Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Telecommunication and Media Informatics

QoE prediction of video conferencing services based on radio and core network parameters

Scientific Students' Association Report

Author:

Márton Molnár
László Varga

Advisor:

Dr. Alija Pašić
Márk Péter Szalay
Gergely László Dobreff

2022

Contents

Kivonat

Az elmúlt években az online videó-konferencia szolgáltatások használata rendkívül megnőtt, köszönhetően az online oktatásnak, és a távmunkának. Ezek a szolgáltatások stabil és gyors internet kapcsolatot igényelnek mindegyik félnél, amire számos körülmény hatással van. Például a késleltetés, a csomagvesztés, sávszélességi korlátok, vagy a gyenge rádiójel is befolyásolja a szolgáltatás minőségét, rontva a felhasználói élményt (QoE). Az 5G hálózatok egyik kulcspontjának tekinthető az URLLC szolgáltatások családja (ultra megbízható és alacsony késleltetésű kommunikáció) , ami rendkívül szigorú követelményeket támaszt a késleltetés felé. Habár a videokonferencia nem teljesen sorolható ebbe a csoportba, egy jól felépített modell segítséget nyújthat a hálózati operátorok számára a csomópontok és erőforrások menedzselésében, rossz viszonyok esetén. Azonban a QoE, mint fogalom számos különböző dologtól függ, ideértve az emberi és a technikai körülményeket is, és meglehetősen szubjektívnek tartják, amely új kihívásokat hoz előtérbe egy modell építésére nézve.

Széles körben használt és elterjedt videókonferencia szolgáltatások például a Microsoft Teams, a Google Meet, vagy a Jitsi, mind WebRTC protokollt használnak a peer-to-peer kommunikáció lebonyolítására két böngésző között. A WebRTC kapcsolatokat nyomon lehet követni, amiből számos paramétert megkaphatunk, ami a működést és a szolgáltatás minőségét hivatott leírni. Ezek a paraméterek, mint például a jitter, a képkocka/másodperc (fps), inter-frame-delay, vagy round-trip-time a felhasználónál tapasztalt QoE értéket nagyban meghatározzák. Azonban ezen metrikák karakterisztikáját általában a csomagszintű, vagy a rádiós közeget leíró paraméterek változása befolyásolja. Ebből kifolyólag képesek lehetünk megbecsülni a QoE értékét, illetve változását a végfelhasználónál csupán a hálózati paraméterek felhasználásával. Ezeket a paramétereket a legfontosabb rádiós metrikákkal kiegészitve, mint pl. RSRP, SINR, RSRQ, Txpower (uplink irányú sugárzott teljesítmény) egy pontos QoE becslést adhatunk a felhasználói oldalon. A modellépítés részét képezik ezen felül a handoverek és a frekvenciaugrások is, amik miatt a modell rádiós szempontból is helytálló.

Új megközelítés a témában, hogy a QoE modell és becslés pusztán csomagszintű és rádiós paraméterek felhasználásával történik, nem pedig a végfelhasználónál található, operátorok számára nem elérhető számolt metrikákkal, mint amilyen a WebRTC log. A hálózati operátorok számára ez egy komoly segítség lehet, ugyanis a node-ok, illetve a vivők beállításait menet közben tudják változtatni, olyan módon, hogy a felhasználói élmény maximalizálása mellett az energiahatékonyságot is figyelembe vegyék.

Habár lehetséges objektív becslést csinálni, egy teljes, pontosan fejlesztett modell létrehozása “ground truth” nélkül nem kivitelezhető. Erre a célra MOS értékek általi címkézést hoztunk létre a videóra és a hangra is, amit a modell felhasznál a felügyelt tanulása során. Az eredmények pedig azt mutatják, hogy lehetséges megbecsülni, megjósolni a QoE-re nagy hatással lévő paraméterek változását pusztán hálózati paraméterek felhasználásával.

Abstract

In the past few years, the usage of online video conferencing services has emerged due to increased demand for remote work and online teaching. These conferencing services require a stable and fast internet connection of each participating user, which can be degraded by numerous network and radio circumstances. E.g., latency, packet loss, bandwidth limitation and poor radio signal all impact the service quality, making the overall Quality of Experience (QoE) worse to the end user. In 5G networks, a strongly focused area is URLLC (Ultra Reliable Low Latency Communication) which holds tight requirements on latency. Even though video conferencing is not technically a URLLC service type, a well-developed model could help the operators manage the nodes and the resources when experiencing poor conditions, to create better QoE for the users in 5G networks. However, QoE is hugely affected by the circumstances, and is often considered a subjective metric which brings new challenges for the model building process.

The best-known video conferencing platforms like Microsoft Teams, Google Meet and Jitsi use WebRTC protocol for peer-to-peer communication between two browsers. The WebRTC sessions can be saved and analyzed after the session, with tens of different parameters available. These parameters, such as jitter, frame-per-second, inter-frame delay or round-trip time can describe the quality of service at the end user which has a strong connection to QoE. Moreover, the characteristics of the WebRTC related parameters usually depend on the packet-level parameters and their changes over time. This way we can predict the parameters that affect the QoE at the client side using packet analysis. Combining packet-level parameters with radio parameters such as RSRP, RSRQ, SINR and Txpower (the uplink transmitted power on different channels) , we can construct an advanced QoE prediction for the end user. Advanced radio scenarios such as handovers or frequency hops are also analyzed and used to build an accurate model.

The novel approach in this topic is that the QoE model and prediction is developed with only packet level and radio parameters. Network operators can benefit a lot from such a model, e.g., they can adjust the nodes' settings or the bearers whenever experiencing poor conditions, in order to create a better overall quality of experience at the end user. The outcome of this project is not only the objective QoE model and prediction, but it could have a possible positive effect on the energy consumption, too.

While it is possible to create an objective prediction, the training of a fully developed prediction model is impossible without a ground truth. For this purpose, a MOS (Mean Opinion Score) labeling is used on the recorded videos and audios which the model could use during the supervised learning process. The results show that it is possible to correctly predict the change of QoE-influential parameters at the end-user, only using in-network parameters.

Chapter 1

Introduction

With the help of the advanced network infrastructure available in the world, people from all around the globe are using a stable Internet connection every day, without traffic limitations, from various devices. This could either be PCs, laptops, tablets, or even embedded systems or factory devices. However, the vast majority of Internet traffic comes from mobile devices [28], and the traffic is particularly entertainment. Either watching a 4K live video or making live video conferences with 50-100 people, nothing seems to be impossible with the help of high-speed mobile networks. Before the 2010s, applications like these were incomprehensible, but today, we are using them every day - and also experiencing their benefits and threats. However, the question is: will the future of mobile networks will be powerful enough in terms of reliability, speed, and availability to serve the constantly growing demands of the people?

The evolution of mobile networks has had a significant impact on us. Starting from GSM (Global System for Mobile Communications), cellular networks play an inevitable role in our lives, giving us the chance to generate a large amount of traffic from almost anywhere in the world. Streaming high-quality videos, playing online games, or making video calls on our phones have become a daily habit. However, the next huge step towards a fully technology-driven life could be 5G. This opens up new opportunities for applications that require ultra-low latency to run smoothly and securely.

Statistics show that in the last quarter of 2021, 54.4% of the global Internet traffic was generated by mobile phones [28], which are mainly video calls and entertainment. Another interesting fact is that in 2017, 73% of Internet traffic was generated only by watching videos on YouTube and different streaming platforms. While these statistics include both wired and cellular Internet connections, the majority of them come from cell phones. As the global COVID-19 pandemic hit the world, live video conferences have become one of the most popular services. Whether it is remote schooling, work-related meetings, or simple video calls, we have to cope with connectivity issues. A new concept called QoE (Quality of Experience) has become a popular buzzword that can be translated as how the user feels while using the service. It is interesting to think about how a user will feel when making a video call, watching a live stream, or playing an online game with a stable and poor network connection. Obviously, network operators will strive for perfection to provide the best experience for their customers. The focus is on the following key aspects: wider bandwidth, lower latency, and better reliability.

Watching a live stream, using a VR/AR (Virtual and Augmented Reality) application, or playing in the cloud requires a stable and reliable connection. Watching a live video with a poor Internet connection will result in lagging, making it less enjoyable for the

end users. Playing a computer game with a limited bandwidth makes it impossible to be competitive and enjoy the game. These services need a constant low-latency connection, and also reliable network access which means there will not be any interruptions or outages during the entire session. It is obvious that e.g., the user of a video streaming platform and the company that purchases remotely controlled vehicles will not experience the same consequences when the used network could not meet the requirements. However, telecommunication companies' goal to serve everyone's future demands when it comes to providing a reliable service.

The importance of a great connection in the past few years has become even more significant. The effect of COVID-19 on the Internet is remarkable: the global peak traffic has increased by 47% [20], and the use of services like Facebook video calling (which had a 100% increase [10]) or any other video conference applications has risen by a lot. Many times, these applications are used on mobile phones/tablets with a mobile Internet connection, which shows why it is even more important to focus on this area.

The mobile operators and the network providers want to make sure that every user feels good about the quality of the network. However, it is complicated to objectively measure the Quality of the Experience (QoE) at the end user's side. QoE has a slight correlation to UE (User Experience), but this term is practical in telecommunication and is a strong relation to QoS (Quality of Service). While QoS is fairly quantitative, QoE is much more subjective. It is a challenge to tell what is a good quality of experience since the metric consists of subjective and human factors which differ.

We can briefly say that even though we are not able to quantitatively measure and describe QoE, however strong correlations can be made with the available objective QoS parameters. There are so-called QoE factors that all contribute to the end result, the user's perceived quality. These are human influence factors, system influence factors, and context influence factors. From these three, system and context influence factors describe the technical side and the environment of the QoE, while human factors can be anything related to the user, such as their age or gender.

System influence factors contain encoding, resolution, sample rate, bandwidth, and jitter, while context influence factors mean the location, and the temporal context (e.g. what time of the day we use the service, location, etc). However it is important to understand that the QoE will never be exactly equal for two people (since it is subjective), even if they have the same technical conditions. Due to these problems, building a QoE model is a difficult task using only the network parameters.

The main contribution of our work is the following.

- We propose an automated data collection system that enables monitoring transport and radio network parameters together with end-device related data at different radio and transport scenarios.
- We demonstrate how to obtain appropriate quality and quantity data for testing and training QoE models.
- We demonstrate that it is possible to build a QoE model for delay critical services relying solely on transport and radio data available network-wide for MNOs and we identify the parameters that are essential for the model.
- We build and compare several models that either use the WebRTC parameters or do not, which are usually not accessible by the operators.
- We also investigate the dependency of the model on platform and media content.

The structure of the work is the following: Chapter 2 summarizes the related works and the technical background. Chapter 3 describes the data collector system that we developed. Chapter 4 elaborates on the proposed Data Collection System and the QoE Survey that we used to collect the network parameters and the mean opinion scores. In Chapter 5 our QoE model is presented and evaluated, together with different models. Finally, Chapter 6 summarizes our thesis and the results that we reached.

Chapter 2

Background and Related work

In this chapter, we will look through the measured video conferencing services and their technological background. We will examine the service-related and the radio environment-related background and the typical problems that can degrade the service quality in mobile networks. Finally, we will present the collected related works and other, mainly webrtc-based QoE models.

2.1 Video Conferencing services

One of the most common services requiring a stable internet connection is video conferencing. Their importance has risen significantly because of the pandemic. Applications like Microsoft Teams, Google Meet or Jitsi are used by millions of people and they need to face the challenges caused by a poor network connection. These applications made it possible to connect with anybody in a few seconds with both audio and video connection. The idea of video conferencing (web conferencing) was introduced in the 1980s, however, the technological boundaries at that time made it a 'failure', which Carmen Egido discussed her paper [23]. The constraints for a web conferencing service are strict and the legacy networks simply could not meet these requirements in terms of bandwidth, latency, and speed. In terms of bandwidth, these services will experience difficulties without 1 Mb/s bandwidth for a good-quality video. The latency requirements are also strong, around 100ms. These numbers show us why it was hard to even imagine using these services back in the days.

In our study, we tested three video conferencing apps in terms of reliability and their capability to adapt to poor network conditions. First, we discuss the main characteristics of these three conferencing applications. Note that our research focuses on applications that are able to run in Chrome browsers, to be able to retrieve the WebRTC parameters, which will be described later.

2.1.1 Microsoft Teams

The first one is the most well-known and widely-used service developed by Microsoft. Teams is the main platform for business meetings and remote lectures, with 270 million active users today [50]. The application can be accessed from browsers, however, some features might not be available with specific browsers, or outdated versions. The service can be downloaded and used as a desktop application. However, there are distributions

for Android and iOS too, which makes it almost as famous as Facebook or Instagram these days. The service enables meetings up to 1000 participants and 4 hours of length or 1.5GB of data before shutting down. You can also connect to a conference with PSTN access (Public Switched Telephone Network) by dialing a specific number. Besides that, many configuration options are available such as giving limited access to specific people or changing technical parameters. The service really focuses on a great overall Quality of Experience to keep millions of users satisfied. The application, however, is not only designed for video calls but a complete workspace for businesses and educational purposes involving different channels, file sharing, or assignments.

Different bandwidth requirements can be found in the official documentation of Teams [39], breaking down the minimum, and the recommended specifications for one-to-one calls and one-to-many meetings. For example, for audio only, a one-to-one call requires 10 kb/s both uplink and downlink as a minimum, but the recommended speed is 58 kb/s for both directions. However, for video traffic, the minimum value is 150 kb/s and the recommended speed is 1.5 Mb/s. This shows that a video stream needs a lot more bandwidth, while these numbers are describing one-to-one calls only. For group calls and screen sharing, these numbers can reach 4 Mb/s. The documentations claim that for the best performance, this bitrate is advised.

2.1.2 Google Meet

Another well-known service is the conferencing service of Google, called Meet. Google Meet [5], according to our knowledge works in browsers only, which is a big difference to Teams. This service is a lightweight web-conferencing application, which excludes functionalities such as document storage, rooms, channels, file hosting or grading (tests and evaluation) and focuses on video calls entirely. This application might be used less frequently in the business environment and is used by individuals for connecting with friends or family. The service has a list of requirements and recommendations on their official documentation [5]. Sharing a HD video coming from your camera needs 2.6 Mb/s bandwidth with 2 participants and 4 Mb/s with 10 participants. The service also has the option to dial in with a PSTN access for people outside organizations that initiated the conferences. Google Meet has grown by 30% only between January 2020 and April 2020, but the exact number of users can not be found publicly. However, there is a source of information stating 100 million people use the application every day [21].

2.1.3 Jitsi Meet

Jitsi meet [11] is 100% open-source software designed for video conferences and can be used without an account. The application can run in almost any browser excluding Internet Explorer. Jitsi meet requires 500 kb/s bandwidth and their website states that a PC with 4 CPUs and 8 GB RAM is necessary for smooth running. A conference room is limited to 75 people but recommends having 35 participants at maximum to maintain the service quality. Jitsi supports different resolutions including SD quality and even 360x180 resolution. The frame rate can be easily fine-tuned for clients who intend to share their screens. The available options are 5, 15, or 30 frames per second, obviously, the larger value creating a performance overhead. In 2020, Jitsi had 20 million active users, and this number might be much bigger now. The service can be used as a desktop client, too.

2.2 Technological background

To understand the correlation between the service quality and the network - both radio and transport -, first the technical background should be discussed.

2.2.1 Service-related background

Different video conferencing services may use different transport protocols and optimization. A common and widely used framework is WebRTC [9] (Web Real-Time Communications) which provides the possibility to implement multimedia services with a peer-to-peer connection, between browsers. It is a useful technique since the traffic is not going through a server and its latency overhead disappears. Technically, WebRTC does communicate with a server but only in the beginning phase of the communication. The two browsers need to signal each other about their existence and location, and when it is achieved with the help of the server, the communication is flowing directly between the two browsers. This indicates faster communication, since the server, as a middle node in the communication is no longer used.

WebRTC differentiates signaling and media. Signaling is achieved with JavaScript, and media is sent with SRTP (Secure RTP [18]), just like VoIP. This protocol has the same characteristics as any other real-time communication protocol, thus avoiding packet retransmission due to latency constraints. The two clients know each other with the help of NAT traversal, allowing them to directly communicate within each other's private network, rather than their public gateways. The new approach is to have a video conference directly through the browsers without any additional plugins using peer-to-peer communication makes the WebRTC protocol revolutionary. The methodology of WebRTC is described in Figure 2.1.

WebRTC sessions are saved and stored in memory until the session is alive. This means that a set of parameters are recorded throughout the session and a log file can be downloaded with all this information about the peers. This contains the IP addresses, connection states and times, and numerous parameters that are described in the WebRTC documentation by W3 [9]. This log file later can be analyzed to retrieve information about the QoS and later derive QoE parameters, too.

An experimental transport protocol developed by Google is called QUIC [35] (Quick UDP Internet Connections). It was designed with the idea of reducing latency while keeping its security and authenticity. Some services already use QUIC, such as YouTube or Hangouts, and almost half of the traffic generated from Google Chrome browsers are going through QUIC. Even watching a live stream on YouTube uses QUIC.

The reason behind the development of QUIC was to 'achieve the safety of TCP and the speed of UDP'. QUIC replaces the four-way TLS handshake with a single handshake, making the establishment much faster. It also has the ability to adapt to quickly changing network states. This is achieved with a unique identifier for every connection which makes the re-establishment faster, simply sending out a new single handshake which allows the fast recovery. And finally, the head-of-line blocking mechanism is no longer a problem in QUIC, since it does not need to wait for the first packet to arrive. The data packets in QUIC can reach their destination independently, avoiding the unnecessary queueing that occurs with a dropped packet. However, there is no possible source for a deep, detailed analysis of the QUIC traffic (like the WebRTC logs).

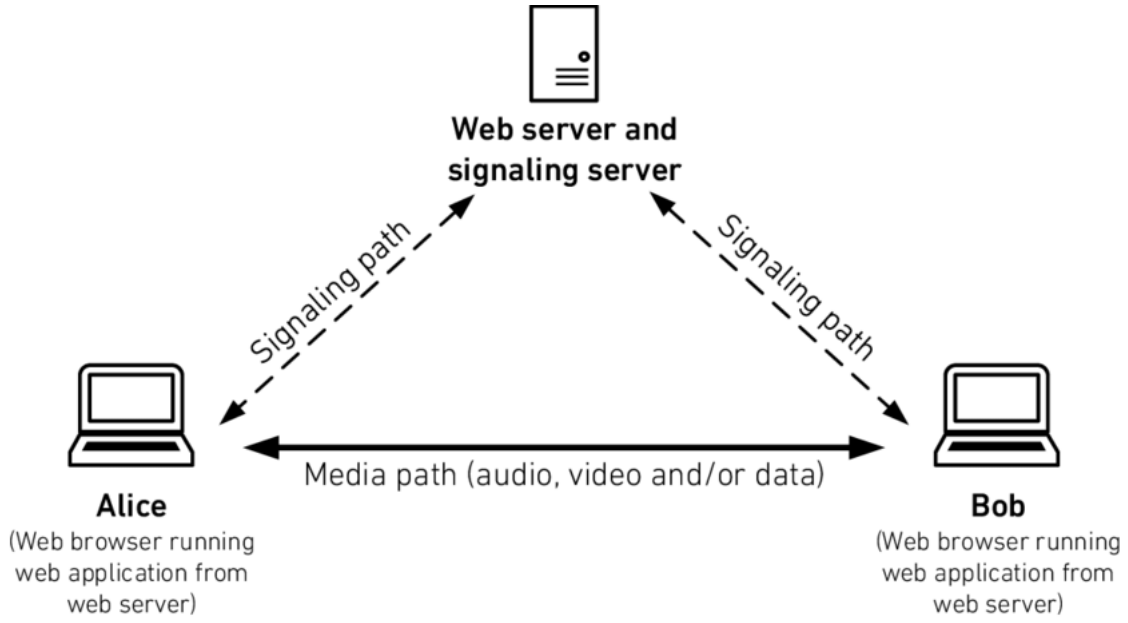

Figure 2.1: WebRTC explained between two browsers and a server [3]

For our study, a live YouTube stream is not eligible, because it does not act like a delay-critical service due to the buffered data.

2.2.2 Radio environment-related background

Cellular networks work with electromagnetic waves that are transmitted between the mobile phone (UE - User Equipment) and the towers. As mobile networks have emerged from GSM to 5G, the used frequency range has reached a point where it is a huge challenge for network operators to transmit a good quality signal. The negative effect of interference, blocking and the spreading of waves is much worse when using millimeter waves. Network architects and engineers face various difficulties and challenges. Different weather conditions can produce scenarios where electromagnetic radiation is absorbed: rain-fading, heavy snow, or a storm could possibly influence the waves and cause a decline in their measurable strength.

Electromagnetic waves can be blocked by initiating a Faraday effect [17]. The authors of this work called 'A Faraday Cage Exploration' tried blocking not only cell phones but other devices with lower frequency waves like FM radio (88-108 MHz), AM radio (535-1635 KHz). While 4G has a range of about 10 miles regarding the wavelength, the UWB (Ultra-WideBand) wave propagation distance is significantly lower. Ganji et. al. [25] write about the difficulties of beamforming millimeter waves in order to overcome data loss due to the physical property of the millimeter waves. It is even possible that pedestrians in crowded areas block millimeter waves thus the importance of a congested cell design and the proper beamforming solutions are both significant.

Blocking the signals, thus generating a disturbed radio environment can be achieved with any material besides medal that blocks electromagnetic signals. These are all legal methods, however only relevant for researchers to decrease the incoming signal strength. The illegal solution is the usage of signal jammers and frequency blockers. 5G networks face the difficulties of signal blocking and interference more than 4G due to the short wavelength.

Even a temporary blocking is possible between the transmitter and the receiver in urban areas, and the lost data needs to be recovered by initiating a new connection with a new transmitter. All these situations are very common and can cause issues in the 5G radio environment.

However, not only blocking, but interference is a serious issue in mobile networks. Since GSM, various techniques were proposed to reduce the effects of EMW interference. Different methods were implemented to have a greater bandwidth utilization, communicate with more devices at the same time, and do this without interference.

Multiple access methods like Frequency Division Multiple Access (FDMA), Code Division Multiple Access (CDMA), and Time Division Multiple Access (TDMA) [32] are all used to divide a radio spectrum, thus lowering the chance of a disturbed signal. Interference was a huge problem in the early ages of mobile networks. Waves with a higher wavelength lead to a bigger angle where two waves can collide.

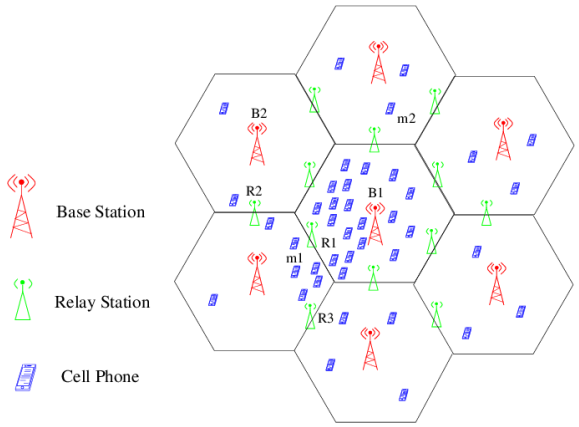

Figure 2.2: Cellular network example with base and relay stations [2]

Besides the base stations, which are located usually in the middle of a cell, relay stations are designed to enhance signal strength for the users and are placed on the cell edges. Usually, a UE is communicating with a relay station, which acts as an internal node to direct the traffic to the base stations. However, relay stations are usually lacking the energy to serve all UEs at the same time. A possible layout for the base and the relay stations can be seen in Figure 2.2.

A common method to overcome the interference issues is to use frequency hopping [22]. This means that a predefined scheme of frequencies is allocated, or reserved, and the device switches from one frequency to another to transmit the data. This happens often when the current used frequency experiences interference or blocking. This is even used during calls to average the interference (interference diversity) and to minimize the fading effect (frequency diversity). Frequency hopping also could mean that a different amount of power is transmitted with the change of the frequency by the UE. Thus, cell phone users are balanced to the amount of power needed for secure and satisfactory communication with the stations [38]. The maximum amount of transmission power allowed by the law for cell phones is 23 dB. When experiencing poor conditions, the devices often transmit the maximum amount of power on the uplink in order to overcome distance, interference, or blocking. Luckily this is easily measurable with the RSRP (Reference Signal Received Power), which is inversely proportional to the uplink transmitted power (txpower). Measuring the RSRP value is relatively easy, there are many applications that show the real-time received data about this parameter. Also, another indicator is that dur-

ing these periods, the cellphone starts to become hot as it has to broadcast with a higher power. Our focus is to show how such parameters (RSRP and the uplink parameters) affect service quality.

The scope of our research is currently 4G, but in the future we will want to change that to 5G.

In 4G, different physical, logical and transport channels are used. Physical channels carry user data and control messages, while logical channels provide services for the MAC layer (Medium Access Control). Transport channels also offer information for the MAC layer. Amongst these, the physical channels are relevant for us as they are responsible for the user's control messages and data. The txpower parameters usually include various physical channels. Besides the shared and control channels, a random access channel and the sounding reference signal is also reported value from the operator side. The different channel parameters are PPUSCH (Physical Uplink Shared Channel), PPUCCH (Physical Uplink Control Channel), PPRACCH (Physical Random Access Channel), PSRS (Physical Sounding Reference Signal). The LTE channels are described in [1].

Services of 5G are designed to operate with a maximum of 10 ms in terms of latency, while 4G networks cannot meet this constraint. One of the main reasons for this is that it uses a scheduling-based transmission mode, named GB (Grant Based) Access. This scheduling mechanism is a four-step procedure, and it is initiated by the UE to grant access from the network to send traffic. The scheduling request-access transmission takes time (around 10ms), which itself initially makes a URLLC (ultra-reliable low latency communications) service unusable through 4G. While for 5G, a GF (Grant Free) access was proposed [37]. This means that the UE has access to the uplink channel and can send data (first send the necessary control information) without the four-step access granting mechanism. Liu et. al. analyze the grant-free access that makes URLLC services feasible in 5G networks [36].

2.2.3 Typical network issues

A new approach is to analyze both the transport and the radio parameters in different network conditions, in order to obtain information about their behavior in different scenarios. When using a cellular network or Wi-Fi, the quality of the received signal determines the quality of the connection. Other properties of the medium will also influence the service quality and the entire connection. Mobile operators sometimes provide various information about the signal quality, such as uplink power, used frequency ranges (DL and UL) or even modulation schemes. These parameters could be used to monitor and analyze the radio environment. To obtain more information about the quality of the network, we can capture the traffic and analyze the individual data packets. This way, correlations can be made between the service quality and the different network parameters that describe the connection. Packet analysis is crucial for network administrators and network management: this way they could understand the reasons behind poor performance or resolve issues and also forbid potential attacks. A data packet can be captured at any node between the source and the end device. With software like Tcpdump [27], Tshark [8] or Wireshark [4], we can analyze the traffic, packet by packet. Amongst these, Wireshark is one of the most popular with its user-friendly graphical interface. The ability to filter the packets by the transfer protocols or the addresses and create graphs makes it convenient to use for a skilled engineer to get the necessary information. Either for security purposes, troubleshooting an error, load balancing, or finding traffic congestion, the use of packet analysis is a big advantage for network providers and engineers. Using

data-intensive services often puts pressure on the network thus it shows the possible weaknesses. These types of issues are much more frequent in cellular networks. For example, if the radio signal quality (RSRP - Reference Signal Received Power) is poor, the UE (User Equipment) will switch to a different cell or fall back to a previous version of the network. In poorly covered areas, devices still struggle to connect to 4G and are forced to use HSPA (High Speed Packet Access) or even 3G. Even though the 4G coverage in 2022 for Hungary is 93%, 97%, and 98% for the three largest telecommunication companies, such issues still happen. The reason why service quality is a widely researched and focused area of telecommunication today is to avoid the slightest possible inconvenience to the end user.

The typical network issues are usually a slow download (or upload) speed, packet loss, and latency problems. Another common issue is the latency variance called jitter. A quick change in these parameters often causes lagging or interruptions in the services. If we investigate the effect of a specific network degradation on the service quality, we can prepare for these events, both in the network and in the software. For example, as a developer, if we know how our service reacts to a certain amount of packet loss, we can prepare the service to handle these situations better with some error-correction methods. Video conferencing applications usually handle these issues in a simple way: either by turning off the incoming or outgoing video or switching to a lower resolution for the videos. Another typical method is when the service starts a retransmission stream, which is able to amend the original video stream. According to our experiences, these services are designed to handle a small rate of packet loss and a few tens of milliseconds of latency, which is quite common. However, they usually reach a point where it is no longer usable and this is what our research focuses on: to find this threshold point and create a QoE prediction, based on our experiences and the available large set of network parameters.

2.3 QoE models

Many researchers have already tried to cover this topic from different aspects using different methods. However, their approaches seem to be different from ours, due to various reasons. The papers we studied either used wired connection, Wi-Fi, focused on subjective QoE only, or used different protocols. However, our study aimed to use the in-network parameters, radio parameters and WebRTC parameters. This could provide a detailed dataset in order to develop an accurate model, which can predict the expected quality of experience for a video conferencing service.

2.3.1 Ericsson's QoE model

Our research is in cooperation with Ericsson, which allowed us to get access to an existing QoE model developed by their team. This model was developed for surveillance cameras, especially for human recognition, which later seemed to be less accurate for our video conferencing services.

The model entirely relies on WebRTC parameters, and it uses the following parameters:

(i) FramesDropped

(ii) $\text{OneWayLatency} = \frac{JitterBufferDelay}{JitterBufferEmittedCount} + \frac{RTT}{2}$

(ii) $\text{QpQuota} = \dfrac{QpSum}{FramesDecoded}$

(iv) $\text{QpDeduction} = \dfrac{QpQuota - 20}{35}$

(v) $\text{BitrateDeduction} = \dfrac{1500000 - bitrate}{1500000} * 1.5$

The purpose of the jitter buffer is to rearrange RTP packets into frames and have smooth playout. The jitter buffer delay value means the elapsed time amount between the first and the last packet's arrival time that belongs to the given frame. Jitter buffer emitted count value means how many frames came out of the buffer. Qpsum only exists for video streams, and this means the sum of QP values, where a QP value depends on the codec. The value of this parameter also depends on the type of content which is played out, according to our experience.

Then, the following algorithm is used to calculate the QoE for each data point, which means that a QoE value is assigned to every second while the service was used.

Algorithm 1 Ericsson QoE model

$QoE \leftarrow 5$
$QoE = QoE - 0.4 * FramesDropped$
if $OneWayLatency > 0.4$ **then**
    $QoE = QoE - OneWayLatency - 0.4$
else if $OneWayLatency <= 0.4$ **then**
    $QoE = QoE$
end if
if $QpQuota > 20$ **then**
    $QoE = QoE - QpDeduction$
else if $QpQuota <= 20$ **then**
    $QoE = QoE$
end if
if $Bitrate > 1500000$ **then** ▷ 1.5 Mbps
    $QoE = QoE - BitrateDeduction$
else if $Bitrate <= 1500000$ **then**
    $QoE = QoE$
end **if**
return QoE

2.3.2 Husic et. al. QoE model

Another algorithm we found applicable to our study was developed by Husic et. al [29]. Their model relies on WebRTC parameters only, however, their work is slightly different. They used two-way video conferencing with real applicants on both sides, thus the QoE is for the whole video conference.

Their formula has only four variables - two for the video and two for the audio. This way, the calculation is much simpler than the model developed by Ericsson's researchers. We assume the outgoing traffic is similar to the incoming traffic, thus *PacketsReceived* is equal to *PacketsSent*, since our dataset used one-way traffic only.

This model also produces a QoE value for every timestamp where WebRTC logs are available.

Algorithm 2 Husic et. al. QoE model

$QoE \leftarrow 3.420$
$QoE = QoE + 0.005 * PacketSentPerSecVideo$
$QoE = QoE - 72.898 * JitterAudio$
$QoE = QoE - 0.001 * PacketsLostAudio$
$QoE = QoE + 0.004 * PacketsReceivedVideo$
return QoE

2.3.3 Other QoE models

A great number of different publications can be found that deal with the prediction of QoE for real-time services, many for video conferencing. Different QoE models and predictions are published [43, 15, 33, 31], with different aspects. These works analyze video-streaming, general web browsing and video conferencing, too. Also, WebRTC is widely researched [16, 29, 51] since this is a relatively new protocol, however, our study was not fully covered in any of these papers.

The work of Ammar et. al [16] deals with WebRTC services and some important WebRTC parameters, while their work does not produce a QoE model. They investigated the behavior of bandwidth, bitrate, packet loss and PLI (Picture Loss Indication).

Their work focuses on the change of these parameters when experiencing poor conditions. They use a quasi-binary classification for the network conditions: good at both parties and bad at both parties. Ammar et. al [16] state that mobile networks tend to turn packet loss into delays, which we later wanted to investigate and state the importance of packet loss and delay in WebRTC-based video traffic.

Nikravesh et. al. [40] dealt with QoE inference from QoS parameters. Their QoE formula is a function of bandwidth, delay, and packet loss. They also created a mapping for application-independent QoS parameters and application-dependent QoE parameters together with a model. In case of a dropping FPS (Frame Per Second) or low bandwidth, the QoE is bad, in other cases, the QoE is good. We analyzed these results and tried to create a more complex model for this use case.

Bocchi et. al. [19] created a QoE model for web browsing, based on HTML elements and JavaScript events. This is not applicable to our study, however, their work shows the diversity and subjective nature of the QoE when sitting behind a screen. This also reflects the hardships of creating a QoE model entirely relying on objective parameters.

An exciting conclusion was formed in the paper of Vucic, Skorin-Kapov [51], in which they state that unlimited bandwidth might have a negative effect on the overall QoE due to congestion. They also stated that this might occur because of the limited capabilities of the used mobile devices: they might not be able to handle such high resolution (and bitrate) and the fast decoding might fail. However, our study was made with a laptop, rather than a mobile device.

Boni Garcia et. al. had a study of ideal and non-ideal network conditions with QoE prediction [26]. They use WebRTC video conferencing services (the application is called AppRTC), and introduce 0%, 15%, 30% and 45% packet loss, and 25ms, 50ms, and 75ms jitter to the network in different measurements. They use Full-Reference models such as VMAF [14], SSIM [6], MSE [44] or VQM [7]. For example, VMAF is developed by Netflix for video quality assessment and they all are used in numerous researches. These models

are complex and their performance depends on various parameters. They state that jitter does not have an effect on the subjective scores for the audio streams.

As previously mentioned in Subsection 2.3.2, Husic et. al. produced an objective QoE model which approximates the results from their manual survey. However they used Wi-Fi connection for the Internet connection thus, they did not collect radio parameters. Our goal is to propose a QoE model, hence we focus on the radio and transport QoS parameters, and omit data sources like WebRTC logs since they are not available for in-network nodes. However, according to the best of our knowledge, there is no available open-source radio and transport parameter-based QoE model, hence we compare our proposed QoE model to the WebRTC parameter-based one by Husic et. al [29].

Chapter 3

Data collection system

In this chapter, we describe the data collector system we have developed to gather a large amount of data automatically. We introduce our tool's hardware and software requirements and the test system's architecture. We propose a methodology for the systematic measurements. This chapter also describes which parameters can be retrieved and how they are collected. Finally, the possible network degradations are listed together with the important KPIs we derived from the collected data.

3.1 System architecture and setup

We developed a testbed in which different delay-critical services can run while parallelly monitoring the network parameters. In addition, while running these services, we can introduce different degradation to the transport network and to the radio environment. The logged data is later used to show how the services react to different challenging scenarios, e.g. high packet loss, high latency, or poor radio signal quality.

Our goal was to create a more accurate QoE model for delay-critical services than the currently available solutions. In order to do that, we propose an automated data collection system for mobile networks to provide a standardized way to analyze delay-critical services under different radio and network conditions (see Figure 3.1). Both uplink and downlink radio and transport data can be captured at the same time. Note that WebRTC logs are also collected for future comparisons.

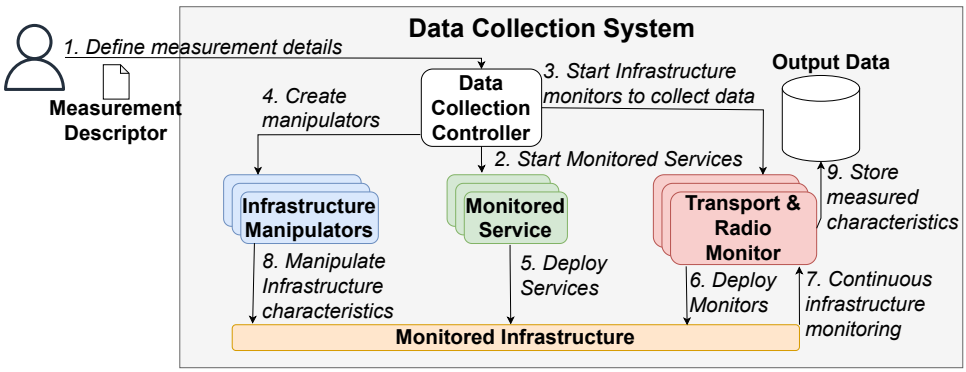

Figure 3.1: Architecture of the Proposed Data Collection System

Figure 3.1 includes the architecture of the proposed data collection system. For the Measurements, we have to create the configuration file (Measurement Descriptor), listing the service, and the monitors. Both transport and radio monitors, together with the infrastructure configurations, which are responsible for the degradations.

The possible *video conferencing* services are Microsoft Teams, Google Meet and Jitsi, while *tc* [47], *tcconfig* [48], *iperf3* [30] are infrastructure configurations. *Tcpdump* [27], the self-developed *radio monitor*, *spindump* [24] and *OBS Studio* [41] are the supported monitors.

We have used this data collection system at Participant B, in the video conferencing use-case of Figure 3.2 to collect data in case of various artificial network degradations. The measurements are described in details in Section 3.2.

The network degradations are performed with the tc tool, which can be used to configure the kernel packet scheduler. It is possible to add different degradation types to the packets easily, to a chosen interface: delay, jitter, packet loss, limited bandwidth. Then, tcpdump captures the packets in .pcap format, and the output of the packet capture can be correlated with the other output log files, such as the radio parameters and the WebRTC parameters.

Since we are using a cellular network, we have the option to collect data regarding the radio environment. Network operators usually provide different radio parameters, which we periodically monitored. For example, metrics about the signal quality [RSRP (Reference Signal Received Power), RSRQ (Reference Signal Received Quality), SINR (Signal-Noise Ratio)], information about the current and neighbor cells, or the transmitted uplink power can be saved. This way, using the timestamp as the joint attribute of the different datasets, it is possible to create such correlations between the radio environment and the captured packets. We planned an effective and logical way to collect the radio data, and the necessary scripts were also developed by us for this purpose.

We proposed an idea for a radio monitor tool, specially for the used 4G USB stick, the Huawei e3372h-320. This device connects to the computer with a USB port and has a slot for a SIM card to be inserted to provide access to the cellular network. The radio data collection tool uses Python, which periodically polls the stick's API, which can be accessed on *localhost*, to gain information about the radio environment. This period can be changed, if we want a data point every 5 seconds only. We set the default value for 1 second, however, it can be polled with the desired frequency. The parameters are saved with timestamps to a .csv file, and they can be used for further analysis. The results are later put together with other sources of data, such as WebRTC logs and the transport data.

For the research, we used an unlimited mobile data plan in order to make sure we will not run out of data while performing a large number of measurements. Since these parameters are sent by the mobile operator, due to the non-deterministic and non-linear traffic going through the network, these reports might be missing sometimes. From one to five seconds, the time elapsed between two data points can be different, and it also depends on the mobile operator, and the type of parameters. The RSRP values are usually updated in only five seconds, while, according to our experiences, the uplink TXPOWER parameters are reported more frequently, with one-second resolution.

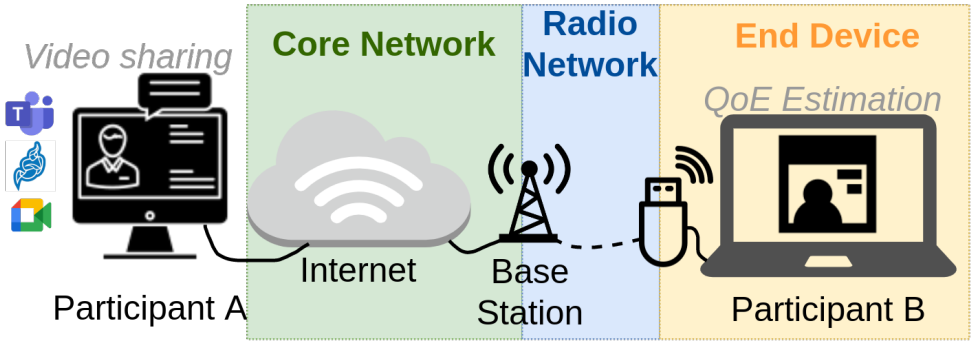

Figure 3.2: Measurement setup

The key thing is that the tool is fully portable and can be used with any computers that have the necessary tools installed together with the external USB stick. Figure 3.2 describes the concept of the test system. As we stated it earlier, the test system is automated, which means that it uses an automated bot browser to connect to the video conferencing service and also to leave the calls when the measurement is finished. Launching a measurement only requires specifying the path of the Measurement Descriptor, which has all the information about the type of degradation, the length of the measurement and every other piece of information. The performed measurements will be described later in details in Section 4.

3.1.1 Hardware specifications

The data collector system can be used on any computer, which has a Huawei e3372h-320 stick inserted and Ubuntu 20.04 (and the necessary packages and external softwares) installed. About the exact hardware specifications, the system is currently installed on two laptops, one with an Intel(R) Core (TM) i5-10310U CPU @ 1.70GHz CPU, and 16 GB of RAM, and the other with a 12th Gen Intel® Core™ i7-12700H × 20 CPU, an NVIDIA GA104M GPU and 16 GB of RAM. A GPU is not necessary for the system to run smoothly, however, the screen recording might cause issues on a poor setup. Also, the monitors are fully configurable, which means that e.g. a different USB stick can be used with a new radio monitor, developed for that model. This means that if we change the scope of our research and focus on 5G networks later, we can use this test system, after modifying the radio monitor tool according to the specifications of that given 5G-capable USB stick.

Another important part of the system is to create a consistent setup for the other end of the communication in terms of network connection. In Figure 3.2, this is Participant A. A dedicated video-conferencing server PC with a fixed IP address was configured for this purpose. The duty of this PC is to function as the second, fixed member during the video conferences, while sharing its screen and playing out different videos, which are described in Section 4.

3.2 Measurement methodology

We planned a practical structure for the measurements. The installed video conferencing server is a fixed participant in the video call, with its screen sharing enabled, as it was described in Figure 3.2. Participant A shares not only the video, but the audio from its computer, however, it is not the audio of its microphone, but the output audio (which normally comes out of the speakers), that is piped in as an input. This is intentional because the videos are streamed from Participant A, and their audio needs to be heard for the participants in the QoE survey, which will be described in Section 4.2. The measurements are divided into periods. The meaning of this periodical structure is that they are differentiated by the network circumstances. Some periods have artificial network degradations, while some do not. Each period will last for a given amount of time, (in seconds) which we can specify when starting a measurement from the command line.

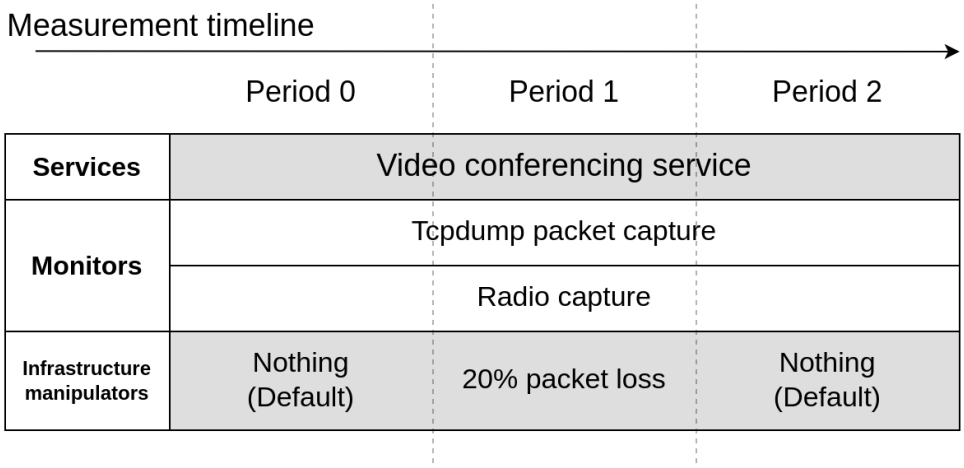

Figure 3.3: Periodic structure of the measurements with 20% packet loss applied

We designed the measurements in a logical way. Each measurement consists of three periods, containing both degraded and regular network conditions, and each period lasts for 30 seconds. During the first and the last periods (Period 0 and Period 2), we do not apply artificial network degradations. In the middle period (Period 1), we apply the specified degradation. In Figure 3.3, the structure of our measurements can be seen. In this example, the applied degradation is 20% packet loss, however, we performed measurements with various degradations, that will be described later in Sec. 4.

At the beginning of each period, a video player opens up at Participant A, and starts to play a specific video, while sharing its screen with Participant B in the call. This is achieved with a simple signaling mechanism: when Participant B starts a measurement, the start command sends an HTTP POST request to Participant A, to start a video. The URL of the video is specified in the request's body, and the videos are stored locally on Participant A. This signaling happens every time, when a new period is started. This way, each period will have the same video content played out during the measurement. In this case, the three 30-second periods contain the same video content, but with different network circumstances.

The reason why we used three periods is that the first and the last periods use an uninterrupted network connection, while the traffic in the middle period is manipulated artificially. This is a great way to create comparisons between Period 1 and Period 2. Period 0 is responsible for the video conferencing service and the necessary monitors to start up and reach a stable state.

The middle periods are then later analyzed in detail, while the edge periods are used as references. The evaluation is processed with the help of the recorded screens: we play back the screen recording session which shows the streamed video coming from the video conferencing server. When the network traffic is manipulated, the effects are usually visible and significant changes are present compared to the normal periods.

The system starts the necessary tools when we start a measurement. These tools are tcpdump, OBS and the script which polls the LTE stick about the network conditions. Then opens the browser with the given service and enter the conference room in which the installed conference server is a permanent participant. The screen is recorded throughout the whole conferencing session, which in our case adds up to 3 times 30 seconds, so an overall 100 seconds with the initialization and the quitting procedure. Then the browser is closed and the connection is aborted.

3.3 Possible degradations initiated with the manipulators

The key feature of the system is artificial network degradation, which allows us to set up the desired amount of packet loss, delay or jitter to the network interface. Reducing the bandwidth, creating corrupt packets and duplicating packets is also possible. In our test measurements, we first investigated which transport layer impairments degrade the perceived user experience. Once these disturbances were identified, a great number of measurements have been carried out to reliably identify the correlations between QoE and the metrics. These degradations were applied when we created the dataset. These degradations will be described in this section.

3.3.1 Packet loss

One of the possible options is initiating packet loss, where the specified amount of the packets (in percentage) will be dropped. A great method to prove the efficiency of the tool is to compare the size of the two .pcap files. The number of packets going through the physical interface will be bigger than the number of packets monitored on the manipulated interface, which was described earlier in Subsection 3.4.2. The selection of the dropped packets is totally random and out of our control, which simulates a real-life scenario. In our research, we examined how the services react to 10, 20 and 30% packet loss, and documented the results.

3.3.2 Delay

We investigated the effects of an artificially induced delay on video conferencing services. We started with 0 ms and went up to 500 ms delay in 50 ms steps, and we came to an interesting conclusion. The presence of a constant delay has no significant effect on the QoE, especially for video conferences. This is mainly because the delay is more like an offset, which is only visible at the first packet which arrives with the delay. This biased video stream is no longer impacted by the negative effect of the delay, however, it

would be critical for other services, for example, cloud gaming. We created experimental measurements for cloud gaming where the 100 ms artificial delay almost made the service unusable. However, since our main focus is on video conferencing, the results achieved with delay made us decide not to create a systematically structured dataset with delayed measurements. To summarize, the final dataset does not contain degradations with delay only, but we used *jitter* instead.

3.3.3 Jitter

Jitter is often referred as the variance in delay. The official jitter calculation is made by taking the time difference of the delays: the time elapsed between the arrival of the current packet and the previous one, and the time elapsed between the arrival of the previous packet and the packet before. This is obviously a continuous calculation process, for all the future packets, respectively.

Within our data collector system, we are able to create artificial jitter, but with slight changes. First, we need to set a delay, as the first parameter. As the second parameter, the variance of the delay needs to be specified. This means that for a 100ms delay, a 50ms variance will result in a 100 +/- 50ms delay, which - in contrast with the simple delay - can cause an overall much worse user experience for video conferencing. We used different jitter amounts: 100ms delay with 50ms variance, 150ms delay with 75ms variance, and 200ms delay with 100ms variance.

3.3.4 Bandwidth limitation

Probably the most significant result could be achieved with bandwidth limitations. We analyzed the bandwidth requirements of the different conferencing services and then set up the three steps of bandwidth degradation. Starting from 1.5 Mb/s, we reduce the bandwidth down to 500 kb/s - which is a really intense limitation - that can cause significant issues in the services.

We claim that 500 kb/s is clearly incapable of handling the traffic of a real-time video conference. We also assume that the effects of a narrow bandwidth will be visible in the QoE, as well as in other parameters.

3.3.5 Radio shielding

Creating proper physical radio shielding is a bit more challenging task since it involves manual work and several factors. Numerous non-deterministic and non-reproducible radio measurements were performed by us in the testing phase, e.g. entering an elevator, an underground cellar, or taking the tram. After, based on our experiences, we established a proper measurement protocol, regarding the methodology of creating a reproducible radio perturbation. The challenge was to create an environment, where the RSRP value always decreases with the same amount. We needed a solution that can be applied as many times as we wanted with the same outcome. For this purpose, we purchased a metal case to block the radio signals in an effective and deterministic way. The metal cage was wide enough to cover the laptop with the USB LTE stick inserted, and produced a deterministic decrease in the RSRP value, with an average amount of 10dBm.

3.4 Measured and derived metrics

The main focus of our test system is the ability to connect a large amount of data in an automated way. This section describes how the different groups of parameters are collected with our tool.

3.4.1 Radio parameters

During the measurements, a Huawei E3372H USB LTE Stick was used to connect to the 4G network. Note that even though we call the device's endpoint every second, this does not necessarily mean that the data changes with a resolution of seconds. It is the internal operation of the device that decides how often the retrieved data on its interface is updated. In our experience, the duration of the data update varies between 1 and 5 seconds. Using our radio monitor tool that we developed, we can obtain various radio-parameters with 1-second resolution from the USB stick. From these parameters, we save the most important ones into a .csv file with the timestamp so that their values can be matched with other data from other sources.

Amongst the available downlink parameters RSRP, RSRQ and SINR are the most important ones. RSRQ is the Reference Signal Received Quality which is (N*RSRP)/RSSI measured over the same bandwidth, where N is the number of used RBs (Resource Blocks, the smallest RB is 180 kHz). This indicates the quality of the received reference signal, and RSRQ is used in scenarios where RSRP alone is not sufficient for handover decisions. SINR is the Signal Noise Ratio which represents the wanted signal compared to the unwanted signals caused by interference and noise.

Not only downlink parameters, but uplink parameters can also be monitored. Uplink parameters are not always measurable with mobile phones, however, the uplink transmitted power (TXPOWER [dB]) holds a lot of information about the radio environment.

From the response, we kept most of the parameters. The purpose of collecting the UL-DL frequencies is to measure the frequency hops in correlation with the handovers and the other parameters. The uplink transmission power metric called TXPOWER contains signal strength values for four different channels, which are described below, together with the other collected parameters:

- RSRP - Reference Signal Received Power measures the signal strength in [dBm]
- RSRQ - Reference Signal Received Quality measures the quality of the received reference signal. [dB]
- SINR - Signal to Interference & Noise Ratio measures the signal quality: the strength of the wanted signal compared to the unwanted interference and noise [dB]
- Cell ID - the physical layer Cell ID, used to detect handovers
- Downlink Transmission Mode (e.g., 7=Beamforming)
- Uplink frequency
- Downlink frequency
- List of the uplink Modulation Coding Scheme indices.
- List of the downlink Modulation Coding Scheme indices.

- PPUCCH - Uplink transmitted power on the Physical Uplink Control Channel, which is used to carry UCI (Uplink Control Information). [dB]
- PPUSCH - Uplink transmitted power on the Physical Uplink Shared Channel, which is used to carry signaling messages, control information and application data. [dB]
- PSRS - Uplink transmitter power for Physical Sounding Reference Signal, which is used at the eNB to estimate the CSI (Channel Status Information) for a range of frequencies. [dB]
- PPRACCH - Uplink transmitted power on the Physical Random Access Channel which is used by UEs to request an uplink allocation from the base station. [dB]

3.4.2 Transport parameters

As we know, Internet is a packet-switched network, which means that the traffic is divided into IP packets and hundreds of packets are transmitted in a second from one device to another. To collect transport parameters, our system uses tcpdump, which generates a .pcap file for the selected interface. We created a virtual interface within the test system to perform the artificial network degradations on it, and push all the traffic through this interface.

Due to security purposes, the packets are encrypted, and the data part of the packets ar not visible. The public headers however are visible, which means that the used protocol and the source + destination addresses can be captured. This means that the information we can obtain from the packets is limited. We can measure the elapsed time segments between two packets, the burstiness of the arrived packets, but the encrypted data will remain unknown for the analysis. Our system collects .pcap files automatically which we later process by automatized scripts, together with the radio data.

Since the WebRTC framework sends data streams, i.e. audio and video streams over UDP using the RTP protocol, only these packets are of interest to us from the intercepted packets, thus we filter these packets only. Note that these packets are encrypted, which means that in reality SRTP and SRTCP packets are being transported. While the header of SRTP packets can be decoded, among other things, the SSRC (source identifier), SRTCP packets do not contain any useful information. We can use the SSRC to identify different data streams. However, the header does not tell us what type of stream (audio or video) it is, so we can only infer. Since we know that video packets are large, we categorize streams with an average packet size below 400 bytes as audio streams and above 400 bytes as video streams. After separating the packets for different data streams, we derive aggregated metrics for each data stream for 1-second intervals. We can derive the lost packets within a given stream from the sequence number field. For more information about this field and the RTP protocol, see [45]. Obviously, packets with an undelivered sequence number will be marked as lost. To determine the burstiness, consecutive packets arriving within 1 ms of each other are categorized into one burst. All of the listed burst metrics are based on this categorization. For more information about the burst metrics, see [34]. During the measurements, we have collected the transport metrics listed below:

- Packets received - Number of received packets in the past 1 second.
- The average/min/max/std of the interarrival time of the received packets in the past 1 second. Interarrival time is the time difference between two consecutive packets.

- The bitrate of the stream calculated from the received packets. The size (in bytes) of the UDP packets are summed. Since it is calculated for the past 1 seconds it is measured in Kbps.
- Packet lost count - The number of lost packets in the past 1 second.
- Packets lost percentage: We derive the number of packets that should have arrived from the smallest and largest sequence number received in the last 1 second. Dividing this by the number of lost packets gives the percentage of packets lost in the last 1 second.
- Number of bursts in the past 1 second.
- The average/min/max/std size (in number of packets) of the bursts in the past 1 second.
- The average/min/max/std size (in bytes) of bursts in the past 1 second. The size of one burst is calculated by summing the consisting packets sizes.
- The average/min/max/std length of the bursts in the past 1 second. The length of one burst is the difference of the timestamp of the first and the last packet in that burst.
- The average/min/max/std length of the separation between bursts in the past 1 second. The separation length of two consecutive bursts is the difference between the timestamp of the last packet of the previous burst and the first packet of the next burst.
- The number of retransmission streams - this means that if one or more retransmission streams appear alongside the main video stream, we can infer that the service has detected a disturbance that it can defend against by starting a retransmission stream.

3.4.3 WebRTC logs

Numerous video conferencing services use the WebRTC framework, so as the ones we used. The benefit of this framework is that the session logs can be saved and analyzed later. A useful feature in Chrome browsers (but only in Chrome browsers) is that by opening *chrome://webrtc-internals*, the current sessions can be seen and downloaded. At the end of the conferencing session, we can download the log file, which is a .txt file called *webrtc-internals-dump.txt*. This log contains different parameters, such as round-trip-time, jitter, the number of decoded frames per second, inter-frame delay, or the sent packets. Our tool automatically downloads these WebRTC logs and saves them next to the .pcap file and the radio parameters. This file also needs a processing step, so we created a script, which performs the conversion from the raw txt file to a csv file with the timestamps for each data point. A list and definition of the reported metrics is available on the following websites. A summary of the metrics [1] is available through the official documentation.

The key feature of the WebRTC logs is granularity. Each second, a large set of parameters can be retrieved and saved, for both the audio and the video streams. However, the WebRTC logs are limited to 1000 data points, which is approximately 16.66 minutes. This means that our measurements need to be shorter than 16 minutes, in order to keep all the information from the WebRTC logs. We designed our measurements according to this limitation, which is described back in Sec. 3.2.

[1] https://www.w3.org/TR/webrtc-stats/#stats-dictionaries

Our research focuses on in-network and radio parameters, however, we used the WebRTC parameters during the machine-learning process, too. The important WebRTC parameters are listed below. Note that these parameters are describing the video stream only. We believe that the following parameters are the most important WebRTC parameters in order to build an accurate QoE model:

- Jitter - the variance of delay
- JitterBufferDelay - the elapsed time amount between the first and the last packet's arrival time that belongs to the given frame
- FramesDropped - the number of frames dropped (not packets)
- RoundTripTime - end-to-end latency
- InterFrameDelay (standard deviation) - the elapsed time between two consecutively decoded frames
- FreezeCount - Count the total number of video freezes experienced by this receiver
- PauseCount - Count the total number of video pauses experienced by this receiver
- FramesPerSecond The number of decoded frames in the last second.
- FramesDecoded - Cumulative number of decoded frames
- PacketsLost - Number of packets dropped in the last second.

We carefully selected this group of parameters from each data source, in order to build an accurate QoE model. To provide a ground truth, we created a human QoE survey, in which participants rate the video and audio quality. This survey will be described in the next section, in Sec. 4.2.

Using the scores from this survey, we could train our models to predict the QoE values based on transport, radio and WebRTC parameters. Also, we tried using fewer data sources for the model and compared their results: e.g. using only radio, or transport parameters to predict the QoE. The detailed process of the model-building will be described in Chapter 5.

Chapter 4

Creating a unique dataset

In this chapter, we present the unique dataset we created during our work. The dataset is the output of our data collection system, however, in addition to the metrics described in Section 3.4 we created a QoE survey to obtain real subjective QoE values. This dataset is used to train and test the QoE model presented in Chapter 5.

4.1 Performed Measurements

We performed more than 300 measurements for this study, using different degradation scenarios and different video conferencing services. Some of them were experimental measurements, thus the final dataset consists of 173 video segments. Not only the amount of data was important, but we created a structured dataset that can be used easily for model building. The measurements follow a pattern and they were produced in the same environment: using the same notebook and the same geolocation, as it was previously described in Section 3.2.

We have prepared measurements for the three most important video conferencing services, which are:

- Microsoft Teams
- Google Meet
- Jitsi Video Conferencing

The goal was to create diversity in both services and the type of video content. For this purpose, we chose three different video types for the conferences. The three videos have significant differences regarding the content: the first video (Video A) is a man talking without quick cuts and scene changes, and here the importance of the audio quality is significantly greater than in the other two. The second video (Video B) is a trailer video for a popular series with quick cuts, interesting sound effects, and intensive scenes. The third video (Video C) is showing a waterfall, a river, and other natural scenes with calm background music. Using these videos we can analyze whether the human QoE values (the manual labeling MOS values) vary for different types of video content with the same network conditions.

	Video	Audio	FPS
Video A	Few cuts	Important	Less important
Video B	Many cuts	Important	More important
Video C	Few cuts	Less important	More Important

Table 4.1: Audio and video preferences of the test videos

In Table 4.1, we collected the audio and video preferences of the videos we used during our research.

Of the 176 video segments in the QoE survey, three videos were the original videos – without streamed through the video conference service –, 12 videos had no degradation and the remaining videos are grouped according to Table 4.2. Videos were distributed proportionally by content (i.e., the aforementioned 3 video types) in the survey. Four types of degradation (bandwidth reduction, jitter, packet loss, and radio shielding) were applied in the measurements to varying degrees. For the radio measurement we used a metal box to induce shielding, while for the other measurements, we used the tc tool.

Radio shielding	**Packet Loss (%)**			**Jitter (ms)**			**BW (Mbps)**		
	10	**20**	**30**	**100 ±50**	**150 ±75**	**200 ±100**	**1.5**	**1**	**0.5**
15	16	17	17	16	17	15	16	16	16

Table 4.2: Number of degradated measurements of different levels and types

The proper function of the manipulators can be observed in the following examples below:

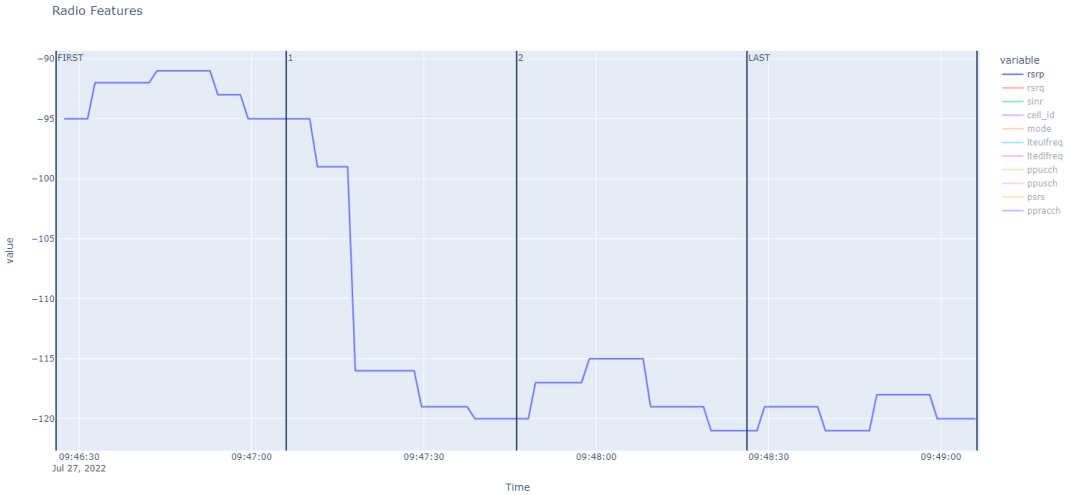

Figure 4.1: RSRP [dBm] change captured with the test system with radio shielding applied at the first horizontal line

An example of a dropping RSRP value (caused by a successful radio degradation i.e., shielding) can be seen in Figure 4.1. This example shows that the method we use for radio shielding usually results in a -10 dBm decrease on average.

Not only the RSRP value is changing when applying radio shielding, but the uplink transmitted power, too. It is inversely proportional to the RSRP value, which means that radio

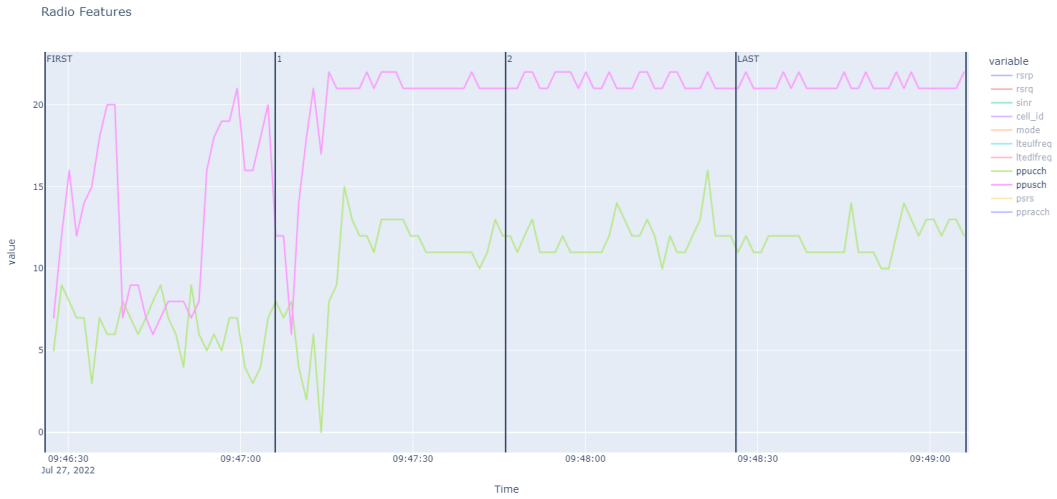

Figure 4.2: Uplink transmission power [dB] values with radio shielding applied at the the first horizontal line. Purple = Physical Shared Channel, Green = Physical Control Channel

shielding results in increased TXPOWER values. Figure 4.2 shows the uplink transmitted power values for two channels.

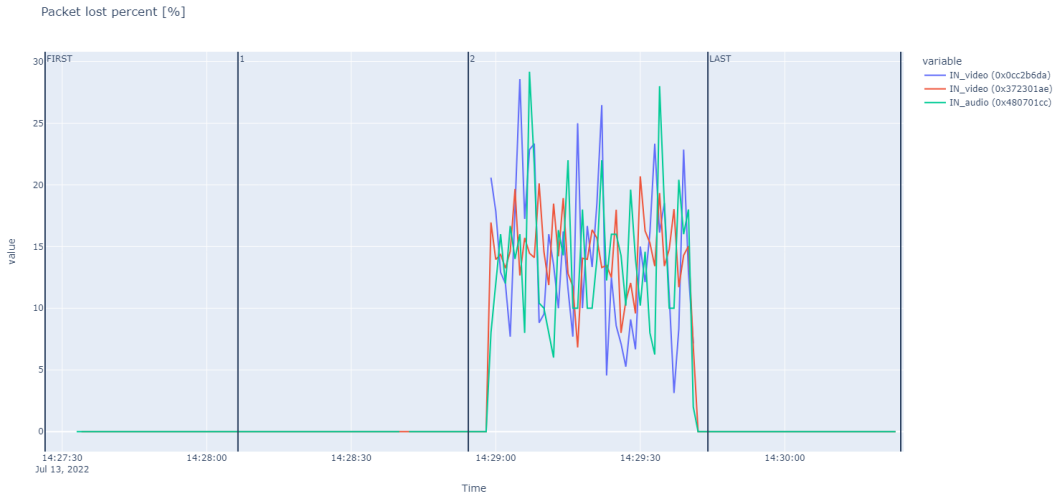

Figure 4.3: Example: Artificial packet loss [20%] for video and audio streams and the appearance of a retransmission stream (blue)

The artificial packet loss that we initiated (20%) can be seen in Figure 4.3, which also shows the appearing retransmission stream (blue) during the degraded period. Of course, the performance of any other manipulator could be visualized in the same manner, these are only some examples.

4.2 QoE survey

For the subjective QoE values, we created a survey, where the participants had to rate the video and the audio quality of the 10-second video segments. These video segments were

randomly chosen and ordered from our dataset, and the participants were not informed whether they are watching an original video or a video that was streamed through the conference server with or without degradation. The survey was implemented following the *ITU-T-P910 recommendation* [49].

The scores from this manual survey were used as ground truth values for the training process of our models. Using this survey, screen recordings from the automated measurements were used to manually label video segments for later use. For this purpose, we used the mean opinion score of each video. In further analyses, data collected from other sources (packet capture, WebRTC logs, and radio data) were used for this 10-second segments only.

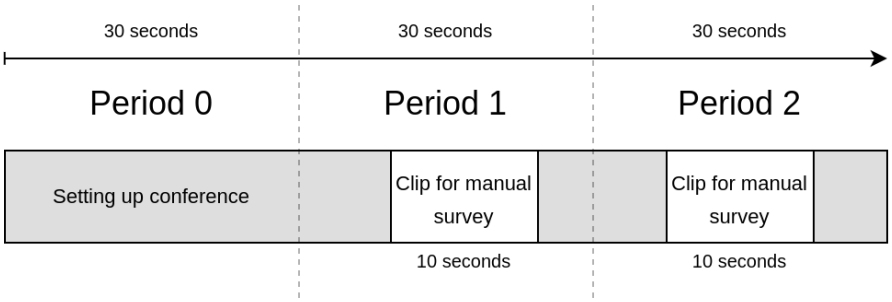

Figure 4.4: Manual QoE Survey - Video Structuring

The survey consists of 176 unique video segments, from different measurements on different services and different videos including scenarios both with and without degradation. Each video was shown twice to each participant at different points of time of the test, allowing us to measure their consistency.

As it can be seen in Figure 4.4, we used 10-second video segments from the second and the third period, making sure the service is up, and running, and the video streaming is working. Also, the 10-second segments do not start right after the new period starts, making sure that the degradations are set up for Period 1, and cleared for Period 2. For this purpose, a Python script was developed which cuts the videos at the necessary timestamps and creates the polls which could be used later.

The survey was divided into four tests, two contain videos from Teams and Jitsi, and the other two contain videos from Jitsi and Google Meet. We used absolute category rating test method with five-level scale for rating video and sound quality. A screenshot from the survey can be seen in Figure 4.5.

There were a total of 16 participants who completed at least one of the four tests. The participants were of all ages and genders. For each video segment, the average of the respondents' audio and video quality responses was assigned, occasionally omitting responses due to inconsistency.

The distribution of QoE values for each video segment is shown in Figure 4.6 for different services and different degradation types and levels. It can be seen that Google Meet is the most sensitive to network degradations. This is most apparent when jitter is induced as the overall user experience is consistently worse than the other two services, but the difference is also visible for packet loss. Interestingly services are most sensitive to very low bandwidth and radio shielding, rather than packet loss or jitter. The claim of [40]

Figure 4.5: Screenshot from the manual QoE survey

that there is no significant quality degradation at low packet loss can be supported. In terms of video content, the mean and variance of the user experience of the trailer and the nature video measurements are the same, but the variance of the opinion scores for the interview video is much higher.

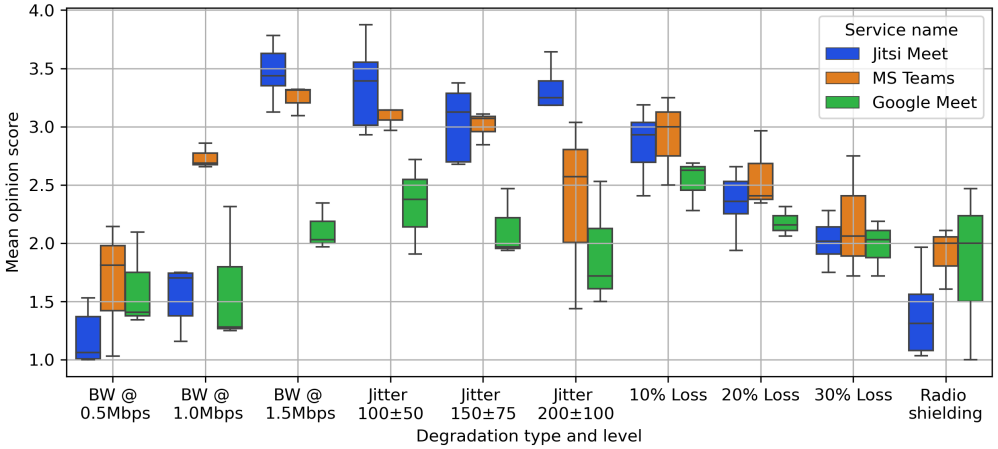

Figure 4.6: Distribution of QoE for different services and degradation types. (QoE scores for Google Meet, Jitsi and MS Teams without degradation are respectively: 2.2 ± 0.37, 3.5 ± 0.52, 3.1 ± 0.37)

The unique dataset we created is used to train, and test the QoE model, presented in Chapter 5. The model is compared with the previously mentioned models in Section 2. The performance of our model is evaluated with different parameter inputs.

Chapter 5

The novel Machine Learning based QoE model

In this chapter, the results of the correlation analysis and the QoE model are presented. First, the preprocessing of the data is presented, and then the correlation between the collected data and the degradation induced during the measurement is discussed using all measurements. Using these findings and results QoE models are built using machine learning algorithms. Finally, the performance of the models is presented and compared with similar models found the literature.

5.1 Data preprocessing

In order to perform correlation analysis and train machine learning algorithms, the dataset from the three different sources must be transformed into a single time-series dataset such that the time between two consecutive data points is the same and there are no missing values. The difficulty in building such a dataset is that there are multiple data streams at a given time: audio and video streams.

Thus, a data structure is designed where a given data point is identified by the data stream's SSRC in the RTP header and the time T, where T has a resolution of seconds. A record in the dataset contains all packet-level and WebRTC log data recorded at time T of a given stream, and also the radio data measured at time T.

This requires that all three datasets for a given measurement are transformed to a granularity of seconds and the missing values to be imputed.

The packet-level data is aggregated with one-second time windows, as described in Section 3.4.2, therefore meeting the requirements. The missing data is replaced by zeros in the packet-level data.

The data from WebRTC logs are generally saved with a granularity of 1 second, however, this is only true on average, usually varying by a few milliseconds. So to be consistent with the packet-level data, these time series are resampled. For continuous metrics, we interpolate the in-between data points, and for discrete metrics, we use the last valid value to resample and impute missing values.

As briefly discussed in Section 3.4.1, radio metrics are not consistently updated every second. It is therefore particularly important in this case to resample the time series. A similar approach was taken for the WebRTC data when using interpolation techniques

for resampling and imputation. However, note that in the present dataset, all radio parameters were assumed to be discrete, even RSRP since it is always an integer.

5.2 Correlation analysis

In order to get an understanding of the importance and usefulness of each of the metrics measured, and to prioritise them for further model building, correlation analysis was performed. The analysis was performed for different disturbance types using all measurements. In this analysis, we did not correlate the metrics with the MOS values, rather with the step functions describing the degradation, which are detailed below. This allowed us to investigate correlation on a larger sample, resulting in a more reliable analysis.

Several methods for correlating time series were tested: Euclidean distance, Pearson correlation, Dynamic Time Warping (DTW). Euclidean distance is of limited use for comparing time series because it is sensitive to noise, time warping and scaling. Pearson correlation is well suited when there is only a difference in scale and/or noise in the time series, but it is not applicable when there is a time lag or warp. The DTW method works very well in cases where there is a time warp or time shift, but when there is a discrepancy in the time series' scale, some kind of normalization is needed to obtain comparable results. Given the above considerations, we decided to use Pearson correlation, as there is no time lag in the measurements, whereas noise and scale differences are observed.

To perform the correlation analysis, a pseudo time series was created for each measurement, showing the amount of degradation as a function of time, i.e. the bandwidth reduction, packet loss, jitter and radio shielding currently applied. For example, for the bandwidth reduction measurements, the following function was used:

$$degradation_{bw}(t) = \begin{cases} 0.5 & \text{if bandwidth is reduced to 0.5Mbps at time } t \\ 1 & \text{if bandwidth is reduced to 1Mbps at time } t \\ ... & \\ 100 & \text{otherwise} \end{cases} \tag{5.1}$$

For the packet loss measurements, for example, the following function was used:

$$degradation_{loss}(t) = \begin{cases} 20 & \text{if the induced packet loss is 20\% at time } t \\ ... & \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

A function with a similar logic was used for the jitter measurement. For the radio measurement, we used a binary function, which was 1 when the jitter was applied and 0 when it was not.

The metrics were correlated with these degradation step functions. For each measurement of a given type of perturbation, the cross correlation was calculated separately. To simplify the analysis of the metrics responding to a given type of disturbance, the mean, median and standard deviation of the correlations of these measurements were also calculated. Figure 5.1 shows the average correlation of all packet loss measurements. The degrad_loss function shown in the bottom row was of interest to us.

Figure 5.2 shows the correlation with the degrad_loss function for different levels of packet loss. The figure shows that some metrics correlate better the higher the packet loss, for

Figure 5.1: Average correlation coefficients for packet loss measurements

example the average of Bitrate and inter arrival time. However, some metrics are highly correlated regardless of the amount of packet loss, such as the number of retransmission streams and the number of packets sent per second. This phenomenon is discussed in the summary of the correlation analysis.

These plots and analyses were performed separately for each of the jitter, packet loss, bandwidth reduction and radio perturbation measurements. The following phenomena and conclusions could be drawn.

One of the most important phenomena observed when testing all three services is that when the client detects poor network conditions, it sends frequent RTCP control messages to the sender, which in response retransmits the video content in a separate stream with a separate SSRC. In this way, the service sends information redundantly in an attempt to maintain quality of service. The claim of [40] that there is no significant quality degradation at low packet loss can be supported, that is the reason why we investigated high packet loss rates.

This phenomenon has always occurred, with the exception of radio perturbations. There are two ways to detect this phenomenon from packet-level data: 1) from the number of RTCP messages sent by the client, 2) from the appearance of a new video stream with a new SSRC. in Figure 5.2, these two metrics we have created are "PCAP_retrans_streams" and "PCAP_packet_sent_per_sec".

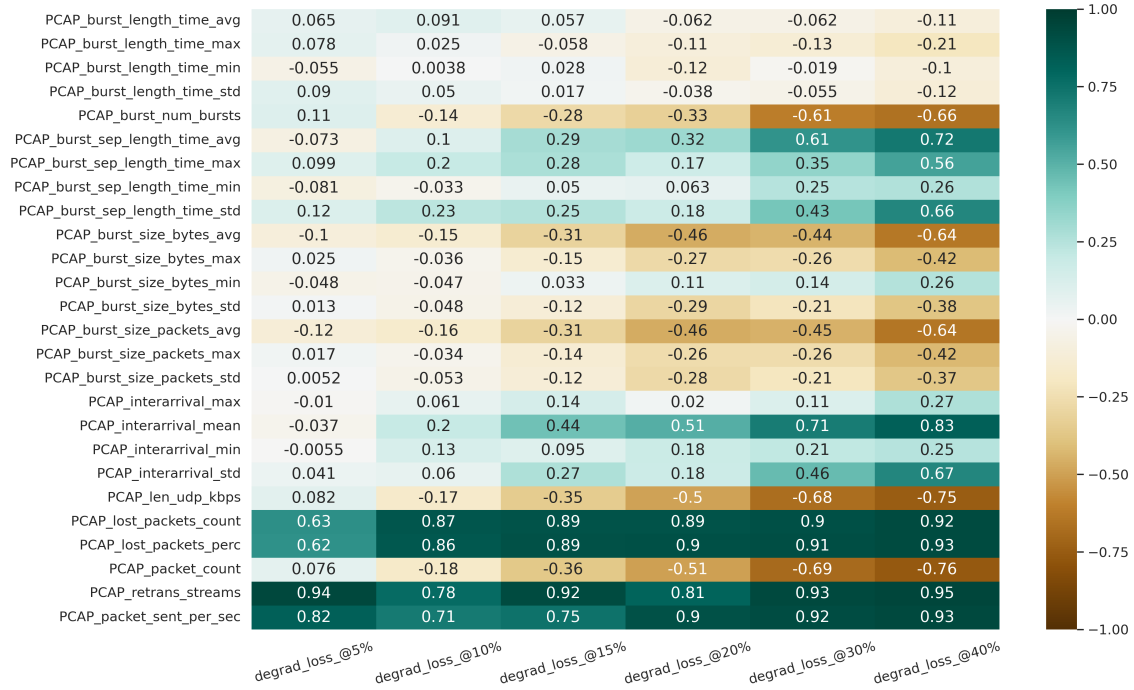

Figure 5.2: Correlation of packet level metrics at different added packet loss percentage

Furthermore, we observed the bandwidth reduction induced by the TC tool actually causes traffic shaping. Thus, the bandwidth reduction is induced by buffering incoming packets in a queue and releasing them at specified intervals. However, if this buffer becomes full, it will maintain the specified reduced bandwidth by dropping packets. From this operation, the video stream loses its burstiness and packet loss occurs. Thus, we are able to detect artificially induced bandwidth reduction by monitoring the burst metrics and packet loss. Since routers and other network devices may employ a similar method, using this dataset we are able to detect artificial bandwidth reduction.

Through a few example measurements, we will illustrate the most important relationships between the different degradation types. Figure 5.3 demonstrates a packet loss measurement. It can be seen that when we increase the packet loss to 20% the inter-arrival time increases, the bitrate of the video stream decreases and the client starts sending RTCP messages frequently to the other party.

Figure 5.4 demonstrates a jitter measurement. It can be seen that when we increase the jitter the number of bursts per second for the video packets drops significantly, the standard deviation of the inter-arrival time for the audio packets decreases and the standard deviation of the inter frame delay increases.

Figure 5.5 demonstrates a jitter measurement. It can be seen that when we applied radio shielding the signal power drops significantly, and handover occurs multiple times, thus creating packet loss. Furthermore, the uplink transmitted power is also increasing – reaching its maximum – to overcome the shielding effect.

Finally, the average correlation of the most important metrics for the different disturbances is summarised in Figure 5.6. These are potentially well-suited for QoE estimation.

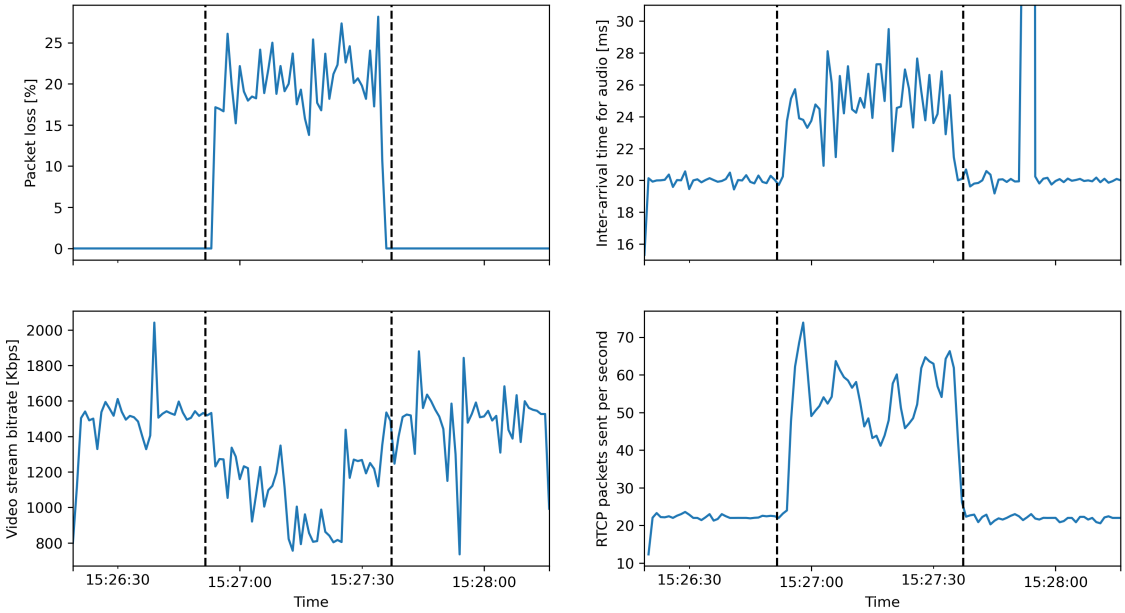

Figure 5.3: Metrics for a packet loss measurement. (Vertical dashed lines indicate the start of a period)

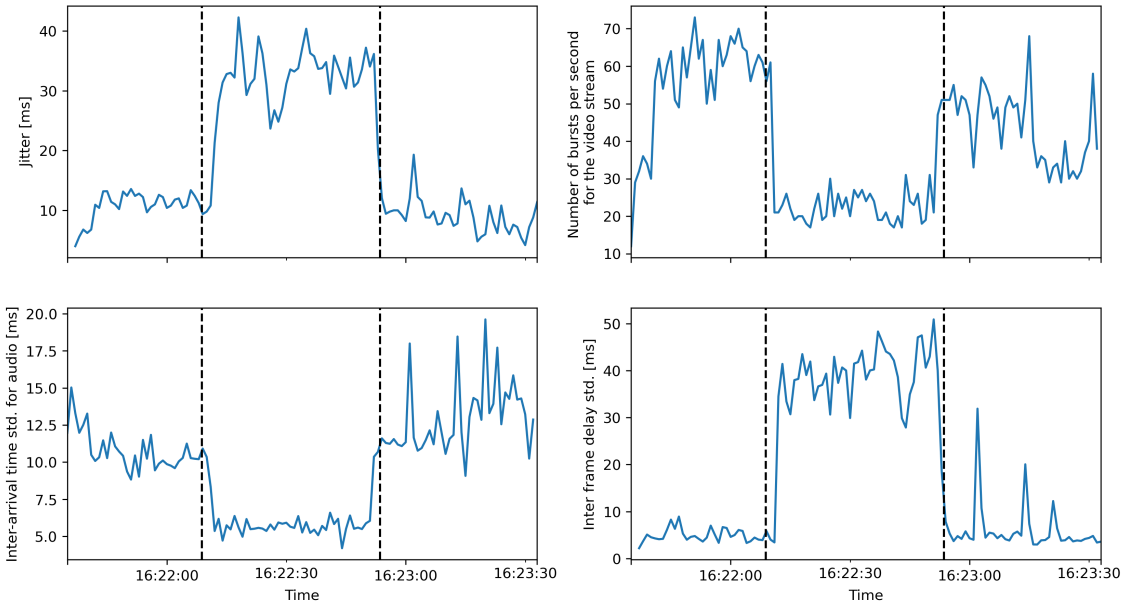

Figure 5.4: Metrics for a jitter measurement. (Vertical dashed lines indicate the start of a period)

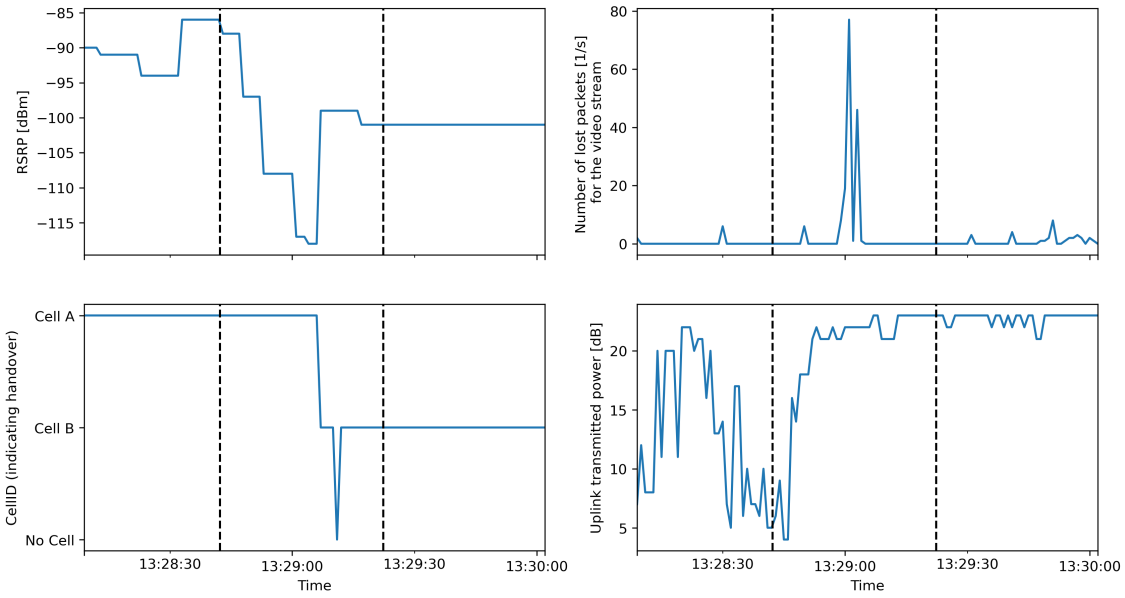

Figure 5.5: Metrics for a radio measurement. (Vertical dashed lines indicate the start of a period)

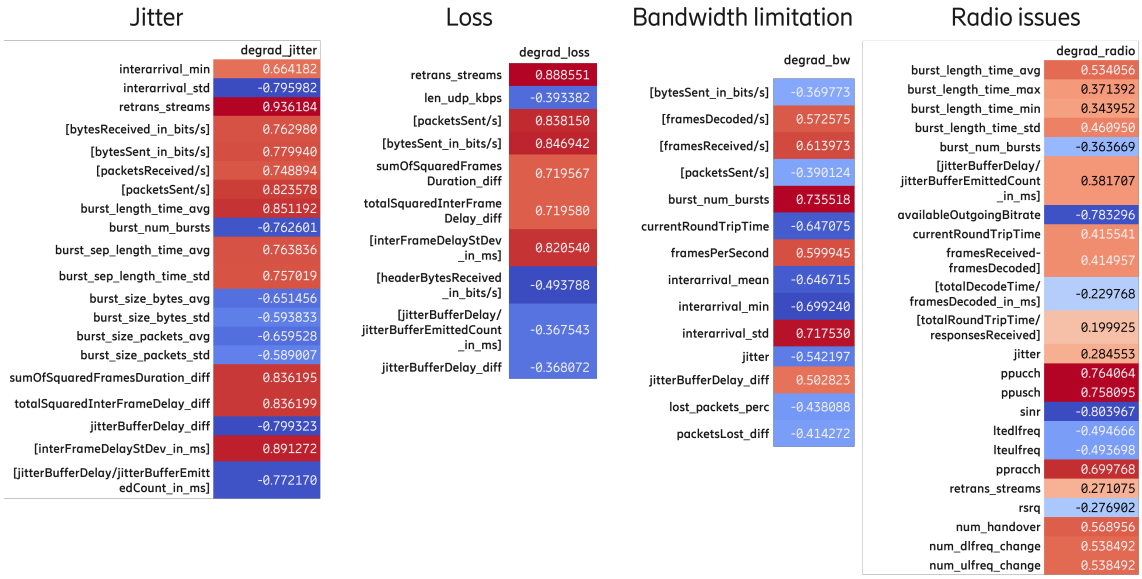

Figure 5.6: The most strongly correlated metrics and correlation coefficient for each degradation type.

5.3 QoE model

5.3.1 Model training methodology

The dataset generated by the method discussed in Section 5.1 is used to train the models. However, the time-series data cannot be fed to the supervised machine learning algorithms unchanged, it must be transformed to obtain a dataset with independent data points. For this transformation, using Takens' theorem [46], time-delayed embedding was performed. According to this, for a given data point at time T, we assign $T-1, ..., T-i$ data points. Furthermore, the mean and the standard deviation of these i data points are added. Thus, for a data point at time T, we assign the immediate past data points, obtaining a multidimensional dataset in which the data points can be considered independent of each other and can be processed by supervised machine learning algorithms.

Since our goal is to create a model that estimates the QoE from the collected network and radio data sources, subjective QoE values from the survey must be assigned. Therefore, only the 10 seconds of data from the measurements are retained where MOS values are available, in accordance of the survey in Section 4.2. Thus, there are 10 data points (related to the seconds) per measurement with a ground truth value, which later becomes the target variable. Note that since in the survey the respondent gave an opinion for the whole 10 second video segment, we assign the same MOS value to all 10 data points per measurement.

After discarding the control original videos and a few bad measurements, our dataset contained the embedded metrics and MOS values for 171 video segments, resulting in 1710 datapoints. In order to produce the best-performing model, we cross-validated several hyperparameter values. For cross-validation, we used Stratified K-Fold procedure. In this process, the dataset is split into K subsets, where over K iterations, one subset of the dataset becomes the test set and the remaining subset of the dataset becomes the training set. However, when choosing the K subset, care is taken to ensure that there is roughly the same number of each measurement type in each subset.

The hyperparameters altered during cross-validation are listed and explained in more detail below:

1. Data sources used
2. Time-delayed embedding parameters
3. Feature selection method
4. Hyperparameters of the machine learning model

In varying the data sources used, we investigate whether the use of all three data sources provides better results than using only radio and packet-level metrics, which are available to a network operator in the middle of the network.

The three parameters of Time-delayed embedding are time-delay, dimension and stride. The meaning of these parameters can be seen in the original article [46].

Feature selection is used because machine learning models have difficulty finding the optimum in the high dimensional dataset and cannot work efficiently due to redundant features. We used 4 different methods: all feature retention, Lasso method, the SelectKBest method of the Sklearn Python library [42] and the Python featureWiz library [12]. These

methods select the most important features from the hundreds of input parameters, then only these are used for training the models.

Lastly, the hyperparameters of the machine learning algorithm are set. Linear regression was used as the baseline solution, here the regularization used was the hyperparameter. In addition, we used Gradient Boosting Machine (GBM) which has a large number of hyperparameters. Only the following were changed: "max_depth", "learning_rate", "max_features", "min_samples_split", "min_samples_leaf", "n_estimators"

In order to find the best-performing model we utilized the SkOpt Python library [13]. It uses Bayesian optimization using Gaussian Processes to find the best hyperparameters. During training, the performance of several regression models was tested: linear regression, RandomForest, AdaBoost, GradientBoostingMachine (GBM). The results of the modelling exercise are detailed below.

5.3.2 Evaluation

In order to get an understanding of the performance of our models, we compared them with similar models in the literature. As discussed in Section 2.3.2 the QoE model developed by Husić et. al. utilizes the internal WebRTC parameters only and has only four variables - two for the video and two for the audio. Their work is slightly different compared to ours since they used two-way video conferencing and gave a QoE value for the whole conversation. The authors took into account both incoming and outgoing video quality, while we only consider incoming video quality. Thus, for the evaluation, we assume that the quality of the outgoing and incoming video is the same: $VideoStream\_PacketsSentPerSec$ and $VideoStream\_PacketsReceivedPerSec$ are equal in Equation 1 of the original article [29], moreover, one QoE value is predicted for one video segment. This model was implemented and evaluated on our dataset. Another QoE model is used for benchmarking. The model provided by Ericsson – which was discussed in 2.3.1 – utilizes WebRTC parameters as well. This approach is based on the bitrate, jitter and packet loss, similar to the previous model, although other parameters play an important role, which depends on the video codec.

To evaluate our models, we present the results of 9 model variations: 1) GBM trained using all 3 data sources, 2) GBM trained using both radio and transport metrics, 3) GBM using only radio metrics, 4) GBM using only transport metrics, 5) Linear regression using all 3 data sources 6) Linear regression using radio and transport metrics, 7) Linear regression using the same parameters as Husić et. al., 8) The QoE model of Husić et. al., 9) The QoE model provided by Ericsson.

Table 5.1 shows the performance of the models, the accuracy was evaluated using several metrics: RMSE (Root Mean Squared Error), MAE (Mean Absolute Error), and R^2 (Coefficient of Determination). Figure 5.7 shows the cumulative distribution function of the absolute error of each model.

As the table and the figure show, the reference models cannot give a good estimate in our case, where we are predicting the user experience of a video conference on a 4G network on various network conditions with a one-second resolution. Model 7) confirms that using only 4 parameters is not sufficient, thus the solution of Husić et. al. cannot give good predictions. The model provided by Ericsson gave spectacularly poor results. This might be because the model used video codec-dependent parameters. These parameters are strongly video content-dependent based on our observations.

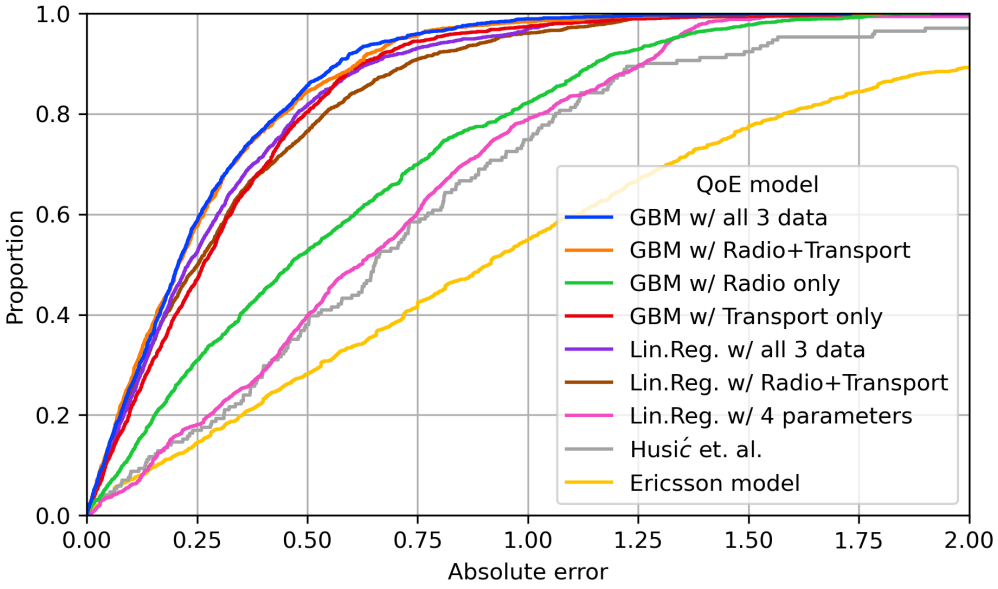

Figure 5.7: Cumulative distribution of the absolute error for each model.

No.	QoE Model	RMSE	MAE	R^2
1	GBM with all 3 data sources	0.35	0.27	0.80
2	GBM with radio & transport	0.36	0.27	0.78
3	GBM with radio only	0.69	0.55	0.21
4	GBM with transport only	0.41	0.32	0.72
5	Lin.Reg with all 3 data sources	0.41	0.30	0.73
6	Lin.Reg with radio & transport	0.44	0.33	0.68
7	Lin.Reg with 4 parameters	0.77	0.66	0.02
8	Husić et. al.	0.88	0.72	-0.27
9	Ericsson QoE model	1.27	1.03	-1.59

Table 5.1: Performance of the models

Furthermore, we can state that the combination of transport, radio and WebRTC parameters gives better results than models that do not use all three data sources. It can be seen that model 1) is better than model 2) and model 5) is better than model 6) using the same machine learning algorithm. The complexity of the problem is shown by the fact that using the same parameters, a more complex GBM (i.e. model 1) can achieve much better results than simple Linear Regression (i.e. model 5). Note that model 1 and model 2 achieve very similar accuracy, although model 1 performs slightly better. We can conclude from this that we can give very good predictions with both radio and transport data. Thus, model 2 can even be used by Mobile Network Operators, since it only requires the collection of radio and core network QoS parameters.

As the results show the various machine learning models that were trained and optimized, model 1) was the most accurate. The ground truth values and the QoE predictions of the trained Gradient Boosting model are shown in Figure 5.8. The dashed line in the figure indicates a deviation of ± 0.5 from the ground truth value, which is considered as an acceptable error. Figure 5.9 shows the distribution of the absolute error made by model 1) for different services and degradation types. An interesting observation is that for very

strong degradations (radio shielding, $200ms \pm 100$jitter, 500Kbps bandwidth reduction) the model inaccuracy variance is relatively larger. Furthermore, no consistent service dependence is observed, i.e. the model is able to estimate QoE with the same accuracy for all services

Furthermore, the MAE values based on video content, for Video A, B and C are 0.31, 0.24, 0.25 respectively. Meaning that it is more difficult for the model to predict the user experience of the interview video (Video A) in terms of video content.

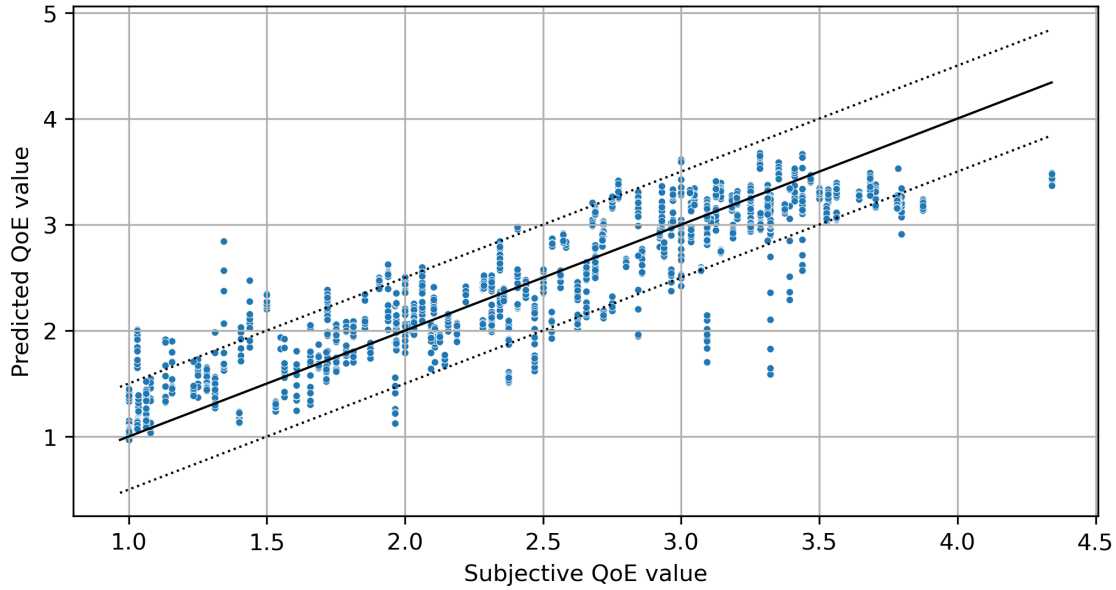

Figure 5.8: Prediction and ground truth values for the two GBM model.

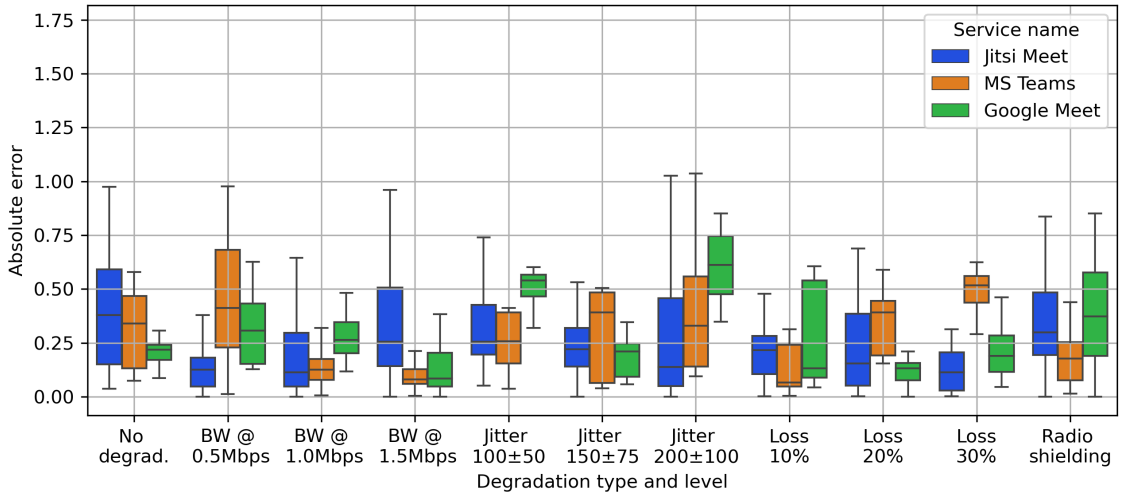

Figure 5.9: Distribution of absolute error for different services and degradation types.

Model 1) was trained by using a 10-second time window for time-delayed embedding. These proved to be useful parameters when examining the feature importance of the model. The most important parameters included the 10-second averages of the following parameters: audio and video inter-arrival time, video packets per second, audio bitrate,

number of bursts in a second and the average duration of a burst of the video packets, audio and video lost packets per second and the number of retransmission streams. Furthermore, the value of RSRP and SINR at time T. The most important WebRTC parameters were: number of packets leaving the jitter buffer per second, delay of the jitter buffer for audio and video packets, decode time of the frames, interruption duration for the audio stream, frames per second, round trip time, standard deviation of the inter-frame delay and concealed events per second for the audio stream. For detailed definitions of metrics, see Section 3.4.

In summary, we built several models and compared their results with available QoE models. We have shown that using radio and transport parameters available to MNOs, our proposed model achieves promising accuracy: a root mean squared error of 0.36. However, if WebRTC logs are also available – collected on the end-user's device – , even better accuracy can be achieved, with a root mean squared error of 0.35 for our proposed model.

Chapter 6

Summary

The aim of our work was to create a data collection system to collect transport and radio data about delay critical services under different network conditions in an automated and standardized way.

Measurements have been conducted in a commercial 4G mobile network using different video conferencing applications (Google Meet, Jitis, MS Teams) with different media content using different network degradation (bandwidth reduction, packet loss, jitter, radio shielding). A standardised survey was developed and used to assign user experience values to the measurements. From examining the QoE values, it is concluded that Google Meet is generally worse at tolerating network degradation than the other two services.

The collected data and QoE values were used to build several models to predict QoE values. Results showed that for predicting E2E service quality using radio and transport parameters available to MNOs, our proposed model achieves promising accuracy: a root mean squared error of 0.36. However, if WebRTC logs are also available – collected on the end-user's device – , even better accuracy can be achieved, with a root mean squared error of 0.35 for our proposed model.

Bitrate, inter-arrival time, burstiness metrics and packet loss as transport parameters and RSRP and SINR radio parameters are important as these can be monitored for encrypted traffic network-wide by MNOs as well. As for WebRTC logs, parameters regarding the jitter buffer, round trip time and various video frame decoding and audio sampling metrics have been identified as important.

The recommended data collection system, QoE survey and modelling methods described in this paper can be used for other delay critical services and future URLLC services.

Acknowledgements

The research leading to these results was supported by Ericsson and the High Speed Networks Laboratory (HSNLab). We would like to thank our advisors and also Attila Báder from Ericsson.

Bibliography

[1] Lte channels. URL https://www.4gmobiletech.com/lte-channels. [Online; Accessed: 2022-05-30]".

[2] Cellular network figure. URL https://www.researchgate.net/figure/A-topology-of-a-cellular-network_fig2_263316201. [Online; Accessed: 2022-05-30].

[3] Webrtc figure. URL https://www.researchgate.net/figure/WebRTC-triangle-architecture-27_fig3_328334940. [Online; Accessed: 2022-05-30].

[4] Wireshark. URL https://www.wireshark.org. [Online; Accessed: 2022-05-30].

[5] Google meet hardware requirements. URL https://support.google.com/a/answer/4541234?hl=en. [Online; Accessed: 2022-10-16].

[6] Ssim: Structural similarity index. URL https://www.imatest.com/docs/ssim/. [Online; Accessed: 2022-05-30].

[7] Description of video quality metric (vqm) software. URL https://its.ntia.gov/research-topics/video-quality-research/guides-and-tutorials/description-of-vqm-tools/. [Online; Accessed: 2022-10-30].

[8] Tshark. URL https://www.wireshark.org/docs/man-pages/tshark.html. [Online; Accessed: 2022-05-30].

[9] Webrtc 1.0: Real-time communication between browsers. URL https://www.w3.org/TR/webrtc/. [Online; Accessed: 2022-05-30].

[10] New messenger desktop app for group video calls and chats, May 2020. URL https://about.fb.com/news/2020/04/messenger-desktop-app/. [Online; Accessed: 2022-09-20].

[11] About jitsi meet: Free video conferencing solutions, Feb 2021. URL https://jitsi.org/jitsi-meet/. [Online; Accessed: 2022-10-12].

[12] Autoviml/featurewiz: Use advanced feature engineering strategies and select best features from your data set with a single line of code., Oct 2022. URL https://github.com/AutoViML/featurewiz. [Online; Accessed: 2022-11-01].

[13] scikit-optimize: Sequential model-based optimization in python., Oct 2022. URL https://scikit-optimize.github.io/stable/. [Online; Accessed: 2022-11-01].

[14] Video multimethod assessment fusion, Oct 2022. URL https://en.wikipedia.org/wiki/Video_Multimethod_Assessment_Fusion. [Online; Accessed: 2022-10-30].

[15] Mohammed Alreshoodi and John Woods. Qoe prediction model based on fuzzy logic system for different video contents. In *2013 European Modelling Symposium*, pages 635–639. IEEE, 2013.

[16] Doreid Ammar, Katrien De Moor, Min Xie, Markus Fiedler, and Poul Heegaard. Video qoe killer and performance statistics in webrtc-based video communication. pages 429–436, 07 2016. DOI: 10.1109/CCE.2016.7562675.

[17] Sandra Lam Andrea Lai and Rachita Navara. A faraday cage exploration.

[18] Mark Baugher, David McGrew, Mats Naslund, Elisabetta Carrara, and Karl Norrman. The secure real-time transport protocol (srtp). Technical report, 2004.

[19] Enrico Bocchi, Luca De Cicco, and Dario Rossi. Measuring the quality of experience of web users. *SIGCOMM Comput. Commun. Rev.*, 46(4):8–13, dec 2016. ISSN 0146-4833. DOI: 10.1145/3027947.3027949. URL https://doi.org/10.1145/3027947.3027949.

[20] Paul Brodsky. Internet traffic and capacity in covid-adjusted terms, Sep 2020. URL https://blog.telegeography.com/internet-traffic-and-capacity-in-covid-adjusted-terms. [Online; Accessed: 2022-05-30].

[21] Jessica Delaney. Video conferencing: Who will lead 2021 (teams, meet, or zoom), Nov 2020. URL https://www.linkedin.com/pulse/video-conferencing-who-lead-2021-teams-meet-zoom-jessica-delaney/. [Online; Accessed: 2022-05-30].

[22] K.M. Dostert. Frequency-hopping spread-spectrum modulation for digital communications over electrical power lines. *IEEE Journal on Selected Areas in Communications*, 8(4):700–710, 1990. DOI: 10.1109/49.54466.

[23] Carmen Egido. Video conferencing as a technology to support group work: A review of its failures. In *Proceedings of the 1988 ACM Conference on Computer-Supported Cooperative Work*, CSCW '88, page 13–24, New York, NY, USA, 1988. Association for Computing Machinery. ISBN 0897912829. DOI: 10.1145/62266.62268. URL https://doi.org/10.1145/62266.62268.

[24] EricssonResearch. EricssonResearch/spindump: Spindump is an in-network latency measurement tool with support for QUIC and TCP. https://github.com/EricssonResearch/spindump, 2021. [Online; Accessed: 2022-05-30].

[25] Santosh Ganji, Romil Sonigra, and P. R. Kumar. Overcoming pedestrian blockage in mm-wave bands using ground reflections. *CoRR*, abs/2110.13884, 2021. URL https://arxiv.org/abs/2110.13884.

[26] Boni García, Francisco Gortázar, Micael Gallego, and Andrew Hines. Assessment of qoe for video and audio in webrtc applications using full-reference models. *Electronics*, 9(3), 2020. ISSN 2079-9292. DOI: 10.3390/electronics9030462. URL https://www.mdpi.com/2079-9292/9/3/462.

[27] The Tcpdump Group. Home | TCPDUMP & LIBPCAP. https://www.tcpdump.org/, 2022. [Online; Accessed: 2022-05-30]".

[28] Josh Howarth. Internet traffic from mobile devices (2022), Aug 2022. URL https://explodingtopics.com/blog/mobile-internet-traffic. [Online; Accessed: 2022-10-12].

[29] Jasmina Baraković Husić, Adna Alić, Sabina Baraković, and Mladen Mrkaja. Qoe prediction of webrtc video calls using google chrome statistics. In *2021 20th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pages 1–6, 2021. DOI: 10.1109/INFOTEH51037.2021.9400661.

[30] Iperf3. iPerf - Download iPerf3 and original iPerf pre-compiled binaries. https://iperf.fr/, 2022. [Online; Accessed: 2022-05-30]".

[31] Luis Roberto Jiménez, Marta Solera, and Matías Toril. A network-layer qoe model for youtube live in wireless networks. *IEEE Access*, 7:70237–70252, 2019.

[32] P. Jung, P.W. Baier, and A. Steil. Advantages of cdma and spread spectrum techniques over fdma and tdma in cellular mobile radio applications. *IEEE Transactions on Vehicular Technology*, 42(3):357–364, 1993. DOI: 10.1109/25.231889.

[33] Asiya Khan, Lingfen Sun, and Emmanuel Ifeachor. Qoe prediction model and its application in video quality adaptation over umts networks. *IEEE Transactions on Multimedia*, 14(2):431–442, 2011.

[34] Roman Krzanowski. Burst (of packets) and burstiness. In *66th IETF meeting*, 2006.

[35] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, et al. The quic transport protocol: Design and internet-scale deployment. In *Proceedings of the conference of the ACM special interest group on data communication*, pages 183–196, 2017.

[36] Yan Liu, Yansha Deng, Maged Elkashlan, Arumugam Nallanathan, and George K. Karagiannidis. Analyzing grant-free access for urllc service. *IEEE Journal on Selected Areas in Communications*, 39(3):741–755, 2021. DOI: 10.1109/JSAC.2020.3018822.

[37] Nurul Huda Mahmood, Renato Abreu, Ronald Böhnke, Martin Schubert, Gilberto Berardinelli, and Thomas H Jacobsen. Uplink grant-free access solutions for urllc services in 5g new radio. In *2019 16th International Symposium on Wireless Communication Systems (ISWCS)*, pages 607–612. IEEE, 2019.

[38] Tariku Temesgen Mehari. Frequency hopping in lte uplink. 2009.

[39] MikePlumleyMSFT. Prepare your organization's network for teams - microsoft teams. URL https://learn.microsoft.com/en-us/microsoftteams/prepare-network.

[40] Ashkan Nikravesh, Qi Alfred Chen, Scott Haseley, Xiao Zhu, Geoffrey Challen, and Zhuoqing Mao. Qoe inference and improvement without end-host control. pages 43–57, 10 2018. DOI: 10.1109/SEC.2018.00011.

[41] Obsproject. Open Broadcast Software. https://github.com/obsproject/obs-studio, 2022. [Online; Accessed: 2022-05-30].

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[43] Nanditha Rao, A Maleki, F Chen, Wenjun Chen, C Zhang, Navneet Kaur, and Anwar Haque. Analysis of the effect of qos on video conferencing qoe. In *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 1267–1272. IEEE, 2019.

[44] Umme Sara, Morium Akter, and Mohammad Shorif Uddin. Image quality assessment through fsim, ssim, mse and psnr-a comparative study, Mar 2019. URL https://www.scirp.org/journal/paperinformation.aspx?paperid=90911.

[45] Henning Schulzrinne, Stephen Casner, Ron Frederick, and Van Jacobson. Rtp: A transport protocol for real-time applications. Technical report, 2003.

[46] Floris Takens. Detecting strange attractors in turbulence. In *Dynamical systems and turbulence, Warwick 1980*, pages 366–381. Springer, 1981.

[47] tc. Introduction to Linux Traffic Control. https://tldp.org/HOWTO/Traffic-Control-HOWTO/intro.html, 2022. [Online; Accessed: 2022-05-30]".

[48] Thombashi. Introduction to Linux Traffic Control. https://github.com/thombashi/tcconfig, 2022. [Online; Accessed: 2022-05-30]".

[49] International Telecommunication Union. ITU-T-P910. https://www.itu.int/rec/T-REC-P.910, July 2022. [Online; Accessed: 2022-10-26]".

[50] Lionel Sujay Vailshery. Microsoft teams daily active users worldwide 2022, Oct 2022. URL https://www.statista.com/statistics/1033742/worldwide-microsoft-teams-daily-and-monthly-users/. [Online; Accessed: 2022-10-12].

[51] Dunja Vucic, Lea Skorin-Kapov, and Mirko Suznjevic. The impact of bandwidth limitations and video resolution size on qoe for webrtc-based mobile multi-party video conferencing. pages 59–63, 08 2016. DOI: 10.21437/PQS.2016-13.