Behaviour of WebRTC in Non-optimal Networks

Simon Johansson

Computer Science and Engineering, master's level
2018

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering
Behaviour of WebRTC in Non-optimal Networks

Simon Johansson

Luleå University of Technology
Dept. of Computer Science, Electrical and Space Engineering

22nd February 2018
The behaviour of WebRTC when the real-time communication with audio is done over a non-optimal network is investigated in this thesis. Different methods for collecting and analyzing data from an online survey are considered.

A test environment was developed from which two online surveys would be conducted, where the outgoing packets had various interferences applied to them by the server. This was made in order to be able to simulate a non-optimal network e.g. WiFi. The participants are told to listen to various audio sequences and are asked to rate the quality as they perceive it.

Although considered, video was not used in the surveys, as it would have increased the complexity of the surveys and increasing the risk of having the participants rejecting the surveys.

Two independent surveys were conducted. The first survey utilized WebRTC for sending the audio, this was compared to the second survey which instead used Icecast.

The result showed that WebRTC behaves well when there was only one type of interference added. Compared to Icecast it had lower performance. However, this could be contributed to the fact that two independent groups were used and the surveys had low participation rates.

The surveys proved the feasibility of conducting online surveys for measuring perceived quality, although the participation rate was extremely low (2.8%), something that has to be considered when performing online surveys.
Firstly, I would like to thank Prof. Peter Parnes at Luleå University of Technology for his help during this master thesis. Secondly, I want to thank Neava for helping me with my master thesis and to Staffan Johansson for all the ideas when designing the survey. And lastly, I want to thank my family and friends that have supported me during this project.

Simon Johansson
CONTENTS

CHAPTER 1 – INTRODUCTION  1
  1.1 Introduction .......................... 1
  1.2 Goal ............................... 2
  1.3 Delimitations ........................ 2
  1.4 Thesis structure ....................... 3

CHAPTER 2 – BACKGROUND  5
  2.1 Before WebRTC ........................ 5
  2.2 WebRTC ................................ 5
    2.2.1 Mediastream ....................... 6
    2.2.2 RTCPeerConnection ................. 6
    2.2.3 RTCDataChannel .................... 6
    2.2.4 Signaling ........................... 6
  2.3 Peerjs ................................ 6
  2.4 Icecast ............................... 7
  2.5 TC ................................... 7
  2.6 Opus .................................. 7
  2.7 Related work .......................... 9

CHAPTER 3 – THEORY  11
  3.1 Overview ................................ 11
  3.2 Mean opinion score ..................... 11
  3.3 Perceived Quality Survey ............... 12
  3.4 Interference ............................ 14

CHAPTER 4 – METHODOLOGY  15
  4.1 Overview ................................ 15
  4.2 Pre-study ................................ 16
  4.3 Hardware and Software .................. 16
    4.3.1 Streaming server .................... 17
    4.3.2 Interference ........................ 18
  4.4 Pre-test ................................ 19
  4.5 Survey .................................. 19
**Chapter 5 – Evaluation**

5.1 Pre-test ................................................................. 21
5.2 Test ................................................................. 23
5.3 Result ................................................................. 23

**Chapter 6 – Discussion**

6.1 Goal ................................................................. 30
6.2 Future work .......................................................... 30
  6.2.1 Tests ............................................................. 31
  6.2.2 Interference ..................................................... 31
  6.2.3 Participants .................................................... 31
  6.2.4 Audio .......................................................... 32
  6.2.5 Video .......................................................... 32
  6.2.6 Test environment ............................................ 32
6.3 Conclusions ......................................................... 33

**Appendix A –**

A.1 Requirement specification ........................................ 35
A.2 Tables ............................................................... 36
A.3 TC Commands ...................................................... 37
  1.3.1 Pre-test commands ............................................. 37
  1.3.2 Test commands ................................................ 38
CHAPTER 1

Introduction

1.1 Introduction

With its rise in popularity in the early 90s, the Internet has significantly changed the way we consume and distribute media. It allowed real-time communication between computers and with the introduction of the web browser, it simplified the process for sending and receiving information. This set a great demand for new ways of communication between computers.

Direct communication between two web browsers could be achieved with plugins\[1\], third-party programs, which allowed the browser to communicate directly with another browser without the need for a server to relay the information. However, plugins introduce security issues and were mostly incompatible with different browsers. Another problem is the prevalence of firewalls and NAT that limits the access of Internet traffic to computers in networks where such systems are deployed.

This created pressure for the development of new standards for real-time communication, one of them being WebRTC (Web Real-Time Communication) which is being investigating in this thesis. WebRTC is a collection of protocols and APIs first developed at Google and then subsequently released as an open source project back in 2011. It gives a set of standardized methods for sending audio, visuals and arbitrary data between computers in real-time. This removes the need for plugins and thus both simplifies the communication and development of web applications.
The success of a new standard is determined by its ability to simplify development and improve user experience. To achieve this researchers have to perform rigorous testing to assure the quality. There exist many different ways of testing, ranging from automated tests using algorithms to determine the quality objectively, to manual participation of end users. In this thesis I focused on allowing university students to participate in a test method called Perceived Quality Survey (PQS). It gathers information about the end users experience by letting them be subjects either to a number of listening tests or watching a number of sequences being played, followed up by telling them to vote on the quality as they perceived it.

1.2 Goal

The goal of this thesis is to investigate what kind of behaviour WebRTC has when the real-time communication is done over a network with non-optimal performance e.g. wireless networks. This is done by gathering students and letting them participate in two independent online Perceived Quality Surveys where the test is done with respect to induced packet loss and delay.

1.3 Delimitations

A big part of the project is the gathering of raw data from the participants. This puts a lot of emphasis on how the survey is to be constructed. It is crucial to keep the survey as simple as possible to encourage the students that are participating to finish the survey. With that in mind, even though WebRTC is fully capable of both sending audio and visual information between the peers, it was decided early on that in this thesis only audio would be considered for evaluation. The reason being that video is evaluated differently than audio and would increase the complexity of the surveys.

Although having a large range of different kinds of sounds, ranging from pop to rock music, would contribute more to the overall picture, only one music sample was used in the thesis. Due to the implementation the only audio format that was available for testing was the .Wav-format, which has a low bit-rate, that could potentially have an effect on the participants regarding their perceived quality.

In order to perform the surveys, there is a need to be able to produce a variety of different interferences to simulate, for instance, a wireless connection. Writing the interferences from the bottom up would be preferred, as it would yield better control and be tailored for the survey, this was deemed out of scope. Instead it was decided to use already existing software.
Only two different types of interference were used in the surveys. The reason being that there is a limitation to what the software for creating the interferences is able to do, but also to reduce the complexity of the test-cases. It was decided to use packet loss and packet delay, since they are able to create gaps[2] in the audio stream which will affect the perceived quality.

As the thesis focuses on the behaviour of WebRTC, no investigation is made on how WebRTC handles the added interference.

The participants in the test groups were mainly university students. When performing a perceived quality survey it is important to have a wide audience to be able to get a complete picture, but in this thesis only students are considered as it is easier to contact and to organize them.

The two surveys were conducted by using two independent groups of students. This was done because it was unknown who participated in the first survey.

1.4 Thesis structure

Chapter 2 gives details about the background for the surveys. Chapter 3 describes the theories that are being used in this thesis. Chapter 4 shows the general steps taken, from the planning all the way up to the implementation of the system and how everything is connected. Chapter 5 contains both the final test and result, including a pre-test that was made. Chapter 6 talks about potential future work and a final conclusion of the result in this thesis.
This chapter describes the background for this thesis. It starts off with a brief explanation of how real-time communication was achieved before and after WebRTC was introduced. Followed up is a short description of some of the technologies that are being used in this thesis. Lastly, there is a presentation over related works that have already been done.

2.1 Before WebRTC

If you wanted to be able to send real-time media over the Internet via a web browser before WebRTC was developed, it would take considerably more development time as most of the things that WebRTC provides had to be implemented. As such most of the solutions that existed were proprietary and originated from big corporations such as Google, Microsoft or Adobe with their products, Google Hangouts, Skype and Flash Player respectively. All of them are able to capture and send audiovisual information between the clients, although they need a browser plugin in order to function.

2.2 WebRTC

The most common setup for web application is the client-server model, where a number of computers called clients connect to one main server. This puts a great deal of pressure on the server since it has to be able to provide everything from setting up the connection to relaying data between the clients.

WebRTC removes the need for a server to be able to relay the information between the clients and instead utilizes the peer-to-peer concept, where everything apart from signaling (how the connection should be established) is handled between the clients. To be able to enable this setup, WebRTC comes with three main components that take care of everything from gathering the audio or video from physical devices to sending
audiovisual data: MediaStream, RTCPeerConnection and RTCDataChannel. These are the main objects and they are associated with one stream between two clients.

2.2.1 Mediastream

Each MediaStream[3] object supports one input and one output. This can be used in conjunction with the getUserMedia() function to enable the web browser to access either the microphone or the web camera to be used as input.

2.2.2 RTCPeerConnection

For every connection made between two clients, a RTCPeerConnection object is created. These are used for setting up audio or video calls. They contain all the information about the connection and can be used to close the connection.

2.2.3 RTCDataChannel

The RTCDataChannel is mainly used for sending arbitrary data between peers.

2.2.4 Signaling

One key aspect to notice is that WebRTC does not define how the signaling should be carried out. The reasoning behind it is explained by JSEP[4]

“The rationale is that different applications may prefer to use different protocols, such as the existing SIP or Jingle call signaling protocols, or something custom to the particular application, perhaps for a novel use case.”

2.3 Peerjs

Peerjs[5] is a Javascript API used for simplifying the development of websites using WebRTC by abstracting away browser differences. This allows the developer to focus more of their attention on the design and less on more specific implementation details.

It is being used as the main tool for the website, as it provides functions for connecting together the clients with the server, sending audio data to the clients and collecting various data from the clients.

For initializing a connection between two peers Peerjs comes with a signaling server, a broker that provides unique tokens for each peer, that is later shared between the clients to enable a connection between them.

To circumvent the problem when any of the clients is behind a NAT, the broker utilizes a STUN[6] server. The main purpose of a STUN server is to give the clients their public
2.4. Icecast

When conducting the survey for WebRTC there was a need to compare it to something which relies on some other technology, this is to get a clearer picture of the behaviour WebRTC might have. It was decided to use Icecast[8], which is an open-source streaming software which relies on the client-server model for sending audio.

2.5 TC

The surveys involve different tests where different interferences are applied, there is a need to be able to produce network related errors such as packet loss or packet delay. This is done by utilizing the TC (Traffic Control) utility program in Linux. TC is part of the Linux ecosystem and can be used to manipulate the Linux kernel packet scheduler. It does this by analyzing the packets that enter the network interface card and scans the header fields for each incoming packet e.g. source and destination ip-addresses. The packets are then classified and then placed in various queues before being sent off.

With the use of a qdisc (queueing discipline) the packets can be manipulated in different ways e.g. introducing bit errors, delaying them or dropping them completely resulting in a packet loss. There are many different types of qdiscs available such as classless or classful. The latter is used since it allows for the creation of classes that can be used when trying to classify and applying appropriate interference on the packets.

2.6 Opus

Opus[9] is a new BSD-licenced codec primarily used in WebRTC for encoding audio data. WebRTC supports many different codecs such as G.711, G.722, iLBC, and iSAC. Opus is mainly used since it is designed from the ground up, having low delays and covers a large span of different audio frequencies.

Opus is able to cover such large audio band due to the fact that it is a combination of two different codecs, each with its own specialty, SILK[10] and CELT[10]. Either one can be used individually depending on the situation but they can also be used simultaneously giving a great deal of flexibility if the network conditions were to change[10].
The SILK codec was originally developed for use in Skype but was later modified to be more compatible to WebRTC’s requirements. It handles all audio which resides within the human voice spectrum. It does this by using Linear Prediction Coding[11].

The CELT codec is contemporary to SILK as it handles everything above the voice band, such as music. Instead of being based on Linear Prediction, it uses Modified Discrete Cosine Transform[12].

![Figure 2.1: Comparison between the different codecs](image)

![Figure 2.2: Overview of the opus codec](image)
2.7 Related work

With its initial release in 2011, a lot of research has been done on WebRTC, trying to determine its behaviour and performance in various scenarios. There have been many different approaches to this, either with people participating or running the audio through algorithms to try to approximate the perceived quality. Another way is to combine the two, each with its own strengths and weaknesses. However, they have mainly been conducted in a controlled test environment. This is in contrast to this thesis where we are subjecting the participants with an array of different tests enclosed in a survey, in a public environment. Nevertheless, they showed many important considerations that have to be addressed when designing surveys and their associated test environments.

In [13], they tackle the problem of trying to automate testing for WebRTC, due to browser and website diversity. A testing framework is presented, that with the help of Google Chrome's built-in flags, is able to circumvent both the microphone and web camera and instead use a test file that can be fed to the mediastream. This allowed them to create a instance of Chrome that would broadcast out the media file to a number of clients. When testing the system, the result showed that WebRTC was able to sustain up to 180 concurrent clients. Anything above that threshold the latency experienced increased dramatically rendering the real-time experience to a halt. They did however not invite any participants to perform these test over a public network nor did they test any perceived quality. Instead the system that was developed launched multiple instances of Google Chrome to act as participating clients. A solution to the problem of having multiple instances of Chrome was to deploy fake browsers, containing the WebRTC stack, as they provide a smaller performance hit, thus enabling more clients to be employed at the same time.

In a 2016 research paper[14], they investigate the opus codec in an attempt to, with the use of a combination of two different methods, measure the perceived quality objectively. This is done by using the POLQA (Perceptual Objective Listening Quality Assessment) algorithm and the non-standardized AQuA (Audio Quality Analyzer) algorithm. As the opus codec is integrated into WebRTC for sending audio information, WebRTC was utilized for creating the connections to the clients and sending the information. After running the opus encoded audio-file over a WebRTC enabled connection, the result from the decoded file was then firstly evaluated by human participants and then evaluated by the algorithms. That is, they compared the audio file prior to encoding it with opus and sending it through the network with the same audio file that had been sent and decoded on the other side. As such, the perceived quality was not done in a real-time setting, as the human participants were only listening to an audio file that had already been sent over the network. It showed however that POLQA could have been potentially used in this thesis as it yields a MOS score that could be compared to the
MOS score produced by the students participating in the survey.

While it might be tempting to only use algorithms when trying to find the perceived quality, as it removes the need to involve any human participants, speeding up testing as contacting and setting up a test environment is not necessary. This does however come with its own dangers as described in the research paper [15]. In the paper they used an algorithm called PESQ (Perceptual Evaluation of Speech Quality) that takes an audio file as an input. To do this they created a test environment where they used one computer that sends an audio file over the network to itself. This was done by using WebRTC and the Web Audio API. To be able to produce interference they used a network simulator, similar to TC, called Dummynet. They only applied one type of interference, a delay with an added jitter. Both the original audio file and the decoded were fed into the PESQ algorithm, producing a MOS score. However, due to the setup of the test environment the MOS score produces were lower than expected. They stated that for future work, instead of just using one computer, a better topology would be to have two computers communicating.

In [16] they implemented a test environment were they tried to test the performance and behaviour of WebRTC when the connection between two different smartphones were made over a LTE network. They did this by using the NS-3 software, a discrete-event network simulator, that is capable of creating a simulation of a LTE network that can be used for testing. They tested this by creating 4 different scenarios where they were applying different and for each scenario increasing levels of interferences to the packets. These interferences range from packet delay with a jitter to packet loss. In this paper they did not have any human participating while conducting the different scenarios and everything was tried out in a controlled environment. Instead of making a perceived quality survey they simply only measured the performance of WebRTC while over a LTE network.
This chapter describes the different theories used for conducting the surveys, starting with a brief overview. This is followed up by describing the theory behind the scales used for rating the quality of the audio as the participants perceive it. Then a description on perceived quality surveys is presented. Lastly, the interferences used in this thesis are described.

### 3.1 Overview

Conducting surveys is a way to enable us to get a first-hand look on how various things might behave when deployed in different scenarios. It shows us areas where improvements must be made in order to guarantee a more satisfying experience and if the theory or product that might be tested is feasible in such conditions. When performing the surveys, the data produced can be analyzed in different ways. For this thesis, MOS (Mean Opinion Score) was chosen, as it is commonly used in these kinds of surveys. MOS can either be acquired subjectively by the use of human participants or objectively by using different algorithms.

### 3.2 Mean opinion score

When performing the surveys, all the students participating were given a choice after listening to one of the tests. Based on the quality of the audio, they were asked to rate it according to the Absolute Category Rating (ACR). It spans between 1 to 5, where 1 represents the worst quality and 5 the best quality possible. It is recommended to use by the ITU-T[17], table 3.1 shows the different ratings. When the surveys were done, the
score was collected and analyzed using MOS that uses this formula:

\[ \sum_{n=0}^{N} \frac{R_n}{N} \]

where \( R \) is the individual rating for each corresponding test and \( N \) is the total number of participants. The result is an average quality score for the corresponding test.

<table>
<thead>
<tr>
<th>Quality</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Excellent</td>
<td>5</td>
</tr>
<tr>
<td>Good</td>
<td>4</td>
</tr>
<tr>
<td>Fair</td>
<td>3</td>
</tr>
<tr>
<td>Poor</td>
<td>2</td>
</tr>
<tr>
<td>Bad</td>
<td>1</td>
</tr>
</tbody>
</table>

*Table 3.1: Quality rating*

One could change the number of levels used with ACR, but this is discouraged. Having a larger range only results in the participants unable to differentiate the differences between the levels[18]. For example, what differs between a score of 74 and 81? Also the differences between lower scores have a larger effect than between scores that have higher values. For most people the quality jump between 4 and 5 is harder to quantify than a jump between 2 and 3. One very important consideration that has to be taken into account when using MOS, is the context of which the data was collected from. This limits the way of which the MOS score can be compared to other experiments.

### 3.3 Perceived Quality Survey

The main part of the thesis are the surveys that would be conducted and with it comes a few considerations. A perceived quality survey can be conducted in two different kinds of environments, either a controlled or public one[18]. In a controlled environment, everything within the room is carefully chosen, from the type of lighting used to the color of the walls. All this to minimize any potential distraction for the participants that might affect their perception. Performing the surveys in this kind of manner gives a great deal of control. This however assumes an ideal situation and might not reflect an ordinary, day-to-day experience. As such it was decided that the surveys would be conducted in a public environment and to achieve this it would be performed online. With this comes a few benefits:
3.3. Perceived Quality Survey

- Management: by performing the surveys online many hurdles are removed such as contacting and trying to decide on a date and time for the survey. This allows the participants to conduct the survey when it suits them.

- Accessibility: by only having to visit a website, removes the need to prepare and organize a proper test environment where the surveys will be performed at.

One important consideration is the duration of each test that the participants will perform. In [18] they recommend that the duration should be somewhere between 5 to 20 seconds. The reasoning for this is that the subject listening to the audio might not be able to remember all the changes in quality, e.g., a drop in quality in the beginning might be overshadowed by changes that occur near the end.

When it comes to the subjects that are to participate in the surveys, a few important aspects need to be dealt with. The number of participants depends on the type of survey that is to be conducted. When dealing with surveys being performed in a controlled environment, [18] recommends that at least 24 subjects are needed. If instead in a public environment, a slightly larger pool is needed, somewhere around 35 subjects. Also, the type of people chosen plays a part in the outcome of the survey e.g., the choice of music being played in the survey could influence their decision.

One important note before starting the surveys is to conduct a pre-screening for any potential impairments that the participants might have, as it will affect the perceived quality[18]. This is not being taken into consideration in this thesis, as the surveys are done online. However, before they can start the surveys, all the participants have to do a small training run, where they will be listening to the audio sample for 20 seconds. This is so that they can be familiar with the sound and to adjust audio volume and such.
3.4 Interference

When sending packets over the Internet, different interferences might occur to the packet. The reason being that the packets are being delivered over a vast number of different networks, all with different network topologies. The current throughput of the network could change at any time, as the change in traffic can be the result of software or hardware failure. A fiber cable being cut could potentially result in an otherwise low traffic network suddenly experiencing overwhelming traffic that will slow down the whole network. For each network that the packet has to go through the risk of it being stuck increases, as such interference is the result of such events. Another source of interference can also be contributed to the medium of which the packets are being sent through. Wireless transmission, sent from either a wireless router that is in close proximity or from a LTE antenna, is at risk of potentially being distorted by various sources such as surrounding electronics or electromagnetic radiation picked up by the antenna. However, when conducting the perceived quality survey in this thesis, there was a need to be able to force a specific kind of interference onto the packets that were to be sent. Although there exist many different kinds of interferences that might occur, the two that were chosen were packet loss and packet delay.

To counter the problem of packets carrying the needed information, for reconstructing the frames in an audio file, getting lost or delayed along the way, today’s codecs come with various ways of handling interferences. They employ a technique called Packet Loss Concealment. It is being done by introducing fake frames that try to mimic the current audio that is being played. Depending on the frequency at which the packets get lost or how long of a delay the packets have, these measures can result in the perception of information loss not being noticeable. However, this only works if there were to be only a few packets being affected. If there would be a number of packets consecutively being lost or delayed, the quality would not be able to be maintained.
This chapter deals with the methodology, the steps taken and choices made, when this thesis was performed. It starts with describing the pre-study that was made followed by the choice of hardware and software and ending with the surveys that were conducted.

4.1 Overview

A lot of research has been conducted trying to figure out how to design and implement surveys in the best possible way, as there are a number of ways one can achieve this. As such, a pre-study was conducted in order to gather information on how the surveys should be performed so that they could produce data that was relevant for this thesis. First and foremost, it was to figure out how the surveys should look like. This led to many questions that needed to be answered before any progress could be made in the thesis:

1. Should the surveys be done online or on site?
2. What are the hardware requirements?
3. What software is needed?
4. How many different test would be required?
5. What kind of interference shall be used?
6. How should the data be evaluated?
4.2 Pre-study

Since having no prior experience with conducting a project containing perceived quality surveys, there was a need to, before starting the implementation phase, to gather essential information that might be necessary. What different kinds of insight could be found from surveys that have already been conducted?

It turns out that a lot of different kinds of studies have been conducted, each and everyone with its own set of requirements and setup. There also exist a number of different frameworks that can be used for conducting tests. One that was considered for potential use was Licode, a framework based on WebRTC. It comes with a set of APIs that allows for creating so called rooms that various clients could connect to and participate in. While looking promising, it was decided not to use it. The reason being that it did not have a feature to specify an audio-file to be used. On top of that there were problems to get it functioning properly, as different dependencies were unavailable as they had become deprecated. Instead Peerjs was selected, as prior experience with it has shown that it is a capable API for creating WebRTC enabled solutions.

The pre-study also revealed multiple methods for how the surveys should be conducted and how one should rate the perceived quality. For rating, the different scales ranged from either single digit scores, 1 to 10, to a continuous scale between 1 to 100. There are also different ways of presenting the audio sequence. One can either only show the sequence once, as with ACR, or one can show the sequence twice with a small delay between them, according to DSIS. The first sequence is a reference and the second sequence is being distorted in various ways. This allows the participants to get a better perception of the quality between them. ACR was ultimately chosen, as it is easier to implement the test in that manner.

4.3 Hardware and Software

For the hardware there were a few requirements. It should be able to host a web server that is able to potentially have multiple concurrent clients connected to it. Depending on the size of the survey, one could spend a fortune on high-end hardware components such as processors with 8 or even 16 cores. However, for this thesis a quad core processor was used, simply for being easily available and relatively cheap. The server needs to be connected to a network with high-speed and high-bandwidth to make sure it is not the bottleneck.

To keep costs at a bare minimum, free software was primarily used. Although, if a particular software was needed, as it would provide us with functionality that otherwise was not present in a contemporary free software, it could be acquired.
4.3.1 Streaming server

One of the requirements for performing the surveys was to get as many participants as possible doing the tests, a cumbersome task if it was to be done locally. Instead a website was developed. This would make it easier to be able to reach a far bigger population.

For hosting the website a server was used, a Quad-Core i5, 8 GB of memory and everything installed on a SSD. The server was connected to a 100 Mbit/s wired connection at LTU. A high bandwidth connection was necessary to make sure the website was able to handle all the participants. The system was run on a Ubuntu LTS 16.04 Server which was needed since it provides the software needed for the interferences. For the web server, Node.js[19] was chosen as it is both lightweight and efficient which utilize the Chrome V8 Javascript engine. Express.js[20] was used as a framework for simplifying the creation of the web application.

Figure 4.1 shows how the system is implemented. It is composed of three components:

1. **WebRTC.js**: The main component that is responsible for sending the audio to the clients by using the Peer.js API. It also collects various WebRTC statistics of the current session.

2. **Test.js**: The component that handles the interferences by applying it to the clients as they are performing the survey. It does this by adding different TC commands depending on which test the client is currently running.
3. **Users.js**: The component that receives and stores the survey results from the clients.

Peer.js is also used for connecting the participants to the streaming server. It does this by connecting to a broker called PeerServer which generates random tokens that can be used by the clients to initialise a connection. One can either run their own PeerServer or use the PeerServer Cloud Service which is provided by the creators, although with the limitation of only being able to sustain 50 concurrent connections. It would be optimal to use the latter as it would remove the need to run additional software. However, it can only provide tokens over an insecure HTTP connection. This is a problem since WebRTC puts a great deal of emphasis on security as it can only be served over a HTTPS connection. Google Chrome does not allow the mixing of secure connections with insecure ones and thus refuses the connection. A SSL certification was provided by Let’s Encrypt[21], it is free to use with the limitation of having to renew the certificate every 90 days, but can be automated.

To be able to send audio data to all the participants, a Google Chrome session was set up to act as the server to which all participants could connect. This is done by letting the server be the first one to connect to the website and be given a special token to indicate it is the server. Every client that connects afterwards will be treated as an ordinary client.

To be able to use a test-file instead of the microphone, Google Chrome comes with a set of flags[13] that can be set on launch. There are two flags that are primarily used.

- `–use-fake-device-for-media-stream`
- `–use-file-for-fake-audio-capture=/path-to/testfile.wav`

The first flag enables us to change the default audio/video input and the second flag specifies a custom audio file that can be used for debugging. This is what allows me to send the music file to all the participants. However, this kind of setup comes with a limitation, that is, the same music file is being broadcasted to all the participants listening, disregarding where they are in the survey. Thus every listener does not get the same experience. This is less of an issue since we are more concerned about the behaviour of WebRTC.

### 4.3.2 Interference

Since there will be a lot of clients connecting to the server, there is a need to be able to single out clients and apply the interferences only on them. TC + Netem do this by filtering out packets depending on their ip-address. Below are two different examples on how TC + Netem can be used.
4.4  Pre-test

- tc qdisc add dev eno1 parent 1:1 handle 10: netem delay 100ms 100ms loss 25%
  25%
- tc filter add dev eno1 parent 1:0 prio 1 protocol ip handle 0x1 u32 match ip dst
  192.168.1.2 flowid 1:1

The first example is for creating a class with a delay of 100ms with a jitter of 100ms and a 25% packet loss. The jitter is defined as a range for which the delay can alternate between, so if a jitter of 100ms is added the resulting effect is a change between [-50,+50] ms. With the packet loss, there is an option to create a burst of packet being lost. This is by defining a percentage for the probability that the current packet will be dropped depending on if the previous packet was dropped.

The second example is for applying a filter to a class, that will result in interference being applied to that specific ip-address. Each class can have multiple filters assigned to it. See appendix A for a extensive list over all the commands used for setting up the different interferences.

When deciding on the amount of interference to apply, one have to consider the fact that in addition to the interference that the server will apply to the packets, the data is being sent over a public network. Consequently, while developing and testing the various levels of severity with the interference, one have to take into account the interference caused by sending the packets through a public network.

4.4  Pre-test

When trying to decide on which type of interference that would be used in the surveys, a couple of questions occurred. Shall the interferences be noticeable or just barely be registered? What kind of interference is more prevalent in networks? For each given interference, one can vary its severity and there are multiple layers that can be used for testing. This brings another question, how many tests are necessary for this thesis? One of the design goals for the surveys were to make it relatively simple. This was chosen to reduce strain and boredom for the participant. Having a larger survey would bring the risk of making the participants reject the survey altogether. More details how the pre-test was conducted in section 5.1.

4.5  Survey

When designing a survey there can be a lot of different question marks regarding what is needed. In cases such as this it can be a good practise to create a requirement specification that states everything needed for a given test. It gives an overview of what needs to be implemented in order to be able to produce a result as specified. In appendix A, there is an example on what it might look like.
CHAPTER 5

Evaluation

This section starts off by describing the pre-test that was initially performed at Neava, followed up by the two surveys where the students were allowed to participate. Lastly, the result from those surveys are evaluated.

5.1 Pre-test

Before starting the surveys, it was decided that a pre-test was needed. The reasoning behind it was to try to figure out the best combination for the different interferences that were going to be used. The two interferences used can be customized at various levels. This ranges from adding a jitter for the delay to deciding the percentage of packets that are to be lost. The problem is to find the ones that make sense in a day-to-day wireless network. The pre-test was done at Neava by letting the employees perform the survey. This was done as it is a lot less time-consuming than trying to gather students.

The pre-test served a couple of reasons:

- To stress-test the system as a whole.
- Sort out the different combinations of interferences, seeing which ones are feasible.
- Filter out potential bugs that might be present.
- Gather potential feedback from the employees.

Before the pre-test was carried out, a series of experiments were done locally in order to find good test-cases. It was found that with packet loss at 75%, the connection could not be maintained with the server. A loss at 70% was the maximum loss at which the connection was sustainable. This was deemed not usable since it would mean the client would have to redo the survey. Table 5.1 shows the interferences used in the pre-test.
To get a better simulation, a jitter of 100ms was added for the delay. To achieve a burst of packets that get lost, each packet loss depends on the previous packet[22]. The pre-test was done with a small test sample (7 people) and it showed that for each incremental of delay with a specific packet loss, the MOS score decreased only slightly, while each incremental of packet loss had a far greater effect on the score. Different effects was also noted, such as the playback rate was slowed down just as the interferences were added, followed by a speedup of the playback rate after the interferences were removed.

<table>
<thead>
<tr>
<th>Test #</th>
<th>Interference</th>
<th>Parameter 1</th>
<th>Parameter 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>Delay + Loss</td>
<td>100 ms</td>
<td>25%</td>
</tr>
<tr>
<td>Test 2</td>
<td>Delay + Loss</td>
<td>250 ms</td>
<td>25%</td>
</tr>
<tr>
<td>Test 3</td>
<td>Delay + Loss</td>
<td>500 ms</td>
<td>25%</td>
</tr>
<tr>
<td>Test 4</td>
<td>Delay + Loss</td>
<td>750 ms</td>
<td>25%</td>
</tr>
<tr>
<td>Test 5</td>
<td>Delay + Loss</td>
<td>100 ms</td>
<td>50%</td>
</tr>
<tr>
<td>Test 6</td>
<td>Delay + Loss</td>
<td>250 ms</td>
<td>50%</td>
</tr>
<tr>
<td>Test 7</td>
<td>Delay + Loss</td>
<td>500 ms</td>
<td>50%</td>
</tr>
<tr>
<td>Test 8</td>
<td>Delay + Loss</td>
<td>750 ms</td>
<td>50%</td>
</tr>
</tbody>
</table>

Table 5.1: Interferences during the pre-test

Welcome to the WebRTC Survey!

This is a perceived quality survey where you will be listening to a music clip with some interference added and then rate the quality. I'm doing this survey in order to see how WebRTC behaves in unreliable networks. This is done by introducing Packet Loss and/or Packet Delay while the music clip is being played.

This survey only works on Google Chrome on a computer.

1. To begin the survey press the "Start" button. This allows you to adjust your volume and get used to the sound.
2. Chrome will ask for permission to use your microphone, press "allow". Otherwise you will not receive any audio from the website. No audio is sent from you to the website.
3. Each test last 20 seconds followed by a score, that spans from 1 (worst) to 5 (best) quality for that particular test.
4. Once every test has been completed, press the "Send Vote" button to submit the results.
5. If the sound suddenly vanishes, either continue with the test and send the result or refresh the webpage and redo the survey. This happens because the WebRTC connection broke due to the interference being added.

Problems or suggestions? Email me at johnl-58@student.tue.nl

Figure 5.1: Website with the survey
5.2 Test

After the pre-test had yielded positive results it was decided to move on to the surveys. The interferences were split up so the individual effect could be gathered, followed up by combining them for the last three tests. One blank test was also added. This was done to get a reference for which the other tests could be compared to. Table 5.2 shows the final combination used for the surveys.

<table>
<thead>
<tr>
<th>Test #</th>
<th>Interference</th>
<th>Parameter 1</th>
<th>Parameter 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>Delay</td>
<td>250 ms</td>
<td>-</td>
</tr>
<tr>
<td>Test 2</td>
<td>Delay</td>
<td>500 ms</td>
<td>-</td>
</tr>
<tr>
<td>Test 3</td>
<td>Loss</td>
<td>-</td>
<td>10%</td>
</tr>
<tr>
<td>Test 4</td>
<td>Loss</td>
<td>-</td>
<td>25%</td>
</tr>
<tr>
<td>Test 5</td>
<td>Nothing</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Test 6</td>
<td>Delay + Loss</td>
<td>100 ms</td>
<td>25%</td>
</tr>
<tr>
<td>Test 7</td>
<td>Delay + Loss</td>
<td>250 ms</td>
<td>25%</td>
</tr>
<tr>
<td>Test 8</td>
<td>Delay + Loss</td>
<td>500 ms</td>
<td>25%</td>
</tr>
</tbody>
</table>

Table 5.2: Interferences during the surveys

With all the survey parameters finalized, it was time to conduct the surveys. For both surveys students were contacted via email and a total of 600 were sent. Facebook was also used for contacting them by posting in a group for LTU students. For the first survey which used WebRTC, a total of 17 students participated over a span of 2 weeks. The second survey where Icecast was used had a total of 18 students participating over a span of 2 weeks. It was suspected that since the survey was done online, the participation rate would be low, as it is easy to forget about the surveys. One solution to this would be to send reminder emails to all the participants. Getting a reminder might have persuaded them into taking part in the survey. Another option was to offer the participants monetary rewards as an incentive.

5.3 Result

The result from the WebRTC survey showed as suspected a lower score for the test where an interference was added, compared to the blank test which had no interference added to it. Figure 5.2 shows the final result from the WebRTC survey.
Figure 5.2: The MOS score for the WebRTC survey

Figure 5.3 shows the result from the Icecast survey. The performance is much more even when compared to the WebRTC survey. However, there might be multiple reasons for this, either it could be contributed to the implementation of the software or the fact that it uses the client-server model instead of peer-to-peer. Another reason might be that two independent groups were used when conducting the surveys.

Figure 5.3: The MOS score for the Icecast survey
5.3. Result

The reason why packets being delayed have similar impact when compared to packets that are lost might be the very nature of real-time communication. When a packet is lost, instead of playing nothing and resulting in a brief period of silence, the codec tries to mask the lack of frames with so called comfort noise. Audio that almost resembles the current audio being played. With each frame needed to be played in sequence, if the delay is big enough it will have the same effect as if the packet would have been lost when propagating through the network.

Another way for codecs to combat packet loss is to use a method called FEC (Forward Error Correction). It determines which packet contains information with higher priority, containing more useful information. Those packets are re-encoded with a lower bit-rate and sent with subsequent packets, so if the original packet was to be lost along the way, the next incoming packets contain the lost packet, at a lower quality. Also if the packets are being sent over a slow network, the codecs have the ability to change the bit-rate depending on the current network quality. This is useful because it allows the clients to maintain the call, although at a lower quality, than if it was to start lagging as the network is not able to provide enough bandwidth to maintain the quality.

![Packet Loss](image)

*Figure 5.4: Mean packet loss for the WebRTC survey*

It can be seen in figure 5.4 that the three first tests that were using only packet delay as its main interference, there still occurred some packet loss. This is the result from the fact that the packets sent to the participating clients go through a series of public networks. Each network increases the chance of something happening to the packet.
When comparing the different interferences used in the WebRTC survey with the MOS score that is produced by the students participating, a couple of things were shown. Firstly, in figure 5.5 it was shown that packet delay had a greater impact on the perceived quality when it was the sole interference in comparison to packet loss. When combining the two interferences, the perceived quality was lower but for each incremental in the delay, it only decreased the MOS score marginally in comparison. One can speculate why the test with no interference did not get a top quality rating. There can be multiple reasons why that is the case. Perhaps the quality of the audio format used needs to be higher. The song itself might not be appropriate for these kinds of surveys, why a larger pool of different type of songs might be needed.

![Figure 5.5: MOS score for each given interference for the WebRTC survey](image1)

![Figure 5.6: The MOS score for each given interference for the Icecast survey](image2)
5.3. Result

Figure 5.6 shows the comparison between the interferences when Icecast was used. Unlike the WebRTC survey, the test with no interference did not get the highest score. This might be contributed to the fact that two independent test-groups were used for performing the surveys.

One thing that was not taken into consideration is the familiarity that the students participating might have when taking part in a survey. Perhaps it is their first time doing a survey where they have to listen to different music sequences with interferences being added. One could argue that this is part of the surveys, as it is being conducted in a public environment where the participants might have little to no experience. This would be in contrast to the surveys being conducted in a closed environment with highly trained participants. The former might be able to represent the actual end users experience more accurately.
Even though the participation rate in the online surveys were extremely low, out of the 600 emails sent out only 2.8% participated, there was enough data to make evaluation possible. From my own personal experience with participating in online surveys, this did not come as a surprise.

The low participation rate was the main concern when performing this thesis. In order to have the result being statistically confirmed, a lot more people would have to participate in the surveys. The cause could be that instead of being a text-based survey, where the participants are asked a couple of questions to answer in writing, they are told to listen to a series of sequences. Having to listen to audio and trying to decide on quality as they perceive it, might be a lot more demanding for the participant. This is one of the main drawbacks when performing surveys online and have to be taken into account when designing future online surveys.

However, an online survey has many advantages. It enables us to reach people that might not otherwise be able to participate in the surveys if it was to be done locally. People living in the countryside would have to travel either by car or public transportation to the location where the survey is conducted. Performing the survey online thus saves both money and the environment, this gets more noticeable the bigger the survey gets.

When conducting the surveys, the data collected are considered sensitive as it is personal information about the participant. This includes the ip-addresses of the different participants. These were saved to potentially be used for preventing the participants from doing the surveys multiple times. However, they ended up not being used, as it was deemed unlikely that the participants would do the surveys multiple times. After the surveys had been conducted, only the most essential information was saved on the server and the rest deleted, including the ip-addresses.
By enabling real-time communication WebRTC has made the web browsers of today powerful enough to replace already existing applications, such as Skype, with web based solutions. Instead of having to download an application, one simply visits a website and logs in. WebRTC is not only bound to be used in the web browser but can also be used when developing desktop or mobile applications that need real-time communication capabilities.

One concern regarding privacy is when WebRTC runs over a VPN connection. As VPN is used for its ability to provide anonymity, WebRTC will reveal the public address for both the VPN and ISP\[23\]. This is caused by the peer-to-peer nature of WebRTC and might not be suitable for certain users, something that has to be considered when deciding to use WebRTC.

### 6.1 Goal

The aim for this study was to investigate how WebRTC behaves in non-optimal networks, and the outcome shows that WebRTC behaves well when the interferences in the network are constrained to a single type of interference. When multiple interferences are combined, WebRTC starts to struggle to maintain an enjoyable experience. Furthermore WebRTC showed us that it is fully capable of delivering an acceptable real-time experience for the end users. It also greatly reduces the code complexity, thus allowing developers to focus on important matters.

The feasibility of conducting online perceived quality surveys to evaluate WebRTC are shown to be plausible. The only hurdle is to be able to achieve a high participation rate among the participants. This is to remove any uncertainties that might be present.

### 6.2 Future work

Although the test environment that was developed and the surveys conducted were successful, there is a couple of things that could have been done better. The main one being that the WebRTC survey did not work properly on mobile devices. Given the prevalence of smartphones, a greater population might have participated in the survey. The problem comes from the fact that the API I used, Peerjs, has had no updates since 2015. The reason for selecting it was previous experience, although not in a mobile context. It might have been better to completely skip it and instead implemented everything.
6.2.  Future work

6.2.1  Tests

In the surveys only 8 different test-cases were used. The small number of tests was a result of trying to keep the surveys as simple as possible. Having a larger survey would be a lot better but this could have the effect of driving participants away from the survey. To get a clearer picture, a larger amount of test cases would have been needed.

For example, packet corruption was not tested at all, neither was packets coming out of order. By having more tests for each test case, a greater level of detail could be achieved. Perhaps it would have been better to make multiple surveys where each only focuses on one type of interference. This would also allow for greater granularity, since smaller incremental could show a more detailed picture of how the behaviour would be affected.

6.2.2  Interference

When it comes to the interferences that were applied to the outgoing audio stream, one could consider randomizing both duration and start of the interferences. In order to keep both the surveys and implementation simple, the interferences were applied in the middle of the music sample and lasted for only 5 seconds.

Another consideration would be that instead of explicitly choosing which test should have what kind of interference, one could have randomized the process where they are selected from a pool of interferences.

It was thought that TC was able to satisfy our needs with the interferences and while TC was able to produce different interferences, it was found that there was a bit lacking in functionality. A better alternative would have been to use a more complex network emulator tool such as Ns-3 to produce the interferences.

One thing that was not considered was how the setup for the interferences might affect the RTP protocol, which WebRTC uses. All the outgoing packets have an interference applied to them, but none of the incoming packets used by RTCP, a sister protocol to RTP, for maintaining quality of service.

6.2.3  Participants

Only students were considered for the surveys. This was mainly because of it being easier to contact students. If more people of different ages and experience would have participated, it would remove any uncertainties within the surveys and a wider audience would also increase the number of people that might participate in the surveys.
6.2.4 Audio

Audio was the primary media tested in the surveys but the choice and variation of music could have been a lot bigger. Only one song was used, which works if you want to get a basic test out, different kinds of music are not as susceptible to certain interferences as others. For example, a calm melody might be more affected by interferences than a rock song where noise is more prevalent. One consideration would be that perhaps the participants own music preference might come into play, affecting how they perceive the quality.

6.2.5 Video

Video is the other side of WebRTC that was not covered in the thesis. If video was also incorporated into the surveys, it would enable more possible scenarios to study. One scenario could be to begin with the participants only listening to audio, followed by video and lastly by using a combination of the two. Doing so would give a complete picture of how WebRTC behaves in different situations. However, video comes with a different set of requirements when it comes to testing. Future surveys should incorporate video to increase their desirability.

6.2.6 Test environment

Although the test environment was able to create a satisfying result, the implementation could be improved. Only one Chrome session was used and it made it cumbersome to change parameters, such as which music file to use. Instead a different approach would have been needed.

With Chrome version 60, there is a feature to create headless instances of Chrome on Linux. This would allow Chrome to be launched inside Docker containers. This in-turn would make configuring and launching different variations of the test environment a lot easier. With this setup, one could create different rooms that the participants would join.

Another approach would be to use cloud services. This would remove the need for hardware altogether and depending on the current load the website is experiencing, the performance can easily be scaled up or down automatically to meet the demand.
6.3 Conclusions

Although multiple research papers have performed various tests for evaluation the perceived quality using WebRTC, few of them have involved only human participants and online surveys. Most of them have been done in a controlled environment and the audio was being compared to a reference audio file.

The system created in this thesis was far from perfect, but it was able to produce a satisfying result. The lack of visual and audiovisual, although not necessary, slightly reduced the usefulness of the surveys as audio is only one part of WebRTC.

The use of free software in conjunction with WebRTC allows for the creation of new and exciting experiences within real-time communication without the need for the developers to acquire expensive third-party licences.

WebRTC is proven to be a much needed addition to the ever growing set of standards to be used in the development of web applications, as it lowers the complexity of the code. It is able to provide a wide array of features in the world of real-time communication even if exposed to networks with non-optimal performance.
A.1 Requirement specification

A brief example on how a requirement specification might look like.

Overview:

1. Send audio/video from the server to multiple clients
2. Apply various interferences to the audio stream.
3. Receive data from the clients

Test case 1:

- Media: Audio, melody
- Interference: Packet-loss
- Vote: ACR
- Evaluation: MOS
- Length: 10 seconds reference stream, 1 second blank and then 10 seconds of stream with interference.
### Table A.1: The perceived quality score for the WebRTC survey

<table>
<thead>
<tr>
<th>Test_1</th>
<th>Test_2</th>
<th>Test_3</th>
<th>Test_4</th>
<th>Test_5</th>
<th>Test_6</th>
<th>Test_7</th>
<th>Test_8</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
A.3 TC Commands

The various commands used with TC to be able to produce the interferences.

1.3.1 Pre-test commands

tc qdisc del dev eno1 root
tc qdisc add dev eno1 root handle 1: htb
tc class add dev eno1 parent 1: classid 1:1 htb rate 1000mbit
tc class add dev eno1 parent 1: classid 1:2 htb rate 1000mbit
tc class add dev eno1 parent 1: classid 1:3 htb rate 1000mbit
tc class add dev eno1 parent 1: classid 1:4 htb rate 1000mbit
tc class add dev eno1 parent 1: classid 1:5 htb rate 1000mbit
tc class add dev eno1 parent 1: classid 1:6 htb rate 1000mbit
tc class add dev eno1 parent 1: classid 1:7 htb rate 1000mbit
tc class add dev eno1 parent 1: classid 1:8 htb rate 1000mbit
tc class add dev eno1 parent 1: classid 1:9 htb rate 1000mbit
tc qdisc add dev eno1 parent 1:1 handle 10: netem delay 100ms 100ms loss 25% 25%
tc qdisc add dev eno1 parent 1:2 handle 20: netem delay 250ms 100ms loss 25% 25%

<table>
<thead>
<tr>
<th>Test_1</th>
<th>Test_2</th>
<th>Test_3</th>
<th>Test_4</th>
<th>Test_5</th>
<th>Test_6</th>
<th>Test_7</th>
<th>Test_8</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Table A.2: The perceived quality score for the Icecast survey
1.3.2 Test commands

tc qdisc netem 10: parent 1:1 limit 1000 delay 250.0ms 100.0ms

tc qdisc netem 20: parent 1:2 limit 1000 delay 500.0ms 100.0ms

tc qdisc netem 30: parent 1:3 limit 1000 loss 10% 25%

tc qdisc netem 40: parent 1:4 limit 1000 loss 25% 25%

tc qdisc netem 50: parent 1:5 limit 1000 delay 250.0ms 100.0ms loss 10% 25%

tc qdisc netem 60: parent 1:6 limit 1000

tc qdisc netem 70: parent 1:7 limit 1000 delay 100.0ms 100.0ms loss 25% 25%

tc qdisc netem 80: parent 1:8 limit 1000 delay 250.0ms 100.0ms loss 25% 25%

tc qdisc netem 90: parent 1:9 limit 1000 delay 500.0ms 100.0ms loss 25% 25%
REFERENCES


