Jan Hagqvist

LAGGINGMETER

The Importance of QoS in Network Gaming

Bachelor's Thesis
TI15STIVE

2019

Kaakkois-Suomen
ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Author (authors)	Degree	Time
Jan Hagqvist	Bachelor of Information Technology	December 2019

Thesis title

71 pages

LaggingMeter
The importance of QoS in network gaming

Commissioned by

Xamk ICTLAB

Supervisor

Martti Kettunen

Abstract

The purpose of this thesis was to develop a system that allows the generation of individual interference or lag in the network connection of gamers. This LaggingMeter system also provides each gamer a button to press when they experience lag in their gameplay. The first research question was how to implement such a system using a modern PC system. The second research question was how modern and efficient such a PC system would need to be. The research was made as a design-based research.

The work is based on a project idea by principal lecturer Martti Kettunen. In the beginning quality of service theory was studied and it was established that there are four important network parameters that affect it. These parameters are delay, delay hopping or jitter, packet loss and bandwidth. These were chosen as the parameters that could be individually changed for each gamer in the system that was implemented.

The basis of the LaggingMeter system is the Source Router software which is run in a bridging computer that is placed between the server and client computers. The bridging computer has two network interfaces, one on the server side and one on the client side. Source Router captures all the network frames coming in from each of the interfaces, identifies the used player computers and interferes the connection of each client computer individually.

The user interface of the Source Router software is web-based and utilizes a WebSocket connection. The operator can set different interference for each of the client computers and monitor the system status using the user interface. Furthermore, a separate software was created to collect the lag button presses from the users. This software is run in a Raspberry Pi computer and it sends the button presses to the Source Router software for storage and further analysis.

As a result of the thesis a system was created that allows interference generation with a software-only approach and can be used on a regular PC system that has two network adapters available. The system was tested on four different PC systems which all ran the system properly. Thus, the minimum level of PC system required was not established.

Keywords

network, gaming, interference, quality of service, lag

Tekijä/Tekijät	Tutkinto	Aika
Jan Haggvist	Insinööri (AMK)	Joulukuu 2019
Opinnäytetyön nimi		71 sivua
LaggingMeter QoS:n merkitys verkkopeleissä		
Toimeksiantaja		
Xamk ICTLAB		
Ohjaaja		
Martti Kettunen		

Tiivistelmä

Tämän opinnäytetyön tarkoituksena oli kehittää LaggingMeter eli järjestelmä, jolla verkkopelaajien verkkoyhteyteen voidaan kehittää yksilöllisiä häiriöitä ja kerätä pelaajilta palaute heidän kokemistaan häiriöistä jokaiselle pelaajalle erikseen tarjottavan painikkeen avulla. Työssä tutkittiin, miten edellä mainittu järjestelmä voidaan toteuttaa modernilla PC-laitteistolla. Lisäksi pyrittiin selvittämään minkä tasoinen PC-laitteisto on riittävä järjestelmän käyttämiseksi. Opinnäytetyö tehtiin kehittämistutkimuksena.

Työ pohjautuu yliopettaja Martti Kettusen projekti-ideaan. Työn alkuvaiheessa perehdyttiin palvelunlaatuteoriaan ja todettiin, että neljä tärkeää verkkoparametria, jotka vaikuttavat palvelunlaatuun ovat viive, viiveen vaihtelu, pakettien häviäminen ja kaistanleveys. Nämä valittiin parametreiksi toteutettavaan järjestelmään, joita muuttamalla käyttäjien yhteyttä voidaan häiritä.

LaggingMeter toteutettiin luomalla Source Router -ohjelma, jota ajetaan palvelimen ja pelaajakoneiden väliin asetettavassa siltauskoneessa. Siltauskoneessa on kaksi verkkoliitäntää, joista toinen kytketään palvelimen puolelle ja toinen pelaajakoneiden puolelle. Source Router kaappaa jokaisen verkkoliitäntöihin tulevan verkkokehyksen, tunnistaa niistä eri pelaajakoneet ja häiritsee jokaisen pelaajakoneen yhteyttä erikseen järjestelmän käyttäjän asettamalla tavalla.

Source Router -ohjelman käyttöliittymä on toteutettu web-pohjaisena käyttäen WebSocket-yhteyttä tiedon välittämiseen. Käyttöliittymän avulla järjestelmän käyttäjä voi asettaa eri häiriöitä eri pelaajakoneille ja tarkkailla järjestelmän tilaa. Lisäksi luotiin erillinen häiriönilmaisupainikkeiden painallukset keräävä ohjelmisto, jota ajetaan Raspberry Pi -pienoistietokoneessa. Tämä ohjelma lähettää painikkeiden painallukset Source Router -ohjelmalle keskitettyä taltiointia varten.

Tuloksena saatiin järjestelmä, jossa häiriöiden luominen tapahtuu täysin ohjelmallisesti ja jonka käyttämiseen riittää tavallinen PC-laitteisto kahdella verkkokortilla. Järjestelmä testattiin neljällä eri tehoisella ja eri ikäisellä PC-laitteistolla ja todettiin, että se toimii käytetyllä 100 Mbps:n nopeudella oikein kaikilla testatuilla PC-laitteistoilla. Tämän vuoksi tarvittavan PC-laitteiston minimivaatimuksia ei kyetty selvittämään.

Asiasanat

tietoverkko, pelaaminen, häiriöt, palvelunlaatu, lägi

CONTENTS

ABBREVIATIONS

ADSL	Asymmetric Digital Subscriber Line
AF	Assured Forwarding
ARP	Address Resolution Protocol
BER	Bit Error Rate
BTC	Bulk Transport Capacity
CoS	Class of Service
CRC	Cyclic Redundancy Check
DiffServ	Differentiated Services
ECMP	Equal Cost Multipath
EF	Expedited Forwarding
FEC	Forward Error Correction
FPS	First Person Shooter
FRR	Fast Reroute
FTP	File Transfer Protocol
IGP	Interior Gateway Protocol
IntServ	Integrated Services
IP	Internet Protocol
IPFRR	Internet Protocol Fast Reroute
ITU	International Telecommunications Union
LAN	Local Area Network
LFSR	Linear Feedback Shift Register
MAC	Media Access Control
MPLS	Multiprotocol Label Switching
OS	Operating System
QoE	Quality of Experience
QoS	Quality of Service
PHB	Per-Hop Behavior
RSVP	Resource Reservation Protocol
SLA	Service Level Agreement
TCP	Transport Control Protocol
TE	Traffic Engineering
UDP	User Datagram Protocol
UI	User Interface
USB	Universal Serial Bus
VoIP	Voice over Internet
V-Sync	Vertical Synchronization
XOR	Exclusive-OR

1 INTRODUCTION

In a world were networking is becoming ever more important the quality of network connections is emphasized. This is especially true in real time communications where even a slightest delay might ruin the user experience. Traditional real time applications include video and voice calls. One important real time application in present-day world is video gaming.

The evolution of video gaming is setting the requirements the network must possess to new highs especially regarding to the fault tolerance and quality of the network connection. The failure of the network to meet these requirements will subject players to an unwanted phenomenon known to players as lag.

The main causes of lag are network delays, the loss of network packets and limited bandwidth. To a gamer who is playing a real time game these will manifest themselves as various issues in the game play such as the player character not moving as expected or the player not being able to pick up items. This thesis will introduce a system to generate deliberate lag for research and quality control purposes by manipulating the parameters mentioned above.

The first part of the thesis introduces Quality of Service theory which is closely tied to network lag. Quality of Service is a way for the network to try to minimize the effects of lag. Several studies have been carried out about network lag. The theory section also includes a review of a few studies.

The second part of the thesis will introduce a system that can be used to intentionally create lag on a network game connection. This part also includes system tests on the lag creation system that will verify the functionality of the system components, as well as a limited small-scale test of the system functionality in a real-world network gaming scenario.

1.1 Aims and method

The commissioner of the thesis is the ICTLAB of the South-Eastern Finland University of Applied Sciences in Kotka, Finland. The institution is also known by the abbreviation XAMK and was formed in 2017 by the merging of two South-Eastern Finland Universities of Applied Sciences. At the Kotka campus

XAMK has two ICT degree programmes, one for game programming and one for networking which is transitioning into network cybersecurity.

The purpose of this thesis is to design and implement a system to cause deliberate lag into the connection between a player and a game server, a so called LaggingMeter. The LaggingMeter system was first proposed by principal lecturer Mr. Martti Kettunen of XAMK ICTLAB in 2012. The system contains a component that will generate intentional lag to a player connection and will also provide each player a button to press when they feel the connection is affected by lag. The work will be limited to implementing the system and will not include gameplay research with the system.

As there is a game programming degree programme at XAMK the LaggingMeter can be used to study the networking aspects of the games the game programming students will create. LaggingMeter can provide important information for the game creator on the quality of the network code of a game. This information can be used to reduce possible lag by testing several different versions of network code with the LaggingMeter system and selecting one that is most resistant to lag. Furthermore, the LaggingMeter can be utilized to test existing games made by game companies by studying what amount of lag is enough for the player to detect it thus helping in quality control.

The implementation of the LaggingMeter will be based on modern PC hardware. There are three main components in the LaggingMeter system. The first component requires a per-user routing of network packets or frames. The second component will gather the button presses of the users. The third component will collect the data from the two previous components to enable data analysis.

The research questions for this thesis are:

- What is required to implement the LaggingMeter system on a modern PC system?
- How modern and efficient the PC hardware must be for the system to work properly?

The LaggingMeter is designed to affect four network parameters: bandwidth, latency, packet loss and jitter. These parameters are the same that network Quality of Service (QoS) functionality is designed to work with. Both are working on the same parameters albeit in opposite directions. Therefore, QoS theory will be covered in the theory section of the thesis.

When starting a new research one of the first things a researcher must decide on is the research method. Sometimes the method can be more freely chosen by the researcher while sometimes the nature of the research subject will define the method. Qualitative and quantitative methods are the most common main classes of research. Whether a qualitative or a quantitative method is used will be dictated by the nature of the phenomenon. (Kananen 2014, 51.)

Both research methods also have sub methods, or the researcher can combine both methods in different portions in the research. Qualitative research, however, forms the basis of all research as it has helped to create the understanding of real-life phenomena. (Kananen 2014, 52.) The research method chosen for this thesis is design-based research which is a research method developed by Finnish researchers (Kananen 2014, 58).

Design-based research is a combination of both qualitative and quantitative methods and targets a change in the research subject which is usually a product, method or organization. There has been wide debate whether design-based research is scientific. Many companies carry out design changes in their products constantly, but this is not design based research. What is required to make such a work a scientific research is the inclusion of a theory section and a scientific take on the matters concerned. (Kananen 2017, 18.)

2 QOS THEORY

QoS is a way to describe and standardize the overall performance of a network in terms of assurance and level of service. It requires a way for the network to prioritize traffic depending on its urgency and guarantees a certain level of performance for the traffic flow. These guarantees are important in real-time applications. (Misra & Goswami 2017, 92.)

The need for QoS stems from the fact that the resources of any given network are limited. In the presence of network congestion this leads into a situation where the network traffic needs to be prioritized by certain quality of service methods. Therefore, network congestion leads into delay and loss of network packets. Data applications can usually deal with delay, but it is detrimental to certain applications such as Voice over Internet (VoIP) where the Internet is used for voice communication. (Saarelainen 2011, 222.)

Other applications where QoS is required are video transmission and industrial control. Delay in robot arm control, for example, might lead it into crashing into something. Delays and losses do not affect non-real time applications that much as they can retransmit the data, but for real-time applications this is not a viable solution. They need the data to be delivered on time. This is where QoS steps in. (Peterson & Davie 2012, 530.)

2.1 QoS service levels

There are several service levels or schemes of QoS available such as best-effort service, differentiated service and guaranteed service. Each network application differs in their service and performance requirements and a suitable service should be selected based on the application's needs. (Vegesna 2001, 6.) For a network to be able to support QoS it needs to have the capability to treat different packets differently (Misra & Goswami 2017, 92).

The service levels have mechanisms on both Layer 2 as well as Layer 3 that need to be enabled in the network devices to effectively enable that service level for the network. The Layer 2 mechanisms target the various link layer technologies whereas Layer 3 mechanisms target the network layer which is Internet Protocol (IP). (Vegesna 2001, 7.)

Service providers offer a certain level of service to the customer specified in a service level agreement (SLA). The SLA uses several different metrics to describe the service level provided. The requirements of the desired applications will be the basis of the SLA requirements. As the usage and importance of IP

based applications has risen the SLAs for IP performance have to be better defined and have tighter requirements. (Evans & Filsfils 2007, 2.)

2.1.1 Best-effort service

The best-effort service level is the basic level that does not guarantee when a packet is delivered to the destination or if it will reach the destination at all. However, dropping a packet usually only occurs when the router input or output buffers would overflow. As no guarantees for service or delivery are made for packet forwarding in best-effort service the service cannot be thought as a part of QoS. Best-effort service has previously been the only service offered by the Internet. (Vegesna 2001, 6.)

Applications like FTP work properly with best-effort scheme, but do not have the best performance. To work well, all applications should receive guaranteed resource allocation such as bandwidth, low delay and minimal packet loss. (Vegesna 2001, 7.)

When best-effort service is used to transmit VoIP or video which are usually transmitted over UDP, one solution to reduce packet loss would be to switch the stream to TCP connection instead as TCP makes guarantees to deliver the data to its destination error-free whereas UDP gives no such guarantees (Kurose & Ross 2013, 639).

This would, however, introduce retransmissions to the stream when a packet fails to reach its destination on the first try. And retransmissions on a real-time application are usually not acceptable because of the increased end-to-end delay that lead to degradation of the audio and video. Instead the encoding algorithms try to conceal the loss of information by encoding the data so that the disturbance will be minimal for the viewer. Forward error correction (FEC) can be also used where the transmitted data includes redundant information to help recover the lost data. (Kurose & Ross 2013, 639.)

2.1.2 Differentiated service

Differentiated service (DiffServ) groups traffic into several classes based on their service requirements. The term Class of Service (CoS) is often used to

refer to this type of QoS scheme where each traffic class has its own QoS mechanisms. Differentiated service does not give certain guarantees of service, but only differentiates traffic by class and gives certain classes the privilege over others. Thus, it is sometimes also called as *soft QoS*. (Vegesna 2001, 7.)

All traffic that enters the Diffserv domain needs to be classified to ensure the per-class SLAs are met. This classification includes giving the traffic one of the four defined Per-Hop Behaviors (PHB) that define how the traffic should flow within the network at each hop. The PHBs are Expedited Forwarding (EF), Assured Forwarding (AF), class selector PHB and default PHB. (Evans & Filsfils 2007, 149–159.)

Expedited Forwarding tells the router that the packet should be routed with minimal delay and loss. The router can only guarantee this if the arrival rate of the packets is limited to the maximum speed of the router. A 100 Mbps router is incapable of ensuring this for a 1 Gbps traffic stream. (Peterson & Davie 2012, 550–551.) This behavior is required in traffic such as VoIP (Evans & Filsfils 2007, 155).

Assured Forwarding is used in data applications where a certain assured bandwidth is required. There are four AF subclasses defined each with three different levels of drop probability. The drop probability indicates the importance of the packet and means that packets with higher drop probability can be dropped in favor of more important packets. (Evans & Filsfils 2007, 156).

The reason for **class selector PHB** is to provide backwards compatibility with the IP precedence field. In practice the IP precedence field has not been used consistently and there has been little reason to deploy this type of PHB. **Default PHB** is the behavior that should be used for all packets not specifically marked by any other PHB. It has no committed resources bound to it. (Evans & Filsfils 2007, 158–159).

For bandwidth-intensive applications the scheme of differentiated service works well. To ensure basic network connectivity at all times it is important to

differentiate and prioritize the network control traffic from the data traffic into its own high priority class. (Vegesna 2001, 7.)

2.1.3 Guaranteed service

This service level introduces the concept of network resource reservation where each application is required to reserve the network resources it needs in advance. This ensures that the network can meet the requirements of the traffic flow. The scheme requires rigid guarantees from the network, and it is therefore sometimes called as *hard QoS*. (Vegesna 2001, 7.)

Reserving a path for a single traffic flow does not scale well over the Internet backbone which services a great number of flows at any given time. However, using aggregate reservations is a scalable way to offer this service as they call for only a minimum state of information in the Internet routers. (Vegesna 2001, 7.)

Integrated Services (IntServ) is one attempt to provide such a guaranteed service. The goal of IntServ is to provide per-flow QoS guarantees for network traffic. IntServ defines several new classes of service and lets applications to choose a suitable class for their own use. An important component of IntServ is the Resource Reservation Protocol (RSVP). (Jha & Hassan 2002, 107–108.)

The IntServ framework classifies applications into three categories: elastic applications, tolerant real-time applications and intolerant real-time applications. Elastic applications have no stringent QoS requirements and can operate with any data rate or delay. Examples of this type of applications are web browsing, telnet connections and file transfer. (Jha & Hassan 2002, 108.)

Real time applications such as audio conferences and video streaming have very strict time bounds, they need to operate in meaning timeliness is paramount. Small packet loss, however, can be tolerated depending on the used coding method or codec. Therefore, audio and video streaming are considered tolerant real-time applications. (Jha & Hassan 2002, 109.)

Demand for even more stringent QoS from the network comes from the intolerant real-time applications. They have very precise bandwidth, delay and jitter requirements. If the requirements are not met their performance will degrade severely. An example of such application is VoIP which requires very low end-to-end delay. (Jha & Hassan 2002, 109.)

For IntServ to work the applications need to communicate their needs for network resources to the network elements along the path to the destination. For this purpose, a signaling protocol is required. The RSVP has been built for this purpose and it carries the requests for resource reservation and QoS specifications along the network as well as signals resource availability. (Jha & Hassan 2002, 133.)

The original driver for RSVP design were multicasting applications. Many streaming applications have more receivers than senders and therefore multicast, where one sender sends to many receivers, is used to deliver the streams. RSVP also utilizes the existing robust connectionless model of the Internet and does not mandate any new routing protocols. (Jha & Hassan 2002, 134.)

RSVP works so that first the sender sends a path message to the receiver. When the message arrives at the receiver it then generates a reservation request that it sends back the same path. When an intermediate router receives the request, it checks whether the needed resources can be allocated and if not, it sends a reservation error message back to the receiver meaning the requested level of QoS cannot be reached. (Misra & Goswami 2017, 102-103.)

If the requested resources are available, the router sends the reservation message backwards to the next router and so forth until the reservation request is received by the sender. This notifies the sender that the resources are available, and it can begin sending the actual data. (Misra & Goswami 2017, 102-103.)

Applications that require this level of service are typically audio and video streams where even the slightest delay or packet loss would cause glitches in

the actual stream. For VoIP applications a maximum latency of 100 milliseconds is required whereas Internet telephony also has an additional bandwidth requirement of 8 kbps. This level of latency is usually acceptable for other multimedia applications as well. To guarantee these service requirements resource allocation is required from the network. (Vegesna 2001, 7.)

2.2 QoS performance metrics

Four metrics are commonly used in QoS to characterize the performance of a connection. These metrics are delay, delay jitter, packet loss and bandwidth or throughput. (Vegesna 2001, 9.) Sometimes delay is also called latency and it is defined as the time a packet requires to travel from the source to the destination. A variation in this delay between consecutive packets is called jitter. (Barreiros & Lundqvist 2016, 8.)

In addition to these it can be argued that per flow sequence preservation and service availability are important aspects in QoS as well. All the metrics above are used to define the basis of the SLA where the SLA is the contractual commitment to provide certain level of quality or QoS for the client network connection. (Evans & Filsfils 2007, 2.)

2.2.1 Network delay

The network delay requirements for time critical applications such as video and VoIP are usually defined as one-way delay whereas the requirements for adaptive applications such as those that use the Transport Control Protocol (TCP) are defined as the round-trip delay (Evans & Filsfils 2007, 4). For VoIP applications the maximum tolerable one-way delay is 150–200 ms. If the delay is any longer the participants will speak on top of each other. (Saarelainen 2011, 221.)

One-way delay is the time difference between the time an IP packet enters the network and comes out of the network. The round-trip delay is comprised of the one-way delay times of a packet going towards the destination and the reply packet returning from the destination excluding the end system processing delays. The network delay can be divided into four separate parts regardless of whether a one-way delay or a round trip is considered. These parts are the

propagation, switching, scheduling and serialization delays. (Evans & Filsfils 2007, 4.)

For real-time applications delay plays an important role as the packets are only relevant to the destination if they arrive within a time period they are expected. If the packets arrive late, they become useless. And not only that but they also become a burden for the destination as they still need to be handled which increases load and delays the handling of the consecutive packets. (Barreiros & Lundqvist 2016, 9.)

Propagation delay is the amount of time a single bit needs to reach the destination router from the source router. It is usually affected by the distance the bit needs to travel as well as the physical media used. When calculating the total propagation delay of a long route the individual links it consists of can be summed together. (Evans & Filsfils 2007, 5.)

It should be noted that network links never go directly from A to B so calculating the route length is not easy. But the International Telecommunications Union (ITU) has a recommendation that can used to estimate the route lengths from the direct distances. For distances less than 1000 km the route length estimate is 1.5 times the direct distance. Distances between 1000 km and 1200 km are estimated as 1500 km route length. And for distances above 1200 km the route length is estimated as 1.25 times the distance. (Evans & Filsfils 2007, 5.)

The propagation delay through a coaxial cable is roughly 4 ms per 1000 km and through optical fiber around 5 ms per 1000km when repeaters are used. Therefore, it is easy to see that on a long route the propagation delay is a great contributor to the network delay. To reduce the propagation delay, the network topology could be changed by adding a new direct link towards the destination or the existing route could be optimized by rerouting it via a shorter path. (Evans & Filsfils 2007, 5–6.)

Switching delay is made up of the time the packet spends inside a router or a switch. The time begins when the packet comes to an incoming router interface and ends when the packet is scheduled in the outbound interface. In

high-performance routers such as backbone routers the switching time can be considered negligible as they typically implement the switching in hardware. (Evans & Filsfils 2007, 6.)

The typical switching delay in such a router is in the order of 10–20 µs, but even in a software-based router the switching delay is typically only 2–3 ms. There is not much that can be done to change the switching delays other than replacing the hardware or software. Fortunately, the switching delays are usually not a great concern when it comes to end-to-end delays, other delay types typically make the most of the total delay. (Evans & Filsfils 2007, 6.)

Scheduling or queueing delay carries on from the previous switching delay in that it is the time difference between the scheduling of the packet to the outbound interface and the start of clocking the packet out of the router to the outbound link. Both the scheduling algorithm and the scheduler queue affect the time as they are the function of the traffic load and queue capacity. To control the scheduling delays traffic load management and scheduling mechanism control is required. (Evans & Filsfils 2007, 6.)

Serialization delay is the time that is required to clock the packet onto a link. Both the size of the packet and the link speed affect the time. Longer packets increase the time and higher link speed decreases the time. When the link speed is very high as in backbone links the serialization delay can be generally considered negligible. For low speed links the serialization delay is a more significant component of network delay. As the serialization delay is a physical constraint it can only be controlled by changing the link speed. (Evans & Filsfils 2007, 6–7.)

2.2.2 Delay jitter

As packets travel through the network the delay is not constant which leads to jitter in the delay. The term jitter in general means variation of a certain parameter. In this case the parameter is the network delay. Generally, delay jitter is considered the variation of the one-way delay of two consecutive packets though it could also indicate the variation of average or minimum delay. The

cause of the jitter are changes in the components of the network delay which were explained in the previous section. (Evans & Filsfils 2007, 8.)

The way jitter is problematic is that it makes the delays of arriving packets inconsistent. This means that the receiving application needs to continuously adapt to new delay values which can be undesirable especially in real-time applications. (Barreiros & Lundqvist 2016, 9.)

When the network topology changes for example due to a failed link it causes a peak in the jitter due to changed propagation delay. Scheduling delay variation is caused when the scheduling queue of a network device changes between empty and full leading to a change in the wait time before the packet can be sent. (Evans & Filsfils 2007, 8.)

Switching delay is usually relatively static especially in hardware-based routers but can vary in software-based routers as some packets may require more processing than others due to packet length or type. Serialization delay stays constant when a link of the same speed is used. However, if due to a topology change the route changes to a link with a different speed the serialization delay will change and contribute to the delay jitter. (Evans & Filsfils 2007, 8.)

The increase of bandwidth in networks has also meant that new more complex queueing strategies are required to counter jitter. Therefore, when transmission speed has been increased new queueing systems have been developed which are more efficient in handling different kinds of traffic. (White & Banks 2018, 23.)

Jitter does not affect much in some applications such as those using TCP protocol. Applications such as video and audio that use streaming are more susceptible to jitter. (Evans & Filsfils 2007, 8.) Thus, they will use de-jitter buffers to overcome the effects of the jitter. The buffers help remove the effects of the delay variation and make the delays seem constant in the end-systems. (Saarelainen 2011, 222.)

2.2.3 Packet loss

Dropped packets on a certain link are considered as packet loss. When a sent packet does not arrive into its destination during a defined period it is considered lost. Packet loss is typically measured as one-way loss because the route between two systems may be asymmetrical meaning that packets going one way may travel a different route than the returning packets. Measuring the loss of each path on its own can be used to estimate the total round-trip loss. (Evans & Filsfils 2007, 9.)

For VoIP applications packet loss means degradation of sound quality and the level of degradation is dependent on the used packing algorithm also known as codec. A packet loss of 1% is still generally acceptable. This means, however, that as much as 99% of the packets still need to be delivered and delivered in time. (Saarelainen 2011, 222.)

The loss rate alone is not a key factor in some applications when performance is considered. Also, the loss distribution needs to be considered. This can lead to a situation that even though the loss rate would be the same the application performance is different due to a different loss distribution. An example of this would be when many consecutive packets are lost versus a lost packet every few packets. (Evans & Filsfils 2007, 9.)

Loss distributions are described by the loss period and the loss distance. The loss period defines the length of the loss and the distance defines the spacing between each loss period. Packet loss is dependent on many factors such as congestion, lower layer errors, network element failures and losses in the end-system applications. (Evans & Filsfils 2007, 9–11.)

Congestion is caused by queues building up leading to dropped packets. To control the losses the traffic load needs to be managed by applying appropriate queuing and scheduling mechanisms. (Evans & Filsfils 2007, 9.) To avoid congestion a simple solution would be to increase the bandwidth enough so that the congestion would disappear. The network could deliver all traffic and would have no need to prioritize one traffic type over another. (White & Banks 2018, 197.)

Sometimes this overprovisioning indeed can be the best solution and a way to ensure QoS assurances are met. Usually this, however, is not possible due to certain traffic patterns. For example, some applications will use all the bandwidth that is available inevitably creating a congestion point somewhere in the network. Others may transmit micro-bursts, short bursts of large amount of data that will consume a lot of bandwidth, but only for a short time. Also, there are protocols like TCP that will flood the network with a lot of data creating congestion to find out the best rate for data transmission. (White & Banks 2018, 198.)

Lower layer errors are typically caused by physical layer bit errors. The cause of these can be noise or attenuation in the transmission channel causing packets to be dropped. To counter these bit errors many link layer technologies and IP transport protocols such as the User Datagram Protocol (UDP) implement cyclic redundancy check (CRC) algorithm or at least a parity checksum. Frames with changed bits and thus an invalid checksum will be dropped. (Evans & Filsfils 2007, 10.)

Therefore, in networks that implement these types of technologies simple bit errors will cause packet loss as packets either arrive correct or not at all. The bit error rates (BER) vary by the used layer 1 or layer 2 technology where fiber optic links are the most reliable and satellite links the most unreliable. (Evans & Filsfils 2007, 10.)

Network element failures are caused by broken network infrastructure such as broken devices. The failures can result in dropped packets until a new route is found around the broken network element if such is available. The period of the loss depends on the used network technologies. In a non-MPLS environment the packet loss will continue until the interior gateway protocol (IGP) converges even if there is an alternative path around the failure. If no alternative path exists, the packet loss carries on for as long as the failure is repaired. The time for IGP convergence should be less than a few hundred milliseconds in a well-designed network. (Evans & Filsfils 2007, 10–11.)

In networks where there is an alternative path, available technologies such as MPLS Traffic Engineering (TE) - Fast Reroute (FRR) or IP Fast Reroute (IP-FRR) can be applied to significantly reduce the time of the failure. This allows quick restoration of the failed link with a typical restoration time of 50 ms or less. (Evans & Filsfils 2007, 11.)

Losses in end-system applications are usually due to receiving buffer overflows or underflows. When the buffer is full and a new packet arrives, an overflow occurs. Underflows are typically the problem in streaming real-time applications such as VoIP or video where the receiving codec needs data, but none is available due to an empty buffer. With careful design of both the application and the network the overflows and underflows can be prevented. (Evans & Filsfils 2007, 11.)

Methods to overcome packet loss include error correction, error concealment, redundant transmission and retransmission. The method that is chosen is dependent on the transport protocol and end application. (Evans & Filsfils 2007, 11.)

2.2.4 Bandwidth and throughput

When the service provider makes an SLA for the customer, they commonly define the bandwidth or the layer 2 access link capacity that the customer is offered. Bandwidth is usually an ambiguous term that can have multiple meanings when link, network or service capacity is considered. (Evans & Filsfils 2007, 12.)

The **link capacity** is measured by the number of bits the link can transport in a second. Both layer 2 and layer 3 need to be considered when defining this value. The layer 2 capacity is defined by the used physical media i.e. layer 1 and usually remains constant, however, some media such as ADSL can have adaptive bit rate. (Evans & Filsfils 2007, 12.)

On layer 3 the link capacity depends on the layer 2 capacity, layer 3 packet sizes and the encapsulation the layer 2 requires transporting layer 3 traffic.

Therefore, the link capacity of IP layer 3 traffic can be derived from the capacity of the underlying layer 2 technology. Sometimes link capacity is referred to as the link bandwidth or the link speed. (Evans & Filsfils 2007, 12.)

When QoS is used to define several classes of traffic that are aggregated together as a traffic stream the traffic stream can be broken into separate classes. All the classes can have their own defined minimum bandwidth which is referred to as the **class capacity** or the class bandwidth. The minimum link bandwidth of a certain path between two network points is defined as the **path capacity** or path bandwidth. (Evans & Filsfils 2007, 12–13.)

Bulk Transport Capacity (BTC) or **throughput** denotes the data throughput the user can attain between a source and a destination. BTC is measured as the long-term average data throughput rate that a transport layer connection can achieve. The used transport layer technology should be adapting its rate of sending to try to maximize the throughput. TCP sessions are an example of such adaptive transport layer connections. (Evans & Filsfils 2007, 13.)

The maximum throughput is limited by the available link capacity but can be significantly lower than the link capacity as it can be impacted by factors such as packet loss and round-trip time. Therefore, it should be noted that the attained throughput may not be the same as the contract defined bandwidth. (Evans & Filsfils 2007, 13.)

2.2.5 Per flow sequence preservation

There is no guarantee in IP that the delivered packets arrive in the same order they were sent. When the sequence number of an arriving packet is less than that of the previous packet the packet is considered to have arrived out of order or re-ordered. This re-ordering may have an adverse impact on some applications and thus any network should be designed to try to prevent packet re-ordering within a traffic flow. (Evans & Filsfils 2007, 18–19.)

To prevent packet re-ordering two key design best practices do exist. The first is to make sure that when IP load balancing is used it is performed on a per flow level and not on a per packet level so that packets that belong to the

same flow will follow the same path. The Equal Cost Multipath (ECMP) algorithms perform this kind of load balancing by using the source and destination IP addresses as well as the source and destination ports and the used protocol as a 5-tuple to identify packets that belong to the same flow and routes them via the same path. (Evans & Filsfils 2007, 19.)

The second-best practice is to make the QoS design such that the scheduling algorithm will always service packets from the same flow in the same order and from the same queue. This principle is fundamental to both the Integrated Services (IntServ) and Differentiated Services (DiffServ) architectures that are used with QoS. All in all, re-ordering packets is a bad practice and should be avoided. (Evans & Filsfils 2007, 19.)

2.2.6 Availability

Availability for IP services can be considered as network availability that only considers the availability of the network or as service availability that also considers the provided service. Furthermore, availability can be thought of as unidirectional or bidirectional where bidirectional applies to most IP applications as many applications need the destination to send a response to the source for proper functionality. (Evans & Filsfils 2007, 20.)

Network availability means the fraction of time that the network is available for use between two certain points in the network. Things that affect the network availability include planned and unplanned outages. An unplanned outage is the result of a broken network component such as a switch or a router or the actual physical infrastructure that can for example be a cable or a fiber. The availability of each individual network component can be used to estimate the network availability. Usually the availability of the network is not enough, however, as many applications have more stringent requirements in the form of service availability. An application such as VoIP might have network connectivity, but there may be such a long delay between packets that the speech of the calling parties might be unintelligible. (Evans & Filsfils 2007, 20–21.)

Service availability is more than just network availability as it means the fraction of time that a certain service is available between two network points with

SLA metrics such as delay, jitter and packet loss considered. Two ways to define service availability exist where the first definition takes network availability into account and service availability only applies when certain network availability is reached. The second defines service availability independent of network availability leading to the fact that service availability cannot exceed network availability as without a network there cannot be a service. Furthermore, service availability is not only affected by the network availability, but also the application performance. (Evans & Filsfils 2007, 21.)

2.3 Quality of Experience

The performance metrics described above play an important part in defining the characteristics of the network. However, also other metrics do exist that try to quantify the performance that the user experiences using a certain network application. The term Quality of Experience (QoE) is used for this user experience. (Evans & Filsfils 2007, 22.)

For VoIP and video applications the used metrics are not the basic ones described in the previous chapter, but rather the quality of the used encoder and decoder and the quality of service of the whole network connection. Together these metrics form the QoE of the application and there usually is some QoE target defined for the application that it tries to meet. (Evans & Filsfils 2007, 22.) One modern application that uses video and audio streaming is online games hosted by a service provider where the game is run at the service provider servers and only the game video and audio is transferred to the end user.

The QoE metrics can be either subjective or objective. Subjective metrics depend on the user perception of the QoS that is given as a feedback by the user. Objective metrics on the other hand use measurements of the transmitted stream characteristics and try to infer the subjective quality that the user is expected to have. (Evans & Filsfils 2007, 22.)

3 STUDIES ON NETWORK LAG

3.1 The perceived effect of network lag

Network lag has been the subject of several studies and three studies have been selected for presentation here. The first is an Internet survey where the researchers of Academia Sinica in Taipei have asked players how they perceive lag, what they think the cause for lag is and how they react to lag (Tseng et al. 2011, 1).

The phenomenon of lag is described to be when the game does not update the screen or respond to the user commands in a timely fashion. The cause for these is usually delay in the network or long processing delay in the user's computer. The lag is caused by the non-QoS guaranteed architecture of the Internet. (Tseng et al. 2011, 1.)

The study found that 22.7% of players encounter lag frequently and a half or 50.7% encounter lag occasionally. 21% of the players consider the lag being serious and 41.9% of the players consider it moderate. Most or one third of the lag is found to be happening intermittently and lasting for a few seconds. 21.2% of the players attribute the lag being due to the access link bandwidth of the game servers and 21% think it is caused by the server equipment itself. 19.2% think their own computer is the cause. (Tseng et al. 2011, 4.)

When players encounter lag, 64.6% suffer through it and 17.9% ignore it completely. 55.9% of the players who leave a game because of lag will try to reconnect back to the game after a few minutes. 76% of the players will let their friends know they are experiencing lag and 42.7% find moderately or strongly that their friends experience lag at the same time. 45% feel that the effect of the lag on their gameplay is moderate or strong, 34.1% feel it is weak and 21% feel it has no effect on their gameplay. (Tseng et al. 2011, 5.)

When asked whether the players report the lag, 34.9% say they will keep silent and not report the lag in any way. 31.5% of the players will go to the Internet forums and complain there. Only 16.4% report their findings to the game company and 15.8% to their ISP. When players were asked how they try to solve the lag problem, 23.6% said they will increase their network bandwidth

and 23% said they will upgrade their computers. 21.9% of the players will try to detect the cause of the lag by the usage of network tools. A total of 86% of the players claim that having lag during gameplay is really annoying. (Tseng et al. 2011, 5.)

3.2 The effect of tick rate

Researchers of the Hong Kong Polytechnic University have researched the effect of tick rate and vertical synchronization (V-Sync) on a networked multiplayer FPS game. The game they used was Counter-Strike: Global Offensive. The study found that increasing the used tick rate will significantly improve player performance. It also found that V-Sync has no impact on player's performance. (Lee & Chang 2015, 1.)

Tick rate dictates how often the game can process information coming in from the player computers, update the state of the game and send the updated information back to the clients. Each game has their own tick rate depending on the game's genre and complexity. An FPS game with a low tick rate has many issues such as incorrect damage registration. (Lee & Chang 2015, 1.)

When the computer renders an image of the game world it will happen at a dynamic rate that varies depending on the complexity of the current view and the capacity of the display adapter. Most monitors on the other hand have a fixed refresh rate which can cause screen tearing or the display of different game frames on the screen at the same time. (Lee & Chang 2015, 1.)

A solution for this is to use V-Sync that will synchronize the display of the game frames by the monitor. If the computer has the next screen frame prepared too early, it will wait for the monitor to finish the current screen draw cycle. And if the frame rendering by the computer takes too much time the monitor will display the last frame again waiting for a new frame from the computer which introduces input lag. (Lee & Chang 2015, 1.)

The study presented six different scenarios where the ping, tick rate and V-Sync settings were different. The used tick rates where 64 and 128, meaning 64 and 128 game updates per second respectively. Four different players took

part in the study and each player played under each of the six scenarios. The study found that when a higher tick rate of 128 was used the mean accuracy of the players improved by 4% due to improved damage registrations by the game. The study did not find any significant difference in player accuracy when V-Sync was on or off. (Lee & Chang 2015, 3.)

3.3 Jitter modeling

Oklahoma State University researchers have studied network jitter in their paper. They state that packet jitter may be the result of packets traveling different routes to the destination but is primarily caused by the varying queuing delays in the network equipment. The researchers examine the characteristics of delay jitter, develop a model for it and confirm the model by comparing it to empirically collected jitter data. (Daniel et al. 2003, 1.)

Previous studies suggest that the probability density function of the network jitter is Laplacian. Furthermore, when a packet is delayed by a great amount a phenomenon of delay spikes or the arrival of multiple packets nearly simultaneously can be detected. The spikes are caused by the first packet having a large delay and the packets following it being held in a buffer and then all being sent directly after each other in a quick burst. (Daniel et al. 2003, 2.)

For their jitter model the researchers end up in a three-component model that uses Laplacian distribution for small time scale delays and white Gaussian noise for large time scale delays added with delay spikes generated by an exponential decaying function (Daniel et al. 2003, 2).

For model verification the researchers first collect real world network packet statistics using an application called NetPerf that was developed for that use. Then they use the collected network statistical data to generate simulated jitter delay values using their model and compare them to the real-world data. (Daniel et al. 2003, 2.)

In their tests the real-world jitter data had a mean of 45 ms and a standard deviation of 5.59. The model generated jitter data also had a mean of 45 ms and a standard deviation of 5.5 being very close to the standard deviation in the

real-world data. Both data also had similar bursts caused by the delay spikes. The generated jitter model can be used for example to test adaptive jitter buffering algorithms. (Daniel et al. 2003, 4–5.)

4 LAGGINGMETER DESIGN

In his project idea document Mr. Kettunen suggests that there are two main sources of network gaming lag. The first one being the overload of the game server and the other being the player's connection which is not working well enough for the game. Mr. Kettunen also mentions that the QoS of the broadband services players use, namely DSL, cable modems and 3G/4G-modems, is not ideal. The main problems in the above services are delay, lost packets, shortage of bandwidth and short blackouts. (Kettunen 2012, 1.)

Mr. Kettunen proposes the creation of a so called LaggingMeter that could be used to measure the sensitivity of different game programs for the underlying network connection. The disruption of the game network flow is experienced by players as lag and the feeling of lag is a very person-dependent where one person might feel the lag is very disruptive and another person would barely notice it at all. (Kettunen 2012, 1.)

The LaggingMeter would be a black box between the game server and the player. A test group would be made to play a game and lagging would be measured as a statistical quantity over a long period. The black box would be able to generate delay, packet loss, shortages and low bandwidth situations. Each player would have their own button box with a button to press when they experience lag. A diagram of the proposed system can be seen in figure 1. (Kettunen 2012, 1.)

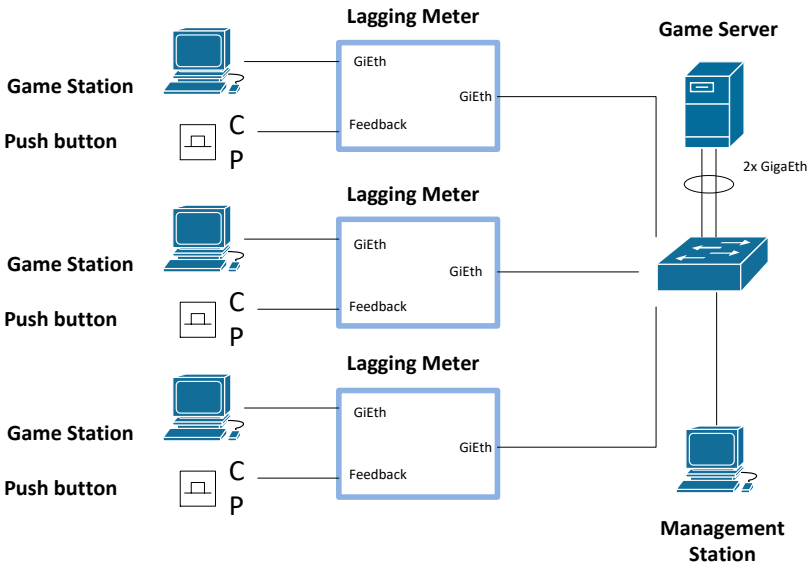

Figure 1. LaggingMeter system (Kettunen 2012, 3)

Mr. Kettunen gives an example of the LaggingMeter operation. When operational, the LaggingMeter would start with no lag in the player's connection. Then over time the quality parameters would be randomly adjusted so that the quality goes worse each time. The test group playing the game will not be aware of the network connection quality. Players would be told to press the lag button whenever they feel there is lag in their connection. The quality data and button press data would be collected for analysis. As a result of the analysis the average level of interference each quality parameter tolerates would be revealed. (Kettunen 2012, 3.)

Sometimes test players can be impartial or objective in pushing the lag button. Therefore, there would need to be times when the network connection would not be interfered at all. If a test player would still press the lag button multiple times when there is no lag present, the player's results should be excluded. As possible methods to implement a LaggingMeter, Mr. Kettunen suggests either the use of programmable logic or a regular PC system with 2 separate one gigabit Ethernet ports. (Kettunen 2012, 4.)

5 SYSTEM IMPLEMENTATION

For implementing the LaggingMeter system that was discussed in the previous chapter it was decided to use a regular PC system to act like a bridge between the client and server networks. Furthermore, it was decided to build two

separate programs, one for generating the actual interference and one for collecting the button press data.

The interference generator would be named Source Router and would run on a regular PC system running Fedora 30 Linux operating system. It would have a user interface (UI) that would work via a web browser. The button press collector on the other hand would run on a Raspberry Pi 3B miniature computer with the default Raspbian Linux operating system and would send the button data to the Source Router program for storage and further analysis. A logical name for this Raspberry Pi program would be Button Press Collector.

5.1 Source Router software

The main operating principle of Source Router software is relatively simple. The PC running the software has two dedicated network adapters, one for the client side and one for the server side. On the arrival of an Ethernet frame on one of the adapters it delivers it to the adapter on the other side by default seemingly working as a network bridge. If the source or the destination of the frame is detected to be one of the known clients, it will go for special treatment instead.

The treatment will introduce delay, delay jitter, packet loss and bandwidth limiting to the network frame. Each client can have their own interference set with the above parameters separately for both transmission directions. The Source Router interference parameters and their range can be seen in table 1.

Table 1. Source Router parameters

Parameter	**Unit**	**Step**	**Range**
Delay	µs	1 µs	0–30 seconds
Delay jitter	µs	1 µs	0–30 seconds
Packet loss	%	1 %	0–100 %
Bandwidth	kbps	1 kbps	0–1000 Mbps

For network frame capture the Source Router software utilizes the Linux libpcap-library which is an open source C-compatible library that allows the user to easily capture network frames as they arrive at a network adapter. The

availability of such a library and the relative easiness of programming on Linux versus using a big and clunky Windows programming environment such as Microsoft Visual Studio was a deciding factor on the usage of Linux as the target system for the software. The operational diagram of Source Router can be seen in figure 2.

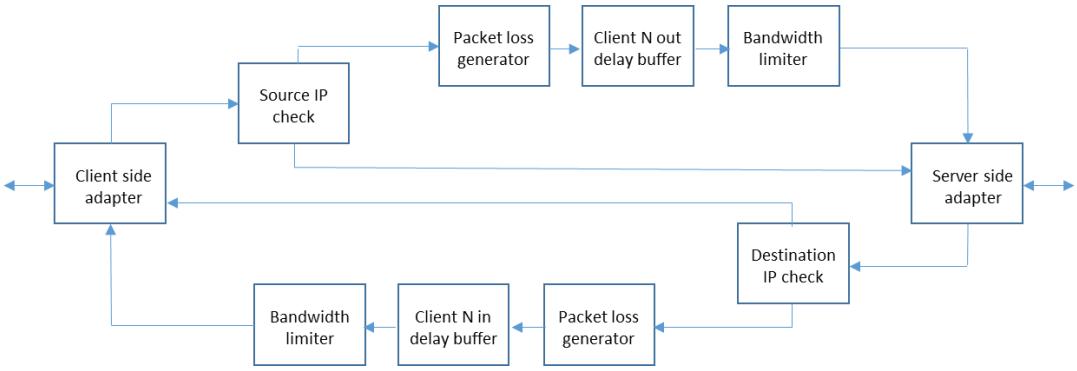

Figure 2. Source Router operational diagram

There is a total of four threads in the software were two of the threads will handle the arriving frames on each network adapter respectively. One of the threads will handle incoming button press data from the Button Press Collector software. The last or the main thread will handle the Web-based user interface. A diagram showing the threads can be seen in figure 3.

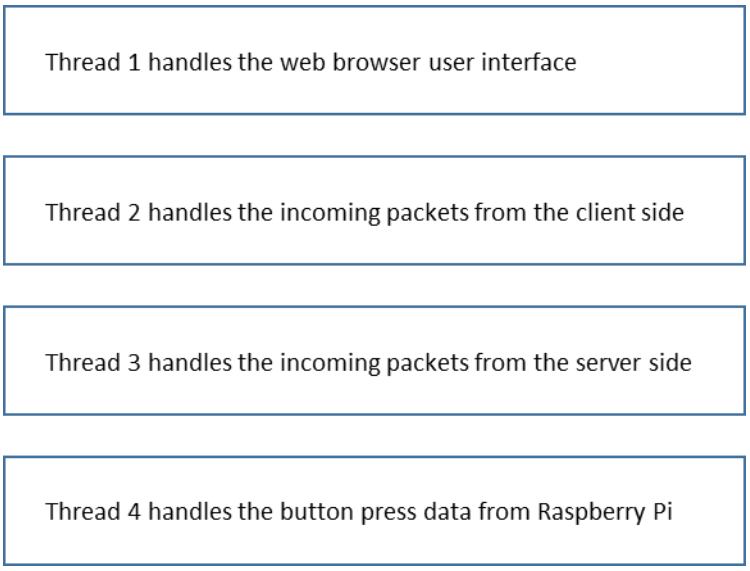

Figure 3. Source Router threads

Each of the two frame handling threads are basically identical except that one will try to detect packets coming in from a known client and the other will de-

tect packets that go out to clients instead. If the captured Ethernet frame contains anything else than an IP version 4 (IPv4) packet it will be forwarded to the opposite side network adapter and sent out as soon as it is received.

This assures that mechanisms like Address Resolution Protocol (ARP) will function without problems or delay. However, it also means that the interference operations the software carries out will only work for IPv4 packets and not for IP version 6 (IPv6) packets. The type of the frame payload will be detected from the EtherType field of the frame where 0x0800 means an IPv4 packet.

If the captured frame contains an IPv4 packet, the source or the destination IP address of the packet, depending on the transmission direction, will be compared to a list of client IP addresses. The software can be configured with up to 16 client IP addresses for this inspection by the user. If the compared IP address matches with a known client IP address the frame will be placed to a delay buffer for that client.

If packet loss is in use the software will use a random generator to decide whether the packet to be buffered is lost and if it is, it will be discarded instead. By default, each client has a frame buffer of 8 megabytes each direction meaning the total buffer memory allocated by the software is 32 times 8 megabytes or 256 megabytes. The buffer size value can be defined by the user.

When all the incoming frames on the adapter are processed the software will check the frame buffer of each client whether there are any outgoing frames due to be sent. If a buffer contains a frame whose send time is due the frame will be sent out from the opposing network adapter it came in. Then the next frame is checked, and this will be continued until all frames with a due time have been sent. All the frames of the first client are processed first before moving to the second client.

If bandwidth limiting is in use and the bandwidth cap has been reached, no more packets will be sent even if their send time is due. The following chapters will further discuss the operation of each of the various functions of the software.

5.1.1 Web browser user interface

The web browser interface allows the system operator to control the Source Router software remotely via a web browser. The user interface can be opened in a web browser by connecting to port 26102 of the Source Router computer. A screenshot of the browser user interface can be seen in figure 4. The interface has the global server status table, the client settings table, the client status table, client settings section and graphs of button presses, packet loss and delay. The user interface uses the term packet to refer to all types of frames passing through the system. JavaScript, jQuery and WebSocket technologies are used to implement the user interface.

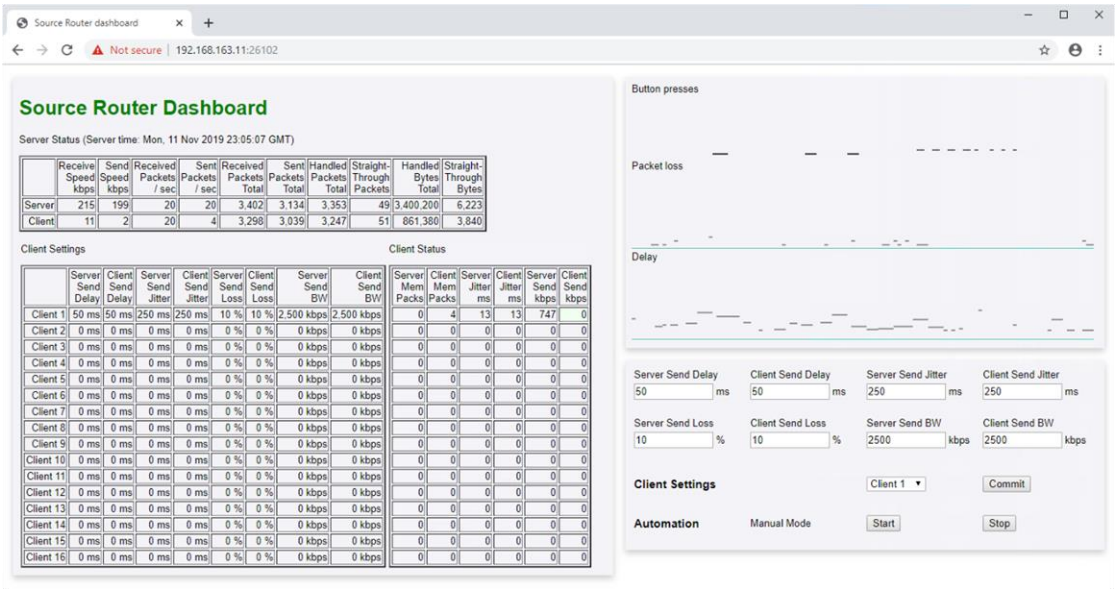

Figure 4. Source Router browser user interface

For proper function of the browser user interface a WebSocket connection to the Source Router software is required. The user interface web page tries to automatically connect to the software when the web page is opened. When a WebSocket connection is successfully established the Source Router Dashboard text is displayed in green color. When there is no connection to the

Source Router software the text is displayed in red color instead. Figure 5 shows both statuses.

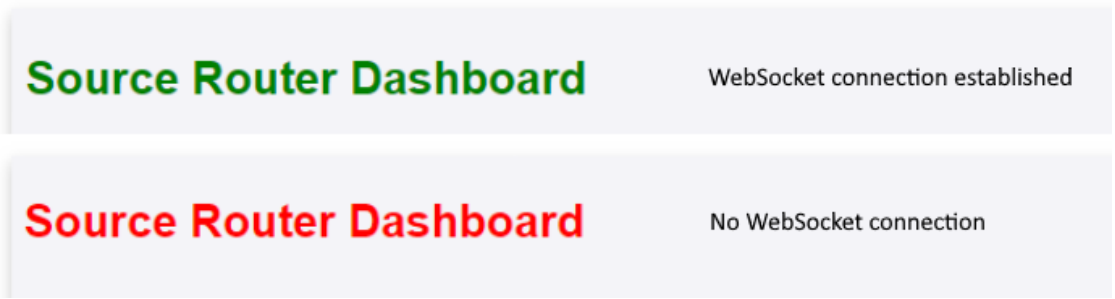

Figure 5. WebSocket connection status

The **server status table** shows global statistics of the system operation. The receive and send speeds of the whole system can be seen in kilobits per second and packets per second. It also shows the total amount of received and sent packets by the system as well as how many of those packets have been handled by the system and how many have gone straight-through the system with no manipulation. Last the number of handled and straight-through bytes can be seen.

The **client settings table** is where the currently used interference parameters for each client can be seen. It shows the delay, delay jitter, packet loss and bandwidth parameters for each client in both transmission directions. These parameters stay in effect until they are changed by the user or the Source Router software is restarted when all the parameters go to zero.

The **client status table** shows the per client status of the system. This includes the number of packets in delay buffers for the client and the current delay jitter value for the client if jitter is enabled. It also shows the data speed in kilobits per second for that client. The cell backgrounds of the speed fields are color coded depending on the client bandwidth setting. When the client is using less than 50% of allocated bandwidth the cell background color is green. The color is yellow when the bandwidth is between 50–80% and red when more than 80% of bandwidth is used. This provides a quick way to see when a client is using all bandwidth allocated to it.

The graph area shows a quick overview of three of the system functions. All graphs are color coded so that each client has its own color in the graphs. Black color in each graph always means client 1. The graphs show data for the last 30 seconds and each graph is updated 4 times per second meaning the step size of the graph is 0.25 seconds.

The **button presses graph** shows the lag indication button presses of all the 16 clients. Each client has its own row in the graph and the graph provides a way for the system operator to see when a user has pressed their lag indication button. If a user is continuously pressing their button the operator can decide whether they will disqualify the user from their game test.

The **packet loss graph** shows the amount of simulated packet loss. The number of lost packets for each client during the last step is shown in the graph. The highest value means that 20 or more packets have been lost. The **delay graph** shows the highest delay value for outgoing packets for each client during the last step. A total of base delay and delay jitter is shown here. The highest value means a delay of 500 milliseconds or more.

The **client settings area** lets the system operator to manually manipulate the interference values of each client in the system. First the desired client needs to be selected from the drop list box which will update the text boxes with current values for that client. Then the values can be changed and when the Commit button is pressed the values are sent to the Source Router software.

Pressing enter in the text boxes will also commit the values. If the jitter settings are the same for both the client and server send directions the random delay jitter for both will be the same so that the jitter values move in tandem. A value of 0 in the bandwidth fields means that bandwidth limiting is turned off and the system operates at full available bandwidth.

The automation Start and Stop buttons will respectively start and stop the system running a preconfigured automation sequence. Each client will step through the same interference pattern, but the operator can select a time range for each step in the script. The system will randomize the step lengths

using this time range so that each client will have their parameters changed at a different time.

5.1.2 Delay buffer

Each client has two delay buffers allocated for it. The first buffer is for server sent packets and the other buffer is for client sent packets. When a client packet is received it is always buffered for that client's own buffer even when the delay settings are zero. When a delay buffer is full any new packets to be buffered are discarded instead. After all incoming packets are handled, the buffers are checked for any outgoing packets and all packets that are due to leave are sent out.

Even though each packet is marked with an individual due time and a changing delay in the form of jitter may mean that the due time of a later packet is coming earlier than the due time of a packet before it, the later packet is not sent if the earlier packet is not due. In other words, packets that come first will always leave first so that the packet sequence is preserved, no packet can overtake another in this process. For some packets this means an increased delay and packets that came in at different times may leave at the same time in a burst.

5.1.3 Bandwidth limiting

Bandwidth limiting works as part of the delay buffer mechanism so that when a packet is due to be sent the bandwidth limit of the client is checked and if the packet would exceed the bandwidth limit it will not be sent, but is kept in the buffer for a later time.

The operator gives the bandwidth limit on a kilobits per second basis. This setting is then divided into 20 parts. Each of these parts give the maximum number of kilobits that can be sent in a 50-millisecond interval. When a new 50 ms interval begins the counter that counts the number of kilobits sent is cleared and new packets can be sent during that interval until the calculated limit is reached when any sends are halted until a new 50 ms interval begins.

For the reason of efficiency, the total length of each frame is counted towards the operator set bandwidth limit. So, not only the IPv4 payload, but also the frame header is considered towards this limit. Therefore, the bandwidth limit is a raw data limit and not an application data limit which means that a small difference in bandwidth may result if an application is desired to be limited to a certain bandwidth.

5.1.4 Random number generation

The simplest and most commonly used random number generator for digital applications is the linear feedback shift register (LFSR). It consists of a shift register with a certain bit length (m) and an exclusive-OR (XOR) gate whose output is fed back to the shift register input. The XOR inputs are the last bit and nth bit of the shift register where n can be a user chosen value. Each time a new random number is required to be generated the shift register is shifted by one bit and a pseudo random number is generated in the shift register. (Horowitz & Hill, 975.) Figure 6 shows the structure of the LFSR.

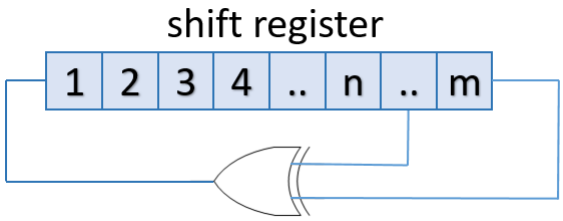

Figure 6. LFSR random number generator.

The LFSR will repeat itself after a certain number of cycles. For a shift register with the length of n, the maximum number of values the shift register can go through is $2^n - 1$ as there is one value the register can never reach. This value is 0 which is an illegal value for an LFSR. A value of 0 would get the LFSR stuck forever as the XOR would regenerate the value 0 as the next value for the shift register. (Horowitz & Hill, 975.)

Not all LFSRs can, however, reach the maximal sequence length. But when the length of the shift-register and the position of the other XOR input value are chosen carefully the maximum length can be reached. It is also possible to use a multi-input XOR gate where more than one middle value of the shift register is used as the XOR input with the last bit. Each such combination will

generate their own individual sequence that differs from others. (Horowitz & Hill, 975.)

The simplicity and speed of an LFSR make it the ideal choice for random number generation in the Source Router software. Especially speed is of essence here as the software is required to bridge Ethernet frames as fast as possible. There are several subtypes of LFSRs all of which are based on shift registers and XOR operations. The LFSR presented above is a classical Fibonacci type LFSR.

Xorshift on the other hand is a more modern take on the LFSR and was published in 2003 by mathematician and computer scientist George Marsaglia. It is a simple and fast random number generator and can be easily implemented in the C language (Marsaglia, 1). This is the same language Source Router was implemented in.

The generation of a new 64-bit random number requires a total of three shifts and three XORs. A total of 275 triples are given where each triple consists of three different shift-register positions that are used as the inputs for the XORs. Furthermore, there are 8 different ways given the triples can be shifted. In all cases some of the shifts happen up and some happen down. The combination of 8 ways and 275 triples gives a total of 2200 different Xorshift sequences for 64-bit use. (Marsaglia, 3.)

This means that even with the same random number seed each of these 2200 sequences would be independent from each other. As it was desired that the random sequences for each of the 16 possible Source Router client systems would be different and uncorrelated Xorshift was thought to be the ideal choice for Source Router random number generation. Furthermore, the provided simple C source code could be easily implemented in the Source Router software.

5.1.5 Packet loss and jitter generation

Both packet loss and jitter functions use an Xorshift random number generator. For packet loss each client has two different Xorshift functions, one for client sent packets and another for server sent packets. Inside these functions there are different Xorshift parameters and seed values for each client to keep the correlation to minimum.

When a packet is about to be stored in the delay buffer the packet loss random generator is used to generate a uniformly distributed random number between 0 and 99. This number is then compared to the client's packet loss setting. If the random value is higher than the packet loss setting, the packet is not stored, but discarded instead to simulate packet loss.

The delay jitter functions similarly to the packet loss function so that each client has two Xorshift functions, again for client and server sent packets. Also, each function has different Xorshift parameters and seed values for every client. However, instead of generating one uniformly distributed random number at a time, four of such numbers are generated and added together for a gaussian distribution.

Since the delay value cannot be negative, however, whenever the value is negative its sign is inverted so that each time a positive value is generated. This value will then be used as the current delay jitter value for the client. The jitter value for each client is updated once a second. When the jitter setting is 100 ms the average jitter is 22.9 ms. The distribution of the jitter values can be seen in figure 7.

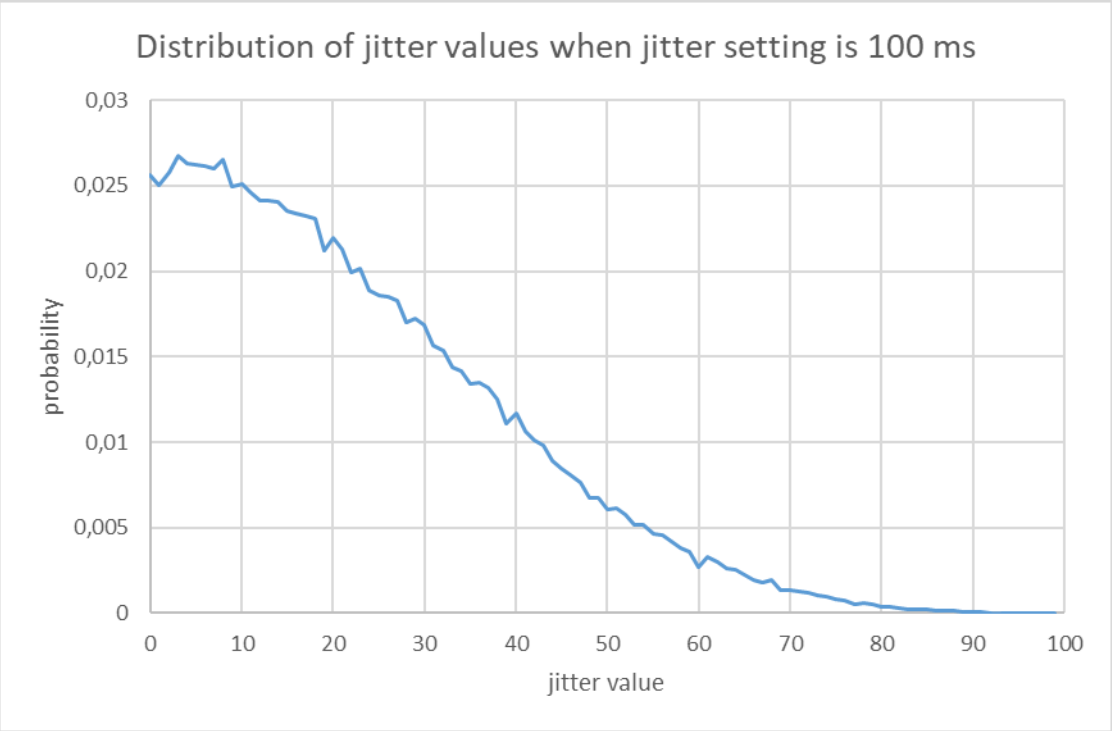

Figure 7. Distribution of jitter values

5.1.6 Configuration and log files

Source Router can be configured with a configuration file called sorou.conf which needs to be in the same folder than the Source Router executable when the software is started. Figure 8 shows an example of the file. In the file the operator can define the client-side and server-side network interfaces with the CLIENTINTNAME and SERVERINTNAME keywords respectively. The operator can also define the amount of delay buffer memory that is allocated per direction for each client with the DELAYMEMSIZE keyword. By default, this is 8 megabytes.

Lastly, the operator needs to define as many of the 16 client IPs as they desire to use. These are defined by the CLIENTIPx keyword where x is the number of the client. The operator can either use numeric IP addresses or the names of the client computers as the software is able to convert these to IP addresses. Comments can be entered by starting a line with the # character.

```
# Client side network interface name
# Example: CLIENTINTNAME = ens3
CLIENTINTNAME = ens3

# Server side network interface name
# Example: SERVERINTNAME = ens2
SERVERINTNAME = ens2

# Delay memory size in kilobytes per client per direction or twice this amount per client
# Example: DELAYMEMSIZE = 8192
DELAYMEMSIZE = 8192

# Client IP addresses, maximum 16 clients supported (CLIENTIP1 - CLIENTIP16)
# Example: CLIENTIP1 = 192.168.163.101
# Example: CLIENTIP1 = lmclient1.ictlab.fi
CLIENTIP1 = 192.168.163.101
CLIENTIP2 = 192.168.163.102
```

Figure 8. Source Router configuration file

A log of the client settings is kept in a file called clientdata.txt which is updated whenever the settings for any of the clients is changed either manually by the operator or automatically by the software. The file is stored in the Source Router directory. An example of the client settings log file can be seen in figure 9. If the file does not exist when Source Router is started a new file will be generated. Each time the Source Router is run it adds a header that explains the file contents.

Each line in the file describes the settings for one client at that point in time. The time is recorded in microseconds since 1.1.1970 and is thus a long number. Next comes the client number that can be between 0–15 for clients 1–16 respectively. The client number is followed by settings for that client and include base delay, delay jitter, packet loss, bandwidth and actual total delay for the client for both client and server send directions. The file is designed in such way that it can be easily imported to Excel for further analysis.

```
# Client setting data, start at: 1574036259390921 us from 1.1.1970
# microsecond, client, client delay, server delay, client jitter, server jitter,
# client packet loss, server packet loss, client bandwidth, server bandwidth,
# client actual delay, server actual delay
1574036259390939,0,0,0,0,0,0,0,0,0,0
1574036259390987,1,0,0,0,0,0,0,0,0,0
1574036259390992,2,0,0,0,0,0,0,0,0,0
1574036275326153,0,10000,10000,10000,10000,10,10,10000,10000,10000,10000
1574036276185222,0,10000,10000,10000,10000,10,10,10000,10000,19979,19979
1574036277187197,0,10000,10000,10000,10000,10,10,10000,10000,16016,16016
1574036278188837,0,10000,10000,10000,10000,10,10,10000,10000,15025,15025
1574036293780863,1,20000,20000,20000,20000,20,20,20000,20000,20000,20000
1574036305807936,2,20000,10000,40000,30000,60,50,80000,70000,20000,10000
```

Figure 9. Source Router client settings log

The button presses by the users are captured by the Button Press Collector software and sent to the Source Router software via network for logging. The log will be stored under the name buttondata.txt in the Source Router directory. An example of this file can be seen in figure 10.

Each line in the file has the time of the button presses in the same microsecond format as the settings file. This is followed by a number where each client has its own bit. When the number is 0, no buttons are being pressed. When the number is non-zero, the bit position tells which clients have their button pressed. The lowest bit or bit value 1 means client 1, bit value 2 means client 2 and so on up to 32768 which means client 16. Button presses and button releases can be extracted from this data. The file can be imported to Excel for further analysis.

```
# Data from buttons, start at: 1573513065292672 us from 1.1.1970
# Each client has its own bit in buttondata, first client has lowest bit,
# value 1 means button pressed
# microsecond, buttondata
1573513065292724,0
1573513067448299,1
1573513068105190,0
1573513070968071,1
1573513072168270,0
1573513072689135,2
1573513073354717,0
1573513074349024,2
1573513075792514,3
```

Figure 10. Source Router button press log

To run automated sequences with the Source Router software a file called sequence.txt can be created. This file will be read by the Source Router when the software is started and contains the steps of the sequence that can be run using the user interface Start and Stop buttons. Each step of the sequence needs to be on its own line.

The line starts with the minimum time in seconds the sequence step is to be run and next comes the maximum step time setting. Source Router will randomly choose a different step time for each client between the minimum and

maximum times to ensure that each client steps through the sequence at a different speed. The last eight values on each line are the setting values to be applied at each step in the same order as in the client settings log. Figure 11 shows an example of such a sequence file.

```
# Client sequence data
# minimum step time, maximum step time, client delay, server delay,
# client jitter, server jitter, client packet loss, server packet loss,
# client bandwidth, server bandwidth
100,200,10,10,20,20,0,0,0,0
200,250,20,20,20,20,1,1,0,0
200,300,30,30,20,20,2,2,0,0
120,240,40,40,20,20,3,3,0,0
```

Figure 11. Source Router sequence file

5.2 Button Press Collector software

The operation of the Button Press Collector software is simple. When started it will try to connect to the Source Router software via network and if the connection is successful the program will go into its main loop where it will poll the I/O pins of the Raspberry Pi for button presses of the player lag indication buttons. To read the I/O pins the software uses the Wiring Pi library.

When a button press is detected the software will generate a 16-bit number containing the status of each of the buttons in a one bit per button format where a bit value of 1 means that the button of that player is currently being pressed. The status of the first player lag detection button is in the lowest bit, the status of the second player button is in the second lowest bit and so on.

The software also records the current time in microseconds from the start of the year 1970 as a 64-bit number and sends the time and button press data to the Source Router software which then shows and saves it for further analysis. Port 26101 is used in the Source Router software for this connection. The time format is the same as used by the Source Router software.

The software can be configured in a configuration file called precol.conf that needs to be in the Button Press Collector folder when the software is started. The configuration file works the same way as the Source Router configuration file and contains only one keyword. This keyword is SOURCEROUTERIP

which defines the IP address of the Source Router software so that the button presses can be sent to it. An example of the configuration file can be seen in figure 12.

```
# Source Router IP address
# Example: SOURCEROUTERIP = 192.168.163.11
SOURCEROUTERIP = 192.168.163.11
```

Figure 12. Button Press Collector configuration file

An add-on card was made for the Raspberry Pi 3B where button boxes can be connected. Two button boxes were made to test the system with. The boxes contain nothing more than a simple push button and two wires that connect to the Raspberry I/O and GND pins. Internally the Raspberry Pi is configured to pull the I/O pins high so that when the button is not pressed a bit value of one is read from the I/O pins.

When the user presses the lag button its I/O pin will be pulled low via the button. The state of each I/O pin is reversed before sending the status to the Source Router software so that a logical one means that a button is pressed and a zero means it is not pressed. The only component that is connected to the I/O pins via the connectors and wires is the button. The Button Press Collector software polls the I/O pins once in every 500 microseconds. Several poll intervals were tested and the selected one seems to be a good one to take care of debouncing so that when the user presses the button there is just one pulse detected from the button. A picture of the Raspberry Pi and a button box can be seen in figure 13.

Figure 13. Raspberry Pi 3B with connector card and a button box

For the connections of the two button boxes to the Raspberry Pi two ground pins and two general I/O pins were selected. The ground pins used are pins 14 and 34 in the I/O connector. These were selected because they are next to the selected I/O pins which are pins 12 and 32 for GPIO18 and GPIO12 respectively. GPIO18 is the pin for client 1 and GPIO12 is the pin for client 2. To support more clients any of the GPIO pins can be used and the Button Collector Software can be modified. The connections can be seen in figure 14.

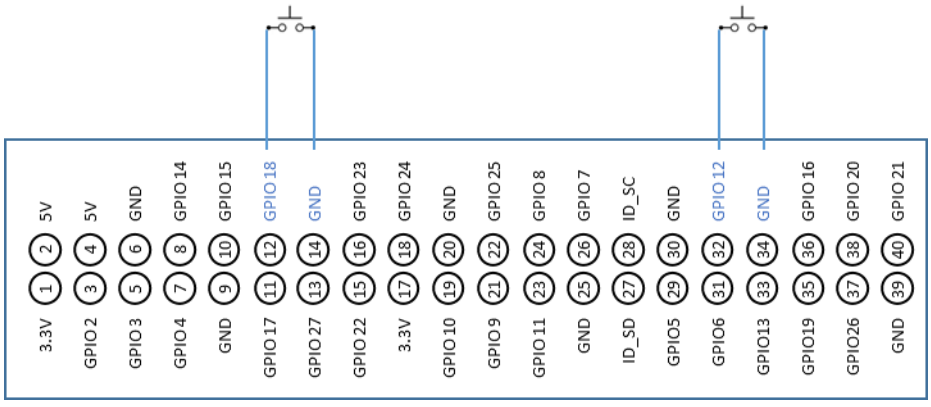

Figure 14. Button box connections to the Raspberry Pi I/O connector

6 SYSTEM TESTING

An important part of building the LaggingMeter system was system testing. This included both testing of the Source Router software and the Button Press

Collector software. However, since the Button Press Collector software is a simple program whose operation was quickly verified, the actual tests concentrated on testing the Source Router software that is the main component of the system.

The bridging computer OS was Fedora 30 and it was equipped with two similar USB-to-Ethernet adapters. Each of these adapters would connect to a USB 2.0 port and add an extra 100 Mbps Ethernet connection to the computer. At first only one of these 100 Mbps USB adapters was used in pair with the internal 1 Gbps network adapter of the computer. This resulted in a system with an imbalance in the network connectivity and the 100 Mbps adapter became a bottleneck. Running internet speed tests with such system proved to get the tests stuck as only 10% of the packets to one direction would be delivered due to buffering overflow.

Therefore, having two similar adapters was found to be a necessity and another similar USB-to-Ethernet adapter was acquired. Using two of these USB adapters in the bridging computer, one for server side and one for client side, proved to fix this issue and the system operation was very reliable. The speed tests would run with no issues and packets were not lost as they could be sent out both directions at the same speed as they came in. These two USB adapters also allowed basically any computer with two USB ports, suitable drivers and capability of running Linux operating system to work as the bridging computer.

Several different tests were thought up to be made on the system to prove that each of the features work on their own and that the whole system works as planned. The next chapters will go through the tests and the test results.

6.1 TEST 1: Four different bridging computers

Source Router software was tested using four different computer systems and the test results were compared between all these systems. The reason for this test was to find out how modern and efficient the PC system needs to be for

the Source Router software to operate properly. Two different network environments where used in the test. One PC system was tested in an office network and three PC systems where tested in a home network.

Two of the PCs were no more than 2 years old and two were several years old. One of the newer PCs was a modern laptop and the other was a desktop. Similarly, the older systems had one laptop and one desktop. In the test results these four test systems will be called DNew, LNew, DOld and LOld. D means desktop and L means laptop. The main specifications of each system are displayed in table 2.

Table 2. Test system specifications

	DNew	**LNew**	**DOld**	**LOld**
Purchased	Dec 2017	Feb 2019	Apr 2012	Jul 2014
CPU	i5-7500	Ryzen 5 2500U	i3-2120	Pentium N3520
Cores / Threads	4 / 4	4 / 8	2 / 4	4 / 4
Memory	16GB DDR4	8GB DDR4	16GB DDR3	4GB DDR3

The DesktopNew test system was in an office network that had strict port security. For this reason, one of the office PCs was set up to share its internet connection to other PCs via Windows Internet Connection Sharing feature. This way the MAC addresses of other computers will not show up to the office network switch, but it will only see the MAC address of the Internet Sharing computer instead. The diagram for the office test system can be seen in figure 15.

The internet connection for the Internet Sharing computer was over 100 Mbps both directions and was so able to handle the 100 Mbps connections of the used USB-to-Ethernet adapters. So, the internet upload and the download speed for the bridging computer and eventually for the client computers was 100 Mbps both directions. The Internet Sharing computer also acted as a server for the iperf bandwidth tests.

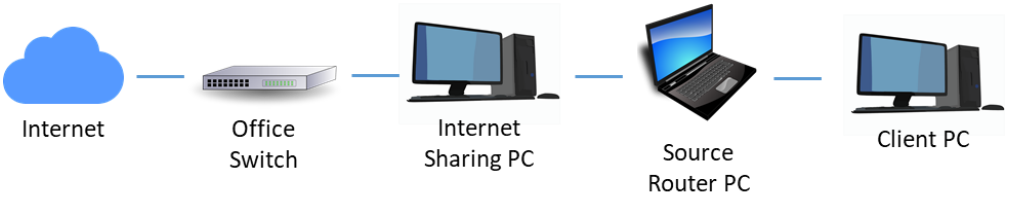

Figure 15. Office network test system

The rest of the test systems were in a home network which had a cable modem connection to the internet. The internet download speed of this network was 100 Mbps and the upload speed was 10 Mbps. However, on the LAN level the client computers had a 100 Mbps connection both directions limited only by the used 100 Mbps USB-to-Ethernet adapters. For iperf bandwidth tests another PC was connected to the cable modem to act as an iperf server for the tests. Figure 16 shows the diagram of the home network.

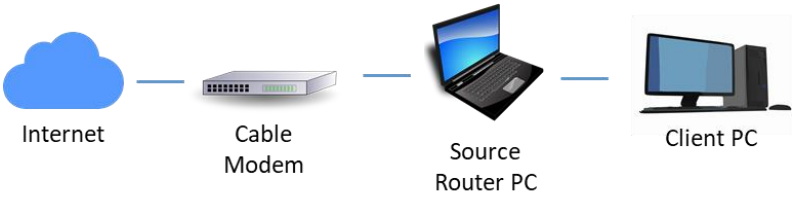

Figure 16. Home network test system

6.1.1 Bypass test

First, to establish a baseline for the home and office networks, simple tests were carried out where the bridging computer running Source Router software was completely bypassed and not included in the network at all. This reflects the performance of the tested network on its own so that the effect of adding the bridging computer in between the client and server computers can be more clearly seen in the results of the next tests.

Packet loss was tested by running the ping command on the client computer and the server was pinged 100 times. Neither network lost any packets in the test as can be seen from the results in table 3.

Table 3. TEST 1, bypass test, packet loss

	Office network	**Home network**
Packet loss	0%	0%

Delay was tested with the ping command pinging the server from the client 100 times. This gives the minimum, maximum and average delay values from which also the LAN jitter can be seen. The minimum delay will be the base delay or the lowest delay the network connection can achieve. The delay test results are presented in table 4.

Table 4. TEST 1, bypass test, delay

	Office network	**Home network**
Delay, minimum	3 ms	0 ms
Delay, maximum	4 ms	4 ms
Delay, average	3 ms	0 ms

Bandwidth was tested with two internet speed test services, fast.com and speedtest.net. With internet tests there is always the added uncertainty that comes from the fact that the internet is a vast distributed system whose workings cannot be clearly predicted. Therefore, the delay, the bandwidth and packet transit in general can vary wildly from test to test.

The maximum theoretical bandwidth for the office network server send speed was 100 Mbps and the client send speed was also 100 Mbps. The maximum speeds for the home network were 100 Mbps and 10 Mbps. Usually server send is called download and client send is called upload, but in these results server and client send are used to signify which computer is sending packets as the Source Router software will manipulate packets sent by either the server or the client computer.

The results from the fast.com test service are presented in table 5. The test was run three times and an average test bandwidth was calculated. It can be seen in the office network test results that for some reason the client send or upload direction was not functioning to its full capability of 100 Mbps during the test. Also, the home network client send speeds were somewhat lower than expected. Here the maximum would be 10 Mbps.

Table 5. TEST 1, bypass test, fast.com bandwidth

	Office network		Home network	
	Server send	**Client send**	**Server send**	**Client send**
BW, test run 1	94 Mbps	70 Mbps	95 Mbps	8.3 Mbps
BW, test run 2	93 Mbps	77 Mbps	96 Mbps	9.0 Mbps
BW, test run 3	93 Mbps	78 Mbps	91 Mbps	8.0 Mbps
BW, average	93 Mbps	75 Mbps	94 Mbps	8.4 Mbps

Table 6 shows the bandwidth measurements for the speedtest.net service. The speed in both directions in the office network was on average the same or 90 Mbps. The speeds in the home network are near the maximum practical speeds one can expect. Overall, the speedtest.net tests are more consistent. However, since the tests are internet tests the general internet traffic can affect the results.

Table 6. TEST 1, bypass test, speedtest.net bandwidth

	Office network		Home network	
	Server send	**Client send**	**Server send**	**Client send**
BW, test run 1	92.03 Mbps	88.25 Mbps	95.41 Mbps	9.85 Mbps
BW, test run 2	89.05 Mbps	91.94 Mbps	95.20 Mbps	9.83 Mbps
BW, test run 3	89.28 Mbps	88.58 Mbps	95.37 Mbps	9.81 Mbps
BW, average	90.12 Mbps	89.59 Mbps	95.33 Mbps	9.83 Mbps

6.1.2 Packet loss test

Packet loss test was the first Source Router enabled test and it was carried out by two methods. The first method was to use the Windows built-in ping command that will ping the device whose IP address is given and then waits for a response from the device for a certain period and reports whether it received a response from the other device or whether there was no response which is assumed to be a lost packet.

The first ping test was carried out by using the built-in Windows ping command. All packet loss tests were carried out on the new PCs first and during the tests it was decided that since iperf was a much quicker test and it gave

more consistent results the ping tests would not be carried out on the old computers. The results of this test are presented in table 7.

The packet loss values from each test are mostly within few percentage points of the set value. However, this ping test seemed to occasionally show very different values than expected. Each ping is sent just once a second so the execution time of the ping test can be several minutes. Therefore, a situation where some other software is transferring data may happen to ruin the results of the ping test. For that reason, this test was skipped for the old test systems. This packet loss test was run with 100 pings from the client to the gateway using the following command:

- ping 192.168.0.1 -n 100

Table 7. TEST 1, packet loss, ping

	DNew		**LNew**	
	Server send	**Client send**	**Server send**	**Client send**
No loss	0 %	0 %	0 %	0 %
25% loss	26 %	28 %	27 %	22 %
50% loss	53 %	53 %	52 %	50 %
70% loss	71 %	77 %	74 %	69 %
90% loss	93 %	91 %	85 %	90 %

The second packet loss test method was to use an external tool called iperf which can carry out connectivity tests between a server and a client and will report data like bandwidth and packet loss when the test is finished. This test completes in a matter of seconds meaning that a chance for other software ruining the results is much lower. Results for this test can be seen in table 8 and are very consistent between each test system. The commands that were used are listed below:

- server command: iperf-2.0.14a-win.exe -s -u -i 1
- client command: iperf-2.0.14a-win.exe -c 192.168.0.16 -u -d

Table 8. TEST 1, packet loss, iperf

	DNew		LNew		DOld		LOld	
	Server send	**Client send**	**Server send**	**Client send**	**Server send**	**Client send**	**Server send**	**Client send**
No loss	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %
25% loss	25 %	24 %	25 %	26 %	24 %	26 %	25 %	26 %
50% loss	51 %	49 %	48 %	51 %	51 %	51 %	52 %	49 %
70% loss	69 %	70 %	71 %	70 %	71 %	70 %	71 %	70 %
85% loss	85 %	84 %	83 %	85 %	86 %	84 %	84 %	84 %

6.1.3 Base delay test

The base delay test was carried out by using the ping command running 30 pings to the gateway. The raw values for minimum, maximum and average ping are presented in table 9. The command used to run this test is listed below:

- ping 192.168.0.1 -n 30

Table 9. TEST 1, base delay

	DNew			LNew			DOld			LOld		
	Server send			**Server send**			**Server send**			**Server send**		
	Min	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**	**Max**	**Avg**
0 ms	1	6	**2**	2	7	**3**	2	6	**3**	2	4	**2**
10 ms	11	15	**12**	12	14	**12**	11	15	**13**	11	14	**12**
50 ms	52	54	**53**	52	54	**53**	51	54	**52**	51	55	**52**
100 ms	101	105	**102**	102	104	**102**	101	105	**103**	101	105	**102**
300 ms	301	305	**302**	302	304	**302**	302	306	**303**	301	304	**302**
	Client send			**Client send**			**Client send**			**Client send**		
	Min	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**	**Max**	**Avg**
0 ms	1	4	**2**	2	5	**2**	1	5	**2**	1	4	**2**
10 ms	11	15	**12**	12	14	**12**	11	15	**13**	11	14	**12**
50 ms	51	55	**52**	51	57	**52**	51	55	**53**	51	55	**52**
100 ms	101	104	**102**	101	104	**102**	101	107	**103**	101	104	**102**
300 ms	301	304	**302**	301	304	**302**	301	309	**303**	301	304	**302**

The delays in the previous table show the total delay between the client and the server. To extract the Source Router generated delay from these results a new table was generated. The table shows just the average delays and has the 0 ms delay subtracted from them. This removes the effect of random network delays and shows the effect of the Source Router to the delays. As can be seen in table 10 the Source Router delays are very accurate on each test system.

Table 10. TEST 1, base delay, relative average values

	DNew		**LNew**		**DOld**		**LOld**	
	Server send	**Client send**	**Server send**	**Client send**	**Server send**	**Client send**	**Server send**	**Client send**
10 ms	10	10	9	10	10	11	10	10
50 ms	51	50	50	50	49	51	50	50
100 ms	100	100	99	100	100	101	100	100
300 ms	300	300	299	300	300	301	300	300

6.1.4 Jitter test

To do the jitter test the ping command was once again utilized. Similarly, as in the previous test the minimum, maximum and average delays in each test were recorded. However, unlike in the previous test, the number of pings was raised to 100 to better catch the changes in the jitter values. Table 11 shows the raw measurements. The following command was used in the test:

- ping 192.168.0.1 -n 100

Table 11. TEST 1, delay jitter

	DNew			**LNew**			**DOld**			**LOld**		
	Server send			**Server send**			**Server send**			**Server send**		
	Min	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**	**Max**	**Avg**
0 ms	1	5	**2**	1	5	**2**	1	6	**3**	1	5	**2**
10 ms	1	11	**4**	1	10	**4**	3	12	**5**	1	10	**4**
50 ms	2	40	**14**	2	43	**14**	3	42	**14**	2	41	**13**
100 ms	2	76	**26**	1	74	**23**	1	78	**24**	2	75	**25**
300 ms	2	268	**79**	5	232	**75**	7	232	**77**	2	267	**78**

	Client send			Client send			Client send			Client send		
	Min	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**	**Max**	**Avg**
0 ms	1	6	**2**	1	4	**2**	1	5	**3**	1	5	**2**
10 ms	2	10	**5**	2	10	**4**	3	10	**5**	2	12	**4**
50 ms	1	41	**14**	2	45	**13**	3	46	**14**	2	47	**15**
100 ms	4	90	**28**	2	90	**27**	1	90	**27**	3	78	**38**
300 ms	3	221	**78**	2	218	**75**	6	222	**82**	6	207	**71**

Again, to extract the effect of the Source Router to the delay values, the average delay values had the average delay of 0 ms jitter subtracted and a new table was generated. These results can be seen in table 12. Once again, each test system shows very similar values for each delay jitter value.

Table 12. TEST 1, delay jitter, relative average values

	DNew		LNew		DOld		LOld	
	Server send	**Client send**	**Server send**	**Client send**	**Server send**	**Client send**	**Server send**	**Client send**
10 ms	2	3	2	2	2	2	2	2
50 ms	12	12	12	11	11	11	11	13
100 ms	24	26	21	25	21	24	23	36
300 ms	77	76	73	73	74	79	76	69

6.1.5 Bandwidth test

Bandwidth test was carried out in three parts by using web services and local area network measurement. The first part was made using the fast.com website that gives a quick bandwidth result. These measurements can be seen in table 13.

The tests were first carried out on the new PC systems. Since the fast.com test was found to be least consistent it was not used with the old PC systems since the two other tests provide enough results to determine the LAN and the Internet speeds for each system. As can be seen from the test results both DNew and LNew systems showed similar results when bandwidth limiting was activated.

Since DNew was in office network with 100M/100M bandwidth and LNew was in the home network with 100M/10M bandwidth, the full speed client send speed of LNew was only 10Mbps. Also, for an unknown reason the full speed test results of DNew for both directions were not very close to the 100Mbps as would be expected. Though, this test was an internet test and internet traffic might have affected the test results.

Table 13. TEST 1, bandwidth, fast.com

	DNew				**LNew**			
	Server send				**Server send**			
Mbps	**Test 1**	**Test 2**	**Test 3**	**Avg**	**Test 1**	**Test 2**	**Test 3**	**Avg**
Full	90	90	87	**89**	94	94	94	**94**
70	67	67	67	**67**	67	68	67	**67.3**
50	48	48	48	**48**	48	45	48	**47**
20	19	19	19	**19**	19	19	19	**19**
5	4.7	4.8	4.8	**4.8**	4.8	4.2	4.8	**4.6**
1	0.93	0.88	0.95	**0.92**	0.88	0.92	0.86	**0.89**
	Client send				**Client send**			
Mbps	**Test 1**	**Test 2**	**Test 3**	**Avg**	**Test 1**	**Test 2**	**Test 3**	**Avg**
Full	76	67	73	**72**	8.0	8.8	8.3	**8.4**
10	8.5	8.7	8.2	**8.5**	7.2	9.2	7.6	**8.0**
7	5.5	6.6	5.9	**6.0**	5.6	6.0	5.9	**5.8**
5	4.7	5.1	4.8	**4.9**	4.9	4.1	4.5	**4.5**
2	2.2	2.2	2.5	**2.3**	2.3	2.1	2.2	**2.2**
1	1.2	1.5	1.1	**1.3**	1.4	1.2	1.0	**1.2**

The second part was made using speedtest.net website that offers a similar service. The results of these tests are more consistent between each test system than the previous results from fast.com. The results can be seen in table 14 for new test systems and in table 15 for old test systems.

Table 14. TEST 1, bandwidth, speedtest.net, new systems

	DNew				LNew			
	Server send				Server send			
Mbps	**Test 1**	**Test 2**	**Test 3**	**Avg**	**Test 1**	**Test 2**	**Test 3**	**Avg**
Full	88.91	89.08	88.64	**88.88**	92.30	89.30	91.91	**91.17**
70	66.89	67.29	66.65	**66.94**	67.44	67.39	67.44	**67.42**
50	46.76	48.79	47.94	**47.83**	48.11	47.16	48.13	**47.80**
20	19.17	19.20	19.17	**19.18**	19.24	19.24	19.30	**19.26**
5	4.78	4.78	4.80	**4.79**	4.80	4.80	4.82	**4.81**
1	0.83	0.95	0.94	**0.91**	0.94	0.95	0.95	**0.95**
	Client send				**Client send**			
Mbps	**Test 1**	**Test 2**	**Test 3**	**Avg**	**Test 1**	**Test 2**	**Test 3**	**Avg**
Full	91.68	91.47	90.31	**91.15**	9.84	9.75	9.84	**9.81**
10	9.53	9.70	9.57	**9.60**	9.56	9.65	9.53	**9.58**
7	6.68	6.65	6.68	**6.67**	6.66	6.68	6.69	**6.68**
5	4.75	4.75	4.74	**4.75**	4.69	4.73	4.74	**4.72**
2	1.85	1.87	1.87	**1.86**	1.87	1.87	1.84	**1.86**
1	0.92	0.93	0.94	**0.93**	0.91	0.88	0.92	**0.90**

Table 15. TEST 1, bandwidth, speedtest.net, old systems

	DOld				LOld			
	Server send				Server send			
Mbps	**Test 1**	**Test 2**	**Test 3**	**Avg**	**Test 1**	**Test 2**	**Test 3**	**Avg**
Full	93.71	90.49	93.42	**92.54**	92.90	92.55	92.49	**92.65**
70	67.42	67.44	67.39	**67.42**	67.40	67.25	67.34	**67.33**
50	48.11	48.15	48.13	**48.13**	48.14	48.09	48.17	**48.13**
20	19.29	19.24	19.25	**19.26**	19.27	19.25	19.27	**19.26**
5	4.81	4.81	4.83	**4.82**	4.83	4.80	4.81	**4.81**
1	0.95	0.95	0.97	**0.96**	0.96	0.95	0.91	**0.94**
	Client send				**Client send**			
Mbps	**Test 1**	**Test 2**	**Test 3**	**Avg**	**Test 1**	**Test 2**	**Test 3**	**Avg**
Full	10.00	10.08	10.04	**10.04**	10.11	9.94	10.08	**10.04**
10	9.64	9.65	9.65	**9.65**	9.56	9.63	9.57	**9.59**
7	6.74	6.70	6.71	**6.72**	6.75	6.63	6.69	**6.69**
5	4.81	4.82	4.84	**4.82**	4.81	4.67	4.84	**4.77**
2	1.93	1.93	1.91	**1.92**	1.82	1.94	1.86	**1.87**
1	0.96	0.95	0.94	**0.95**	0.97	0.92	0.96	**0.95**

Lastly, a LAN based test software called iperf was used to measure the bandwidth in LAN environment only. This type of bandwidth test provided the most consistent test results between each test system and would be the preferred test method if any further testing on the whole system was to be made. Once again only the full 100 Mbps speed results show some difference. This is when the bandwidth limiting is turned off in the system. The results can be seen in table 16 for new test systems and in table 17 for old test systems. The commands used in the test, where BW means the bandwidth to be tested, were:

- server command: iperf-2.0.14a-win.exe -s -u -i 1
- client command: iperf-2.0.14a-win.exe -c 192.168.0.16 -u -d -b BW

Table 16. TEST 1, bandwidth test, iperf, new systems

	DNew				**LNew**			
	Server send				**Server send**			
Mbps	**Test 1**	**Test 2**	**Test 3**	**Avg**	**Test 1**	**Test 2**	**Test 3**	**Avg**
Full	95.0	95.0	95.1	**95.0**	93.6	93.6	93.6	**93.6**
70	68.1	68.1	68.1	**68.1**	68.1	68.1	68.1	**68.1**
50	48.7	48.7	48.7	**48.7**	48.6	48.7	48.7	**48.7**
20	19.5	19.5	19.5	**19.5**	19.5	19.5	19.5	**19.5**
5	4.87	4.88	4.89	**4.88**	4.88	4.89	4.89	**4.89**
1	0.98	0.98	0.98	**0.98**	0.99	1.00	0.99	**0.99**
	Client send				**Client send**			
Mbps	**Test 1**	**Test 2**	**Test 3**	**Avg**	**Test 1**	**Test 2**	**Test 3**	**Avg**
Full	95.2	95.2	95.3	**95.2**	93.8	93.9	93.9	**93.9**
70	68.1	68.1	68.2	**68.1**	68.1	68.1	68.0	**68.1**
50	48.7	48.7	48.6	**48.7**	48.6	48.6	48.7	**48.6**
20	19.5	19.5	19.5	**19.5**	19.5	19.5	19.5	**19.5**
5	4.88	4.88	4.88	**4.88**	4.87	4.90	4.90	**4.89**
1	0.98	0.99	0.99	**0.99**	0.97	1.00	0.99	**0.98**

The measurements in the iperf software provided results that were so consistent between runs so that the iperf tests for the old systems were run only twice. A major factor contributing in this consistency may be the fact that this test was run in a LAN environment.

Table 17. TEST 1, bandwidth test, iperf, old systems

	DOld			LOld		
	Server send			Server send		
Mbps	**Test 1**	**Test 2**	**Avg**	**Test 1**	**Test 2**	**Avg**
Full	94.5	94.9	**94.7**	95.1	95.0	**95.1**
70	68.2	68.2	**68.2**	68.1	68.1	**68.1**
50	48.6	48.7	**48.7**	48.6	48.6	**48.6**
20	19.5	19.5	**19.5**	19.5	19.5	**19.5**
5	4.88	4.88	**4.88**	4.86	4.85	**4.86**
1	0.99	0.98	**0.98**	0.96	0.96	**0.96**
	Client send			**Client send**		
Mbps	**Test 1**	**Test 2**	**Avg**	**Test 1**	**Test 2**	**Avg**
Full	89.9	89.5	**89.7**	95.4	95.4	**95.4**
70	68.0	68.2	**68.1**	68.1	68.0	**68.1**
50	48.7	48.6	**48.7**	48.6	48.7	**48.7**
20	19.5	19.5	**19.5**	19.5	19.5	**19.5**
5	4.88	4.88	**4.88**	4.89	4.83	**4.86**
1	0.99	0.98	**0.99**	0.97	0.97	**0.97**

6.1.6 CPU usage test

The CPU usage was measured by running the Linux utility called top that displays statistics over various processes. Each PC system was measured when the Source Router software was idle with no traffic. The results are presented in table 18. Another set of measurements was made where a client computer was running a full speed download test from speedtest.net downloading nearly 100 Mbps. The results of these measurements are shown in table 19.

Table 18. TEST 1, CPU usage when idle

	DNew	LNew	DOld	LOld
Meas. 1	18.6%	9.0%	13.0%	31.1%
Meas. 2	18.9%	8.6%	13.9%	30.4%
Meas. 3	18.9%	9.0%	12.6%	31.1%
Meas. 4	19.3%	9.6%	13.3%	30.8%
Meas. 5	18.6%	9.3%	12.6%	30.5%
Average	**18.9%**	**9.1%**	**13.1%**	**30.8%**

Each measurement was made by hand by taking five different measurements at different times from the display of the top program. For an unknown reason the new desktop system had lower CPU usage when the Source Router was working at full speed than what it had when idle. A possible reason for this is that the CPU is running at a lower speed when the traffic is idle, and its speed is boosted when there is full traffic.

This renders the results of the CPU usage test in case of DNew incomparable between idle and full speed. A better way to test would have been to put a high enough load on the computer using other software in the background so that the CPU would have been near full load in both idle and full speed tests.

Table 19. TEST 1, CPU usage at full traffic

	DNew	**LNew**	**DOld**	**LOld**
Meas. 1	7.7%	15.0%	18.9%	43.7%
Meas. 2	8.0%	15.2%	19.3%	44.4%
Meas. 3	8.3%	14.6%	18.9%	44.0%
Meas. 4	7.3%	14.3%	17.9%	44.2%
Meas. 5	7.6%	15.0%	19.2%	44.4%
Average	**7.8%**	**14.8%**	**18.8%**	**44.1%**

6.2 TEST 2: More thorough feature testing

It was deemed unnecessary to test all client slots thoroughly as the software will function the same for any of the 16 clients. The only difference between clients is the initial IP address comparison which will determine which client number or slot in the table is to be used.

When the client number is found out from the IP address table all the same programs functions are run with a different table index, so that if everything works for client 1, everything works for any other client as well. Therefore, the thorough feature testing was only carried out for client slot 1. The functioning of all the other client slots was verified by only a brief test with no recorded results.

6.2.1 Packet loss test

For packet loss testing the better method from the previous tests was used. This was to use the iperf software which by default sends roughly 900 packets each way between the server and client in a 10 second interval to perform the test and records the number of lost packets to provide the packet loss measurement. The test results for the test are presented in table 20 and show that the results are within 2 percentage points of the setting value. The commands used in the test are below:

- server command: iperf-2.0.14a-win.exe -s -u -i 1
- client command: iperf-2.0.14a-win.exe -c 192.168.0.16 -u -d

Table 20. TEST 2, packet loss

	Server send	Client send
No loss	0.0%	0.0%
1% loss	1.2%	1.0%
2% loss	2.1%	2.6%
3% loss	3.0%	3.2%
4% loss	4.4%	3.5%
5% loss	5.6%	4.5%
6% loss	5.7%	6.6%
8% loss	8.8%	8.7%
10% loss	10%	10%
15% loss	15%	16%
20% loss	21%	19%
25% loss	25%	23%
30% loss	28%	29%
35% loss	34%	34%
40% loss	40%	42%
45% loss	45%	45%
50% loss	49%	50%
60% loss	61%	59%
70% loss	70%	71%
80% loss	78%	80%

6.2.2 Base delay test

The base delay test was carried out by the ping command. This time to establish a more robust set of measurements 60 pings were sent between the client and the server and the minimum, maximum and average results were recorded. The measurements can be seen in table 21. The results for very low delay values are inaccurate as the method shows the total delay between client and server with other network components adding a small random delay in the results. But other than that, the delay values are very much as expected. The command used in the test was:

- ping 192.168.0.1 -n 60

Table 21. TEST 2, base delay

	Server send			**Client send**		
	Min	**Max**	**Avg**	**Min**	**Max**	**Avg**
0 ms	1	7	**1**	1	7	**1**
1 ms	2	7	**2**	3	5	**3**
2 ms	3	5	**3**	3	5	**3**
3 ms	4	6	**5**	5	7	**5**
4 ms	5	7	**5**	5	11	**5**
5 ms	6	12	**6**	6	9	**7**
10 ms	11	17	**12**	11	18	**12**
20 ms	21	23	**21**	22	24	**22**
30 ms	31	34	**32**	31	41	**32**
40 ms	42	50	**42**	41	47	**42**
50 ms	52	56	**52**	51	57	**52**
100 ms	101	105	**102**	101	104	**102**
200 ms	202	204	**202**	201	209	**202**
300 ms	302	305	**302**	301	304	**302**
400 ms	401	404	**402**	400	404	**402**
500 ms	501	507	**502**	501	504	**502**
750 ms	751	754	**751**	751	754	**752**
1000 ms	1001	1003	**1001**	1001	1004	**1002**

6.2.3 Jitter test

The jitter test was made the same way than the base delay test above except this time 100 pings were used to better include the changing jitter values in the results. The results are shown in table 22. Even longer tests might have given even more consistent values as in the highest jitter values a difference between the results can be seen. In the lower jitter values the consistency of the results is much higher which shows that the jitter feature is working as expected. The lowest values show almost no difference due to the same reason as in the base delay test where the varying delays in other network equipment adds noise in the values. The command used in the test was:

- ping 192.168.0.1 -n 100

Table 22. TEST 2, delay jitter

	Server send			**Client send**		
	Min	**Max**	**Avg**	**Min**	**Max**	**Avg**
0 ms	1	4	**2**	1	4	**2**
1 ms	1	6	**2**	2	5	**2**
2 ms	1	5	**2**	1	9	**3**
3 ms	1	7	**2**	2	6	**3**
4 ms	2	8	**3**	2	7	**3**
5 ms	2	10	**4**	1	7	**3**
10 ms	2	11	**5**	1	10	**4**
20 ms	1	22	**6**	1	17	**7**
30 ms	1	29	**9**	1	23	**9**
40 ms	2	33	**11**	1	29	**12**
50 ms	2	36	**13**	1	41	**14**
100 ms	1	68	**26**	1	66	**26**
200 ms	2	169	**50**	3	138	**46**
300 ms	3	193	**68**	2	256	**70**
400 ms	5	283	**93**	4	338	**94**
500 ms	4	354	**108**	3	294	**128**
750 ms	8	652	**181**	9	553	**176**
1000 ms	7	721	**274**	4	750	**235**

6.2.4 Bandwidth test

The bandwidth test was made with the most consistent bandwidth test from the four-computer test which was to use the iperf software in LAN. The results are very consistent between the server and client send sides and are shown in table 23. There is a small difference between the bandwidth setting and the actual result. This is likely caused by the fact that the bandwidth setting means the raw data speed including the network frame headers whereas the results only show the payload data speed and exclude the frame headers. The following commands were used in the test, where BW means the tested bandwidth:

- server command: iperf-2.0.14a-win.exe -s -u -i 1
- client command: iperf-2.0.14a-win.exe -c 192.168.0.16 -u -d -b BW

Table 23. TEST 2, bandwidth

	Server send	**Client send**
0.1 Mbps	0.104 Mbps	0.098 Mbps
0.2 Mbps	0.197 Mbps	0.201 Mbps
0.5 Mbps	0.491 Mbps	0.492 Mbps
1 Mbps	0.964 Mbps	0.939 Mbps
2 Mbps	1.96 Mbps	1.97 Mbps
4 Mbps	3.91 Mbps	3.91 Mbps
6 Mbps	5.84 Mbps	5.87 Mbps
8 Mbps	7.79 Mbps	7.75 Mbps
10 Mbps	9.74 Mbps	9.79 Mbps
15 Mbps	14.6 Mbps	14.6 Mbps
20 Mbps	19.5 Mbps	19.5 Mbps
25 Mbps	24.4 Mbps	24.4 Mbps
30 Mbps	29.3 Mbps	29.3 Mbps
40 Mbps	38.9 Mbps	38.9 Mbps
50 Mbps	48.7 Mbps	48.7 Mbps
60 Mbps	58.4 Mbps	58.4 Mbps
70 Mbps	68.0 Mbps	68.1 Mbps
80 Mbps	77.8 Mbps	77.8 Mbps
90 Mbps	87.5 Mbps	87.5 Mbps
100 Mbps	87.1 Mbps	93.7 Mbps

6.3 TEST 3: Online gaming test

A brief one-hour test was made where two players were playing the Fortnite battle royale FPS game through the created Source Router software and during the gaming the individual features of the software were tested to see their effect on the gameplay. The subjective view of the two players was that they could not recognize whether the Source Router was used to bridge the game network connection or not when all Source Router interference was turned off.

During the gameplay the gaming computers were also individually subjected to various lag situations manually controlled from the Source Router web UI. Fortnite has a built-in network monitor mode that shows the delay, packet loss and amount of transferred data separately for incoming and outgoing traffic. When the delay, jitter and packet loss were changed from the Source Router software the display of the built-in statistics did change accordingly. Limiting the game bandwidth did make the game lag, but only when the bandwidth was limited to less than 300 kbps both ways.

No accurate measurements were being able to be carried out since the network statistics are only shown on the screen and not logged by the game, but it can be estimated that the accuracy of delay was in the range of few milliseconds and packet loss within 1–3%. While the results of this gaming test are not statistically enough to make big conclusions, they at least point into the direction that the Source Router software would not appear to affect the gameplay detrimentally when the game connection passes through it and interference is not enabled. Furthermore, when interference is enabled, the results appear to be as expected.

7 CONCLUSIONS

This thesis was designed to find out how a LaggingMeter system could be implemented on a modern PC system and how modern and efficient such a PC system would need to be. As a result, a working LaggingMeter system was successfully implemented by using modern PC hardware. The system can be used to make lagging tests as proposed by Mr. Kettunen in his paper. The LaggingMeter was implemented mostly in software with only the lag presence

input from clients requiring hardware in the form of a standard push button. This push button is to be connected to a standard Raspberry Pi miniature computer.

However, no special hardware is required to implement the system. Therefore, it has been shown that when a modern PC hardware is used, the introduction of specific lag to a client's network connection can be completely achieved by a software only system which can be used to manipulate four Quality of Service parameters independently in both data transfer directions.

The created system allows the manipulation of four important network parameters: delay, delay jitter, packet loss and bandwidth. Each of these parameters were tested separately to find out that they work in the implemented system as expected. The functioning of these parameters was also verified in a small-scale gaming test.

The reliability of the test results from the parameter tests is not perfect, partly due to the used test methods and partly because there were other network devices in the network causing variation in the measured parameters. However, the measured parameters are very close to their setting values in all the tests. Also, because the system relies on software to do the packet bridging, a situation may arise where the PC is starved of resources and this might influence the working of the system. No such situation was, however, experienced during the creating and testing of the system.

Four different PC systems were tested with the implemented LaggingMeter. What comes to the hardware requirements of the used PC system the minimum requirements for running the system were not able to be established in the tests. The system test results show that even the older PC systems that were used in testing the LaggingMeter were able to complete the tests with equal results to the newer PC hardware when the connection speed was 100 Mbps. However, it was established that even a 5-year-old medium range PC system is likely to run the LaggingMeter system without a problem.

The setting accuracy of the delays in the microsecond level is more than enough as many networks measure delays in the range of milliseconds. Also,

the accuracy of 1 kbps for the bandwidth is very accurate when most networks work in the Gbps range. However, since even a small packet loss can be very detrimental to online gaming, the accuracy of the packet loss parameter could have been at least ten times more accurate to better differentiate different amounts of packet loss.

The LaggingMeter uses a web browser user interface for parameter entry and status display. Creating the user interface with JavaScript and WebSocket technologies was a learning experience and provided an insight into how bidirectional data traffic can be implemented with modern web browser so that the user can both input data from the browser as well as see the system status information in return.

While the implemented LaggingMeter system is fully functional its usage relies mostly in manual operation. It is possible to run the system with automatic sequences where the interference parameters change on each client in the same way, but at different points in time. This type of automation is, however, somewhat limited. As a further development of the system, a more advanced automation system could be created, possibly by the usage of a scripting language that would allow each client to run different kind of interference.

Another development idea would be to try to model the jitter functionality in a more realistic way. The study on jitter modeling presented in the thesis could serve as a starting point in this work. On the other hand, it may be questioned whether this is necessary as the implemented jitter model may be enough to the purposes of the system. A jitter study could be carried out using the jitter of the LaggingMeter system and comparing it to the jitter generated by a real Internet game server.

Even though there is no hard limit in the system to prevent 1 Gbps operation, since the acquired USB network adapters supported only 100 Mbps speed, no real-life tests with 1 Gbps speed were made. Therefore, testing the system with 1 Gbps speed would give information how well it can cope at that speed and whether the system can even reach full 1 Gbps speed with the imple-

mented Source Router software. Also, making such tests with multiple different PC systems running the Source Router software would reveal whether older systems can support such a fast network speed.

The LaggingMeter system appears to run fine on older PC systems that have less processing power and utilizes a Raspberry Pi miniature computer for lag indication collection. This brings up the question whether a Raspberry Pi could be used for both tasks. This would be another way to further develop the system allowing everything to be put on one Raspberry that could be used as a portable LaggingMeter system that would be easy to install anywhere.

Even as is, the system can perform the task it was built for, namely creating individual lag per player and getting the feedback from the players via a push button. Every system can and should, however, be constantly improved. Therefore, it is advised to run the system as implemented while studying its behavior and finding out what works and what could be improved. With the new findings about the system and the suggestions above the system can be improved further.

In conclusion, it was verified that a working LaggingMeter system can be built using one modern PC system with two network adapters and a dedicated bridging software. To collect the lag experience data from the users, an external Raspberry Pi system can be used. The used PC system can be a few years old mid-range system when a connection speed of 100 Mbps is used.

REFERENCES

Barreiros, M. & Lundqvist, P. 2016. QOS-Enabled Networks. 2nd edition. Chichester: John Wiley & Sons, Ltd.

Daniel, E., White, C. & Teague, K. 2003. An interarrival delay jitter model using multistructure network delay characteristics for packet networks. Available: https://www.researchgate.net/publication/4071919_An_interarrival_delay_jitter_model_using_multistructure_network_delay_characteristics_for_packet_networks [referenced 22.10.2019].

Evans, J. & Filsfils, C. 2007. Deploying IP and MPLS QOS for Multiservice Networks. San Francisco: Morgan Kaufmann.

Horowitz, P. & Hill, W. 2015. The Art of Electronics. 3rd edition. Cambridge: Cambridge University Press.

Jha, S. & Hassan, M. 2002. Engineering Internet QoS. Norwood: Artech House Inc.

Kananen, J. 2014. Verkkotutkimus opinnäytetyönä. Jyväskylä: JAMK University of Applied Sciences.

Kananen, J. 2017. Kehittämistutkimus interventiotutkimuksen muotona. Jyväskylä: JAMK University of Applied Sciences.

Kettunen, M. 2012. LaggingMeter. PDF-document. Kotka: Kymenlaakso University of Applied Sciences.

Kurose, J. & Ross, K. 2013. Computer Networking. 6th edition. London: Pearson Education Ltd.

Lee, W-K. & Chang, R. 2015. Evaluation of Lag-Related Configurations in First-Person Shooter Games. PDF-document. Available: https://www.researchgate.net/publication/304293798_Evaluation_of_lag-related_configurations_in_first-person_shooter_games [referenced: 22.10.2019].

Marsaglia, G. 2003. Xorshift RNGs. PDF-document. Available: https://www.jstatsoft.org/index.php/jss/article/view/v008i14/xorshift.pdf [referenced 13.9.2019].

Misra, S. & Goswami, S. 2017. Network Routing. Chichester: John Wiley & Sons, Ltd.

Peterson, L. & Davie, B. 2012. Computer Networks. Burlington: Morgan Kaufmann.

Saarelainen, K. 2011. IP-puhe. Helsinki: Readme.fi.

Tseng, P-H., Wang, N-C., Lin, R-M. & Chen, K-T. 2011. On the Battle Between Lag And Online Gamers. PDF-document. Available: https://www.researchgate.net/publication/224255033_On_the_battle_between_lag_and_online_gamers [referenced: 22.10.2019].

Vegesna, S. 2001. IP Quality of Service. Indianapolis: Cisco Press.

White, R. & Banks, E. 2018. Computer Networking Problems and Solutions. Boston: Addison-Wesley.

LIST OF FIGURES

LIST OF TABLES