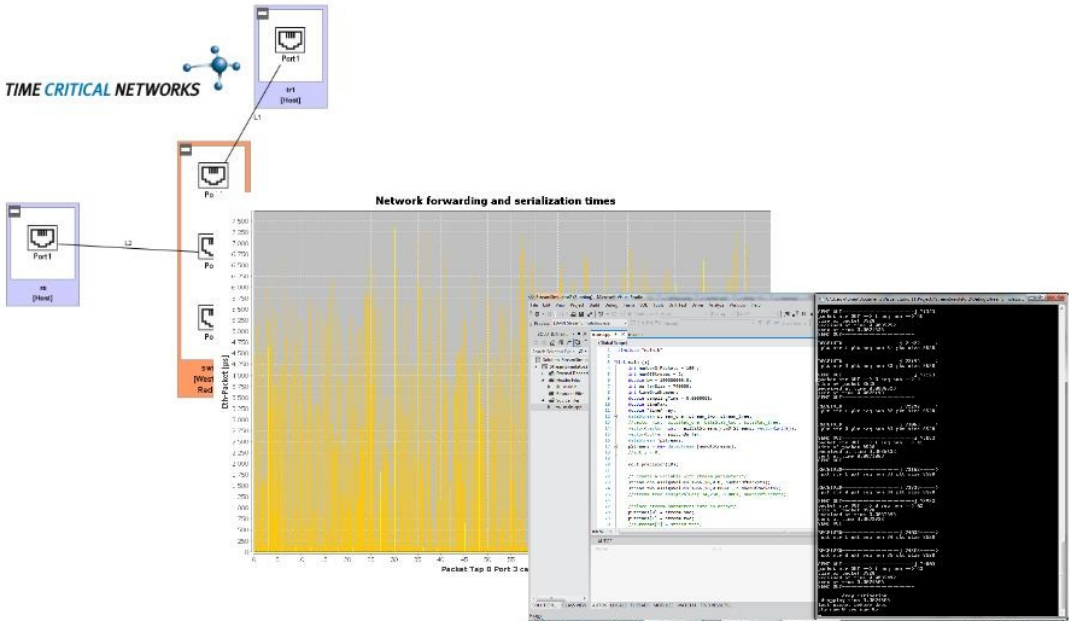
CHALMERS



Towards Automated Network Configuration
Simulations In A Switched Ethernet Network

Master's thesis in Communication engineering

Kateryna Mariushkina

Joel Pettersson

Department of Signal and System
Chalmers University of Technology
Göteborg, Sweden, December 2012
Report no EX011/2013

Towards Automated Network Configuration
Kateryna Mariushkina
Joel Pettersson

Master's Thesis

Report no EX011/2013
Chalmers University of Technology
Department of Signal and System
SE-41296 Göteborg
Sweden Tel. +46-(0)31 772 1000

Department of Signal and System
Göteborg, Sweden 2012

Abstract

This project attempts to create a mechanism for automatic configuration of network parameters depending on the current network load in order to avoid packet-loss. The system should be accommodated for transmission of real-time video streams through switched Ethernet networks.

Investigation of the problem incorporated a real-time video test with a netANALYSER card to capture in/out going network traffic, determining the packet loss point for each of the network configurations. The same test was replicated by using traffic generating cards and the software tool TCN TimeAnalyzer. The tests showed the possibility of predicting network behaviour without involving real streams but rather managing the network before the real transmission has begun.

A packet capture software tool and a switch simulator was developed and integrated into the packet loss prediction mechanism. The packet capturing software produced the statistics of the network traffic for later offline simulations. The switch simulator mimicked the real network switching behaviour. It estimated the time and amount of traffic when the buffer memory was full.

A solution is proposed for independent offline control of the network behaviour. It can later be applied to an online version and incorporated into numerous real-time systems for in-vehicle or video communications.

Contents

List of Figures

Acronyms

Bps - Bits Per Second

CAN - Controller Area Network

CC - CRSC Count

CP-EP - Capture Packets - Extract Parameters

CRC - Cyclic Redundancy Check

CSRC - Contributing Source IDs Enumerate Contributing

DCT - Discrete cosine Transform

HTML - Hypertext Markup Language

IAR - Ingenjörsfirman Anders Rundgren

IGMP - Internet Group Management Protocol

IP - Internet Protocol

IPv4 - Internet Protocol version 4

Fps - Frames per second

GUI - Graphical User Interface

LAN - Local Area Network

M - Marker

MAC - Media Access Control

M-JPEG - Motion Joint Photographic Expert Group

MPEG - Moving Picture Experts Group

MTU - Maximum Transmission Unit

NPF - Netgroup Packet Filter

P - Padding

QoS - Quality of Service

RAM - Random Access Memory

RTCP - Real Time Control Protocol

RTP - Real Time Protocol

SA - TCN StreamAnalyzer

SIP - Session Initiation Protocol

SSRC - Synchronization Source Identifier

TA - TCN TimeAnalyzer

TAP - Traffic Access Points

TCN - Time Critical Networks

TCP - Transmission Control Protocol

ToS - Type of Service

TTL - Time to Live

UDP - User Datagram Protocol

WAN - Wide Area Network

X - Extension

1

Introduction

The vehicle industry develops rapidly using new technologies and setting new challenges for the existing ones. In-vehicle communication is facing big changes nowadays. It is caused by the increased demands for data transmission inside the vehicle. Ethernet as a mean of in-vehicle communication system may solve the problems that have appeared with the new demands.

The amount of in-vehicle traffic has increased dramatically. Nowadays along with new electronics in cars video cameras were accommodated to increase the safety of the vehicle movement. Video surveillance of the traffic or the environment around the vehicle has compound the heavy in-vehicle traffic load. Existing CAN (Controller Area Network) system is unable to provide large bandwidth for video streams [1]. It has 1 Mbit/s bandwidth capacity on the distances of 40m. With new technologies added to the vehicles, this bandwidth may not be adequate for in-vehicle networks [2]. MOST 150 (Media Oriented Systems Transport) has been developed to carry multimedia stream with a maximum bandwidth 150 Mbit/s [3]. In addition, with the expected increase in bandwidth usage, these systems may cause problems in packet delivery.

Moreover mechanical functions are now controlled and/or replaced by the electronics. For each application there exists specifically developed technology [4]. They function inside one vehicle but due to different protocol formats correspon-

dence between them is impossible. In addition every automobile has kilometers of wires to enable communication between sensors, engine and controlling systems. They add extra weight to the vehicle and introduce complexity of mounting and organizing the communication system.

Therefore new approach was needed to fight these issues. Ethernet can support the bandwidth up to 1 Gbit/s thus enabling transmission of data for various applications simultaneously without the risk of information loss. Ethernet is a very flexible system that allows the access and the exchange of the information on the different protocol levels. Since Ethernet can substitute existed isolated communication systems, the weight of the cables, space and the complexity will be decreased, providing lighter cars with fuel economy.

Having listed advantages of Ethernet it worth mentioning why this technology was not used earlier in time-dependent applications before. Time-dependent applications cannot tolerate time delays in network traffic delivery. Ethernet is an unreliable service, and can introduce jitter and delays to the system [5, 6].

Due to the increasing demand for solutions to the problems of bandwidth and data delivery requirements, companies as Time Critical Networks AB (TCN) have evolved. TCN is currently developing tools for simulating and evaluating Ethernet networks. The tools are developed in order to predict the worst case forwarding delays of data traffic [6, 7].

Most research in networks focuses on minimizing the packet drops by diagnosing the system when packet drops occur. Then the network load is lower in order to decrease the possibility of dropping occurring again. This is due to the fact that most applications can handle packet drop in data streams [8, 9].

The first objective of the thesis is to investigate a simple network topology populated with video and audio streams and adjust them using the tools developed by TCN for network investigation. The second objective is to verify if the analysis engine can predict packet drop occurrence. The outcome of the observation should be used to find a near-optimal set of configurable network and/or applica-

tion parameters that avoid packet drops completely while optimizing application performance. A ”human-in-the-loop” approach will be used to simulate the tasks that would ultimately be performed by the automated system.

The project report is organized in the following manner. Chapter 2 provides an overview of the network communications. The description is oriented to provide knowledge about network switches, their functions, levels of the protocol suit, protocols and, most importantly, Ethernet technology. This chapter also explains jitter and delays in computer networks and their influence on Quality of Service (QoS). Since this project deals with multimedia communication, key video and audio codecs are presented.

In chapter 3, section 3.1 a summary of software that are used in the thesis is given. Section 3.2 presents testing of the network, environment and the measured results. Here three tests were carried out: real-time video transmission, TimeAnalyzer/StreamAnalyzer and tests with traffic generating cards produced by Ingenjörsfirman Anders Rundgren(IAR) company. Based on the test results it was possible to predict or give a feedback about the network throughput ability. Finally, section 3.3 presents the developed software for packet capturing and switch simulations. Both components are included in the loop of traffic monitoring process.

All elements from the previous section were tested in a real environment where a video conference was transmitted through an Ethernet switch network. Results are illustrated in chapter 4. The tests included all developed software and were based on the conclusion derived from the previous section.

Chapter 5, section 5.1 discuses the outcome and the result from the testing. Section 5.2 suggests improvements of the created system and its possible modifications.

2

Background

2.1 Computer communication

For an absolute understanding of this project basic knowledge of computer communication is needed. In this section, the most important concepts used throughout the project are presented.

2.1.1 Network and switches

A computer network is comprised of a number of hardware devices connected with each other by wired or wireless channels to enable information exchange. Devices, sometimes called computers, hosts or end systems, can receive and transmit information from and to the network.

There are many different network organizations and topologies. The lowest level of the network organization is the local network. It simply connects a number of computers with each other e.g, in the office or at home. They usually have a small geographical extent, high speed connection and in some cases limited access rights to network services. End devices are connected with each other by communication links and packet switches [5].

There are two possible network communication configurations, circuit switched

and packet switched networks. The difference between them lies in the resource allocation. In circuit switched networks resources are assigned in the beginning of the communication between end users until the end of the session. Thus, it provides a stable line and set bandwidth. On the contrary, in packet switched networks the information is sent out without any pre-reservation. This creates a possibility of delays and losses if the link has to be used by several users at the same time.

In packet switched networks the information that travels through the network can be very different, e.g. audio files, pictures, or documents sent by mail. In order to be transmitted through the network, the information is divided into smaller packets. These packets are sent through the communication links then. Packet switches take arriving packets and forward them to the end system. It usually depends on the switch itself which of the forwarding methods it uses. In the case of store-and-forward transmission, the packet that arrives is completely copied into the buffer of the switch until the last bit of the packet has arrived. Then it forwards the packets in a first-in-first-out fashion to the communication link. This type of switch transmission is used in this thesis.

Since switches just connect, redirect or forward arrived packets, they are invisible for the end users in the network. The speed in different links connected to the switch may also differ from each other. Therefore switches have output queue buffers to store packets before the link is ready to accept and transmit information. Switches also provide filtering and forwarding functions. Switches can filter out packets that were damaged or incompletely received and switches can also direct packets to the needed end user interfaces.

Switches work at the link layer using the Media Access Control (MAC) addresses of the connected devices. The MAC address is a 6-byte address assigned to the computer network adapter when manufactured. Based on MAC addresses, the forwarding table of the switch is automatically constructed with indexes corresponding to each MAC address.

Network participants can connected not only to the switch but to the hub and

router as well. Data that is sent through the hub is broadcasted to the network to all the participants. Hubs do not filter or direct packets. Routers usually connect two different networks: world and local area network,for example. Routers forward packets along to the best route and communicate with each other to obtain the information about routes.

The Transmission Control Protocol (TCP) and Internet Protocol (IP) are used in the Internet network to have the control over the system of circling information.

2.1.2 Switching models

As explained in section 2.1.1 the data that is to be sent through the network is divided into packets with additional information attached to them in order to route them to the required node. There are two ways for the network to treat data packets: datagram and virtual circuit.

The datagram packet switched network treats each packet, called datagram, as a separate independent piece of information without any relation to the prior sent data. Therefore, each packet is sent through the path that is the best for it, according to the information received from the node's neighbours. Packets carrying different parts of one message may traverse different routes in the network. It may cause arrival to the end user in reordered sequence. Then it is a task of the receiving side to recover the data [10].

With the virtual circuit method, the path for all packets is established before sending them. It is similar to the circuit switched network. The decision about the travelling route is already made for all the packets. But it is possible to utilise pre-established line by other users if those packets are in the node's buffer at this moment. Virtual circuit method increases the speed of packets when the communication for a long period of time with constant data exchange is required. This method also enables error correction and packet retransmission.

Datagram switched network is flexible and fast in case of a single-time data transmission. It better copes with network congestion since the information is chopped

into packets following different routes from node to node [10]. Precisely that type of data handling is used in the project.

2.1.3 Protocols and layers

Data exchange between established links can involve a complex acquiring process whether or not the receiver is ready to accept and process the data. The whole interaction process is thus divided into smaller tasks (layers). The layers are vertically arranged in a stack (IP stack), where they perform independently from each other. Each upper layer operate on the data accepted from the lower layer and passes data further to the top layers for processing. All computers that communicate within the network must have the same functions of the IP stack. This is checked in the layer that sends out blocks of data to the peer layer of the other computer. The control over such process is carried out by protocols.

The communication architecture has been standardized, as a result TCP/IP protocol stack appeared. The protocol stack contains 5 layers of data processing in the vertical stack: application, transport, internet, link, and physical layers. At each level, data can be cut into parts, depending on the requirements, and put back together at the peer level before transmission. Every layer can use one or more protocol standards[5].

2.1.4 Ethernet

Initially computers were connected through a transceiver sharing the same medium or channel. Information transmitted from one machine was broadcasted to all the participants in the channel. The network card did not discard information even if the destination address was not corresponding to receiving computer's address. Bandwidth was shared. Ethernet was simpler than Token Ring or Token Bus (technologies competing with the Local Area Network (LAN) standard at the same time) but was unstable for large networks introducing data loss and poor QoS.

Ethernet is the most used technology in LAN nowadays. Ethernet is normally used in a star topology with a switch connecting the nodes, see Figure 2.1. Older

Ethernet systems used to have hubs instead of switches in this setup. Hubs had the disadvantage of creating collisions in the network, if more than one node where sending data at the same time. Changing the hubs to switches has eliminated most of the collision problems in the network. Nowadays modern switches are full duplex thereby eliminating collisions [5].

There are several types of standards for Ethernet such as IEEE 802.3, IEEE 802.2 Logical Link Control, but in this project only Ethernet Version 2 standard will be used. The frame structure of this type is divided in six fields see Figure 2.2 [11].

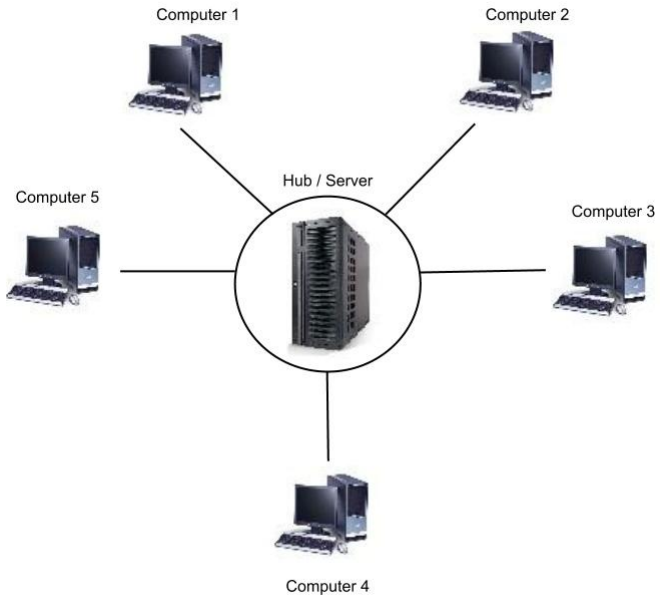


Figure 2.1: Schematic picture of a star shape topology network

8 bytes	6 bytes	6 bytes	2 bytes	46-1500 bytes	4 bytes	12 bytes
Preamble	Dest address	Source address	Type	Data	CRC	Interframe gap

Figure 2.2: The element of an Ethernet frame

Preamble is sent in the beginning of each Ethernet frame. It is an 8 byte field, with the first seven bytes filled with 10101010. It is required to synchronize the transmission. The last byte is 10101011, where last two bites 11 in the end are required for the receiver to signal the upcoming data.

Destination address contains the MAC address of the receiver.

Source address contains the MAC address of the sender.

Type field describes what kind of protocol is used in the transmission. Ethernet can handle several different protocols.

Data field contains IP payload information. The minimum size of this field is 46 bytes, and the maximum size or Maximum Transmission Unit (MTU) is 1500 bytes. If the data exceeds 1500 byte, the frame needs to be fragmented.

Cyclic Redundancy Check (CRC) is used to detect bit errors at the receiver side.

Interframe gap is a 12 byte gap between each Ethernet frame.

2.1.5 IPv4

Internet Protocol version 4 (IPv4) is the protocol used for forwarding and addressing over the internet. Today there exist two different versions of this protocol, i.e. IPv4 and the newer one IPv6. IPv6 is not used in this project, so it will not be handled in this thesis.

IPv4 operates in the internet layer. In internet layers a packet, called a datagram, is divided into several fields. The different fields fulfil different functions. The layout can be seen in Figure 2.3 [12]. In this project the Options field is never used, thus the total header length is 20 bytes.

Version number specifies what version of IPv4 is used.

Internet header length specifies the length of the datagrams header. The length may vary depending on the type of IPv4.

Type of Service (ToS) specifies QoS desired by the sender for the datagram when it travels through the network.

Datagrams length is used to specify the length of the whole datagram.

16-bit identifier contains flags and a 13 bit fragmentation offset. These fields represent information about datagram fragmentation.

Time to live (TTL) is used by the routers and switches to identify if a datagram has arrived later than its decoding time.

Protocol contains the protocol type of the next level.

Header checksum flags if the header has been corrupted. Error check is done by using 1 complement.

Source IP address stands for IP address of the transmitter.

Destination IP address stands for IP address of the receiver.

Options this field is seldom used, the datagrams that are used in this project do not have this field.

Data is the field where payload is located.

2.1.6 UDP

The User Datagram Protocol (UDP) is a protocol mainly used for sending media over networks. UDP has an uninterrupted service to its application but no data flow control, no congestion control and no retransmission. The absence of control mechanisms has, of course, side effects on the data stream. The packets can arrive out of order, i.e they can be dropped or duplicated. In media streaming, which is the focus of this report, very little is done to tackle these problems. UDP has capabilities to check if error occurred during the transmission by looking at the checksum. If an error is detected, it will only drop the segment and no further action will be taken. The checksum field in UDP is optional and the user can decide whether to use it or not. In this project checksum is not used. The UDP

header consists of 8 bytes, see Figure 2.4. It consists of four different fields all 16 bits long; Source port, Destination port, Length and Checksum [5, 10] .

Source port identifies the port of a sender. This field is optional.

Destination port is a port of a receiver. A destination port must be specified.

Length specifies the length of a datagram in bytes. The length can vary between 0- 65527 bytes (if the length is zero then only the header is sent). The practical maximum is 65 507 bytes, leaving 20 bytes for the IP header.

Checksum is used for error checking in the header and the data. This field is optional.

The UDP protocol does not closely co-operate with a layer below the IP protocol layer. Therefore, a pseudo - IP header is added to a UDP packet. This header is 12 bytes large and contains information about: version, length of the header, length of the data, identification, flags protocols, header checksum, destination and source addresses. This information is added for extra protection for incorrectly routed datagrams [13].

Bit offset	0-3	4-7	8-15	16-18	19-31
0	Version	IHL	Type of service	Total length	
32	Identification			Flags	Fragment offset
64	Time to live		Protocol	Header checksum	
96	Source IP address				
128	Destination IP address				
160	Options (if header length >5)				

Figure 2.3: The fields in the IP header

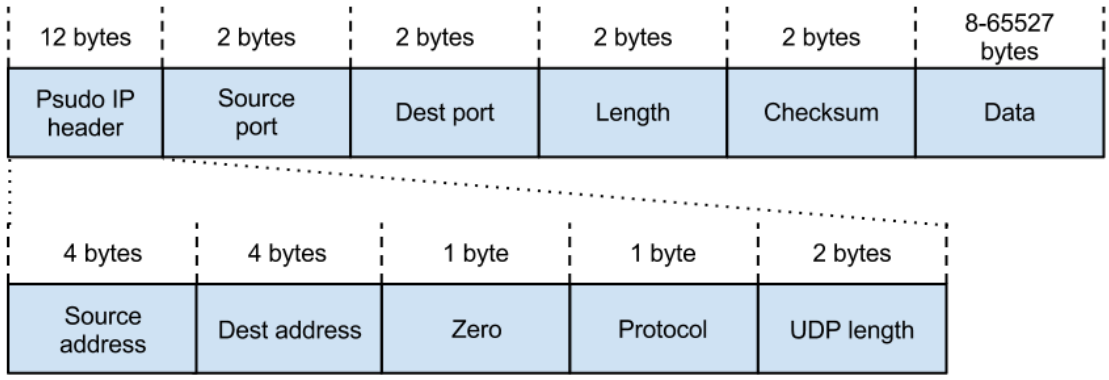


Figure 2.4: The elements of a UDP frame

In this project UDP datagrams are packetized in Ethernet frames. The Ethernet frame size allows an MTU of 1500 bytes information with headers. Though a UDP datagram can contain up to 65535 bytes with headers, it is truncated into smaller frames when carried through the network. UDP datagrams can include up to 42 Ethernet frames. In this case each Ethernet frame that has been sent contains only a part of the video frame. The last Ethernet packet of the video frame has a smaller size, carrying only the rest of the information and signalling that the frame has ended. The Ethernet frame that is used in this project is never larger than 1066 bytes. With 4 bytes interframe gap the maximum size of the frame is 1070 bytes.

2.1.7 RTP/RTCP

The Real Time Protocol (RTP) is widely used in multimedia applications such as video conferencing programs that need streaming of live data. It was developed for sending real-time audio and video. One of the big advantages of RTP is the ability to provide a multicast transmission, not only unicast [5]. Multicast enables to send one-to-many or many-to-many end users.

RTP packets are compiled in the level above the UDP level. The media stream is packetized in RTP. When the receiver obtains a UDP segment, it extracts the RTP packets. The receiver decodes the packets in the application layer. The RTP

protocol is only seen by the end program/application. All other services will only detect UDP.

The RTP header has a minimum size of 12 bytes, contains four main fields and six smaller fields listed below and is visualized in Figure 2.5.

Version is 2 bits long and specifies the type of payload.

Padding (P) is 1 bit long and marks if a packet is padded. When the buffer overflows it discards the smallest packets, otherwise the buffer discards packets randomly.

Extension (X) is 1 bit long and indicates if there is an extension in the header.

CRSC Count (CC) is 4 bits long and contains the number of CRSC identifiers.

Marker (M) is 1 bit long. In Confero it indicates the last packet in a video frame.

Payload type is 7 bits long and specifies the type of encoding. If the sender changes codec in midstream, it updates this field.

Sequence number is 16 bits long. All frames sent from the transmitter have a sequence number. Receivers identify missed packets by tracking the sequence number.

Time-stamp is 32 bits long and is used for the receiver to know when to play the samples. It is not used by synchronization between media streams such as video and sound.

Synchronization source identifier (SSRC) is a 32 bits long field that identifies the source of the RTP stream.

Contributing source IDs enumerate contributing (CSRC) is a 30 bit long field and this is used to identify the contributing sources such as audio and video [5, 14, 15].

The RTP standard is used in pair with the Real time control protocol (RTCP). RTCP is not used to send any media information. It sends periodically statistics about the jitter, packet drops and number of packets. It acquires this information by monitoring the sequence number, the size and the time stamp of the packet sent. The RTP statistical information can be used by any other application at both sides[15].

2.1.8 Multicast

Multicast is most commonly used when the sender wants to send information simultaneously to many participants in a one-to-many type distribution. The sender's data is automatically copied in either the router or the switch. There exists other types of routing schemes that could be used, but multicast is the most commonly used for media streaming and internet TV. The advantage of using multicast instead of broadcast is that it saves bandwidth. The data is transferred to those participants that are connected to the multicast address unlike broadcast that transferees data to all the computers in the network. Reduction of the number of participants that receive transmitted data decreases utilized bandwidth. It works as follows: the sender sends out the data to a single address that is unique for each multicast session. Every participant in this group will receive the data sent to this address. The address span that is dedicated for multicast is 224.0.0.0 - 239.255.255.255. In order for multicast to work over larger networks, several protocols are needed. One example is Internet Group Management Protocol (IGMP). IGMP is used to avoid the flooding with an unrequired traffic. In local network with one switch, as used in this project, IGMP is not necessary. In LAN the multicast acts as a broadcast and send to all ports [5, 16].

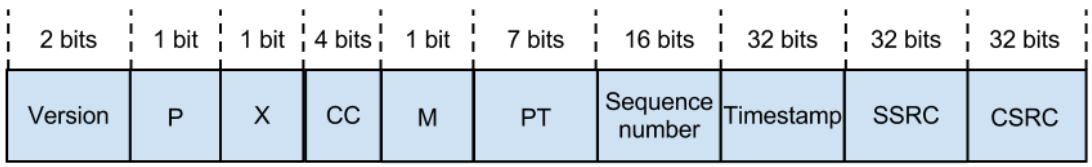


Figure 2.5: The RTP header

2.1.9 QoS

Modern networks need to be able to handle traffic that is delay sensitive, for example video or audio data streams. This traffic is sensitive to packet loss (packet loss introduces image distortions and audio disturbance after decoding) and require high bandwidth usage. In order to guarantee the possibility to use these applications, a certain level of quality in the network is required. This term is known as QoS. QoS addresses the delay in the network, throughput, jitter and packet loss.

In the following subsections, a short summary of the different parameters will be given.

2.1.9.1 Delay

Delay is a description of how much time a single bit spends in the network, usually represented in seconds. Delays in a network can cause packet drop, but unfortunately it is impossible to completely avoid them. Every node of a systems adds delay. Processing delay, Queuing delay, Transmission delay and Propagation delay are among the delays that can significantly effect the system.

- Processing delay is introduced when switches and routers examine the header of packets in order to determine their destinations.

- Queuing delay is equal to the time that packets spend in the switch's buffer and before they are transmitted. The delay can vary from packet to packet. If a burst of packets arrive to an empty buffer, the first packet will not experience any delay but the last one can suffer from large delays. In addition the packets may arrive at random time intervals [5, 17].

- Transmission delay or store and forward delay, is the amout of time needed for a packet to be transmitted through a given link. It could be estimated with the equation

$$dt = L/R \tag{2.1}$$

  where dt is the transmission delay, L is length of the packets in bits and R is the rate of the transmission in bits/sec.

- Propagation delay is the time between sending and receiving the last bit of a packet. The propagation delay depends on the speed of the medium. The propagation delay is calculated as the distance between the sender and the receiver divided by the speed of the transmission.

2.1.9.2 Throughput

Throughput is the measure of the network capacity and is specified in bits per second (bps). Throughput represents the actual amount of traffic that can pass through the system. Throughput computations can be done to estimate the performance of the different network elements. Throughput is defined as a number of bits divided by the time that takes to transport them. This calculation an be adapted relatively to the application [5, 18, 19, 20].

In theory delay and throughput are independent of each other, but practice can show different result. If the traffic in the network is transmitted close to its maximum throughput, it will increase the delays in the network. When the delay and throughput of the network are known, it is possible to calculate the volume of the data that is in the network. This is called the delay-throughput product. This product is a constant and is defined as:

$$T * D = C \tag{2.2}$$

where T is the throughput, D is the delay and C is a constant. With this equation it is easy to see that in order to increase the throughput the delay has to be decreased [18, 20].

2.1.9.3 Jitter

Jitter is a term that describes the fluctuation of the delays in the network. In datagram switching networks jitter can also be referred to a problem of nonsequential packet order at the receiving end. In audio and live media streams it is important that packets arrive in the successive order. If audio packets arrive with a large delay or are dropped it will create disruption in the decoded sound. In video streams jitter can cause image distortion. Audio stream is more sensitive to

jitter than video stream because video codecs provide algorithms to compensate possible losses of the data. Whereas is much harder to do with audio [18, 21].

2.2 Video codecs

This section gives a summary of different video codecs used in this project.

2.2.0.4 MPEG-2

Moving Picture Experts Group (MPEG-2) is a codec built around the discrete cosine transform (DCT), motion compensation and entropy coding. DCT is a mathematical model for transforming the information in each video frame from time domain to the frequency domain. Using this model the high frequency information is discarded in favour of higher compression since the human eye is less sensitive in perceiving the high frequencies. After quantization step the interframe coding with motion compensation is done. Each frame is divided into 16 by 16 pixel macroblocks. These macroblocks are then matched with a region in a previous frame by help of motion vectors. This way of video coding is normally called temporal prediction with motion compression. It gives good compression rate compared to only using intraframe coding. In intraframe coding each frame is compressed, but temporal prediction is more sensitive to errors. To solve this MPEG-2 uses some intra coded frames (known as I frames) in intervals, enabling the resynchronization if errors would occur. The temporally coded frames are known as P frames. MPEG-2 normally uses bidirectionally predicted frames, known as B frames. These frames are predictions from the previous and subsequent frames in the video stream.

2.2.0.5 H.264/MPEG-4/AVC

H.264 Advanced Video Codec (AVC) was created as a joint project between ITU-T Video Coding Experts Group (VCEG) and International Standardisation for Organisation (ISO)/International Electrotechnical Commission (IEC) Motion Picture Experts Group (MPEG). H264 uses both spatial and temporal compression.

As MPEG-2 H.264 uses I,P and B frames. Each of these frames are then divided into at least one block. Doing this increases the resilience against errors. Compared to MPEG-2, H264 gives normally double the compression at the cost of complexity. H264 is approximately four times as complex for the encoder and twice for decoder compared to MPEG-2.

2.2.0.6 M-JPEG

Motion Joint Photographic Experts Group (M-JPEG) is one of the codecs used in this project. It works by encoding each video frame independently of the other frames as still images. It is similar to how MPEG and H.264 uses I frames but with the difference that M-JPEG does not utilize any P or B frames. The codec is based on the same compression algorithm as used in JPEG for still images. It uses a lossy version of intra-frame compression based on DCT.

In Confero M-JPEG quantization parameters can be chosen from the range of 0-100. Zero, in this case, corresponds to the maximum compression. In this project the setting for the quantization will only range from 50-100, meaning that compression of the images will change between 30:1 to 4:1.

M-JPEG uses only the spatial redundancy and not temporal redundancy like MPEG or H.264, it is less complex than the others, at the cost of a much higher bandwidth usage.

2.3 Network simulation and estimation software

This section gives a short introduction to the programs used to test the network and create the programs Capture Packet - Extract Parameters (CP-EP) and Switch-Simulator.

2.3.1 TCN TimeAnalyzer

TCN TimeAnalyzer (TA) is a product currently under development by TCN. TA is a tool for analyzing and modelling the worst case scenarios in switched Ether-

net networks by calculating forwarding delays of Ethernet frames. Packet drop estimations also include the possibilities of predicting switch memory overflow.

A network is evaluated by virtual simulation of data flows transmitted from end users that are connected via switch. It is possible for the user to create Ethernet frames depending on the main protocol used (UDP or TCP). The user also needs to set parameters such as size, interval and maximum time of delivery. These parameters will later be used in the latency calculations.

When creating flows for the network, the user compiles them from the frames, imitating the real observed flow, e.g. an MPEG-2 flow consisting of I,P and B frames organized according to the following order IPBBPBB. Each I, P or B frame carries different information to the receiver and thus have different size, not necessarily the same as its' neighbour. Therefore the real flows will be simulated by repeating the same frame order and size accordingly. Traffic is simulated right after all the flow and network parameters have been specified. The result is saved in an Hypertext Markup Language (HTML) document. The saved outcome specifies the worst case of buffer utilization. It thus informs the user about the possibility of a buffer overflow. The worst case forwarding field delay provides information about comparison between the maximum pre-defined and estimated delays. If it exceeds the maximum allowed delay then the field is red, otherwise it is green. The two other fields correspond to the best case forwarding time and the worst case jitter. The later is calculated by a simple subtraction between the worst and best forwarding delays, as shown in Figure 2.6 and 2.7.

2.3.2 TCN StreamAnalyzer

TCN StreamAnalyzer (SA) is also under development by TCN. SA provides convenient visualization and verification of network traffic. By testing Ethernet traffic in switched networks it is possible to see the latency, packet drops, serialisation time and jitter for each network packet. SA is currently able to operate only with .hea files. These files are recorded by netANALYZER software that is provided along with the Hilscher network card. NetANALYZER records and saves all the

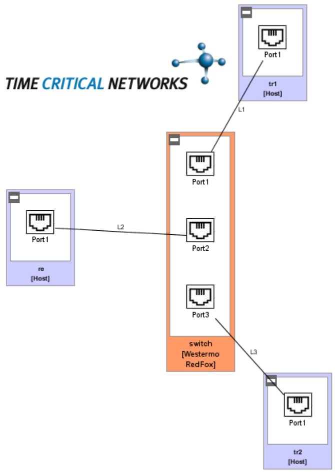


Figure 2.6: A picture of the test system in TA

Memory Buffer Constraints

Switch	BC (bits)	WCBU (bits)
switch	700000	80896

Frame Delay Constraints

# of flows considered: 2

Flow: video2

Frame	Path	D (µs)	WCFD (µs)	BCFD (µs)	WCJ (µs)
video	tr2.Port1 switch.Port2	7700	183	180	3
video	tr2.Port1 switch.Port3	7700	273	180	93
video	tr2.Port1 switch.Port2	7700	183	180	3
video	tr2.Port1 switch.Port3	7700	273	180	93
video	tr2.Port1 switch.Port2	7700	183	180	3
video	tr2.Port1 switch.Port3	7700	273	180	93
video	tr2.Port1 switch.Port2	7700	183	180	3
video	tr2.Port1 switch.Port3	7700	273	180	93
video	tr2.Port1 switch.Port2	7700	183	180	3
video	tr2.Port1 switch.Port3	7700	273	180	93
video	tr2.Port1 switch.Port2	7700	183	180	3
video	tr2.Port1 switch.Port3	7700	273	180	93
video	tr2.Port1 switch.Port2	7700	183	180	3
video	tr2.Port1	7700	273	180	93

Figure 2.7: The result from TA

Ethernet frames passing trough the ports of Hilscher network card. Then, these frames can be analysed by SA offline.

NetANALYZER card has two TAPs (Traffic Access Points) and each TAP has two

ports, in total there are four ports. Port 0 and port 1 belong to TAP A. In the same manner, port 2 and port 3 belong to a TAP B. Both TAPs receive and send packets. Ports 0 and 3 are typically connected to the end users, ports 1 and 2 are connected to a switch, enabling to increase the number of participants in the network, see Figure 2.8. With this setup traffic from two computers can be monitored, measured and time stamped with an accuracy upto 10ns. Measurements were isolated from any outside influence to produce as exact results as possible. The traffic from each TAP is then recorded by the program and saved to a .hea file. SA uses the .hea file as an input to generate plots of network traffic for every port. Figure 2.9 shows captured traffic by TAP 3. Four bursts of packets are clearly seen in the figure (beige "triangles"). In addition packet bursts experienced delay in the network, that explains why forwarding time of every successive packet is increasing.

The SA Graphical User Interface (GUI) is divided into two parts. The bottom window gives the static information of the measured traffic: source and destination addresses, protocol type and ports. The last field if the traffic is parallel or sequential. If the field is set to parallel, it means that the traffic has only been measured over one tap and Sequential means that the traffic has been measured over two taps. This is because only this traffic flow has a valid starting and stopping time.

SA provides RTP statistics over the saved traffic. Based on tracking packet sequence number in each flow the program is able to display the number of dropped packets, if any, and intervals of loss, see Figure 2.10.

The plot makes it possible to see the maximum delay on individual packets. It is also possible to zoom in to see the serialisation time off the packets.

2.3.3 Confero

Confero is a real-time multimedia communication program developed by the company Alkit Communications AB. The main application of this program is online video conferences.

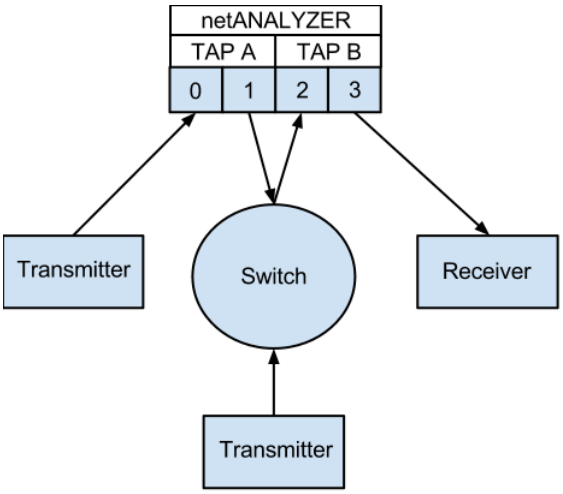


Figure 2.8: Network test system

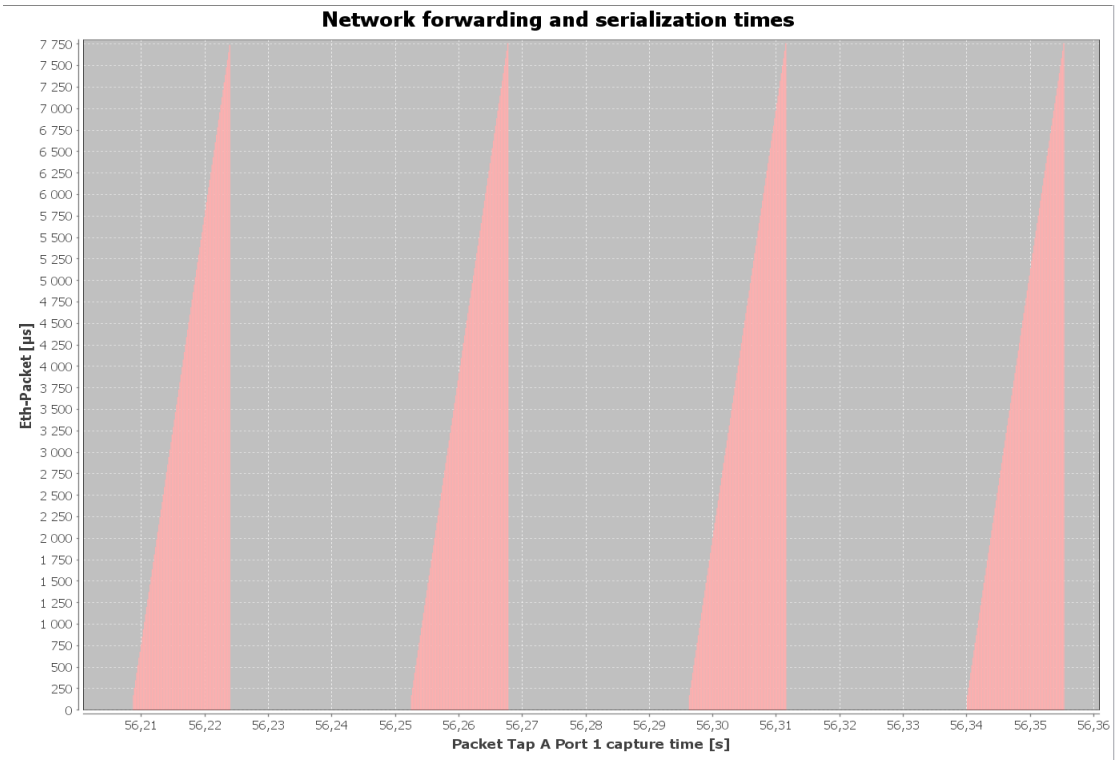


Figure 2.9: SA plot of packet bursts with increasing packets' queuing delays

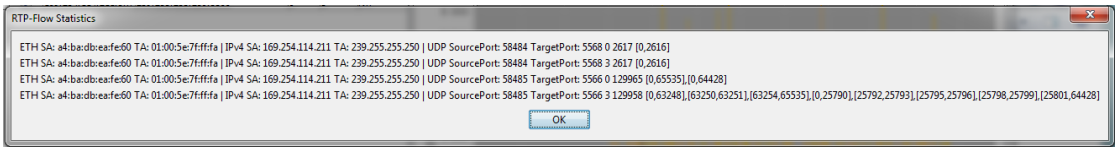


Figure 2.10: Measured packet drops in RTP statistic from SA

In this project, the program fulfils different functions along the process of work. In the beginning of the project, it is used to create traffic in the network through sending video and audio streams between computers. The quality of the video stream is increased until the moment when the switch cannot handle the traffic without a packet loss. After this critical point, Confero will then be used to see if this point can be avoided by knowing the traffic load.

Confero gives the user a lot of freedom to choose among different settings. The main settings that will be changed in this project is the audio and video codec, quality settings and sizes of the video frames. The video codecs that Confero provides and that are used in the project are: M-JPEG, MPEG-2 and H.264. These codecs have different compression rates, complexity and corresponding quality. The user is able to alter bandwidth use for transmission by changing the video parameters. The default audio codec is G.722.1. Both the video and audio is sent over the network as RTP.

2.3.4 Wireshark

The open source software Wireshark has been built over the WinPcap tool including functions for network traffic analysis. Wireshark is a convenient user interface program that allows direct and easy access to the network packets and makes the information acquisition simpler [22]. As shown in Figure 2.11, Wireshark displays packets with source and destination IP addresses, protocol, frame size and additional information about each packet. Headers are decoded and can be read by the user.

A small set of basic Wireshark functions have been used in the beginning of the project to provide a user with information about the inner structure of the packets

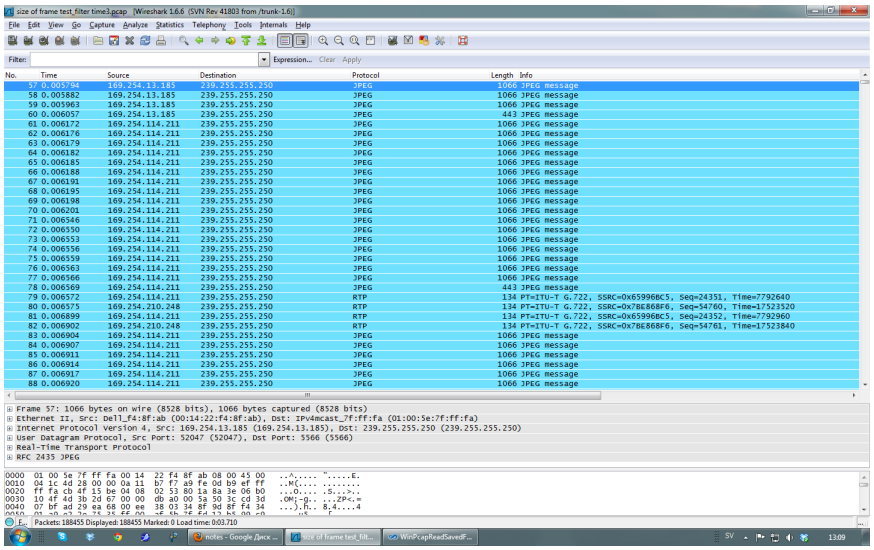


Figure 2.11: Wireshark monitoring Confero traffic

sent in and out by a running Confero program. Ethernet traffic captured by Wireshark can be saved as a .pcap file.

2.3.5 WinPcap

One of the softwares used in this project, is an open source packet capturing tool called WinPcap. WinPcap was created for Windows machines. It allows a real-time access to the raw Internet packets at the kernel level. WinPcap includes libraries that provide the ability of packet processing right after the network adapter has captured them and more complex processing at the higher application levels. The access to the network adapter is provided by a network interface driver called Netgroup Packet Filter (NPF), that is a part of the WinPcap installation pack. The libraries provided are packet.dll and wpcap.dll. The first one, as mentioned,

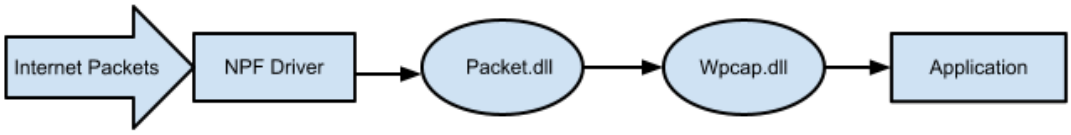


Figure 2.12: Schematic picture of how Winpcap work

allows working independently with the Internet traffic captured by the card while the second is used at the high application level of the stack, providing different functions, depending on the application purposes, see Figure 2.12 [23].

The project uses the WinPcap libraries to capture internet traffic and utilizes the filter function to obtain only UDP packets from the network, then it processes them further for the header analysis retrieving necessary information for the network analysis.

3

Network load testing

The greater part of the project is devoted to investigating the switch behaviour when traffic has different origin and load. Three major tests were carried out: real-time observation of RTCP statistics, stream reproduction in artificial environment and network simulation with TA.

3.1 First test: Network setup

Below, we describe how the network setup was tested for diagnostic purposes. At this point of the project it was needed to learn.

- Is it possible to create enough traffic to fill the switch's buffer? If possible then which traffic parameter leads to this result?
- How does the network traffic look like? What protocols were used? How many frames were required to send the video frames?
- How does the network behave under heavy traffic load?

3.1.1 Hardware setup

Measurements of the switch buffer capabilities is an important part of this project. It is required for further tests and project development to acquire the switch buffer

limits. To keep the setup as simple as possible it was decided that three computers would be used in an initial testing setup. These computers would be connected by a single switch in a star shape topology, see Figure 2.1.

In this test two switches were used: D-Link DES 1005d and D-Link 1008d. DES 1005d is a five port, unmanaged, store and forward, layer 2 switch with a total Random Access Memory (RAM) buffer of 512 Kbits. The difference between the switches is that the DES 1008d has 8 ports and 1 Mb in RAM. The reason for choosing these switches is that they are simple to manage and widely used. The Ethernet cables connecting the system are standard 100BASE-TX Ethernet cables.

Three computers have Confero running during the test. Two of the computers are equipped with basic web cameras. These two computers are transmitters and generate the traffic in the network. The third computer acts as a receiver, Figure 3.1. It was estimated that one computer alone could not create enough amount of traffic to overflow the switch. With this setup the traffic from the two transmitters will be concentrated at the receiver port. Caused by the difference in speed between incoming and outgoing traffic, a queue is created in the switch.

3.1.2 Software setup

In the first test, two programs were mainly used, Confero and Wireshark. Confero generates video traffic in the network while Wireshark surveys and records propagated packets inside the network. Transmission is saved by Wireshark for a detailed examination of the traffic produced by Confero.

Tests of real-time systems have many factors and variable that are difficult to control and they may influence the final result. Sources of flaws and errors are also problematic to find. Parameters that can be controlled by Confero are:

- Transmission type. Confero has the ability to use either Session Initiation Protocol (SIP), unicast or broadcast transmission. In order to keep things as simple as possible multicast was used.

- Video size settings were changed between small (160*120 pixels), medium (320*240 pixels) and large (640*480 pixels). The size S of the video frame in bytes is directly proportional to the number of pixels N and can be calculated as 3.1

$$N * 24/8/1024 = S[Kbytes] \tag{3.1}$$

  without any compression.

- The frame rate can be changed between the default value of 25 frames per seconds (fps) to the maximum 50 fps. Altering the frame rate causes changes in the sampling time of each video frame. This was mainly used to increase the likelihood of data stream contention in the switch port. With more frequent bursts of packets, the probability of collision is increased.

- The quantization (Q) setting is the compression rate of the codec. The Q

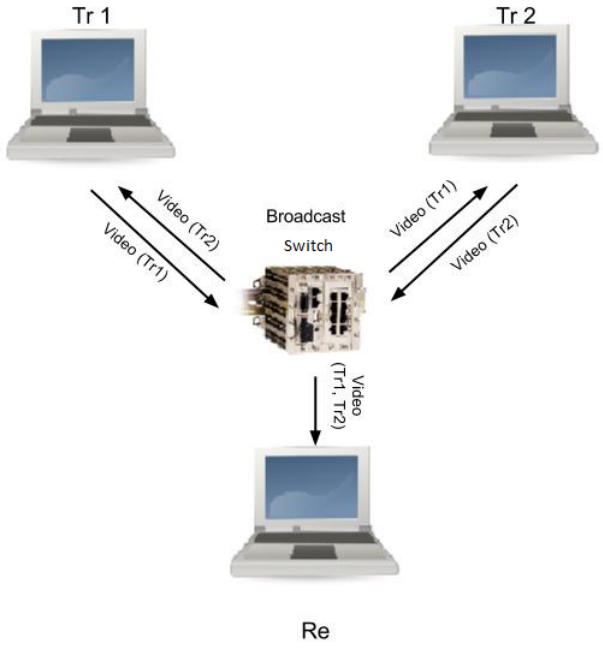


Figure 3.1: Test network configuration

scale in Confero is between 1-100 and the default setting is 50. When using $Q = 50$ on e.g a M-JPEG codec, the compression rate is about 30:1 (for stable outcome a test picture was used). When using $Q = 100$ the compression is only about 4:1.

- Codecs used during the test were M-JPEG, MPEG-2 and H.264. They were chosen because of their complexity and compression rate.

All computers used Windows 7 as operating system.

3.1.3 Test results

A built in Confero function monitored RTCP packets in the network and displayed the collected information. RTCP packets convey statistics of the transmitted multimedia flows of every participant, see section 2.1.7. Packet drops were achieved with high Q settings on the M-JPEG video codecs. Drops occurred when Q took values from 97 to 100 while using the test picture with a size of 640*480.

Wireshark was running on one of the machines recording the transmitted data streams. In the end of each transmission (one for each Q value), received packets were saved in a .pcap file for analysis. The saved traffic contained expected RTP video, audio and RTCP packets. A correlation between increasing numbers of RTP frames and higher Q values can be seen in Figure 3.2. The packets' size did not correspond to the maximum Ethernet packet size. Ethernet packets, as mentioned in section 2.1.4, are able to have the maximum size of 1500 bytes/packet. Video packets sent by Confero had the largest packet size corresponding to 1066 bytes [1]. The packet header occupies 42 bytes leaving 1024 bytes for the payload.

Audio packet size remained constant over time as expected. G722.1, that was the default audio codec for Confero and continuously used throughout the project, had a packet size of 134 bytes.

[1] Alkit did not have any particular reason for this frame size. It was something they started to use and then did not change as the work progressed.

All test equipment was disconnected from the Internet. Nevertheless, a number of default applications tried to connect or synchronize with the Internet. Therefore, multimedia flows had a small level of ”pollution” caused by default Windows 7 system applications. Even though extra traffic represented a very small part of the transmitted data it could still influence the results and needed to be avoided as much as possible.

Apart from packet loss and extra traffic discovered by observation, it was found that heavy network load causes decline in a frame rate. Further tests that involved other machines lead to a conclusion that it was the graphic card that could not process a high quality video flow.

Audio showed to be more sensitive to jitter. Certain sound disturbances appeared in high quality flows. These were not caused by packet loss but by jitter in the network. According to RTCP statistics, audio packets arrived to the receiver, but too late to be decoded within the given multimedia conference session. Audio packets were dropped by the receiver causing clicking disrupting sound.

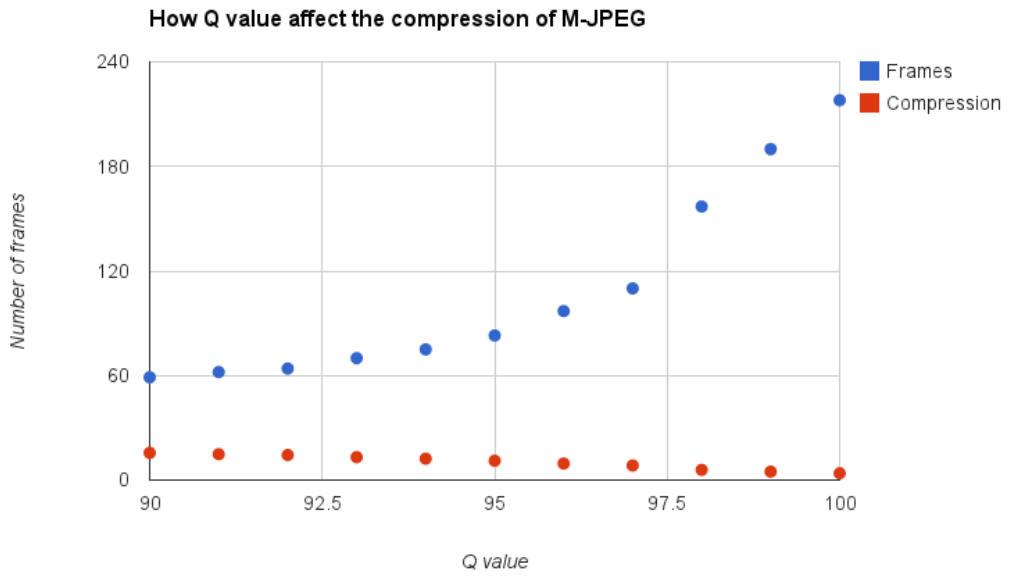


Figure 3.2: The effect of the Q value on M-JPEG compression

3.2 Second test: Testing programs

The Second test involves extra elements as a Hilscher network card (netANALYZER) and an industrial switch. These additional components provide exact time resolution for each packet and provide tools for traffic analysis.

3.2.1 Hardware setup

Due to the graphic card problems, changes in the hardware setup was needed. Substituting the laptop at the transmitter side minimized the risk of losing data at the receiver side. Limited company resources did not provide possibility for substantial changes of all the elements in the tests, but the likelihood of dropping in frame rate was minimized.

Some initial testing using SA showed that there was some extra transmission previously unknown from the earlier testing. DES 1005d and DES1008d switches have a built in flow control function. This function sends a pause frame to all senders when the buffer is about to overfill [24]. This flow control became a problem. It became much harder to define exactly when the switch will drop packets.

In order to continue measurements, it was decided to the change switch to a Westermo RedFox. RedFox is a 8 port managed layer 3 switch created for industrial applications. This switch has a buffer size of approximately 700000 bits [25]. RedFox also has the flow control function. However as RedFox enables manual switch managing through an access interface, the flow control function can be turned off then.

3.2.2 Software setup

The second test was meant to provide more accurate measurements of the network traffic and switch behaviour. Instead of relying only on the built-in Confero functions and Wireshark surveillance, this test also used netANALYZER and SA. Prior to the test execution, the network was simulated in TA with settings from the previous test. The TA simulation has to create a clear understanding of the

network and all included elements. TA is mostly used to predict and emphasise what flow settings measurements have to be focused on. The model was only created to simulate the M-JPEG video flow, due to the fact that M-JPEG was the only codec that can achieve the targeted packet drop. H.264 and MPEG-2 was also inconsistent in the numbers and the size of frames used for a predetermined video frame.

3.2.3 Test results

Finding the lowest setting where packet drop occurred using only Confero is problematic, since the program settings do not provide a high precision control of the traffic size parameters. In order to handle this problem, a test model was created in TA. This model would then serve as a reference point for initial settings for the measurements.

Problems with TA became apparent during testing of the model system. TA is a prototype product and was never tested for large flows. Running test with a large flow took more than 8 hours. It was originally planned to incorporate TA with the CP-EP program, but due to the long computation time of the program this was not efficient. Results also gave incorrect time delay estimation of individual packets, though it was not a target goal of the test. The only outcome that was of interest was the prediction of the worst case buffer utilization. The prediction gave expected result. According to the simulations, the RedFox switch buffer is full when a burst of UDP datagram has a size correspondent to 85 packets. That amount of packet can be converted to a Q value corresponding to 95-96 when using the test picture.

The test result from TA was then applied to the real test station. Firstly the Q value was set to 96 (Q=96 corresponds to 97 frames). Running a test with these settings showed packet drop in the system. Lowering the setting to Q = 95 indicated no packet loss. Because the drop is not measured straight away, it cannot be defined that no drops will occur in the future. So it was decided to leave the test running for a long period. It did not yield any definitive result. Drops were measured but a very low count. After running the test for two hours, only

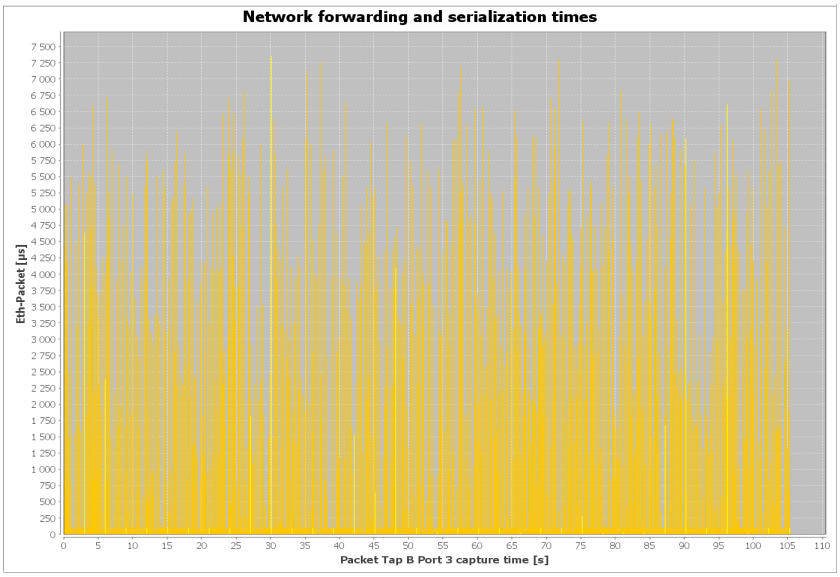


Figure 3.3: Traffic captured from Confero, representing forwarding time of each packet burst in μs

40 drops was detected. These drops were only measured from one test, and were recorded when one computer started from the screen saver mode. In a later test, in which the screen saver mode was turned off, no drops were recorded.

When looking at the Confero traffic in SA, packets' forwarding time was randomly distributed, see Figure 3.3. The packet time delay over 7500+ μs will cause packet drop in the switch. From these measurement of the switch it was not possible to give a more accurate value of the maximum delay.

3.3 Third test: Simulating traffic

This test was created by the information learnt from earlier tests. In this test all the traffic was created by traffic simulations, in order to control the parameters as much as possible.

3.3.1 Hardware Setup

The new test station consists of three IAR cards. These cards have the ability of creating a burst packets as traffic in the network in accordance with specific required parameters. With these cards it is then possible to find a time when flow bursts collide. The test quickly gives the worst case scenario, avoiding spending time with Confero real-time transmission.

IAR cards have the IP addresses: 192.168.0.10, 192.168.0.20, 192.168.0.30. These card can operate both as a transmitter and a receiver. The network configuration setup remains the same. Three end users are connected through a netANALYZER card in a star shaped topology. Unfortunately only three cards were available when this test was carried out. Two cards act as transmitter and each generates the simulated video traffic. The third card acts as the receiver of the simulated video stream.

3.3.2 Software Setup

The only new program introduced for this test is the control program for the cards. In this program it is possible to set necessary parameters. Parameters changed under this test were:

- Packet size of each Ethernet packet.
- Delay between packets within one packet burst.
- The number of packets in one burst.
- The number of bursts in the stream.
- Delays between the bursts.

Figure 3.4 shows the structure of the simulated video traffic. One packet burst, in this simulation, represents one video frame, where packets, consequently organized and encoded, carry chunks of the video frame data. The time distance between packet bursts is the time difference between sending each video frame.

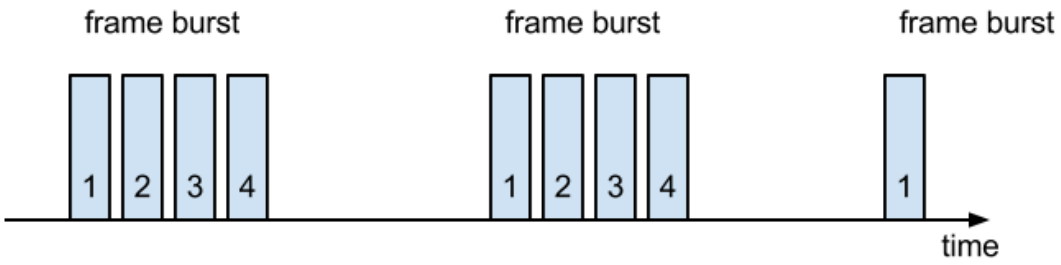


Figure 3.4: Ethernet data stream frame burst in time

3.3.3 Test results

When measuring the traffic the two transmitting cards would start sending at the same time. The traffic consisted of ten bursts of frames. One burst represents one video frame. The IAR cards introduce time drift in the frame bursts when the number of bursts exceeds ten. After each burst they got little more out of sync, restraining the possibility of achieving and controlling packet collision.

There was also a mismatch of interframe gap between simulated and real transmissions. According to the analysis by SA, interframe gap was 20 bytes or 1,6 μs: interframe gap (12 byte) and the preamble (8 byte), see section 2.1.4. Nevertheless the interframe gap of the simulated flow corresponded to 24 bytes. The additional 4 bytes appeared to be appended for Ethernet specification, resulting in a total frame size of 1070 bytes. Ethernet uses CRC in order to detect errors in the transmission. When looking at the traffic generated by Confero it had the same extra four bytes. CRC is added and removed by the hardware. Wireshark connects to the lowest levels of the IP stack after CRC was removed, and is therefore unable to display it.

After fine tuning the measurements, it was possible to predict the amount of frames needed to achieve packet drops. According to the test, the limit for the switch is 87 frames. This result is somewhat higher than the prediction made from TA. TA predicted that packet drop would occur at 85 frames. But both predictions are in the order of a Q value between 95-96. The most likely reason for the difference in value is that in TA the switch buffer has a size of 700kb [25]. This test however,

does not use any estimation of the buffer size. Switch buffer estimation was done according to the following calculations

$$\frac{L}{P_t} = P_n \tag{3.2}$$

$$P_n \cdot 8 \cdot P_l = B[bits], \tag{3.3}$$

where L is the lowest latency before the packet drop, P_t is a time to send one packet, P_n is the maximum number of packets that buffer can contain, P_l is the size of every packet in bytes, B is the total size of the buffer.

The lowest latency measured with packet drop was around $L = 7650$ μs and the time to send one packet $P_t = 87.2$ μs (1070 byte + 8 byte preamble + 12 byte interframe). With these numbers, the maximum number of packets that the buffer can hold is $P_n \approx 88$. Using the P_n from 3.2 in 3.3, multiplying with 8 to convert to bits and multiplying with the length of one packet gives $P_l = 1090$. This gives the total buffer size to be $B \approx 770000bits$. These numbers are in accordance with the measured results. There was no conclusive evidence to why the buffer size has increased with 10% as compared to the previous test done by TCN[25].

The drawback of creating the traffic in a test situation is that it is hard to create a transmission that would exactly mimic the real traffic from Confero. In the constructed test, it was not possible to have audio sent from the same card as the video. This implied that an extra card was needed to send the audio. At the time of testing there was only three cards that were fully functional. As a result, the RTCP was not represented in the test model. The effect of this is fortunately not a big issue. When Confero sends the M-JPEG stream the last packet is smaller than the other if RTP padding is not turned on.

3.4 The developed programs

The program CP-EP, that was developed for packet capturing, is based on the WinPcap library. The WinPcap library is an open source library and has been

described in detail in section 2.3.5. For a more detailed description of the development environment setup see Appendix A .1

3.4.1 Obtaining network traffic parameters

WinPcap provides access to the raw packets that are received by the network card. Packets are still unprocessed by any application of the higher levels of the IP-stack. Raw network packet are picked by the CP-EP with the help of the WinPcap driver and functions provided with it. Offline packet handling showed to be more reliable, therefore a specified number of captured packets from a transmission is saved on a hard drive as a .pcap file. Saving data into a .pcap file enabled parallel control of program execution by Wireshark.

The raw network packets have been received in an unordered manner from different sources. Therefore the first procedure was aimed to reading and decoding packets' IP headers and organizing them according to the data stream parameters. This is required to handle each data stream separately later on. One data stream is made up of network packets that have the same source and destination IP addresses and port. According to this classification one source can send out data from different applications. It is important to be able to separate these data from each other. Moreover, this specification enables to conclude how many participants there are in the conference and what data streams are exchanged between them.

Organized by the IP addresses and ports data streams have to be analyzed from the point of frame bursts, since the project deals with video streams. Each video stream would have a certain structure, that can be described as periodical frame bursts. The stream structure is visualized in the Figure 3.4.

One condition is that the network topology and data stream parameters do not change over the observed time period. The duration of the frame bursts is constant, or within a small deviation in the order of nanoseconds. Taking that into account, CP-EP extracts the minimum time period for each data stream.

As was mentioned earlier, tests were carried out with different video and audio conference settings. All codecs, that were at the user's disposal, were tested to

study the data stream outcome for different codecs, their structure as well as switch and network behaviour. CP-EP is able to distinguish the video and audio codecs, used by Confero.

Finally, to have a clear picture of the data structure in the stream, CP-EP provided the size of a whole frame burst.

All the above obtained parameters were needed for simulating the exact data flow in TA and SwitchSimulator in order to predict the worst case scenario.

TA requires the size of the frame burst and the time interval between them. The payload type was needed for the user to understand the structure and size distribution within one UDP datagram.

For the readers convenience, a more detailed algorithm and the settings are explained separately in Appendix A .1 and section 3.4.2.

3.4.2 CP-EP algorithm

Prior to packet processing or even capturing the system has to have access to them. A protocol stack is an integral part of every network-handling machine and in this case it needs to be avoided. WinPcap provides a driver that interacts directly with the network card. Thereby getting around packet processing by the protocol stack. The later packet processing utilizes the packet.dll and wpcap.dll function libraries. Steps :

1. Find devices for packet capturing. Function used : **pcap_findalldevs_ex**. This function searches for devices located on a machine (can also access a remote machine) and lists them[26].

2. Open a device to begin capturing. Function used: **pcap_open**. This function opens a device among the ones listed in the previous step. It captures packets but keeps only the first specified number of bits[26].

3. Since the list of the devices is no longer needed, the list can be removed from the memory. Function used: **pcap_freealldevs**. This function erases the list of the devices obtained in step 1[26].

4. Having packets streaming in, it is time to filter out only those of interest. One constraints of this project is that only Ethernet/UDP protocol packets are under surveillance. Firstly it is necessary to select only Ethernet packets by finding out the link layer parameters. Function used: **pcap_datalink**. This function returns the link layer parameter, according to pcap.lib the parameter **DLT_EN10MB** corresponds to Ethernet (10Mb, 100Mb, 1000Mb, and up)[26].

5. The filtering proceeds with keeping only UDP packets. Function used: **pcap_compile**. This function builds a filter based on a set of filter parameters (protocol type, ports etc.)[26].

6. Apply filter to the current capture. Function used: **pcap_setfilter**. This function utilizes the output from the previous step and relates it to the current capture[26].

7. The filtered Ethernet packets are then ready to be saved into a file. Function used: **pcap_dump_open**. This function opens a file on the hard drive to write processed data to it[26].

8. At this point everything is ready for saving the data into a dump file. Function used: **pcap_loop and pcap_dump**. The first function captures a specified number (e.g. 10000) of filtered packets, and saves them as .pcap file with help of the **pcap_dump** function[26].

9. Prior to the next step it is required to change the format of the data coming from the source. Function used: **pcap_createsrcstr**. This function transforms the format, so the other application can read it. This is not always necessary [26].

10. Saved packets can undergo a header decoding procedure to obtain the required information for the analysis. Function used: **pcap_loop**, **dispatcher_handler**.

Pcap_loop reads every packet separately in the captured file and extracts headers using the **dispatcher_handler** function. The latter decodes IP, UDP headers and saves them into a global variable that contains only headers[26].

11. It is now possible to differentiate streams according to the source and destination addresses as well as port numbers. Function used: **CompareInStruct**. The function creates a two-dimensional array. Each row is related to a separate stream and its' columns contain stream parameters, with the IP addresses it also contains the time of packet arrival and size. The information is presented in a suitable format for further processing in the next step.

12. To have a better overview of the data captured, the video and audio codecs are decoded. Function used: **payloadType**. This function finds a payload type in the packet header. Certified payload types are documented in [27]. Depending on the type of the payload this function passes stream parameters to the **packet_size_jpeg**, **packet_size_h264**, **audio_parameters** functions.

    (a) In the case of the M-JPEG payload type. Function used: **packet_ size_ jpeg**. This function looks for the marker that is signaling the end of the datagram. If the marker is equal to 1, the current packet is the last in the captured datagram.

    (b) In the case of the H.264 payload type. Function used: **packet_size_ h264**. This functions searches for the packet that is know as the sequence parameter set. It is transmitted before any UDP datagram frame and sets video parameters for applications decoding video streams.

    (c) In the case of audio payload type (G.722 codec). Function used: **audio_ parameters**. This function obtains not only the packet size but the time interval as well. This is due to the fact that those parameters are constant if the codec is not changed during the recorded time. Therefore it is much easier to handle them simultaneously in one function.

13. Time intervals between video frames. Function used: **timePeriod**. This function reads time headers for every packet in the stream. If the time difference between two packets is much larger compared to the other inter packet time differences, then it is considered as inter video frame time interval.

14. After the parameter in steps 12 and 13 have been received, they are saved to a .txt file. Users can then open the file. This file is rewritten every time when the program runs.

3.4.3 Switch behaviour simulation

One of the reasons for packet loss is switch memory overload. Under the conditions when the sending speed of data streams is higher than the receiving speed. The buffer is then forced to store and queue packets before their departure. That leads to the buffer fills up with data quicker then it is able to stream it out. Finally this leads to complete overfill and the switch is unable to accept more data packet loss sets in. Ethernet flow control functions can lower the bit rate automatically by sending a pause signal, though it may not be enough to prevent loss in the transmission. Yet this project is based on calculating the worst case scenario, in the simplest possible case and for that reason Ethernet flow control was disabled. This is true for all the later simulations.

The CP-EP program provides the set of parameters for being able to understand the type of the network topology and data streaming within it. In the development stage, the simplest network configuration was used: two computers, connected with a switch, broadcasting video. Given the stream parameters, the next objective is to find out whether the switch will overflow and will start dropping packets or if it can handle the given streams without discarding any packets.

TA provides such an analysis however as has been mentioned earlier in section 3.2 tests may take several hours to run and in the conditions of real time applications such time delays simply are unacceptable. Therefore it has been decided to simulate switch behaviour separately from other parts of the project. That would give a clear picture of the switch behaviour in the given environment.

3.4.4 SwitchSimulator

All the tests in this project were carried out with a store-and-forward switch. This mean that the switch could send a packet only after it had been completely received and stored in the buffer. Therefore, the switch simulator operates in the manner described and showed below.

A small set of streams were simulated with the following input parameters: packet size, interframe gap, see section 2.1.4, time delay and number of packets. Time delay relates to the time difference between arrival time of the first packet of the some stream and the first packets of other streams that arrived later. In Figure 3.5 it can be clearly seen that packet 1 in stream 3 will arrive to the port prior to the other streams' first packets. The moment when the first bits were received by the switch is considered as the starting time. All the delays are related to that moment. The current switch simulation may provide calculations for various switch buffer sizes. Thus, buffer size is one of the input parameters.

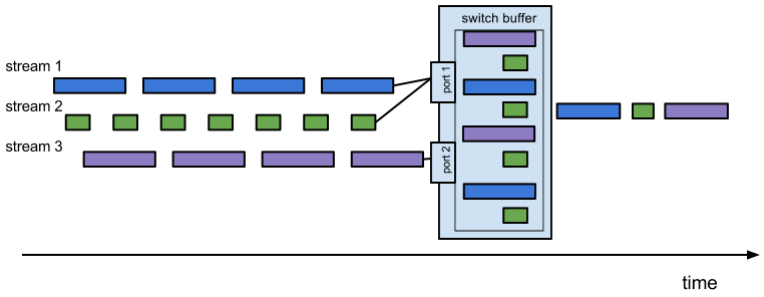


Figure 3.5: Packet storage in switch memory

In the simulations the speed of the input/output channels is 100 Mbit/s. The total number of bits received on the entry side is the speed of the channel multiplied by the total number of receiving ports. The output speed remains constant - 100 Mbit/s. Data streams are considered to be similar to M-JPEG video streams. Other video codecs as MPEG-4 and H.264, as mentioned in section 2.2, show better compression and adaptation for the real-time video streaming. Nevertheless, the UDP datagram structure is heavily dependent on the transmitted data i.e., the

number of P-, B - frames vary. The I,P and B frames are followed by small size frames that change in size and number unpredictably. Therefore it is decided to simplify the simulation phase and consider only M-JPEG video streams.

As soon as a few streams are received, the switch stores bits of different packets separately. Having obtained the whole packet, the switch checks if it can send it at the same moment. If the check gives negative result, the packet is queued.

The simulation is based on creating a clock or sampling time. Every 1μs the switch ports check the number of bits passed to them from each of the stream. At the same time the buffer checks if packets are completely received and if the first packet in the queue is sent out. Thus every sampling time the buffer saves $1\mu s \cdot 100Mbit/s \cdot N$ bits and drains $1\mu s \cdot 100Mb/s$ bits, where N is the number of streams.

Subtracting the queue size from the buffer size decreases the remaining buffer space. When the difference is zero, the buffer stops accepting any incoming data. When this happens, parameters of the last packet saved into the memory are then displayed for the user, as shown in Figure 3.6. If the buffer is able to manage more data after the whole transmission is completed, the user obtains information that load does not cause any packet loss due to buffer overflow.

This simple simulation proved to work for different network and stream configurations, automatically adjusting switch actions without the user's interference.

The simulation does not consider complicated queuing theory and probability of losing packet before the switch. The focus was on how the switch behaves under the given simulation parameters, without processing packets for a checksum error or introducing jitter between streaming packets.

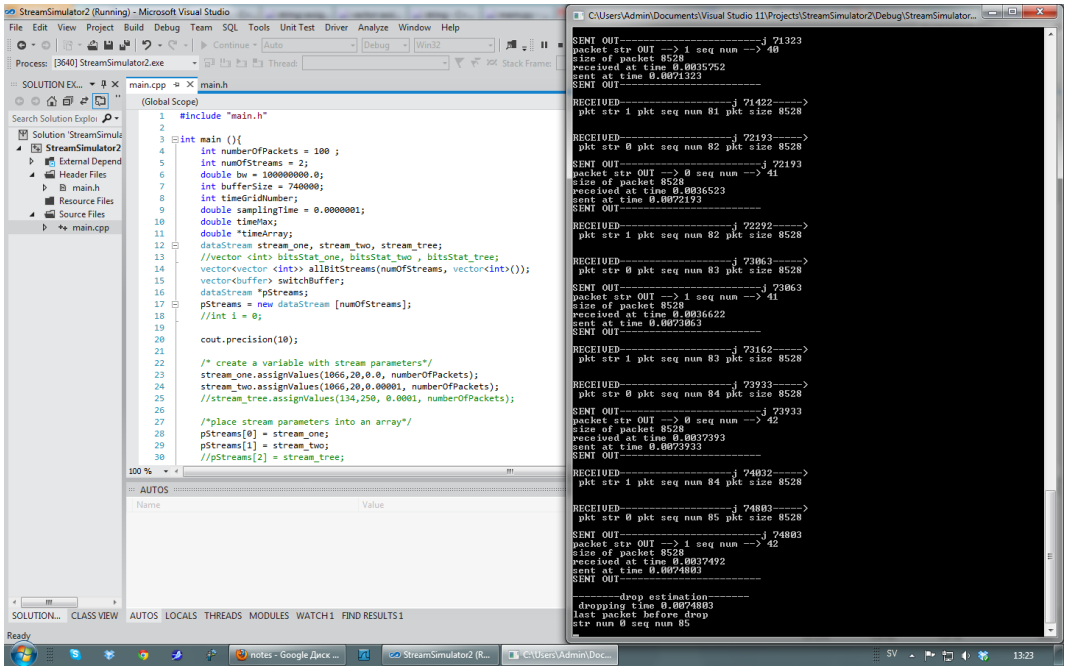


Figure 3.6: Result of the SwitchSimulator

4

Tests of the developed system

The last phase of the project aims to link all obtained test knowledge and developed systems to a real application. Due to lack of time all the manipulations are offline and manual.

4.1 Measurements

Three computers participate in a video conference. They are in a local network connected through a Westermo switch. The size of the switch memory was calculated in the earlier test with IAR cards and was already predefined as 770000 bit for these measurements, see section 3.3.3. Two computers broadcasted video with set video size and quality. The third participant received two video streams simultaneously, as shown in Figure 3.1. After the transmission began the CP-EP program captured the specified number of packets, required for analysis. When the program has completed the analysis, it saves the extracted parameters into a DataStreams.txt file in the same folder where the program is located. In accordance with received parameters local network and data streams can be created in simulations to test the network robustness in terms of packet loss. This was tested in the following four tests.

1. Confero statistics observations includes a monitoring statistics window that a program provides as one of the additional functions.

2. TA replicates the network topology from the information in DataStreams.txt file. This tool gives a prediction on packet delay and possible drop.

3. Two IAR cards create similar data flows as in the real time video conference. The flows is captured by the netANALYZER card that saves data into a .hea file. SA reads the saved .hea file and visualizes the flow as well as gives out information about packet loss and delay.

4. The switch simulation may be similar to test two: reproducing artificial flows and feeding the user with time and packet sequence numbers of the switch buffer overflow moment.

Every test uses its own method and algorithm of analysis. IAR cards are based on the real-time stream creation while SwitchSimulator and TA simulate and estimate virtual streams. In addition, TA requires considerable amount of time for processing data. While the switch simulator is only focused on one part of the whole network - the switch buffer. Therefore, some deviation in results are expected.

All four test supplied the user with information on: 1 - were there any packet drops with the given video size and quality, if yes - then when was the drop (time and packet sequence number); 2 - if no - then the network is robust enough to continue the transmission.

In the case of data loss, the user must lower the video parameters. The switch simulator provides a sequence number of the last packet that was received before packet loss. This sequence number corresponds to the number of frames in one UDP burst that leads to packet drops. A table of quality size and frame number can help to set the right Q parameter. The last procedure is repeated until no loss of packets is observed. Figure 4.1 gives a graphical overview of the testing system.

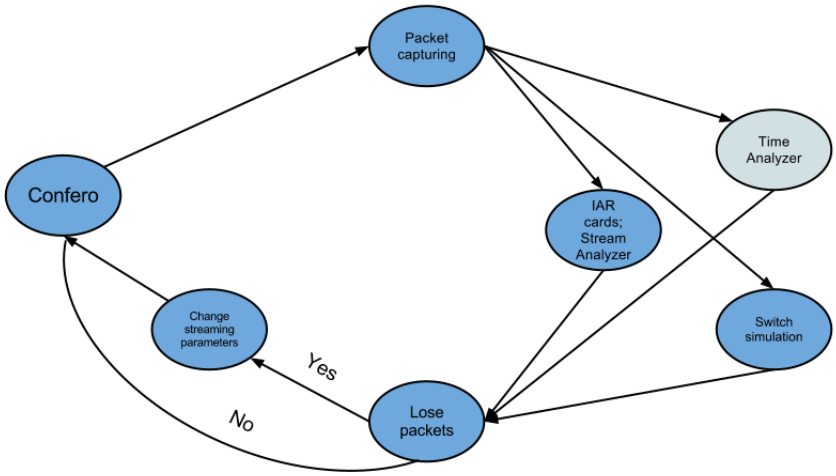


Figure 4.1: Scheme of testing the developed system

4.2 Video transmission and traffic capture

The network topology corresponds to the earlier described setup: two computers transmit video data. They are linked together with a Westermo switch to a third computer that operates as a receiver. Data flow directions and topology can be observed in Figure 3.1.

All three computers have Confero software installed but only one computer uses the CP-EP program.

Figure 4.2 shows the receiver side when the video conference is running between three computers. The video size and the quality of the video are set to the maximum level and test pictures were sent to reach stability in the network load.

When the transmission begins, the CP-EP program starts to run on one of the computers. It is done through the command window. As shown in Figure 4.3, the command line prints out the name of the network adapter that can be chosen for a capturing procedure. The chosen network adapter captures a predefined number of packets (10000 in this case).

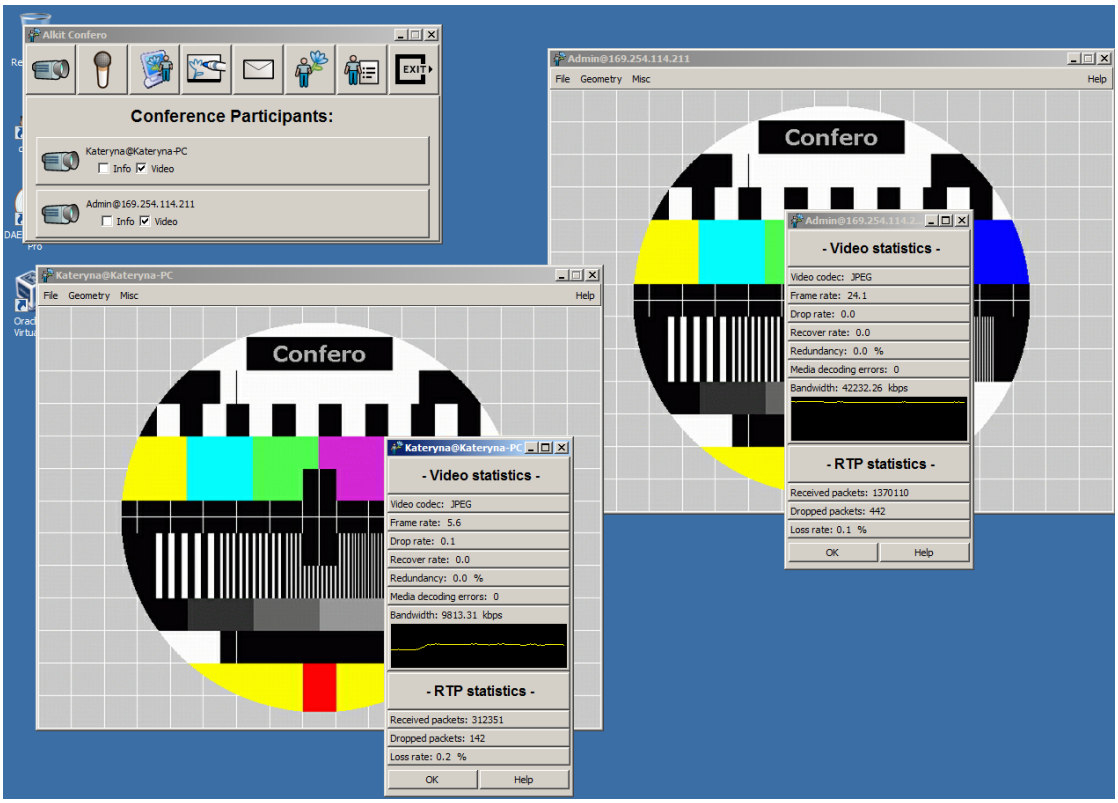


Figure 4.2: Confero video conference with maximum video size (receiver screen)

The program saves stream parameters into a .txt file that is found in the same directory as where the program is executed. In Figure 4.4, the information in the text file corresponds to two IP addresses and ports of the video senders (stream 0 and stream 1), two other streams are RTCP packets. The designation of each port can be found in the official table [27]. There, port 5566 is preassigned for UDP transmission and port 5567 is preassigned for Multicast Object Access Protocol (m-oap) transmission. Confero utilizes port 5567 to send RTCP statistics over the network. In the list of addresses and ports below, the user can also see what payload type (M-JPEG in this case), total size of the UDP datagram and time between UDP datagram bursts. The parameters above are sufficient to proceed with the next step - TA simulations.

The features and functionality of TA were described in section 2.3.1. After feeding in the stream parameters the following result can be observed in Figure 4.5. In the

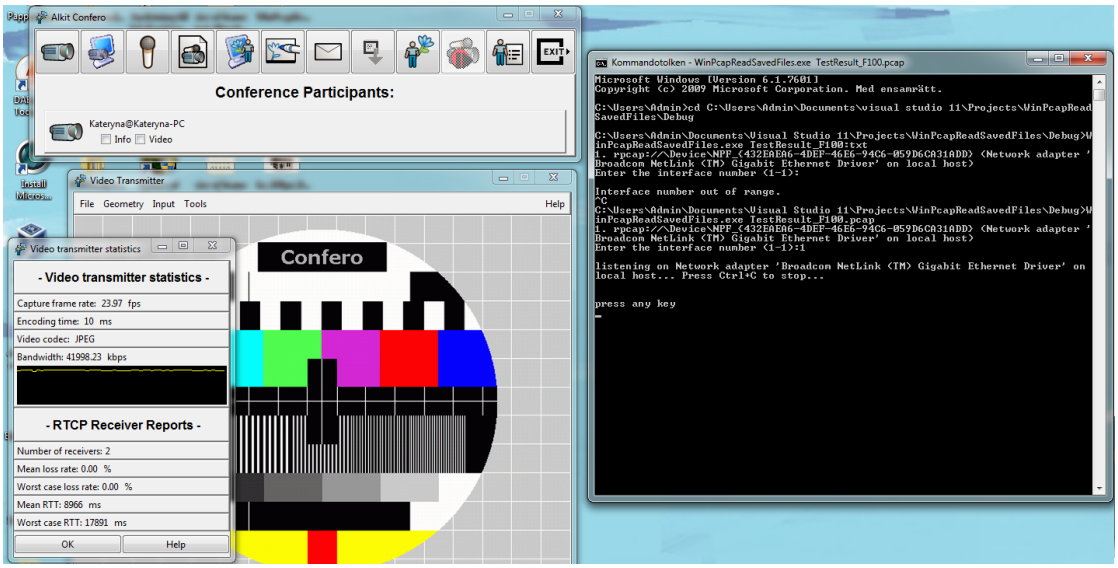


Figure 4.3: Packet capturing with CP-EP

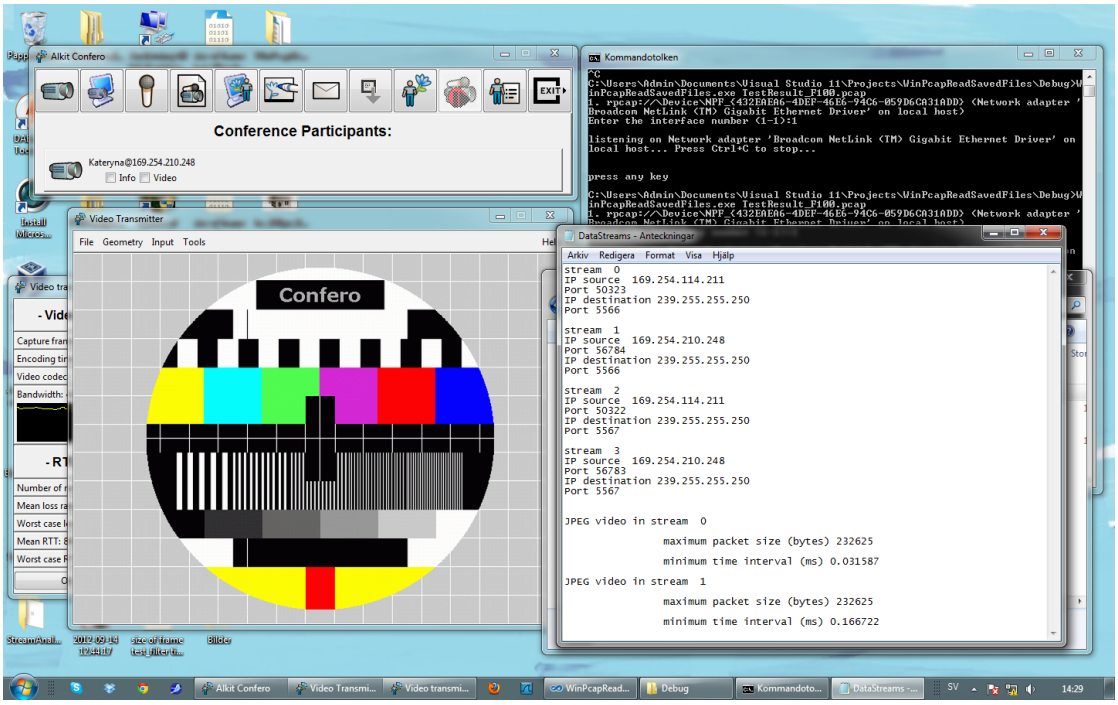


Figure 4.4: Obtained data streams' parameters

left most window, flows, all streams are visible; their priority, number of Ethernet frames in every UDP burst and the type of the transmission (broadcast). The

outcome of the simulation is saved to a.html file, the content of which is shown in the Figure 4.6.

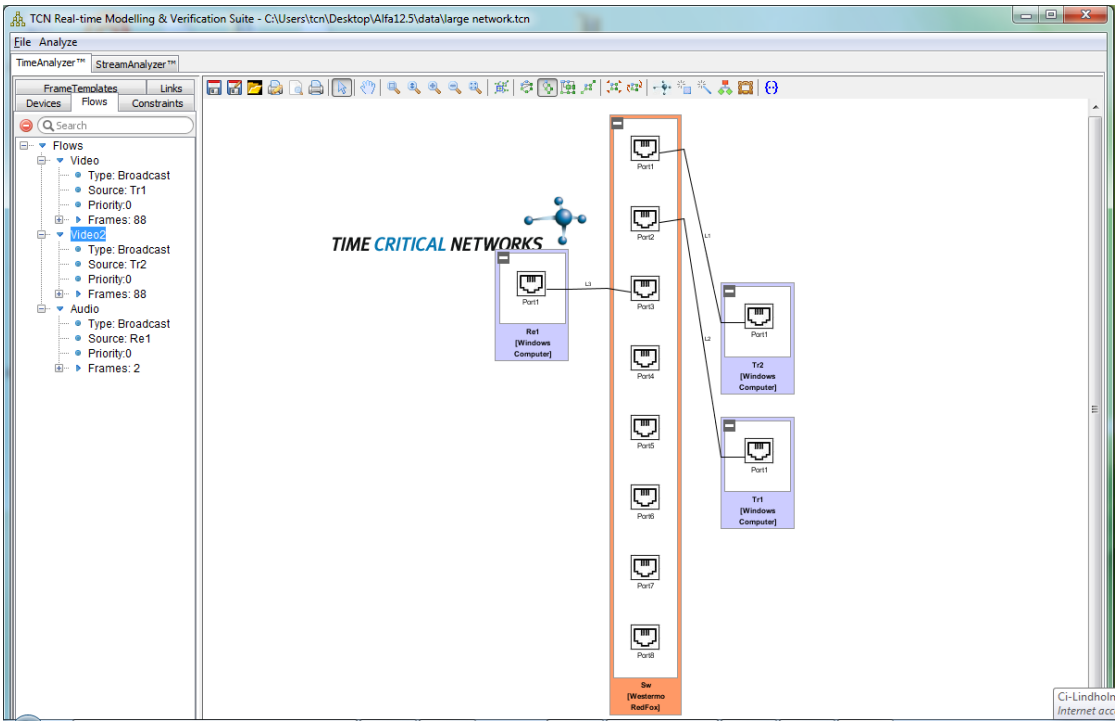


Figure 4.5: TA streams simulation

Memory Buffer Constraints

Switch	BC (bits)	WCBU (bits)
Sw	700000	727568

Figure 4.6: TA simulation results

It is now time to check if a real time simulation will give the same results as the calculations by TA. Two IAR cards produce two flows with size and period identical to a Confero video conference. In Figure 4.7 netANALYZER captures the traffic and the program displays the streaming data accordingly to the ports of a network card. The recreated stream is placed into a .hea file which is later read by SA. The

later tool provides visualization of the traffic activity in the simulated network, as seen in Figure 4.8. Due to the heavy load of the switch the forwarding time is increasing for every packet. Growth of the forwarding time will proceed until the switch will not be able to process both streams and thus one of the streams will be banned from sending more data. In Figure 4.9 it is seen that cards stop sending the green stream, thus at one point in time only blue stream transmits. In the real life system, packet loss would occur in both of the streams. Forwarding time envelope would have some fluctuations in contrary to the constant value after the switch overload.

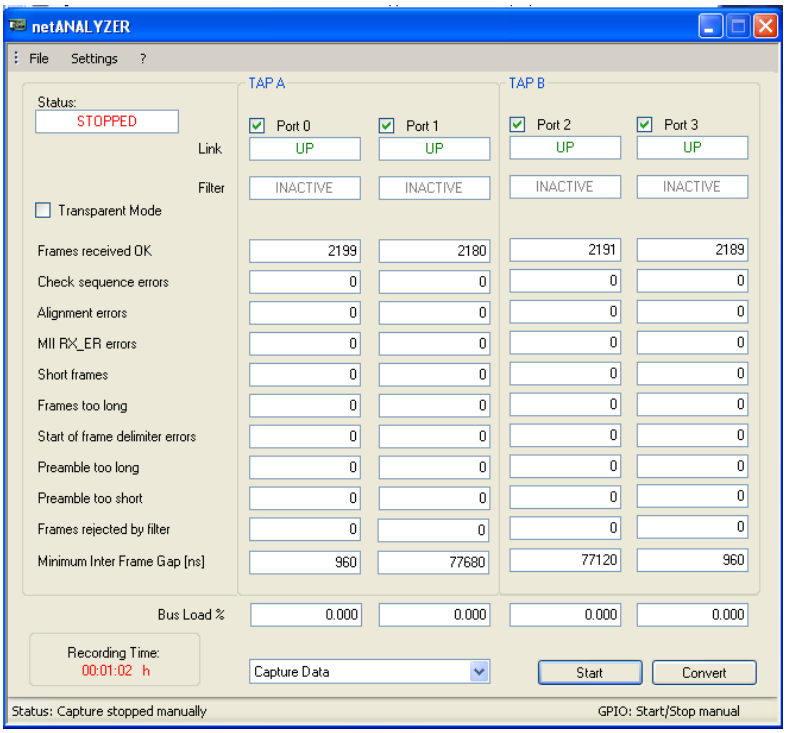


Figure 4.7: netANALYZER capturing packet in simulated traffic

The final step of the offline packet drop prediction mechanism is simulating the network load with given streams passing through a network switch. Following the same procedure as in TA - defining the numbers of streams and their size, simulation can then start. As can be seen from Figure 4.10, the last line in the

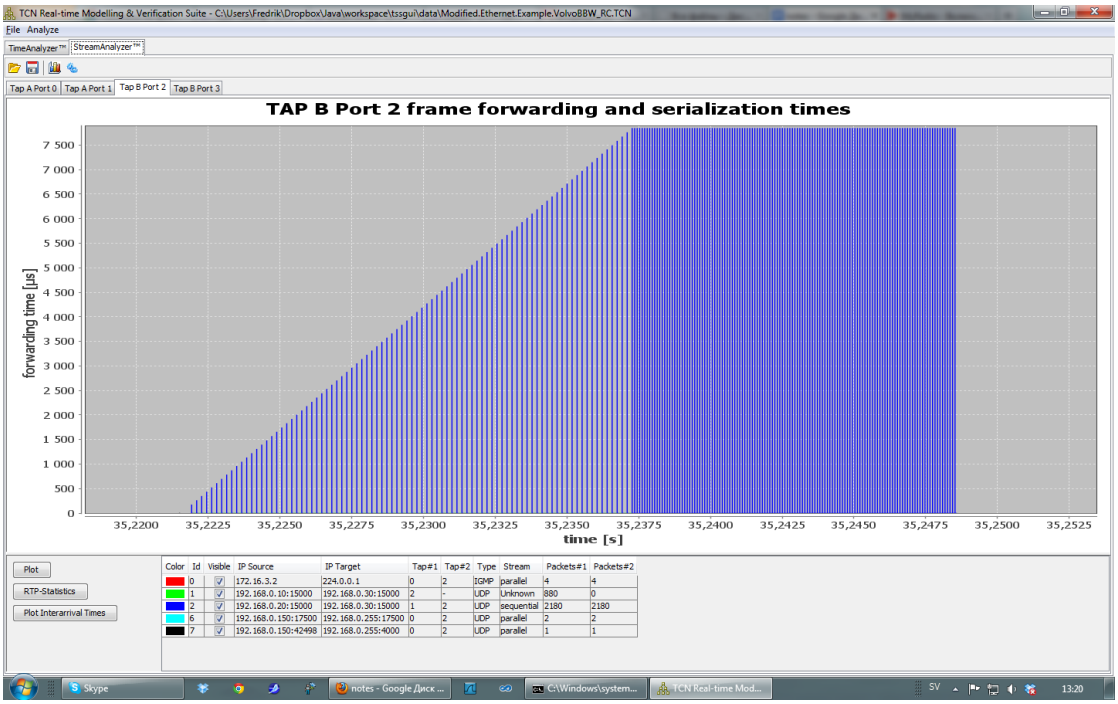


Figure 4.8: Forwarding time of the packets in one UDP datagram

command window shows that the last packet went in the switch before the danger of losing packet occurred.

Therefore it is safer for a user to choose a video Q value that would correspond to 88 frames or less. From Table 4.1, we can see that Q 95 satisfies the suggested amount of frames.

After a process of simulating, checking and confirming this prediction with four

Q value	90	91	92	93	94	95	96	97	98	99	100
Frames	59	62	64	70	75	83	97	110	157	190	218
Compression	16	15	14	13	12	11	9	8	6	5	4

Table 4.1: Number of frames in one UDP datagram in correspondence to Q value and compression

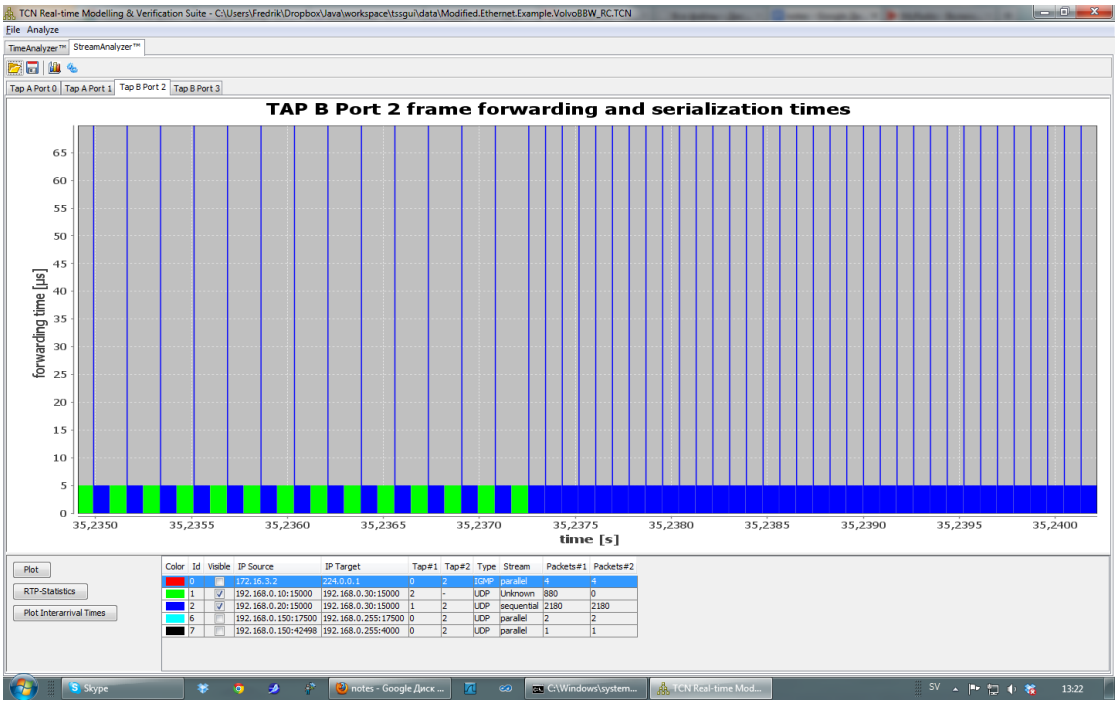


Figure 4.9: Enlarged visualization of two simulated by IAR cards video streams (blue and green), where a packet drop occurs in a green stream

different ways it was clear that the network is overloaded with video streams, and the receiver does not get all the sent information. Therefore the video quality must be reduced. According to the last simulations, the limit for this network would be a video at Q value of 95.

In accordance with this prediction the suggestion, tests and simulations are then tested with an updated Q parameter. All the steps are repeated which are not described in detail.

The video conference began with Q95, and Figure 4.11 indicates zero packet loss in the Confero statistic window. Meanwhile packets were captured, processed and saved. A decrease in amount of UDP burst can be observed in Figure 4.12.

The same real-time video flow corresponding to the quality of 95 replicated by IAR cards shows an increased forwarding time but no packet drop in transmission, see Figure 4.13. The linearity of the forwarding time is continuous and does not

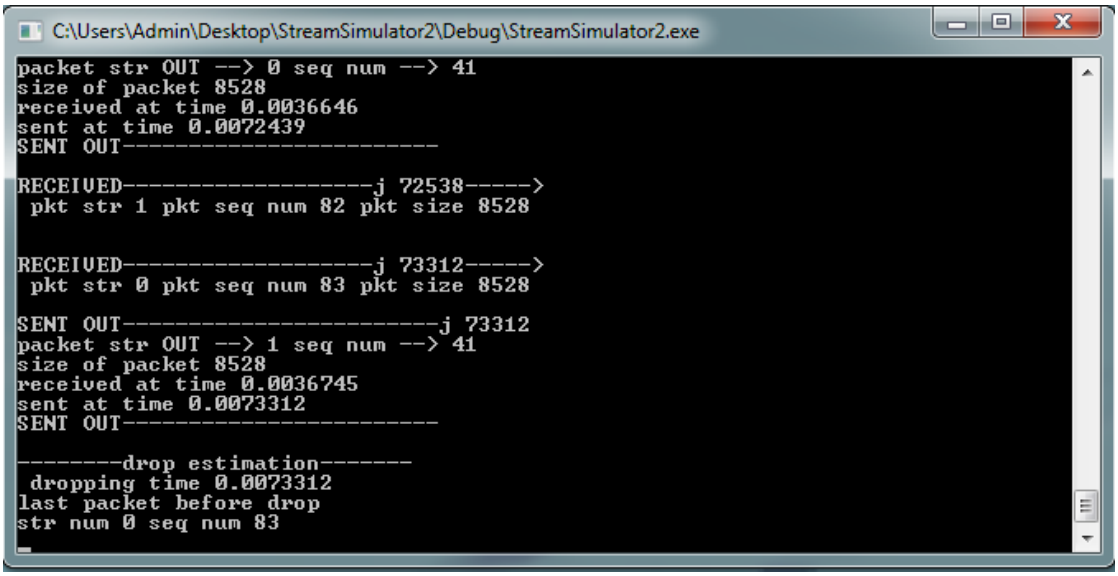


Figure 4.10: Result of SwitchSimulator estimation. "received at time means the time the packet arrives to the switch. Sent at time means tells at what time the packet left the switch

present any loss. Constantly increasing forwarding time would eventually cause packet drop if the quality was larger than 95 (if number of packets in the frame was bigger, as in the previous case). Enlarged view on one UDP datagram confirms (Figure 4.14) the prediction - all the cards were able to send out packets constantly without halting.

The switch simulations show no buffer overload. Thus, the queue still contains packets and the switch send them out after the transmission was finished, see Figure 4.15.

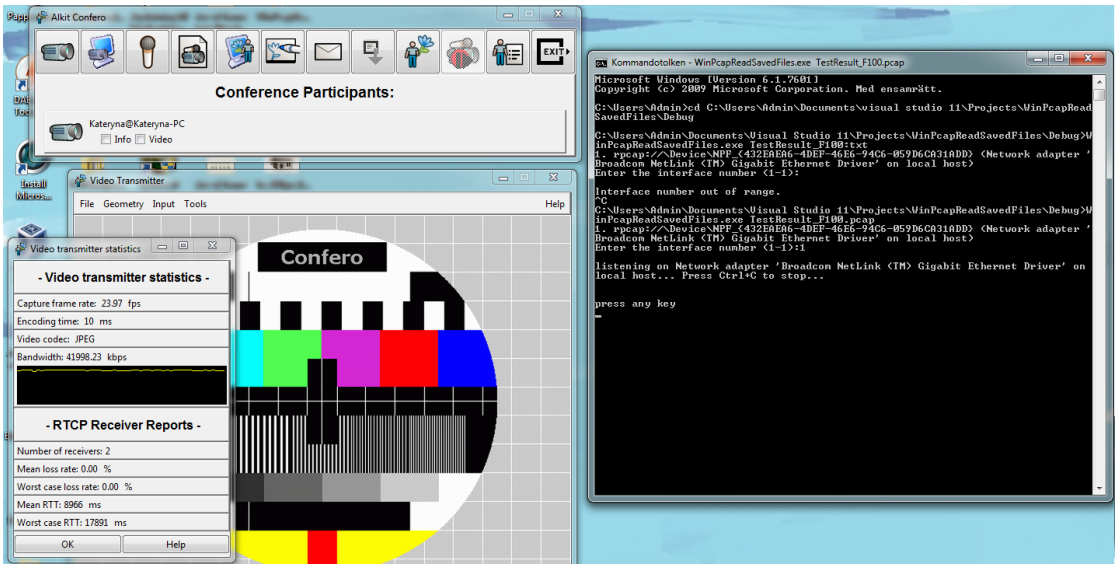


Figure 4.11: Confero video conference. Quality 95. Statistics and packet capturing.

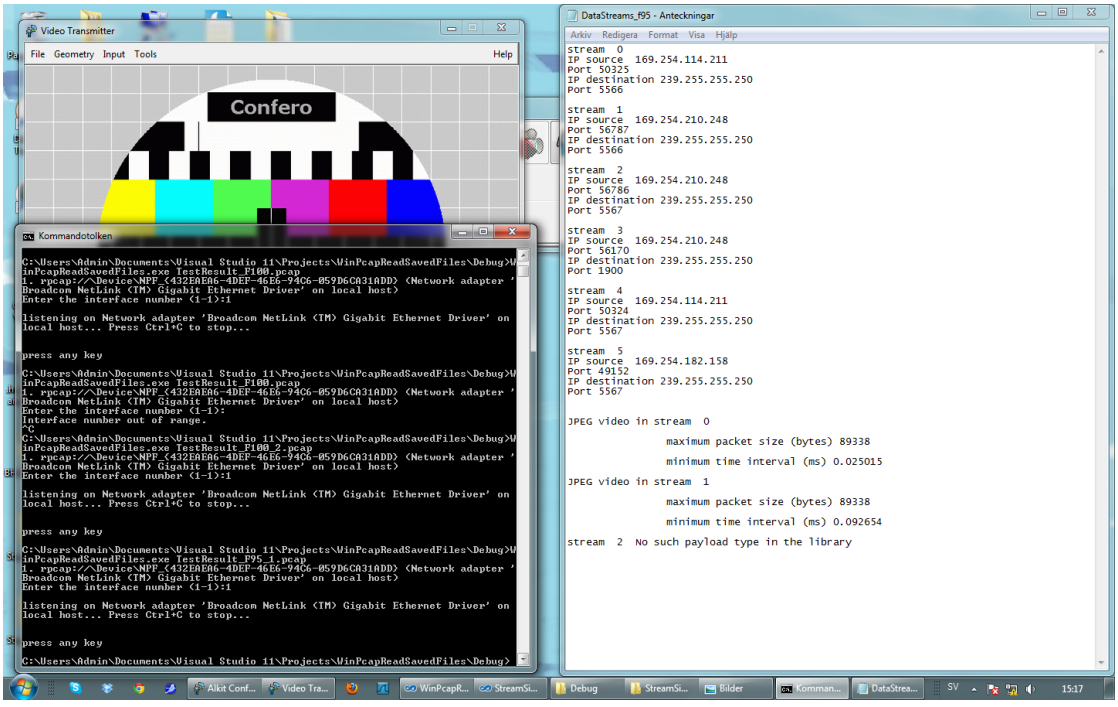


Figure 4.12: Capturing of the video stream with Q95

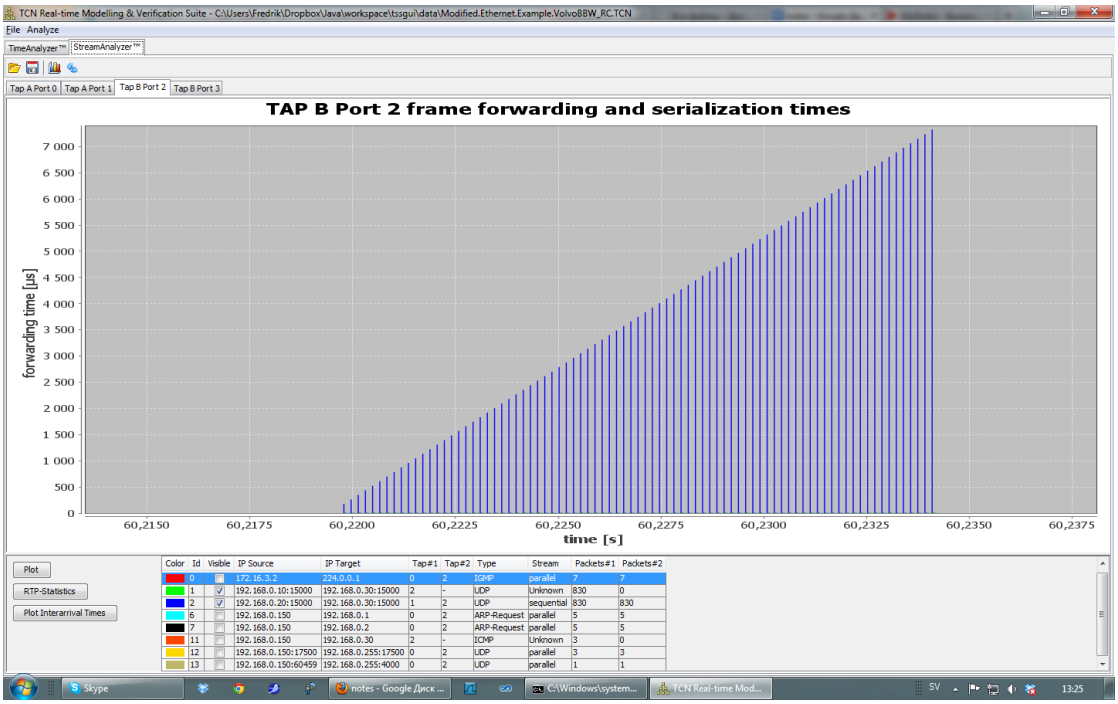


Figure 4.13: IAR card simulation of the video flow Q95

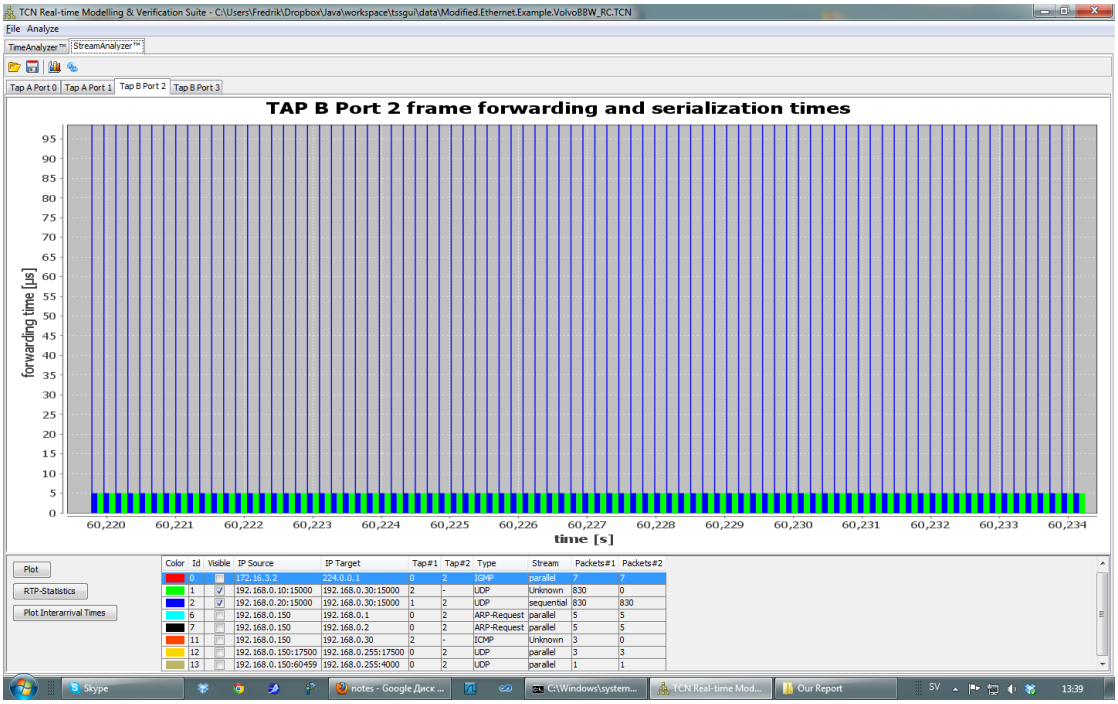


Figure 4.14: IAR card simulation of the video flow Q95. Enlarged

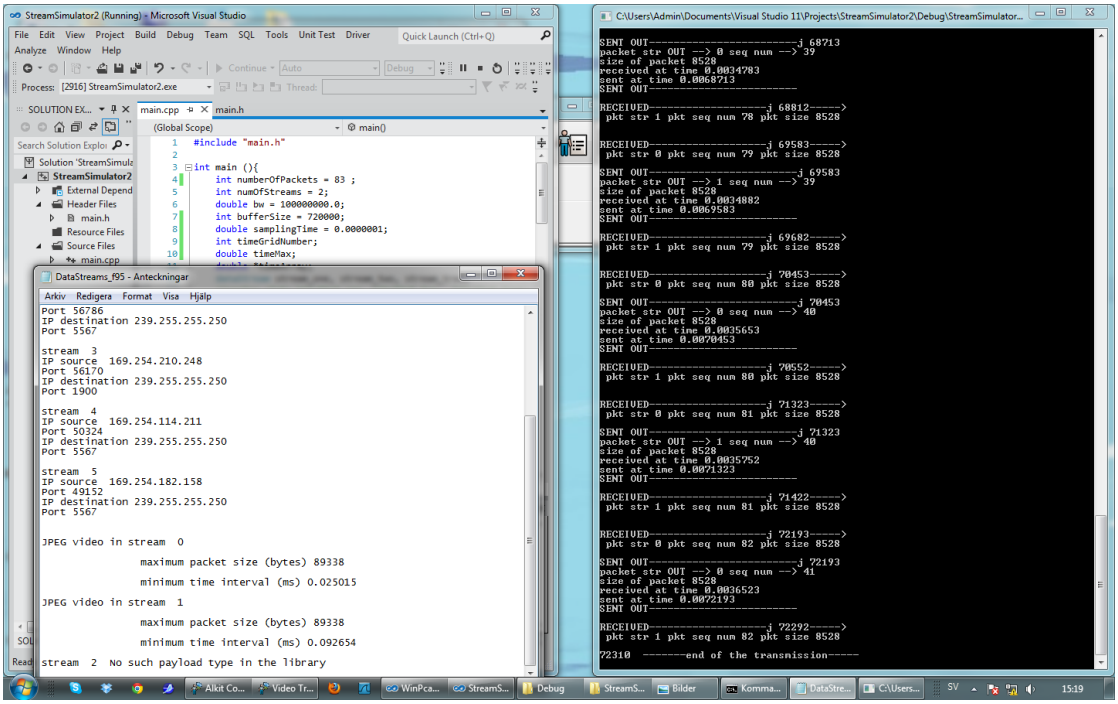


Figure 4.15: Switch simulation with parameters satisfying video flow of Q95

5

Conclusions and future work

This Chapter will present the conclusions reached in this project. It will also give suggestions on future work and improvement.

5.1 Conclusion

In this thesis, parts of a prototype system for dynamic network configuration has been developed, it can react to the network throughput variations and alter input parameters.

Such system was created and in addition it used a few other elements (switch and IAR cards simulations). Test with these components showed the possibility to predict network behaviour correctly.

Observations indicated deviation in results between the real-time streaming and the predictions made. In this work, network simulation does not take all the possible factors that affect a real-time streaming into account. One of the reasons is that throughput is constant or may not correspond to the maximum Ethernet speed that was assumed in the predictions. Furthermore a short test time doesu not give reliable results. Having done enough tests for statistical conclusions would provide an overview on the probability of receiving either of the outcomes. Considering

that, the output variations are in within the acceptable limits between theory and practice.

Tests with a real-time video conference showed that there exists packet loss at Q96, or between Q95 and Q96 for this test system. That corresponds to 83 and 97 Ethernet packets for one UDP datagram burst, respectively.

Network simulations showed that it is possible to have a loss-free transmission at Q95 and packet-loss transmission at Q96. The lucidity of the outcome was impaired by the lack of control over the video quality in the range between Q90 and Q100. Gaps in 5 to 28 packets between each quality value prevented us from carrying out precise observations of the video transmission performance. Therefore the IAR cards tests and switch simulations were helpful for predicting an accurate buffer size.

5.2 Future work

Incorporated software elements created a system of interactive units. Their joint work requires numerous offline manipulations by the user. The tool for video conferencing is easily manageable and controllable but developed and utilized components should be further improved and simplified.

TimeAnalyzer could not provide fast estimations of the network environment when it was loaded with video streams. That was caused by the complex computational algorithm of the program. Simplifications or replacement of this algorithm may solve the problem. Some improvement of the IAR cards can provide a possibility to simulate two streams of data from one card. That replicates two applications that can run one one computer.

The program that was developed for capturing network traffic has a large potential for improvement. For example all the captured traffic is saved on the hard drive and only after that all the computations and operations are done. That can be avoided by processing the captured data immediately as it was received. Winpcap

library contains functions that provide that option, thus they can be integrated into the system without the loss of its' functionality. In addition investigation of possible media formats will help to extend system's library of recognized codecs and protocols.

SwitchSimulator is a raw product where all the parameters are still manually typed in by the user(size of each frame, interframe interval, time delay between streams, switch memory and number of frames in one burst). Later development can minimize that step by associating those parameters with codex and switch used. The user will have to choose a type of a data stream from the library and the switch model.

One of the disadvantages of the simulator is inability to process more that one frame burst.Simulator also stops any data procession if the buffer of the switch is full. Those options were not needed in this project but it might be very useful to see a big picture of the simulated environment. They can be added as new functions to the program code. Additionally a separate function should take care of the speed drop in the incoming traffic.

It is expected to improve those disadvantages for a easy utilization of all the components. It is particularly anticipated to have all the parts working under online mode, automatically transferring parameters between each other and altering system adjustments according to estimations.

The improved system can find its implementation in the car industry, especially in in-vehicle communications. Intensive work has been done in the field of adapting video streams in car systems. An internal car network has a complex network topology, handling large amounts of data. Most data is sent from sensors and vital car elements indicating a current state. It is therefore crucial to provide a transmission without a single packet loss inside the network. Hence the project can be adjusted to be utilized in such a systems.

Bibliography

[1] CAN specification 2.0 (accesed: 2012-07-30).
URL www.can-cia.de/fileadmin/cia/specifications/CAN20A.pdf

[2] M. Liu, L.Brohne, J. Lext, Guaranteeing hard real-time requirements of in-vehicle multi-hop communication over ethernet, Tech. rep. (2012).
URL http://www.timecriticalnetworks.com

[3] A. Grzemba, MOST: the automotive multimedia network ; from MOST25 to MOST150, Electronics library, Franzis, 2011.
URL http://books.google.se/books?id=Wub-XwAACAAJ

[4] K. Matheus, Ethernet in cars: an idea whose time has come (2012-06-22 (2012)).
URL http://www.sae.org/mags/aei/11142

[5] J.F.Kurose, K.W.Ross, Computer networking: a top-down approach, Fifth Edition, Pearson, Boston, Mass, 2010.

[6] B.Andersson, J.Lext, Guaranteeing hard real-time communications over standard ethernet, Tech. rep. (2010).
URL http://dl.dropboxusercontent.com/u/5573907/Guaranteeing%20Hard%20Real-Time%20Com%20whitepaper%20v3.pdf

[7] TCN, Time critical networks (2012-07-28 (2012)).
URL www.timecriticalnetwork.com

[8] X.Song, W.Hui, J.Kuo, A practical bit stream organization algorithm for robust H.264/SVC transmission, Journal of Visual Communication and Image Representation 21 (8) (2010) 871–879.

[9] A.Rahbar, O.Yang, LGRR: A new packet scheduling algorithm for differentiated services packet-switched networks, Computer Communications 32 (2) (2009) 357 – 367.

[10] W.Stallings, Data and computer communications, Seventh Edition, Pearson Prentice Hall, Upper Saddle River, N.J, 2004.

[11] Info Cellar (2012-07-03).
URL `http://www.infocellar.com/networks/ethernet/frame.htm`

[12] Internet Protocol (2012-09-09 (1981)).
URL `http://tools.ietf.org/html/rfc791`

[13] J. Postel, User datagram protocol (2012-09-09 (1980)).
URL `http://tools.ietf.org/html/rfc768`

[14] L.Peterson, B.Davie, Computer Networks: A Systems Approach, Forth Edition, Morgan Kaufmann, 2007.

[15] H. Schulzrinne, S.Casner, R. Frederick, V. Jacobson, RTP: A transport protocol for real-time applications (2012-09-09 (2003)).
URL `http://tools.ietf.org/html/rfc3550`

[16] M. Christensen, Considerations for internet group management protocol (IGMP) and multicast listener discovery (MLD) snooping switches (2012-06-06 (2006)).
URL `http://tools.ietf.org/html/rfc4541`

[17] Queueing theory basics (2012).
URL `http://www.eventhelix.com/realtimemantra/congestioncontrol/queueing_theory.htm`

[18] D.E.Comer, R.Droms, Computer networks and internets: with internet applications, Prentice Hall, Upper Saddle River, N.J, 2001.

[19] G.Fairhurst, Ethernet (2012-07-20 (2009)).
URL www.erg.abdn.ac.uk/~gorry/eg3567/lan-pages/enet.html

[20] O.Berzin, Bandwidth, delay, throughput (2012-05-23 (2009)).
URL http://www.ccieflyer.com/pdf/2009-Mar-Oleg-Berzin.pdf

[21] M.Johansson, Multimedia communication, collaboration and conferencing using Alkit Confero, Tech. rep., Alkit Communication (2011).
URL http://confero.alkit.se/confero_whitepaper.pdf

[22] U.Lamping, Wireshark user's guide (2012-05-05 (2012)).
URL http://www.wireshark.org

[23] W. Team, Winpcap documentation (2012-06-01 (2009)).
URL http://www.winpcap.org/docs/docs_412/html/main.html

[24] D-Link, DES-1005D/ DSS-5+ 5-Port 10/100Mbps Switch (2011).
URL ftp://ftp.dlink.biz/des/des-1005d/documentation/DES-1005D_man_en_3-00_ALL_en_090817.pdf

[25] T.Lundquvist, J.Lext, Modelling the Westermo Redfox switch, 2009-12-18.
URL http://www.timecriticalnetworks.com

[26] Unix-compatible functions (2012-09-12 (2009)).
URL http://www.winpcap.org/docs/docs_412/html/group__wpcapfunc.html

[27] Real-time transport protocol (RTP) parameters (2012-06-10 (2012)).
URL http://www.iana.org

[28] Creating an application that uses wpcap.dll (2012-08-15 (2006)).
URL http://www.winpcap.org

A

Appendix

A .1 Development environment setup

Before any program was created several operations needed to be set to create an environment on the developing machine. The software used was Windows 7 as operating system and Microsoft Visual Studio 11 Beta as compiler.

Steps:

1. Download the latest version of WinPcap (developer's pack) from the website winpcap.org.
2. Launch Visual Studio. Create a new project in Visual Studio: File → New → Project.
3. In the New Project window: Installed → Templates → Visual C++ → General→ Empty project. Type in project, solution names and location in the bottom of the window. Press OK.
4. In the Solution Explorer right-click on the Solution Files. In the menu choose Add → New Item.
5. In the Add new item window: Visual C++ → C++ File(.cpp). Type in the name of the file (usually main.cpp). Press OK.
6. Extract downloaded WinPcap into Solution directory of above created project.
7. Right-click on the project in Solution Explorer, choose Properties.
8. In Property Pages window:
   (a) Go to Configuration Properties → C/C++ → General. In Additional Include Directories add path to ..../WdpPack/Include.
   (b) Go to Configuration Properties → C/C++ → Preprocessor. In Preprocessor Definitons add WIN32;WPCAP;HAVE_ REMOTE.

(c) Go to Configuration Properties → Linker → General. In Additional Library Directories add path to ..../WdpPack/Lib.

(d) Go to Configuration Properties → Linker → Input. In Additional Dependencies add winpcap.lib;Packet.lib. If there are other libraries included already, there is no need in removing them. Press OK.

9. Open main.cpp file. Type in # include < *pcap.h* >. The compiler is now ready to work with WinPcap [28].