from California and "Radio Free Vat" (RFV) which also transmitted from California. The WRN source transmitted packets at 80 ms intervals, each packet containing 5 kilobits of audio data. The UCB source transmitted at double the rate, at 40 ms intervals with each packet containing 2.5 kilobits worth of audio data. For the 19th April 1996 trace RFV transmitted at 80 ms and for the 8th May 1996 trace it transmitted at 40 ms intervals.

During the course of the experiment, 14 different sets of traces have been collected. The duration of the traces varies from 14 to 99 minutes.

For most of the traces the loss on the backbone links of the Mbone network is observed to be small (2% or less), as compared to the average loss seen by a receiver. However, due to occasional outages lasting from a few seconds to a few minutes in some backbone links, spatially correlated loss between receivers reaches 20%, in a few datasets.

There is a significant amount of consecutive losses at each site. One or more extremely long loss bursts, lasting from a few seconds up to 3 minutes (around 2000 consecutive packets),

occur in almost every trace. Such long loss bursts have been reported in [60] for the case of point-to-point connections.

Most of the loss bursts consist of isolated single losses, but a few very long loss bursts contribute heavily to the total packet loss.

Some receivers see periodic packet loss lasting for approximately 0.6 seconds (8 consecutive packets) and occurring at 30 second intervals. This is possibly due to the routing updates as reported in [69].

Any packet loss which is the result of routing updates will apply to all packets no matter what data the packets contain since the loss is not caused by the packets. Any losses which are directly caused by the data packets themselves would be amplified when dealing with video data, as it can be an order of magnitude larger than audio.

The packets all contained audio data but the results of this experiment apply to other types of data such as video.

2.4.5. TCP behaviour

Most multimedia data is carried using UDP based protocols. TCP is not the natural transport protocol for multimedia due to timeliness constraints but the bulk of traffic on the Internet is carried by TCP; HTTP, BitTorrent , SMTP etc. This means that any multimedia conferencing application will have to exist in an environment where the majority of traffic it is competing against is TCP based. In order to be a good network citizen and not degrade the quality of service for others it is important to be TCP friendly. This means that any system introduced should operate along side TCP in a way that is no more detrimental to the network than TCP would be. In order to fulfill these requirements it is important to understand how TCP operates.

In 1987 the Internet suffered a series of congestion collapses [70]. This was because there was no rate control on TCP so any application was free to send at any rate it wished[8]. If there was packet loss (due to congestion on a network) then packets were retransmitted. This situation quickly degenerates to a state where most of the traffic on the network is retransmissions of packets which were lost due to congestion.

In 1988 Jacobson et al [71] provided a solution to this problem in the form of TCP Tahoe which was the first version of TCP to implement a congestion control mechanism. The three algorithms which dictated the new congestion control mechanism of TCP are:

1. Additive Increase Multiplicative Decrease
2. Slow start
3. Fast retransmit

Additive Increase Multiplicative Decrease (AIMD) increases the window size of the TCP connection by one when no packet loss is detected. When packet loss is detected, the window size is halved. This leads to a '*sawtooth*' trace behaviour (Figure 7) where the bandwidth used increases until congestion is detected at which point the bandwidth is dropped. This view of TCP is not completely accurate, the '*bursty*' nature of packet loss on the Internet means that when a packet is lost the following packet is also likely lost. This means that a single burst of packet loss will greatly reduce the transmission rate.

[8] This is similar to UDP at the moment although even then TCP was a reliable protocol via retransmissions.

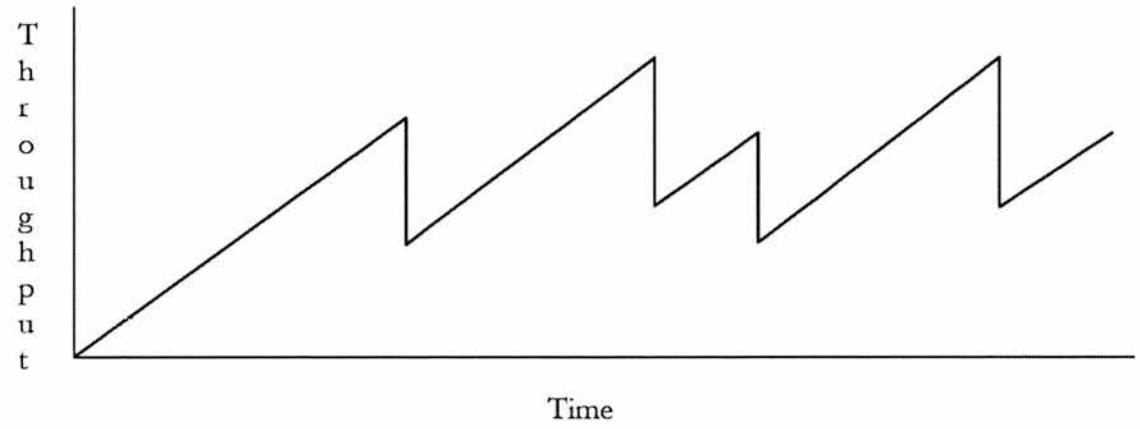


Figure 7 AIMD 'sawtooth' behaviour

With AIMD the initial problem is what to set the initial window size to. If the initial size is set too low then short lived connections will not be able to fully utilise the bandwidth available to them. If the initial window size is set too high then the large initial burst may cause an overflow at a router and cause packet loss. The solution to this problem was the *slow start* algorithm. This works by setting the initial window size to one and then doubling the window size each round trip time until packet loss is detected. When packet loss is detected, the *slow start* is ended and the AIMD period is started. Slow start is also used when the connection goes dead while waiting for a timeout on an ACK packet. The problem of setting the initial window size has been addressed by Miller [72]. His solution to the problem was to take window size measurements for different paths. When a TCP connection was being setup the window size to use initially was configured from previous measurements.

Finally *fast retransmit* solved the problem of coarse grain TCP timeouts leading to idle periods. With fast retransmit an ACK is sent on every packet reception. When packets are received out of order the previous ACK is retransmitted. The duplicate ACKs at the sender side are used to trigger retransmission of missing packets.

This original algorithm has since been replaced by TCP Reno [73] which has several optimisations over the original.

Since TCP is the most widely used transport level protocol on the Internet it is desirable to interact with it in a way which is not detrimental to TCP streams and to the network as a whole. The term for this behaviour is TCP considerate, or TCP friendly. Braden et al [74] provide the definition for TCP friendly as '*responsive to congestion notification, and in steady-state it uses no more bandwidth than a conformant TCP running under comparable conditions (drop rate, RTT, MTU, etc.)*' [74].

In order to implement an equivalent to TCP's congestion control, it is necessary to understand the effects of packet loss on the overall performance of a TCP connection. Background packet loss affects the bandwidth of a TCP connection in the following way [75], see Equation 1.

$$Bandwidth = \frac{1.3 * MSS}{RTT * \sqrt{Loss}}$$

Equation 1 TCP Bandwidth, reproduced from [75]

MSS is the maximum segment size (packet size), in bytes, used on the connection. The RTT is the round trip time expressed in milliseconds and loss is the measured probability of loss per packet being experienced by the connection. The Bandwidth is in kilobits per second. Equation 1 can be used to determine if a network stream is TCP friendly or not. There are several other TCP friendly equations which target the various incarnations of TCP, for example [76] presents a throughput equation for modeling TCP Reno while [77] does the same for NewReno [78].

Krasic et al [79] have argued the case for using TCP as the transport protocol for interactive multimedia while others [80-83] have argued against TCP, citing packet retransmission and congestion control as a reason for not using it. The arguments against rely on packet retransmission increasing the latency and thusly reducing the interactive nature of a conference (be it audio or video). Congestion control creates a *'sawtooth'* bandwidth usage which could cause the absolute quality to vary.

The use of Explicit Congestion Notification (ECN) [84] leads to a reduction in packet loss [85]. Packets which would normally be lost due to congestion are no longer lost leading to the situation where the network is being used more optimally. ECN could relieve the problems of packet retransmission and thusly reduce the delay on a given path enabling TCP to be used as a transport protocol for multimedia data. This would rely on ECN enabled routers at all points along the path, arguably a more likely situation than the widespread adoption of resource reservation. Floyd [84] states that ECN reduces the delay for low bandwidth TCP connections but this statement is based on TELNET [86] traffic which is an order of magnitude smaller per connection than even low bandwidth audio data so cannot be used as a basis for the case of multimedia data.

2.4.5.1. UDP rate control

Rate based applications are those which do not use a congestion window to control the amount of data on the network. Most rate-based applications have latitude in their choice of data rates since they use UDP as a transport protocol. Applications using TCP are limited by the congestion control algorithm. In order to implement a TCP friendly congestion control algorithm, these applications should simply choose to send at a rate no higher than a TCP connection would if it were operating along the same path.

In times when there is no congestion these applications may send at their desired maximum rate. As long as the overall loss rate is low enough that an equivalent TCP connection would attain at least the same bandwidth the connection may continue to send data at its preferred rate. If the loss rate rises so that a TCP connection would not be able to maintain the data rate the connection should reduce its data rate.

In order to accurately compute the bandwidth a TCP connection would obtain, it is necessary for the application to know the round trip time, maximum transfer unit (MTU) and the current loss rate on the network path. Current values for round trip time and loss should be kept by using recent averages over the last several round trips. The estimate of the round trip time should be made at least once per round trip time [87].

After modifying their sending rate, prior to performing additional increases or decreases to their sending rate, applications should wait for at least one round trip time to obtain a new RTT and loss rate measurement. This should prevent applications from over-reacting to congestion. If the loss rate is high enough that TCP bandwidth falls below the minimum rate required by the application, the application should either exit gracefully or inform the user and allow her to make a choice on what to do.

2.5. Conclusions

The issues to be addressed by a videoconference support system are as follows: audio and video have demanding network quality of service constraints; the Internet does not provide QoS, but exhibits patterns which may be exploited; as TCP is the dominant protocol on the Internet, videoconference traffic should be TCP friendly and not act as a resource hog; participants prefer consistency in QoS rather than changes during a session.

Networked audio and video have very tight quality of service constraints. The end-to-end delay for interactive audio video traffic should be no more than 150 ms with the jitter for the connection below 40 ms. This ensures that there is no perceivable delay in the connection. Above 75 ms jitter renders the connection unusable while between 40 and 75 ms the jitter is noticeable but tolerable. Audio and video should be synchronised to within 400 ms of each other with low quality video and to within 50 ms to 100 ms when using high quality video.

Internet traffic exhibits patterns of traffic which repeat within the space of twenty four hours during the working day and also similar patterns during the weekends. Packet loss demonstrates a '*bursty*' behaviour where packet loss is more likely to occur consecutively. Packet loss shows a spatial correlation. For a large multicast session using the Mbone every packet is lost by at least one receiver. High levels of packet loss are the most significant factor.

TCP is by far the dominant method of data transfer on the Internet. As such any mechanism for providing quality of service for network traffic will have to exist and interact with TCP and not have a detrimental effect of the network.

Bouch et al and Hands et al showed the quality expectations that people have are based upon the previous quality that they have experienced. For example, VHS quality was considered good enough for a long time until DVD video was released and now the quality of VHS looks poor. It is not that the quality of VHS has changed it is just that the observer now has a higher expectation for the video quality. It is the change in quality that is distracting and not the absolute quality itself. A low expectation of quality means that lower qualities are accepted whereas with a high expectation of quality lower qualities are dismissed as being not good enough and are seen as distracting.

Chapter 3

Related Work

3. Related work

The problem of providing quality for multimedia applications has been tackled many times before. This chapter describes various methods which have been pursued in the past. We first look at generic quality of service frameworks which aim to provide quality of service at the network level. Then we move onto the more specific multimedia frameworks which have quality of service provision. We then examine toolkits for the creation of multimedia applications and protocols which have been designed for the transport of multimedia. These include protocols which have quality of service support as well as those that do not. Finally a description of generic methods to aid quality of service support is described including methods for dealing with packet loss.

3.1. Network QoS approaches

Here the quality of service approaches which are most commonly used in research to provide network level quality of service are examined. These techniques are generic and may be used for any application which requires tight bounds on the quality of service.

3.1.1. Resource reservation

Resource reservation techniques are well known [88] and have been well studied, for example [89]. These techniques rely on closely restricting access to finite resources by controlling admission.

When an application wishes to use a resource it requests a certain quality of service from the resource manager. In the case of a network resource this may be a certain bandwidth with bounds on the delay, jitter and packet loss. Since it is excess usage of a path which creates delay, jitter and packet loss, any stream which would adversely affect the others can be denied

access to the resource, or negotiation can be used in order to allow access with lower QoS demands (e.g. Available Bit Rate in ATM [90]), and therefore less likely to impact on others.

The use of resource reservation requires state to be stored at each hop along the route which the data stream is taking. This introduces a potentially large amount of state which must be stored. Routers at the core of the network which are potentially dealing with millions of streams could be overwhelmed and as such resource reservation has been shown not to scale. It is therefore not a suitable solution for wide scale deployment of end-to-end quality of service provision on the Internet. This does not exclude it from use at the network edge where the amount of traffic and therefore state is at more manageable levels.

Two well known examples of resource reservation are ATM [90] and Integrated Services (IntServ)[91]. IntServ typically provides three classes of service which traffic may fit into:

1. Guaranteed – for delay bound service agreements
2. Controlled-load – for a statistical delay service (nominal mean delay)
3. Best-effort – which is further broken down into three more categories
    a. Interactive burst
    b. Interactive bulk
    c. Asynchronous

Within these classifications the application also sets the parameters of network conditions that it would like to experience. These parameters are then used at each hop along the route in order to provide the quality of service for the traffic stream.

Unfortunately, the required level of investment is unlikely to come unless there is significant cause which seems unlikely according to [92] which fails to find a definitive reason to move over to a resource reservation capable Internet architecture.

3.1.2. Aggregated flows

With aggregated flows traffic streams are placed into one of a number of traffic classifications. These classifications are then treated differently within the core of the network.

An example of aggregated flows is Differentiated Services (DiffServ) [93]. Aggregated flows are simpler than either ATM or IntServ. DiffServ is an extension of TCP like IntServ but (unlike IntServ) it has been designed to scale to the global level that is required of systems deployed on the Internet. Traffic at the edges of the network is classified into classes and routers at the core level of the network have to schedule traffic between these traffic classes. Routers at the edge of the network maintain per flow state but since this is a much smaller quantity than at the core, this is not a problem.

Routers at the core of the network typically only have to deal with a limited number of classifications, up to 10 or so, which is much more manageable. Applications have to place their network stream into one of the predefined classes rather than having a complete traffic specification which is available in IntServ or ATM.

There is some question as to the scaling capability of the techniques which use resource reservation [94] since these techniques require state to be stored at all the routers along the network path.

3.1.3. Less than best effort

Less than Best Effort (LBE) is a technique whereby packets can be marked as 'unimportant' and therefore do not need the tight controls on quality of service that a more interactive application might need. When congestion occurs at a router the LBE packets can be dropped first. This altruistic approach leaves network paths clear for high priority traffic when congestion is detected.

Current examples [95] of LBE are used as a complimentary technique along with DiffServ. LBE could be deployed more widely since there is no state to store except in the packet header and the decision to drop the packets is made at the router during times of congestion.

Traffic can be marked as LBE either by a site's egress router or by the user directly. Unfortunately it is doubtful that people would be willing to mark their traffic as LBE since to each person their traffic is no less important than anyone else's. So it would most likely be performed by a QoS broker at the edge of the network.

Less than Best Effort is a simple system which offers a primitive form of quality of service. It does not provide assurances for traffic traversing the network and in fact has the opposite effect as some traffic will be sacrificed in order to reduce queues at network routers. The majority of traffic is TCP based so any lost packets will be retransmitted, thereby resubmitting the lost packets to the network, although at a lower transmission rate then previously.

3.1.4. Conclusions

All of these systems require a large level of deployment in order for them to work effectively. DiffServ can be employed at the network edges and still have an effect whereas ATM and IntServ requirement a deployment throughout the network in order to be effective. This

would mean a large scale upgrade of the network fabric. The cost of this venture would be rather large with only a relatively small number of people and organisations requiring this kind of upgrade. There is also some question as to the scaling capability of the techniques which use resource reservation since these techniques require state to be stored at all the routers along the network path.

The main challenge for any of the quality of service systems reviewed is deployment; they require a large amount of investment into the network in order to work effectively. Unless there is significant cause to move over to an architecture which provides QoS it is unlikely to appear in the short to medium term.

3.2. Multimedia quality of service frameworks

This section introduces a variety of frameworks which aim to provide quality of service for multimedia applications.

3.2.1. OMEGA

The OMEGA [96] architecture is a project at the University of Pennsylvania. It is the result of investigating the relationship between application quality of service demands and the ability of the local operating system as well as the communications infrastructure to support these demands. OMEGA assumes a network that provides bounds on delay, errors, and that can meet bandwidth demands together with an operating system which provides run time quality of service. This assumption is not valid for the Internet and therefore OMEGA is unusable at the moment on the general purpose Internet. Communication is preceded by a call setup where application requirements are negotiated, resources reserved and guarantees are made at several levels such as network and operating system.

Guarantees on resources are negotiated during the call establishment phase by the QoS Broker protocol [97] which also performs negotiations for network, end-system and application to application QoS. This negotiation allows resource providers to ask applications to change their resource usage.

3.2.2. Tenet architecture

The University of California at Berkeley has developed a family of protocols which have been designed to run over ATM networks. The Tenet Architecture [98, 99, 100] consists of the following protocols;

- Real Time Channel Administration Protocol (RCAP)
- Real Time Internet Protocol (RTIP)
- Continuous Media Transport Protocol (CMTP)

The Real Time Channel Administration Protocol provides generic support for establishing connections, resource reservation as well as signalling for the other protocols. RCAP spans both the transport and network layers.

The Continuous Media Transport Protocol is designed to support continuous media. CMTP is a lightweight protocol which runs on the top of the Real Time Internet Protocol. CMTP provides sequenced and periodic delivery of continuous media samples. It provides quality of service control over throughput, delays and error bounds for a media stream.

The Real Time Internet Protocol is a lightweight protocol for the delivery of real time data over the Internet. Its main function is to deliver packets which meet the channels' quality of service requirements. RTIP performs rate and jitter control as well as scheduling based on the quality of service parameters of each connection. It provides an unreliable, simplex,

guaranteed performance. This protocol assumes that all packets will arrive in order, since the assumption is that all packets take the same route through the network. However, as stated in [99] RTIP packets can be encapsulated into IP datagrams for wide scale use where routers do not support RTIP natively. If this is the case the quality of service guarantees can no longer be supported since we default to the quality of service provided by IP, i.e. best effort. Of course the majority of routers on the Internet do not natively support RTIP and therefore the default of IP which means that the QoS aspects are lost.

Each RTIP end-point contains a regulator and a scheduler. The regulator monitors the traffic on each connection and shapes it in order for it to meet the traffic specification of the connection. The scheduler ensures that no packet remains in a node longer than their local delay bounds.

The Real-Time Message Transport Protocol (RMTP) is an abstraction upon RTIP. RMTP is at the same network layer as TCP but it is lighter weight and unreliable. It was deemed that retransmission of lost packets was unnecessary due to the real-time nature of the application. The retransmitted data would be out of date by the time it had been received and therefore of very little use. The main purpose of RMTP is to provide packet fragmentation and assembly, meaning that RMTP is able to transport messages which are larger than the maximum RTIP packet size.

Tenet requires that all layers in a network's architecture are able to provide guaranteed performance services. This being the case, network technologies such as FDDI [101] and ATM are ideal for Tenet. Tenet requires this so that it is able to provide mathematically provable quality of service; in contrast it is not possible to guarantee delay in a network which is built using IEEE 802.3 Ethernet.

3.2.3. Heidelberg quality of service model

The HeiProject [102] at IBM's European Networking centre in Heidelberg have created a quality of service model which is designed to provide guaranteed quality of service in end systems as well as the network. It includes support for continuous media in the form of HeiTS/TP which provides support for quality of service mapping as well as media scaling. The key to providing the end-to-end quality of service is the Heidelberg Resource Administration Technique (HeiRAT) which consists of a quality of service management scheme including the following

- Quality of service negotiation
- Quality of service calculation
- Admission control
- Quality of service enforcement
- Resource Scheduling

The Heidelberg quality of service model was designed to handle the different quality of service demands from many receivers in a multicast group. Also it supports quality of service adaptively via flow filtering [103] and resource sharing [104] within the network, while providing media scaling [105] and CODEC translation [106] at the end systems.

3.2.4. OSI quality of service framework

The OSI QoS [107] framework concentrates mainly on the quality of service support for OSI communications. The OSI QoS framework defines the terminology and concepts for quality of service and provides a model which identifies objects of interest. The key concepts in the OSI model are:

- Quality of service requirements
- Quality of service characteristics
- Quality of service categories
- Quality of service management functions

There are two parts to the OSI QoS Framework management system which perform the monitoring, maintenance and the controlling of the end-to-end quality of service of a session.

The task of the policy control function is to determine which policy should be applied to a specific layer of an open system. Policies are layer specific and may include aspects of security, time-critical communications or resource control.

The task of the quality of service control function is to determine, select and configure the appropriate protocol entities to meet the layer specific quality of service requirements. The quality of service control function combines two system wide capabilities: tuning the performance of the protocol entities and modifying the capabilities of the remote system by using OSI systems management.

3.2.5. TINA quality of service framework

The TINA QoS framework [108] describes a framework for specifying the quality of service aspects of distributed telecommunications within a Computer Architecture context. The TINA QoS framework is designed to address the computational and engineering viewpoints of the distributed applications. It is based partly on the work from the ANSA QoS Framework and the CNET Framework.

In the computational model the quality of service parameters required to provide guarantees to objects are stated declaratively as service attributes. In the engineering viewpoint quality of

service mechanisms employed by resource managers are considered. Since the quality of service requirements are stated declaratively the applications are relieved of the burden of coping with complex resource management which would be needed for quality of service guarantees.

3.2.6. MASI end-to-end model

The MASI end-to-end model [109] is a quality of service framework which has been developed at the Laboratoire MASI in the Université Pierre et Marie Curie. Its primary objective is the provision of end-to-end quality of service. The MASI framework offers a generic framework for quality of service support in distributed systems which are running over ATM networks. This model is motivated by the following requirements:

- The need to map quality of service requirements from the ODP layer to specific resource modules in a clean and efficient manner.
- The need to resolve multimedia synchronisation requires multiple related ODP streams.
- The need to provide suitable communication protocol support for multimedia services which are being developed at the Université Pierre et Marie Curie.

The requirement of ATM means that MASI requires network level QoS provision which is not available on the general purpose Internet.

3.2.7. Network aware internet video encoding

Network Aware Internet Video Encoding (NAÏVE) [110] is a video encoding system which was designed to degrade gracefully when exposed to packet loss. In this system each packet of data is a self contained unit with no inter-dependencies on other packets. Each packet contains all the information that is needed to make the updates to the picture. The reasoning for this is that it makes efficient use of the network. If a single packet is lost then only a single

frame update is lost whereas with other encoding mechanism subsequent packets are rendered useless until a key frame is received.

NAÏVE is designed to degrade when it is exposed packet loss but there is no support for setting the initial sending rate.

3.2.8. Conclusions

The techniques discussed in this section often require provision of quality of service management from a lower level system such as DiffServ or ATM. As has been discussed in section 3.1, these architectures are not widely deployed and are therefore not able to provide support for multimedia quality of service in the short or medium term. Table 3 shows a summary of the requirements and capabilities of the reviewed QoS frameworks.

Multimedia QoS Framework	Network QoS	OS QoS	Admission Control	Group	Interactive Media
Omega	√	√	√		
Tenet	√	√	√		
Heidelberg	√	√	√	√	√
OSI	√	√	√		
TINA	√	√	√		
MASI	√	√			
NAIVE					√

Table 3 Summary of QoS Architectures

Many of these frameworks require that the network has some form of quality of service provision. The Internet currently does not have QoS provision outside of research networks

such as the Internet-2's Qbone [111]. The others attempt to create network QoS mechanisms on which to base their communications. The assumption of network level quality of service is therefore not safe. The argument could be made that once quality of service is provided at the network level then with current operating systems and computational hardware it is a relatively easy task to provide a videoconferencing system with end-to-end quality of service. The problem of network quality of service provision must first be solved.

3.3. Streaming protocols

This section describes the protocols which are used to transport media across the network. Firstly *Application Level Framing* is introduced since it is used in most systems to aid in providing better quality in the face of packet loss. Then a series of streaming protocols is presented.

3.3.1. Application level framing

In Application Level Framing (ALF) [112], the data for transport application data units are placed into each packet in a way which most makes sense for the application. It is the application that should be able to choose how to deal with packet loss, not the transport protocol as is the case with TCP. This impacts upon quality of service over the network since the data can be packetised in such a way to reduce the impact that packet loss has on the system. For example when using ALF with audio data the phonemes can be placed into packets so that if there is packet loss the data is easier to reconstruct or hide.

ALF has had a major influence upon the design of multimedia protocols as well as CODECs. Protocols are now designed with the concept of ALF in mind. CODECs for multimedia transport are now designed for packet recovery based on the ALF encoding.

3.3.2. Real time protocol

The Real Time Protocol (RTP) [106] provides an end-to-end transport mechanism for real time data such as audio and video over unicast or multicast network services. RTP does not provide any form of resource reservation and does not guarantee the quality of service that will be provided within the sessions. With RTP each type of media e.g. audio, video etc., is given a separate stream, which involves a separate address and port. So for example for a session that uses audio, video and a shared white board there would be three RTP streams.

Loss Collection RTP (LC-RTP) [113] is a caching technique for use with RTP. Essentially a caching server listens in on transmission of RTP data from the server to a client and makes a recording of the packets. It not only records the data that is received but also records packet losses. When an RTP session is over the caching server contacts the sending server and requests the missing packets. This process is repeated until the caching server has the entire media file. Then when the file is requested again from a different client but using the same cache the file is streamed from the cache with hopefully a reduced amount of experienced packet loss. LC-RTP is not suitable for use in real-time videoconferences (with the exception of for archival purposes) since error correction takes place after the session has finished.

3.3.2.1. Real time control protocol

The Real Time Control Protocol (RTCP) [106] attempts to alleviate the problems of lack of resource reservation and quality of service by providing the application using RTP a feedback mechanism for monitoring how the streams are performing in terms of jitter, bit-rate, delay and packet loss on the network paths that are being used.

The monitoring provided within RTCP takes the form of two types of report. Sender's reports are generated by a datasource while receiver's reports are generated by a datasink. The reports are transmitted to every participant within a session but these report messages are

capped at 5% of the total bandwidth for the session. This is fine for small numbers of participants but as the number of participants increases then the frequency that each participant sends its reports decreases in order to maintain the overhead of the reports at 5% of the total bandwidth.

3.3.2.2. Real time streaming protocol

Real Time Streaming Protocol (RTSP) [114] is a control protocol which compliments RTP. It provides a control mechanism in a HTTP friendly way. It allows the starting, stopping pausing and rewind/fast forward of media.

In the mainstream the use of RTP/RTCP coupled with RTSP has been adopted by Realmedia[9] server [115] and player [116] which has brought RTP widespread use.

3.3.3. Additive increase multiplicative decrease streaming protocols

This class of streaming protocols use a congestion control mechanism similar to that in TCP. The Additive Increase Multiplicative Decrease system increases the rate of sending data until congestion is detected then the rate is decreased. At this point the cycle restarts, the sending rate is increased until packet loss is detected leading to a reduction in the sending rate and so on. Two protocols of this form are presented and discussed.

3.3.3.1. Rate adaptation protocol

The aim of the Rate Adaptation Protocol (RAP) [83] is to create an architecture for delivery of layered-encoded stored real-time streams over the Internet. Applications using RAP should '*adapt their transmission rate adjusted properly and promptly*' so that '*a fair share of bandwidth for all the flows that coexist along the same path*'[83]. A target application could be a web-server or a video-on-demand server that provides access to a variety of multimedia streams for large

[9] http://www.realnetworks.com/

numbers of heterogeneous clients. The rate server's rate of transmission is continuously adjusted by the RAP in a TCP friendly fashion.

It has been previously shown that the AIMD algorithm converges to a fair state [117]. RAP used AIMD due to this convergence to a fair state allowing it to be TCP friendly and well behaved.

RAP performs a loss-based rate control and does not use any explicit congestion signals (such as ECN [84, 118]) from the network. Instead it relies upon the packet loss experienced when competing for a network path with other network streams.

The result of the simulation of RAP shows that it behaves in a TCP friendly fashion over a wide range of scenarios. The results showed that using Random Early Detection (RED) [119] queue management results in fairness between TCP and RAP traffic.

3.3.3.2. Loss delay adjustment

The Loss-Delay based Adjustment algorithm (LDA) [120] is a sender based adaptation technique. It uses RTP and the accompanying reporting protocol RTCP for media transport and feedback information concerning packet loss and round trip times. An additional enhancement was made to RTP so that it is able to estimate the bottleneck bandwidth. Based on this information the sender increases or decreases the transmission rate. With periods of zero loss the sender increases its transmission rate by an Additive Increase Rate (AIR) which is estimated using loss, delay and bottleneck bandwidth values reported in the feedback reports.

Their algorithm is divided into three parts which handle the feedback information, the mechanism for setting the adaptation parameters and the actual adaptation mechanisms.

For each incoming stream the receiver sends a report indicating the fraction of data loss, the timestamp of the last received sender report (t_{LSR}) for that stream and the time elapsed in between receiving the last sender report and sending the receiver report (t_{DLSR}). Knowing the arrival time (t) of the RTCP packet the sender can calculate the round trip time (T) as follows

$$T = t - t_{DLSR} - t_{LSR}$$

Equation 2 Round Trip Time Calculation, reproduced from [106]

The calculation does not require the synchronisation between the clocks of the sender and the receiver.

RTP was enhanced with the ability to estimate the bandwidth of the limiting network connection within the path (known as the bottleneck bandwidth, i.e. the bandwidth at the bottleneck) using the packet pair approach described in [54]. The concept used in this approach is that if two packets can be caused to travel together such that they are queued as a pair at the bottleneck, with no packets intervening between them then the inter-packet spacing will be proportional to the time required for the bottleneck router to process the second packet.

To calculate the bottleneck bandwidths an application defined section is added to RTCP packets, including the source sequence number, the sequence number (SEQ) of a data packet that will start a stream of probe packets and the number (n) of probe packets that will be sent. Then, n data packets starting with packet numbered SEQ are sent at the access speed of the end system. At the receiver site, the bottleneck bandwidth is calculated as follows

$$bandwidth = \frac{probe\_packet\_size}{gap\_between\_2\_probe\_packets}$$

Equation 3 Bandwidth estimation

The probe is performed using data packets so therefore additional bandwidth required for the probe is restricted to an increase in data stream size as well as the corresponding RTCP increase. This approach requires large packets which can suffer from jitter. Therefore video packets are ideal for this. Audio packets are smaller and more prone to user perceived quality of service issues than video.

3.3.4. Summary

Application Layer Framing has become the standard method of encapsulating data within multimedia streaming protocols.

There are a number of protocols which have their throughput behaviour modelled after TCP; most using an additive increase multiplicative decrease as a method in which to be TCP friendly. As previously discussed in section 2.4.5, TCP is unsuitable for real-time interactive multimedia for a number of reasons one of which is the steady state behaviour of TCP, where the throughput varies even for a stable maximum segment size, packet loss and round trip time. This makes any protocol which attempts to imitate this behaviour unsuitable since the throughput it uses will vary depending on when packet are loss rather than the overall loss rate on the path. Equation based systems aim to solve this problem by using a mathematical model of TCP's behaviour, this can create a smoothed bit-rate trace but over time the bit-rate will track the fair throughput for TCP. This means that the audio/visual quality of a videoconference can still fluctuate due to the background traffic but instead of sharp increases the changes will be slower, but still visible.

3.4. Current solutions

This section describes the current solutions for providing some kind of quality of service support for audio/videoconferencing systems.

3.4.1. Rate adaptation

Rate adaptation systems use feedback or bandwidth measurement techniques in order to change the bit-rate at which they are sending their data. There are three types of rate adaptation systems: receiver initiated, sender initiated and proxy initiated.

3.4.1.1. Receiver initiated adaptation

With this method of rate adaptation, for an example see [121], the receiver of the stream informs the sender of the stream that the stream bit-rate should be changed. The main benefit to this is scalability since the servers don't have to process feedback from many clients that are connected. This system is also quite simple since the only connection that needs to be monitored is the stream from the server to the receiver.

The disadvantage is that the receiver just starts the adaptation but the re-encoding of the media stream must still be performed elsewhere, either at the sender or at a proxy. There are two possible solutions to this problem

- Allow applications to choose from different versions of the original video stored at the proxy or sender. This is suitable for non interactive media where the data can be encoded ahead of time.

- The client may request for dynamic adaptation at the server/proxy. It may however take sometime for the server/proxy to decide which version to send to the receiver. This is suitable for interactive media where the stream is encoded dynamically.

3.4.1.1.1. Multiple versions

This system involves creating multiple versions of a given media stream. Then when a client wishes to view the stream the most appropriate stream is chosen given the end-to-end conditions.

This system is limited in terms of the adaptation to the network conditions that is possible. If a network path does not fit the requirements for one of the pre-encoded streams then the user does not get the best possible quality of service.

3.4.1.1.2. Multiple description coding

Multiple Description Coding (MDC) features the encoding of a video stream into two independent streams. In contrast to layered encoding, where the presence of the lower layers is necessary for the higher layers to be decoded the streams provided by MDC are independent and can be decoded separately but if both streams are received they can be decoded together in order to get an enhanced quality.

3.4.1.1.3. Layered encoding

With this system the media is split into several layers, each layer carries part of the stream and is dependant on the previous layer. The full quality media is composed of all the layers. This gives a number of quality settings starting at a low quality (using a single layer) with each additional layer providing extra information and therefore quality to the stream. The oft cited example [122, 123] of this is that of MPEG layered encoding. MPEG video is made up of three types of frame I, P and B. The base layer, which is the only layer needed to view the media, is made up of only I frames. Additional quality can be had by receiving the additional P frame layer and the B frame layer. The flexible representation of layered flows, and the changes of achieving easy prioritisation of the video layers, is particularly appealing for

network transport of video. Layered encoding can provide simple and effective solutions to a number of problems:

- Rate adaptation is performed by adding or dropping layers according to the experienced network conditions. The granularity of the bit-rate changes is fixed and is therefore rather inflexible. Alternatively, there is a scalable two-layer CODEC in which the basic information is encoded into a base layer. The second layer provides the enhancement information and can be encoded to use the remaining bandwidth. This is only optimal for unicast environments where the enhancement layer is tuned to a single network path. In a multicast environment there can be a diverse range of network capabilities meaning that the enhancement layer is only optimal for one network path.

- Layered video works well on networks which have some form of prioritisation and quality of service. This allows the base layer to have a high priority while the enhancements layers are given a lower priority.

- Unequal error protection can be employed to protect the lower layers, as the impact of packet loss on quality is more severe for the lower layers than for the upper layers.

Receiver Driven Layered Multicast (RLM) [124] was the first published scheme which described how layered encoding could be used for transmission and control of multimedia data. In RLM receivers use probing experiments to see if they are able to join a specific layer. The receiver can drop layers when congestion is detected, and therefore lowers the quality of service and when extra bandwidth is available the receiver joins extra layers and thusly improves the quality.

3.4.1.2. Sender initiated adaptation

In sender initiated adaptation, it is the sender within a session that is responsible for deciding when and how to adapt the media stream.

The advantage of the sender-initiated adaptation is that the content provider is in full control of the content quality and it can make decisions on how to adapt the stream. Sophisticated communication with the clients is not needed.

The disadvantage of the sender based approach is that it does not scale as well as the receiver based approach as the servers have to change the way they serve the content.

3.4.1.2.1. Bandwidth negotiation

The server-client system is able to make an estimation of the available capacity between the two ends prior to the transmission of video. After the negotiation phase, the server transmits a stream that closely matches the capacity characteristics of the path. This method is inflexible but simple, and it is extensively used in streaming media products.

3.4.1.2.2. Control of encoding parameters

This is a common way of adapting to changing bandwidth availability. A rate controller regulates the output bit-rate by appropriately selecting the value of the quantisation parameter by means of rate-distortion optimisation. Other parameters that can be used to increase or decrease the encoder's output rate include frame skip rate, the spatial resolution, and dropping of chrominance components.

3.4.1.3. Proxy based adaptation

One big benefit of the proxy-based adaptation [125, 126] is that it is totally transparent to the senders and receivers. Neither the sender nor the receiver has to change anything in how they, respectively, send and receive the media stream. Due to its transparency, the proxy approach allows incremental deployment which is a cost effective solution to scalability

problems. The disadvantage to this is that it requires deployment on the network between the client and the server in order to operate.

Most proxy-based solutions currently only deal with client heterogeneity. They concentrate on a few static adaptation methods. Second with a proxy based approach content providers have no control over how their content will appear to different clients. Another aspect is that the proxy has to track the bandwidth of both sides of the stream; the sender-to-proxy and the proxy-to-client.

3.4.2. Repair based systems

Repair based systems attempt to conceal packet loss caused by real packet loss or packets arriving too late to be decoded (jitter) by generating new packets to replace the ones which were lost. There are two main classifications of repair based systems, sender and receiver. These are described below.

3.4.2.1. Sender based repair

Sender based repair systems are dependent on the sender including extra information in the transmission stream in order to perform the packet loss repair. The implication for this is that bandwidth is used on this additional information, which could have been used for encoding the media stream.

Forward Error Correction (FEC) is a sender based error-resilience technique that relies on the transmission of redundant packets from which the content of lost packets can be recovered [127].

Media Independent FEC involves the application of block or algebraic code on a codeword of k data packets to generate additional n-k check packets to aid the correction of losses. Several block coding techniques are proposed, the most popular being parity coding and

Reed-Solomon coding. Parity coding is achieved by applying an exclusive-OR (XOR) operation across groups of packets to create parity packets. The Reed-Solomon encoding is based on the properties of polynomials, and has excellent correcting properties.

The advantages of media-independent FEC is that they do not rely on the content of the packets, it is quite simple to implement, and it is not computationally expensive. The drawbacks are increased decoder complexity and increased encoding delay, which affects the interactivity of the session.

In Media Specific FEC the basic idea is to transmit each unit of the media stream in multiple packets. Therefore if a packet is lost another packet containing the same data should be received. This approach has been advocated by Hardman et al [128] and Bolot et al [129] for use on the Mbone and was extensively simulated by Podolsky et al [130].

Interleaving [131] is a useful technique for reducing the effects of loss when used with smaller packet size and when end-to-end delay is unimportant. Data units are resequenced before transmission so that data units which were adjacent to each other are now separated by a guaranteed distance. Since packet loss in the Internet is '*bursty*' [132] an entire segment of the data stream is lost during packet loss. If the packets lost are from the same time in the media stream then it is difficult to reconstruct the data since it is from an extended time period. With interleaving when packet loss occurs the lost packets represent small amounts of data from throughout the media stream which are easier to reconstruct either mentally [133] or by using packet regeneration techniques [134]. This is due to the fact that it is easier to conceal smaller errors that it is to conceal larger ones.

The problem with interleaving is that the media stream packet must be re-ordered both at transmission and at the reception. This adds a large amount of delay to the media stream

which when used in an interactive application reduces the quality of service experienced by the users of the system.

A widely deployed reliable multicast scheme is Scalable Reliable Multicast (SRM) [69]. When a member of an SRM session detects packet loss it waits a random amount of time, determined by its distance from the original source of the lost data and then multicasts a repair request. The retransmission timer is calculated such that although a number of hosts may miss the same packet, the host closest to the point of failure is most likely to timeout first and issue the retransmission request. Other hosts which also see the loss, but receive the retransmission request message, suppress their own request for retransmission in order to avoid message implosion[10]. This is important since in a large Mbone session every packet is lost by at least one receiver [135].

SRM and other reliable multicast protocols are suitable for the reliable multicast of data objects since shared data objects typically require data integrity so reliable delivery of messages is important at the expense of message delay. Reliable multicast protocols are unsuitable for audio or videoconferencing due to the retransmission of lost data packets which adds an unacceptable delay to the media stream. This is the same reason why TCP does not lend itself to the transmission of interactive multimedia.

Retransmission is able to repair large amounts of packet loss but at the expense of increased delay leading to a loss of interactivity within a videoconference and thus requiring stricter floor control[11] policies in order to operate.

[10] Otherwise a packet loss near the sender would result in a large number of simultaneous retransmission requests. This could result in the situation where the sender is overwhelmed by hundreds or thousands of retransmission requests.

[11] The method used to ensure who is in control of a task. With a videoconference it controls who is able to speak and at what time.

3.4.2.2. Receiver based repair

This form of error recovery takes place at the receiver of the media stream. These techniques can be used when the sender based recovery techniques have failed or when the sender is unable to participate in sender based repair schemes, or when it would be uneconomical to do so, such as when packet loss and available bandwidth is low.

Receiver based repair (sometimes called error concealment) produce a replacement for any lost packets. These systems work quite well with audio data with low levels of loss (less than 15%) and for small packets (4 ms to 40 ms of audio data). When the losses approach the length of phonemes (between 80 ms to 100 ms [136]) these techniques break down since the entire phoneme was missed by the listener [137].

3.4.2.3. Insertion based repair schemes

With insertion based repair a simple fill-in packet is inserted into the media stream when packet loss is detected. The simplest case is splicing, where the data after the loss is played immediately after the data before the loss. This is essentially a zero-length insertion. As an alternative silence substitution inserts silence into the media stream for the same amount of time as represented by the lost packets. Better results are obtained by inserting noise or reinserting the previous packets.

This scheme is distinguishable from others by the fact that that it does not use the received signal in order to reconstruct the missing segments. The results obtained using this technique are generally poorer than other systems but they are computationally inexpensive.

Lost units can be concealed by splicing together the audio on either side of the loss; no gap is left due to a missing packet, but the timing of the stream is disrupted. This technique has been evaluated by Gruber and Strawczynski [138] and was shown to perform poorly with

packet losses above 3%. Splicing works best with low loss rates and short clipping lengths (4 ms to 16 ms).

The use of splicing can also interfere with the adaptive play-out buffer required in packet audio systems, because it makes a step reduction in the amount of data available to buffer. The adaptive play-out buffer is used to allow for the reordering of disordered packets and removal of network timing jitter, and poor performance of this buffer can adversely affect the quality of the entire system. This means that slicing together audio on either side of a lost unit is not an acceptable repair technique for interactive media.

With silence substitution, the gaps which are left by packet loss are filled with silence in order to maintain the timing relationship with the surrounding packets. This technique is only effective with short packet length (less than 4 ms) and low loss rates (less than 2%) [139] making it suitable for interleaved audio over low-loss paths.

The performance of silence substitution degrades rapidly as packet sizes increase, and quality is unacceptably bad for the 40 ms packet size which is in common use within network conferencing systems [128].

Silence substitution although not effective with common network conferencing systems is widely used since it is easy to implement.

Noise substitution works in the opposite way to silence substitution. Instead of inserting silence into the media stream background noise is inserted into the gap left by lost packets. Studies of human perception of interrupted speech [140] have shown that phonemic restoration occurs for speech repair when using noise substitution but not for silence substitution.

When compared to silence the use of noise substitution in the form of white noise has been shown to give both subjectively better sound [133] and improved intelligibility [140].

Repetition replaces lost packets with copies of packets that arrived immediately before the packet loss. This technique has a low computational overhead and is relatively simple to implement. The subjective quality can be improved by gradually fading repeated units. The GSM system advocates using repetition for the first 20 ms of packet loss with the same amplitude followed by fading the repeated signal to zero amplitude over the following 320 ms [141]. The use of repetition with fading is a good compromise between the other poorly performing insertion-based concealment techniques and the more complex interpolation-based and regeneration concealment methods.

3.4.2.4. Interpolation based repair schemes

This class of packet repair system use packets from both before and after the lost packets in order to produce a replacement for the loss. Interpolation based repair systems have the advantage that they can better produce a replacement for the missing packets when compared to insertion based systems. The drawback, however, is that they must have data from both before and after the loss in order to create a replacement, potentially increasing delay.

Waveform substitution uses audio from before and optionally after the loss to find a suitable signal to cover the loss. Goodman et al [134] studied the use of waveform substitution in packet voice systems. They examined both one and two-sided techniques that use templates to locate suitable pitch patterns either side of the loss. In the one-sided scheme the pattern is repeated across the gap, but with the two-sided scheme interpolation occurs. The two sided schemes generally perform better than the one sided and both perform better than either silence substitution or packet repetition.

Wasem et al [142] presented a refinement of waveform substitution by using a pitch detection algorithm either side of the loss. Losses during unvoiced speech segments are repaired using packet repetition and voiced losses using a repeated waveform of appropriate pitch length. This method was found to behave marginally better than waveform substitution.

Time scale modification allows the audio on either side of the loss to be stretched to fill the loss. Sanneck et al [143] present a scheme that finds overlapping vectors of pitch cycles on either side of the loss, offsets them to cover the loss and averages them where they overlap. Although computationally demanding, this technique appears to work better than either waveform substitution or pitch waveform replications.

3.4.2.5. Regeneration based repair schemes

This class of system uses knowledge of the CODEC in use in order to derive CODEC parameters, in order that lost audio can be synthesized. These systems perform well but are CODEC dependent and somewhat computationally expensive.

For CODECs which are based on transmission coding or linear prediction it is possible that the form decoder can interpolate between states. For example, a speech coder could interpolate the state of the linear predictor coefficient either side of short losses and use either a periodic excitation the same as the previous frame, or a gain matched random number generator, depending on whether the signal was voiced or unvoiced. For longer losses, the reproduced signal is gradually faded. The advantages of CODECs that can interpolate state rather than recording the audio on either side of the loss is that there are no boundary effects due to changing CODECs, and the computational load remains approximately constant. However is should be noted that CODECs where interpolation may be applied typically have high processing demands.

With this technique the speech on both sides of the loss is fitted to a model, after which the packets for the period of loss are regenerated based on the model. Chen et al [144] have done some work on interleaved μ-law encoded speech repair by combining the results of autoregressive analysis on the last received set of samples with an estimate of the excitation made for the loss period. This works well since the size of the interleaved blocks is short enough to ensure that the speech characteristics of the last received block have a high chance of being relevant.

This method could possibly introduce additional delay since packets from both sides of the packet loss are required to regenerate the missing portions.

3.4.3. Adapting to network conditions

Changes in the four main aspects of network stream: throughput, delay, jitter and packet loss require different adaptation mechanisms in order to cope with the changes. This section presents some of the methods which can be used to adapt to changes in the various network parameter measurements.

3.4.3.1. Adaptation to delay

Figure 8 shows delay in a loss free system. Unfortunately, nothing can be done in order to reduce the effect of network delay in a videoconferencing system, since the minimum time that we can send data across the network is defined by the delay. There is delay that comes from the capture/encoding and decoding/playback process. We are able to affect this delay by choosing CODECs which are not computationally intensive as well as other methods such as using small buffers.

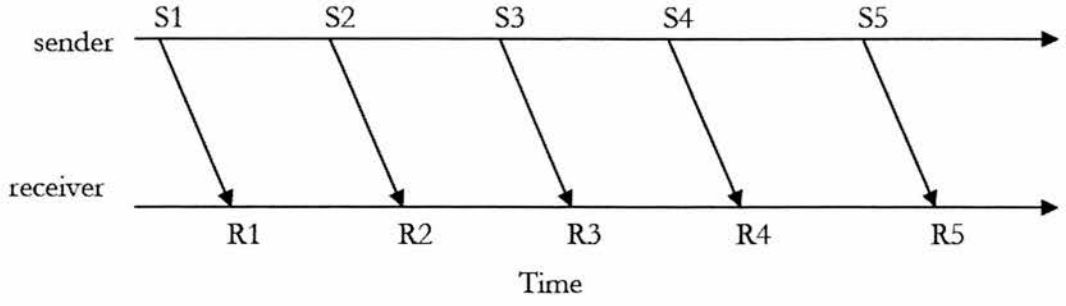


Figure 8 Delay in a loss free stream, reproduced from [145]

3.4.3.2. Adaptation to jitter

Jitter, is defined as the difference of the inter-packet arrival time. Figure 9 shows that the time between the arrivals of successive packets differs.

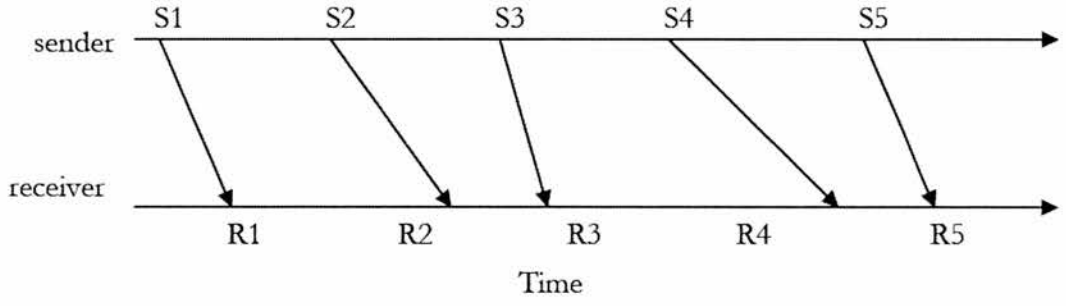


Figure 9 Jitter in a loss free stream, reproduced from [145]

The variation of the arrival time of the packets leads to problems in the playback of a media stream. Claypool and Tanner [145] found that '*the presence of even low amounts of jitter or packet loss results in severe degradation in perceptual quality, while higher amounts of jitter and packet loss do not degrade perceptual quality a proportional amount*'. This can be smoothed out using a play-out buffer. The type of application is crucial in the choice of the size of the play-out buffer. An interactive application cannot afford a large buffer and in fact the smaller the buffer the better in order to maintain the interactive nature of the audio and video without leaving jitter in the stream.

Non interactive applications such as Video-On-Demand (VOD) can have a large play-out buffer of several seconds or more.

There are two types of play-out buffer which can be used.

- Fixed – In this case, each data unit is delayed for a fixed time-interval T after it was generated. If a data unit (packet) arrives T seconds after it was generated it is discarded. The value of T is a trade-off between resistance to jitter and interactivity.

- Adaptation – In this method, the play-out point of each data unit is adaptively changed according to the experienced delay. The changes of the play-out points are performed at media suitable points. In the case of a voice media stream a media suitable point would be at the beginning of a talkspurt[12].

3.4.3.3. Adaptation to packet loss

Packet loss in the modern Internet is unavoidable and therefore streaming media applications must be tolerant to this form of data loss.

In order to increase compression efficiency, encoding algorithms try to remove redundancy in the signal by exploiting spatial and temporal redundancies. This has the effect that packet loss is more noticeable in the decoded media stream.

3.4.4. Summary

Internet packet loss rates are typically below 15% which means that in most situations a receiver based repair system will perform better than a sender based system. Receiver systems work best with packet sizes between 4 and 40 ms. Silence substitution can be used for packet losses lower than 2% as long as the size of the packet is small (4 ms or less). In general, noise

[12] A talkspurt is defined as an interval of voice activity between two periods of silence.

substitution will perform better than silence substitution but at the cost of additional processing. Splicing operates best below 3% packet loss with packets containing between 4 and 16 ms of audio data. Above 15% packet loss, redundant information needs to be encoded into the media stream. This is where sender based systems work best. Finally retransmission can operate in situations where all other systems fail, but at the expense of delay and throughput which can reduce the usefulness of an interactive session, requiring stricter floor control policies in order for the conference to work effectively.

3.5. Related systems

There are three forms of network aware system. These are active monitoring, passive monitoring and hybrid.

Passive monitoring systems do not inject any new data into the network and therefore rely only on the data which is being placed onto the network by the applications. This class of network monitoring system has the advantage that it does not 'waste' bandwidth on measurements so the maximum amount of network is available to the applications. The drawback is that the measurements cannot be tailored and may not be an accurate reflection of the network paths in question, they are just a reflection of the use that the application is making of the path. An advantage is that you do not require an end point to take part in the measurement as the application end point is used.

Active monitoring techniques do inject traffic into the network and measure how it behaves. They assume that the probe traffic will behave in the same manner to the application traffic once it has been placed on the network. The advantage to this form of system is that very accurate measurements can be made since they do not require any inferences or estimates to be made concerning the network traffic. The downside of course is the additional network

traffic which is generated that uses bandwidth which otherwise would have been allocated to the application.

Hybrid is a combination of the two approaches. It uses passive techniques when there is enough application traffic but switches to active when the application traffic has dropped off. Thusly this approach should combine the best of each extreme, but may have the disadvantage that the measurements made during the active phase may not reflect the traffic in the passive phase.

3.5.1. Remos

Remos [146] was designed to provide resource information to distributed applications. The Remos architecture divides its services between *Collectors*, *Modelers* and *Predictors*.

Collectors are responsible for acquisition and consolidation of any and all information needed by the application. Collectors may use a range of methods for collecting the information. Collectors can be organised into a hierarchical fashion for the purpose of scalability. At the lowest level a Collector is responsible for collecting network information for the LAN on which it is connected. No state is stored at the *collectors* and they only report what they see.

Modelers sit between the application and the *collectors*. They are responsible for processing the information concerning the network which has been gathered by the *collectors* into a form which is of interest to an application. For example the *collector* may have reported several available bandwidth measurements but the application is only interested in the lowest. It is the job of the *modeler* to provide processing in order to supply this measurement. An application communicates with a single *modeler* which coexists on the same node as the application, whereas the *modeler* gathers information from many *collectors* which are spread all around the network.

Predictors are the final part of the architecture. They are responsible for turning measurement histories into predictions of future behaviour. Remos uses the predictors which were developed as part of the RPS Toolkit [147]. *Predictors* in Remos have API support for the prediction of network behaviour but only the predictors for host load have been implemented.

3.5.2. Shared passive network performance discovery

Shared Passive Network Performance Discovery (SPAND) [148] passively collects network performance information from a collection of hosts. This information is cached and shared which allows a group of hosts to gain information about network performance in a way which does not introduce additional traffic.

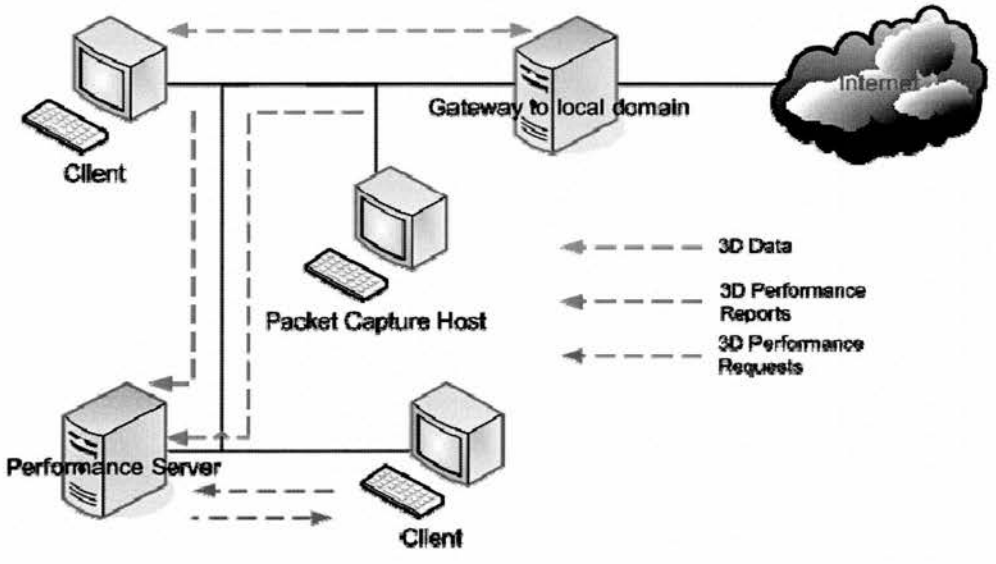


Figure 10 SPAND Architecture, reproduced from [148]

SPAND consists of *Clients*, *Performance Servers* and *Packet Capture Hosts*. A Client has a modified network stack which is able to transmit performance reports to the Performance Servers. The performance reports include source and destination tuples, a timestamp, the size and duration of the sample as well as the number of packets received and lost. The report

also includes the transport protocol used and the application hint, which serves as a hint as to how the application was using the connection.

3.5.3. Wren system

Zangrilli and Lowekamp [149] are developing the Wren system. They employ a hybrid monitoring approach using passive tracing of existing applications, but actively injecting traffic into the network when needed to maintain the flow of bandwidth measurements.

The primary requirements which they set for The Wren System are

- No modification to the application code or the network infrastructure
- Application performance must not suffer. The monitoring software should not compete with the application for network or CPU resources

Due to these requirements the Wren System has been developed utilising a kernel-level packet trace facility. The benefits claimed for this kernel-level approach are

- Traffic can be captured without modifying the application
- The amount of additional overhead can be minimized
- Packets can be accurately time-stamped with nanosecond precision
- Kernel-level protocol information such as TCP's window size can be directly recorded.

The use of a kernel level system reduces the ability to widely deploy such a system since specialised host computers must be set up in order to use it.

Currently the Wren system is for use within unicast environments but Zangrilli and Lowekamp have started to look at multicast traffic and network measurements. They have

identified RTP and RTCP as the transport and reporting mechanisms to be used. They go as far as using RTCP as the protocol for their measurements. RTCP has well known problems in scaling to large groups [150] although there have been potential solutions, for example [151]. So for the Wren system in the multicast environment the system will not scale to large amounts of traffic and still keep accurate measurements.

3.6. Summary

There are two basic approaches presented to the issues of networked quality of service; resource reservation where resources along the network to be used are reserved for use by the system and aggregated flows where network traffic is classified into one of various traffic classes so that it can be treated appropriately. Finally Less than Best Effort was presented where traffic is marked as unimportant and dropped if problems are being experienced on the network. Each of these approaches requires additional network infrastructure to be deployed in order to support them and so are difficult to reach a large deployment.

SPAND and the Wren system are examples of network monitoring and prediction systems. SPAND adds additional information into the network while Wren uses a hybrid monitoring techniques in order to obtain network information. Neither of these systems operate within the multicast environment which is required for the effective use of videoconferencing.

The quality of service systems reviewed all present the problem of requiring QoS from lower layers. SPAND and Wren have shown that it is possible, at least in the case of TCP to use network monitoring and prediction to improve an application's knowledge of the network conditions. The presented networked quality of service frames require lower levels of quality of service support from the network infrastructure which as argued in [152] is unlikely to be provided in the short to medium term. This makes them unlikely to be deployable on a wide scale on the Internet, since support for the underlying network QoS is still lacking.

Methods of adapting media streams to levels of quality of service which a network provides were presented. This leads to the proposition that coupling a network prediction framework with detailed knowledge of the application can lead to an increase in the user perceived quality of service. In the case of videoconferencing attempting to maintain a consistent quality has been shown to be beneficial to the user perceived quality of service.

Chapter 4

Conference Controller Architecture

4. Conference controller architecture

4.1. Introduction

The main problem with Internet based videoconferencing relates to quality of service. In order to increase the quality of videoconferencing a system is needed which takes into account the characteristics of Internet traffic. These assumptions are as follows:

1. No largely deployed network level quality of service mechanisms
    a. No resource reservation
    b. No admission control
    c. No constrained delay times
    d. Variations in delay and bandwidth
2. Best effort delivery
    a. Unreliable
    b. No guarantee of packet ordering
    c. Variations in packet loss

These assumptions are for traffic at the IP level. Protocols, such as TCP, which build on IP to provide other services (such as reliability) are not included here since packet retransmission is typically the means of reliable transfer. As discussed previously packet retransmission introduces too much delay for use in an interactive videoconferencing situation.

Since we cannot reserve bandwidth, we have to work with the bandwidth which is available within the best-effort model that the Internet provides. It is known that there are strong temporal patterns of bandwidth usage[13] over network paths. Coupling this with the requirement for consistent quality means that a videoconference session should be configured to make consistent use of available network resources. However, simply tracking and constantly adapting to available bandwidth [153] can easily lead to large variations in the quality of service provided to the users during the lifetime of a conference. This variation in quality is what Bouch et al and Hands et al found to be detrimental when interacting with media streams.

As part of the need for consistency, expertise in selecting the encoding and decoding parameters for a conference session is required. So, a further goal for the system is to provide this expertise, so that users of multimedia conferencing do not need to be proficient in the configuration of the technologies that allow them to meet in a virtual space.

The Conference Controller Architecture (CCA) was developed to tackle these issues. It is a framework for providing an appropriate and consistent quality of service within an Internet based multimedia conference. The framework employs monitoring, maintenance and prediction to ensure consistency. In addition, the CCA can take the decision on how best to configure a conferencing session.

The information presented in chapter 2 on the psychological and the network requirements of audio and video gives information on the encoding of audio and video for the transportation across networks, including recommendations on which CODECs, bit-rates and packet recovery mechanism work best with given conditions on a network. This

[13] See section 2.4 on page 7 for details of those patterns.

information is used in order for the CCA to provide the session configuration information. So based on the bit-rate that it believes should be used for the conference in order to maintain quality and consistency the CCA allocates a proportion to audio and the rest to video. The jitter estimations inform the decision on play-out buffering and this coupled with the information on delay is used to inform the decision for the encoding CODEC and settings to be used.

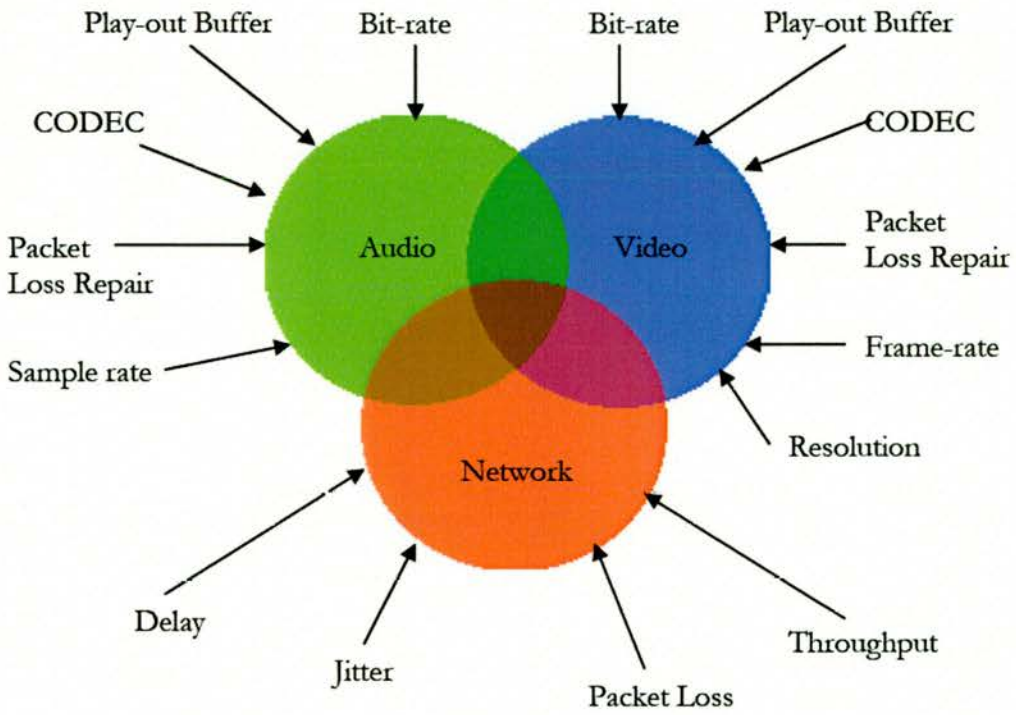


Figure 11 CCA Approach to maintaining videoconferencing quality

The aspects discussed to date (the physiology and the network monitoring) are combined in Figure 11. The measurements from the network give the CCA the constraints which it has to work within. Knowledge of audio and video perception in different situations is then used to shape the videoconference parameters in order to gain the best quality for the duration of the videoconference.

4.2. Overview

An overview of the Conference Controller Architecture is presented in Figure 12. The CCA consists of three entities; the Traffic Data Repository (TDR), the Conference Controller (CC) and the Participant Agent (PA). The TDR and CC reside on the server and the PA resides on the end user's computer. The Traffic Data Repository contains a database of the traffic reports from the network. The Conference Controller uses the Traffic Data Repository to store and retrieve the information concerning past network conditions which it can use to configure and maintain sessions. The Participant Agent is the end user component. It provides the video conference monitoring as well as the audio or videoconferencing capabilities. The Participant Agents can either be implemented from scratch as was done with the testing in this project or an existing videoconferencing application could be taken and the conference monitoring and configuration aspects of the PA added.

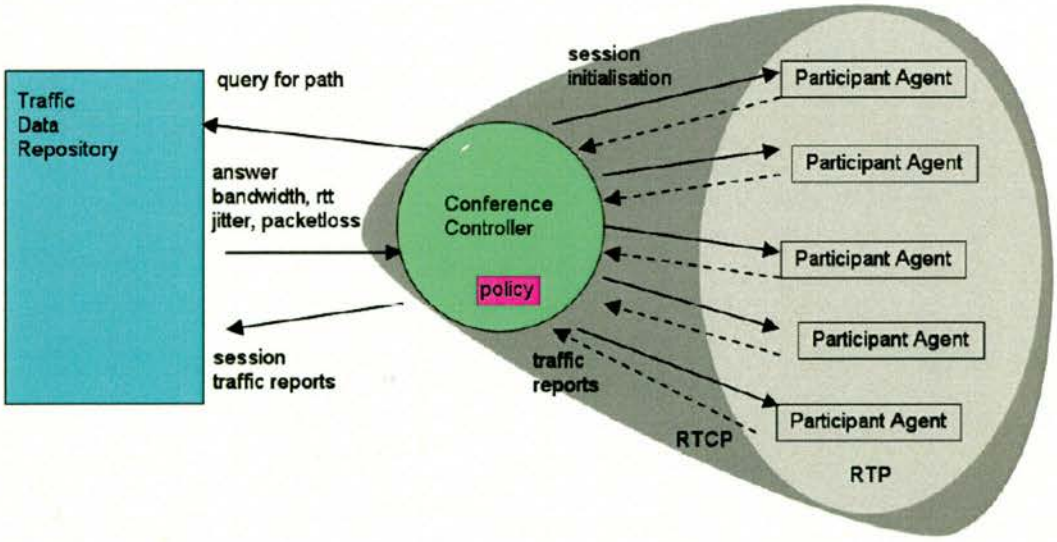


Figure 12 The Conference Controller Architecture

The Conference Controller Architecture operates in the following manner. First; a videoconference is scheduled for a specific time and duration with a number of participants. Each participant has an associated IP address or IP block. The Conference Controller

receives this information and contacts the Traffic Data Repository and requests information concerning all the network paths which will be involved in the videoconference. At this stage the network paths are expressed as a start and end point (source and destination IP addresses) but are stored in the database as a full network path giving each hop along the route. This means that a change in route can be detected even if the source and destination points have not changed. Each measurement record links to a route (source to destination). The routes are measured at the start of a conference in a manner similar to the *traceroute* UNIX command, with multiple pings with the modulation of the IP TTL field. At the start of a conference the network route is measured and network traffic measurements are retrieved from the Traffic Data Repository which has taken the same route between the two end points. This means that local networks can share the same network traffic information.

The Traffic Data Repository returns information concerning the network paths which are expected to be involved in the session. The Conference Controller summarises the information and passes the summary, as well other information (participant list, full path information, number of participants and others) to a *policy*. The *policy* to be used can be chosen in a number of ways. Firstly the videoconference client could provide the policy. This has advantages in that the policy can be tailored to the application. The second method uses policies which are stored in a database and accessible by the Conference Controller. This was the chosen method employed in all the development, testing and evaluation since it meant that the client applications (Participant Agents) can be kept simple and only have to deal with the audio and video. It also means that it is much easier to keep several different policies 'live' in the system.

The policy has two distinct stages. Firstly there is the initial conference configuration. This uses the data from the Traffic Data Repository which has been passed into the Conference

Controller. From this data, predictions of the throughput, delay, packet loss and jitter which are likely to be experienced over the network paths are made. These predictions are used to configure the settings for the conference. The policy provides application specific knowledge. In the case of videoconferencing it encompasses knowledge on choices of CODEC, bit-rates, trade-offs between encoding parameters and various other attributes which are discussed in Chapter 2.

The configuration for a conference can be performed ahead of time. For example if a conference is known to start at 2 pm the initial configuration process could take place at 1:55 pm. There is a limit to how far ahead a conference could be configured. Sensibly, it should be configured after the last conference which could be taken into account has occurred. In practice it is hard to predict the time of the last conference which will be important. It could be that either a weekly or daily pattern is present in the network data. There is no way of knowing which without processing the policy. Since this is the case, it makes sense to only perform ahead of time configurations up to 24 hours minus the length of the conference ahead of time, for example if the conference is an hour long this would mean only up to 23 hours before the start of the conference. The conference parameters which have been decided are saved until a Participant Agent requests the configuration for the conference. The number of participants which are expected to take part in a conference is made available to the policy. This combined with the predictions of available bandwidth informs the choice of floor control mechanisms to be used; for example 'turn taking' or have every client transmitting audio and video at the same time.

The videoconference can now start, assuming that there is more than one participant. The Participant Agent starts sending multimedia in the form of RTP encapsulated audio and video data. The Participant Agents make measurements of the network traffic. These

measurements are filtered and summarised and sent to the Conference Controller. The sending rate of the measurements is variable; the minimum used is 1% of the conference throughput with additional measurement being sent to the CC when the received streams do not conform to the set parameters. The Conference Controller feeds these measurements into two places. After the initial configuration has been performed the policy moves into conference maintenance mode. The incoming traffic measurements from the Participant Agents are used to monitor the behaviour of the conference and the network. During the monitoring phase if a problem is detected, such as a much higher packet loss than was expected, a new configuration can be sent out to the Participant Agents. The measurements from the Participant Agents are also sent to the Traffic Data Repository so that they can be used to aid predictions of network resource availability for future videoconferencing sessions.

The next three sections describe the Traffic Data Repository, the Conference Controller and the Participant Agent in detail. Policies are described within the context of the Conference Controller.

4.3. Traffic data repository

The Traffic Data Repository (TDR) is the simplest of the modules within the system. It is used by the Conference Controller to store the traffic reports. The Traffic Data Repository consists of two parts. The long term storage is provided by (but not tied to) MySQL [154]. The Traffic Data Repository is a passive component, only responding to requests.

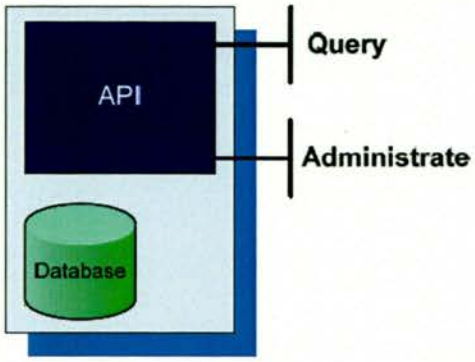


Figure 13 The TDR Architecture

There are six tables, which are used to store the information about each network path - jitter, delay, throughput, packet loss, packet size and path.

All measurement tables contain the following attributes as well as the measurement specific attributes which will be described individually.

- Source IP address – The sender of the packet stream
- Destination IP address – The receiver of the packet stream
- Path ID – This indicates the complete path
- Start timestamp – The start timestamp for the measurement in standard UNIX format
- End timestamp – The end timestamp for the measurement in standard UNIX format
- Number of measurements – The number of measurements which have been summarised here.

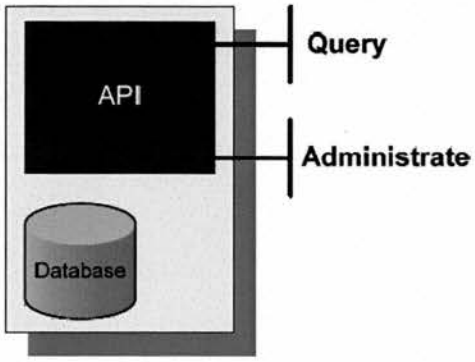


Figure 13 The TDR Architecture

There are six tables, which are used to store the information about each network path - jitter, delay, throughput, packet loss, packet size and path.

All measurement tables contain the following attributes as well as the measurement specific attributes which will be described individually.

- Source IP address – The sender of the packet stream
- Destination IP address – The receiver of the packet stream
- Path ID – This indicates the complete path
- Start timestamp – The start timestamp for the measurement in standard UNIX format
- End timestamp – The end timestamp for the measurement in standard UNIX format
- Number of measurements – The number of measurements which have been summarised here.

Firstly the throughput table contains the bit-rate measurements. Each bit-rate measurement is stored as an integer and is in bits per second.

The Packet loss table, as well as the standard attributes contains a packet loss measurement. This is stored as a fraction of one hundred within an eight bit integer field. So a value of 127 would mean 50% and the resolution of the measurement is 0.39%.

The Delay table contains the delay measurements. They are stored as an integer and are in milliseconds.

The Jitter table contains a jitter measurement which is stored as an integer and expressed in milliseconds. The jitter measurement is the maximum inter packet delay which has been seen on the path under measurement.

The packet size table contains measurements of the packet sizes. These are average packet sizes stored as a 16 bit integer. This gives a maximum packet size of 65,535 bytes. It is technically possible to have IP packets larger than 65,535 bytes (using IPv6 jumbograms [155]) but it would be unwise to use them within a videoconference because of the delay that would be introduce. A packet of this size would act as an additional buffer which would increase the delay experienced within the videoconference. For this reason and the fact that IPv4 does not support jumbograms the maximum size of 65,535 was chosen.

Finally the path table stores complete path information from the source to the destination. It provides detailed path information; giving hop by hop path information. The fields are

- IP Address – The IP address of the router at this hop.

- Hop number – This is the hop count of the router. Zero is used for the first hop while 255 is used for the final hop, with one being used for the first hop, two being using for the second and so on. This allows searches to be made for paths which start and end with given IP addresses.

The Traffic Data Repository provides two interfaces to the data; the first is an administration interface that allows the alteration of the information stored within the database. This interface allows for the addition, removal and inspection of data. The administration interface is provided so that the data within the Traffic Data Repository can be 'cleaned'. For example older records can be removed or generalised by merging records. Cleaning is performed by the removal of the majority of records which indicate that the multimedia session was operating satisfactorily and that there were no problems. Most of these records add nothing to the available information since they are just repeating what has already been seen. A small number of records are kept which show the session operating normally so that future sessions can use them for reference. The cleaning of the database is performed once a week. Data which is older than six weeks is not used when configuring a conference so it is summarised. The summary process takes the highest, lowest and mean measurements for a time period and removes the rest of the measurements.

The second interface is the commonly used interface. It is used for the querying and addition of measurement records by the Conference Controller.

4.4. Conference controller

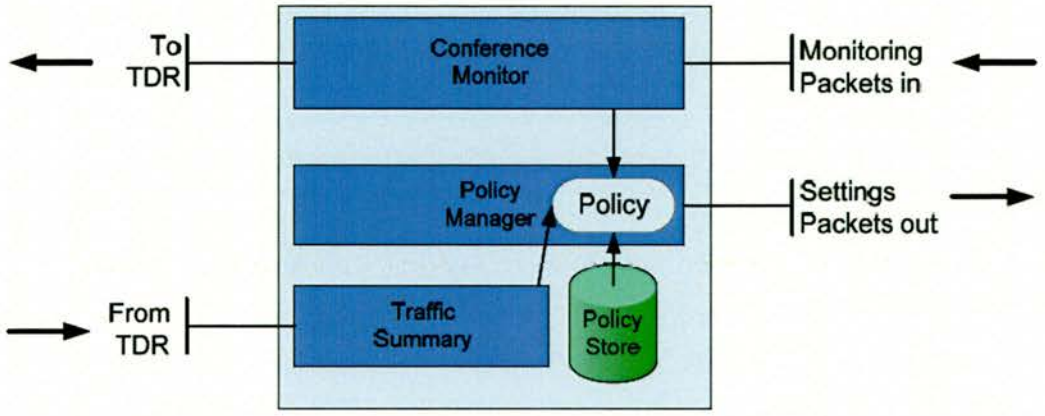


Figure 14 Conference Controller

The Conference Controller (CC) has two phases of operation

- *Conference initial configuration* – This is when information from past sessions is examined and used to provide predictions on future network conditions. From this prediction a suggestion as to the configuration of the session parameters is made. These parameters include choice of audio/video CODEC, methods of packet loss repair and so on. Section 4.4.2 on page 90 details the operation of the CC when it is in this phase.

- *Conference monitor and maintenance* – After the session has started the CC enters this phase. It receives feedback which takes the form of monitoring packets from the Participant Agents and uses this to revise the session parameters if needed as well as storing the measurements in the TDR for use by future sessions. This uses the Conference Controller, Participant Agents and the policy in maintenance mode. Section 4.4.3 on page 94 details the operation of the CC when it is in this phase.

All the configuration and maintenance of sessions is performed by a policy. This section goes onto describe the policies, and the two phases of operation that the CC can be in.

4.4.1. Policies

Policies play a central role within the system; they control how the videoconference session parameters will be initially configured as well as maintained throughout the videoconference. The policies can be stored at the Participant Agent and uploaded to the CC when the conference is to start or they can be stored by the policy manager. Each approach has its advantages. The policy stored at the application means that the application has full control over how it will be controlled; giving the application writer control of their own traffic usage. Storing the policy in a central repository means that the policies can be shared by multiple videoconferences. The CCA provides both methods but all testing and results were gathered using the latter method.

The policy has access to the information from the database for a given set of network paths. A network path is a source and destination IP address, for example 172.16.32.2 and 172.16.33.2. This pair of IP addresses specifies the route in one direction. The information for the IP address pair of 172.16.33.2 and 172.16.32.2 is needed in order to obtain information about the return path. It is not safe to assume that the forward and return paths are symmetrical as *'hot potato'* [156] routing (where an Internet Service Provider (ISP) attempts to route a packet out of its network as soon as possible in order to reduce costs) would almost certainly ensure this is not the case.

Policies are stored within a table in a MySQL [154] hosted database. The table has two fields which are the name of the policy (used to identify the type of policy) and the policy itself. The name is a string, while the policy is a binary large object (BLOB). The policies are implemented in JavaScript (specifically using the Mozilla Rhino engine [157, 158] as the interpreter) and are included using IBM's BeanScripting framework (which since moved to

Jakarta [159]) allowing policies to be written in a number of different languages, such as Python or Perl.

Policies have the following structure with the example given in JavaScript

```
function AVInfo initialise (Settings, Measurements) {}
function trafficReports (startDate, endData, srcIP, destIP, bandwidth, variance,
delay, dropped, packetSize, total, jitter) {}
```

Figure 15 Generic policy structure

The first function to be called is *initialise ()* and takes an instance of the *Settings* and *Measurements* objects, which are passed in as part of the environment. The *Settings* object contains information relating directly to the conference about to be configured. It contains the expected number of participants of the session, the expected number of people who will be vocal within the session (for a tutorial this would be one, the tutor) the scheduled start time and end time, the name of the conference, and a list of participating IP addresses. The *Measurements* object contains all the measurements (jitter, delay, throughput, packet size and packet loss) from the Traffic Data Repository relating to the paths which are to take part in this video conference. An example of an initialise function is shown in Figure 16. The line numbers do not appear in the configuration script but aid the explanation. The example policy script only addresses the throughput of the network paths involved and not the delay, jitter or packet loss. The initialise function shown configures a video-only conference using the H.261 CODEC and the mean bit-rate from the previous sessions as the sending bit-rate. The configuration information is stored in an *AVInfo* object. This object has a number of attributes for video and audio configuration. Configuration for CODECs are given, for video, by setting the videoCodec attributes. Similarly the bit-rate is set with the videoBitRate attribute.

```
function AVInfo initialise (settings, measurements) {
1)  totalBandwidth = total (measurements.bandwidth);
2) noBMeasurements = size (measurements.bandwidth);

3) avinfo.videoOn = true;
4) avinfo.videoCodec = H261;
5) var bw = (totalBandwidth / noBMeasurements);
6) avinfo.videoBitRate = bw;
7) return (avinfo);
}
```

Figure 16 Example initialise function within policy

When the policy script starts, it has a number of different pieces of information available from the environment. The main information that will be used within the script is the network measurements from the Traffic Data Repository. The measurements are available to a script writer within the *Measurements* object. This object is part of the global namespace and is always available and cannot be modified. Summaries of network measurements are generated by the Conference Controller and included in the *Measurements* object. These are estimates of the network conditions based on weekly and daily analysis. Along with each of these estimates there is a confidence rating which reflects how similar the data from the previous weeks or days were. Finally there is a best guess, which is the estimated network conditions from the weekly and daily dataset which has the highest confidence rating. A policy author can, if they wish, just take the best guess offered by the Conference Controller through they are free to analyse the data themselves.

The *Measurements* object contains five measurements, with one namespace for each type of measurement. These are;

- *Bandwidth*
- *Packet loss*
- *Delay*

- *Jitter*
- *Packet size*

Each of these namespaces is an element of an array. The data stored at each element being the actual measurement for the network connection. For example, using measurements.bandwidth[0].bandwidth within the Javascript policy would refer to the first bandwidth measurement returned from the traffic data repository.

There are also the three summary namespaces; *weekly*, *daily* and *best*, where *best* is a link to the closest fit, whether that is the daily or weekly summary.

Each of the bandwidth, packet loss, delay and jitter arrays contain different object types each of which reflecting the type of information that is required for the type of information being stored. The objects reflect the database table which stores the data relating to them in that the fields name are the same; so that the bandwidth object shares its fields with the bandwidth table from within the Traffic Data Repository.

Detailed descriptions and rationales of the collection and storage of network path measurements are given in section 4.7 Collecting network path on page 103.

The second global object is the *Settings* object. This provides information about the conference:

- The number of people that are expected to be attending the conference
- The number of people that are deemed active participants and the number of passive participants
- The IP addresses of the hosts that are involved
- The scheduled start time of the conference

- The scheduled end time of the conference
- The name of the conference

This information is provided so that the policy can make use of it during the configuration of a session. For example a policy may configure the session so that the active participants within the session are allocated more bandwidth and thus a higher quality since they will be the primary persons transmitting during a session.

'Active participant' refers a participant who is expected to contribute to the video conference, i.e. a participant that speaks. Active participants will typically be tutors or lecturers (in tutorials or lectures) or in the case of a meeting, everyone. Of course this may not be the case and it is up to the policy to choose when to allocate the active participants extra resources and when to treat them the same as everyone else within the session.

Now that the background knowledge of how the policies are implemented has been presented, the process of configuring, monitoring and maintaining a session is described.

4.4.2. Conference initial configuration

The main part of the policy is dedicated to the initial configuration of the multimedia conference. If this part of the configuration process is accurate then the goal of not having to change the configuration will have been achieved; there will be no need to change the session parameters during the course of the session. If the initial settings chosen have a low quality and the network is able to support a higher quality then an increase in quality should go unnoticed. If the starting point is higher than the network is able to provide over the course of the conference then the quality will have to be reduced. This is more likely to be noticed as drops in quality are perceptually more noticeable than increasing quality. When a network path is being used for the first time within the scope of the CCA there will not be enough

information known about it to make an informed decision on the QoS that it is likely to be provided. There are three potential ways to deal with this:

1. Accept that nothing is known about the paths and wait until the conference starts to gather traffic information and learn about the network paths.

2. Perform a quick bandwidth measurement of the path and use this as the basis of the configuration. This would take the form of sending a small number of probe packets through the network in order to determine the available bandwidth. This would also provide an end-to-end delay measurement.

3. Performing a long profiling process. This would take the form of sending a large number of probe packets between the end points within the conference. The time at the beginning of the conference where people are waiting for others to join could be used for this purpose. The problem with this is that when the final person joins the conference instead of being able to start the conference there would be a pause while their network link was profiled.

The first method has no information about the paths involved and therefore it would seem sensible to start the traffic usage at a low level and build up throughout the lifetime of the conference. The problem here is that we are trying to maintain a perceptual quality and if we start low (for example with textual chatting) and build up the user will have to go through text chatting, audio chatting and on to a full videoconference. This is not a desirable situation due to the changing qualities and conferencing techniques which would only serve to confuse the user of the system.

The second technique, performing a bandwidth measurement helps to remove some of the problems but not all of them. It provides an estimate of the bottleneck bandwidth but only at

the moment that the measurement was made. Therefore it cannot be relied upon to be accurate for the length of the conference. The bit-rates of a video conference are only one aspect of a session configuration and without knowledge of packet loss, delay and jitter a session configuration can not be optimised for the link since worst case scenarios will have to be chosen for delay, packet loss and jitter. This will mean low quality conferencing.

The third choice, having an extended measurement period provides the best information as the number of measurements samples is much larger. A packet generator sends traffic from the sender to the receiver over the link that is to be tested. Each packet has a time stamp and sequence number. When the packet is received it is sent back to the sender. This allows direct measurement of round trip time which can be used to estimate a delay, bottleneck bandwidth, packet loss and jitter. This testing period could last anywhere from several seconds to several minutes. It is performed while participants are in a session but are still waiting for other people to join, for example when a group of six is to take part and only four have as of yet connected. When everyone has connected, the conference can start. This does mean that the longer a participant is waiting for other members of the group the more knowledge will be gathered about that individual's network link. If everyone joins at the same time then the testing period would be very short, in the order of seconds.

An alternative is to have a brief detection phase at the start of the conference when all the participants have joined. This would mean that network path statistics are known about all paths. The chosen method is to gather path information while participants are waiting for others and when everyone has joined have a brief network detection phase before the videoconference starts. (This would only be needed when a path is completely unknown).

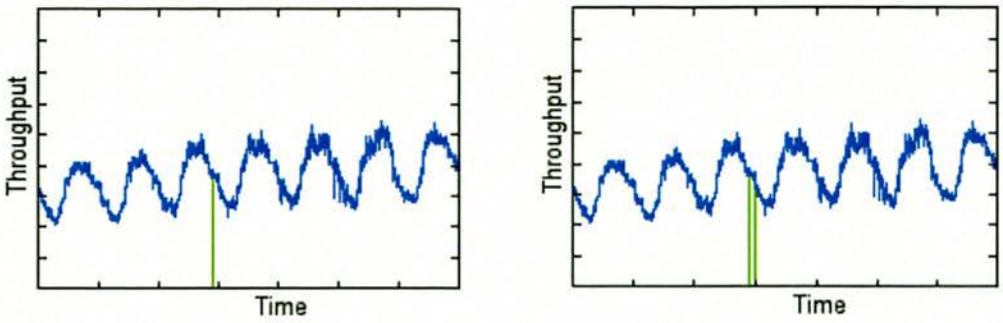


Figure 17 Instant testing (left) vs extended testing (right)

Figure 17 shows the difference between a single measurement and a longer period of measurements. The single measurement only provides information for a single point in time which may or not be representative of available bandwidth on the network path. A longer period of measurement, although providing more information, will still be limited to information which can be discovered from the measurement period. The longer period of testing will be able to provide information concerning the packet loss on the path but again this information cannot be guaranteed to be accurate for the lifetime of the conference. In order to provide the most robust quality a packet repair scheme which can operate on higher level of packet loss is required.

If an end-point is within a network segment that the traffic data repository has knowledge about then we can use that knowledge. Since IP addresses are allocated in blocks we can use information from IP addresses from the same network block as an IP address for which there are no network statistics known. This information should not be trusted to the same level as the direct measurement but can be used as a guide towards what might happen and allow some estimates of network conditions. The advent of wireless network technologies presents a potential problem. The end point network quality may vary widely, particularly as packet loss can vary widely.

4.4.3. Conference traffic monitor

Conference traffic monitoring is performed at the Participant Agent and the results are fed back to the Conference Controller and into the Policy Script. The conference traffic monitoring information is entered into the policy asynchronously when each CCA traffic report (described in section 4.6) arrives. The name of the function that is executed from the policy when a new traffic report arrives is *trafficReport* and it has the following parameters

- **Date** startTime – The start time stamp of this measurement. Times are accurate to the millisecond.
- **Date** endTime – The end time stamp of this measurement. Times are accurate to the millisecond.
- **String** srcIP – The IP address of the sender of the media stream.
- **String** destIP – The IP address of the receiver of the media stream.
- **Integer** bandwidth – The current bandwidth of the media stream being received from the sender in bits per second.
- **Integer** delay – The delay in milliseconds between the sender and the receiver.
- **Float** dropRate – The fraction of packets lost. A floating point number was chosen as the measured packet loss may be fractional and this gives a higher precision for any calculation which is based on the packet loss.
- **Integer** total – The total number of packets received.
- **Integer** jitter – The jitter measurement in milliseconds.

- **Integer** packetSize – The mean average packet size for the media stream between the start time and end time stamps.

```
function trafficReport (startTime, endTime, srcIP, destIP, bandwidth, delay, dropRate, total, jitter,
packetSize) {
avinfo.videoBitrate = (1.3 * packetSize) / delay * sqrt (dropRate);
send (avinfo);
}
```

Figure 18 Example traffic report handler

Figure 18 shows an example *trafficReport* function which uses the CCA in order to implement a pure equation based rate control. This does not use the history from the current or previous runs but instead behaves in a similar way to TFRC [153] by using its fair share of the current network resources.

4.4.4. Conference maintenance

The Conference Controller aims to provide a consistency of quality to the users within the conference. During the course of a session Conference Monitoring packets are sent from the Participant Agents to the Conference Controller. This information is used to monitor how well the initial conference configuration matches the conditions which are currently being experienced on the network paths involved in the conference.

Each Participant Agent sends monitoring packets to the Conference Controller during the course of a conference. This information enters the policy and provides information which is used as the basis for deciding if the session parameters of a conference should be changed. If the session configuration needs to be changed this is considered a failure since the configuration should be valid for the entire of the conference.

One method of detecting upcoming problems in the network exploits the fact that routers queue packets when routes are becoming congested. This has the effect of increasing the delay on the path. If this increase could be detected it would allow problems to be anticipated before they happen. In 1998 Moon et al [62] and in 2000 Jaing et al [160] found a high correlation relating to round trip times in UDP traffic.. In 2003 Martin et al [161] came to the conclusion that an increased round trip time for TCP is correlated to packet loss only between 6% and 34% of the time, depending upon the network path taken. They stated that 'the correlation between loss event and increases in TCP RTT samples is weak'. This indicates that using the increase in round trip time is not a viable option.

Explicit congestion notification (ECN) [84] would allow congestion to be detected before packet loss but unfortunately it is not widely deployed enough to ensure that all problems could be caught by an ECN enabled router., which would mean that a second system would have to be deployed in order to catch all problems.

Packet loss could be used to indicate congestion. This would inform the CCA that the sending rate needs to be reduced. In order for this to work it would require that the CCA increased the sending rate at other times. This would create a '*sawtooth*' pattern which is similar to the one exhibited by TCP.

Problem detection can also be done by using the measurements that have been obtained for packet loss and the RTP throughput measurements. The CCA monitoring report contains the bit-rate at which the media stream was sent. The bit-rate received is known since this can be measured directly from the media stream. The levels of expected packet loss on the network path should be taken into account. If the receiver's bit-rate falls significantly short of the sender's bit-rate then this is an indication that unexpected packet loss has occurred on the network path in use. When performing this calculation the sender's bandwidth usage should

be calculated as the sum of the bandwidths of all streams which are currently being sent. The receiver's bandwidth is just the measurement of the currently received streams. This is the combination of all the streams which are currently being received.

4.5. Participant agent

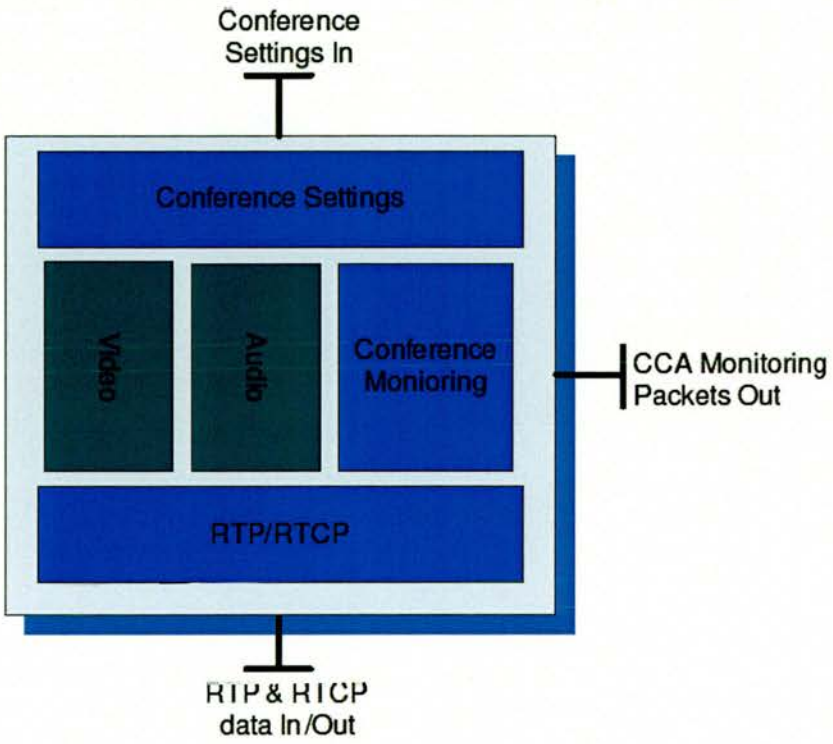


Figure 19 Participant Agent

The Participant Agent (PA) architecture shown in Figure 19 is the user facing part of the system and as such performs the actual videoconference section of the system; an example PA is shown in Figure 20. It performs capture, encoding, transmission as well as reception, decoding and playback of audio and video stream.

Figure 20 Example videoconference application

During the course of a session the Participant Agent performs monitoring of the RTP network traffic that it is receiving. From this monitoring it can discover

- The jitter
- Round trip time
- Bandwidth used
- Packet loss

- Packet size

4.6. Monitoring the conference

The monitoring of a conference performance is carried out at the Participant Agent.

The format for the CCA monitoring packet is as follows. Figure 21, below, shows the format of the header for the CCA monitoring packet. The packet is 32 bits wide.

0	8	16	24	32
Version Number	Number of reports	Reserved	Duration	
Time Stamp				

Figure 21 CCA monitor packet header

The version number indicates the version of this report. The version described here is version one of the reports. The version number is an eight bit unsigned integer field giving a possible 255 versions. The next byte of the header is an unsigned integer. This is used to indicate the number of reports that are held within this traffic report. This field indicates the number of reports that are to follow. Next is an unused byte. It is expected that this will be used for flags in the future. The next byte is used for the duration of the report period in seconds.

0	8	16	24	32
Source IP Address				
Ratio Lost	Number of Packets			
Bandwidth				
Delay		Jitter		
Packet Size		Reserved		

Figure 22 CCA Monitoring measurements format

Figure 22 shows the repeating section of the CCA monitoring packet. There is no receiver IP address in the packet since this can be determined from the UDP packet header, the receiver

of the media stream being the host which sent this monitoring packet. Each report section consists of the following fields;

- Source IP Address – This field is 32 bits long and contains the packed IP address. The first byte from the IP address is contained within the most significant bits of the integer with the second byte in the next most significant bits and so on. This is the IP address of the host which was sending the media stream which is not necessarily the host which sent the report, as the report came from the receiver of the stream.

- Ratio Lost – This is the ratio of the packets in relation to the number of packets in the *'Number of Packet'* field which have been lost. The number is an unsigned integer in the range 0 to 256. This gives a minimum of below 1%.

- Number of packets – This is the number of packets that were sampled in order to obtain the ratio of lost packets.

- Bandwidth – The bandwidth measurement is a 4 byte unsigned integer. The measurement is in bits per second.

- Delay – Delay is a 2 byte unsigned integer value. It is the delay which has been calculated for the path and it is expressed in milliseconds. This gives a maximum value of over a minute. A network with a round trip time of over a minute would be useless for video conferencing, but using a single byte only gives a maximum round trip time of a ¼ of a second, which could easily be exceeded on the Internet.

- Jitter – Jitter is a 2 byte unsigned integer value which contains the jitter in milliseconds.

- Packet size – This is the mean packet size of the packet stream.

- Reserved – Finally there is a 2 byte reserved field which is used to align the traffic report for easier unpacking.

There is no cap on the bandwidth which can be used by the CCA monitoring packets, but in normal usage it is kept low, at 1% of the bandwidth used since there is little information which needs to be sent to the Conference Controller in order for it to monitor the conference. However additional reports are sent if any of the measured parameters exceed a given tolerance. This tolerance is set during the conference configuration stage of the setup. Once this tolerance is exceeded the Participant Agent generates traffic reports which contain the measurements. They are sent back to the Conference Controller where the policy can decide the course of action to be taken and send out updates as needed.

Using this technique bandwidth usage for the traffic reports is kept low allowing more of the available bandwidth to be utilised for multimedia traffic. If there is a problem, then traffic reports are generated to inform the system that a problem has occurred.

Since UDP is used as the transport protocol for the Conference Controller Monitor reports there is the possibility that packets will be lost. Therefore if no changes have been made to the configuration of the session, a new report detailing the problem will be sent. This process will continue until the session has been configured so that the measured parameters no longer set off the alarm.

4.6.1. Network address translation

There is the possibility that packets will go through Network Address Translation (NAT) [162]. This has the effect of changing the source address for the reports. This, in most cases, would not affect the CCA since NAT is typically used for the connection from a small site to

the Internet with a private address space sharing a single global IP address on the Internet [163]. Normally the connection to the Internet is the limiting factor within a network since high bandwidth local networking technologies are relatively cheap. This also allows us to rule out information from IP addresses that are in the private address range. Since these addresses are allowed to repeat but never be connected to the Internet, NAT and proxies allow us to use a valid public IP address for the traffic reports for a given network. If this were not done it is possible that traffic reports for two different sites, in terms of network technologies i.e. different bandwidths, jitter, delay and packet loss could be viewed as the same IP address. This would have the effect that at least one of the networks could be incorrectly utilised during multimedia conferencing sessions.

This system would be able to operate if multiple ends of the connection were behind network address translating firewalls via the use of UDP hole punching [164]. Unfortunately this doesn't solve the problem of gaining access to the real IP address of the host.

4.6.2. Dynamic host configuration protocol

Dynamic Host Configuration Protocol (DHCP) [165] is sometimes used by Internet Service Providers to allow a limited number of IP addresses to be used to support a larger number of client connections. A single IP address is used for multiple clients. A large collection of clients is supported by a smaller set of IP addresses by relying on the fact that not all the clients will connect at the same time.

The implication of DHCP for the CCA is similar to NAT. A number of clients will be identified by a single IP address. If the pool of IP addresses is only used with a modem or only used with ISDN then the problem is reduced. When using a network connection from the DSL family (ADSL, SDSL etc) the range of network conditions is varied, bandwidth can easily range from 256 Kbps to 2048 kbps; these being bandwidths which are commonly

available in the UK at the time of writing. It is extremely likely that the high end of the range will continue to increase. This causes the problem that an IP address from a DHCP pool could go to any speed of connection. This could be countered by DSL being an always on technology. The always on nature means that the DHCP address pool and the number of clients has to be relatively equal, also the IP address leases are likely to be long term meaning that the CCA has time to 'learn' about the traffic behaviour of a specific connection.

4.7. Collecting network path measurements

A key component of the success of the Conference Controller Architecture is the collection of network measurements. The section describes how the measurements are collected by the Participant Agents.

4.7.1. End-to-end delay estimation

There are several potential ways to measure end-to-end delay. The first of which is when an RTCP Receiver's Report is received a timestamp is taken, A. The last senders report field (LSR) and the delay since last sender report (DLSR) are taken from the packet and the following calculation is performed.

$$RoundTripTime = A - LSR - DLSR$$

Equation 4 Round Trip Time calculation

This round trip time can be used to calculate an estimate for the delay between two points within the network by taking the round trip time and halving it. It is worth noting that this is only a crude estimate of the delay since it assumes that the delay in both directions on the path is symmetrical. This however is often not the case since many paths are asymmetric. Therefore the calculated round trip time is only an estimate of the real value for a given network path.

The end-to-end delay cannot be estimated without the synchronisation of the clocks of the computers involved. The nearest that can be performed is round trip time estimation which can be estimated by determining the difference between the current time and the LSR field minus the DLSR field from the RTCP report. If no sender's reports have yet been received then there is no possibility of determining the round trip time via this method.

It is possible to use the NTP timestamp from the incoming RTCP sender's report and compare it to the current time. This would require the sender's clock and the receiver's clock to be synchronised. Any clock drift between the sender and receiver would alter the delay measurement. There has been work towards compensating for the clock drift see [166] [167] for examples.

A final method of collecting the round trip time statistics is to use a separate traffic channel similar to the '*ping*' utility, the sending of a packet containing a timestamp to a receiver and then back again. The round trip time can then be calculated by subtracting the received timestamp from the time that the packet was received.

There are problems with all of these methods. The use of the NTP timestamp to estimate the end-to-end delay will only work if the systems clocks are synchronised, which may easily not be the case. All the other methods provide estimates of the round trip time. The asymmetry in Internet network paths means that it is not possible to calculate absolute end-to-end delay values and only estimations can be made. The estimation of the end-to-end delay is calculated as half of the round trip time.

These problems associated with the estimation of the end-to-end delay means that the estimation cannot be relied upon to be accurate and thusly it cannot be used for precise

timing information. At best it provides hints on the likely end-to-end delay but it is the best that can be provided without synchronised clocks.

4.7.2. Jitter estimation

The jitter is the delay variation between the packets arriving from the sender. There are three classifications of jitter [168]

1. Constant jitter – This is roughly the constant delay variation.

2. Transient jitter – This is characterised by a large increase in delay experienced by a single packet.

3. Short term delay variation - This is characterised by an increase in delay that persists for some number of packets and may be accompanied by an increase in packet to packet delay variation.

Jitter is made up of two timing variations; the variation in the sending rate and the variation in delay which is added to the packet on route. The variation in the delay which is added on route can be attributed to the following:

- multipath effects, where different packets take different routes to the destination which have different network properties.

- delay at routers, where the packets are queued at routers within the network. The delay is dependant on the queue length which is in turn dependant on how much the router is being used.

Various methods have been proposed for measuring jitter. The equation for mean packet to packet delay variation (MPPDV) which is the basis for the jitter measurement in RFC1889 [106] is shown in Equation 5.

$$MPPDV = mean(abs(t1 - t2))$$

Equation 5 Mean packet to packet delay variation

Where t1 and t2 are the end-to-end delay estimation times of two consecutive packets.

In the case of RTCP the estimated jitter reflects the most recent few hundred milliseconds worth of measurements. Therefore if an RTCP report is sent every 10 seconds then there will be no information for 95% of the time.

Mean absolute packet delay variation is calculated as

$$MAPDV = mean(abs(t1 - a1))$$

Equation 6 Mean absolute packet delay variation

Where t1 is the delay time and a1 the nominal arrival time.

This value can be misleading if a delay change occurs, as a constant offset would be included. As even fixed jitter buffers can adapt to delay shifts this means that the reported jitter value would not necessarily be a good indicator of ideal jitter buffer size or discard rate.

An alternative approach is to determine the mean absolute packet delay variation with regard to a short term average or minimum value.

$$Di = (15.Di-1+ti-1)/16$$
$$Pi = ti - Di\_if\_ti > Di$$
$$Ni = Di - ti\_if\_ti < Di$$
$$MAPDV2 = mean(Pi) + mean(Ni)$$

Equation 7 Mean absolute packet delay variation calculation in RTP

Where Di is the mean delay, Pi is the positive deviation and Ni is the negative deviation.

Y.1541 [169] defines delay in terms of the difference between the minimum and maximum transmission delays during a given time interval. The measurement of IP Delay Variations (IPDV) is defined as

$$IPTD\min = Minimum\_IP\_delay\_\text{variation}$$
$$IPTDupper = 99.9\_percental\_of\_IP\_transmission\_delay$$
$$IPDV = IPTDupper - IPTD\min$$

Equation 8 IP delay variations, reproduced from [169]

Each of these methods is suited for different types of jitter but none of them capture the information that is needed for configuring videoconferencing applications. Jitter measurements are required in order to choose buffers and aid in the choice of CODEC. In order to do this effectively the distribution of absolute jitter measurements must be known. Given this information (jitter measurements which represent the maximum jitter measurements for a number of time periods) a choice can be made as to how much of the jitter should be compensated for. For example if 99.99 percent of the maximum jitter measurements are under 40 ms with 0.01 percent at 1 second it makes sense to configure for the 99.99 percent of cases as the 0.01 percent case is rare but there is little that can be done to improve quality of service when the delay is at one second.

The amount of variation in inter-packet arrival time can be calculated in one of two ways. The first is to take the jitter estimate field from the RTCP reports. This approach has the advantage that little calculation is needed, since this has already been done by the RTP/RTCP subsystem. Within RTCP the interarrival jitter is defined as the mean deviation [106] of the difference in packet spacing at the receiver compared to the sender for a pair of packets. If the difference in packet spacing is D, Si is the RTP timestamp from packet i and Ri is the arrival timestamp for packet i, then for two packets i and j, D may be expressed as

$$D(i,j) = (Rj - Ri) - (Sj - Si) = (Rj - Sj) - (Ri - Si)$$

Equation 9 RTP delay calculation

The jitter is calculated continuously as each RTP packet is received from the source.

$$jitter_{new} = jitter_{old} + \frac{(D(i-1,i) - jitter_{old})}{16}$$

Equation 10 RTP jitter calculation from [106]

Whenever an RTP Reception Report is issued the value of $jitter_{new}$ is calculated. The gain parameter (16) gives a good noise reduction ratio while maintaining a reasonable rate of convergence [170].

The second method uses the differences in consecutive delay measurements as calculated from the Participant Agent's point of view. The disadvantage of this method stems from the problems of the delay estimates, which cannot be relied upon to be accurate due to factors which have been discussed in 4.7.1. The most accurate method of estimating the delay, being separate measurements similar to a ping, suffers from the fact that the measurement is actually a halved round trip time. This inaccuracy is increased when it is used to calculate the

inter-packet arrival times since two delay measurements are needed, both of which are possibly inaccurate, the difference between the two is the inter arrival time.

The chosen method of inter-packet delay measurement is to report on jitter, the maximum jitter between consecutive packets over a period of time n to n+m:

$$\max((t_{n+1} - t_n), (t_{n+2} - t_{n+1}), \ldots, (t_{n+m} - t_{n+(m-1)}))$$

Equation 11 Jitter measurement in the CCA

This method of measuring jitter favours the reporting of high levels of jitter. This was chosen since the jitter measurements are used to configure buffers and to hint to delay which should be taken into account when choosing (for example) CODECs or play-out buffers. The measurement consists of a start and end time for the measurement.

4.7.3. Throughput estimation

The throughput used during the session is the sum of

1. RTP data packets
2. RTCP monitoring packets
3. CCA control packets
4. CCA monitoring packets

The bandwidth used by the RTP data can be discovered by measuring the amount of data which makes up the multimedia streams received from the network. This should not include any packets that are reconstructed or generated to cover packet loss. RTCP does not include a bit-rate measurement. The closest is the octet count which is the number of octets which have been transmitted since the last senders report. Therefore in order to calculate the RTP

throughput the measurement has to be made independent of RTCP. The RTCP bandwidth can be calculated in the same manner and a record of sizes for the CCA control and monitoring packets is kept.

The bandwidth used is the total of all of the throughputs for the various parts of the system. It should be noted that the RTP data packets, RTCP monitoring packet and the CCA Control packets are multicast to all participants within the session whereas the CCA monitoring packets are only sent to the Conference Controller from each of the Participant Agents.

The RTP data throughput is measured within the Participant Agent by the player which is presenting the material to the user. This throughput is the one which is the most interesting since it is the value that the multimedia streams will be set to in order to achieve the best results.

RTCP is set-up so that it consumes no more than five percent of the overall bandwidth of an RTP session. In contrast, the monitoring of the Conference Controller Architecture has no restrictions on the amount of bandwidth that it consumes, but it only uses bandwidth when it is making a change to the configuration of the session which will only happen when the session is mis-configured in some way. Of course, there is the problem that the time at which there is the most need to update the session configuration is the time when the network will have the lowest level of performance.

4.7.4. Congestion estimation

The congestion information concerns the amount of packet loss that has been experienced by the multimedia traffic as it traverses the network. RTP packets have a sequence number which is incremented by one each time a packet is sent. Detecting packet loss is a case of

recording the last received sequence number. Packet loss is registered when the next received packet has a sequence number more than one higher than the last received packet. The number of packets which have been lost can be calculated by simply adding one to the last received sequence number and subtracting that from the sequence number of the packet which has just been received. A consequence of this is that packet reordering is treated as packet loss. This may not be the correct classification from the network's perspective since the packet was delivered, but from the application's point of view a packet arriving too late is lost and therefore should be treated as such.

This process already takes place as part of the RTCP reporting process, with the result of the calculation being placed into a packet loss field. The reported packet losses are summarised therefore reducing precision; further since packet loss is '*bursty*' any packet recovery should be configured for higher packet loss than that measured in order to provide adequate protection. Packet loss is reported as the number of lost packets between two time periods (start-time to end-time).

4.8. Addressing session configuration

This section addresses the configuration of audio/video transmission and reception and how exploiting knowledge about likely network conditions can be used to configure a videoconferencing session for the maximum quality. The CCA is able to use layered encoding or multiple versions if there is a small number of participants using relatively low bandwidth connections.

It is important to realise that in this context quality relates to the overall quality of the videoconferencing session. This takes into account both audio and visual quality but also the delays between saying something and the other parties being able to hear it. The packet

recovery techniques which are used also play an important part in the process since they are able to cover up packet loss within the system.

When configuring the session parameters for a videoconference the known details about the networks paths in use are estimates of the following

1. delay
2. jitter
3. packet loss
4. throughput bandwidth
5. packet size

This information is used in the configuration. The throughput limits the maximum bit-rate for the chosen CODECs. This must be shared by all transmission streams. If there is an audio and video stream then they must share the bandwidth. It is not sensible to split the bandwidth evenly for each of the streams, since the audio stream is more important for conveying information than the video. Bandwidth must be given to the audio stream in preference to the video stream up until the point where the audio stream gives a 'reasonable' quality (defined as telephone quality for the purposes of this thesis). After that point video can be introduced but at minimal frame-rates and overall visual quality. This introduces the visual cues which aid in the understanding of conversations. The video quality is slowly increased while the audio quality is quickly increased up until the point where 'good' quality (defined as radio quality for the purposes of this thesis) audio is being used. From this point on it is the video quality which is increased while the audio remains at the same quality. The reason why quality has been given in terms of 'telephone' and 'radio' is that if different CODECs are used then the absolute settings of bit-rate, sample rate etc will be different in order to give reasonable or good quality audio. The reason why different CODECs would be

employed in different situations is that different CODECs have different properties which relate to how well lost packets can be reconstructed, how long it takes to encode the data which impacts upon their effectiveness within a videoconference when used on network paths with varying end to end delay times. These issues have to be addressed in the session configuration.

In order to be a good network citizen the bandwidth which we consume for a multimedia stream should be no more than that consumed by a TCP stream when exposed to the same network conditions. This places an upper bound on the bandwidth the multimedia data consumes. There are two ways which the data could be accounted for. Firstly we could count each RTP stream as equivalent to a TCP stream while the alternative is to count all RTP streams as equivalent to a single TCP stream. The approach taken is to treat each RTP stream as a single TCP connection and the bandwidth that may be consumed is the sum of all the RTP connections.

The argument can be made that uneven allocation of network resources is desirable since some forms of traffic are inherently more important than others, for example real time video has a greater importance than peer-to-peer file transfers. While this can be argued it does not address how the split between the importances of these network streams is to be allocated. Just because a video stream is important in this case it does not mean that it will be important in the next, take for example a videoconference and a webcam of a fish tank. It is clear that the former has a higher value than the latter. Even this does not hold very far since the use and user will affect the importance. Even though the case can be made for uneven allocation of network resources to differing packet streams this system makes no attempt to enforce it. The argument for this is two fold, firstly since TCP is the dominant protocol used on the Internet at this time it is important to behave in a way that is consistent with TCP. Secondly,

since the adoption of QoS technology at the core of the network is unlikely to happen any solution must be a network edge deployment. Therefore, it is not possible to enforce the dominance of a single media stream without increasing packet loss for all the traffic on the network. Many systems have been designed to support the uneven allocation of network resources; for examples see [91, 93, 95], but they require a wide scale of network deployment.

The throughput is the sum of all the audio, video streams as well as the RTCP packets, the CCA monitoring packets and also if necessary the CCA configuration packets. When available bandwidth is quite restrictive, silence suppression can be used in several different configurations. Firstly with silence suppression turned off all participant agents will transmit both audio and video all of the time. This is quite wasteful of resources since silence is transmitted over the audio channel even when no-one is speaking. The second configuration is 'always on' video and the audio with silence suppression. This has several effects. Firstly more bandwidth is available for the audio channel since it does not have to be shared amongst all participants. Secondly background noise is not transmitted from participants who are not actively taking place in the discussion which would be a source of distraction within the videoconference. The third possible configuration is to have only one transmitter at a time. This is in effect silence suppression not only operating on the audio but also on the video. So when someone is no longer speaking their video stream freezes. This configuration provides the most bandwidth for the person that is speaking. The situation where more than one person talks at a time, should be taken care of by normal social cue mechanisms within face to face speech. The final configuration which is possible still has silence suppression turned on. The audio is dropped when the person is not speaking but instead of dropping the video it is instead transmitted at a reduced quality. At first glance this appears to not fit in with the attempt to maintain a consistent quality of the audio and video. Instead of reducing aspects of the video such as colour depth or CODEC it is preferred to reduce the frame-rate.

If we only change the frame-rate we can still reduce the bandwidth with little loss of quality. When someone is talking their lip movements and body language must be in sync with their speech in order for it to look right. When they are not speaking there is no need to have the higher frame-rates that are needed to provide the lip movements which are needed to synchronise the audio with the video movements, gross movement could still be picked up to some degree with lower frame-rates.

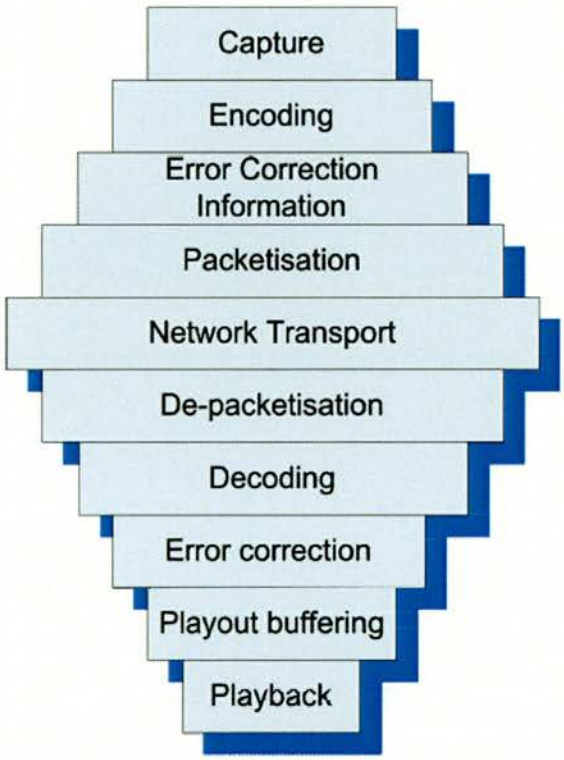


Figure 23 Sources of end-to-end delay

The end-to-end delay of a connection should be less than 150 ms [30]. The representation of the components that make up the end-to-end delay is shown in Figure 23.

The capture and playback aspects represent the media entering and exiting the computer. There is little that can be done to shape this aspect of the end-to-end delay. This is also the

case with the packetisation and depacketisation processes which represent taking the encoded media data and encapsulating it into the RTP format.

The point at which we are first able to influence the end to end delay is at the encoding/decoding stage. There are several possibilities to shape the delay generated by this stage. The first relates to the choice of CODEC. Different CODECs introduce different delays when encoding. Normally the CODECs which introduce higher delays are also the ones that provide the higher quality since they have more data and more time to work with. The set encoding quality also has an impact on the delay. Higher quality settings will require more processing time and therefore increase the delay. Therefore they can only be used if the network end-to-end delay is low.

The addition of sender based packet recovery information may also add to the delay. There is the option of leaving out sender based recovery information if the packet loss on the paths involved is low enough that receiver based techniques will operate sufficiently well, i.e. if the packet loss is below 15%.

The delay experienced by the media stream as it traverses the network cannot be controlled to any reasonable degree and cannot be reduced below a minimum delay, composed of transport across the physical wire and the processing time required at the routers along the path. If a network bottleneck is close to the end point generating the traffic then transmitting data at a rate above the bottleneck bandwidth means that packets will have to be queued at a router which has the effect of increasing the end-to-end delay. Keeping the sender's bit-rate below the bandwidth at the network bottleneck ensures that the delay is minimised.

Jitter is the result of changes in queue length at the routers along the network path in use. Packets which are received will almost certainly not be received at exactly the right time for

playback. In order to compensate for this play-out buffers are used. They delay the playback of packets in order to smooth out the jitter. The minimum play-out buffer should be chosen so as to add the minimum amount of delay to the media stream. Jitter below 40ms is not perceivable so jitter should be kept below this level by the use of smoothing buffers.

The estimates of packet loss inform the choice of packet repair technique. The two main classifications, receiver and sender, are distinguished by the need to add extra information into the bit stream for sender based techniques. The advantage of using some of the available bandwidth for error correction information is that the media stream is more resilient to packet loss and therefore able to reconstruct packets from streams with higher levels of packet loss than if receiver based techniques were used. Receiver based techniques are based purely at the receiver and therefore do not require the additional information to be encoded into the media stream. This leaves more bandwidth available to the actual media rather than error correction information.

Receiver based systems are known to operate effectively up to around 15% packet loss, while sender based systems operate above this level. At low levels of packet loss noise substitution can be used to fill in lost packets. This relies on the participants being able to mentally reconstruct the lost packets.

4.8.1. Summary

This chapter has presented the Conference Controller Architecture, which provide a framework for quality of service based on a network measurement approach. It consists of three modules, Traffic Data Repository, Conference Controller and two or more Participant Agents. Of these the Conference Controller is the core of the system, providing a mechanism to run policies. These policies encompass the expertise of configuring and maintaining videoconference sessions. Experiences of past videoconferences are provided in the form of

network measurements which can be used to deduce how well the video conference performed on the network during a given session.

Chapter 5

Scenario Driven Network Emulation

5. Scenario driven network emulation

In the process of testing the CCA as part of the development it was necessary to schedule videoconference test sessions. Unfortunately, there was not a constant supply of participants who were willing and able to repeatedly perform in a consistent manner during the test sessions. Furthermore, the CCA is intended to support conferences consisting of a wide variety of network connections with different characteristics, which is difficult to replicate on demand for development and testing using real networks. Hence the need for a test bed and the creation of the Scenario Driven Network Emulator (SDNE) [171, 172].

The SDNE provides two facilities to aid the development and testing of real time interactive applications. Firstly, a set of emulated networks allow experimentation without the need to use real networks. Secondly, the SDNE provides a method of simulating user and group characteristics such as the number of users in a conference and their patterns of audio communication.

The architecture of the SDNE is shown in Figure 24. There are three sections: the configuration interface and scenario organiser, the emulation manager, and the node controllers. There are two optional aspects; the user interface and the maps. The maps are only used when the system is being used to emulate wireless nodes.

The interface to the system consists of files for input and optionally a window which displays the movement of nodes when wireless modes have been configured. The emulation manager is a centralised controller which co-ordinates a session. Finally, there are a number of node controllers, one for each emulated node.

The node controllers take instructions from the emulation manager on how to configure their virtual network interface. This includes values for parameters such as bandwidth, loss rate, delay and jitter. The exact behaviour is controlled by a traffic shaper in each node controller, StAnd Net, which is described in this chapter. The SDNE architecture is shown in Figure 24.

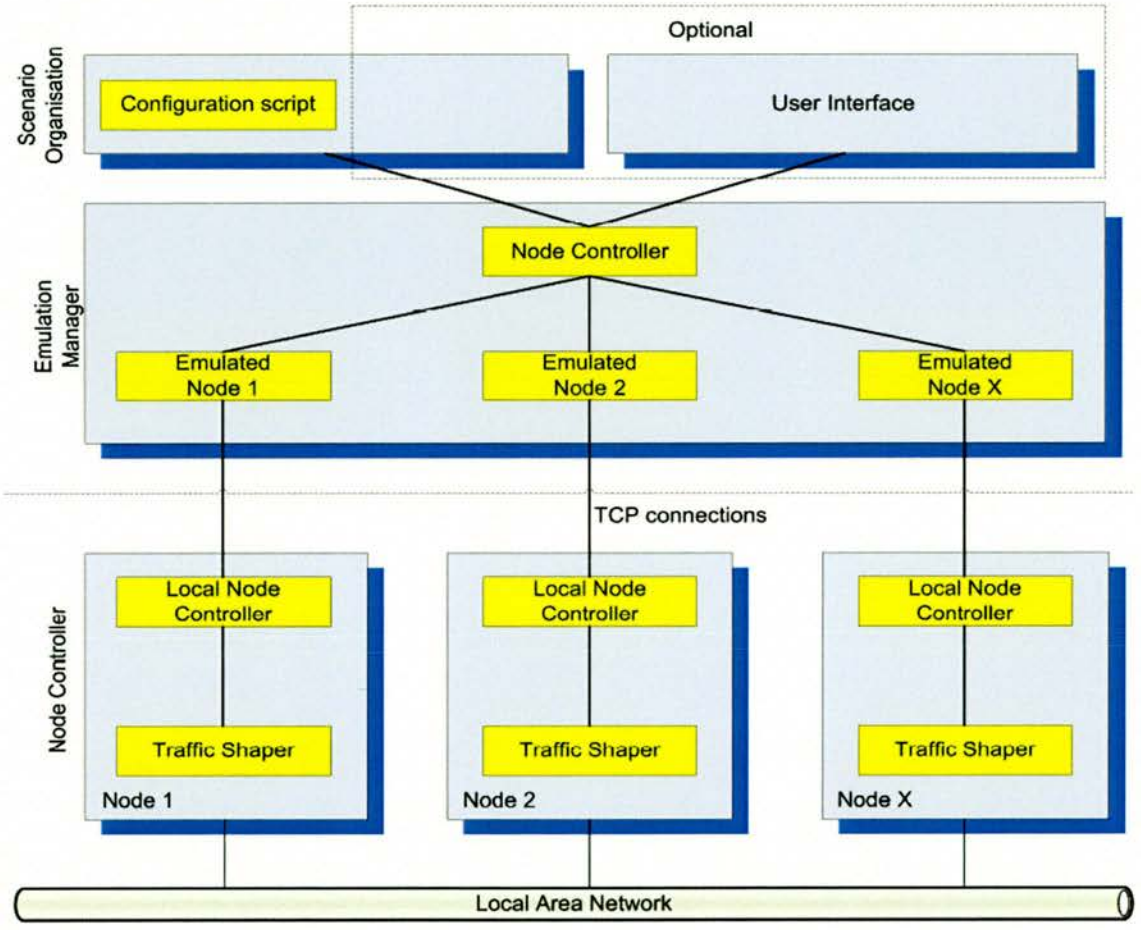


Figure 24 SDNE Architecture

5.1. Network emulation design considerations

An initial issue was whether to opt for simulation or emulation of the network connections. Experimentation by simulation involves creating a computer model of the application to be

tested and of the network environment in which that application will run. The model is then run and measurements can be made. Simulation, as a method of experimentation, has potential advantages in that no additional hardware is typically required to perform experiments. The amount of time needed to perform each experiment is however dependent on the complexity of the computer model. If the model is relatively simple, or the hardware the simulation is running on is relatively fast, a simulation run may require less time than an emulation or live experiment. On the other hand, if a complex model is used it is possible that it will require more time than would be needed to perform emulation of live experiments. The quality of data retrieved from a simulation is dependent on the quality of the model.

An emulation-based experiment takes the actual application and places it in an artificial environment. In this case the application consists of a set of clients using the Conference Controller Architecture and an artificial network environment. This gives the opportunity to use the real application within various networking conditions to see how the system performs and allows a level of control over the test environment which cannot be attained without difficulty when using live experiments. Another advantage of emulation over simulation is that it allows the use of specialist hardware such as video capture.

Each of the methods described have their strengths and weaknesses. Simulation techniques work well for systems that can be modelled more easily than writing an actual system or in the situation where the actual system doesn't have to be built. An emulation approach provides a test bed for the actual system. Since an implementation of the Conference Controller Architecture has been produced and used in real sessions, testing it in an environment based on emulated networks allows the actual implementation to be used without having to re-implement large sections of the code for use on a network simulator.

This also has the advantage that it allows hardware, such as video and audio capture devices to be used for producing more realistic network traffic. Accordingly, emulation was chosen as the basis for a test bed.

5.1.1. Requirements of the network emulator

In order to emulate a variety of network connections the following parameters should be configurable:

- Delay
- Throughput
- Packet loss
- Jitter
- Packet drop probability
- Multipath effects[14]
- Packet duplication
- Multicast

In addition, the platform (OS, Hardware) on which an emulator runs and also the number of real nodes on which it can be run, may influence performance and realism.

[14] Multipath effects are the result of packets within a network stream taking different network paths in order to reach their destination. This can result in packet reordering if one of the paths to the destination takes a shorter amount of time to traverse than others allowing the packet travelling down the shorter route to 'overtake' the packets which are travelling down the longer route.

5.1.2. Survey of existing emulation systems

There are two general approaches to the creation of network emulators. The first predominantly uses software to emulate the network and network conditions, the second uses hardware to emulate the network and any prevalent conditions. Any actual test bed is likely to contain elements of both approaches.

5.1.3. Hardware based network emulation

Hardware based emulation requires that the physical connections between the machines are altered in order to create the network topology that is required. Each machine that is used on the network must have all the physical network cards that are required in order to create that network. For example, a router which takes part on four networks would require four network interface cards.

5.1.3.1. Emulab

Emulab [173] from the University of Utah aims to provide a universally available 'Internet Emulator' which provides a balance between control and realism. It consists of 168 PC nodes, 40 of which are PIII 600 MHz and the other 128 are PIII 850 MHz. Each of the nodes has 5 network cards. 4 Cisco 6509 high-end switches provide the network, 2 of which act as the test bed backplane. Each of the nodes has 4 network cards connected to the test bed backplane. The final network interface is linked to the core router for the test bed and acts as a control interface providing access to the node for configuration and also access to the node from outside the test bed. The test bed backplane can be controlled by remote configuration tools and allows the nodes to be placed on VLAN's (Virtual LAN's) [174] in arbitrary ways to build up a test network. The test bed backplane is a network switch which can be set-up to emulate any network topology. There are also nodes within the system that are used solely to perform traffic shaping. Configuration of the virtual network topology is done using TCL [175] scripts and uses the same syntax as NS (Network Simulator) [176].

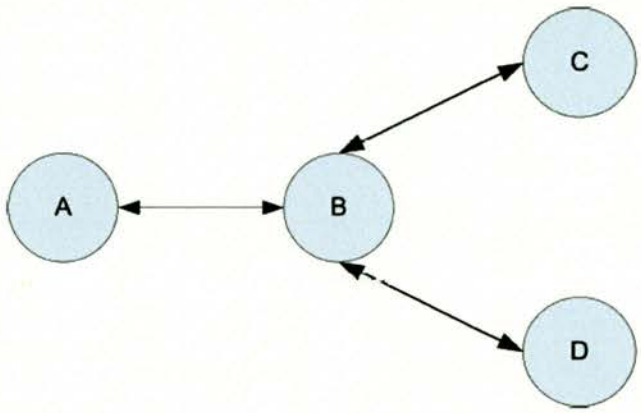


Figure 25 A Simple Network topology, reproduced from [177]

Figure 25 is created from the instructions in Figure 26.

```
1) # This is a simple ns script. Comments start with #.
2) set ns [new Simulator]
3) source tb_compat.tcl

4) # define the 4 nodes in the topology.
5) set NodeA [$ns node]
6) set NodeB [$ns node]
7) set NodeC [$ns node]
8) set NodeD [$ns node]

9) # Next define the 3 links between the nodes.
10)$ns duplex-link $NodeA $NodeB 100Mb 50ms DropTail
11)$ns duplex-link $NodeB $NodeC 100Mb .1ms DropTail
12)$ns duplex-link $NodeB $NodeD 100Mb .1ms DropTail

13)# Set the OS on a couple.
14)tb-set-node-os $NodeA FBSD-STD
15)tb-set-node-os $NodeC RHL-STD

16)# Set IP address of node B on the port going to node C
17)tb-set-ip-interface $NodeB $NodeC 192.168.42.42

18)# Go!
19)$ns run
```

Figure 26 Instructions to create the network taken from [177]

Emulab allows the customisation of the nodes. Installation of new operating systems which were not part of the original cluster, or additional packages is supported. The system allows traffic shaping to be performed between any nodes within the VLAN. The parameters that can be altered are the bandwidth, delay and packet loss. Currently delays of 3 milliseconds or less are too small to be emulated correctly.

All of the hardware and software for use in the emulation is hosted at the University of Utah. This could potentially cause problems. For example there needs to be a network path from the local site to the University of Utah in order to use the system. It is not unusual for problems such as network outages to occur which could potentially restrict access to their system. If large audio and video files or streaming video is needed for the duration of one of the experiments this data would need to be transported to the test bed, which may take a considerable amount of time depending on the prevailing network conditions. Actual perceived quality of outputs such as video would be impossible to assess as the public Internet sits between the test bed and the user. At the time of writing 16 projects (14 of which are external to the University of Utah) have used Emulab but there are already two new Emulabs being constructed at The Universities of Kentucky and Stuttgart.

This form of network emulation provides a very good solution if an Emulab is available locally, or if local resources are limited. However, this is a not the case at St Andrews where a Beowulf cluster of 64 nodes is available locally for use in experimentation. The disadvantages of remote administration and execution do not make the use of a remote Emulab attractive or appropriate for the SDNE.

5.1.4. Software oriented network emulation

Using software to emulate networks allows rapid reconfiguration of the network and the network conditions since all changes are done in software and without the need to rearrange

physical equipment. Software network emulation can use either physical or virtual network device interfaces. Using virtual network devices allows any machines on a network to be used as all they need is a single physical network device and a network route which allows it to access the other machines on the virtual network. This adds flexibility since network configurations can be created and altered without the need for specialised hardware.

Four software oriented network emulation tools were reviewed; NIST Net, the Linux Traffic Shaper, Dummynet and Ohio Network Emulator (ONE).

5.1.4.1. NIST Net

NIST Net [178], from the National Institute of Standards and Technology (NIST), is a network emulation package that runs under Linux (kernel versions 2.0.xx to 2.4.xx). There is no limit on the type and number of network interfaces that can be used with the NIST Net traffic shaper. This means that it can be used in conjunction with both physical and virtual network devices in order to connect to more networks than there are physical devices.

Timing for NIST Net is provided by the standard PC clock. When NIST Net was originally written the Linux clock rate was 100 Hz. This is insufficient for traffic shaping so NIST Net takes over the clock function and resets it to 8192 Hz, which is the maximum allowable value. Resetting the clock rate could have potential problems in that some clock ticks may be missed if device drivers disable interrupts for too long.

NIST Net shapes traffic based on a table of three-tuples. The parts are:

1. a specification which is used to match the packets
2. effects to be applied to matching packets
3. statistics about packets which have matched the specification

NIST Net is able to configure packet delay (both fixed and variable); packet reordering; packet loss (random and congestion dependent); packet duplication and bandwidth limitation. The non-bandwidth dependent packet delay may be fixed or random, with the shape of the random distribution being set at runtime. Non-congestion related packet loss and duplication are provided with a similar system. As well as the ordinary packet loss a congestion dependent packet dropping scheme is provided. For this Derivative Random Drop (DRD) [179] is used. The minimum and maximum thresholds are settable parameters within NIST Net thereby allowing the steepness of the drop probability ramp to be set. Explicit Congestion Notification is implemented through DRD. Bandwidth limits are calculated and imposed on an instantaneous basis. When a packet arrives which matches a bandwidth limiting entry the amount of time the packet would take to transmit is calculated. Any subsequent packets are then delayed by at least this amount, plus any amount they should be delayed for bandwidth limiting.

When using NIST Net it is only possible to shape incoming traffic so a router must be created within the emulated network environment in order for traffic shaping to take effect.

The NIST Net documentation [180] raises the issue that delays are not enforced until the bandwidth limit is reached for the path. After experimenting with this feature using the *ping* command and setting the delay on a network path to 5 seconds and the bandwidth to 100 megabits per second it was confirmed that this is in fact the case.

NIST Net imposes very little additional overhead when it is used as a router. 5 - 7 microseconds is quoted for a Pentium 200 MHz with 32 megabytes memory emulating a 100 megabits per second network, where overhead is the additional delay created by using NIST Net.

NIST Net provides a simple graphical user interface, which makes it easy to configure and monitor the parameters for any given network link. The NIST Net network emulator also provides a command line tool with options for setting the network shaping parameters as well as monitoring the current condition of the NIST Net traffic shaping. The parameters that can be viewed include the current bandwidth over a given link, the current packet size and the length of the queue in a router.

During the experimentation with NIST Net routes would sometimes fail. This causes no traffic to be forwarded down the routes involved. The problem was isolated by turning off NIST Net and the packet would start to flow again. This problem is caused by the interaction between NIST Net and the experimental test bed which was used. An upgrade of the Linux kernel to 2.4.20 did not solve the problem. This leads to NIST Net being unsuitable for use within the SDNE system since the interactions between NIST Net and the SDNE system create unforeseen effects.

5.1.4.2. Linux traffic shaper

The Linux Traffic Shaper [181] is a standard part of the Linux Kernel, as of kernel version 2.0.36. The traffic shaper requires that the user has a good understanding of networking and routing since a shaper pseudo-network device is added to the system. The traffic shaper is treated like a normal network device in that valid routing rules must be set in order for the device to operate. The *shapecfg* command line tool is used to change the settings of the shaper device. Unfortunately the traffic shaping that the Linux kernel shaper device can perform is somewhat limited as it can only shape bandwidths and not any of the other network characteristics such as delay, jitter or packet loss.

The Linux shaper device has the advantage that it is able to shape outgoing traffic so that it will produce network traffic with the correct characteristics as an end node and it is able to act as a router within an emulated system.

Traffic can be shaped for given paths and given ports, so it is possible to restrict the network characteristics for specific protocols. For example it is possible to assign different bandwidths for FTP and HTTP access to the same host.

5.1.4.3. Dummynet

Dummynet [182] is a traffic shaper for use with FreeBSD [183]. It enforces queue length, bandwidth, delays, packet losses and multipath effects. It has been part of FreeBSD since 1998 and is now available in FreeBSD 4.3-STABLE and later. Configuration is done via the *ipfw* (the FreeBSD IP firewalling) command which operates on the network devices. It is used extensively in the research community [173, 184-186].

Dummynet provides the simulation of multiple paths to the same location. This allows packet reordering to be emulated since each virtual path can have different network characteristics.

Dummynet operates on both the input and output packet queues if it is being used within a router. It is therefore important that the rules are configured carefully so that the same rule is not applied twice.

5.1.4.4. Ohio network emulator

The Ohio Network Emulator (ONE) [187] is a traffic shaper for Solaris-based machines. The user is able to control network characteristics such as delay, queuing and bandwidth. The traffic shaping is performed on a per-interface basis, rather than on a destination basis. ONE was initially developed by Ohio Universities Internetworking Research Group but is now

maintained in conjunction with the NASA Glenn Research Centre's Satellite Networks and Architectures Branch and as such it is particularly suited to the emulation of network links within satellite systems.

5.1.4.5. Comparison of software based approaches

NIST Net provides many of the features which are needed in order to fully test the CCA. It lacks a native method of implementing jitter but this could be added by using a small script which constantly changes the delay on a given network path, although this would add a level of complexity. It runs on top of the GNU/Linux operating system which would facilitate its use on part of the departmental Beowulf cluster[15].

The Linux Traffic shaper does not provide many of the features that are required for the CCA tests. For example it lacks the ability to delay traffic and to drop packets. It is therefore not suitable for testing the CCA.

ONE provides almost all of the required functionality - delay, queuing of packets and bandwidth shaping are supported – but not jitter. ONE requires Solaris and more importantly can only shape traffic on a physical interface which restricts flexibility of configuration.

Dummynet has the advantage over the other reviewed systems that it is able to emulate multi-path effects within the system but requires additional hardware where it can be installed.

Neither ONE nor the Linux Traffic shaper meets all the requirements for the CCA test bed network emulator. Both NIST Net and dummynet provide the required functionality, with a

[15] The cluster consists of 64 Pentium III 400 Mhz based computers each with 384 megabytes of memory. The head of the cluster provides NFS exported home directories and acts as a router between the cluster and the rest of the departmental network.

small addition to each in order for jitter to be introduced to the system. NIST has the advantage that it could be run across several nodes of the departmental Beowulf cluster which runs GNU/Linux. However, after evaluating NIST Net on the Beowulf it became clear that the problem of not delaying traffic when the bandwidth utilisation was below a certain level was more of a serious drawback than first anticipated. This, along with the lack of a suitable alternative, prompted the development of StAnd Net which aims to meet all the requirements for the network emulator component of a CCA test bed.

5.2. StAnd Net

StAnd Net [188], architecture shown in Figure 27, is an IP level network traffic shaper. It was inspired by NIST Net and includes all the useful features of NIST Net with the addition of the missing features which are required for the testing of the CCA.

In order to make StAnd Net simple to install and use it was decided to abandon the timing information from the Linux real time clock. Instead timing information is provided for the 2.4 series of kernels by Love's variable hertz kernel patch[16] [189], which is included by default for 2.5 and above series of kernels. This provides timing information down to a millisecond with a default installation of the 2.6 branch of the kernel but increased granularity can be obtained by increasing the kernel jiffies[17] value, so delays for ½ a millisecond are possible if required. The default value of 1024 Hz provides a one millisecond granularity which provides a reasonable accuracy of traffic shaping.

[16] This was originally designed to increase the responsiveness of interactive applications.

[17] It is a variable that keeps increasing forever until it wraps back to zero at which time it still keeps increasing. It gets increased at the HZ (a defined constant) rate as a result of a hardware interrupt. It is used for various timing functions within the kernel.

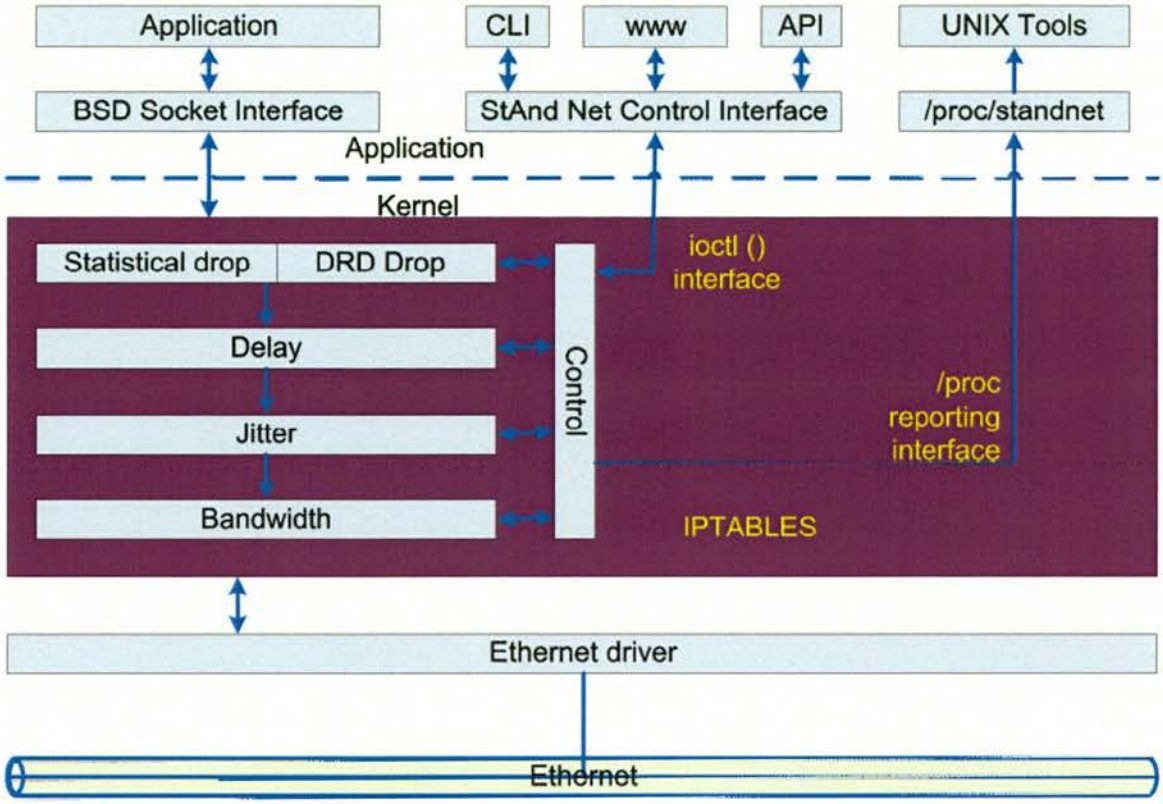


Figure 27 StAnd Net architecture

Instead of creating the traditional command line and a console graphical user interface found in NIST Net and others, an application programming interface, a command line, and a web-based interface (an example is shown in Figure 28) were created. The motivation for the web interface is that computers engaged in network emulation do not have monitors attached, as is the case of a Beowulf cluster, so it makes sense to move the control interface to a users' computer. The APIs are motivated by work on wireless network emulation [172] where it was necessary to programmatically control the emulated network settings in order to correctly emulate a wireless environment. Finally the command line tool allows for ease of configuration when it is not suitable to use either the API or the web interface, and allows for StAnd Net to be used from within scripts.

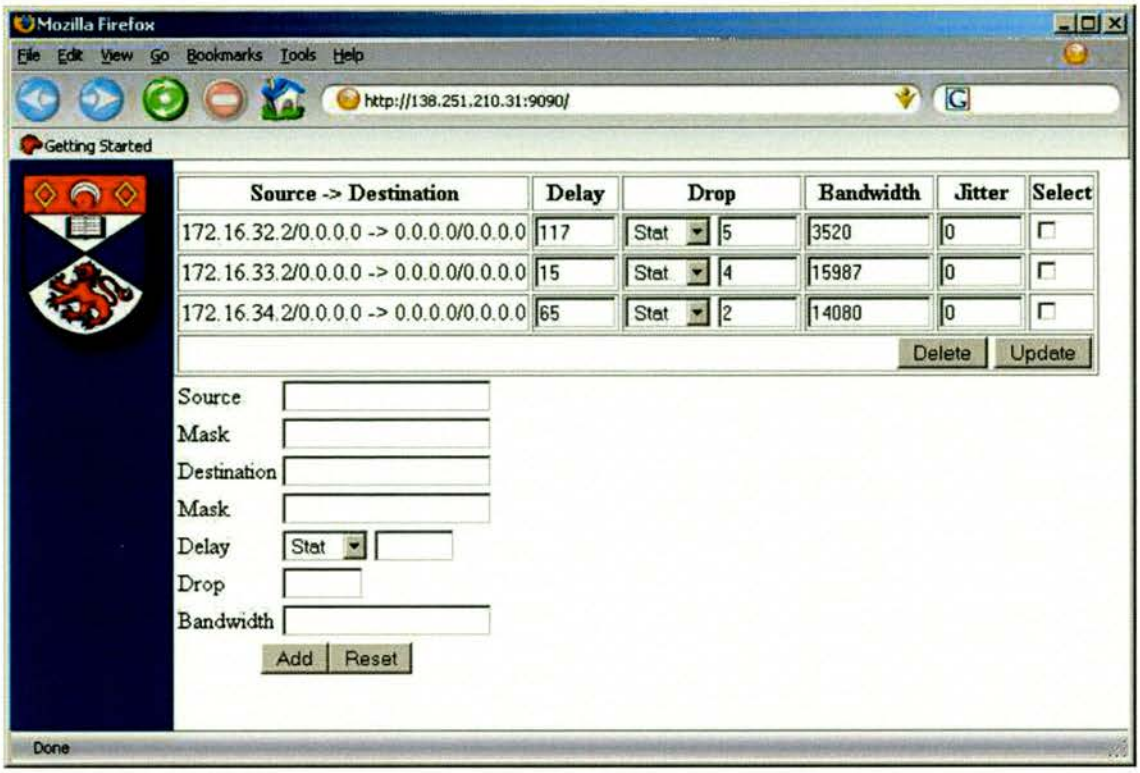


Figure 28 StAnd Net web based interface

StAnd Net is implemented as a loadable kernel module which uses IP_HOOKS from the netfilter [190] system to intercept packets as they traverse the network stack. It is thus able to shape either incoming or outgoing traffic or both for the series 2.4.0 Linux kernel and above. This allows complex network topologies to be built while allowing the hosts to shape their own network traffic.

Two packet dropping algorithms were implemented so the most appropriate for the situation can be chosen. There is a simple statistical approach where a percentage of packets are dropped. Unfortunately this algorithm has limitations. Firstly since it is based on a pseudo-random number generator it creates packet loss at around the level requested, usually within 1%. The pseudo-random number generator is based on the current jiffies value within the kernel. As such it is dependent on the time at which the packet arrived. As some events happen at regular intervals (such as the update of the clock) some jiffy values are blocked

meaning that some deviation from the set percentage is to be expected. For congestion dependent packet loss Derivative Random Drop (DRD) [179] was implemented. DRD was chosen over Random Early Detection (RED) [119] due to implementation simplicity and low processor overhead. The potential problem of DRD creating a coordination of packet drops and retransmissions across multiple flows does not affect StAnd Net since the packet flows are isolated from each other.

Two application level interfaces to StAnd Net are provided; the first of which is a control library which simplifies access to the features of StAnd Net allowing third party applications to shape their own traffic. This interface was used to create a command line client, World Wide Web interface and a Java API, via the use of the Java Native Interface [191]. This interface allows the addition, editing and removal of traffic shaping rules. Reporting is left to the second interface. This is provided as a number of virtual files in the *proc* file system, thereby allowing reporting to be performed using the standard UNIX tools (*cat*, *grep*, *cut*, etc). The split in functionality between the two interfaces was made because UNIX already provides a large number of text processing tools which can easily be used on the logs generated by the proc file system interface.

Under Linux the proc file system is a virtual file system. It is a direct link to the kernel and the status or configuration of various parts of the kernel is available through virtual files where application level read and writes to these files are passed on to the kernel. StAnd Net provides the reporting information under */proc/standnet*. Each rule has a corresponding entry in */proc/standnet* (see Figure 29 for an example) which reports on the current settings as well as information concerning the current state of the rule. The report includes the number of packets processed, the number of packets dropped, the delay currently being used (this

depends on the bandwidth and current queue length), the length of the queue and the current throughput of the rule.

```
-r--r--r--  1 root  root  0 Oct 26 12:11 172.16.32.2-0.0.0.0
-r--r--r--  1 root  root  0 Oct 26 12:11 172.16.33.2-0.0.0.0
-r--r--r--  1 root  root  0 Oct 26 12:11 172.16.34.2-0.0.0.0
-r--r--r--  1 root  root  0 Oct 26 12:11 172.16.35.2-0.0.0.0
-r--r--r--  1 root  root  0 Oct 26 12:11 172.16.36.2-0.0.0.0
-r--r--r--  1 root  root  0 Oct 26 12:11 172.16.37.2-0.0.0.0
```

Figure 29 The reporting entries in /proc/standnet/

Figure 30 gives an example of the output from the reporting system. This is the result of using the *cat* UNIX tool on /proc/standnet/172.16.32.2-0.0.0.0.

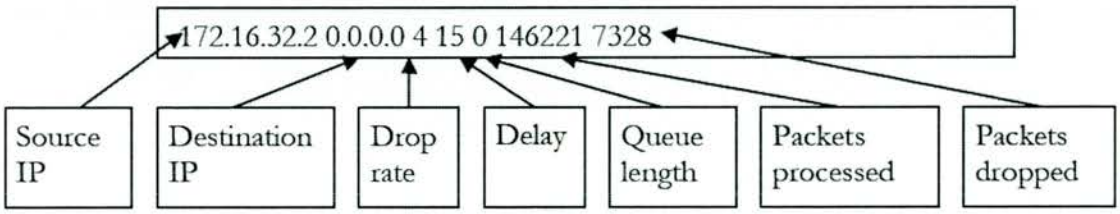


Figure 30 The reporting entries in /proc/standnet/172.16.32.2-0.0.0.0

Each line from the reporting interface contains all the information to recognise the rule which created it as well as the number of packets which are currently in the queue and the number of packets which have been processed.

5.2.1. Evaluation

The evaluation was performed using Fedora Core 2 test 1 which by default comes with the Linux kernel version 2.6.1. It was used since it is currently the only distribution of Linux which comes with a 2.6 kernel by default. A clock granularity of 1024 Hz was used.

5.2.1.1. Statistical drop

The statistical drop drops a packet based on a percentage. The packet is dropped if a randomly generated number is equal to or below the set drop percentage. This method depends on the random numbers being evenly distributed. The random number generation uses a slightly modified version of the algorithm suggested in [192]. Figure 31 shows the algorithm that was used.

```
if (!randval) randval = jiffies;
randval = ((randval * 1103515245) + 12345) & 0x7fffffff;
accept = drop < (randval % 100);
if (!accept) drop packet;
```

Figure 31 Random number generator used in StAnd Net

In order to test the ability of StAnd Net to drop packets correctly a test was performed. A thousand ping requests were made at each percentage loss level. A thousand ping requests were made at one percent packet loss, a thousand ping requests at two percent packet loss and so on up to one hundred percent packet loss. Figure 32 shows the set packet loss on the x axis versus the measured packet loss on the y axis.

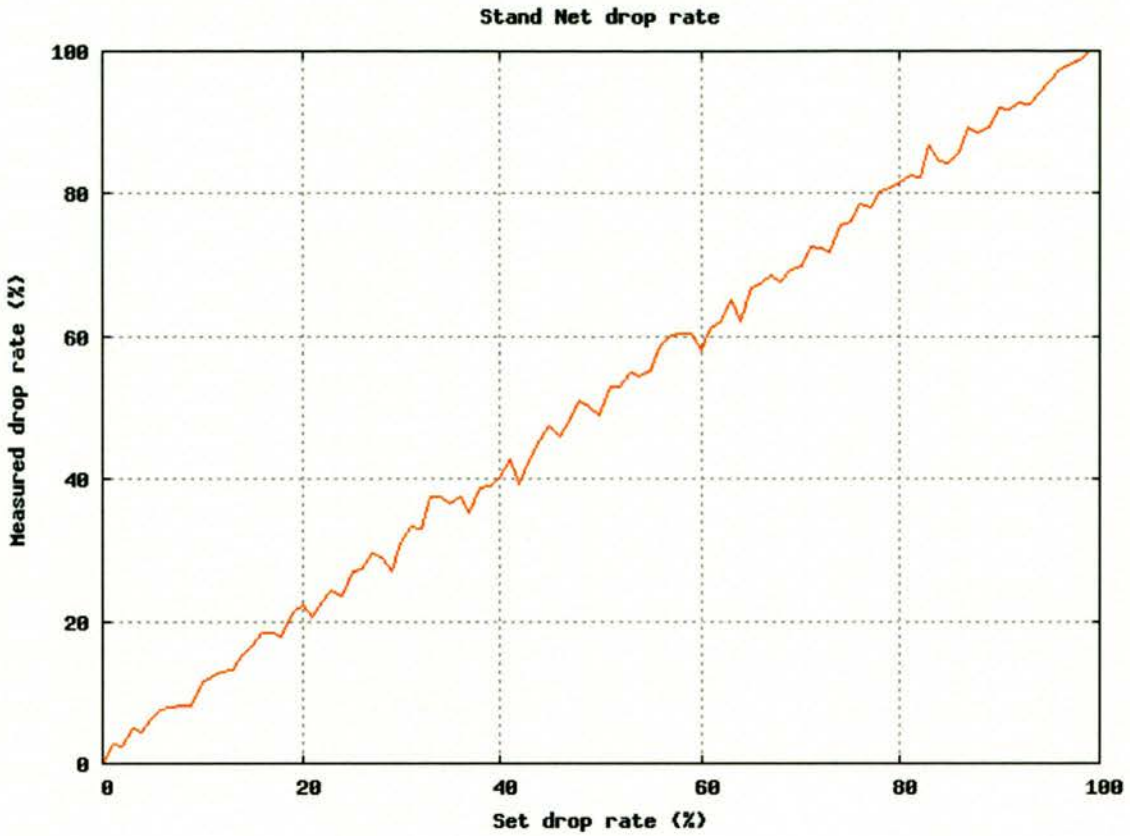


Figure 32 Packet loss results

It was found that there was a tendency for measured drop rate to be below the set rate. The difference was on average 0.43 below for each set percent packet loss. The full results for this experiment are included in appendix C.1.

5.2.1.2. Delay

In order to test the accuracy of the introduced delay a thousand ping requests were performed at each of the delay intervals. So the delay was set to one millisecond and a thousand ping requests performed, the delay was then set to two milliseconds and a thousand ping requests performed. This procedure was carried out up to a delay of two hundred milliseconds. Figure 33 shows the result of plotting the set delay (x axis) against the measured delay (y axis).

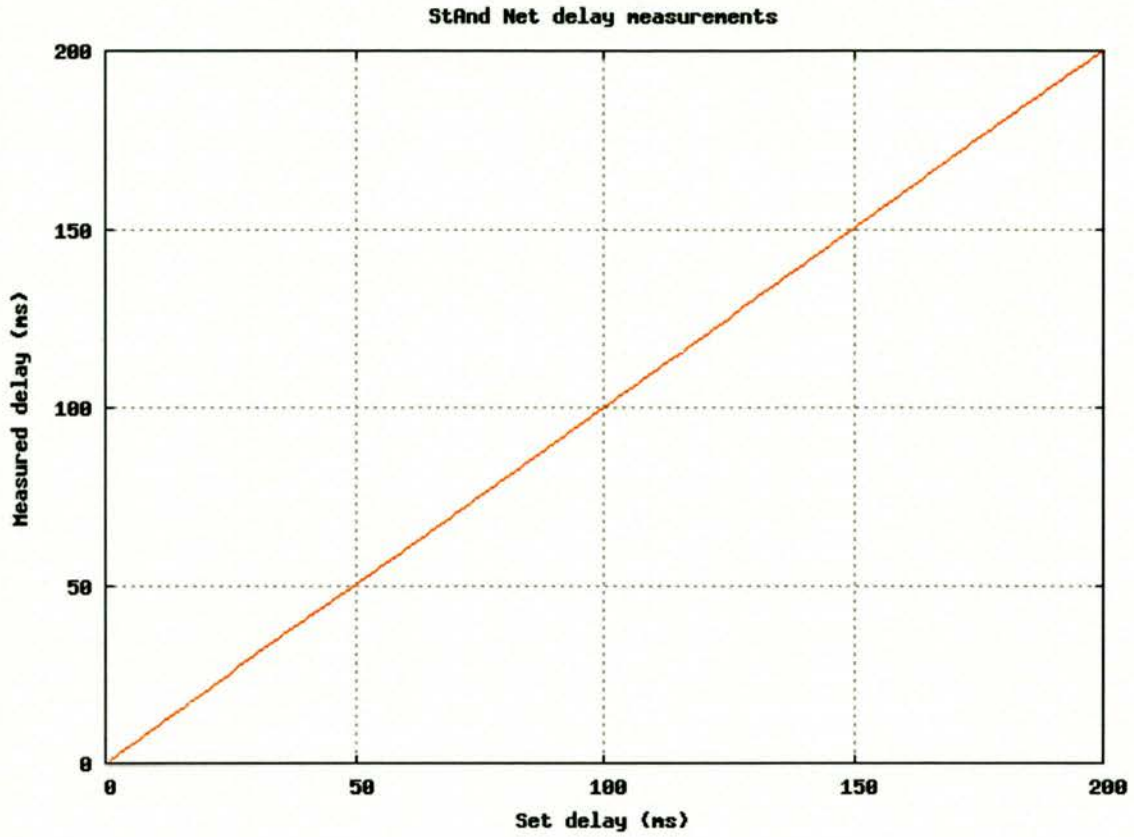


Figure 33 Delay results

On average there is a difference of 0.11 ms between the set delay and the measured delay. This can be attributed to delays in the operating system and on the network. The full results for this experiment are included in appendix C.2.

5.2.1.3. Bandwidth

To test the bandwidth shaping of StAnd Net we configured a number of file transfers from one host to another, using the File Transfer Protocol (FTP). StAnd Net was used to shape the bandwidth between the two hosts. Five network technologies were emulated. A file transfer using FTP was performed 50 times for each of the network types. The network technologies chosen were taken from [193]. The results of the test are shown in Table 4.

Network Type	Minimum kbps	Maximum kbps	Average kbps
100	793.798	806.299	800.895
10	721.126	747.445	740.927
ADSL	13.212	13.403	13.318
Modem	3.160	3.250	3.215
ISDN	14.628	15.014	14.870

Table 4 Summary of network bandwidths

The results for this showed that StAnd Net is able to shape bandwidth. There is a slight difference in the set bandwidth and the measured bandwidths which can be ascribed to protocol overheads. The results for the 100 Mbps when compared to the 10 Mbps network show that the transfer rate above 10 Mbps is not limited by the network but is limited by the application.

5.2.2. StAnd Net overhead

StAnd Net creates an additional delay which is added to any packet which pass through it. Table 5 shows a series of delay measurements taken when using and not using StAnd Net. The mean round trip time delay was taken from ten thousand '*ping*' measurements. The first measurements taken were a base line. This was the delay experienced upon the network path (which was between two hosts on an isolated network, connected via a 100 Mbps Ethernet switch). It showed that a mean delay of 0.152 ms is exhibited by the hardware and software (excluding StAnd Net) which make up the system. Further round trip time measurements where made at various levels of delay (30, 80 and 130). They showed that an additional delay of between 0.327 ms and 0.519 ms. When the base delay is taken into account it is between 0.175 ms and 0.367 ms of delay which is added due to StAnd Net. When using a clock granularity of 1024 Hz it is not possible to compensate for the delay created by StAnd Net

(by reducing the amount of intended delay). This is because delay can only be created as a number of 1/1024 of a second. If a higher jiffy value is used the accuracy increased to a maximum of 1/8192 of a second (122 usec) but this does come with the potential for missing clock tick due if drivers turn off interrupts for more than one clock tick.

Delay set (ms)	Mean delay measured (ms)	Additional delay (ms)
0 (StAnd Net not loaded)	0.152	0.152
30	30.486	0.486
80	80.519	0.519
130	130.327	0.327

Table 5 Measuring delay in StAnd Net

5.2.3. A Comparison of StAnd Net and other emulators

Table 6 shows the capabilities of each of the reviewed network emulators and StAnd Net.

The Linux Kernel Traffic shaper is the most limited as it can only shape bandwidth. However, it is able to do this on a per port basis and therefore apply different limits to different protocols.

The Emulab project provides the basic infrastructure that is needed for network emulation but only at the level of basic operating system support. This means that extra software has to be installed in order to perform the CCA tests, for example the java JDK, JMF or MySQL. The Emulab system is unsuitable if additional hardware is required. Since Emulab is hosted at the University of Utah a fault in the network would give problems running the experiments since there would be no way to use the test bed. Using a locally hosted solution reduces the problems of availability as well as the problems associated with specialist software and hardware being installed within the test bed.

None of the network emulators have explicit support for multicast traffic, while StAnd Net allows net masks to be used in conjunction with the network address allowing any valid IP or range of IPs to be specified.

StAnd Net was the only network emulator to provide a method of introducing jitter into a network path. The only option with the other reviewed emulators is to write a script which periodically changes the delay between the boundaries of the desired jitter emulation.

	NIST Net	**Linux Kernel Traffic Shaper**	**DummyNet**	**Ohio Network Emulator**	**StAnd Net**
Delay	X		X	X	X
Bandwidth	X	X	X	X	X
Congestion	X		X	X	X
Jitter					X
Multipath			X		
Duplication	X				
Per Port		X			
Input/Output	Input	Output	Both	Input	Both
Multicast					
Platform	Linux	Linux	BSD	Solaris	Linux
Drop	DRD [179]	N/A	WF2Q [194]	RED [119]	Statistical & DRD [179]

Table 6 Summary of traffic shaper capabilities

5.2.4. Summary

Using StAnd Net coupled with IP aliasing [195] allows complex network topologies to be created virtually. It is readily installable on modern systems and does not require a recompilation of the kernel. It provides a simple and accurate method of shaping the bandwidth, delay, packet loss and jitter. The API provides a way to control the traffic shaper parameters from separate programs. The interface provided in */proc* allows the leverage of the standard UNIX tools (grep, cat, etc) for reporting the behaviour of the traffic being shaped. The bi-directional traffic shaping allows for edge network deployment and for experimentation without requiring a central node acting as a router.

5.3. Scenario organisation

The *Scenario Organisation* contains two items, a Configuration Script which describes the overall network topology, and, if mobile nodes are specified, one or more maps.

5.3.1.1. Configuration script

The system is setup and controlled by a configuration script. IBM's Bean Scripting Framework (BSF) [196] was used to allow different languages, such as Perl, [197] VBScript [198] and Python [199] to be used to implement the policy. Each configuration script has two global objects which are created by the system, these are (in Perl) $server and $ui. They correspond to the server which is running the emulation and finally an optional user interface, which may be turned off for unattended experiments or for use on 'headless'[18] computers.

Figure 34 is an example configuration script. It creates four nodes. All lines starting with a # are comments. The first two lines import the libraries which implement the SDNE system. The third line creates a new emulation environment. The first parameter gives the length of

[18] A computer which does not have a console attached.

the average sentence in seconds. The second parameter is the length of the emulation, given in seconds. Line four sets the node which will act as a router within the emulated network. Lines five to eight create four nodes. The first parameter (host) gives the name of the host which will be used for this node. The second parameter (type) gives the network type of the host; for example 100 is 100 Mbps ethernet whereas isdn is 128 Kbps ISDN. Lines nine to twelve add the previously created nodes to the network. Finally line thirteen starts the network emulation.

```
1)  use EmuNetwork;
2)  use EmuNode;

# Each sentence slot is 2.5 seconds, the emulation will
# last for 30 minutes
3)  my $network = new EmuNetwork (2.5, 60 * 30);

# Set the router of the system
4)  $network->router ("nog31");

# create four nodes
5)  my $node1 = new EmuNode ('host' => 'nog32',
                             'type' => '100',
                             'talkative' => '25');
6)  my $node2 = new EmuNode ('host' => 'nog33',
                             'type' => '10',
                             'talkative' => '25');
7)  my $node3 = new EmuNode ('host' => 'nog34',
                             'type' => 'isdn',
                             'talkative' => '25');
8)  my $node4 = new EmuNode ('host' => 'nog35',
                             'type' => 'modem',
                             'talkative' => '25');

# Add the nodes to the network
9)  $network->add ($node1);
10) $network->add ($node2);
11) $network->add ($node3);
12) $network->add ($node4);

# Create the network
13) $network->run ();
```

Figure 34 Example configuration script

5.3.2. Emulation manager

The Emulation Manager consists of a Node Controller and as many Emulated Nodes as there are nodes within the emulation.

5.3.2.1. Node controller

The *Node Controller* signals all the *Emulated Nodes* at specified time intervals that they are to update their network

5.3.2.2. Emulated node

An *Emulated Node* provides a representation of a node within the system. Each *Emulated Node* mirrors the settings on the actual physical node, which is controlled by the *Node Controller.*

5.3.3. Local node controller

There is one *Node Controller* per *Emulated Node.* It is a process which runs on a host within the Beowulf cluster. The *Node Controller* is connected to an *Emulated Node* within the *Emulation Manager* via a standard TCP network connection, through which it receives its configuration instructions. The *Node Controller* takes a host computer which only has a wired network and turns it into a wireless node within the emulated network. The *Node Controller* is composed of three subcomponents; the *Local Node Controller* and a *Traffic Shaper.*

5.3.3.1. Local node controller

The *Local Node Controller* is a client process which runs on a Beowulf node. It receives commands from the *Node Controller,* relating to which nodes the local node is able communicate with, and then alters the *Traffic Shaper* and *Virtual Wireless Network* parameters so that the network conditions experienced by the local node are representative of the emulation.

5.3.3.2. Traffic shaper

The final component is the traffic shaper. This performs the shaping of the network traffic so that it conforms to the parameters which were set in the configuration file.

5.4. Configuring a CCA test bed

The underlying physical network and nodes are shown in Figure 35. The cluster consists of 64 PII 450 MHz machines connected on a private 100 Mbps Ethernet network. The machine at the head of the cluster, which acts as a router to the rest of the network as well as a file server, is called noggin. Each of the nodes within the cluster is called a nog and have

hostnames from nog10 to nog74. Nog74 is unique in the cluster since it contains the installation image for the rest of the cluster. Only 9 machines were needed for this test bed, nog31 – nog39, which were running the latest version of the RedHat version 7.1.

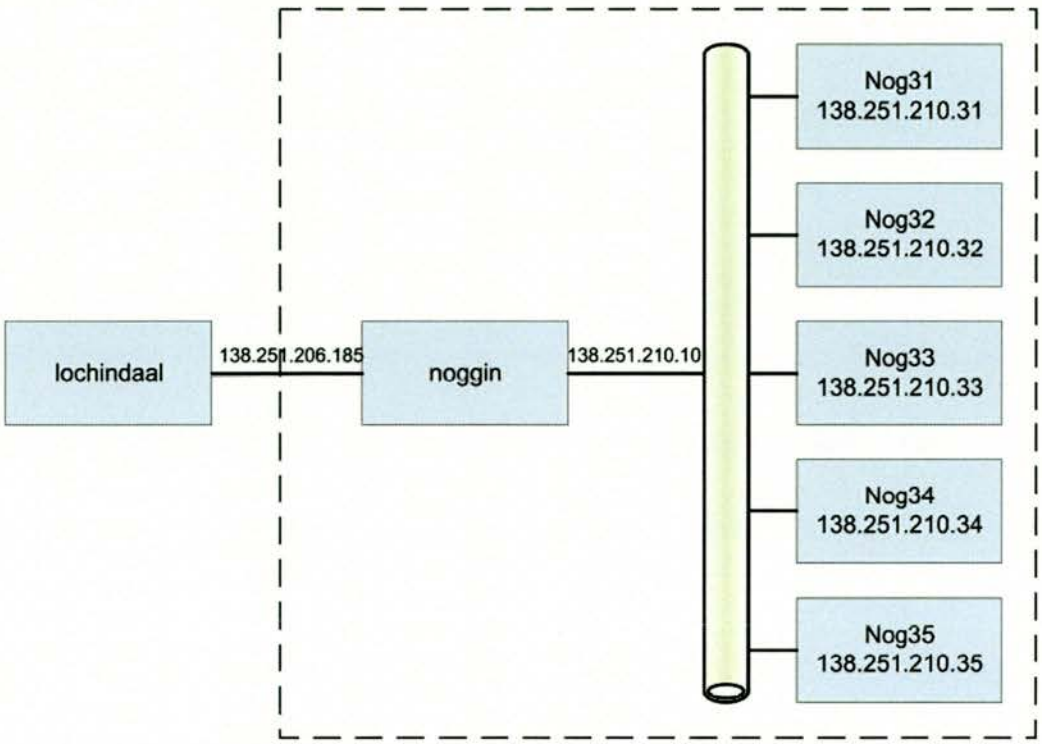


Figure 35 The physical organisation of the test bed

One machine which is not part of the cluster, lochindaal[19], was used for the testing. It hosted the Conference Controller Architecture.

The virtual network, shown in Figure 36 used as the test bed, is quite different from the physical network.

[19] Lochindaal.dcs.st-and.ac.uk (lochindaal) is the author's desktop computer and is a PIII 800 Mhz running Redhat Linux version 9.0.

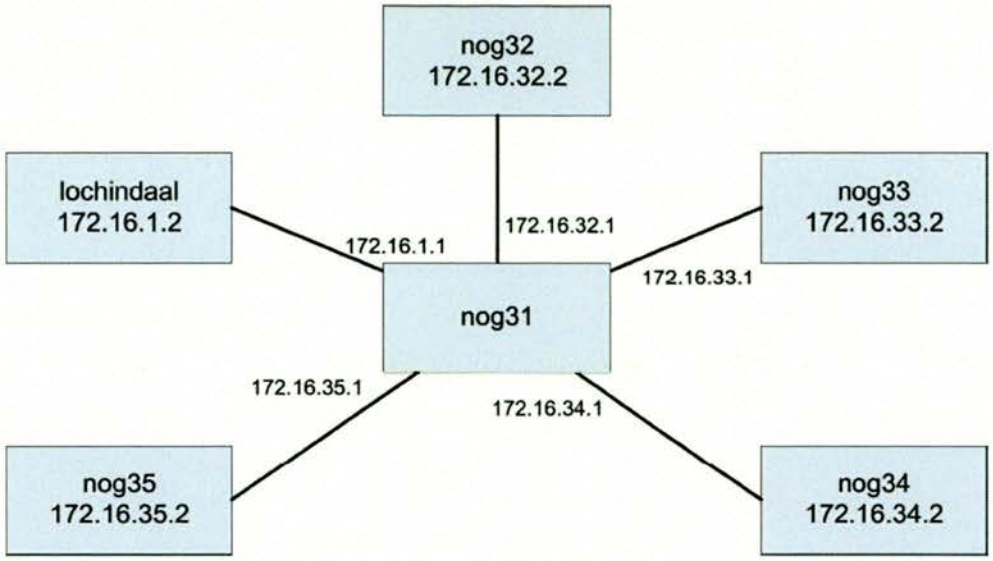


Figure 36 The virtual arrangement of the test bed

In the example presented in Figure 36 there are five separate networks, with each of nog32 to nog35 placed on a separate virtual network and finally one which connects lochindaal to the cluster. Nog31 is on all of the networks and acts as a router not only for the unicast traffic but for the multicast traffic as well.

Multicast traffic is unusual in that the destination IP address of the packet does not match the receiving host since it goes to all receivers in the multicast group. Therefore, 172.16.1.2 -> 224.1.1.1 may be a route to many hosts so it is not possible to just shape multicast network traffic in one location. In order to solve this problem, all routes must be virtually split in two. First there is the route from the sender to a central point. This central point is nog31 in this test bed which connects all networks. This emulates the network connection which is used by the sender to connect to the Internet. Finally, there is the route from the central router to the receiver (or receivers). There is one route (and therefore set of traffic shaping) for each node in the network. Overall, this corresponds to the sender uploading the media stream while the receiver downloads it. The traffic has to be shaped in two separate actions in order to

emulate the path from a sender to a receiver. Using a central router will not work (the separate routes taken for multicast traffic is not discernable in a single machine); instead each node in the network is responsible for shaping its own network traffic.

The network devices used on the test network are all virtual network devices and use the same physical network device (eth0) as the node's normal network address. The virtual devices have a name in the form <physical device>:<virtual device number> so the first virtual device on eth0 is called eth0:1.

The Participant Agents communicate using multicast IP and as such it was necessary to use multicast router software. Mrouted [200] was chosen since it is standard and in common use.

Lochindaal is separated from the nogs by noggin and therefore it is unable to be part of the multicast session directly. Lochindaal could be made to be part of the multicast network in various ways. The first way to achieve this is by running a multicast router on noggin but this would have the by-product of adding all the machines on the 138.251.206.0/255.255.255.0 to the multicast group and since this network is part of the Mbone it will make the virtual network part of the Mbone. This is undesirable since we could not be sure of the traffic on the network and so the expected results could not be correctly assessed. It is also possible that the multicast traffic will degrade network performance for other users. The second method would be to have a multicast tunnel from nog31 to lochindaal. This could be achieved by installing the *mrouted* package on lochindaal and create a multicast tunnel to nog31, but unfortunately this has the side affect of adding all of lochindaal's network and therefore the Mbone to the test bed. The solution to this problem was to use mTunnel [201], a multicast tunnel application that only tunnels specific multicast groups.

There are several restrictions that this set-up enforces. The first is that since we share the physical network between not only our virtual networks but also the other machines that are connected to the network we are limited in the number of networks and the amount of traffic we can generate and emulate due to the physical limits of the hardware. For example since we only have a physical 100 Mbps network we could not reliably emulate 2 100 Mbps networks. The second issue is that since the physical network is broadcast all packets on it are ordered due to the fact that only one computer can send a packet at a time on the network. This has the effect that all packets on the virtual network will be appear in the same order on all the networks. This does not matter for this test bed.

5.4.1. Test bed configuration tool

As part of the test bed a configuration tool was created. This allows the easy configuration of emulated networks as well as the amount of RTP traffic that is going through the network. In order to configure the network a script is written which informs the tool how to configure the network. The configuration scripts take a similar form to the TCL scripts that are used to control the Network Simulator (NS) [202]. There are some important distinctions. The main differences, the syntax aside, is that in the NS configuration script the IP addresses that the individual nodes will use must be supplied. In the SDNE system the virtual IP address of the node is determined from the host name of the actual node that is being used. For example, if nog34 is being used within the emulated network then the virtual IP address it will be assigned will be 172.16.34.2. The first 2 bytes, 172.16 are static and are used by all nodes within the test bed. The third byte is determined from the hostname and the final byte can have two possible values; 1 meaning that this node is acting as a router for the network and 2 which indicates that the node is acting as a leaf node on the network.

An example configuration script is shown in Figure 37

```
use EmuNetwork;
use EmuNode;

# Each sentence slot is 7 seconds, the emulation will
# last for 3600 seconds
my $network = new EnuNetwork (7, 3600);

# Set the router of the system
$network->router ("nog31");

# Create four nodes, assign them to actual hosts and
# set their network type; 100, 10, isdn and modem
# Each node of the network has a 25% share of talk time.
my $node1 = new EmuNode ('host' => 'nog32',
                  'type' => '100',
                   'talkative' => '25');
my $node2 = new EmuNode ('host' => 'nog33',
                  'type' => '10',
                  'talkative' => '25');
my $node3 = new EmuNode ('host' => 'nog34',
                  'type' => 'isdn',
                  'talkative' => '25');
my $node4 = new EmuNode ('host' => 'nog35',
                  'type' => 'modem',
                  'talkative' => '25');

# Add the nodes to the network
$network->add ($node1);
$network->add ($node2);
$network->add ($node3);
$network->add ($node4);

# Create the network
$network->run ();
```

Figure 37 Example Test bed configuration script

The script shown in Figure 37 creates four nodes and uses one router, each of the nodes, nog32, nog33, nog34 and nog35 are assigned a network type of 100 Mbps switched Ethernet, 10 Mbps switched Ethernet, ISDN and 34 Kbps modem respectively. As they are added to the network the percentage of time that each node transmits useful information is configured. This number relates to when the simulated speaker is talking. The simulated speech may not map directly onto the Participant Agent sending multimedia traffic, for example an always on floor control policy could be in use where audio and video is always transmitted. The percentage talk time allows the system to use silence suppression without

the need of having a live audio feed. The audio can be pre-recorded and played back as and when needed.

In order to emulate people speaking a number of time slots are created and the emulated speakers are allocated to a number of slots. The number of time slots they are allocated is dependent on the talkative parameter when the network node was created. When the time index of the session is within a slot's time index then the emulated person who owns that slot is speaking. The allocation of nodes to time slots is performed in a pseudo random fashion. An array is created that is the number of time slots long. So for example if the session were to last 20 seconds and a time slot was 2 seconds then the array would be 10 elements long. Then as each node is added to the network (with the *$network->add ()* method) it is allocated the requested percentage of the overall time slots and added to the array. When all the nodes have been added to the array the system chooses two elements within the array at random and swaps them over within the array. This is performed anywhere between one million and two million times (the exact number of times is chosen at random) to ensure that the speech pattern of each emulated user is random. Of course; the percentage of talk time that each node gets is still maintained within the system.

In order to inform the nodes on the network when the emulated person is talking and when they are not talking a central server sends out instructions over the network informing each node if it has speech to transmit. This process takes place on the physical network and is kept separate from the virtual emulated network which is under study so that the instructions do not interfere with the tests. As each client starts it makes a TCP network connection to the Controller it is registered with and the connection kept open for later use. Once all the nodes that will be transmitting have registered themselves the Controller loads the controller event list file, which was generated by the configuration script. This file states when each node

should start and stop transmitting. The Controller waits one minute after the last node has connected before it starts issuing instructions to start and stop talking. This pause allows each node to start the transmission processor. This includes acquiring a data source for the test data as well as starting the transmission. After 45 seconds the transmission is stopped from within the client in readiness to accept the instructions from the Controller. Following another 15 seconds the first node to transmit is given the instruction to start.

This central model of control was adopted since it avoided the problem associated with each node being in control of when it should transmit. These problems stem from the difficulty in maintaining a consistent view of time within a distributed system. There is a simple way to ensure that each node starts and end its transmissions in a way which is synchronised with the other nodes within the system. Each node will have a slightly different view of the current time therefore the control of transmission could not be left up to the individual nodes as transmissions could overlap in a way that would be undesirable to the system or in fact transmission would not happen within the timeframe of the experiment (for example is a node's view of the current time is widely different from the rest of the nodes within the test bed). By default, the controller process is run from the machine acting as the router for the network.

5.4.1.1. Testing the configuration tool

In order to test the test bed configuration tool various sessions were configured. These varied the number of nodes as well as varying the talk times for the node. The first of these tests involved two nodes each having a talk time of 50%. The script that was used to create this configuration is shown in Figure 38.

```
use EmuNetwork;
use EmuNode;

# Each sentence slot is 2.5 seconds, the emulation will
# last for 1800 seconds
my $network = new EnuNetwork (2.5, 1800);
# Set the router of the system
$network->router ("nog31");

# Create four nodes, assign them to actual hosts and
# set their network type; 100, 10
# Each node of the network has a 50% share of talk time.
my $node1 = new EmuNode ('host' => 'nog32',
                'type' => '100',
                'talkative' => '50');
my $node2 = new EmuNode ('host' => 'nog33',
                'type' => '10',
                'talkative' => '50');

# Add the nodes to the network
$network->add ($node1);
$network->add ($node2);

# Create the network
$network->run ();
```

Figure 38 Configuration script, 2 nodes 50% talk time

The nodes within the emulated network were configured so that they emulated a 100 Mbps and a 10 Mbps network so that the traffic shaping did not interfere with the measurements being taken. The test lasted one hour and a sentence was assumed to last 2.5 seconds [203]. It is possible that several sentences will be spoken by a single person before they stop talking. The bandwidth used during the session was recorded. Figure 39 shows the bandwidth usage for the entire hour. Figure 39 shows the swapping of the node which is currently transmitting, while Figure 40 shows one minute's worth of time taken from the beginning of the results. This exploded view of the graph more clearly shows the nodes swapping who has the floor and therefore who is transmitting during the session. The swap over points are not straight vertical lines as may have been expected but instead show the gradual increase and decrease that is typical from averaging the bandwidth used over a period of time. The

switching as one user stops talking and the other starts talking can be clearly seen. With the first test each node was to speak for 50% of the time, the measured results show a different picture, each node was transmitting for around 54% of the total time; with nog32 and nog33 transmitting for 980 (16 minutes) and 960 (16:20 minutes) seconds respectively. This is to be expected as the video transmission system does not respond immediately to start and stop calls and in fact can take a second or two in order to fully stop or fully start a transmission. These overlaps are exactly what we would see with real people talking not only since the software would have the same start-up and shutdown lead times as when using automated events but it is not unusual for two people during a conversation to end up talking at once; this is quickly resolved as one person backs off to allow the other to speak. In the videoconferencing domain this problem is exaggerated since some of the normal personal cues are absent.

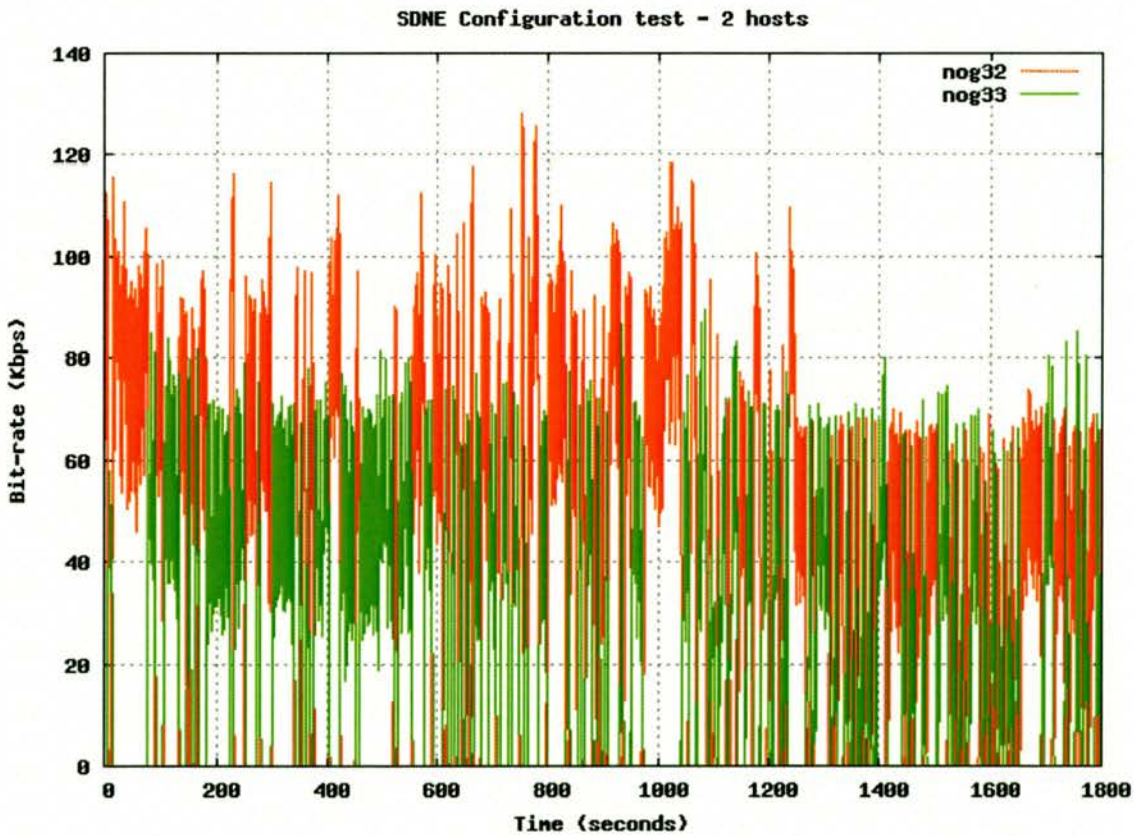


Figure 39 Bandwidth use for test session

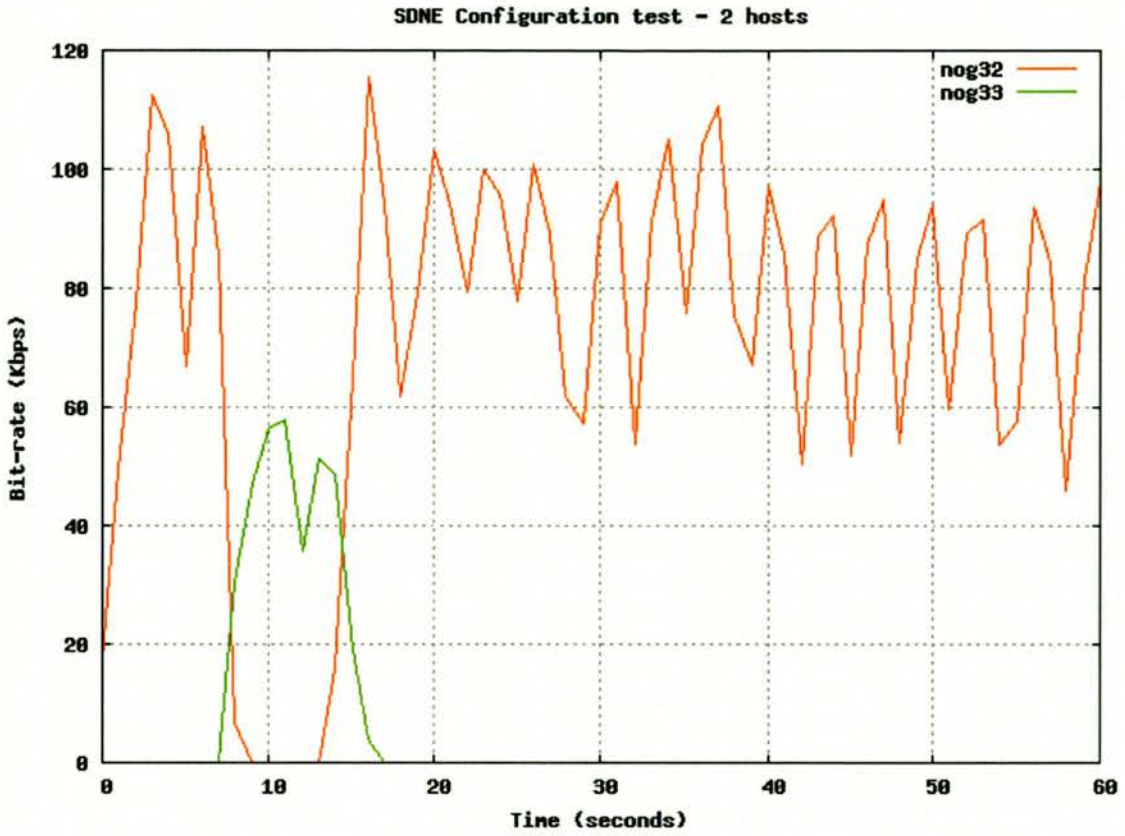


Figure 40 Exploded view of configuration test

This test was rerun using an emulated five host network in order to check that the controller system is able to keep up with controlling more realistic numbers of hosts and in order to test whether the percentage of overlap that is experienced within a larger group is similar to that of the 2 host system.

The network was configured as 5 hosts each connected via 100 Mbps network. The talk time allocated to each node was 20 percent of the total available talk time. The experiment was configured so that the experiment lasted one hour. Figure 41 shows the graph of the resulting bandwidth usages in Kbps over the time in seconds. The amount of time used was 24.61%, 23.25%, 24.75%, 23.27% and 24.39% of the total available time for each of nog32 to nog36 respectively. They are close to the requested settings of 20% per node, but consistently

around 4% over the set talk-time. Figure 42 shows the bandwidths in Kbps over the time in seconds. The overlapping segments of the graph account for the increase in the talk time over the requested talk time. These are created because the averaging process which is used to measure the bit-rate takes some time to register the bit-rate as zero and during that time a second node has already started transmitting.

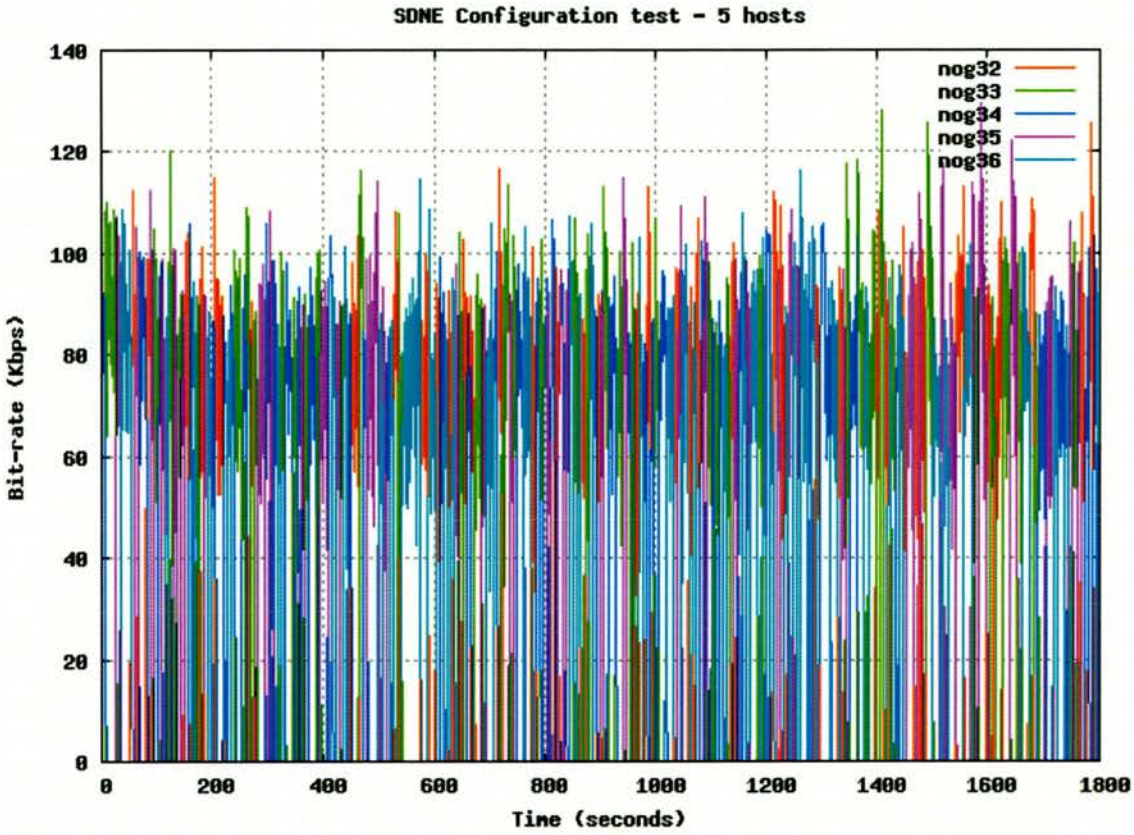


Figure 41 5 Host emulated network bandwidth usage

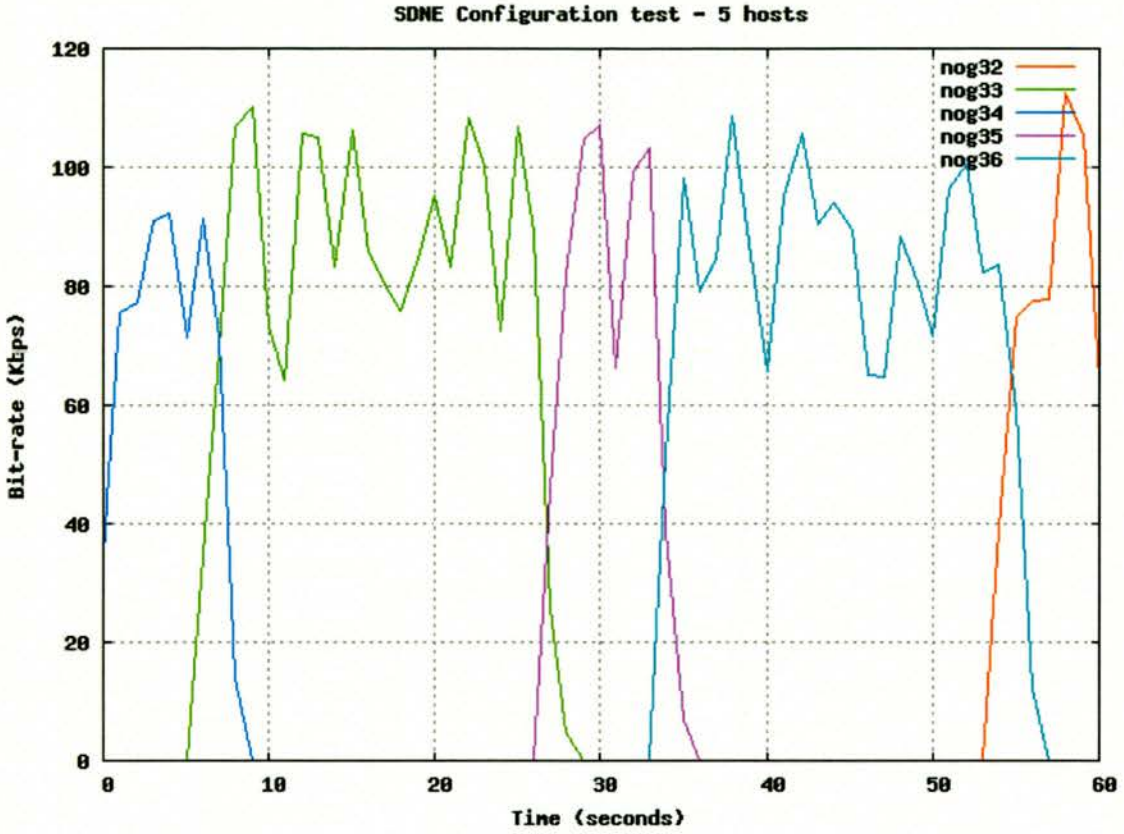


Figure 42 First 60 seconds of 5 Host emulated network bandwidth usage

5.5. Conclusion

This chapter has described the design and implementation of a scenario driven network emulation system, where virtual network topologies and virtual users can be created to aid the performance of implementation, testing and evaluation of multi-user networked applications. The flexibility it provides allows a wide variety of network topologies to be created with the behaviour of people also being factored into the emulation.

StAnd Net provides traffic shaping in order to manipulate the bandwidth, delay, packet loss and jitter.

Chapter 6

Conference Controller Architecture Evaluation

6. Conference controller architecture evaluation

This chapter describes the evaluation of the Conference Controller Architecture. The goal of the study is to evaluate the Conference Controller Architecture when used to configure and manage videoconferencing sessions between small numbers of users.

The study aims to test the CCA as designed and implemented; to determine if video conferences which are configured by the CCA are initialised in an acceptable amount of time; and to find if the configuration mechanism adheres to the principle suggested by Bouch and Hands findings that consistency of quality is more important than absolute quality, while utilising the network resources effectively.

The key components under study are:

1. Session configuration time
2. Initial session configuration
3. Conference maintenance

The session configuration time is an evaluation factor in terms of scalability and transparency. This is expressed as the amount of time it takes from the command being issued to start the configuration until a session configuration is available to be sent to the participant agents. Unfortunately, there are no hard and fast bounds which can be placed on this. The configuration time should be low enough so that the user is unable to notice it (transparency). Larger configuration times can be tolerated for longer than might be expected since at the beginning of a videoconference (or a meeting in general) there is normally a small amount of time "wasted" while people join. Ad-hoc meetings normally involve low numbers of participants and are organised quickly so they should be configured quickly.

The configuration mechanism is the part of the policy which sends out the first configuration information for the conference. This is the best estimate that the CCA has been able to make based on the previous information it has gathered for the network paths. If this configuration information is accurate there should be no need for the CCA to reconfigure the conference settings during the course of the conference. A potential problem is that the CCA will be too conservative and provide a configuration which is well below the stream's fair share of the available network resources. This will mean that any conference which is hosted by the CCA will not have as high an audio/visual quality as is possible within the limits of performing within a TCP friendly manner.

The testing goes on to evaluate how well the CCA is able to maintain a conference throughout its lifetime. After the conference has started consistency is the main issue. Fluctuations in the audio or visual quality will be quickly noticed and lead to a lowered experience. Therefore alterations after the initial configuration should be avoided if possible. Reconfigurations of the conference parameters are necessary in the following situations:

1. available bandwidth has dropped below the initial estimation

2. the round trip time has increased beyond the initial estimation

3. the measured packet loss has exceeded the maximum acceptable level for the packet loss scheme in use

4. the jitter has exceeded the maximum acceptable level and a reconfiguration of smoothing buffers and CODECs would help

The second and third situations would be coupled with the first since the available bandwidth calculation is limited by packet loss and round trip time. In the case of the first condition, a

drop in bandwidth, a new sender rate must be chosen. The last three can be solved by re-choosing encoding parameters given the new network information. In order to maintain consistency large variations in the encoding bit-rate are to be avoided. This includes increasing bandwidth usage during a conference.

6.1. Experimental design

These experiments were performed using the SDNE system described in chapter 5. The SDNE system was configured with a virtual topology as shown in Figure 43. A multicast router was run on nog31 in order to connect all the networks together. Lochindaal hosted the Conference Controller and the Traffic Data Repository. It was connected to the test environment via a multicast tunnel due to the physical configuration of the local network. Each of the nog nodes was allocated a separate virtual network in the 172.16.0.0/255.255.0.0 IP address range.

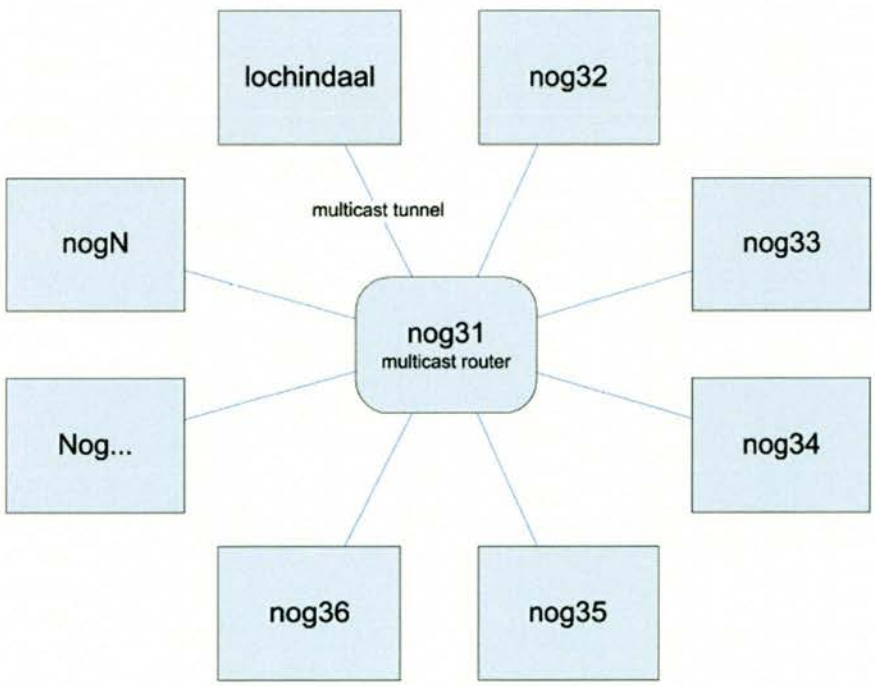


Figure 43 SDNE configuration for experimentation

Different scenarios were loaded into the SDNE system to emulate different multimedia conferencing scenarios.

The RTP traffic which was used for the videoconference came from a series of pre-recorded sequences of audio and video which had been recorded at a high quality and then transcoded within the participant agents to match the conference session parameters which had been suggested by the Conference Controller. The audio/video stream was stored on each node in order to minimise network traffic. Figure 44 shows an example frame from the test video.

Figure 44 Example frame from the test video

Real-time UDP emitter (RUDE) [204], which was developed as part of the Faster 2000 [205] project was used to create background traffic with the test environment. RUDE is a replacement for the MGEN [206] tool. MGEN can only generate packets every 10 ms, but RUDE does not have this limitation and is therefore able to generate packets at a more fine grained resolution. This gives a more realistic traffic trace as more frequent packets can be generated which is similar to a background traffic level of a network link.

6.2. Results

The results of the experiments and a discussion of those results are presented below. There are two aspects to the testing of the CCA. Firstly; profiling of the system in order to evaluate how quickly it is able to configure a conference. Secondly; the suggested configurations provided by the CCA are evaluated. This aspect has four stages, the CCA with a constant loaded network, the CCA with an increasing amount of background traffic, the CCA with a decreasing amount of background traffic and finally the CCAs reaction to a network path upgrade.

In each case an account of the experiment is presented and followed by the results and a discussion of the implications of those results.

6.2.1. Initial configuration timings

These test the time it takes to configure a session. This is the time from the Conference Controller being told to configure a session to the XML formatted configuration information being ready to be sent out to a client. This is split into several stages. There is a database stage where data from past sessions is retrieved. There is the transfer of this data from the TDR to the CC, which can be performed via the network. Then there is the time taken to reformat the data for use in the policy. Finally there is the execution time of the policy. At this point the CCA is ready to inform the PAs of the configuration.

This experiment was performed by repeatedly instructing the CCA to configure sessions. This was performed 20 times. The start and end were marked by time stamps. The emulated videoconferences were allowed to play-out so that additional data could be added to the TDR and this could be taken into account when looking at the timings for the next configuration.

A simple policy was chosen which reads all the data from the database for a given set of network points and averages the bandwidths, delays, jitter and packet loss for each path. The lowest bandwidth, and highest delay, jitter and packet loss were then taken as the prevailing network measurements. The configuration process has the following structure, with timestamps taken at each point.

- Receive list of network points
- Start data transfer from the TDR
- End data transfer from the TDR
- Start policy script
- End policy script

The start of the evaluation time was time stamped when the CC received a list of all the network points that were to be involved in the conference. The end point was time stamped by the CC sending out the initial videoconference configuration parameters.

It was found that on average, it took just over 332 ms to retrieve the data for a single participant per 30 minute period. A likely number of network traces being used per participant is 12, this would be the previous 6 days, e.g. a conference on Monday would use data for Sunday, Saturday, Friday, Thursday, Wednesday and Tuesday, and the previous 6 days from the previous 6 weeks, for example each Monday from the previous 6 weeks. Six was chosen since this is long enough that most holidays, where people are not using the network as heavily are contained within this period meaning that data from before the holiday would still be available.

This timing information shows that for a low number of participants, up to 6, the system is able to configure a conference within 12 seconds assuming the data from 6 previous sessions. Thus small conferences of a hand full of people (up to 6) can be configured ad-hoc and without prior planning, but there will be a slight delay at the beginning of the conference while the configuration is created. With 30 participants and a history of six previous videoconferences the configuration time is just under one minute.

After the performance analysis tests were carried out it became obvious that the original architecture suffered from a performance bottleneck, which hampered its ability to initially configure a multimedia conferencing session in a timely fashion. The bottleneck is the transferring of data between the CC and the TDR. The TDR and the CC are currently two distinct components, which run as separate processes and communicate via CORBA [207]. This was done so that there could be one TDR for many CCs, allowing greater sharing of network traffic information. Profiling the system found that the network transfers took between 87.49% and 88.53% of the time with the average being 87.81%. Removing the separation between the CC and TDR would remove this overhead, but the advantage of being able to have multiple CCs which can share the network data from a single TDR is lost. The removal of this overhead takes the configuration time for a 6 person conference down to just under 1.5 seconds but with the loss of shared information.

6.2.2. Session configuration and maintenance

For these tests, where a test consisted of several videoconferences, the TDR database was wiped, then a series of videoconferences were configured and allowed to run.

These tests are designed to show the performance of the CCA and its ability to configure and maintain that configuration throughout the lifetime of a conference. There are four aspects which are of interest. First there is the behaviour of the CCA when exposed to constant

background traffic, next there is the case when the available bandwidth on a path has increased, this could be due to an upgrade in the network infrastructure. Although it is unlikely that the actual physical network fabric which is being used would be upgraded, a routing change could result in an increase in network capabilities. Next is the case of increasing and decreasing background traffic. As previously shown network traffic patterns follow a cyclic pattern of high daytime and low evening and night traffic. These patterns create an increasing, reasonably constant and decreasing load on the network.

The bandwidths reported are the bit-rate suggestions provided by the policy within the Conference Controller Architecture. In each case a discussion of the example conference is given.

6.2.2.1. Constant loaded network

The testing for the constant load upon the network used the RUDE traffic generator in order to generate background UDP traffic at various levels. The example given uses RUDE generating a consistent level of background traffic throughout the conferences. The CCA initially had no information about the network paths to be used and was allowed to build up its knowledge during the subsequent conferences. The configuration files for RUDE, the CCA, as well as the policy in use for the experiment, are included in Appendix D.1.

An example run is described below. For the run described a constant load of 30 Kbps was generated. The bandwidth limiting technology was ISDN with a bi-directional bandwidth limit of 128 Kbps. There were 6 participants in the conference and a meeting was emulated.

The suggested bit-rate, was at the maximum 75.75 Kbps and the minimum was 74.85 Kbps, the mean was 75.31 Kbps with a standard deviation of 0.29.

Figure 45 presents the suggested bit-rate traces for the 9 videoconferences within this example.

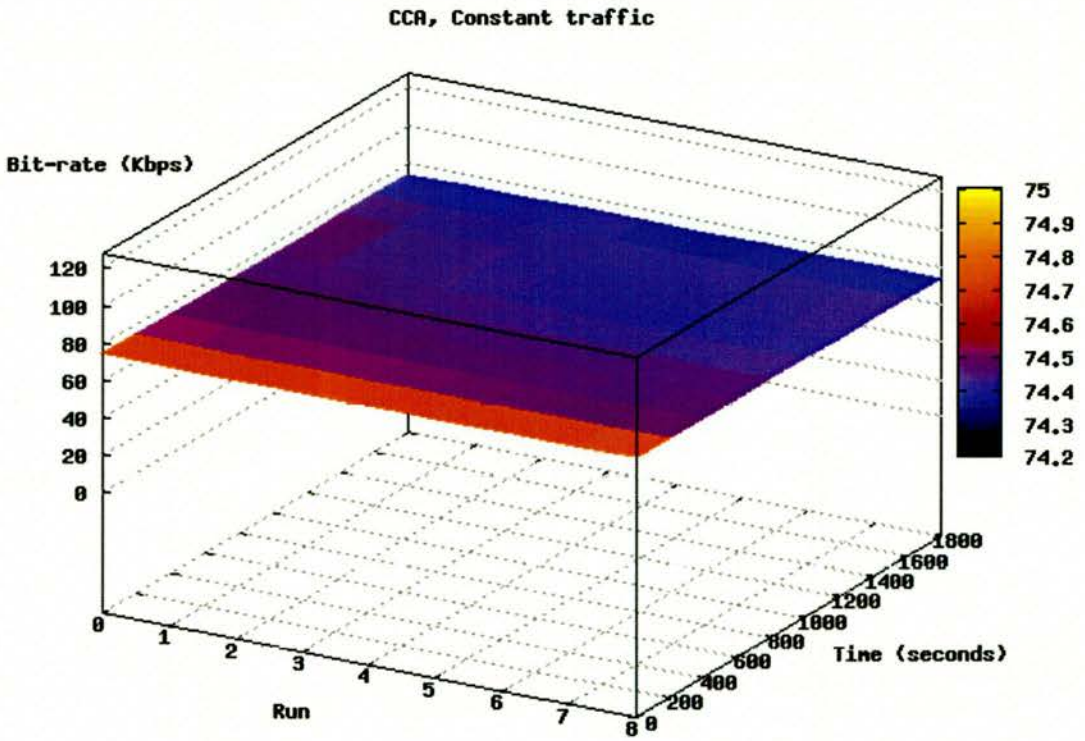


Figure 45 Contoured surface, constant loaded traffic

Run number	High bit-rate (Kbps)	Low bit-rate (Kbps)	Mean bit-rate (Kbps)	Standard deviation
0	75.83	71.49	74.88	0.46
1	75.64	73.79	74.46	0.23
2	75.80	73.08	74.50	0.40
3	76.55	71.81	74.60	0.67
4	79.35	70.23	74.30	1.15
5	75.85	70.021	74.45	0.46
6	76.45	72.25	74.20	0.76
7	79.57	73.09	74.45	0.44
8	75.71	74.29	74.60	0.19

Table 7 Summary for the background traffic loaded network

The first session has an average suggested bit-rate of 74.88 Kbps with the highest suggested bit-rate of 75.83 Kbps and the lowest being 71.49 Kbps and having a standard deviation of 0.46. Table 7 shows a summary of these measurements across all the runs in this example.

The results show that the CCA is able to configure a conference when the background traffic is constant. The configuration of the conference parameters varies very little during the course of the conferences shown. In this scenario the CCA provides no benefit over a purely TCP friendly equation based system as both are ruled by the maximum sending rate which a TCP friendly equation provides. Of course, network traffic does not remain constant and ebbs and flows of background traffic are to be expected.

6.2.2.2. Changing load network

This section of the testing examines the CCA's ability to configure videoconferences when the network load is changing. There are two ways in which the network traffic could be

changing, increasing load and decreasing load, a varying load which has no general trend other than it is reasonably static is included for completeness. Additionally, the situation where the network is upgraded is investigated. This would occur due to a change in physical fabric of the network which is being used to transport the packet end-to-end. For example a routing change may occur which means that packets now traverse a route which had a much higher bandwidth.

6.2.2.2.1. Increasing load

This series of experiments was performed in order to test the CCA's ability to configure conferences when the load upon the network was increasing. This used RUDE to generate an increasing load during the video conference. The sessions were emulated so that they appeared at the same time each day and would therefore have to endure the same network traffic conditions. In the example detailed the background network load was increased from 0 Kbps to 30 Kbps over the space of 30 minutes. The configuration files for RUDE, the CCA, as well as the policy in use for the experiment, are include in Appendix D.2

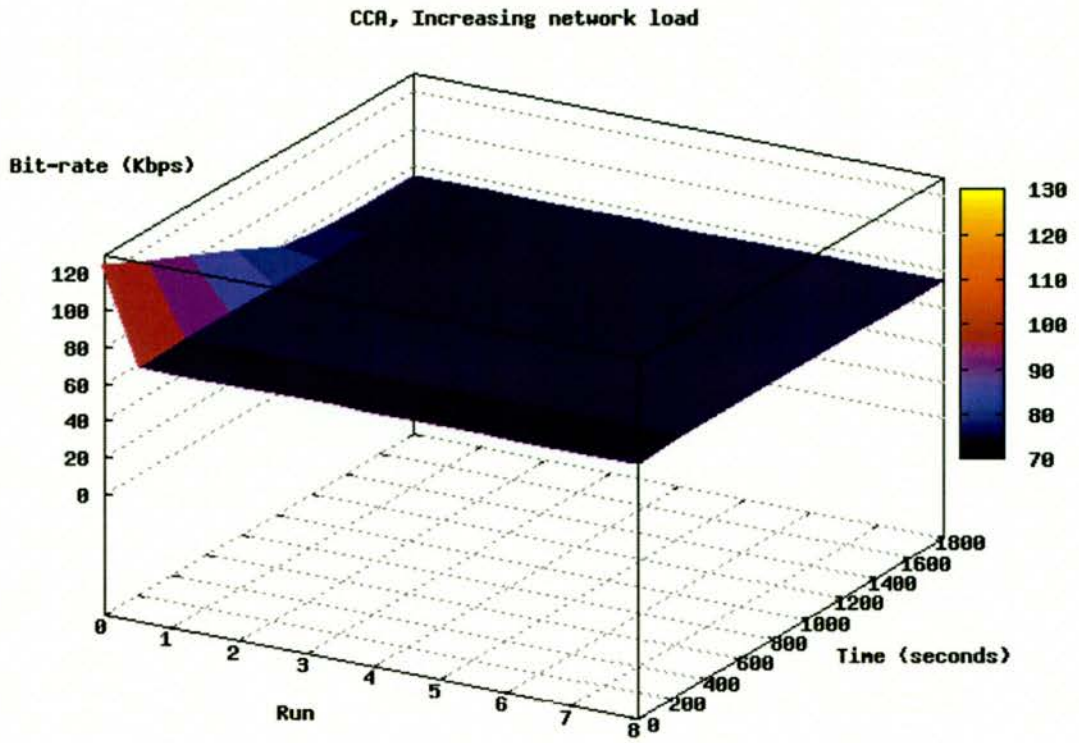


Figure 46 Increasing network load

At the beginning of the conference there is nothing known about the network therefore the bandwidth measurement initialised the conference at the highest point possible at that time. As the network load increases the CCA backs off its bandwidth usage. When the background network load levels off the CCA reaches a steady state and little change to the transmission bit-rate by the PAs is made. The interesting aspect of the configuration comes at the second run. Due to the previous network trace the CCA assumes that the network will behave similarly therefore it configures the conference for the quality it was able to maintain during the previous run. The trend continues over the runs for the subsequent conferences.

This shows a marked difference when compared to a purely equation based tracker system which would follow a smoothed variation of the available bandwidth and could create a large variation in the audio/visual quality of the conference.

6.2.2.2.2. Decreasing load

This series of experiments was performed in order to test the CCA's ability to configure conferences when the load upon the network was decreasing. This used RUDE to generate a decreasing network traffic load during the video conference. The sessions were emulated so that they appeared at the same time each day and would therefore have to endure the same network traffic conditions. In the example given the network load goes from 30 Kbps to 0 Kbps over the entire 30 minutes of the conference. The configuration files for RUDE, the CCA, as well as the policy in use for the experiment, are include in Appendix D.3

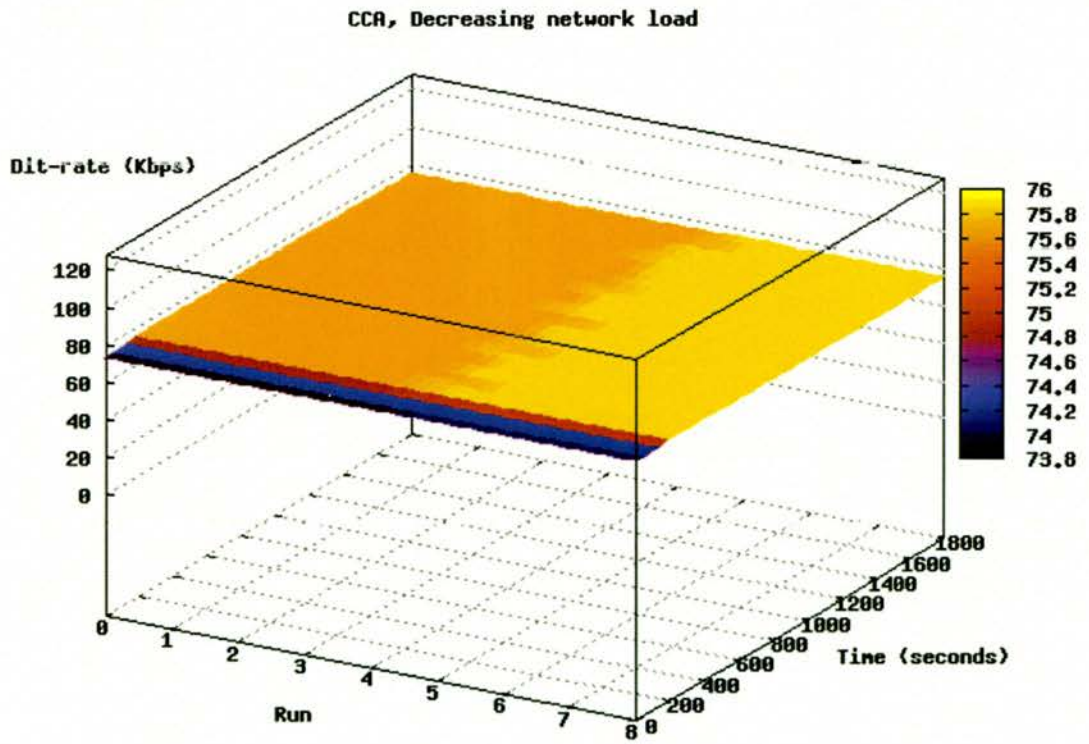


Figure 47 Decreasing network load

Figure 47 presents an overview of the data received from the experiment. At run five there is a slight increase in the maintenance configuration bandwidth, on the order of 0.2 Kbps. These small fluctuations would not degrade the quality of the conference since they are maintained in a stable state and they are very small being on the order of 1 Kbps at the maximum. Network traffic produced within a video conference is likely to vary more than this due to the variations of encoding used for the audio/video CODECS.

6.2.2.3. Upgraded network

The starting test for these concentrated on a high bandwidth connection without any background traffic. The TDR database was wiped at the start of the test and then allowed to populate during the course of the experiment. The configuration files for the SDNE and the CCA policy which were used in the experiment are included in Appendix D.4, RUDE was not used to add additional load to the network. This experiment concerns a mixed DSL and Ethernet based video conference. The DSL had a limit of 512 Kbps upload and download.

The initial configuration was limited in order to show the reconfiguration aspects of the CCA, the initial configuration limited to 128 Kbps then after this initial configuration the emulated network was changed to a 512 Kbps DSL, with upload and download both set to 512 Kbps. This shows the behaviour of the CCA when there is a change in the network.

The initial configuration was 125 Kbps. The subsequent configurations use an increasing bit-rate. An equation based configuration was used which used the TCP throughput equation. The initial bit-rate configuration was artificially limited by the policy since little is known about the network.

Each conference (run) was schedule to last 30 minutes and there were 9 runs performed. These cluster into three segments on the graph. Firstly, around the initial configuration point.

Secondly, at just under 400 Kbps and finally at a stable point that the network is able to support. Figure 48 shows the results for this experiment.

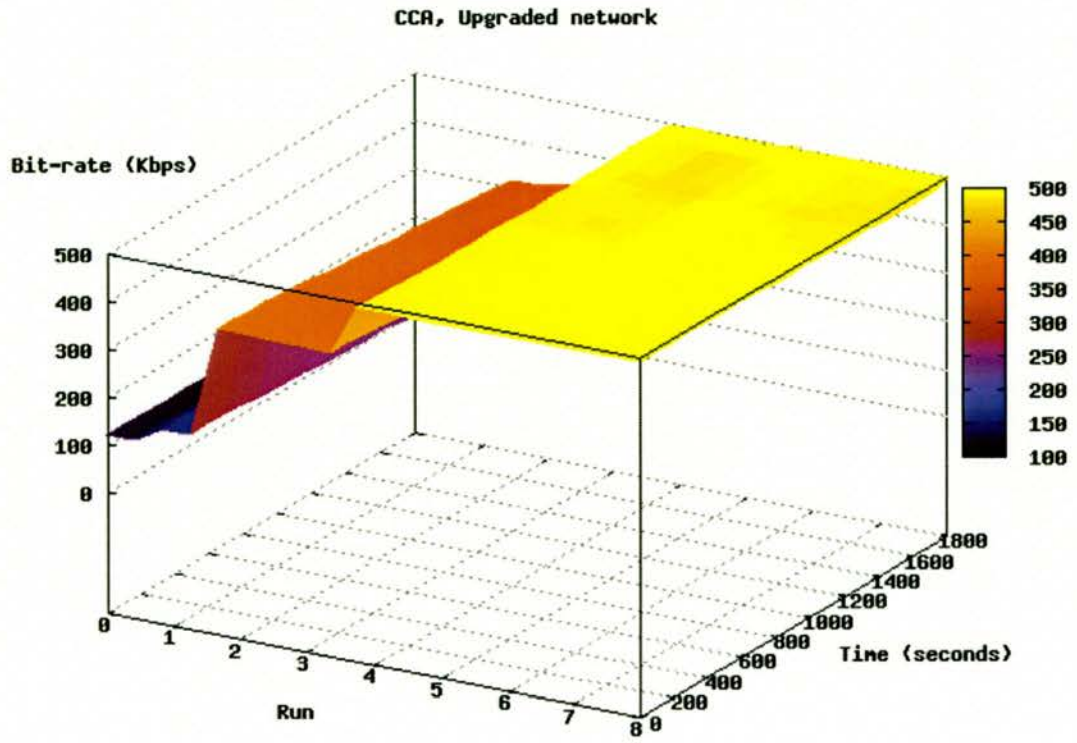


Figure 48 Upgraded network

Run	Min	Max	Mean	Standard deviation
0	104.87	126.36	116.70	7.64
1	122.35	161.90	142.50	12.60
2	337.49	394.13	368.73	15.60
3	360.92	386.80	369.48	8.60
4	487.45	492.82	490.64	1.41
5	483.54	493.80	488.84	3.53
6	490.87	495.75	494.05	1.24

7	486.47	493.31	490.67	1.91
8	485.01	494.77	490.49	3.11

Table 8 Summary for the unloaded network evaluation

This experiment demonstrates the Conference Controller Architecture's ability to adjust its bandwidth usage when an increase has been detected. This happens when the bandwidth on a path has been updated in the infrastructure.

The CCA shows its cautious approach to increasing the bandwidth it will be using when additional network capacity becomes available. There is no way for the CCA to know if the network has been upgraded or if, for example the increase in the network capacity is due to the network being used less as would be the case during e.g. student holidays within a University. This presents a problem since if the CCA reacts too slowly to the increased capacity when it is due to a long term increase in network capacity then a CCA hosted conference would not benefit quickly. If, however the CCA does quickly expand its bandwidth usage when available capacity becomes available, but the increase is due to a temporally lightly loaded network then the initial starting point of the CCA conference during the time of normal load would be too high. This would cause a sharp change in quality at the beginning of the conference as the CCA estimates a new point of consistency.

6.2.2.4. Live experiments

This series of tests looks at the CCA in a real network environment. These tests were performed as validation of the CCA when operating within the test bed environment of the SDNE system.

This test involved a meeting being scheduled between a number of machines on the Internet. Since not all the machines taking part in the test were able to communicate directly via

multicast channels multicast tunnels were used to connect the dispersed sites with the tunnels being provided by mTunnel [201].

6.2.2.4.1. Meeting

For this test a group of three nodes were used. The first two nodes were hosted within the School of Computer Science at St Andrews. The first node was hosted on a 100 Mbps Ethernet, the second on 11 Mbps wireless Ethernet. The final node in the configuration was hosted on a 256/512 Kbps ADSL line. The CCA policy was set to be a meeting implying that absolute quality of the video can be traded against delay in order to ensure a low delay situation.

The initial configuration used 64 Kbps per node given a total bandwidth suggestion of 192 Kbps. The bandwidth was split with 11 Kbps used for the audio stream and the rest used for the video stream. The video was always transmitted whilst the audio used silence suppression. The audio bandwidth was reserved for each participant so that all participants could transmit sound at the same time if required. The tests were carried out in a way so that the ADSL network connection was not being used for anything other than the videoconference. This was done in order to provide a stable network environment.

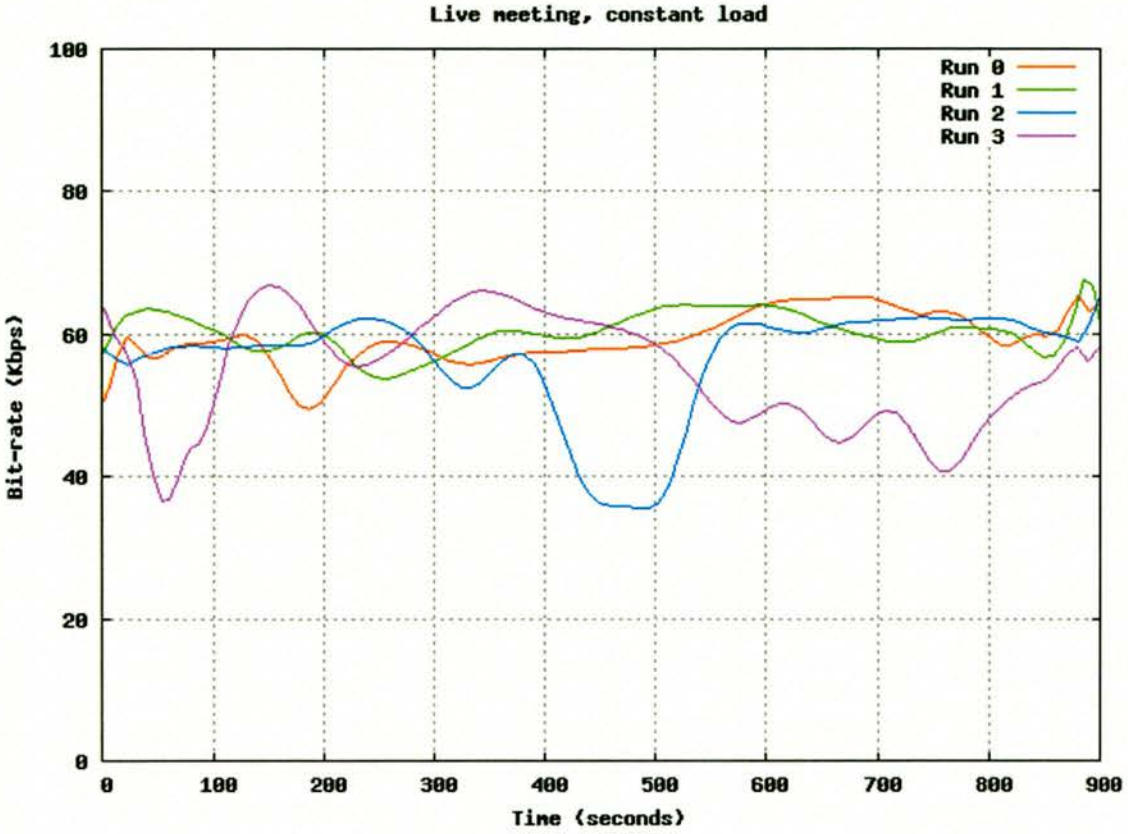


Figure 49 Live meeting, constant load

Figure 49 shows a trace of the transmission rate for a Participant Agent within the meeting. There were four videoconferences, each lasting 15 minutes. The x axis shows the time in seconds while the y axis shows the bit-rate used. There was no artificial load placed upon the network so the limiting network technology was the ADSL connection. The interesting aspect of this is the consistency of initial configuration of the session since the network is not changing the configuration suggested by the CCA does not change.

6.2.2.4.2. Tutorial

For this test a group of four nodes was used. Three of the nodes were hosted at the School of Computer Science within the University of St Andrews while the final node was hosted via a 256/512 Kbps ADSL connection. Two of the machines within the School of Computer Science were hosted on 100 Mbps wired Ethernet while the third was hosted on 11 Mbps

wireless Ethernet. An artificial load of 128 Kbps was added to the ADSL connection after 2 runs in order to emulate a change in the available bandwidth for a connection.

An initial configuration of 128 Kbps for video and 11 Kbps for audio was used. Silence suppression which was tied to the video was used. This means that when silence is detected both audio and video are no longer transmitting. This saves network resources. It would also be acceptable to reduce the frame-rate of the video (and therefore the bit-rate) when silence is detected.

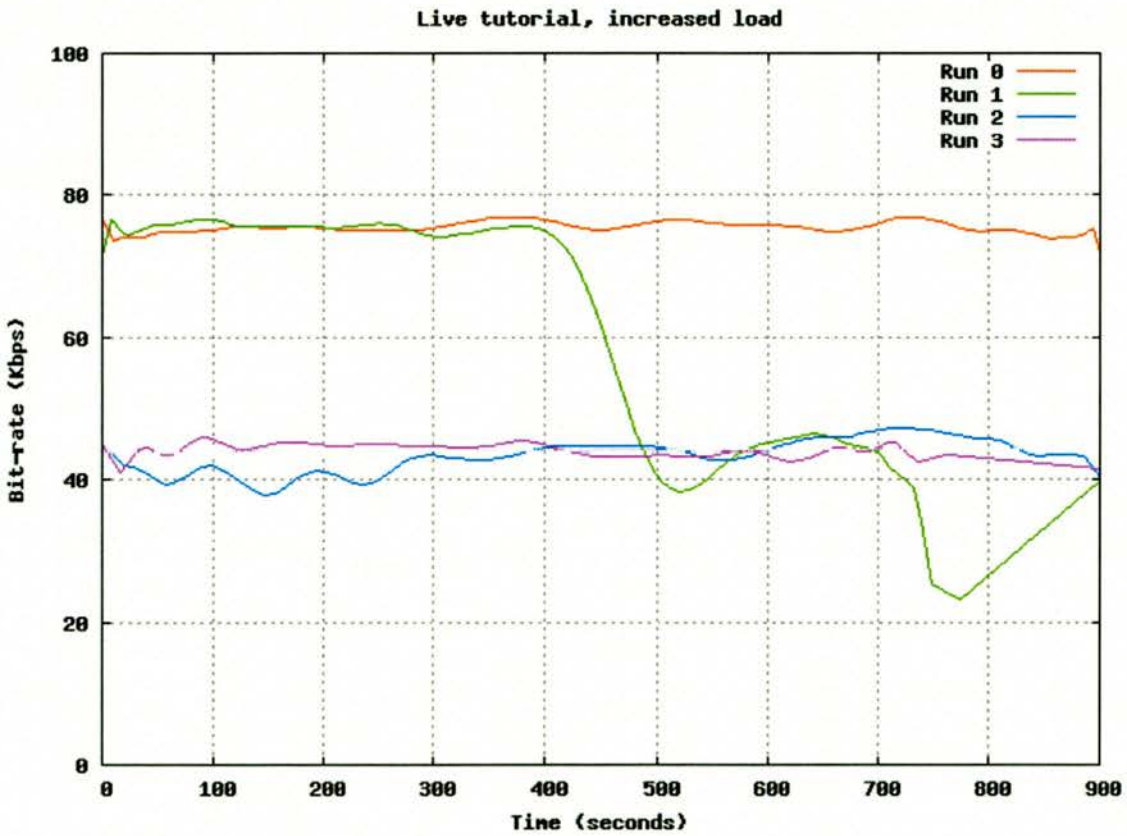


Figure 50 Live tutorial, increased load

Figure 50 shows the trace of transmission rate for the Participant Agents within the tutorial. There are four videoconferences shown each lasting 15 minutes. The x axis shows the time in seconds while the y axis shows the bit-rate. Midway through the second conference the

network was artificially loaded using RUDE (at 128 Kbps). The second run (run 1) clearly shows the increase in background traffic by dropping the bit-rate used within the tutorial. The rest of the second run presents some problems for the video CODEC (H.263 from the JMF) which creates some fluctuations in the encoding due to the overloaded network and the change in the bit-rate which takes some time for the video CODEC to process (the audio is kept the same 10 Kbps using GSM from JMF). The video quality changes for the latter half of the run but is stable for the following two runs.

6.3. Conclusions

The traffic measurements provided by the Conference Controller Architecture coupled with a policy script which is able to make appropriate use of them is able to configure videoconferences for long term consistency. The experiments performed demonstrate that even with changing network conditions the CCA was able to provide a configuration for a consistent experience throughout the conference, typically to within 1 Kbps.

The system is able to re-evaluate and redeploy its usage of the network resources when an increase in the physical network resources has been made. This is made in both a TCP and human friendly way. The attempt to increase the network usage is only made at the beginning of the conference when the conference is initially configured and not during the course of the conference when the audio/visual quality is subject to the scrutiny of the users. This has the effect of meaning that additional network resources can be utilised to enhance the quality presented to the user while maintaining the audio/visual quality of the conference.

When used on networks with varying loads the CCA configures the conference to a sustainable rate which is below its fair share throughout the duration of the conference.

Chapter 7

Conclusions

&

Future Work

7. Conclusions & future work

7.1. Conclusions

This thesis has investigated the problems of videoconferencing over the Internet and reported on the design and creation of a framework, the Conference Controller Architecture, which addresses these problems. In addition, a test bed for the development of the CCA was designed and created, which has wider applicability, and recommendations on how to select QoS parameters have been made which take the participants' perceptions into account. The case has been presented for the use of network monitoring and condition prediction in order to provide an improved QoS experience for users of videoconferencing applications.

The thesis has made 2 contributions:

1. A predictive quality of service framework for video conferencing on the Internet, the Conference Controller Architecture (CCA).

2. The Scenario Driven Network Emulation (SDNE) test bed which can be used for aiding the development of real-time groupware applications.

An implementation of the CCA was developed and tested using the SDNE. The CCA fits into the traditional QoS classifications as "less than best effort" since it does not attempt to maximise the absolute quality at any given moment. Instead it takes a longer term view that consistency of quality is more important within a videoconference and it aims to provide a trade-off between the absolute audio/visual quality which is experienced and the consistency of that quality.

An algorithm which is able to configure and maintain a conference for maximum consistency was presented. By default, when no information is known, its behaviour tracks the available bandwidth using a smoothing algorithm. When previous conferences have been run the system uses the past information to configure itself. The previous traces are compared against the theoretical capabilities of the network.

The Scenario Driven Network Emulation system provides a test bed for the development and evaluation of group based applications without the need for repetitive testing upon groups of people.

The suggestions for the configuration of videoconferencing sessions are based on the evidence for the psychological impact of audio and video degradation. These are coupled with network measurements which show that it is possible to provide a consistent QoS while being TCP friendly.

7.2. Future work

There are three areas where this research can go forward. The first is for providing quality of service for streaming media. The second is for providing configuration information for application overlay networks. Thirdly, an extension to the Conference Controller Architecture could be developed for providing quality of service information for users of wireless networks.

7.2.1. Quality of service for streaming web based media

Web-based streaming media, as represented by RealMedia or Windows Media Player, has seen a surge in popularity as computer hardware and networks have become better able to handle the larger load. One problem they have is that they use a suggestion from the user along with network measurements in order to configure the stream for use. If a high quality

media stream is chosen, this has high network throughput requirements. If the network path is not able to support the required bit-rates then a lower quality media stream is used instead. This can lead to harsh reductions in quality when a large amount of traffic is on the network. The bottleneck in the network does not have to be one which is reached just due to the media stream, for example the normal ebb and flow of network usage could create these temporal based bottlenecks.

Using the Conference Controller Architecture these temporal based network bottlenecks can be anticipated[20] and the media stream can be configured to deal with them.

This environment is ideally suited to configuration by the CCA as in most cases the length of the media stream is known in advance. The media streams can be pre-encoded for certain bandwidths and loss rates. In point-to-point media streaming as used by RealNetworks [208] there is no need for end-to-end interactivity. This solves the potential scaling problem that would be experienced if each stream had to be encoded on a per client basis.

7.2.2. Quality of service in application level overlay networks

Since this research started there has been a downturn in usage of IP multicast and the Mbone, with no direct replacement. The most promising basis for future real-time groupware is that of application level overlay networks providing multicast, referred to sometimes as overcast. These are application level multicast groups which are setup ad-hoc as the participants join the group. Savage et al [209] discovered that in 30% to 80% of cases there is an alternative network path which is able to provide a significantly increased quality.

Overlay networks as stated are created by the application. A network is formed as instances join the group. Each instance of the application has the ability to route traffic to other nodes.

[20] Assuming that the CCA has been previously used during the time which these throughput limits occur.

As such they are able to form in the most efficient manner for routing the traffic within given parameters.

There are three important components of an application level overlay network:

1. Entrypoints – This is the place where a media stream would enter the network.

2. Reflectors – This is a point in the network were the media stream is received and then forwarded on to either another reflector or to an endpoint.

3. Endpoints – This is the final destination of the media stream where it is displayed to the user.

In a normal application level overlay network for multimedia conferencing all three of these components exist within a single application on a user's computer. It is also possible that reflectors could be set up within the network to aid in the creation of overlay networks in efficient ways.

This proposed area of research provides the most effective way of configuring the network topology of the overlay network given the parameters imposed by the application. If multimedia is taken as the example then it is easy to see that delay, jitter, bandwidth and packet loss all play an important part in the quality that the users of such a system will receive. The quality they receive is due to the network paths that are taken. With this form of peer to peer application the network paths which are to be used can be chosen to a limited degree by the application. This adds an extra dimension to the ability of a system to manage the available quality of service.

Incorporating the research presented here into overlay network applications would mean the creation of an extra component in the Conference Controller Architecture: an Overlay Network Controller (ONC). The job of this component would be to calculate the most effective way to deliver the multimedia within the given quality of service constraints by organising the topology of the application overlay network. Various methods could be employed in order to provide the best quality of service for the user's of such a system.

1. Media streams could be split into two or more concurrent streams and sent via different routes to the same destination. This would allow for a higher absolute quality than could be achieved by using a single route.

2. Media streams can be sent via the network path which allows for the best quality for a given time of day and duration of session.

The TDR would need no changes. The ONC would use the list of participants and their network locations and the information gathered by the PA about the network paths in order to suggest network topologies.

Topologies could be optimised for differing conditions depending upon the type of conferencing undertaken. With the three examples presented in this thesis the preferred conditions for the overlay network topology would be:

1. Meeting – Aim to minimise delay and jitter between all participants, keeping packet loss down. A trade off between high bandwidth and low delay must be maintained for high interaction while maintaining the visual and audio quality.

2. Tutorial – Trade off between maximisation of bandwidth from the tutor and minimisation of delay and jitter between all participants.

3. Lecture – Maximise bandwidth from the lecturer to all other participants. Keeping the exact timing for each stream is unimportant as there will be little interaction between the participants of the lecture and only limited interaction between the lecturer and the participants.

The policies which inform the system of settings for the conferences will have to be altered in order to take into account the new information about the topology of the network. The ONC and the policies will have to be able to communicate in order to reach a trade off between the conference quality and the overlay network topology that can realistically be achieved with the participants of the conference.

7.2.3. Wireless quality of service

One of the problems of wireless networks relates to TCP and the higher packet losses that are experienced. TCP uses packet loss as an implicit indication of congestion. Normally packet loss means that the sender should decrease its window size and send less data. In the wireless environment the extra delay and the spurious time-outs caused by link-level loss are *not* an indication of congestion, and the sender should not therefore back-off in the same way.

When dealing with UDP traffic in the wireless domain there is no feedback and so little is known about how much packet loss is experienced.

Wireless extensions to the CCA would be required. This extension would monitor the wireless traffic and take measurements of the signal strength as well as measurement relating to the network performance of the data stream. Then when a network connection is made the signal strength is checked against the current database of measurements. A suggestion of expected network conditions could then be provided based on the location and the time.

The approximate location of a user can be calculated using a form of triangulation based on the signal strength [210] of the wireless communication. This form of location tracking can be extended so that only one wireless access point is required for the function of a system since it is not the location of the user that is of interest, but the network conditions that will be experienced.

7.3. Final comments

The ever increasing connectivity and improved quality of network connections have highlighted the potential for computer-based video conferencing across the Internet to become a routine application that provides acceptable quality to groups of participants who are not technical experts in the field of networking. To realize that potential some hurdles must be overcome. The CCA is a very promising solution to these problems. It takes participant's perceptions, network fairness, and the best-effort variability of the Internet into account when setting up and maintaining conference sessions. It constantly learns as it runs, thereby passing on infrastructure improvements to users in a controlled manner.

Glossary

ACK Acknowledgement. Normally a bit set in the header of TCP to indicate that the previous information was received.

Active Participant A member of a multimedia session who is expected to be transmitting data for a large portion of the duration of the conference. An example of this class of participant would be a tutor within a tutorial or any member of a meeting.

APDU Application protocol data unit.

ATM Asynchronous Transfer Mode. A type of network which has built in resource reservation capabilities.

AIMD Additive Increase Multiplicative Decrease. The congestion control mechanism used in TCP.

ALF Application Level Framing. A method of splitting data up for packetisation which ensure that packet loss has a minimal effect of the media stream.

Bandwidth The amount of data which can be transported from a sender to a receiver per unit time, normally expressed in bits per second.

Bottleneck Bandwidth The bandwidth at the bottleneck of the network path.

CCA Conference Control Architecture. Predictive quality of service framework.

CC Conference Controller. Central component of the CCA which provides initial suggested session settings and ongoing suggestion based on feedback.

CIF Common Intermediate Format. 352x288 in PAL format or 352x240 in NTSC.

CODEC Compressor Decompressor. A library which is used to encode/decode a media stream.

COM Component object model. A software reuse system used by Microsoft in its Windows operating systems and products.

Compression The conversion of digital data which typically represents audio or video into a more compact form for easier transport.

CORBA Common Object Request Broker Architecture. A distributed object model created by the Object Management Group.

Delay The time it takes a packet to travel from a sender to a receiver.

ECN Explicit Congestion Notification. A method of notifying a sender of packet loss without using dropping packets.

Encoding Delay Also known as algorithmic delay. It is a delay that is intrinsic to the algorithm of a lock or subsystem and is independent of CPU speed.

FEC Forward error correction. A method of inserting additional information into a datastream so that if packets are lost they can be reconstructed using the data which has been received.

Groupware Software designed to be used by two or more people in order to support their interactions. A simple and everyday example would be video conferencing software.

ISDN Integrated Services Digital Network. A type of digital phone line which operates at 64Kbpsecond. Normally provided as ISDN-2 which gives 2 64 Kbps channels.

internetMCI MCI is an international ISP which operates facilities in North America, Latin America, Europe, Africa and the Asia-Pacific region. At the time of writing MCI has 20 million customers worldwide.

HDTV High definition television. Roughly 1000 pixels by 1000 pixels at 1MB per frame.

Jiffies A variable, in the Linux kernel which keeps increasing until it reaches a maximum value. Once it reaches its maximum value it reset to zero and then carries on increasing. It is increased at the current system HZ (a defined constant) rate as a result of a hardware interrupt.

JMF Java Media Framework. A set of APIs and libraries which provide multimedia capture, transport and decoding for Java.

Jitter The inter-packet arrival time.

Media scaling Changing the media to another resolution.

Mbone A virtual overlay network on the Internet for use with IPv4 multicast traffic.

Media A form of multimedia referring to audio or video.

Meeting The act of assembling for a purpose. Meetings are interactive events where any member may speak at any given moment.

NAT Network address translation. A method of sharing a single globally routable IP address with an entire local area network.

Netfilter A framework within the Linux kernel which allows callback functions to be placed into the network stack so that the network stack can be extended to perform functions which it was not originally designed to do.

MTU Maximum transfer unit. [211] defines it as 'the size of the largest packet that can be transmitted'.

NAÏVE Network Aware Internet Encoding.

NTSC National Television System Committee. Colour television standard developed in 1962 in the US. It uses 525 lines per frame with a frame-rate of 29.97 frames per second. It is used in the US, Canada, Japan, in most of the American continent and in various parts of Asia.

PA Participant Agent. End point of the multimedia communications. This is the user facing application and is part of the Conference Controller Architecture.

PAL Phase Alternating Line. A TV standard introduced during the 1960's. It has 625 lines per frame with a frame-rate of 25 frames per second. It is used in most western European countries, Australia and some parts of Africa, South America and Asia.

Passive Participant A member of a multimedia session that is not expected to transmit a lot of data. Example could be an audience member during a lecture.

Phonemic restoration The ability of the human brain to subconsciously repair the missing segments of speech.

PPP Point to point Protocol. A method for connecting a computer over a phone line to another computer.

Proc file system A virtual file system under UNIX which is located at /proc. It provides a view into the current state of the kernel.

Quality of Service (QoS) The quality of the application.

Round Trip Time (RTT) The time it take a packet to travel from a sender to a receiver and then from the receiver back to the sender.

Receiver The receiver of the media stream.

Reliable Provides in order, guaranteed packet delivery.

RTP Real Time protocol. A media transport protocol which expands IP and uses application level framing.

RTT see Round Trip Time.

RTCP Real Time Control Protocol. A companion to RTP which provides session feedback.

RAT Robust Audio Tool. The audio component of the Mbone tools.

Sender The transmitter of a media stream.

T1 A leased line phone connection which has a data rate of 1.544 Mbps sometimes known as a DS1 line.

Telepresence Telepresence is a system where a human is able to take part in some activity in a remote location. Telepresence systems provide a channel of communication in either the audio or audio/visual domain where the visual part of the system is typically the view of the user via a webcam.

Teledata Teledata is a system where an additional means of communication is provided. Typically it consists of a shared workspace. An example of teledata would be a shared document editor where multiple people could see the document which was being edited even though they were in different physical locations.

TCL Tool Control Language. A scripting language originally developed to add graphical user interfaces (using TK see below) to components which were written in C.

TCL/TK Tool Control Language/ToolKit. See above for a description of TCL. TK or Toolkit provides the user interface toolkit.

Teleconference A form of conference in which participants at two or more location are able to hear each other. Typically performed using telephones.

TDR Traffic Data Repository. Persistent store for network statistics used within the Conference Controller Architecture.

Throughput The amount of data that can be transported from the sender to the receiver measured in bits per second. This is an application level measurement.

Tutor A member of a tutorial. This participant is not a student on the course being taught but is a supporting member of staff which answers student's questions. Tutors typically speak more than the students within a tutorial although a tutorial can be a highly interactive event.

VIC Videoconferencing Tool. Video component of the Mbone tools.

Videoconference A form of conference in which participants at two or more locations are able to see and hear each other.

White noise A synonym for background noise. Noise which is normally expected to exist in the environment for example in an office the sounds of computers.

QCIF Quarter Common Intermediate Format. A videoconferencing standard which specifies 174x144 pixels at 20 frames per second.

References

[1] I. Dorros, "PICTUREPHONE," in *Bell Laboratories Record*, vol. 47, 1969, pp. 136-141.

[2] J. P. Molnar, "PICTUREPHONE Service-A New Way for Communicating," in *Bell Laboratories Record*, vol. 47, 1969, pp. 134-135.

[3] S. Casner and S. Deering, "First IETF Internet audiocast," *ACM SIGCOMM Computer Communication Review*, vol. 22 no. 3, pp. 92-97, July 1992.

[4] Steve Casner, "The MBONE FAQ", web page, http://andrew.triumf.ca/pub/mbone/mbone.faq.html, Accessed 22nd June 2005

[5] T. Parker, "Cornell University's CU-SeeMe Page", web page, http://cu-seeme.cornell.edu/Welcome.html, Accessed 22nd Feburary 1998

[6] B. R. Abdon and R. T. Raab, "IRRI's experience with desktop Internet-based videoconferencing to support human capital development in NARS," International Institute for Communication and Development October 1999, Available from http://www.wrplatinum.com/BekijkSamenvatting.asp?Inhoudsnummer=233.

[7] A. W. Davis and I. M. Weinstein, "The Business Case For Videoconferencing," Wainhouse Research, Duxbury March 2002, Available from http://www.wrplatinum.com/BekijkSamenvatting.asp?Inhoudsnummer=233.

[8] P. Ekman, "Facial sign: Facts, fantasies, and possibilities," in *Sight, Sound and Sense*, T. Sebeok, Ed. Bloomington: Indiana University Press, 1978, pp. 124-156.

[9] P. Ekman, "About brows: Emotional and conversational signals," in *Human ethology*, J. Aschoff, M. v. Carnach, K. Foppa, W. Lepenies, and D. Plog, Eds.: Cambridge University Press, 1979, pp. 169-202.

[10] A. Pease, *Body Language: How to Read Others' Thoughts by Their Gestures*, ISBN 0859697827, Sheldon Press, 1997.

[11] MCI WorldCom, "Meetings in the UK: A study of trends, costs, and attitudes toward business travel and teleconferencing and their impact on productivity," MCI WorldCom, London, Whitepaper, 2000, Available from http://e-meetings.mci.com/global/pdf/MiUK.pdf.

[12] MCI WorldCom, "Meetings in America," MCI WorldCom, Whitepaper, June 1998, Available from http://e-meetings.mci.com/meetingsinamerica/uswhitepaper.php.

[13] E. A. Isaacs, T. Morris, T. K. Rodriguez, and J. C. Tang, "A comparison of face-to-face and distributed presentations," presented at ACM Conference on Human Factors in Computing Systems, Denver, CO, 7th-11th May 1995.

[14] A. Watson and M. A. Sasse, "Evaluating audio and video quality in low-cost multimedia conferencing systems," *Interacting with Computers*, vol. 8 no. 3, pp. 255-275, 1996.

[15] J. C. Tang and E. A. Isaacs, "Why do users like video? Studies of multimedia-supported collaboration," *Computer Supported Cooperative Work*, vol. 1 no. 3, pp. 163-196, 1992.

[16] A. H. Anderson, L. Smallwood, R. Macdonald, J. Mullin, and A. M. Fleming, "Video data and video links in mediated communication: what do users value?," *Human Computers Studies*, vol. 52 no. 1, pp. 165-187, January 2000.

[17] G. Matarazzo and A. Sellen, "The Value of video in work at a distance: addition or distraction?," *Behaviour and Information Technology*, vol. 19 no. 5, pp. 339-348, 1st September 2000.

[18] A. S. Patrick, "The human factors of MBone videoconferences: Recommendations for improving sessions and software," *Computer-Mediated Communications*, vol. 4 no. 3, March 1999.

[19] J. Mullin, M. Jackson, A. H. Anderson, L. Smallwood, M. A. Sasse, A. Watson, and G. Wilson, "The ETNA Taxonomy," University of Glasgow and University College

London, final report, February 2002, Available from http://www-mice.cs.ucl.ac.uk/multimedia/projects/etna/taxonomy.pdf.

[20] R. B. Ochsman and A. Chapainis, "The Effects of 10 Communication Modes on the Behaviour of Teams During Co-operative Problem-Solving," *Man-Machine Studies*, vol. 6 no. 5, pp. 579-620, 1974.

[21] S. Gale, "Human aspects of interactive multimedia communication," *Interacting with Computers*, vol. 2 no. 2, pp. 175-189, August 1990.

[22] C. Rudman, R. Herta, C. Marshall, and E. Dykstra-Erickson, "Channel overlead as a driver for adoption of desktop video for distributed group work," in *Video-Mediated Communication*, K. E. Finn, A. J. Sellen, and S. B. Wilbur, Eds. Mahwah, NJ: Lawrence Erlbarm Associates, 1997, pp. 199-243.

[23] E. A. Isaacs and J. C. Tng, "What Video Can and Cannot Do for Collaboration: A Case Study," *Multimedia Systems*, vol. 2 no. 2, pp. 63-73, August 1994.

[24] L. Carles, "Benefits and limits of video channel on mediated interactions," Geneva Interaction Lab, TECGA / University of Geneva May 2001, Available from http://craft.epfl.ch/webdav/site/craft/users/157990/public/LaureREportV2.pdf.

[25] W. Buxton, "Integrating the Periphery and Context: A New Taxonomy of Telematics," presented at GI'95 Graphics Interface Conference, Québec, Québec, 17th-19th May 1995.

[26] M. Hollier and R. Voelcker, "Towards a multimodal perceptual model," *BT Technology Journal*, vol. 15 no. 4, pp. 163-172, October 1997.

[27] L. A. Rettinger, "Desktop videoconferencing: Technology and use for remote seminar delivery," North Carolina State University, Raleigh, NC, Masters thesis, July 1995, Available from http://www.visc.vt.edu/succeed/dt5/pdf/larthesis.pdf.

[28] B. O'Conaill, S. Whittaker, and S. Wilbur, "Conversations over video conferences: an evaluation of the spoken aspects of video-mediated communication," *Human-Computer Interaction*, vol. 8 no. 4, pp. 389-428, 1993.

[29] K. Nakazono, "Frame rate as a QoS parameter and its influence on speech perception," *Multimedia Systems*, vol. 6 no. 5, pp. 359-366, September 1998.

[30] D. Miras, "A Survey of Network QoS Needs of Advanced Internet Applications," University College London, London, Working Document, November 2002, Available from http://qos.internet2.edu/wg/apps/fellowship/Docs/Internet2AppsQoSNeeds.pdf.

[31] D. W. Massaro, M. M. Cohen, and P. M. T. Smeele, "Perception of Asynchronous and Conflicting Visual and Auditory Speech," *Acoustical Society of America*, vol. 100 no. 3, pp. 1777-1786, September 1996.

[32] D. Qian and M. D. Gross, "Collaborative Design with Netdraw," presented at CAAD Futures, Atlanta, Georgia, USA, 7th-8th June 1999.

[33] T. Tung, "Mediaboard: A Shared Whiteboard Application for the MBone", MSc, Computer Science, University of California, Berkeley, 1997

[34] Y. Zhang, S. Caron, and N. D. Georganas, "MMwb: A Multimedia Whiteboard for Distance Learning Applications," presented at 13th International conference on Computer Communication (ICCC '97), Cannes, France, 19th-21st November 1997.

[35] J. K. Kies, R. C Willings, and M. B Rosson, "Controlled laboratory experimentation and field study of video conferencing for distance learning applications," Virginia Polytechnic Institute and State University, Technical Report, June 1996, Available from ftp://cosmos.kaist.ac.kr/pub/users/hakim/kies96.doc.

[36] J. K. Kies, J. Kelso, and R. C. Williges, "The use of scenarios to evaluate the effects of group configuration and task on video-conferencing communication

effectiviness," presented at Third Mid-Atlantic Human Factors Conference, Blacksburg, VA, March 1995.

[37] V. Bruce, "The Role of the Face in Communication: Implications for videophone design," *Interacting with Computers*, vol. 8 no. 2, pp. 166-176, June 1996.

[38] J. K. Kies, R. C. Williges, and M. B. Rosson, "Evaluating desktop video conferencing for distance learning," *Computers & Education*, vol. 28 no. 2, pp. 79-91, February 1997.

[39] A. Bouch and M. A. Sasse, "The case for predictable media quality in networked multimedia applications," presented at ACM/SPIE Multimedia Computing and Networking (MMCN'00), San Jose, USA, 25th -27th January 2000.

[40] D. S. Hands and S. E. Avons, "Recency and duration neglect in subjective assessment of television picture quality," *Applied Cognitive Psychology*, vol. 15 no. 6, pp. 639-657, 12th November 2001.

[41] ISO Standard, "Information technology - Generic coding of moving pictures and associated audio information: Systems - Part 1," ISO, ISO Standard, 14th December 2000, Available from http://www.standards.com.au/Catalogue/script/Details.asp?DocN=AS5183965897 39.

[42] JTC 1/SC 29, "Information technology -- Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s -- Part 2: Video," ISO Standards, ISO Standard, 12th August 1993, Available from http://www.standards.com.au/Catalogue/script/Details.asp?DocN=stds000011757.

[43] International Telecommunication Union, "Video coding for low bit rate communication," International Telecommunication Union, ITU Recommendation, January 2005, Available from http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-H.263.

[44] D. Miras, "Network QoS Needs of Advanced Internet Applications," Internet2 QoS Working Group, University College London, London November 2002, Available from http://qos.internet2.edu/wg/apps/fellowship/Docs/Internet2AppsQoSNeeds.pdf.

[45] T. Turletti, "H.261 Software Codec for videoconferencing over the Internet," INRIA, Research Report, 1834, January 1993, Available from ftp://www.inria.fr/rodeo/ivs/papers/inria_report1834.ps.gz.

[46] International Telecommunication Union, "Pulse code modulation (PCM) of voice frequencies," International Telecommunication Union November 1988, Available from http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-G.711

[47] International Telecommunication Union, "7 kHz audio-coding within 64 kbit/s," International Telecommunication Union November 1988, Available from http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-G.722..

[48] International Telecommunication Union, "Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 Kbit/s," International Telecommunication Union November 1996, Available from http://www.itu.int/rec/recommendation.asp?type=products&parent=T-REC-g.

[49] Nortel Networks, "Voice over packet: An assessment of voice performance", white paper, http://www.nortel.com/products/library/collateral/74007.25-09-01.pdf, Accessed January 2004

[50] International Telecommunication Union, "G.728: Coding of speech at 16 kbit/s using low-delay code excited linear prediction," International Telecommunication Union September 1992, Available from

http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-G.728.

[51] J. D. Rosenberg, "G.729 error recovery for Internet telephony," Columbia University, Project Report, May 1997, Available from http://www.cs.columbia.edu/techreports/cucs-016-01.pdf.

[52] M. Freerick, "How much delay is too much delay?," presented at 112th AES Convention, Munich, Germany, 10th-13th May 2002.

[53] K. Thompson, G. J. Miller, and R. Wilder, "Wide-Area Internet Traffic Patterns and Characteristics (Extended Version)," *IEEE Network*, vol. 11 no. 6, pp. 11-23, Novermber 1997.

[54] J. C. Bolot, "End-to-end packet delay and loss behaviour in the Internet," presented at SIGCOMM Symposium on Communication Architectures and Protocols, San Francisco, California, USA, August 1993.

[55] J. C. Bolot and A. Vega Garcia, "Control mechanism for packet audio in the Internet," presented at The Conference on Computer Communications (IEEE Infocom), San Fransansico, California, March 1996.

[56] V. Paxson, "Measurement and Analysis of End-to-End Internet Dynamics", Computer Science, University of California at Berkeley, 1997

[57] M. S. Borella, D. Swider, S. Uludag, and G. B. Brewster, "Internet Packet Loss: Measurements and Implications for End-to-End QoS," presented at International Conference on Parallel Processing, August 1998.

[58] M. Yajnik, S. Moon, J. Kurose, and D. Townsley, "Measurement and Modelling of the Temporal Dependence in Packet Loss," presented at IEEE INFOCOM, March 1999.

[59] M. Yajnik, J. Kurose, and D. Towsley, "Packet Loss Correlation in the MBone Multicast Network," IEEE Global Internet Conference, London, Nov 1996.

[60] V. Paxson, "End-to-End Internet packet dynamics," presented at SIGCOMM Symposium on Communications Architectures and Protocols, Cannes, France, September 1997.

[61] H. Balakrishnam, V. N. Padmanablah, S. Seshan, M. Stemm, and R. H. Katz, "TCP Behaviour of a Busy Internet Server: Analysis and Improvements," presented at IEEE INFOCOMM, March 1998.

[62] S. Moon, J. Kurose, and D. Townsley, "Correlation of packet delay and loss in the Internet," University of Massachusetts, Amherst, Technical Report, November 1998, Available from ftp://gaia.cs.umass.edu/pub/Moon98_Corr-TR-98-11.ps.gz.

[63] V. Paxson, "End-to-end internet packet dynamics," *IEEE ACM Transactions on Networking*, vol. 7 no., pp. 277-292, June 1999.

[64] D. Loguinov and H. Radha, "End-to-End Internet Video Traffic Dynamics: Statistical Study and Analysis," presented at IEEE INFOCOM, June 2002.

[65] S. Uhlig and O. Bonaventure, "The Macroscopic Behavior of Internet Traffic: a Comparative Study," University of Namur, Technical Report, 2001, Available from http://www.infonet.fundp.ac.be/doc/ tr/Infonet-TR-2001-10.html.

[66] J. McCarthy, "Studioes sue MP3 startup Napster," in *CNN sci-tech*, 1999.

[67] CacheLogic, "The True Picture of Peer-to-Peer Filesharing", web page, http://www.cachelogic.com/research/index.php, Accessed 22nd June 2005

[68] University of Southern California Information Sciences Institute, "Internet Protocol", RFC 791, September 1981, Available from http://www.ietf.org/rfc/rfc791.txt, Accessed 1st Feburary 2005

[69] 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*, vol. 5 no. 6, pp. 784-803, December 1997.

[70] J. Nagle, "Congestion Control in IP/TCP Internetworks", RFC 896, 6th Jan 1984, Available from http://www.faqs.org/rfcs/rfc896.html, Accessed 1st Feburary 2005

[71] V. Jacobson and M. J. Karels, "Congestion Avoidance and Control," presented at ACM Computer Communication Review; Proceedings of the Sigcomm '88 Symposium, Standford, CA, August 1988 1988.

[72] A. Miller, "Best Effort Measurement Based Congestion Control", PhD, Department of Computer Science, University of Glasgow, 2002

[73] L. S. Brakmo and L. L. Peterson, "TCP Vegas: end to end congestion avoidance on a global internet," *IEEE Journal on Selected Areas in Communications*, vol. 13 no. 8, pp. 1465-1480, October 1995.

[74] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, April 1998, Available from http://www.faqs.org/rfcs/rfc2309.html, Accessed 2005 1st Feburary

[75] S. Floyd, "Connections with Multiple Congested Gateways in Packet-Switched Networks Part1: One-way Traffic," *Computer Communications Review*, vol. 21 no. 5, pp. 30-47, October 1991.

[76] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe, "Modeling TCP Throughput: A Simple Model and its Empirical Validation," presented at ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication, 1998.

[77] A. Kumar, "Comparative performance analysis of version of TCP in a local network with a lossy link," *IEEE ACM Transactions on Networking*, vol. 6 no. 4, pp. 485-498, August 1998.

[78] S. Floyd and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 2582, April 1999, Available from http://www.faqs.org/rfcs/rfc2582.html, Accessed 1st Feburary 2005

[79] C. Krasic, K. Li, and J. Walpole, "The Case for Streaming Multimedia with TCP," *Lecture Notes in Computer Science*, vol. 2158 no. 2001.

[80] S. Cen and J. Walpole, "Flow and congestion control for internet streaming applications," presented at Multimedia Computing and Networking (MMCN'98), San Jose, USA, January 1998.

[81] W. Tan and A. Zakhor, "Internet video using error resilient scalable compression and cooperative transport protocol," presented at ICIP, October 1998.

[82] J. R. Li, D. Dwyer, and V. Bharghavan, "A transport protocol for hetergeneous packet flows," presented at IEEE Infocom'99, March 1999.

[83] R. Rejaie, M. Handley, and D. Estrin, "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet," presented at INFOCOM '99, March 1999.

[84] S. Floyd, "TCP and Explicit Congestion Notification," *ACM Computer Communication Review*, vol. 24 no. 5, pp. 10-23, October 1994.

[85] W. Feng, D. Kandlur, D. Saha, and K. G. Shin, "A Self-Confguring RED Gateway," presented at IEEE INFOCOM, March 1999.

[86] J. Postel and J. Reynolds, "TELNET PROTOCOL SPECIFICATION", RFC 854, May 1983, Available from http://www.faqs.org/rfcs/rfc854.html, Accessed May 2004

[87] V. Paxson and M. Allman, "Computing TCP's Retransmission Timer", RFC 2988, November 2000, Available from http://www.faqs.org/rfcs/rfc2988.html, Accessed Janurary 2005

[88] R. Braden, L. Chang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP)", RFC 2205, September 1997, Available from http://www.faqs.org/rfcs/rfc2205.html, Accessed November 2001

[89] M Villapol and J Billington, "Analysing Properties of the Resource Reservation Protocol," presented at 24th Internation Conference on Application and Theory of Petri Nets, Eindhoven, Netherlands, June 2003.

[90] J. M. Pitts and J. A. Schormans, *Introduction to ATM Design and Performance: With Applications Analysis Software*, 2nd ed, ISBN 0471963402, Wiley & Sons, 1996.

[91] J. Wroclawski, "The Use of RSVP with IETF Integrated Services", RFC 2210, September 1997, Available from http://www.ietf.org/rfc/rfc2210.txt, Accessed 1st February 2005

[92] L. Breslau and S. Shenker, "Best-effort versus reservations: a simple comparative analysis," presented at ACM SIGCOMM'98, Vancouver, British Columbia, Canada, August 31st - September 4th 1998.

[93] S. Blake, D. Black, M. Carlson, E. Davies, A. Wang, and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998, Available from http://www.faqs.org/rfcs/rfc2475.html, Accessed June 2003

[94] A. Mankin, F. Baker, B. Braden, S. Bradner, M. O'Dell, A. Romanow, A. Weinrib, and L. Zhang, "Resource ReSerVation Protocol (RSVP) Version 1 Applicability Statement Some Guidelines on Deployment", RFC 2208, September 1997, Available from http://rfc.net/rfc2208.html, Accessed 1st Feburary 2005

[95] S. Shalunov and B. Teitelbaum, "QBone Scavenger Service (QBSS) Definition," Internet2 Technical Report, Proposed Service Definition, March 2001, Available from http://qos.internet2.edu/wg/wg-documents/qbss-definition.txt.

[96] K. N. A. J. Smith, "Design, Implementation and Experiences of the OMEGA End-Point Architecture," University of Pennsylvania May 1995, Available from http://www.exp-math.uni-essen.de/~dreibh/diplom/NS96.ps.gz.

[97] K. Nahrstedt and J. Smith, "The QoS Broker," *IEEE Multimedia*, vol. 2 no. 1, pp. 53 - 67, Spring 1995.

[98] D. Ferrari, "The Tenet Experience and the Design of Protocols for Integrated Services Internetworks," *Multimedia Systems Journal*, vol. 6 no. 3, pp. 179 -185, May 1998.

[99] A. Banerfea, D. Ferrari, B. A. Mah, M. Moran, D. C. Verma, and H. Zhang, "The Tenet Real-Time Protocol Suite: Design, Implementation, and Experiences," *IEEE ACM Transactions on Networking*, vol. 4 no. 1, pp. 1-10, 1996.

[100] B. Wolfinger and M. Moran, "A Continuous Media Data Transport Service and Protocol for Real-Time Communication in High Speed Networks," presented at Second International Workshop on Network and Operating System Support for Digital Audio and Video, Heidelberg, Germany, November 1991.

[101] A. Mills, *Understanding FDDI*, ISBN 0132199734, Prentice Hall, 1995.

[102] C Volg, L Wolf, R Herrtwich, and H Wittig, "HeiRat - Quality of Service Management for Distributed Multimedia Systems," *Multimedia Systems Journal*no., November 1995.

[103] G. Pasquale, E. Polyzos, E. Anderson, and V. Kompella, "The multimedia multicast channel," presented at 3rd International Workshop on Network and Operating System Support for Digital Audio and Video, San Diego, USA, 1992.

[104] L. Zhang, S. Berson, S. Herzog and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Function Specification," RFC 2205, September 1997, Available from http://www.faqs.org/rfcs/rfc2205.html, Accessed 1st January 2005

[105] G. Gopalakrishna and G. Parulkar, "A Real-time Upcall Facility for Protocol Processing with QoS Guarantees (poster)," presented at 15th ACM Symposium on Operating Systems Principles, December 1995.

[106] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996, Available from http://www.faqs.org/rfcs/rfc1889.html, Accessed July 2002

[107] ISO, "Quality of Service Framework," International Standard Organization, UK 1995

[108] TINA-C, "The QoS Framework," 1995, Available from http://www.tinac.com/.

[109] L. Besse, L. Dairaine, L. Fedaoui, W. Tawbi, and K. Thai, "Towards an Architecture for Distributed Multimedia Application Support," presented at Proc. International Conference on Multimedia Computing and Systems, Boston, May 1994.

[110] H. Briceno, S. J. Gortler, and L. McMillan, "NAIVE - Network Aware Internet Video Encoding," presented at 7th ACM International Multimedia Conference, Orlando, Florida, October 1999.

[111] "QBone Home Page", web page, http://qbone.internet2.edu/, Accessed September 2004

[112] D. D. Clark and D. L. Tennenhouse, "Architectural Considerations for a New Generation of Protocols," presented at ACM symposium on Communications architectures & protocols, Philadelphia, Pennsylvania, United States, 26th-28th September 1990.

[113] M. Zink, A. Jonas, C. Griwodz, and R. Steinmetz, "LC-RTP (Loss Collection RTP): Realiability for Video Caching in the Internet," presented at NGITA'00, Lwate, Japan, 4th-7th July 2000.

[114] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Stream Protocol (RTSP)", RFC 2326, April 1998, Available from http://www.ietf.org/rfc/rfc2326.txt, Accessed 2005 1st February

[115] Real Media, "Products and Services > Media Servers", web page, http://www.realnetworks.com/products/media_delivery.html, Accessed 9th June 2005

[116] Real Media, "RealPlayer", computer program, http://uk.real.com/player/?&src=ZG.uk.rp.rp.hd.def, Accessed January 2005

[117] D. Chiu and R. Jain, "Analysis of the increase and decrease algorithm for congestion avoidance in computer networks," *Computer Networks and ISDN*, vol. 17 no. 1, pp. 1-14, June 1989.

[118] K. K. Ramakrishnan and S. Floyd, "A Proposal to add Explicit Congestion Notification (ECN) to IP", RFC 2481, January 1999, Available from http://www.faqs.org/rfcs/rfc2481.html, Accessed 16th December 2004

[119] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1 no. 4, pp. 397-413, August 1993.

[120] D. Sisalem and H. Schulzrinne, "The Loss-Delay Based Adjustment Algorithm: A TCP-Friendly Adaption Scheme," presented at NOSSDAV, 1998.

[121] B. Noble, "System Support for Mobile, Adaptive Applications," *IEEE Personal Communications*no., Feburary 2000.

[122] B. Lamparter, A. Albanese, M. Kalfane, and M. Luby, "PET - priority encoding transmission (video): a new, robust and efficient video broadcast technology," presented at 3rd ACM International Conference on Multimedia, San Francisco, California, 1995.

[123] T. Komura, K. Fujikawa, and K. Ikeda, "Layered Transmission and Control of Packet Transmission Order for Multimedia Broadcasting," presented at INET2000, 2000.

[124] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," presented at ACM SIGCOMM, Standford, CA, August 1996.

[125] Y. Chawathe, S. A. Fink, S. McCanne, and E. A. Brewer, "A Proxy Architecture for Reliable Multicast in Heterogeneous Environments," *ACM Multimedia*no., pp. 151-159, 13th-16th September 1998.

[126] P. Bellavista, A. Corradi, and C. Stefanelli, "Application-Level QoS Control for Video-on-Demand," *IEEE Internet Computing*, vol. 7, no. 6, pp. 16-24, November/December 2003.

[127] C. Perkins, O. Hodson, and V. Hardman, "A survey of packet loss recovery techniques for streaming audio," in *IEEE Network*, vol. 12, 1998, pp. 40-48.

[128] V. Hardman, M. A. Sasse, M. Handley, and A. Watson, "Reliable audio for use over the Internet," presented at INET'95, Honolulu, Hawaii, 27th-30th July 1995.

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

[130] M. Podolsky, C. Romer, and S. McCanne, "Simulation of FEC-based error control for packet audio on the Internet," presented at IEEE INFOCOM'98, San Francisco, April 1998.

[131] J. L. Ramsey, "Realization of optimum interleavers," in *IEEE Transactions on Information Theory*, 1970, pp. 338-345.

[132] V. Markovski, F. Xue, and Lj Trajkovic, "Simulation and analysis of packet loss in video transfers using User Datagram Protocol," *Supercomputing*, vol. 20 no. 2, pp. 175-196, September 2001.

[133] G. A. Miller and J. C. R. Licklider, "The Intelligibility of Interrupted Speech," *Journal of the Acoustical Society of America*no., pp. 167-173, March 1950.

[134] D. J. Goodman, G. B. Lockhart, O. J. Wasem, and W. -C. Wong, "Waveform substitution techniques for recoverying missing speech segments in packet voice

communications," *IEEE Transactions on Accoustics, Speech , and Signal Processing*, vol. ASSP-34 no. 6, pp. 1440-1148, December 1986.

[135] M. Handley, "An examination of MBone performance", http://www.icir.org/mjh/mbone.ps, Accessed May 2002

[136] R. Warren, *Auditory Perception: A New Synthesis*, ISBN 0-08-025957-x, Pergamon Press, 1982.

[137] A. Watson and M. A. Sasse, "Measuring perceived quality of speech and video in multimedia conferencing applications," presented at Sixth ACM international conference on Multimedia, Bristol, UK, 12th-16th September 1998.

[138] J. G. Gruber and L. Strawczynski, "Subjective effects of varible delay and clipping in dynamically managed voice systems," *IEEE Transactions on Communications*, vol. COM-33 no. 8, pp. 801-808, August 1985.

[139] N. S. Jayant and S. W. Christensen, "Effects of Packet Losses in Waveform Coded Speech and Improvements Due to Odd-Even Sample-Interpolation Procedure," in *IEEE Transactions on Communications*, vol. COM-29, 1981.

[140] R. M. Warren, *Auditory Perception*, ISBN 0-08-025957-x, Pergamon Press Inc, 1982.

[141] ETSI Rec GSM 6.11, "Substitution and muting of lost frames for full rate speech channels," 1992, Available from http://www.3gpp.org/ftp/Specs/html-info/0611.htm.

[142] O. J. Wasen, D. J. Goodman, C. A. Dvorak, and H. G. Page, "The effect of waveform substitution on the quality of PCM packet communication," *IEEE Transactions on Accoustics, Speech , and Signal Processing*, vol. 36 no. 3, pp. 342-348, March 1988.

[143] H. Sanneck, A Stenger, K. Younes, and B. Girod, "A new technique for audio packet loss concealment," presented at IEEE Global Internet, November 1996.

[144] Y. L. Chen and B. S. Chen, "Model-based multirate representation of speech signals and its application to recovery of missing speech packets," *IEEE Transactions on Speech and Audio Processing*, vol. 15 no. 3, pp. 220-231, May 1997.

[145] M. Claypool and J. Tanner, "The Effects of Jitter on the Perceptual Quality of Video," presented at 7th ACM international confernce on Multimedia (Part 2), Orlando, Florida, US, 30th October - 5th November 1999.

[146] B. Lowekamp, N. Miller, R. Karrer, T. Gross, and P. Steenkiste, "Design, Implementation and Evaluation of the Remos Network Monitoring System," *Grid Computing*, vol. 1 no. 1, pp. 75-932003.

[147] P. A. Dinda and D. R. O'Hallaron, "An Extensible Toolkit for Resource Prediction in Distributed Systems," School of Computer Science, Carnegie Mellon University, Technical Report, CMU-CS-99-138, July 1999, Available from http://reports-archive.adm.cs.cmu.edu/anon/1999/CMU-CS-99-138.pdf.

[148] S. Seshan, M. Stemm, and R. H. Katz, "SPAND: Shared Passive Network Performance Discovery," presented at 1st Usenix Symposium on Internet Technologies and Systems (USITS '97), Monterey, CA, December 1997.

[149] M. Zangrilli and B. B. Lowekamp, "Using Passive Traces of Application Traffic in a Network Monitoring System," presented at Thirteenth IEEE Internation Symposium on High Performance Distributed Computing, Honolulu, Hawaii, June 2004.

[150] B. Aboba, "Alernatives for Enhancing RTCP Scalability", internet draft, http://www.cs.columbia.edu/~hgs/rtp/drafts/draft-aboba-rtpscale-01.txt, Accessed 20th June 2002

[151] J. Chesterfield and E. M. Schooler, "An Extensible RTCP Control Framework for Large," presented at The 2nd IEEE International Symposium on Network Computing and Applications (NCA-03), Cambridge, MA, USA, April 16th-18th 2003.

[152] M. Bateman, C. Allison, A. Ruddle, and R. Nicoll, "An Advisory QoS Service for Grid Based Learning Environments," presented at 4th International LEGE-WG Workshop Towards a European Learning Grid Infrastructure Progressing with a European Grid, Stuttgart, Germany, 27th-29th April 2004.

[153] M. Handley, S. Floyd, J Padhye, and J. Wider, "TCP Friendly Rate Control", RFC 3448, January 2003, Available from ftp://ftp.isi.edu/in-notes/rfc3448.txt, Accessed October 2003

[154] "MySQL: The World's Most Popular Open Source Database", web page, http://mysql.org/, Accessed July 2004

[155] D. Borman, S. Deering, and R. Hinden, "IPv6 jumbograms", RFC 2675, August 1999, Available from http://www.faqs.org/rfcs/rfc2675.html, Accessed 7th January 2005

[156] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford, "Dynamics of Hot-Potato Routing in IP Networks," presented at SIGMETRICS/Performance '04, New York, USA, 12th -16th June 2004.

[157] Mozilla, "Rhino - Javascript for Java", web page, http://www.mozilla.org/rhino/, Accessed April 2003

[158] N. Boyd, "Rhino: JavaScript™ Scripting for the Java™ Platform", web page, http://servlet.java.sun.com/javaone/javaone99/pdfs/e629.pdf, Accessed April 2003

[159] Apache Software Foundation, "Jakarta BSF - Bean Scripting Framework", web page, http://jakarta.apache.org/bsf/, Accessed November 2004

[160] W. Jiang and H. Schulzrinne, "Modeling of Packet Loss and Delay and Thier Effect on Real-Time Multimedia Service Quality," presented at The 10th Internetional Workshop on Network and Operating System Support for Digital Audio and Video, North Carolina, USA, June 26-28 2000.

[161] J. Martin, A. Nilsson, and I. Rhee, "Delay-Based Congestion Avoidance for TCP," *IEEE ACM Transactions on Networking*, vol. 11 no. 3, June 2003.

[162] K. Egevang and P. Francis, "The IP Network Address Translator (NAT)", RFC 1631, May 1994, Available from http://www.faqs.org/rfcs/rfc1631.html, Accessed 2005 1st Feburary

[163] D. Senie, "Natwork Address Translator (NAT)-Friendly Application Design Guidelines", RFC 3235, January 2002, Available from http://www.rfc-archive.org/getrfc.php?rfc=3235, Accessed 1st Feburary 2005

[164] M. Holdrege and P. Srisuresh, "Protocol complications with the IP network address translator," RFC 3027, January 2001, Available from http://www.faqs.org/rfcs/rfc3027.html, Accessed 13th March 2003.

[165] R. Droms, "Dynamic Host Configuration Protocol", RFC 2131, March 1997, Available from http://www.faqs.org/rfcs/rfc2131.html, Accessed 28th November 2004

[166] S. B. Moon, P. Skelly, and D. Towsley, "Estimation and Removal of Clock Skew from Network Delay Measurements," presented at The Conference on COmputer Communications Infocom'99, New York, NY, USA, 21st-25th March 1999.

[167] O. Hodson, C. Perkins, and V. Hardman, "Skew detection and compensation for Internet audio applications," presented at International Conference on Multimedia and Expo, New York, NY, USA, 20th July - 2nd August 2000.

[168] "In depth - Jitter", web page, http://www.voiptroubleshooter.com/voiptr_jittersources.htm, Accessed July 2004

[169] "Network performance objectives for IP-based service," ITU-T Y.1541, May 2002

[170] James A Cadzow, *Foundations of Digital Signal Processing and Data Analysis*, ISBN 0023180102, Macmillan USA, 1987.

[171] M. Bateman, C. Allison, and A. Ruddle, "Scenario Driven Network Emulation," presented at The 3rd Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting, Liverpool, UK2002.

[172] M. Bateman, C. Allison, and A. Ruddle, "A Scenario Driven Emulator for Wireless, Fixed and Ad Hoc Networks," presented at The 4th Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (PGNet'03), Liverpool, UK, 2003.

[173] B. White, J. Lepreau, L. Stoller, R. Hibler, C. Barb, and A. Joglekar, "An Integrated Experimental Environment for Distributed Systems and Networks," presented at The Fifth Symposium on Operating Systems Design and Implementation, Boston, MA, December 2002.

[174] "Cisco VLAN Roadmap", web page, http://www.cisco.com/warp/public/538/7.html, Accessed November 2001

[175] B. B. Welch, *Practical Programming in Tcl and Tk*, 3rd ed, ISBN 0130220280, Prentice Hall PTR, 1999.

[176] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu, "Advances in Network Simulation," *IEEE Computer*, vol. 33 no. 5, pp. 59-67, May 2000.

[177] J. Lepreau, C. Alfeld, S. Aggarwal, D. G. Andersen, D. S. Anderson, R. Christensen, A. Clements, J. Duerig, E. Eide, R. Fish, S. Guruprasad, M. Hibler, R. Kempter, M. Newbold, R. Ricci, T. Stack, L. Stoller, and K. Webb, "Emulab.net - Emulab Tutorial", web page, http://www.emulab.net/tutorial/docwrapper.php3?docname=tutorial.html, Accessed April 2003

[178] "NIST Net Home Page", web page, http://snad.ncsl.nist.gov/itg/nistnet/, Accessed November 2001

[179] M. Gaynor, "Proactive Packet Dropping Methods for TCP Gateways", web page, http://www.eecs.harvard.edu/gaynor/final.ps, Accessed March 2004

[180] "NIST Net usage instructions", web page, http://snad.ncsl.nist.gov/itg/nistnet/usage.html, Accessed November 2001

[181] LWN, "LWN - The Linux traffic shaper", web page, http://lwn.net/1998/1119/shaper.html, Accessed May 2004

[182] L. Risso, "Dummynet: A simple approach to the evaluation of network protocols," *ACM Computer Communications Review*, vol. 27 no. 1, December 1997.

[183] "The FreeBSD Project", web page, http://www.freebsd.org/, Accessed September 2004

[184] J. Garcia and A. Brunstrom, "Transport Layer Loss Differentiation and Loss Notification," presented at First Swedish National Computer Networking Workshop, Stockholm, Sweden, 8th-10th September 2003.

[185] J. Nykvist, "On Disputes in Policy Based Path-Vector Routing," presented at IWIN 2003 Internation Workshop on Interconnectin Networks, Umeá, Sweden, 16th-17th June 2003.

[186] A. Balk, D. Maggiorini, M. Gerla, and M. Y. Sanadidi, "Adaptive MPEG-4 Video Streaming with Bandwidth Estimation," presented at Second International Workshop on Quality of Service in a Multiservice IP Networks, Milano, Italy, February 2003.

[187] M. Allman, A. Caldwell, and S. Ostermann, "ONE: The Ohio Network Emulator," School of Electrical Engineering and Computer Science, Ohio University, report, TR-19972, August 1997, Available from http://irg.cs.ohiou.edu/one/.

[188] M. Bateman, C. Allison, and A. Ruddle, "StAnd Net: A Linux Traffic Shaper," presented at Fifth annual postgraduate symposium on the convergence of telecommunications, networking and broadcasting (PGNet'04), Liverpool, UK, 28th - 29th June 2004.

[189] R. Love, "Robert Love Explains Variable HZ", web page, http://kerneltrap.org/node/view/464, Accessed 12th January 2004

[190] R. Russel and H. Welte, "Netfilter hacking HOWTO", web page, http://www.iptables.org/documentation/HOWTO//netfilter-hacking-HOWTO.html, Accessed July 2004

[191] Sun Microsystems Inc, "Java Native Interface", web page, http://java.sun.com/docs/books/tutorial/native1.1/, Accessed November 2004

[192] A. Josey, F. P. Murphy, M. Browm, C. Hughes, B. Bachar, M. Brown, D. Butenhof, D. W. Cragun, L. Dwyer, J. Farley, A. Gollan, K. D. Gordon, G. Miller, F. P. Murphy, F. Prindle, A. K. Roach, C. R. Jr., N. Stoughton, and K. Tsuji, "IEEE Std 1003.1, 2004", http://www.unix.org/single_unix_specification/, Accessed December 2004

[193] J. Heidemann, K. Obraczka, and J. touch, "Modeling the Performance of HTTP Over Several Transport Protocols," *ACM/IEEE Transactions on Networking*, vol. 5 no. 5, pp. 616-630, October 1997.

[194] F. C. R. Bennett and H. Zhang, "WF2Q: Worst-case fair weighted fair queuing," presented at IEEE INFOCOM, San Francisco, CA, USA, 24th -28th March 1996 1996.

[195] J. Drake, "Linux Networking HOWTO", web site, http://tldp.org/HOWTO/Net-HOWTO/, Accessed May 2004

[196] S. Weerawarana, "Bean Scripting Framework (Version 2.1) User's Guide", web page, http://www.alphaworks.ibm.com/tech/bsf/, Accessed 6th July 2002

[197] L. Wall, T. Christiansen, and J. Orwant, *Programming Perl*, 3rd Edition ed, ISBN 0596000278, O'Reilly & Associates, 2000.

[198] P. Lomax and R. Petrusha, *VBScript in a Nutshell*, 2nd Edition ed, ISBN 0596004885, O'Reilly & Associates, 2003.

[199] M. Lutz, *Programming Python*, 2nd Edition ed, ISBN 0596000855, O'Reilly & Associates, 2001.

[200] B. Fenner and Et Al, "mrouted 3-9-beta, mrinfo and other tools", computer program, ftp://ftp.parc.xerox.com/pub/net-research/ipmulti/, Accessed March 2003

[201] P. Parnes, K. Synnes, and D. Schefström, "mTunnel, A multicast tunneling system with a user based Quality-of-Service model," presented at 4th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services, Darmstadt, Germany, 10th-12th September 1997.

[202] S. McCanne and S. Floyd, "The Network Simulator", web page, http://www-nrg.ee.lbl.gov/ns/, Accessed November 2001

[203] R. L. Birdwhistell, *Kinesics and Context: Essays on Body Motion Communication*, ISBN 0812210123, University of Pennsylvania Press, 1970.

[204] J. Laine, S. Saaristo, and R. Prior, "rude", web page, http://rude.sourceforge.net/, Accessed 1st January 2004

[205] J. Harju, T. Alakoski, J. Lemponen, A. Dumitrescu, J. Kuikka, J. Majalainen, S. Saaristo, H. Vatainen, and I. Vesa, "Faster Pro", web site, http://www.atm.tut.fi/faster/, Accessed December 2004

[206] " MGEN - The Multi-Generator Toolset", Web page, http://mgen.pf.itd.nrl.navy.mil/, Accessed May 2005

[207] T. J. Mowbray and R. Zahavi, *The Essential CORBA: Systems Integration Using Distributed Objects*, paperback ed, ISBN 0471106119, John Wiley & Sons, 1995.

[208] "RealNetworks", web site, http://www.realnetworks.com/, Accessed Jan 2005

[209] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson, "The End-to-End Effects of Internet Path Selection," *ACM SIGCOMM Computer Communication Review*, vol. 29 no. 4, pp. 289-299, October 1999.

[210] *Digital RoamAbout 915/2400 DS/PC Card and ISA Network ADapter: Installation and Configuration*, ISBN, DEC, April 1996.

[211] R. Braden, "Requirements for Internet Hosts -- Communication Layers", RFC 1122, October 1989, Available from http://www.faqs.org/rfcs/rfc1122.html, Accessed 1st Feburary 2005

Index

Appendix A

A. Appendix A

A.1. Policy script author's guide

This documents how to write policies for using within the CCA.

A.1.1. Available functions

print (string)

Given a string as the argument this function outputs the string to the terminal. This function is provided in order to aid the testing process.

send (AVInfo)

Given an AVInfo which describes the settings for a session, it sends the session settings via multicast to the participants within the sessions. This method should be used after the sendInitial. The sendInitial should be used for the first session setup.

sendInitial (AVInfo)

Given an AVInfo, it sends the settings via multicast to any client which requests them. The session settings are saved within the CCA and sent to the clients as they request it.

IP join (IP1, IP2, IP3, IP4)

Given the four tuples from an IP address it joins them together into a single 32 bit int and returns it.

IP split (IP)

Given an IP address that is stored as a single 32 bit number it splits the IP address up into its individual tuples and returns them in an array. The numbers are in the order that they appear in an IP address, ie for the IP 138.251.206.158 the array is [138, 251, 206, 158].

String realToVirtual (IP as a String)

Given an IP address from a machine in the beowulf cluster it returns the virtual address assigned to that machine.

addTimer (delay, period, callback)

This function sets up a fixed period call-back. The function 'callback' will be called every period milliseconds starting in delay milliseconds time. An example of the use of this is addTimer (5000, 1000, "printHello"). This use of the addTimer function means that the

printHello function will be called every 1 second starting in 5 seconds time. It is advised that this function is not used with low periods, ie less than 1 second.

total (measurements list [, sourceIP, destIP])

This function returns the total of all the measurements given. So for example if this were called as total (measurements.bandwidth) it would add up all the bandwidth measurements and return total. The optional elements of the function; sourceIP and destIP, allow the total to be restricted to only the measurements that are from the source to the destination. The IP addresses if provided should be in the packed format.

min (measurement list [, sourceIP, destIP])

This function returns the lowest measurement from the list of measurements given. The optional elements; sourceIP and destIP are used to restrict the list of measurements to be searched to only those measurements that match the source and destination IPs given. The IP address if provided should be in the packed format.

max (measurement list [, sourceIP, destIP])

This function return the highest measurement from the list of measurements provided. The optional elements; sourceIP and destIP allow the function to be restricted to only the elements which have the source and destination IP addresses that are the same as sourceIP and destIP. The IP address, if provided, should be in the packed format.

count (measurement list [, sourceIP, destIP])

This function returns the number of elements in the supplied measurements list. It should not be used without the sourceIP and destIP parameters since that information is available from the measurements.<type>.size property (where type is one of bandwidth, delay, jitter or congestion). The sourceIP and destIP parameter limit the count to only the measurements which match the source and destination IP addresses.

delayTobps (delay, packetSize, packetLoss)

This function return the bits per second throughput for a TCP stream given the delay, packet size and packet loss.

avgmax (measurement list [, sourceIP, destIP])

This function return the highest running weighted average measurement from the list of measurements provided. The optional elements; sourceIP and destIP allow the function to be restricted to only the elements which have the source and destination IP addresses that are the same as sourceIP and destIP. The IP address, if provided, should be in the packed format.

avgmin (measurement list [, sourceIP, destIP])

This function return the lowest running weighted average measurement from the list of measurements provided. The optional elements; sourceIP and destIP allow the function to be restricted to only the elements which have the source and destination IP addresses that are the same as sourceIP and destIP. The IP address, if provided, should be in the packed format.

A.1.2. Standard function names

function trafficReports (startDate, endData,srcIP, destIP, bandwidth, variance, delay, dropped, packet size, total, jitter)

This is a standard method which should be implemented by all policies that wish to receive information about the current state of the network.

function AVInfo initialize (settings, measurements)

This is the function that is called before any other. It returns an AVInfo object which should contain the initial settings to be used within the session.

It is assumed that no processing for the setup or maintenance of session will be performed outside of the two functions that have been described.

A.1.3. Available objects

A.1.3.1. AVInfo object

The AVInfo object is a holder object which is used when transporting the session settings information from the policy to the participant agents. The object is created in the normal way as shown in Figure 51.

```
var avinfo = new AVInfo ();
```

Figure 51 Creation of an AVInfo object

From this point onwards the AVInfo object differs from normal objects. The AVInfo object does not have any set attributes and can therefore be customised to suit any application. If a value is assigned to an attribute then the attribute is created and can later to accessed. For example if you wish an attribute called videoBitRate to be part of an AVInfo object then you would use the code in shown in Figure 52.

```
avinfo.videoBitRate = 50000;
```

Figure 52 Creation of an attribute

This method of creating attributes for the object when they are requested means that this system is extensible and able to support applications which were not considered. For example the work carried out addresses videoconferencing applications and therefore an AVInfo object is used to carry information relating to audio and video CODECs as well as colour depths and bit rates, but equally the system could be used with multiuser document editing and as such would need to set information relating to awareness widgets, the granularity of locks and the timeliness of updates. The only restriction on attributes of the AVInfo object (other than those imposed by Javascript) are that the Participant Agent should know how to interpret the names into appropriate settings.

The following attributes for the audio video conference system have been implemented.

videoFormat

This controls the video CODEC that is in use and is a string. Valid settings for this parameter are

1. h263 – For the H263 for using in RTP.

2. jpeg – For motion jpeg when used with RTP.

audioFormat

This controls the audio CODEC that is in use and is a string. Valid settings for this parameter are

1. gsm – For the GSM audio CODEC.

videoFrameRate

This controls the frame rate for the video and it is a number and should be in the range 0 to 30. Less than 0 is meaningless. 30 was chosen as the maximum since this is the highest frame-rate that is used in television (29.97 is used for NTSC video). The actual numbers that the frame-rate can be is limited by the CODEC.

audioSampleRate

This controls the sample rate (per second) of the audio CODEC and should be greater than one. The maximum sample rate is dependant upon the audio CODEC in use.

videoBitRate

This controls the number of bits per second that the video CODEC will attempt to encode to. The minimum and maximum values for this are dependant upon the video CODEC that is in use. It is worth noting, when writing policies that this setting is not always followed exactly and there might be a slight difference in the bit rate set and the bit rate that is measured.

audioBitrate

This controls the number of bits per second that the audio CODEC will attempt to encode to. The minimum and maximum values are dependant upon the audio CODEC in use. It is worth noting, when writing policies that this setting is not always followed exactly and there might be a slight different in the bit rate set and the bit rate that is measured.

videoMulticastAddr

This is the multicast address of the video traffic for the session. The address should be in the standard IP tuple format of ww.xx.yy.zz.

videoMulticastPort

This is the port number of the video traffic for the session. The port number should be above 1024 since these ports are not privileged and therefore do not require special access to be used.

audioMulticastAddr

This is the multicast address of the audio traffic for this session. The address should be in the standard IP tuple format of ww.xx.yy.zz.

audioMulticastPort

This is the port number of the audio traffic for the session. The port number should be above 1024 since these ports are not privileged and therefore do not require special access to be used.

videoSilenceSuppression

This is a boolean value. This setting controls if the video from a given user will always be sent or if the video will only be sent while the user is talking. A setting of true enables this feature while a setting of false or the absence of this parameter indicates that this feature should be turned off. This feature is useful for use on low bandwidth paths.

audioSilenceSuppression

This is a boolean value. This setting controls if the audio from a given user will always be sent or if the audio will only be sent while the user is talking. A setting of true enables this feature while a setting of false or the absence of this parameter indicates that this feature should be turned off. This feature is useful for use on low bandwidth paths or as a method to control audio feedback.

virtualVideo

This setting controls if a video CODEC should be used to send the live video stream or if a virtual representation of the person should be used. This is a boolean value and a setting of true indicates the use of a virtual representation of the person while a setting of false or the absence of the parameter means that this feature should be disabled.

virtualAudio

This setting controls if the audio CODEC should be used to send the live audio data or if speech recognition and speech synthesis should be used. This setting should only be used when there is a need for extremely low bandwidth communications since a large delay is introduced due to the speech recognition and speech synthesis process.

videoOn

This setting controls whether video should be sent. It is a boolean value where true or the absence of the parameter means that the video should be sent whereas a value of false means that the video shouldn't be sent.

audioOn

This setting controls whether video should be sent. It is a boolean value where true or the absence of the parameter means that the video should be sent whereas a value of false means that the video shouldn't be sent.

canAlterVideo

This settings controls if the user is allowed to turn on or off their video stream. This is a boolean value and a setting of true or the absence of the parameter means that the user is allowed to turn off their video stream, and a setting of false means that the video stream should always be on.

canAlterAudio

This settings controls if the user is allowed to turn on or off their audio stream. This is a boolean value and a setting of true or the absence of the parameter means that the user is allowed to turn off their audio stream, and a setting of false means that the audio stream should always be on.

speechRecognition

This setting controls the use of speech recognition as a method of transmitting speech. Speech recognition is used to capture the phonemes from the user. These phonemes are then played back to the receiver.

videoHeight

This is the height in pixels that the video should be encoded to. This is an integer value. Valid values for this attribute are dependant upon the video width and the video CODEC that is in use. The default value for this is 144 if the parameter is missing or if the value cannot be recognised as an integer.

videoWidth

This is the width in pixels that the video should be encoded to. This is an integer value. Valid values for this parameter are dependant upon the video height and the video CODEC that is in use. The default value for this is 176 if the parameter is missing or if the value cannot be recognised as an integer.

A.1.3.2. AVGTracker

The AVGTracker provides a simple and easy way to maintain a running average over time.

add (src, dest, data)

Where src is the source IP address, dest is the destination IP address and data is the number which should be added to the running average. This adds a measurement to the queue defined by src and dest. This method doesn't return anything.

min (src, dest)

Where src is the source IP address and dest is the destination IP address. This method returns the minimum measurement in the current queue.

max (src, dest)

Where src is the source IP address and dest is the destination IP address. This method returns the maximum measurement in the current queue.

avg (src, dest)

Where src is the source IP address and dest is the destination IP address. This method returns the running weighted average of the measurements which have been added.

A.1.3.3. JSTimer

The JSTimer provides substitute for a standard java timer. This allows the periodic execution of a script function. It is used in the following manner. On creation it is passed a delay, period and the name of a function. The delay is how long before the period execution of the function should be started. The period gives the amount of time to wait between calls to the function. Finally function is the name of the function to call. The function does not take any parameters. All time are specified in milliseconds. An example of this object's use is shown below.

```
new JSTimer (1000, 5000, "example");

function example () {
  print ("Example now being called");
}
```

Figure 53 Example use of a JSTimer

In the example shown in Figure 53 the JSTimer waits one second and then calls the function 'example', every five seconds thereafter the function 'example' is called.

Appendix B

B. Appendix B

B.1. CCA settings DTD

The CCA encodes the setting configuration for a session into an XML format. The DTD is shown below

<!ELEMENT ACTIVE (AUDIO, VIDEO) >

<!ELEMENT AUDIO (BITRATE?, CODEC?, REPAIR?, SAMPLERATE?, STEARO?,

SILENCESUPPRESSION?, TRANSMIT?, RECEIVE?, SPEECHRECOGNITION?) >

<!ATTLIST AUDIO ADDRESS NMTOKEN #IMPLIED >

<!ATTLIST AUDIO PORT NMTOKEN #IMPLIED >

<!ELEMENT BITRATE (#PCDATA) >

<!ELEMENT CODEC (#PCDATA) >

<!ELEMENT COLOURDEPTH EMPTY >

<!ELEMENT COLOURSPACE EMPTY >

<!ELEMENT FRAMERATE (#PCDATA) >

<!ELEMENT PARTICIPANT (AUDIO, VIDEO, ACTIVE, PASSIVE) >

<!ATTLIST PARTICIPANT ADDRESS CDATA #REQUIRED >

<!ELEMENT PASSIVE (AUDIO, VIDEO) >

<!ELEMENT RECEIVE (#PCDATA) >

<!ELEMENT REPAIR EMPTY >

<!ELEMENT SAMPLERATE (#PCDATA) >

<!ELEMENT SETTINGS (PARTICIPANT+) >

<!ELEMENT SILENCESUPPRESSION (#PCDATA) >

<!ELEMENT SPATIALRESOLUTION EMPTY >

<!ATTLIST SPATIALRESOLUTION HEIGHT NMTOKEN #REQUIRED >

<!ATTLIST SPATIALRESOLUTION WIDTH NMTOKEN #REQUIRED >

<!ELEMENT SPEECHRECOGNITION (#PCDATA) >

<!ELEMENT STEARO (#PCDATA) >

<!ELEMENT TRANSMIT (#PCDATA) >

<!ELEMENT VIDEO (BITRATE?, CODEC?, REPAIR?, FRAMERATE?,

COLOURDEPTH?, COLOURSPACE?, SPATIALRESOLUTION?,

SILENCESUPPRESSION?, TRANSMIT?, RECEIVE?) >

<!ATTLIST VIDEO ADDRESS NMTOKEN #IMPLIED >

<!ATTLIST VIDEO PORT NMTOKEN #IMPLIED >

Appendix C

C. Appendix C

C.1. StAnd Net packet drop measurements

Set drop (%)	Measured drop (%)	Set drop (%)	Measured drop (%)	Set drop (%)	Measured drop (%)	Set drop (%)	Measured drop (%)
1	2.7	26	27.3	51	52.9	76	78.5
2	2.4	27	29.5	52	53	77	78.1
3	4.9	28	29	53	55	78	80.4
4	4.5	29	27.2	54	54.4	79	80.8
5	6.4	30	31.2	55	55.2	80	81.7
6	7.6	31	33.1	56	58.7	81	82.6
7	7.8	32	33.1	57	60.3	82	82.3
8	8.3	33	37.6	58	60.4	83	86.8
9	8.2	34	37.6	59	60.5	84	84.7
10	11.4	35	36.5	60	58.1	85	84.4
11	12.1	36	37.5	61	61.1	86	85.9
12	12.9	37	35.3	62	62.2	87	89.3
13	13.1	38	38.9	63	65.2	88	88.5
14	15.1	39	39	64	62.1	89	89.2
15	16.6	40	40.3	65	66.8	90	92
16	18.5	41	42.7	66	67.3	91	91.7

17	18.3	42	39.4	67	68.7	92	92.9
18	17.9	43	42.9	68	67.7	93	92.5
19	21.1	44	45.5	69	69.3	94	94.3
20	22.9	45	47.4	70	69.9	95	95.7
21	20.6	46	45.9	71	72.5	96	97.5
22	22.9	47	48.1	72	72.5	97	98
23	24.3	48	51	73	71.8	98	98.9
24	23.7	49	50.3	74	75.7	99	100
25	27	50	49	75	76	100	100

C.2. StAnd Net delay measurements

Set delay (ms)	Measured delay (ms)	Set delay (ms)	Measured delay (ms)	Set delay (ms)	Measured delay (ms)	Set delay (ms)	Measured delay (ms)
1	1.124	51	51.121	101	101.116	151	151.114
2	2.121	52	52.117	102	102.121	152	152.118
3	3.118	53	53.118	103	103.115	153	153.120
4	4.122	54	54.109	104	104.117	154	154.117
5	5.115	55	55.121	105	105.115	155	155.121
6	6.116	56	56.119	106	106.115	156	156.119
7	7.121	57	57.117	107	107.119	157	157.113
8	8.119	58	58.114	108	108.117	158	158.116
9	9.115	59	59.111	109	109.117	159	159.121
10	10.120	60	60.119	110	110.114	160	160.114
11	11.120	61	61.116	111	111.113	161	161.118
12	12.117	62	62.114	112	112.114	162	162.118
13	13.112	63	63.119	113	113.118	163	163.116
14	14.119	64	64.121	114	114.117	164	164.118
15	15.120	65	65.118	115	115.111	165	165.121
16	16.114	66	66.118	116	116.119	166	166.116

17	17.118	67	67.121	117	117.120	167	167.111
18	18.118	68	68.112	118	118.116	168	168.117
19	19.120	69	69.114	119	119.121	169	169.114
20	20.117	70	70.119	120	120.118	170	170.116
21	21.117	71	71.117	121	121.119	171	171.117
22	22.120	72	72.118	122	122.121	172	172.121
23	23.122	73	73.112	123	123.115	173	173.121
24	24.110	74	74.121	124	124.117	174	174.119
25	25.113	75	75.121	125	125.117	175	175.114
26	26.120	76	76.121	126	126.121	176	176.116
27	27.123	77	77.118	127	127.114	177	177.117
28	28.118	78	78.119	128	128.116	178	178.113
29	29.119	79	79.116	129	129.116	179	179.121
30	30.116	80	80.119	130	130.119	180	180.116
31	31.117	81	81.120	131	131.121	181	181.112
32	32.114	82	82.119	132	132.117	182	182.119
33	33.114	83	83.121	133	133.113	183	183.117
34	34.119	84	84.117	134	134.113	184	184.112
35	35.115	85	85.118	135	135.118	185	185.119
36	36.111	86	86.112	136	136.117	186	186.113
37	37.116	87	87.113	137	137.117	187	187.116
38	38.117	88	88.117	138	138.120	188	188.119
39	39.120	89	89.117	139	139.119	189	189.114
40	40.114	90	90.112	140	140.118	190	190.113
41	41.118	91	91.117	141	141.121	191	191.121
42	42.115	92	92.115	142	142.118	192	192.118
43	43.120	93	93.117	143	143.122	193	193.116
44	44.113	94	94.119	144	144.117	194	194.114
45	45.109	95	95.119	145	145.113	195	195.114
46	46.115	96	96.119	146	146.115	196	196.116
47	47.112	97	97.119	147	147.116	197	197.116

48	48.118	98	98.118	148	148.121	198	198.118
49	49.111	99	99.113	149	149.118	199	199.120
50	50.119	100	100.116	150	150.115	200	200.119

Appendix D

D. Appendix D

D.1. Constant loaded network

D.1.1. SDNE configuration script

```
use EmuNetwork;
use EmuNode;

my $network = new EmuNetwork (2.5, 1800);

# Set the router of the system
$network->router ("nog31");

my $node1 = new EmuNode ('host' => 'nog32',
                  'type' => '100',
                   'talkative' => '16');
my $node2 = new EmuNode ('host' => 'nog33',
                  'type' => '10',
                  'talkative' => '16');
my $node3 = new EmuNode ('host' => 'nog34',
                  'type' => 'isdn',
                  'talkative' => '16');
my $node4 = new EmuNode ('host' => 'nog35',
                  'type' => 'adsl',
                  'talkative' => '16');
my $node5 = new EmuNode ('host' => 'nog36',
                  'type' => 'isdn',
                  'talkative' => '16');
# convener
my $node6 = new EmuNode ('host' => 'nog37',
```

```
                'type' => '100',
                'talkative' => '20');

# Add the nodes to the network
$network->add ($node1);
$network->add ($node2);
$network->add ($node3);
$network->add ($node4);
$network->add ($node5);
$network->add ($node6);

# Create the network
$network->run ();
```

D.1.2. RUDE configuration script

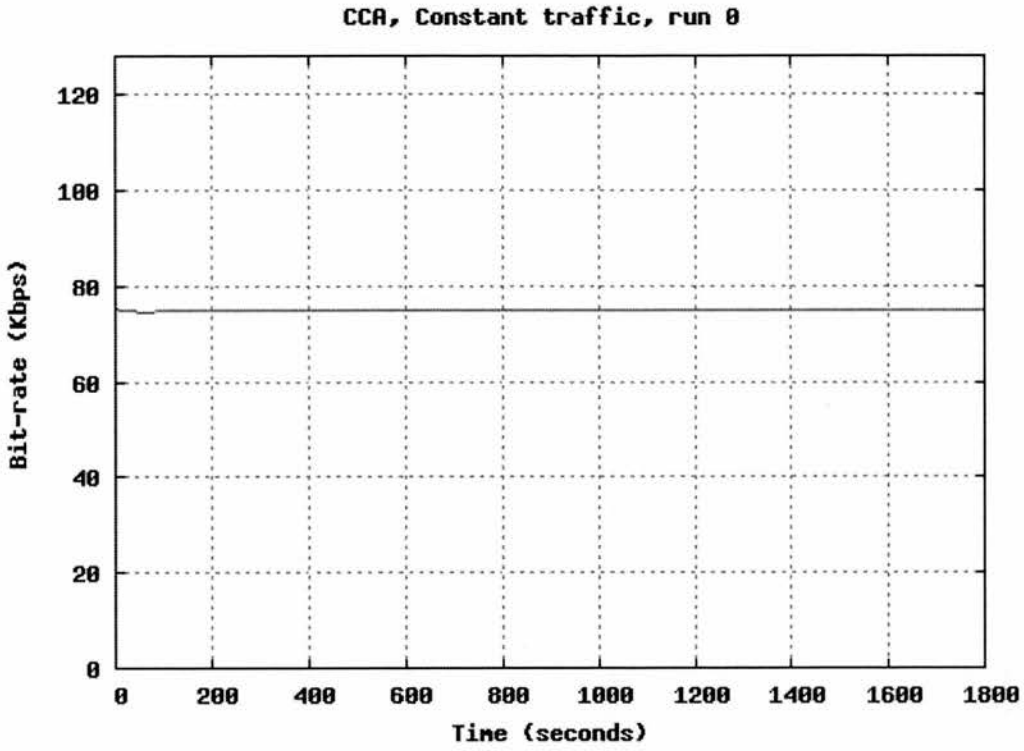
```
START NOW
0000 0030 ON 3002 224.1.2.3:10001 CONSTANT 31 125
1810000 0030 OFF
```

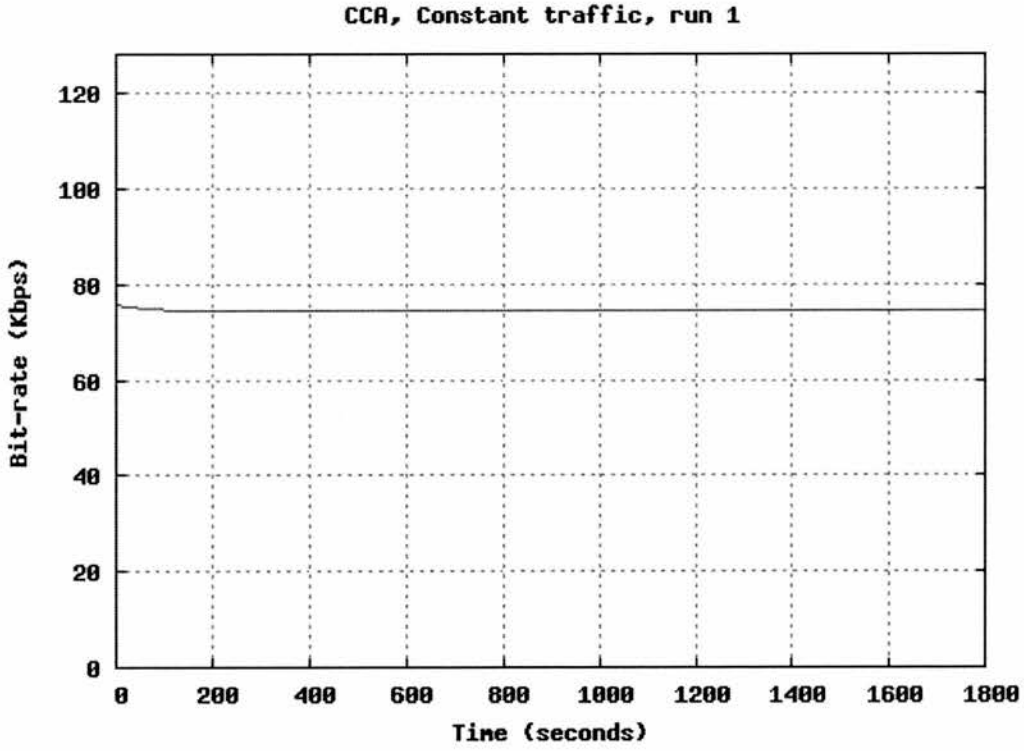
D.1.3. CCA policy script

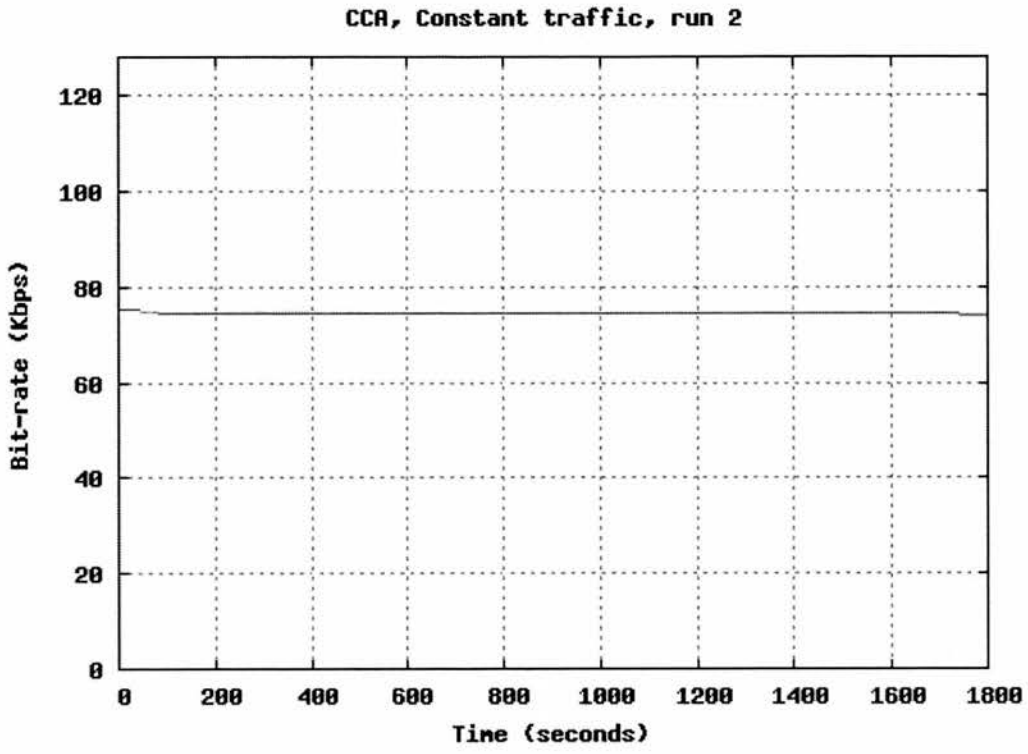
```
var avinfo  = new AVInfo ();
var avgbandwidth = new AVGTracker ();
var avgdelay = new AVGTracker ();
var avgdropped = new AVGTracker ();

function trafficReport (startDate, endDate, srcIP, destIP, bandwidth, variance,delay,
dropped, total, jitter) {
 avgdelay.add (srcIP, destIP, delay);
 avgdropped.add (srcIP, destIP, dropped);
 avgbandwidth.add (srcIP, destIP, bandwidth);

 var eqBandwidth = delayTobps (avgdelay.avg (srcIP, destIP), 1500, avgdropped.avg (srcIP,
destIP));

 var totalBW = avinfo.videoBitrate + avinfo.audioBitrate;

 if (eqBandwidth < totalBW) { // problem
   avinfo.videoBitrate = eqBandwidth - (avinfo.audioBitrate);
   send (avinfo);
   }
}

function AVInfo initialise (setting, measurements) {
 var bandwidth = measurements.bestguess.bandwidth;
 var delay = measurements.bestguess.delay;
 var jitter = measurements.bestguess.jitter;
 var loss = measurements.bestguess.packetloss;

 var noParticipants = settings.groupSize;
 var noActiveParticipants = settings.noTutors;

 var eqBandwidth = delayTobps (delay, 1500, loss);

 var bw = 0;
 if (eqBandwidth < bandwidth) {
  bw = eqBandwidth;
  } else {
  bw = bandwidth;
  }

 if (eqBandwidth > (bandwidth + 128000)) { // jump up slowly
  bw = bandwidth + 128000;
  }
```

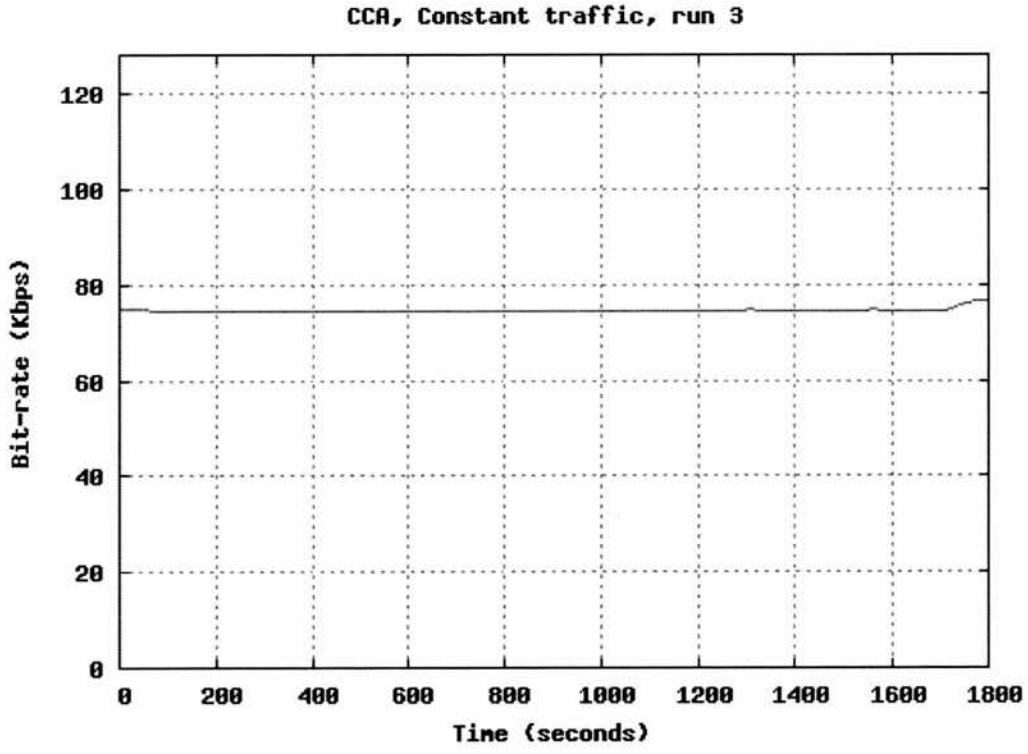
```
avinfo = configureAVInfo (bw, delay, jitter, loss);
return (avinfo);
}
```

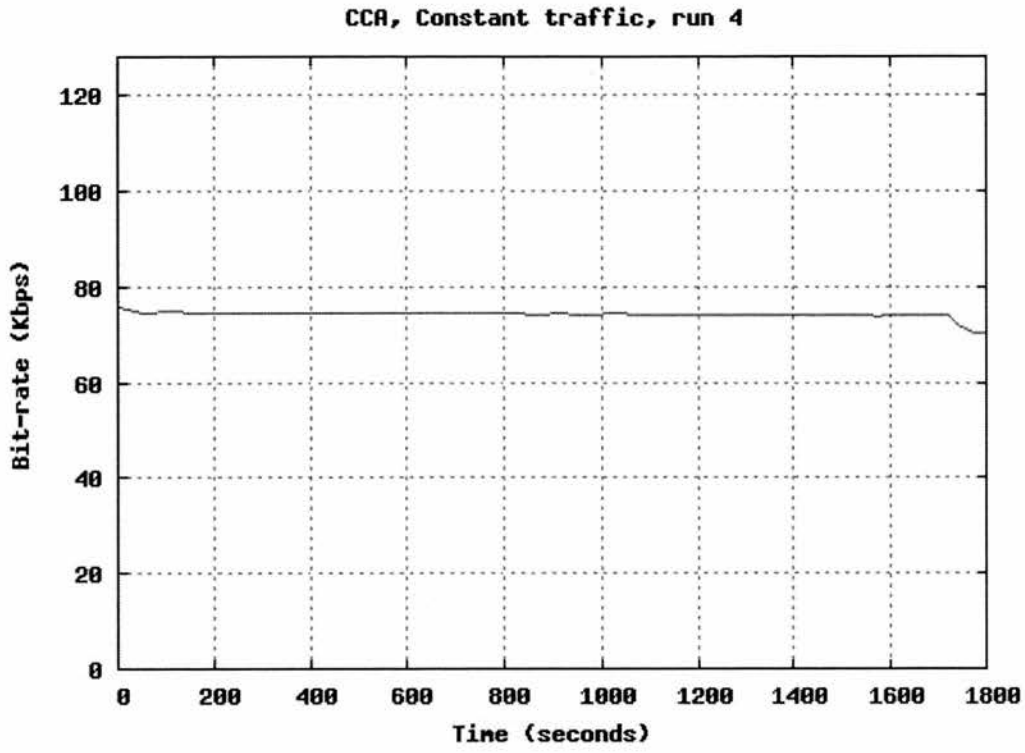
D.1.4. Results

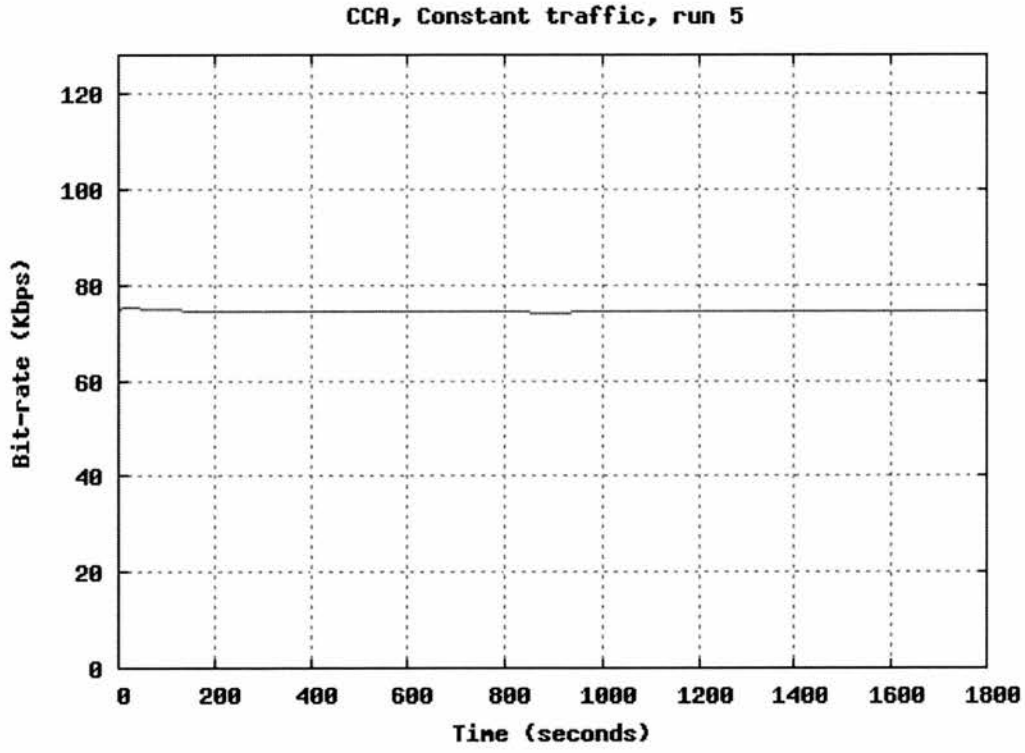

CCA, Constant traffic, run 0
Bit-rate (Kbps)
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)

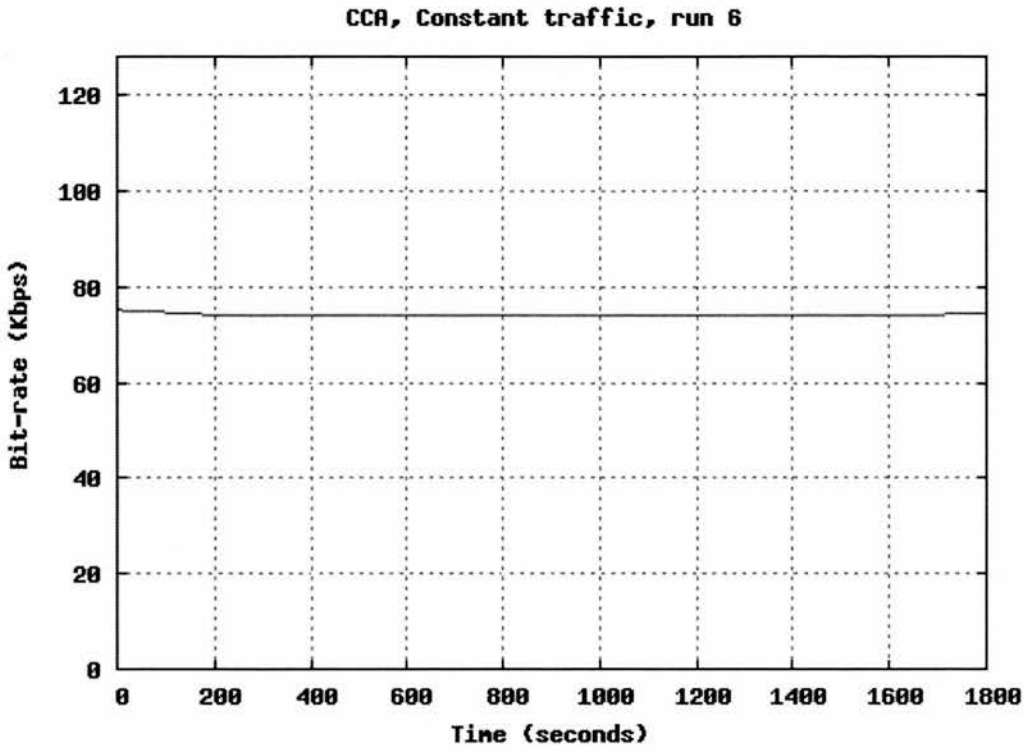

CCA, Constant traffic, run 1
Bit-rate (Kbps)
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)

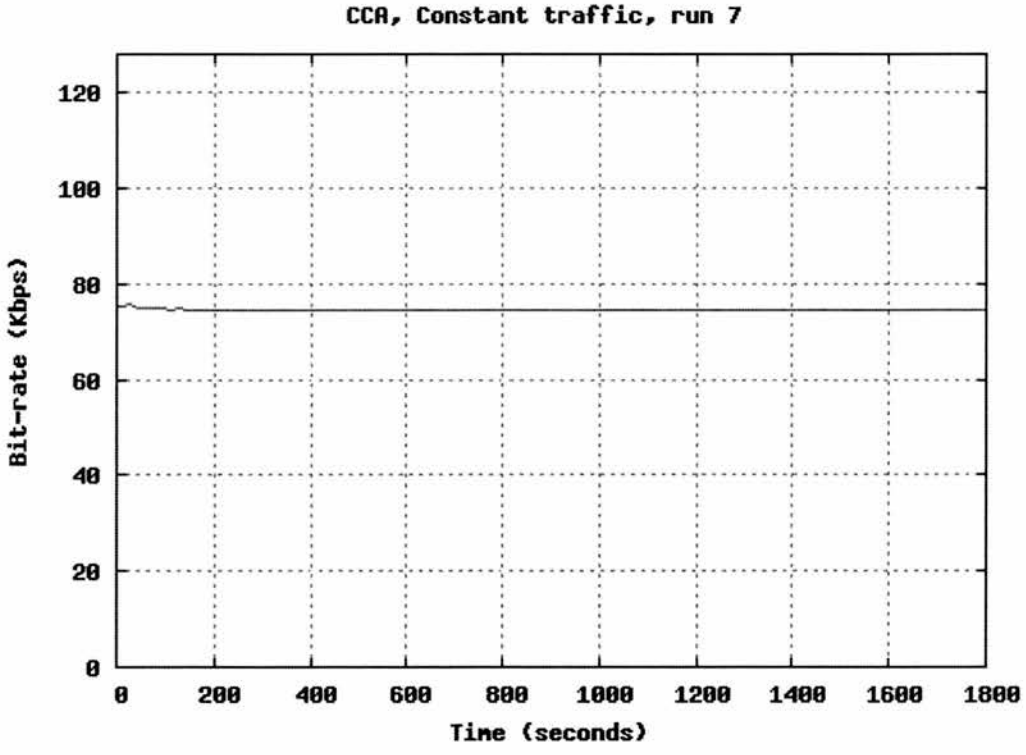

CCA, Constant traffic, run 2
Bit-rate (Kbps)
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)

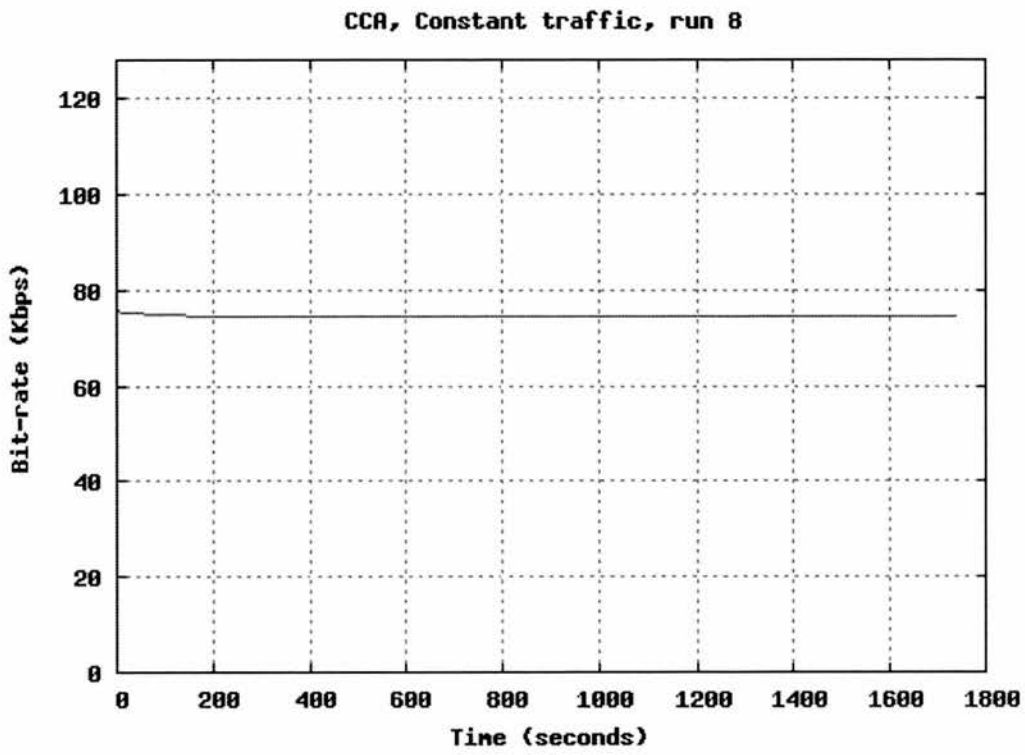

CCA, Constant traffic, run 3
Bit-rate (Kbps)
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)


CCA, Constant traffic, run 4
Bit-rate (Kbps)
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)


CCA, Constant traffic, run 5
Bit-rate (Kbps)
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)


CCA, Constant traffic, run 6
Bit-rate (Kbps)
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)


CCA, Constant traffic, run 7
Bit-rate (Kbps)
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)



D.2. Increasing network load

D.2.1. SDNE configuration script

```
use EmuNetwork;
use EmuNode;

my $network = new EmuNetwork (2.5, 1800);

# Set the router of the system
$network->router ("nog31");

my $node1 = new EmuNode ('host' => 'nog32',
                  'type' => '100',
                   'talkative' => '16');
my $node2 = new EmuNode ('host' => 'nog33',
                  'type' => '10',
                  'talkative' => '16');
my $node3 = new EmuNode ('host' => 'nog34',
                  'type' => 'adsl',
                  'talkative' => '16');
my $node4 = new EmuNode ('host' => 'nog35',
                  'type' => 'adsl',
```

```
                    'talkative' => '16');
my $node5 = new EmuNode ('host' => 'nog36',
                    'type' => 'isdn',
                    'talkative' => '16');
# convener
my $node6 = new EmuNode ('host' => 'nog37',
                    'type' => '100',
                    'talkative' => '20');

# Add the nodes to the network
$network->add ($node1);
$network->add ($node2);
$network->add ($node3);
$network->add ($node4);
$network->add ($node5);
$network->add ($node6);

# Create the network
$network->run ();
```

D.2.2. RUDE configuration script

```
START NOW
0000 0030 ON 3002 224.1.2.3:10001 CONSTANT 1 125
60000 0030 MODIFY CONSTANT 2 125
119000 0030 MODIFY CONSTANT 3 125
178000 0030 MODIFY CONSTANT 4 125
237000 0030 MODIFY CONSTANT 5 125
296000 0030 MODIFY CONSTANT 6 125
355000 0030 MODIFY CONSTANT 7 125
414000 0030 MODIFY CONSTANT 8 125
473000 0030 MODIFY CONSTANT 9 125
532000 0030 MODIFY CONSTANT 10 125
591000 0030 MODIFY CONSTANT 11 125
650000 0030 MODIFY CONSTANT 12 125
709000 0030 MODIFY CONSTANT 13 125
768000 0030 MODIFY CONSTANT 14 125
827000 0030 MODIFY CONSTANT 15 125
886000 0030 MODIFY CONSTANT 16 125
945000 0030 MODIFY CONSTANT 17 125
1004000 0030 MODIFY CONSTANT 18 125
1063000 0030 MODIFY CONSTANT 19 125
1122000 0030 MODIFY CONSTANT 20 125
1181000 0030 MODIFY CONSTANT 21 125
1240000 0030 MODIFY CONSTANT 22 125
1299000 0030 MODIFY CONSTANT 23 125
```

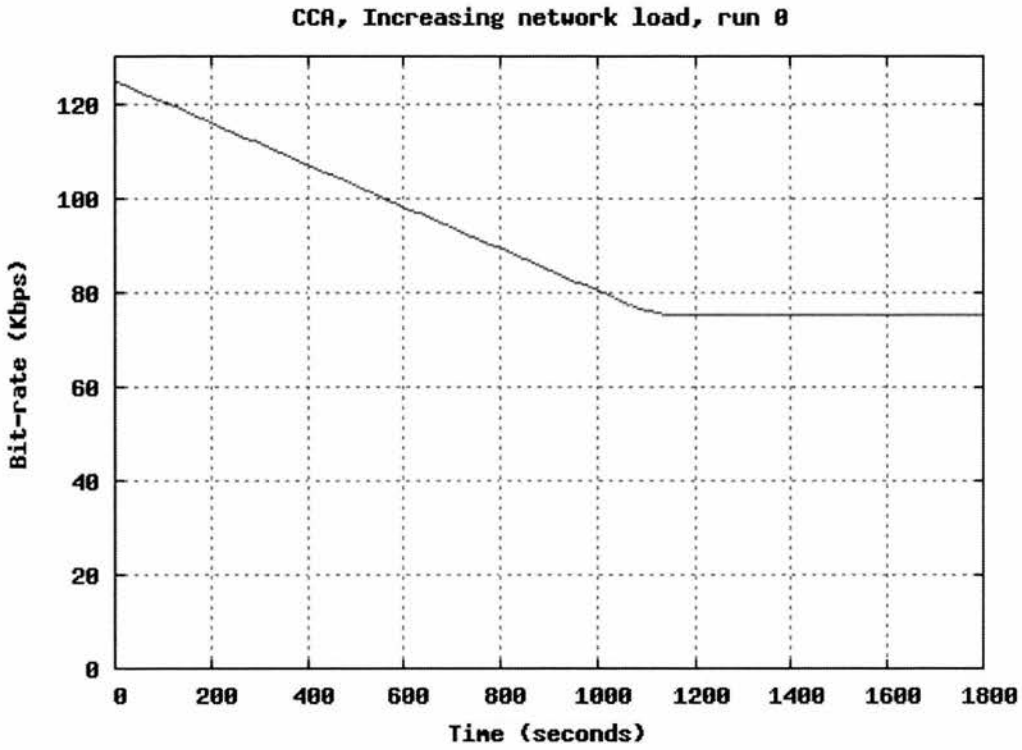
1358000 0030 MODIFY CONSTANT 24 125
1417000 0030 MODIFY CONSTANT 25 125
1476000 0030 MODIFY CONSTANT 26 125
1535000 0030 MODIFY CONSTANT 27 125
1594000 0030 MODIFY CONSTANT 28 125
1653000 0030 MODIFY CONSTANT 29 125
1712000 0030 MODIFY CONSTANT 30 125
1771000 0030 MODIFY CONSTANT 31 125
1810000 0030 OFF

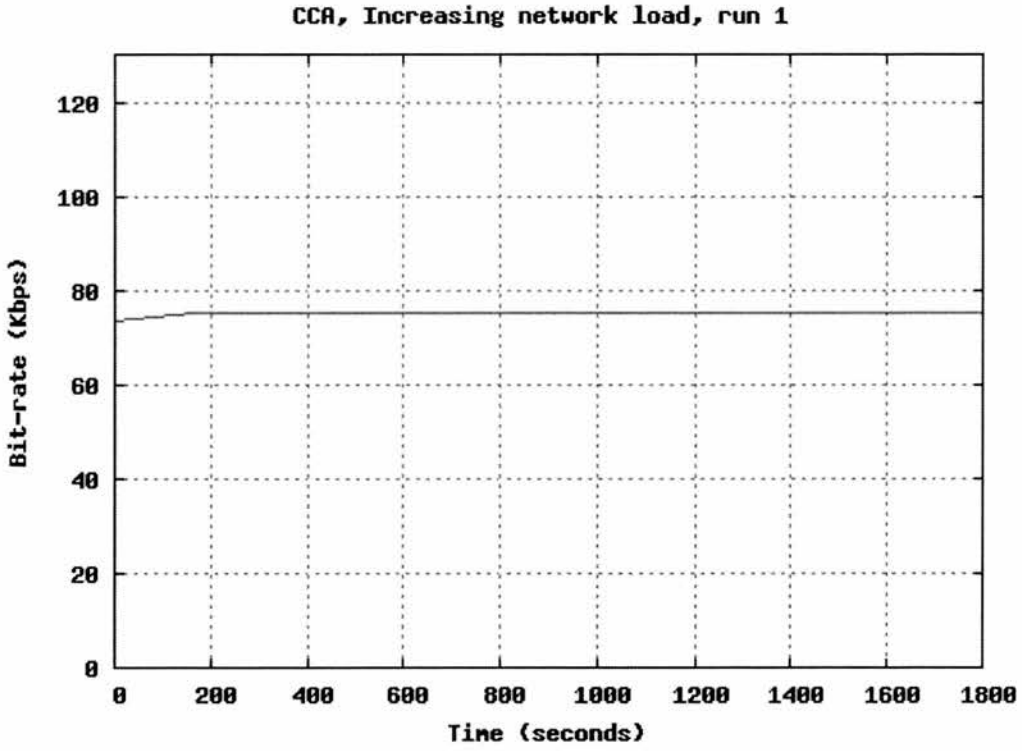
D.2.3. CCA policy script

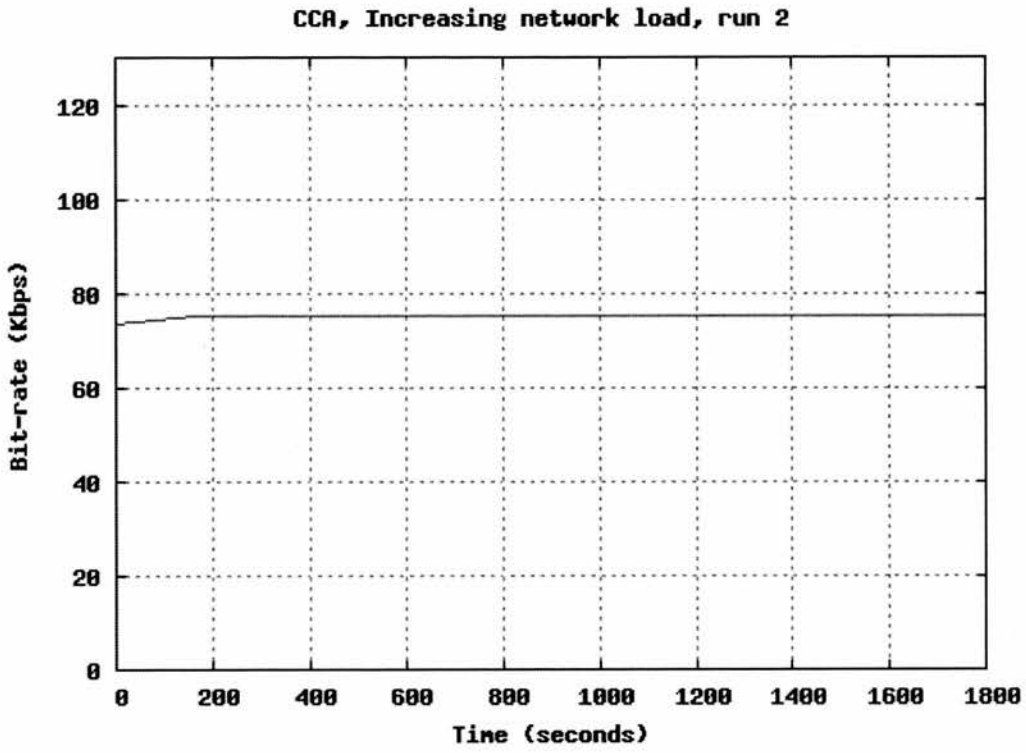
```
var avinfo  = new AVInfo ();
var avgbandwidth = new AVGTracker ();
var avgdelay = new AVGTracker ();
var avgdropped = new AVGTracker ();

function trafficReport (startDate, endDate, srcIP, destIP, bandwidth, variance,delay,
dropped, total, jitter) {
 avgdelay.add (srcIP, destIP, delay);
 avgdropped.add (srcIP, destIP, dropped);
 avgbandwidth.add (srcIP, destIP, bandwidth);

 var eqBandwidth = delayTobps (avgdelay.avg (srcIP, destIP), 1500, avgdropped.avg (srcIP,
destIP));

 var totalBW = avinfo.videoBitrate + avinfo.audioBitrate;

 if (eqBandwidth < totalBW) { // problem
  avinfo.videoBitrate = eqBandwidth - (avinfo.audioBitrate);
  send (avinfo);
  }
}

function AVInfo initialise (setting, measurements) {
 var bandwidth = measurements.bestguess.bandwidth;
 var delay = measurements.bestguess.delay;
 var jitter = measurements.bestguess.jitter;
 var loss = measurements.bestguess.packetloss;

 var noParticipants = settings.groupSize;
 var noActiveParticipants = settings.noTutors;

 var eqBandwidth = delayTobps (delay, 1500, loss);

 var bw = 0;
 if (eqBandwidth < bandwidth) {
  bw = eqBandwidth;
  } else {
  bw = bandwidth;
  }

 if (eqBandwidth > (bandwidth + 128000)) { // jump up slowly
  bw = bandwidth + 128000;
  }
```

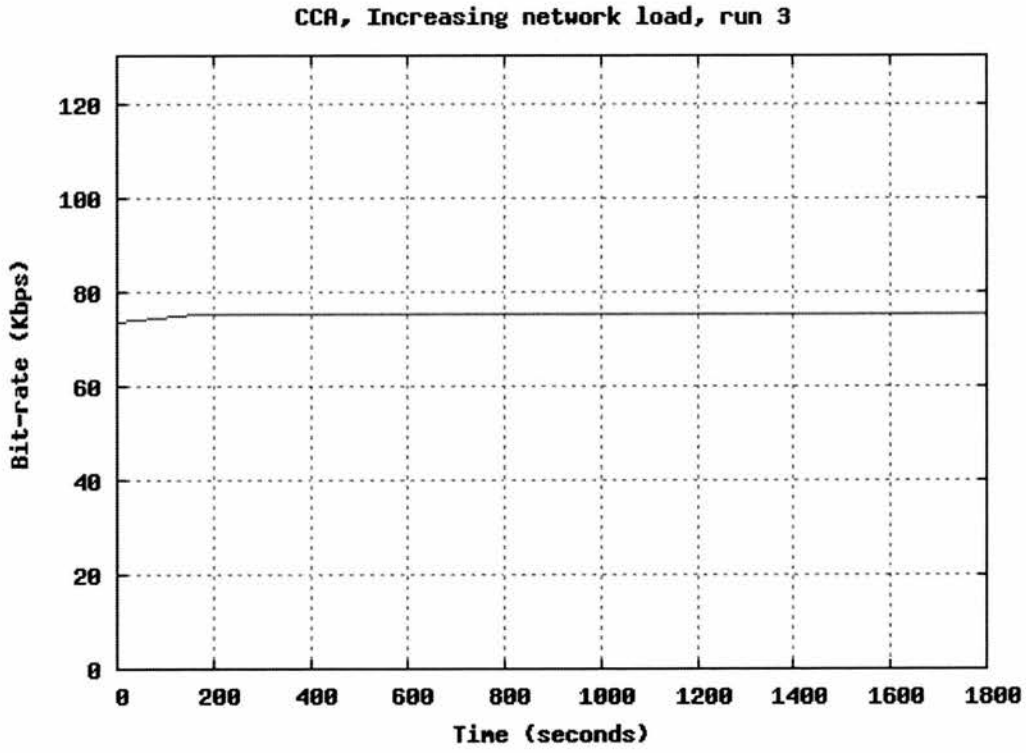
```
avinfo = configureAVInfo (bw, delay, jitter, loss);
return (avinfo);
}
```

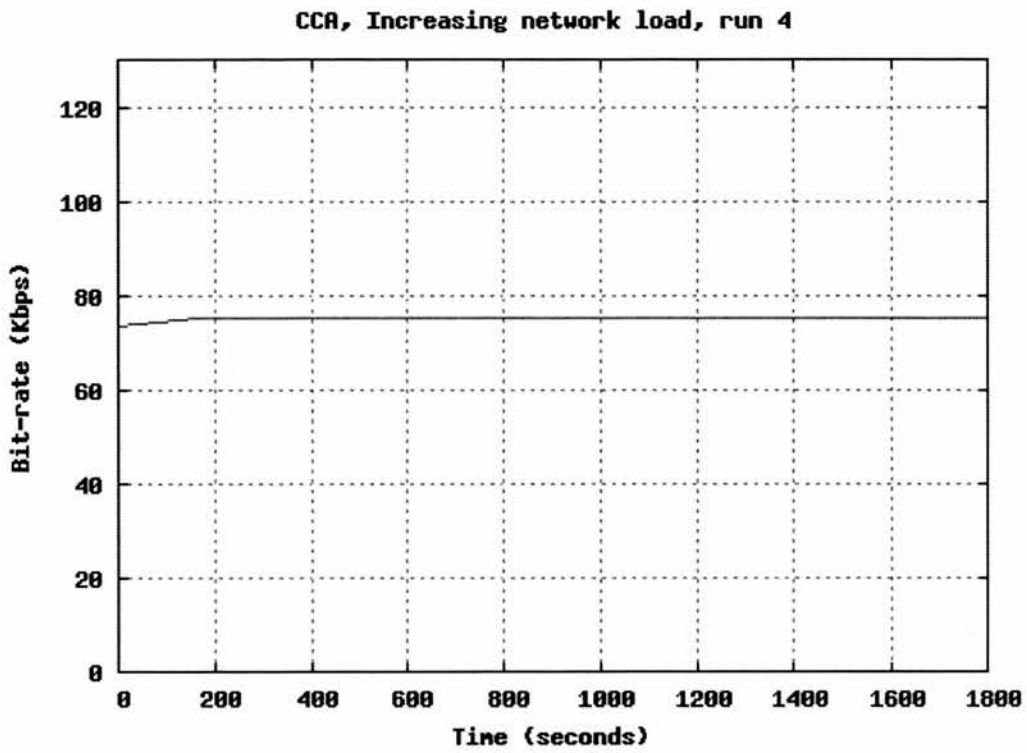
D.2.4. Results

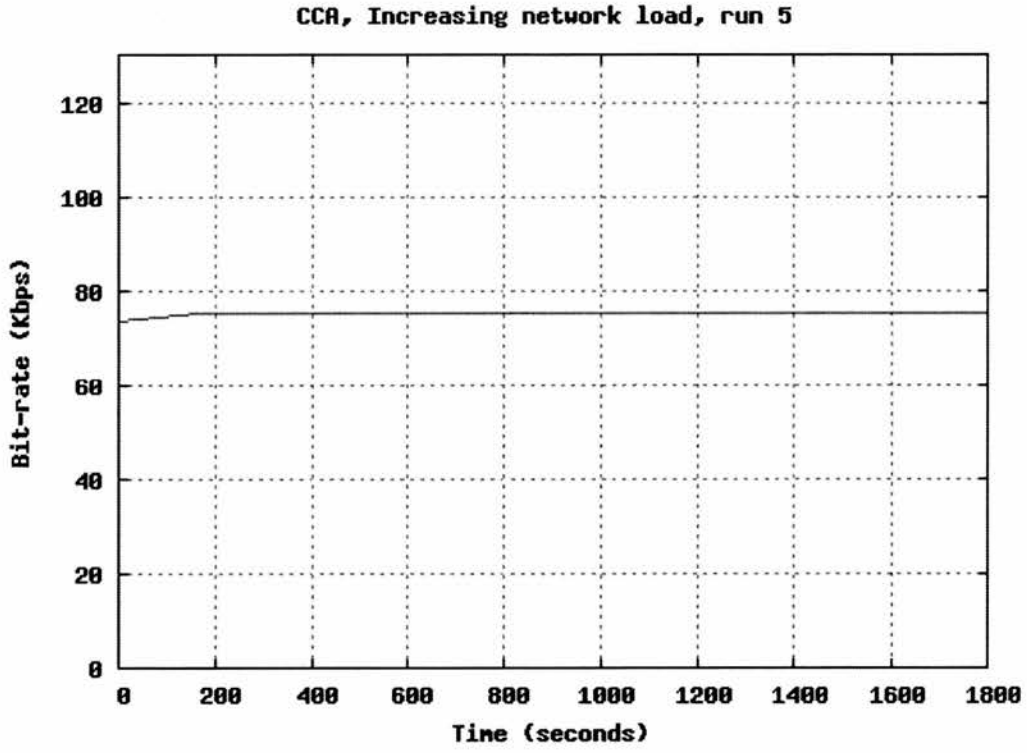

CCA, Increasing network load, run 0
Bit-rate (Kbps)
0
20
40
60
80
100
120
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)

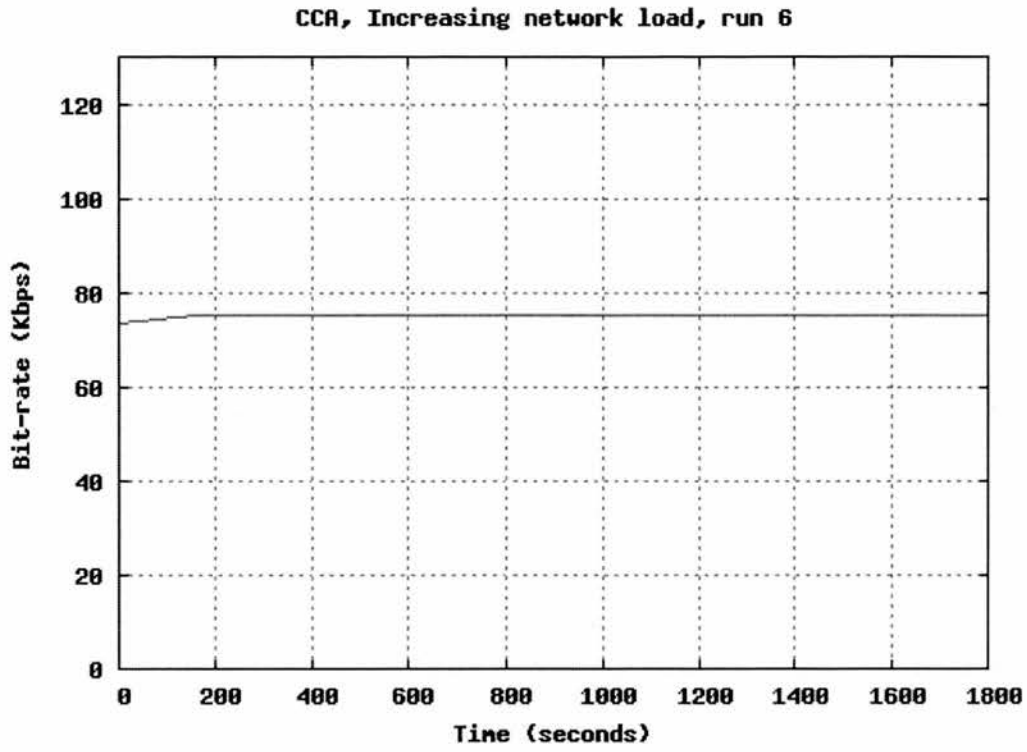

CCA, Increasing network load, run 1
Bit-rate (Kbps)
0
20
40
60
80
100
120
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)

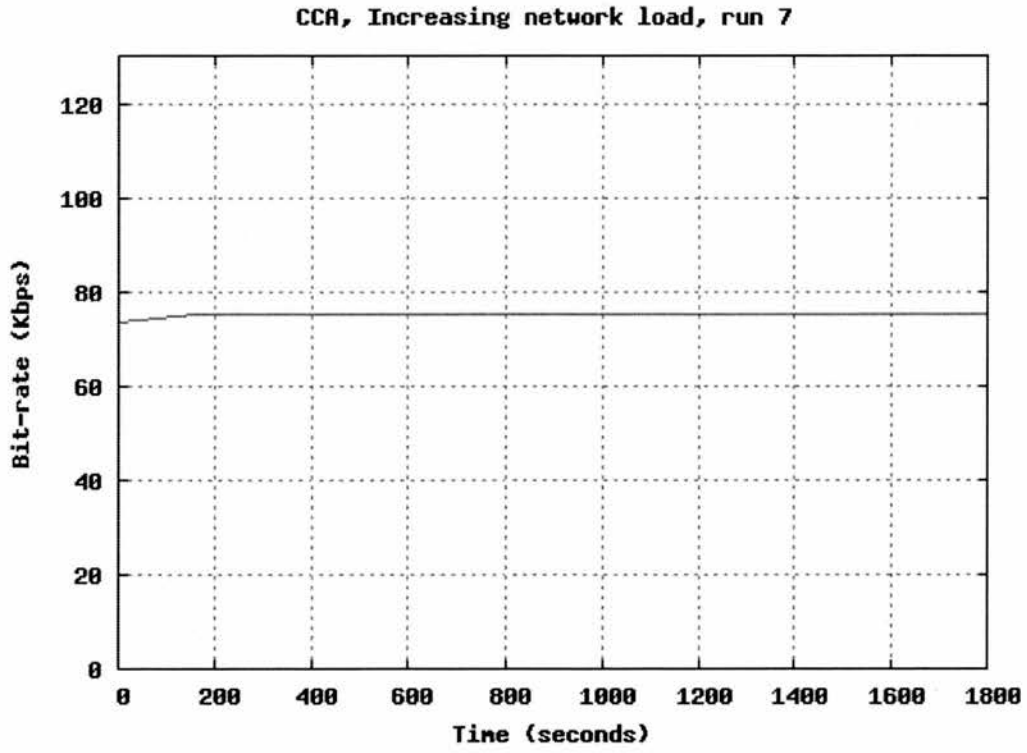

CCA, Increasing network load, run 2
Bit-rate (Kbps)
0
20
40
60
80
100
120
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)

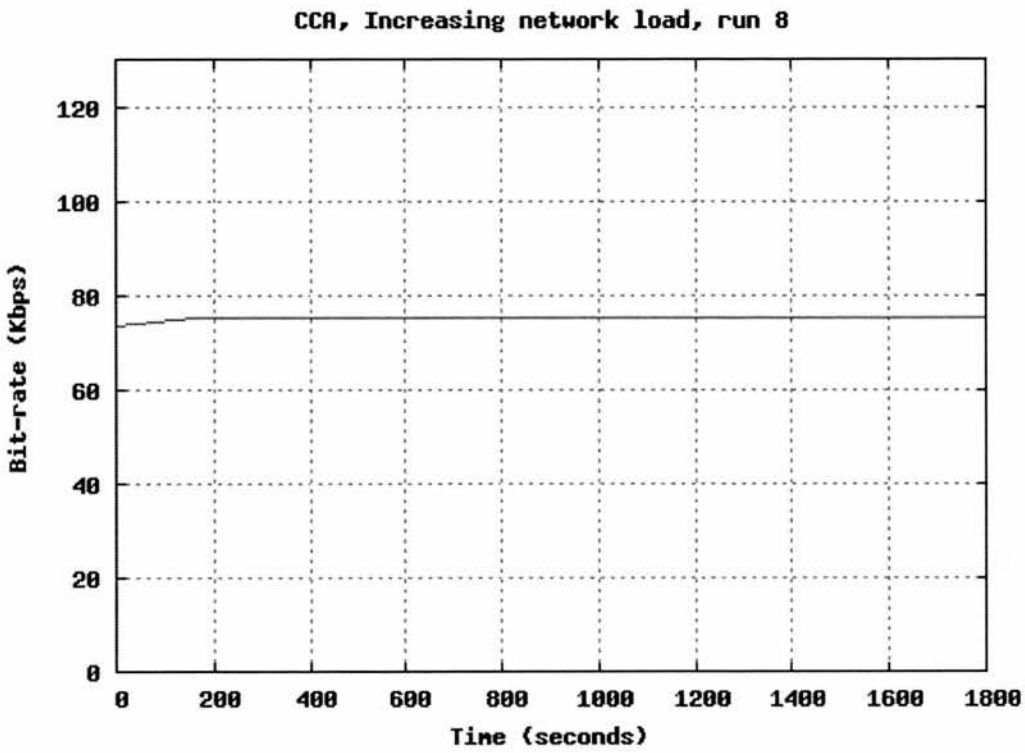

CCA, Increasing network load, run 3
Bit-rate (Kbps)
0
20
40
60
80
100
120
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)


CCA, Increasing network load, run 4
Bit-rate (Kbps)
0
20
40
60
80
100
120
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)


CCA, Increasing network load, run 5
Bit-rate (Kbps)
0
20
40
60
80
100
120
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)


CCA, Increasing network load, run 6
Bit-rate (Kbps)
0
20
40
60
80
100
120
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)


CCA, Increasing network load, run 7
Bit-rate (Kbps)
0
20
40
60
80
100
120
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)



D.3. Decreasing network load

D.3.1. CCA configuration script

```
use EmuNetwork;
use EmuNode;

my $network = new EnuNetwork (2.5, 1800);

# Set the router of the system
$network->router ("nog31");

my $node1 = new EmuNode ('host' => 'nog32',
                         'type' => '100',
                         'talkative' => '16');
my $node2 = new EmuNode ('host' => 'nog33',
                         'type' => '10',
                         'talkative' => '16');
my $node3 = new EmuNode ('host' => 'nog34',
                         'type' => 'isdn',
                         'talkative' => '16');
my $node4 = new EmuNode ('host' => 'nog35',
                         'type' => 'adsl',
                         'talkative' => '16');
```

```
my $node5 = new EmuNode ('host' => 'nog36',
                         'type' => 'isdn',
                         'talkative' => '16');
# convener
my $node6 = new EmuNode ('host' => 'nog37',
                         'type' => '100',
                         'talkative' => '20');

# Add the nodes to the network
$network->add ($node1);
$network->add ($node2);
$network->add ($node3);
$network->add ($node4);
$network->add ($node5);
$network->add ($node6);

# Create the network
$network->run ();
```

D.3.2. RUDE configuration script

```
START NOW
0000 0030 ON 3002 224.1.2.3:10001 CONSTANT 31 125
60000 0030 MODIFY CONSTANT 30 125
119000 0030 MODIFY CONSTANT 29 125
178000 0030 MODIFY CONSTANT 28 125
237000 0030 MODIFY CONSTANT 27 125
296000 0030 MODIFY CONSTANT 26 125
355000 0030 MODIFY CONSTANT 25 125
414000 0030 MODIFY CONSTANT 24 125
473000 0030 MODIFY CONSTANT 23 125
532000 0030 MODIFY CONSTANT 22 125
591000 0030 MODIFY CONSTANT 21 125
650000 0030 MODIFY CONSTANT 20 125
709000 0030 MODIFY CONSTANT 19 125
768000 0030 MODIFY CONSTANT 18 125
827000 0030 MODIFY CONSTANT 17 125
886000 0030 MODIFY CONSTANT 16 125
945000 0030 MODIFY CONSTANT 15 125
1004000 0030 MODIFY CONSTANT 14 125
1063000 0030 MODIFY CONSTANT 13 125
1122000 0030 MODIFY CONSTANT 12 125
1181000 0030 MODIFY CONSTANT 11 125
1240000 0030 MODIFY CONSTANT 10 125
1299000 0030 MODIFY CONSTANT 9 125
1358000 0030 MODIFY CONSTANT 8 125
1417000 0030 MODIFY CONSTANT 7 125
```

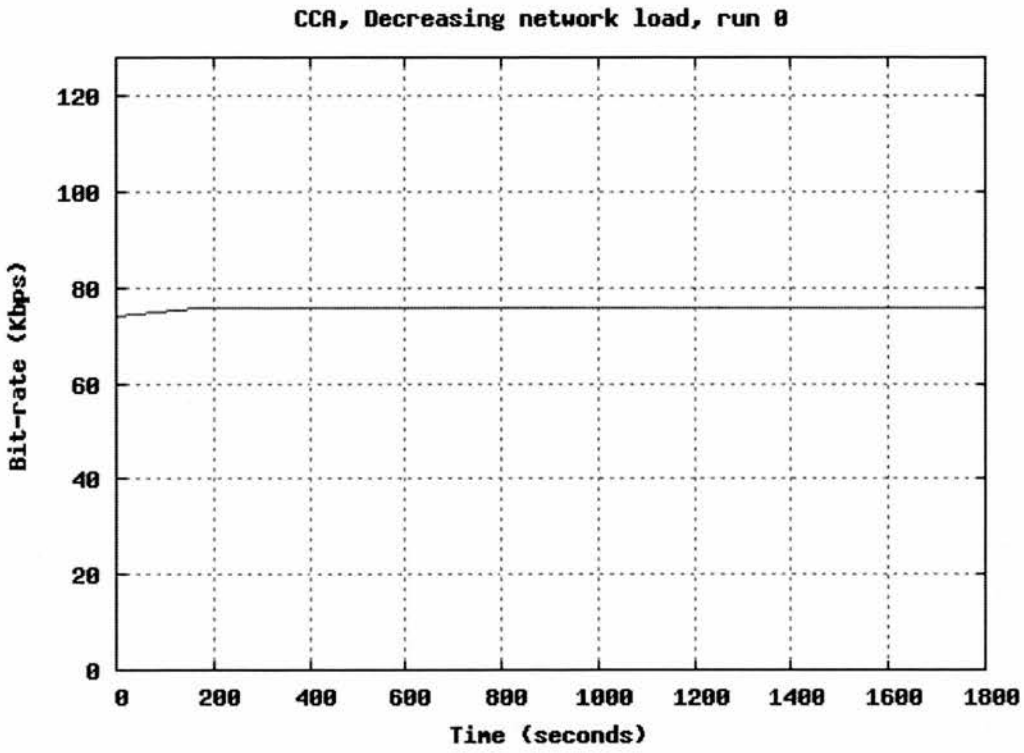
1476000 0030 MODIFY CONSTANT 6 125
1535000 0030 MODIFY CONSTANT 5 125
1594000 0030 MODIFY CONSTANT 4 125
1653000 0030 MODIFY CONSTANT 3 125
1712000 0030 MODIFY CONSTANT 2 125
1771000 0030 MODIFY CONSTANT 1 125
1810000 0030 OFF

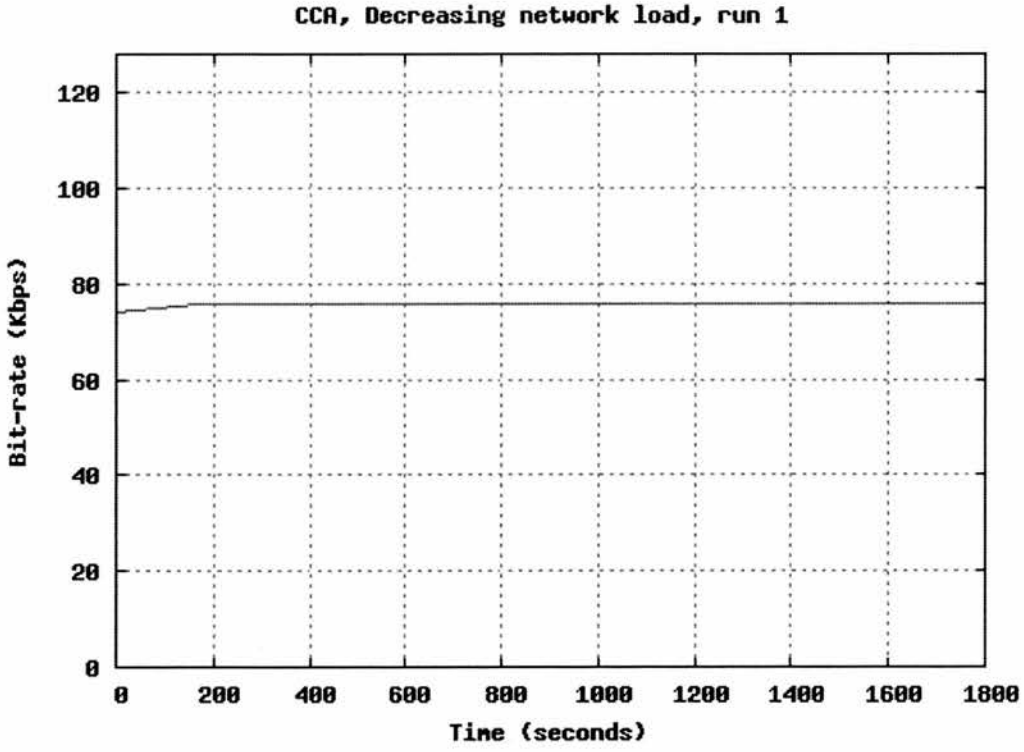
D.3.3. CCA policy script

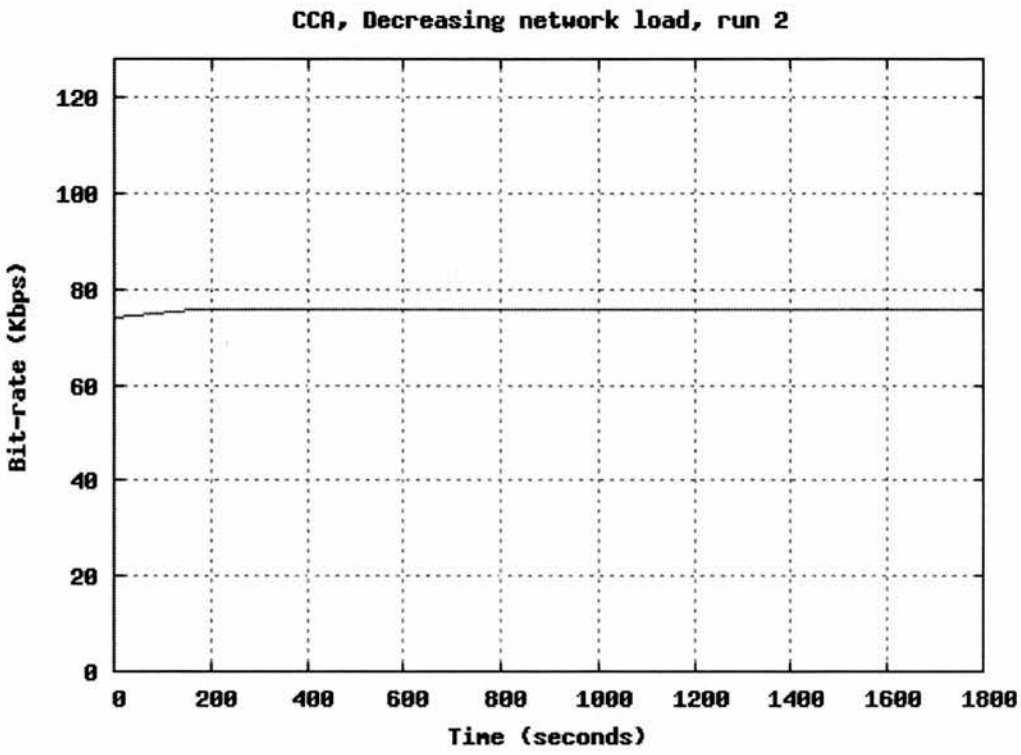
```
var avinfo  = new AVInfo ();
var avgbandwidth = new AVGTracker ();
var avgdelay = new AVGTracker ();
var avgdropped = new AVGTracker ();

function trafficReport (startDate, endDate, srcIP, destIP, bandwidth, variance,delay,
dropped, total, jitter) {
 avgdelay.add (srcIP, destIP, delay);
 avgdropped.add (srcIP, destIP, dropped);
 avgbandwidth.add (srcIP, destIP, bandwidth);

 var eqBandwidth = delayTobps (avgdelay.avg (srcIP, destIP), 1500, avgdropped.avg (srcIP,
destIP));

 var totalBW = avinfo.videoBitrate + avinfo.audioBitrate;

 if (eqBandwidth < totalBW) { // problem
  avinfo.videoBitrate = eqBandwidth - (avinfo.audioBitrate);
  send (avinfo);
  }
}

function AVInfo initialise (setting, measurements) {
 var bandwidth = measurements.bestguess.bandwidth;
 var delay = measurements.bestguess.delay;
 var jitter = measurements.bestguess.jitter;
 var loss = measurements.bestguess.packetloss;

 var noParticipants = settings.groupSize;
 var noActiveParticipants = settings.noTutors;

 var eqBandwidth = delayTobps (delay, 1500, loss);

 var bw = 0;
 if (eqBandwidth < bandwidth) {
  bw = eqBandwidth;
  } else {
  bw = bandwidth;
  }

 if (eqBandwidth > (bandwidth + 128000)) { // jump up slowly
  bw = bandwidth + 128000;
  }
```

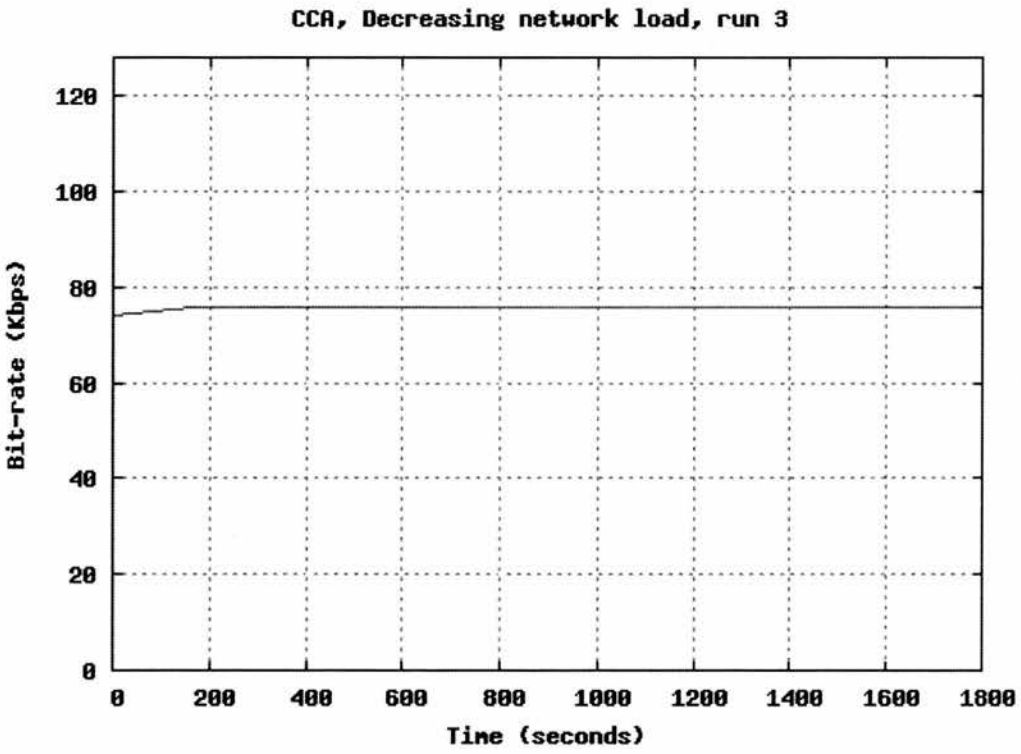
```
avinfo = configureAVInfo (bw, delay, jitter, loss);
return (avinfo);
}
```

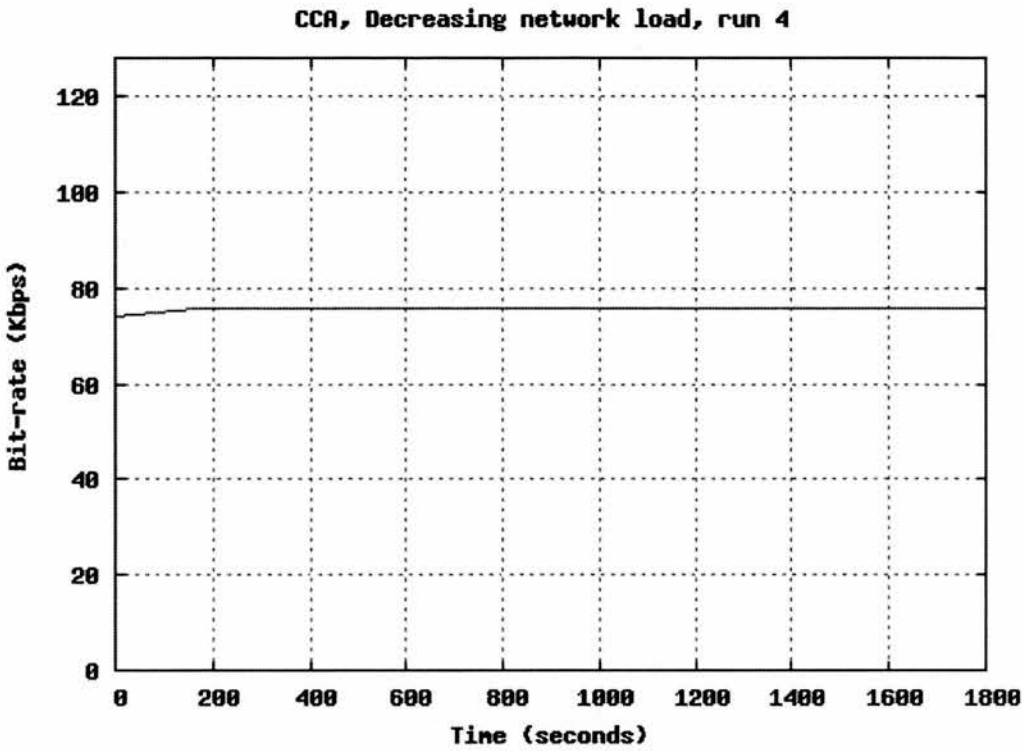
D.3.4. Results

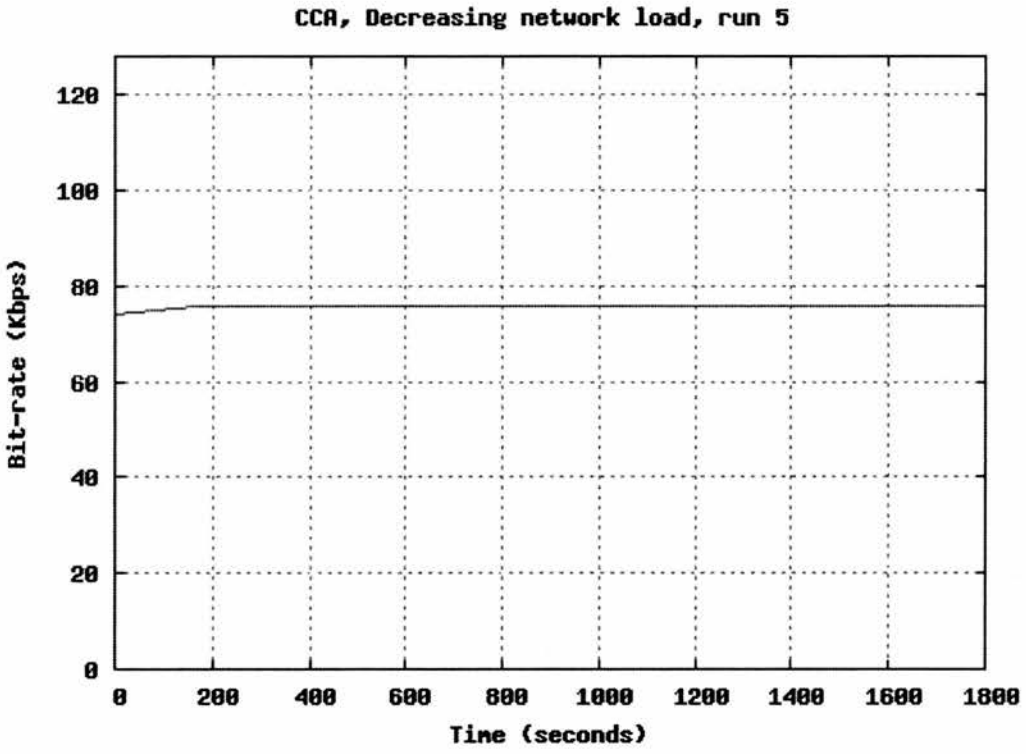

CCA, Decreasing network load, run 0
Bit-rate (Kbps)
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)

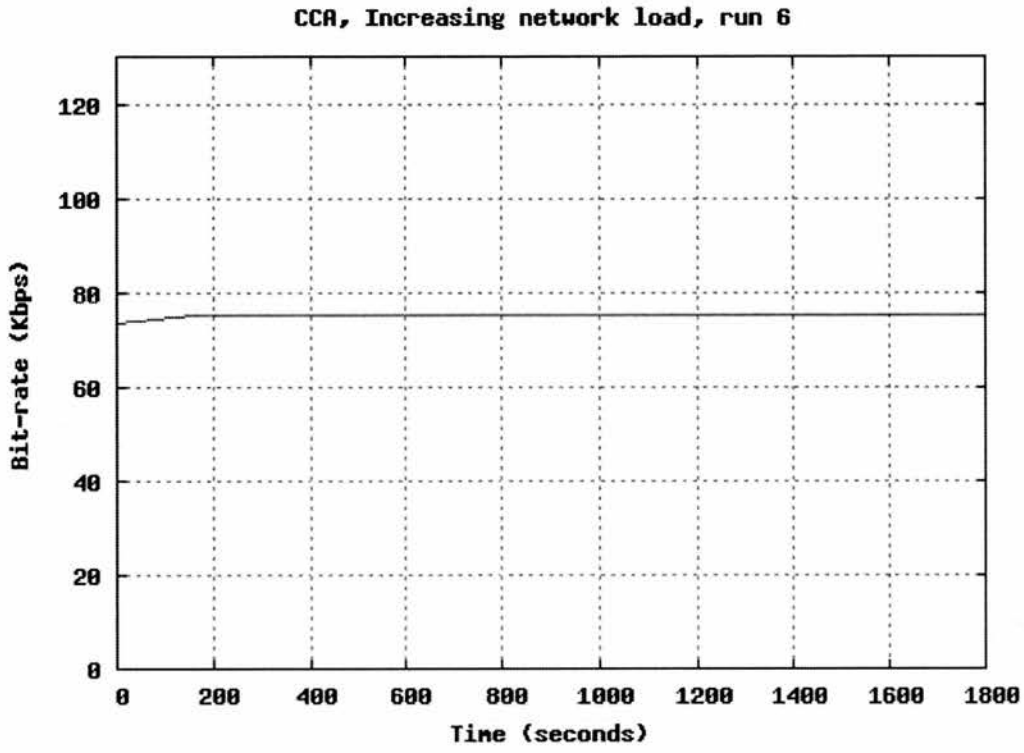

CCA, Decreasing network load, run 1
Bit-rate (Kbps)
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)

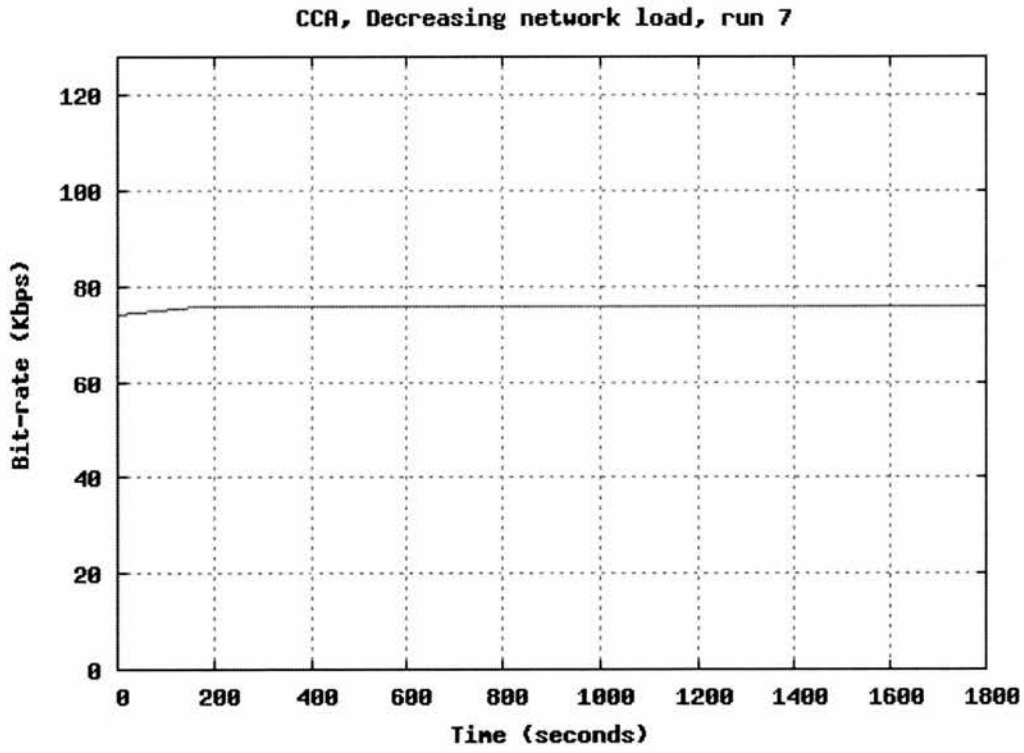

CCA, Decreasing network load, run 2
Bit-rate (Kbps)
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)

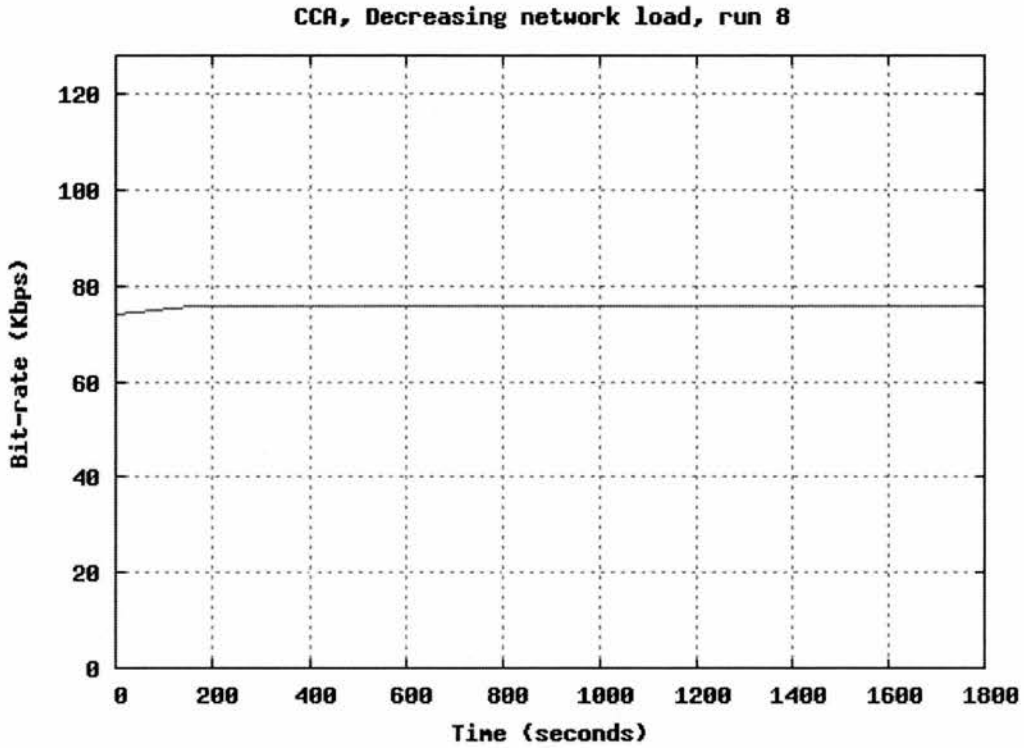

CCA, Decreasing network load, run 3
Bit-rate (Kbps)
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)


CCA, Decreasing network load, run 4
Bit-rate (Kbps)
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)


CCA, Decreasing network load, run 5
Bit-rate (Kbps)
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)


CCA, Increasing network load, run 6
Bit-rate (Kbps)
0
20
40
60
80
100
120
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)


CCA, Decreasing network load, run 7
Bit-rate (Kbps)
0
20
40
60
80
100
120
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)



D.4. Upgraded network

D.4.1. SDNE configuration script

The following configuration scripts were used to create the network and participants. The first script was used for the first video conference after that the second script was used. The second script upgrades the bottleneck connection from ISDN (symmetric 128 Kbps) to ADSL (symmetric 512 Kbps).

D.4.1.1. SDNE configuration 1

```
use EmuNetwork;
use EmuNode;

my $network = new EnuNetwork (2.5, 1800);

# Set the router of the system
$network->router ("nog31");

my $node1 = new EmuNode ('host' => 'nog32',
```

```
                    'type' => '100',
                    'talkative' => '16');
my $node2 = new EmuNode ('host' => 'nog33',
                    'type' => '10',
                    'talkative' => '16');
my $node3 = new EmuNode ('host' => 'nog34',
                    'type' => 'adsl',
                    'talkative' => '16');
my $node4 = new EmuNode ('host' => 'nog35',
                    'type' => 'adsl',
                    'talkative' => '16');
my $node5 = new EmuNode ('host' => 'nog36',
                    'type' => 'isdn',
                    'talkative' => '16');
# convener
my $node6 = new EmuNode ('host' => 'nog37',
                    'type' => '100',
                    'talkative' => '20');

# Add the nodes to the network
$network->add ($node1);
$network->add ($node2);
$network->add ($node3);
$network->add ($node4);
$network->add ($node5);
$network->add ($node6);

# Create the network
$network->run ();
```

D.4.1.2. SDNE configuration subsequent

```
use EmuNetwork;
use EmuNode;

my $network = new EnuNetwork (2.5, 1800);

# Set the router of the system
$network->router ("nog31");

my $node1 = new EmuNode ('host' => 'nog32',
                    'type' => '100',
                    'talkative' => '16');
my $node2 = new EmuNode ('host' => 'nog33',
                    'type' => '10',
```

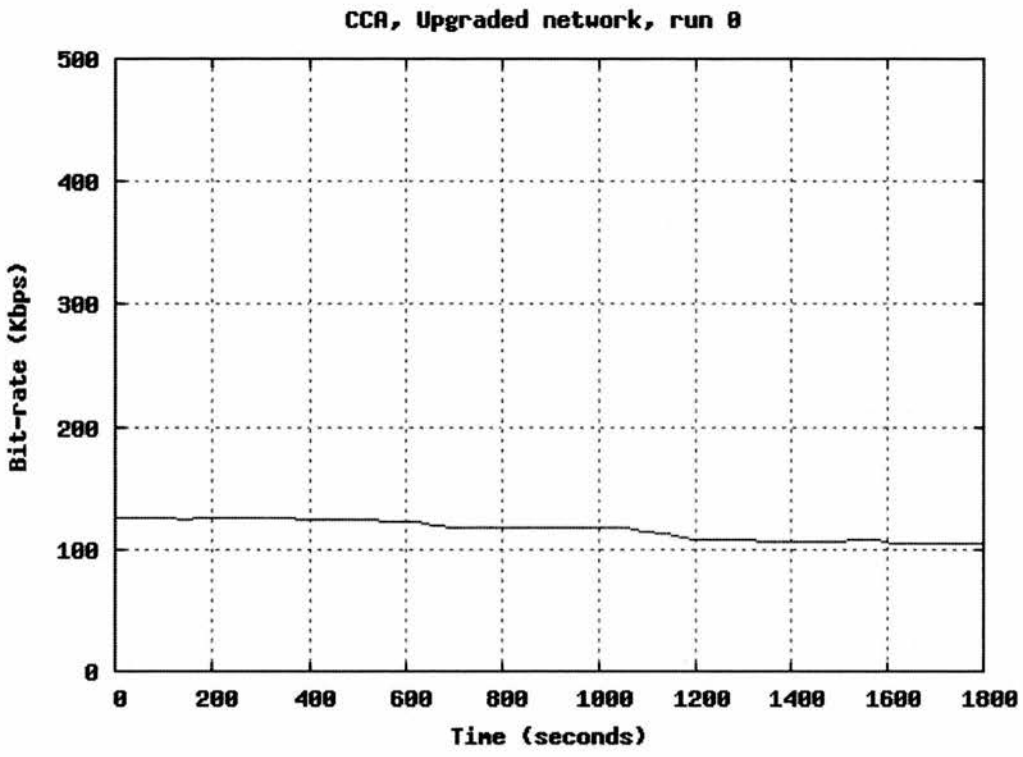
```
                    'talkative' => '16');
my $node3 = new EmuNode ('host' => 'nog34',
                    'type' => 'isdn',
                    'talkative' => '16');
my $node4 = new EmuNode ('host' => 'nog35',
                    'type' => 'adsl',
                    'talkative' => '16');
my $node5 = new EmuNode ('host' => 'nog36',
                    'type' => 'adsl',
                    'talkative' => '16');
# convener
my $node6 = new EmuNode ('host' => 'nog37',
                    'type' => '100',
                    'talkative' => '20');

# Add the nodes to the network
$network->add ($node1);
$network->add ($node2);
$network->add ($node3);
$network->add ($node4);
$network->add ($node5);
$network->add ($node6);

# Create the network
$network->run ();
```

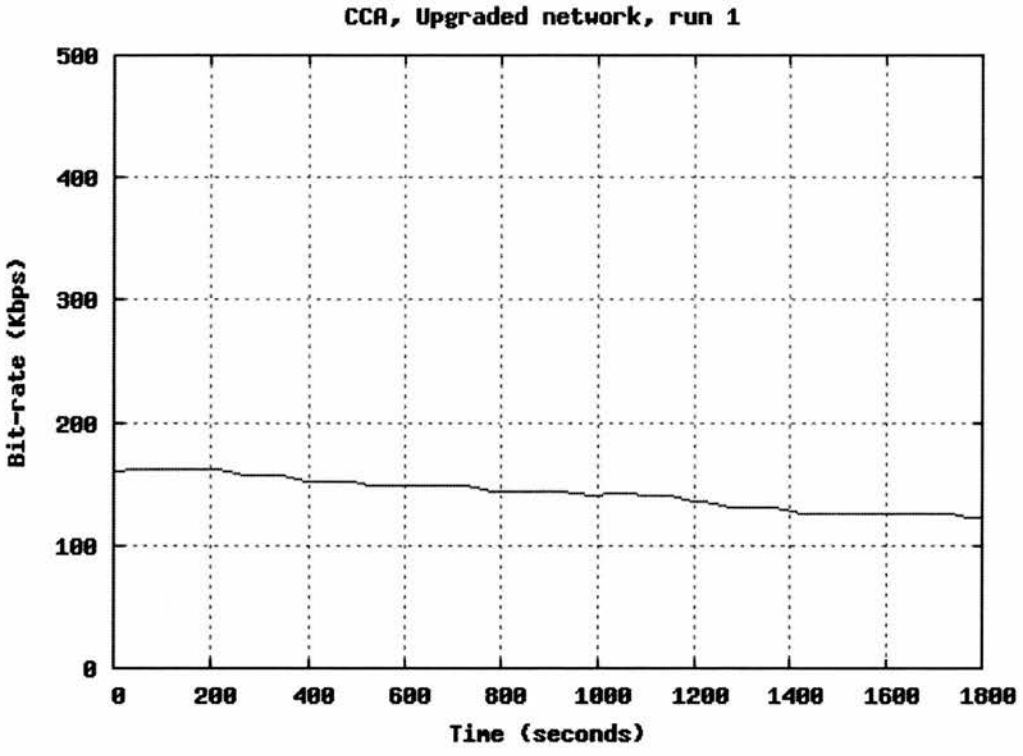
D.4.2. CCA policy script

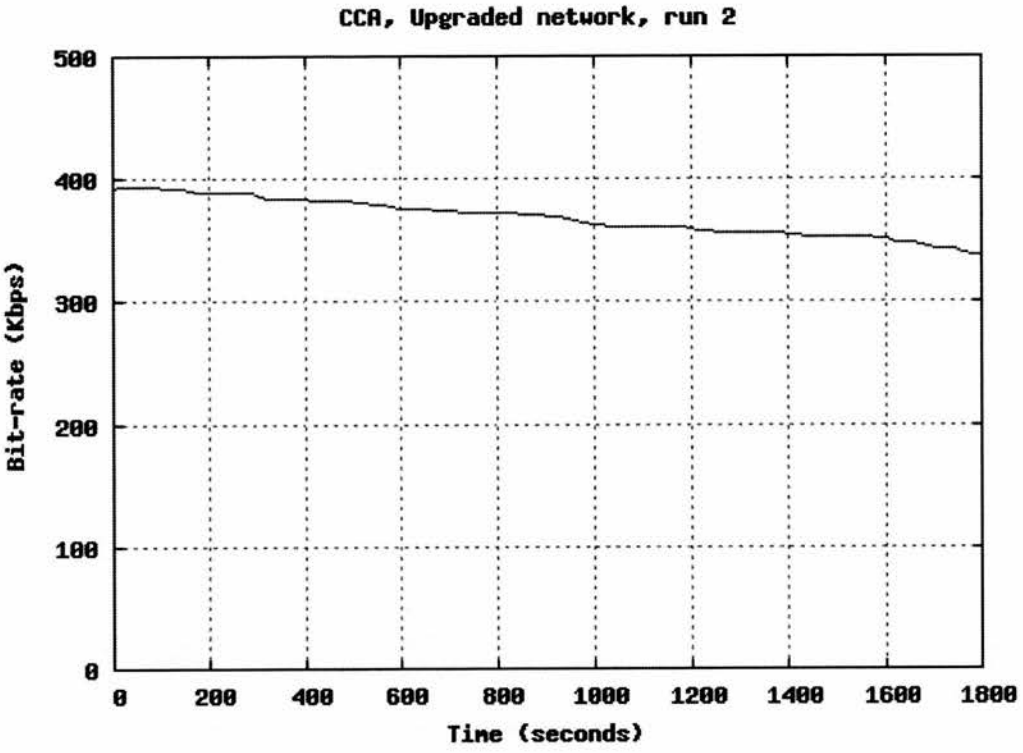
```
var avinfo  = new AVInfo ();
var avgbandwidth = new AVGTracker ();
var avgdelay = new AVGTracker ();
var avgdropped = new AVGTracker ();

function trafficReport (startDate, endDate, srcIP, destIP, bandwidth, variance,delay,
dropped, total, jitter) {
 avgdelay.add (srcIP, destIP, delay),
 avgdropped.add (srcIP, destIP, dropped);
 avgbandwidth.add (srcIP, destIP, bandwidth);

 var eqBandwidth = delayTobps (avgdelay.avg (srcIP, destIP), 1500, avgdropped.avg (srcIP,
destIP));

 var totalBW = avinfo.videoBitrate + avinfo.audioBitrate;

 if (eqBandwidth < totalBW) { // problem
  avinfo.videoBitrate = eqBandwidth - (avinfo.audioBitrate);
  send (avinfo);
  }
}

function AVInfo initialise (setting, measurements) {
 var bandwidth = measurements.bestguess.bandwidth;
 var delay = measurements.bestguess.delay;
 var jitter = measurements.bestguess.jitter;
 var loss = measurements.bestguess.packetloss;

 var noParticipants = settings.groupSize;
 var noActiveParticipants = settings.noTutors;

 var eqBandwidth = delayTobps (delay, 1500, loss);

 var bw = 0;
 if (eqBandwidth < bandwidth) {
  bw = eqBandwidth;
  } else {
  bw = bandwidth;
  }

 if (eqBandwidth > (bandwidth + 128000)) { // jump up slowly
  bw = bandwidth + 128000;
  }
```

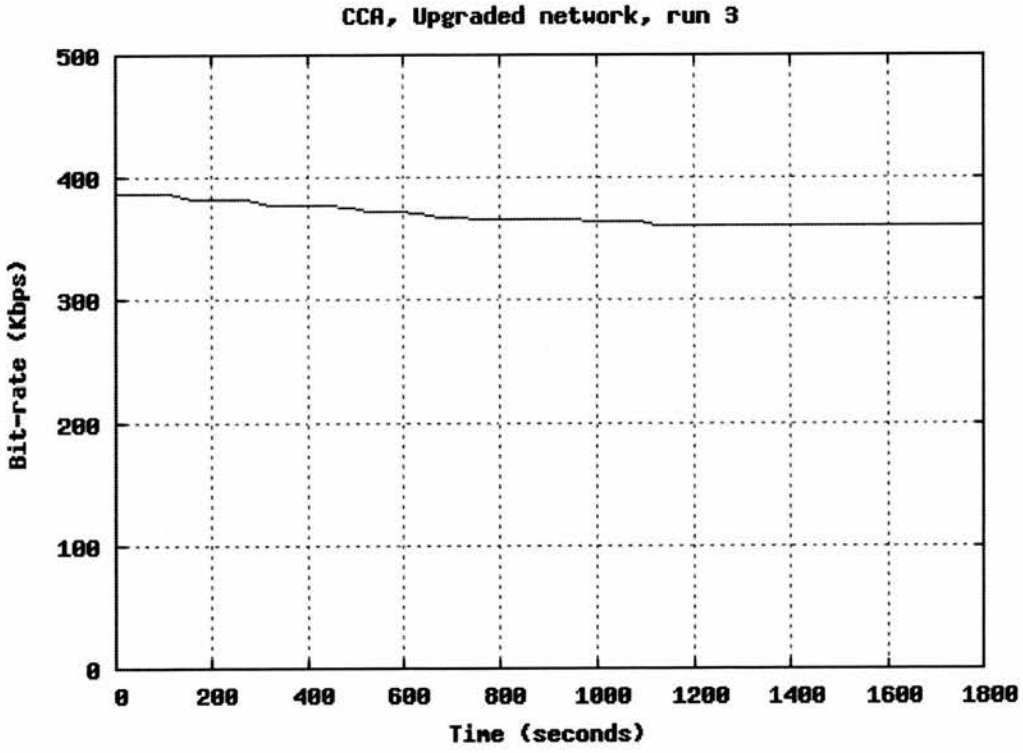
```
avinfo = configureAVInfo (bw, delay, jitter, loss);
return (avinfo);
}
```

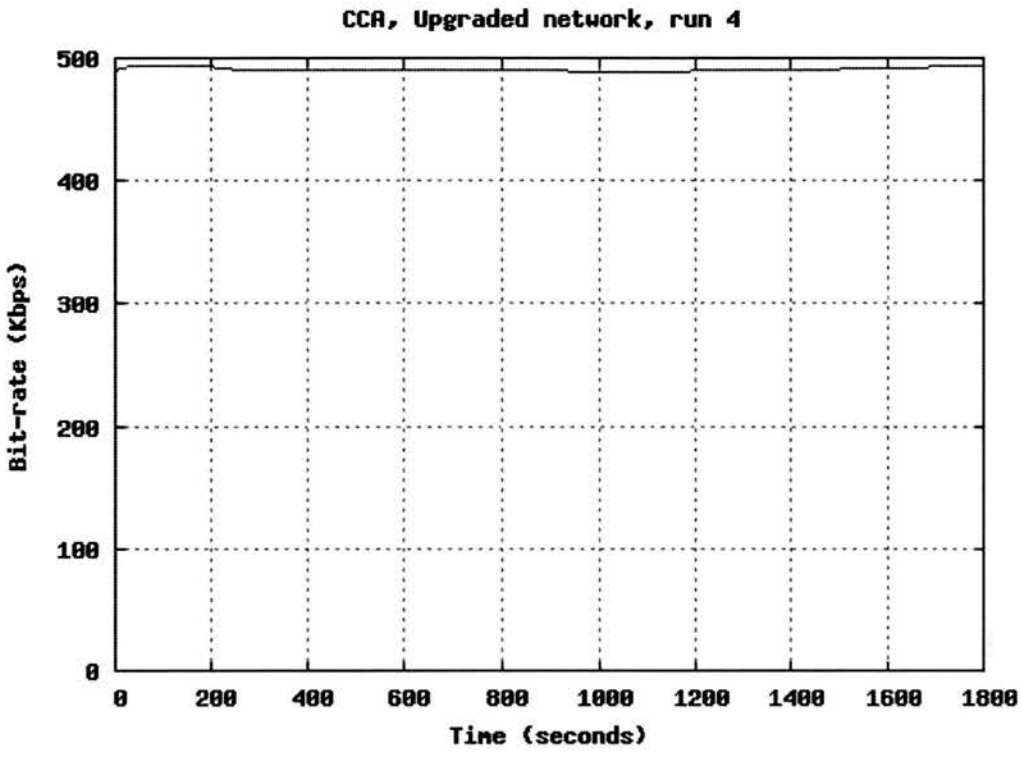
D.4.3. Results

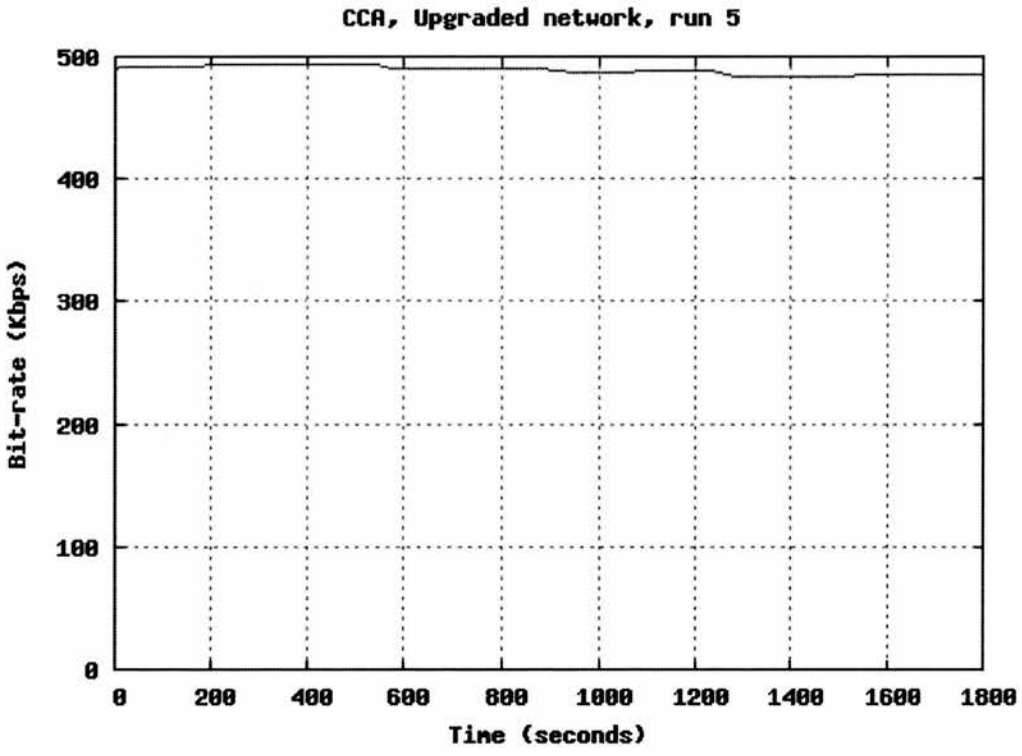

CCA, Upgraded network, run 0
Bit-rate (Kbps)
Time (seconds)
0
100
200
300
400
500
0
200
400
600
800
1000
1200
1400
1600
1800

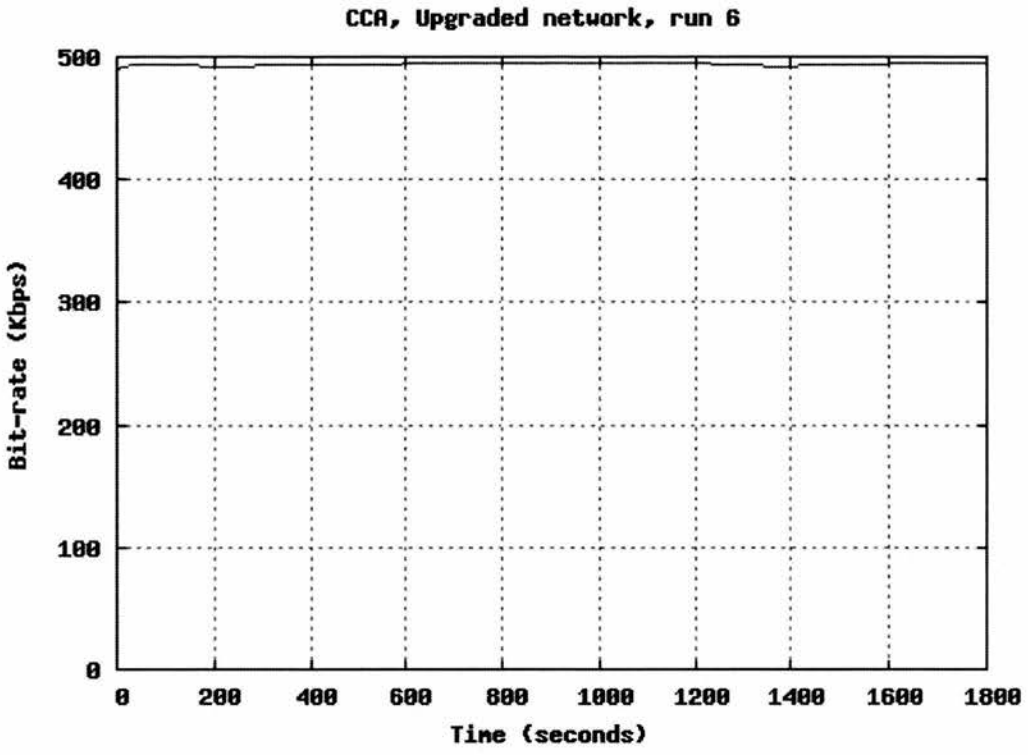

CCA, Upgraded network, run 1
Bit-rate (Kbps)
Time (seconds)
0
100
200
300
400
500
0
200
400
600
800
1000
1200
1400
1600
1800

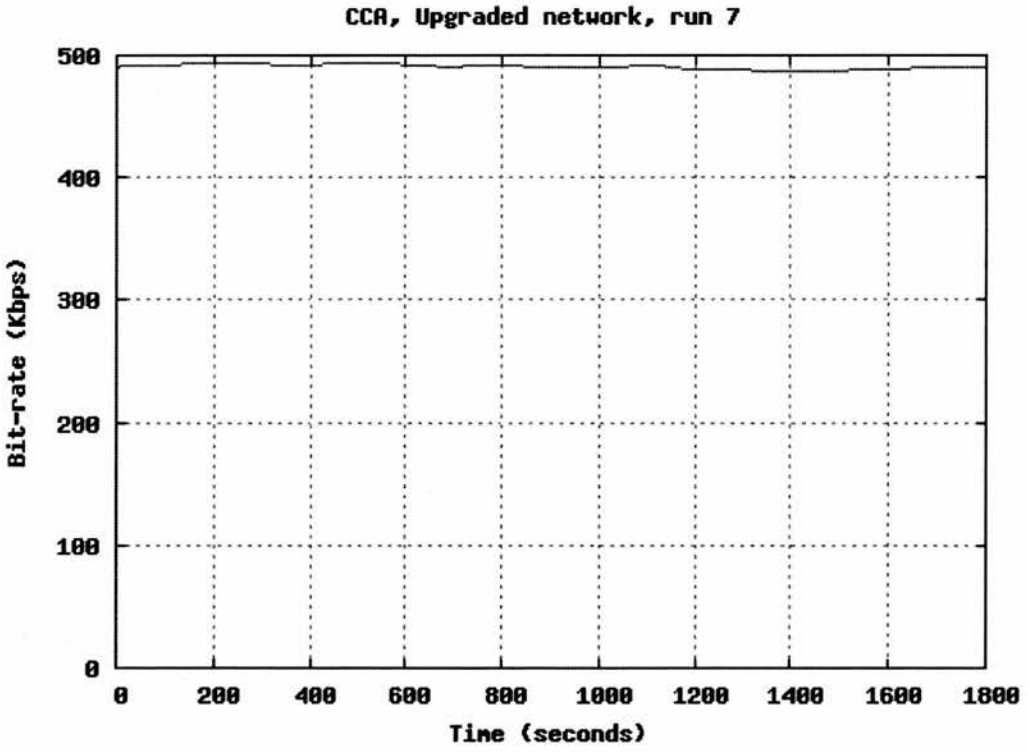

CCA, Upgraded network, run 2
Bit-rate (Kbps)
500
400
300
200
100
0
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)

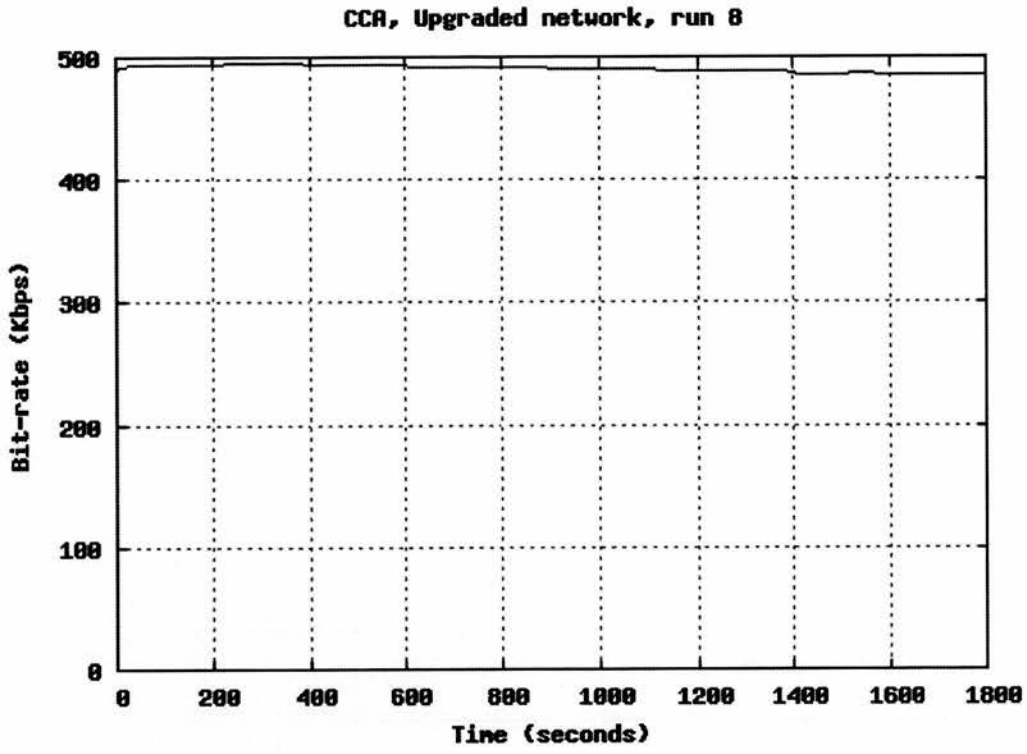

CCA, Upgraded network, run 3
Bit-rate (Kbps)
500
400
300
200
100
0
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)


CCA, Upgraded network, run 4
Bit-rate (Kbps)
0
100
200
300
400
500
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)


CCA, Upgraded network, run 5
Bit-rate (Kbps)
0
100
200
300
400
500
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)


CCA, Upgraded network, run 6
Bit-rate (Kbps)
0
100
200
300
400
500
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)


CCA, Upgraded network, run 7
Bit-rate (Kbps)
0
100
200
300
400
500
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)


CCA, Upgraded network, run 8
500
400
300
200
100
0
Bit-rate (Kbps)
0
200
400
600
800
1000
1200
1400
1600
1800
Time (seconds)