Stable Communication Over a Wireless Network Under Heavy Load

CARL NILSSON NYMAN
MATTIAS EKLUND
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

LUND UNIVERSITY

MASTERS THESIS

Stable Communication Over a Wireless Network Under Heavy Load

Author:
Carl NILSSON NYMAN,
Mattias EKLUND

Supervisor:
Dr. Björn LANDFELDT

A thesis submitted in fulfillment of the requirements for the degree of Master in Computer Science Engineering

for the department of

Electrical and Information Technology

December 17, 2018

Abstract

Carl NILSSON NYMAN, Mattias EKLUND

Stable Communication Over a Wireless Network Under Heavy Load

For our master's thesis we have worked with a company called Uniti to help them establish a system for communicating with their autonomous vehicle during a public demonstration. This included handling a video feed from the vehicle as well as assuring low latency for said feed. For the purposes of this thesis the targeted latency was below 200ms.

In order to construct a solution which met our requirements we looked at several combinations of wireless mediums and video encoding standards. For our thesis work we considered Wi-Fi and LTE as network mediums as well as JPEG and H.264 as video encoding standards.

We implemented several combinations of network mediums and video encoding standards to see whether or not they would perform well within the latency criteria. We then tested these solutions in a wireless environment with a moderate amount of load on it. During the tests we took sample data which we later analyzed. This data was in turn used to determine whether or not a certain solution was within margins for our project as well as reliable enough.

We found that a Wi-Fi connection yielded results within the required boundaries when using both JPEG and H.264 as the video coding standards, with varying but yet acceptable results. On the other hand LTE was found to deliver less desirable results although LTE could provide certain other benefits such as being functional at longer ranges.

Acknowledgements

We would like to thank our supervisor Björn Landfeldt and our examiner Christian Nyberg for their help on advising and assisting in formulating our thesis. We would also like to thank both Kristofer Jansson and Kristoffer Syversen as well as the rest of the team at Uniti Sweden AB for hosting us during our master thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

RTP	**R**eal-time **T**ransport **P**rotocol
LTE	**L**ong **T**erm **E**volution
FPS	**F**rames **P**er **S**econd
VOD	**V**ideo **O**n **D**emand
RTMP	**R**eal-**T**ime **M**essaging **P**rotocol
ms	**m**illi**s**econds
JPEG	**J**oint **P**hotographic **E**xpert **G**roup
H264	MPEG-4 Part 10, Advanced Video Coding
MPEG	**M**oving **P**ictures **E**xperts **G**roup
kB	**k**ilo **B**ytes
MB	**M**ega Bytes
ITU	**I**nternational **T**elecommunication **U**nion
ITU-T	ITU **T**elecommunication Standardization Sector
ITU-R	ITU **R**adiocommunications Sector
ISO	International Organization for Standardization
IEC	**I**nternational **E**lectrotechnical **C**ommission
RFC	**R**equest **F**or **C**omment
MJPEG	**M**otion JPEG
IEEE	**I**nstitute of **E**lectrical and **E**lectronics Engineers
ETSI	**E**uropean **T**elecommunications **S**tandard **I**nstitute
GSM	**G**lobal **S**ystem for **M**obile communications
EDGE	**E**nhanced **D**atarates for **G**SM **E**volution
UMTS	**U**niversal **M**obile **T**elecommunications **S**ystem
HSPA	**H**igh **S**peed **P**acket **A**ccess
HSPA+	Evolved HSPA
3GPP	**3**rd **G**eneration **P**artnership **P**roject
IMT	**I**nternational **M**obile **T**elecommunications
WiMAX	**W**orldwide **i**nteroperability for **M**icrowave **A**ccess
3G	**3**rd **G**eneration
4G	**4**th **G**eneration
csv	**c**omma **s**eparated **v**alues
DCT	**D**iscrete **C**osine **T**ransform
ROS	**R**obot **O**perating **S**ystem

List of Symbols

t_n	Timestamp, where n is either 1 or 2	ms
l	Latency	ms
f	Frames per second	frames/s
n_{frames}	Number of frames between timer update	
ϕ	1 second in desired unit[1]	
s	Standard deviation of latency	ms
x_i	Data point in set X	ms
$\overline{x}$	Mean value of data points in X	ms
N	Number of elements in X	absolute

[1] If the desired result should be given in seconds then $\phi = 1$, if the result should be in milliseconds then $\phi = 1000$

Chapter 1

Introduction

This chapter will provide an introduction to this work through supplying motivation and background. This is divided into the following parts:

Background This section describes the background to the project.

Related Work This part describes works and concepts which are related to our work.

Motivation Describes the basis and motivation for this project.

Problem Definition Provides a formal definition of the goal of this project.

Method This section describes the approach used to pursue the goal of this project.

Disposition Breaks down the report into different parts and explains each part.

1.1 Background

Automobile connectivity has become a more and more popular concept. In modern society there has been an increasing amount of functionality in vehicles which would have been impossible to perform just a decade ago. These functions range from improvements in handling a vehicle, to more easily integrated entertainment into your vehicle, to overall safety in vehicles. For safety there is an obviously apparent interest since it can help prevent accidents and possibly save people's lives.

Modern safety enhancements for vehicles include everything from equipping extra cameras on your rear for safer reversing, to installing range sensors which stop the vehicle when outside objects get in close proximity. Another modern enhancement in automobiles which is on the rise is the capabilities of vehicles to drive themselves. This does however come with a load of requirements, for a vehicle to be able to handle itself properly it needs to be able to properly observe its surroundings. It needs to be able to detect other vehicles, pedestrians, road signs (both permanent and temporary) as well as process the information it gathers in a reasonable amount of time. For self-driving vehicles to be seen as reliable all of the systems that the vehicle builds upon must work with high reliability in of themselves with a low computation time.

A sizable component in working with smart automobiles is the fact that most vehicles operate without wires attached and therefore have to handle communication with the outside with the help of wireless communications. Cars have used wireless transmission for a long time for capturing radio signals to supply drivers with entertainment and important operational information for roads and traffic. FM/AM Radio transmission does however not carry a lot of information, which is why a more competent method for communicating data must be used. This solution must be able to compensate for the amount of data needed to transmit as well as the speeds at which the data must be transmitted.

In modern society wireless communication is becoming an increasingly popular solution for communication between different devices. There is still however a lingering distrust in wireless communication as a means for sending and/or receiving data, since wireless communication is thought to bring about higher latencies and more frequent packet-loss. In the past wireless solutions were also seen as a way to degrade the bit-rate of your Internet connection. However with the use of modern routers it is possible to achieve speeds that eliminate this problem [15]. The possible shortcomings do however still to this day cause both people and companies to connect systems by wire when it isn't too inconvenient to do so. Furthermore, looking at modern wireless solutions there are several possibilities, but the most common ones are LTE and Wi-Fi. LTE stands for Long-Term Evolution and is the family of wireless solutions which rely on large radio towers which connect to devices in a radius of up to several kilometers. Wi-Fi is a different solution which

uses a smaller router to transmit wireless communication within a smaller radius of up to several meters. It is however in most cases difficult to say whether LTE or Wi-Fi is more reliable.

Streaming data to and from a device is an increasing trend for modern electronic purposes i.e. Video streaming, on-line video games, or surveillance. In a vehicle it is desirable for modules such as cameras and range sensors to be able to manage these tasks over a wireless connection as it might be either impossible or really expensive to connect them by wire. A lot of devices today work through the use of wireless mediums, but most of them are evolutions of earlier concepts. For example in modern times almost all television monitors use IR remotes but a couple of decades ago most of them had wired remotes or buttons on the unit itself.

The increasing use of wireless devices does however also burden the technology. With the increased number of wireless devices the networks become more loaded which increased latency, the possibility of collisions, and packet loss which can in extension lead to the quality of the connection being compromised. If you allow for buffering, the problem of unstable connections is lessened since the stream gets to get ahead of the output which lets it compensate for sudden delays. As the allowance for buffering is removed and frames are displayed as they are received certain stuttering may occur.

1.2 Related Work

The road maps that we have today are simple and made for humans that can understand simple instructions when navigating, in other word, these are not meant for use in autonomous vehicles. Autonomous vehicles need more detailed maps of the roads to be able to safely navigate. There are a couple of different sources to get a hold of such maps from today, for instance OpenStreetMap or DeepMap. These maps however are not flawless and needs to be constantly updated to provide accurate information. This is where cameras in road vehicles come in. Each vehicle in traffic could supply mapmakers with up-to-date information close to real time by uploading the video recorded during a trip to a server for processing. This could be done in real time through the use of LTE network or when the vehicle is within range of a Wi-Fi which might be preferable as the amount of data that needs to be transmitted might be quite high.

For quite a few years now, automobile manufacturers have shipped their products with an increasing amount of advanced driver assistance systems (ADAS) to improve the safety on the roads. These include, but are not limited to adaptive cruise control, automatic parking, collision avoidance system, lane change assistance, night vision pedestrian protection system, and vehicular communication systems.

The development of autonomous vehicles have taken off in the last few years with Google's subsidiary Waymo starting their work in 2009 [24], Tesla launching their model S with autonomy level[1] 2 [21, 14], and Renault currently testing their level 4 autonomous car [17]. Recently Volvo was given permission to try their autonomous vehicles in the city of Gothenburg [4], to be able to achieve their goal of a completely autonomous vehicle on the road by the year 2021. Autonomous vehicles use computer vision based on machine learning algorithms to detect objects around the vehicle. This requires a large amount of data during the training of the object detection algorithms. Vehicles with cameras could capture footage while the vehicle is in use, and later upload it to a server for further processing and tagging.

Real time traffic information systems are used in statically installed places along some highways and other places. These use cameras connected to a server that updates static signs along the highway to inform drivers of conditions up ahead [2]. Vehicles that are connected to the Internet can share traffic information with other vehicles, this allows for navigation systems to take the current traffic into consideration when planning a route. Suggestions or alterations to the route can be made as new information is received by the navigation system, this is to avoid congestions that were not present when the route was initially planned. This can alleviate heavily congested roads of traffic and allows for normal flow to be resumed quicker. It also saves the driver the anguish of getting stuck in traffic for hours on end and will in the end collectively save a lot of time during daily commutes.

The infotainment system that is shipped with an automobile today will most likely not be updated during the lifespan of the automobile. This means that auto manufacturers have to put considerable effort into testing the software before launch, adding much time to the development cycle, and any issues that are discovered after the vehicles release will subsequently not be fixed on models in use, unless it is critical for the vehicle to operate safely. Equipping vehicles with wireless Internet accessibility enables vehicle manufacturers to roll out over the air updates to the software running on the systems in the vehicle.

1.3 Motivation

In our master thesis we have worked for a company called Uniti to help them create a wireless solution to be able to monitor video feed from a vehicle in motion. This was constructed for a demo where they were going to display their vehicle, driving autonomously in front of a crowd. For Uniti it was important to be able to monitor the vehicle's progress through the vehicle's

[1] The level of autonomy indicates to which degree the vehicle is autonomous. Level 0 has no autonomous features, 1-3 have some, and 4-5 are autonomous enough to allow for no driver interaction during operation.

cameras, which the vehicle in itself was to use for adapting to its surroundings. With the help of our implementation this was supposed to help them to be able to manage things like emergency stops or remote control if needed.

For our purposes we worked with studying and analyzing video streams. Real time streaming of video has a wide range of applications and it also has a range of requirements on different metrics. If a real time video stream does not have proper responsiveness and stability then it is definitely noticeable. In the case of the Uniti demo this is especially important since noticeable delay can be quite problematic and responsiveness is key.

This was however the intended use of our implementation at the inception of our thesis. Due to delays and restructuring of the work for the demo our cooperation on the demo was eventually scrapped. The focus for the project however remained on how to effectively monitor a vehicle for the purpose of the demo.

1.4 Problem Definition

Area	Requirement	Comment
Hardware	nVidia Drive PX2	
OS	Linux Ubuntu 14.04	OS supported on the PX2
Latency	100ms	Average
ROS	Solution should be ROS compatible	
Wireless	Should work over LTE or Wi-Fi	Congested network[2]

TABLE 1.1: Uniti Requirements

Uniti wanted a robust solution to monitor their vehicle during autonomous operation, that would allow for remote emergency stopping of the vehicle if deemed necessary based on real time video being streamed to the remote control device. The video stream was justified as the person monitoring the vehicle might not have line of sight of the vehicle.

From earlier implementations done at Uniti, a video stream had been set up, however the latency of the stream was to high (in the area of 300ms) for the vehicle to be monitored remotely and required the person monitoring the vehicle to have line of sight of the vehicle to be deemed safe. Uniti wanted the latency to be reduced below 200ms, but stated that that was the higher end of their range and would be more comfortable with a latency around 100ms.

[2]During the event, a large number of people will attend and they will very likely use their mobile phones to record and upload or stream the event. The solution needs to be robust enough to handle networks under severe loads.

Seeing as latency was an important point for Uniti, fast video encoders were to be investigated, for this purpose JPEG and H.264 were chosen.

The solution should work without the operator having line of sight of the vehicle and with no cables attached. For this purpose either LTE or Wi-Fi were to be used. These were both tested to deduce which one would supply the most stable connection and also the connection with the lowest latency.

The implementation should also be able to run on Linux as the Operating system that was supported on Uniti:s platform was Ubuntu 14.04 LTS, and it should also be able to be implemented into a ROS node for easy integration with their system. ROS, or Robot Operating System, is an open-source operating system for robots. It provides hardware abstraction, message-passing, and package management, as well as tools and libraries for building and running code across multiple computers.

1.5 Method

The method which we have employed for our Master's thesis project can be abstracted into three general parts.

1.5.1 Implementation

To be able to gather data for our project and perform tests we needed a software implementation where we could with the help of different settings set up different test environments. This implementation was constructed with the help of several different 3rd party frameworks and was created to use our different pieces of hardware such as cameras and routers.

1.5.2 Data gathering

In this step we visited two different locations to gather data, this was because we wanted to find networks under relevant load. To generate our data we set up a video stream through a sending, and a receiving computer paired with either a Wi-Fi or LTE router. In Lund at Ideon we performed our Wi-Fi tests during work hours. In Malmö at Malmö Arena we set up our LTE tests during an almost sold out hockey game. In both cases the video streams were monitored using Wireshark as well as tapped into to extract our data.

1.5.3 Data analysis

After all of the necessary data was collected we used our Wireshark data to both extract the video from our capture data, and to dump our capture data into a database. We collected the capture data from both the sending and the

receiving ends which is why we employed a database to compare the data from both ends. This comparison together with the video footage is then used to extract the different video streams latency, packet-loss rate as well as standard deviation.

1.6 Disposition

This report is divided into 6 parts.

Chapter 2: Theory Explains the theory behind and the concepts needed to understand this project.

Chapter 3: Method Gives a description of the method and approach taken in this project.

Chapter 4: Implementation Discusses the implementation and how it was constructed

Chapter 5: Results Presents the results and findings from this project.

Chapter 6: Discussion Provides a discussion on the findings in this project as well as a discussion about future work.

Chapter 2

Theory

This chapter will provide all the necessary information and theory needed to understand this report and is divided into the following parts:

Metrics This section covers the metrics that were identified as important for this project.

Codecs This section covers the codecs investigated in this project.

Wireless transmission medias This section covers different wireless transmission medias

Packet analyzer This section covers packet analyzers and what they do.

2.1 Metrics

When measuring a network connection certain metrics are measured to be able to distinguish networks with certain qualities. These metrics include latency and packet-loss.

2.1.1 Latency

A key component in this project has been to look at and minimize the different sources of latency of a video transmission. For the sake of this report latency refers to the time between the time when a frame is captured by a camera until the time when it is ultimately displayed on the receiving end. The different activities that can introduce significant latency when handling a frame are:

Capturing The time it takes for the camera to capture one frame of a video.

Encoding The time it takes for the transmitting computer to encode the frame.

Sending The time it takes for the transmitting computer to pack the frame into and send the resulting packets.

Transmission The time it takes for the network to transmit the packets, end to end.

Receiving The time it takes for the receiving computer to unpack the frame.

Decoding The time it takes for the receiving computer to decode the frame.

Displaying The time it takes for the receiving computer to display the frame.

These sources all introduce latency in different ways and all of them have different options and solutions to them. Since this project only studies different encoding schemes and wireless mediums, this project focuses mostly on the encoding/decoding and transmission optimizations. However the capturing and displaying sections are affected by the specification of frame rate and resolution.

The frame rate will introduce a native latency, for example, a video stream running at 30 frames per second (fps), there will be at least 33.333... milliseconds between each frame, thus introducing a native 33.333... ms delay. Increasing the frame rate will reduce this delay, for instance, 60 fps gives a native delay of 16,66 ms delay, however this will also increase the amount of data needed to be encoded, sent, and decoded.

The resolution of the video transferred will also affect the latency however not natively. The resolution of the frames can affect the encoding, but can significantly reduce the amount of data that needs to be transmitted. If you capture an image in raw RGB format, four (4) bytes will often be used per pixel, where only three (3) bytes holds actual data. That means that for a video with a resolution of 640x480, each frame will need 1,228,800 bytes (1,228.8

kB) which results in 36,864 kB (36.864 MB) per second. When using codecs you can reduce this amount of data by for instance only sending the differences between frame images which would mean that, for example, only a quarter of the image would need to be sent again.

Latency can be calculated in two ways, however both ways include recording the end screen on which the stream is being displayed [5].

The first way is to insert a timer into the stream, and then calculate the difference between the displayed timestamps in the video. This gives that latency can be calculated as such:

$$l = t_2 - t_1 \tag{2.1}$$

where l is the latency and t_1 and t_2 are timestamps.

The other way is to place something that can change state visibly on demand and count the number of frames until the embedded frames update. This gives that the latency can be calculated as:

$$l = n_{frames} * \frac{\phi}{f} \tag{2.2}$$

where nf is the number of frames, f is the frame rate in frames per second, and ϕ is one second given in the desired magnitude, for instance, if the result should be given in milliseconds, then $\phi = 1000$ as one (1) second is 1000ms.

As the approach used in this project used an injected timer in the video stream, both approaches can be utilized.

Calculating the latency of the stream is one part but when considering the entire video stream which is measured it is not only important to be able to see what the average latency is, but also what the variance and standard deviation from the latency is. Variance tells how far from the mean value measured values be expected to stray. The variance is unknown and needs to be approximated. The approximation of the variance is given by

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu)^2 \tag{2.3}$$

and the standard deviation is given by

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{2.4}$$

Where x_i is a data point in the data set, μ is the mean of the values in the data set, and N is the number of elements in the set.

The use of variance as a metric for the spread of the data points is easily justified as if the measured points do not deviate much from the mean, σ will become small, and if the values are spread, σ will be large.

From the gathered data an interval for the distribution of the values can be estimated using the following formula;

$$I_\mu = (\overline{x} - \lambda_{\alpha/2}d, \overline{x} + \lambda_{\alpha/2}d) \qquad \text{if } \sigma \text{ is unknown } (d = s/\sqrt{n}, f = n - 1) \quad (2.5)$$

Where I_μ is the interval, μ expected value of the distribution,$\overline{x}$ is the mean of the measurements, $\lambda_{\alpha/2}$ is the lambda quantile for a given value of α, α is the targeted accuracy, s is the estimated standard deviation (square root of the variance), and n is the number of samples or data points used.

As more or less all values of the equation above are approximated from measurements, there will be some uncertainty in the result.

2.1.2 Buffer

A buffer is a frequently used concept in computing and communication, which helps with handling work loads which arrive at random intervals in variable amounts. A buffer generally works by having an amount of data stored in queue before the processing unit, as to make sure that the processing unit is always processing something. Buffers almost always work in a FIFO (First-In-First-Out) manner and therefore the data to first arrive will also be the first data that gets processed. On occasion the rate at which the buffer fills up may not out-match the rate at which data is processed, which can lead to the buffer getting empty and in turn stalling the processing unit, this is known as buffer underrun. [6].

In the context of this project there is a buffer on the receiving end of the video stream, which is intended to give the stream margins in case the network would be unstable for a short while. This buffer is made to be a few image frames deep so that a few frames can be lost before the video stream starts buffering. This does however have a clear downside, it brings about an inherent latency which is as long as the frame amount in the buffer. For different encoding standards it is however critical to avoid stalls when buffering as much as possible, as it may degrade the quality of the video transmitted.

2.1.3 Packet Loss

When a network connection is established and packets are being transmitted, there is always a risk that a packet may get lost along the way from the host to the receiver. This can be caused by a packet taking a bad route through the network or by a faulty access point in the network. It can however also be

caused by a network being unstable or flooded. A network is unstable when it is unable to operate optimally because of disturbances, this can for example be caused by a Wi-Fi router being slightly out of reach of a transmitting unit and therefore not receiving all the packets sent to it. A flooded network on the other hand is caused by an access point being unable to handle all the packet sent to it and this causes it to drop packets which are unable to fit into its queue. An example of this could be when there are a lot of people at a convention or festival which all use their mobile phones simultaneously, which causes the nearby access point to get flooded.

It isn't always a problem with packet loss in network connectivity as packets which follow the Transmission Control Protocol (TCP) contains measurements to prevent packet loss from becoming an issue. This is solved by using packet resends to acknowledge that a packet has arrived, which makes it easy for the sender to resend a packet in the case that it does not receive a confirmation of the packet's arrival[18].

However in modern network communication it is common to use "User Datagram Protocol" (UDP) instead of TCP as it is better suited for certain tasks. UDP is often used in connections where the overall time it takes for a packet to arrive is more important than the guarantee that all packets arrive. This protocol does however not have any measurements in place to cover for packet loss, which is both of a positive and negative impact. Since no packet resending is applied, the sender wont have to wait for any kind of confirmation before sending the next packet in a sequence, which is useful for higher level protocols such as the Real-time Transport Protocol (RTP). RTP is a protocol which is a higher level encapsulated in UDP which is commonly used for data streams which require as short of a transmission time as possible for example video streams or voice chat[16, 10].

Calculating the rate of packet loss of a video stream is pretty straightforward, taking the amount of packets received by the receiving end and dividing it by the amount of packets which were sent by the sender yields the percentage of packets which arrive.

2.2 Codecs

Video compression is the process of taking a video file or stream and minimizing the amount of data from the file or stream which is required to store or send. In modern computers this is done by applying an algorithm known as a video codec which takes the video and encodes it, this converts it to a smaller file which is then transmitted or stored. When the file later is transmitted or used, an inverse decoding algorithm is used which restores the file to the original video file or as close as possible. The time spent encoding, sending, receiving and decoding are factors which may increase latency.

A video codec in this context is the combination of the algorithms used for encoding and decoding. It is important to note that the algorithms used for

coding and decoding must match as different video codecs cannot encode and decode each others compressed formats. For the purpose of this report video codec will be referred to as codec.

Different codecs have different priorities that affect their area of use, since codecs use large algorithms to encode and decode data, they might require a significant amount of processing power. Some codecs might rely on a lot of processing power in encoding and decoding to be compatible with a lower bit-rate. On the other hand another codec might rely on high bit-rates to compensate for a low processing power. This does however illuminate how different codecs might use different methods to optimize for latency, quality, and bit-rate. It is also important to note that a codec only employs a collection of standards for the encoded data to follow and for the decoder to be able to handle it, this is why different implementations of the same codec might yield different results. A codec must not be to restrictive in its standards as it might compromise the flexibility of the codec. This is because a codec used for video streaming has different priorities than a codec for storing video on a disc. With modern codecs and their flexibility the same codec can be used to store data on a medium as well as streaming video.

To minimize the time between taking and displaying an image in a video transmitted from one device to another, a time efficient codec should be used.

2.2.1 MJPEG

Motion JPEG (MJPEG) encodes each frame in a digital video separately as a series of JPEG images. The encoding algorithm employed by MJPEG handles each frame independently, thus making this an intra-frame encoding algorithm as opposed to H.264 which uses an inter-frame encoding algorithm. However, this also means that the hardware requirements are lower for MJPEG as it is not as computationally heavy as inter-frame based compression schemes, and allows it to be more tolerable towards rapid motion changes in the image.

The major drawback of MJPEG is that in its basic form, it has no document that defines a single exact format that is universally recognized as a complete specification for use in all contexts, e.g. Microsoft documents that MJPEG files are stored in AVI file format, whilst Apple documents how MJPEG is stored in QuickTime files, and RFC 2435 [3] describes how MJPEG is implemented in an RTP stream.

JPEG

JPEG is a lossy encoding standard which is used to encode single frames or images. It is a collection of different lossy and lossless compression algorithms. In JPEG a picture is divided into blocks of 8x8 pixels for each color

space (Red, Green and Blue). The encoding and decoding parts of this algorithm both contain the same components.

The first part of JPEG is transformation, this is a lossless step. In this step a discrete cosine transform (DCT) is applied to a 8x8 matrix of pixels. This helps to reduce the potential size tremendously as a matrix of similar values will post-transform contain a significant amount of zeroes which can be removed further on.

The second part of the scheme is Quantization, this is a lossy step in the encoding since rounding is performed. The general action taken in this step is creating a new matrix by taking the transformed matrix from the last step and dividing it with a quantization matrix. Since real numbers require a potentially endless amount of bits to represent all numbers are rounded to the nearest integer. Rounding in JPEG is performed by adding 0.5 to the real value and then truncating the result. In this step high frequencies are also changed to zero as high frequencies represent sudden changes in the value of pixels. These sudden changes are not recognizable by human vision and are therefore redundant for image quality. JPEG has 100 quantization matrices, these range from Q1 which offers poor image quality but a high compression ratio, to Q100 which offers the best quality but worst level of compression.

The third part of the process is encoding, which entails converting the matrix from the earlier step to a transmittable line of numbers. In JPEG this is done by working through the matrix in a zigzag pattern from index (0,0) to index (n,n), and appending the numbers to the result. Since the DCT concentrates higher values around lower indexes this results in the higher and most probably non-zero numbers to be located earlier in the sequence. This process is fully reversible and therefore lossless.

When decoding the JPEG sequence every of the mentioned step is performed in reverse, which means that the encoding is used to recreate the matrix. After the matrix has been reconstructed a dequantizing step is performed. Finally an inverse DCT is performed and the 8x8 grid of pixels is restored. This reconstructed pixel matrix may however have certain errors pertaining to the irreversible rounding or choice of quantization matrix.[7]

2.2.2 H.264

H.264, also known as MPEG-4 Part 10 or Advanced Video Coding (AVC), is a commonly used video encoding which superseded H.263. It was developed in a joint partnership between ITU-T (International Telecommunication Union Telecommunication Standardization Sector) and MPEG (Movie Picture Experts Group). ITU-T named the codec H.264 while MPEG called it MPEG-4 part 10 as it was a new part in their MPEG-4 development [22]. H.264 is one of the most widely used video codings with it being used to, for example, store video on storage mediums, streaming video from online services etc.

H.264 employs an algorithm for encoding and decoding video which is highly efficient, which is one of the many reasons that it is widely used for both streaming and storage purposes. Outside of its efficiency it is also very flexible with the required bit-rate for its use, being able to work well with rates from 10Mbit/s to sub 1Mbit/s.

The H.264 encoding scheme does not handle frames independently like MJPEG but rather lets the encoding of a frame depend on other frames in the stream. H.264 encodes frames by designating 3 different type of frames:

I-Frame - Inter Frame This is a reference frame for the algorithm which contains all the image data for the frame

P-Frame - Prediction Frame This is a predicted frame derived from the last I-Frame which contains partial image data

B-Frame - Bidirectional Prediction Frame This is a predicted frame which is derived from the closest I-Frame and P-Frame, this frame contains the interpolated difference between those frames.

The encoding algorithm works by constructing a sequence of frames which creates sections delimited by I-Frames which contain a P-Frame and a variable amount of B-Frames between the I-Frames and P-Frames. An example with a B-Frame length of 2 can be seen in figure 2.1.

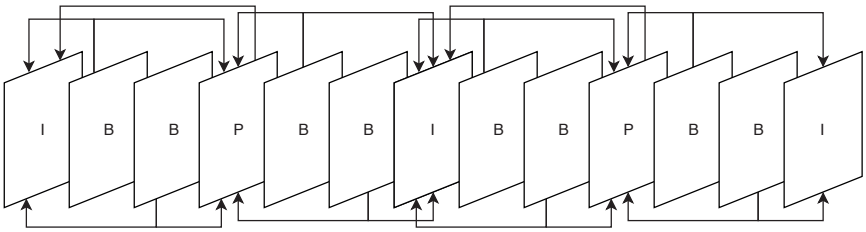

FIGURE 2.1: Interframe Predictions

The standard for H.264 defines a number of different levels which specify requirements upon the minimum decoding performance. This is helpful when designing a video stream environment, as the receiving end won't be required to decode anything beyond its processing capabilities [13].

2.3 Wireless Transmission Mediums

Every day wireless connectivity becomes more and more integral to businesses and everyday life. Wireless connectivity requires some kind of medium to route and transmit data without wires. There are a lot of different solutions to this problem, for example radio frequencies or infrared lasers.

Different wireless mediums are more suited to different types of tasks, for instance, Wi-Fi is well suited for usage in office and home networks, while

LTE is well suited for usage where network infrastructure is not as well implemented, such as outdoors or in the countryside. Even though the range of LTE is not an exact figure as it depends on the cell type [9], it far outreaches the range of Wi-Fi networks, however, this range comes with decreased transfer speeds.

To compare these some key parameters were identified. For LTE and Wi-Fi the specifications for the parameters are presented in table 2.1.

	LTE	**802.11n**	**802.11ac**
Nominal Data rate[1]		150Mbps	433Mbps
Aggregate Nominal Data rate[2]		600Mbps	1.73Gbps
Average Data rate	24.68Mbps		
Coverage[3]	83.92%	12-70m	12-35m

TABLE 2.1: Wireless Communication Comparison

Latency was also identified to be a key parameter for this project, however as finding the latency for each communication method during video streaming is part of the work in this thesis this will be presented later.

2.3.1 IEEE 802.11

IEEE 802.11 is a set of media access control and physical layer specifications for implementing wireless local area network computer communication in the 900 MHZ and 2.4, 3.6, 5.8, and 60GHz frequency bands. It has its origins in a 1985 ruling by the U.S. Federal Communications Commission that released the ISM band for unlicensed use. NCR Corporation/AT&T (now Nokia Labs and LSI Corp.) invented a predecessor to 802.11 in 1991 in Nieuwegien (the Netherlands) which was brought to the market under the name WaveLAN with data rates of 1-2 Mbit/s.

For any given variation of 802.11, maximum achievable throughputs are given in the layer-2 data rates or are based on measurements done under ideal conditions, in other words, the highest achievable speeds possible for the implementation.

This technology is most commonly realized in Wi-Fi routers which is a wireless communication solution often found in households or business offices. The standard has evolved over the years resulting in a number of versions allowing for progressively increased bit rates. Some common versions used for Wi-Fi routers are 802.11b/g, 802.11n, and 802.11ac.

[1]Single stream, 1x1, 40MHz for 802.11[12]
[2]Multiple streams, 4x4, 40MHz for 802.11[12]
[3]Denotes coverage percentage as of November 2017 in Sweden for LTE[19], and indoor range for 802.11

MCS

Modulation and coding scheme (MCS) value describes every possible combination of number of spatial streams, modulation type, and coding rate.

By using multiple-input and multiple-output (MIMO) it is possible to run up to 4 spatial streams in 802.11n. Effectively, it allows for the use of the same area of frequency space to transmit and receive multiple data streams, thus increasing the possible bit rate. With later revisions of 802.11ac, the number of possible spatial streams is increased to 8 [1].

To communicate data through the air, the data stream needs to be modulated. For higher bit rates, more complex modulation is used. The more complex modulations are however more susceptible to interference and may suffer performance degradation if line of sight to the access point is obstructed. If there are a lot of Wi-Fi traffic in the area the modulation can be shifted to a less efficient but more robust modulation to handle the interference.

Coding rate describes how much of the data stream that is actually usable data, that is, how much of the information sent is data packets. This is expressed in fractions with the most efficient rate being 5/6 (or 83.3%).

Guard intervals provide protection against propagation delays, echoes, and reflections to which digital data is normally sensitive to. In 802.11 a guard interval of $0.8\mu s$ is used. Optional support for a $0.4\mu s$ was introduced in 802.11n for increased data rates. This does however come with a cost and results in a higher packet error rate when the delay spread of the channel exceeds the guard interval or if the timing synchronization between the transmitter and receiver is not precise.

Channel width specifies the amount of bandwidth that is used in the available spectrum. It is possible to speed up wireless networks by using more bandwidth, however it can also cause a higher amount of interference for other access points in the area. The typical channel width for $2.4GHz$ Wi-Fi is $20MHz$, and it is not uncommon to use $40MHz$ on $5GHz$ Wi-Fi.

All of this determines the available bit rate of the connection, and thus the maximum possible throughput of data. Depending on the usage of the spectrum in the area, only certain channels may be available.

2.3.2 LTE

Long-Term Evolution (LTE) is a high-speed wireless access standard and registered trademark owned by the European Telecommunications Standard Institute (ETSI) in use in telecommunications. The technology is based on preexisting Global System for Mobile communications (GSM) and Enhanced

Data Rates for GSM Evolution (EDGE), as well as Universal Mobile Telecommunications System (UMTS) and High Speed Packet Access (HSPA) technology. However the capacity and speed is increased by using a different radio interface in conjunction with core network improvements.

The 3rd Generation Partnership Project (3GPP) is behind the development of the LTE standard which is specified in the Release 8 document series. Minor enhancements were made in Release 9.

LTE is commonly marketed as 4G LTE & Advance 4G, even though LTE does not meet the technical criteria of a 4G wireless service as specified in the 3GPP Release 8 and 9 document series. Originally the requirements were set forth by the ITU-R organization in the IMT Advanced specification. However, due to pressure from marketing agencies, and the significant improvements that Evolved High Speed Packet Access (HSPA+), WiMAX, and LTE brings to the existing 3G technologies, ITU later decided that LTE together with the aforementioned technologies can be called 4G technologies. On the other hand, the LTE Advanced standard formally fulfills the ITU-R requirements to be considered IMT-Advanced. To differentiate LTE Advanced and WiMAX-Advanced from the current 4G technologies, ITU has defined them as "True 4G".

2.4 Packet Analyzer

Packet analyzers are programs or hardware that can intercept and log traffic that passes over a digital network. The act of intercepting and logging traffic is called packet capture. These have a wide variety of usages, for instance they can be used to monitor specific metrics of a connection, or it can be used to extract the data that is being transmitted. There are a lot of packet analyzer software available, some of them are made with specific goals in mind, for example they might be made to capture a specific protocol. There are however programs such as WireShark or TCPDump which are open-ended, these allow you to capture all IP packets from specific connections.

TCPDump runs on command line and displays TCP/IP and other packets being transmitted over a network device [20]. It supports capture filters, allowing the user to filter the captured traffic to specific protocols or port. As the captured data is written to standard output as plain text the data needs to be handled manually by the user after the data has been captured.

WireShark works in a similar manner but runs with its own graphical user interface. It too supports capture filters, and come with the added feature of display filters which allows for filtering post data capturing , and packet decoders which allows for decoding of captured packets and data streams [8].

Chapter 3

Method

This chapter explains the the method used in this project. The chapter is divided into the following parts:

Design Describes the information gathering and alignment to the restrictions of the project.

Data gathering Describes the approach taken to gather data which could later be analyzed.

Data analysis Describes the approach taken to analyze and handle the data gathered during the previous phase.

3.1 Design

This section covers the information gathering for, and the design, of the system.

3.1.1 Software Considerations

There are plenty of video streaming software readily available already, so it was decided that a solution that would fit the requirements that Uniti had for the project should be pursued before resolving to building an in-house solution.

OpenCV An open source library containing computer vision and machine learning software libraries.

Nginx A server hosting solution, with plenty of built in streaming libraries.

gstreamer An open source media framework.

Initially OpenCV was considered as it was already being used for other purposes at Uniti and thus the library was readily available on the system. However this library is not really built for video streaming, but rather for image manipulation and computer vision. Thus this solution needed to be bundled with a library for handling encoding, packaging, and transmitting the video stream. It worked, to some extent, but the end to end latency did not meet the requirements.

More complete solutions were also considered, such as the readily implemented Nginx web server. It showed promise as it came with load balancing, Video on Demand (VOD), and could supply MP4 or FLV over HTTP. However during testing of the system, latencies were so high (in the range of 5-10 seconds) that this solution was quickly deemed unfit for the purpose.

Gstreamer is an open source media framework that allows for fast command line prototyping, and has an comprehensive core library, is compact, has lightweight data passing, and gave access to an API for various programming languages. During some lightweight testing Gstreamer was found to yield low inherent latency. The low inherent latency and ease of use led to the conclusion that Gstreamer was the best choice for the project out of the tested frameworks.

A prototype was implemented using the command line tools and over a controlled environment the latency were well within the requirements. This prototype was used later during the data gathering phase as it would provide accurate emulation of the finished system.

3.1.2 Hardware Considerations

As the platform on which the system would run would not be available to us at all times, the need for a test platform was obvious. This was constructed in the way described below so that parts of the test platform could later be transfered to the Uniti car.

Aspects considered were:

Cameras The specifications of the test camera needed to be close to the cameras used in the actual vehicle to simulate accurate loads for the encoder.

Underlying system The computer that would manage the encoding of the video stream needed lesser than or equal computational power to the computer that would go in the vehicle to provide a good base for evaluation.

The project required a camera for the test setup, and for this a rather basic USB camera module was selected. It was not much more than a PCB with a lens on it. But it came with a range of modes of operation and could deliver 1080p/720p at 30fps and lower resolutions at 60fps, and as it was a USB connected device, it would work with more or less any system that had a USB port.

As for the underlying system a raspberry pi 3 B+ was considered, however, as a USB camera was chosen instead of a Raspberry pi camera, the GPU of the Raspberry could not be utilized in the encoding. This resulted in very low frame rates, roughly 10 fps. As this would not emulate the actual system, the Raspberry was dropped in favor of a more powerful computer.

3.2 Data Gathering

The goal of this project was to calculate and conclude the best parameters for operating a system as described in 1.1. The scope of our tests was limited to focusing on four major cases, using either Wi-Fi or LTE as a network medium and H.264 or JPEG as encoder. The main focus of the tests was to deduce the amount of inherent latency in any of these four cases, as well as standard deviation and average packet loss.

3.2.1 Setup

The major problem of performing the experiments are the usage of RTP for streaming video. Since there is no process to assure that packets are delivered, the end-to-end latency becomes harder to calculate [10]. Even with a capture of the network traffic from both units it is difficult to discern the end-to-end latency as two real-time clocks may differ by a significant amount of

milliseconds [11]. It is therefore not possible to calculate the transmission time over a network by comparing the system timestamps for each packet sent and received.

Although transmission time over the network would have been interesting to observe, this is not of importance to this project, as the interest lies in the end to end latency, that is, the time from when an image is captured by the camera to the time when it is displayed on the receiving end.

To observe this latency a timer was added to the video stream, and then the receiving computers monitor was recorded to create a loop, and by doing so creating an "infinity mirror" effect with several frames each containing a timestamp where the timestamp in each consecutive infinity mirror frame is incremented by the latency for each step (see figure 3.1).

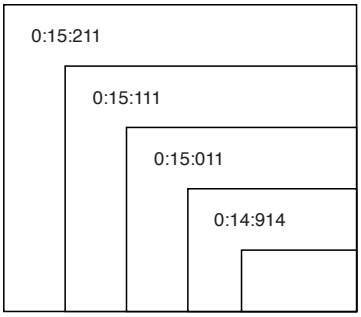

FIGURE 3.1: Infinity Mirror

Unfortunately, each iteration of the infinity mirror loses some image quality as it is recaptured over and over again. Therefore, most of the time only the first one or two timestamps were readable, this is however enough to calculate the end to end latency of the system with high enough precision.

Since this method relies on the displaying monitors refresh rate there is a margin of error that needs to be taken into account. The monitors refresh rate is important here as the timestamps captured only can increase in $1/f$ increments, where f is the refresh rate of the monitor. This means that there is always a possible margin of error at $1/f$ seconds.

Wireshark was used to monitor and log the traffic between the two stations. From these logs the packet loss could easily be determined. The captured data was also used to recreate the video from the stream so that it could be analyzed frame by frame.

3.2.2 Wi-Fi

For testing the system over Wi-Fi, an area with high amounts of activity over Wi-Fi needed to be found. Ideon Science Park in Lund hosts over 100 different companies in its business park and they allowed for measurements

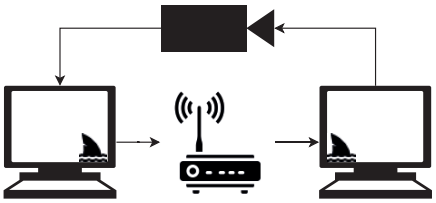

FIGURE 3.2: Wi-Fi Test Setup

to be taken during busy hours. They did however not allow measurements in some areas as they were already somewhat overloaded and they did not want additional interference in those areas.

The setup for the Wi-Fi tests consisted of two computers with Wi-Fi modules, a camera, and a Wi-Fi router. The camera was pointed at the receiving computers monitor and images were captured by the transmitting computer (see figure 3.2).

Quick experiments showed that for stationary computers within the range of the Wi-Fi. the MCS levels were fairly stable as long as they were in line of sight from the access point, at around 13-15 for 802.11n at roughly 2m. When the computer was not in line of sight but at the same distance the MCS levels decreased to between 6-12.

For moving computers within the Wi-Fi range (2-10m), the MCS was a bit more unstable. With line of sight the MCS wandered in the 6-14 range, and without line of sight the MCS dropped down to the 3-7 range.

The signal strength during the measurements where line of sight was allowed was roughly at -43$\mathrm{dB\,m}$, and the signal strength for the measurements where line of sight was not allowed resulted in a signal strength of -50$\mathrm{dB\,m}$

For MCS level 3 at 20MHz with regular guard intervals the resulting bit rate would be 26$\mathrm{Mbit/s}$. In comparison, the average data rate for LTE in Sweden was 24.68$\mathrm{Mbit/s}$ in November of 2017 [19].

The router was placed with one wall in between it self and the transmitting and receiving computers. This to remove line of sight between the devices as this would most likely not be possible during the demo event.

Both computers ran Wireshark to capture the traffic to and from the Wi-Fi modules.

This test was done over a couple of days (not consecutive). The first experiment focused on getting data on packet loss over Wi-Fi when there was high activity over the Wi-Fi. Later measurements focused on end to end latency as well as control data to verify the system clock drift.

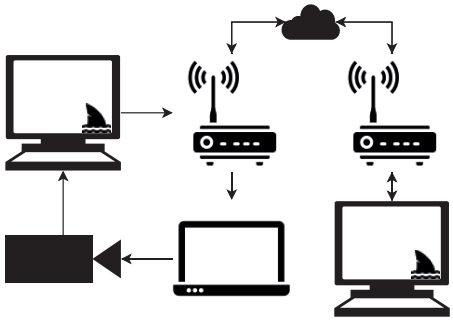

FIGURE 3.3: LTE Test Setup

3.2.3 LTE

LTE tests required a venue where there would be high enough activity over the network as well as similar distance to a base station. The base station at the demo event was located within 300m. Finding a venue that both hosts a high amount of activity and allows a setup of the equipment for this project proved difficult, but it was eventually found at a large arena.

Malmö Arena in Malmö hosts a range of entertainment and sports events which can host more than 10000 attendants. There is a base station located on top of Malmö arena, placing it somewhat closer than what would be optimal, at roughly 50m. Studying maps however, it was determined that roughly the same amount of walls would be between the test site and the base station as there would be at the demo event. Malmö Arena helped out with tests for this project and provided a room during one of their events. It was important that cellphones would be allowed to be used during the event.

Setup for this experiment was a little different as the computer that the stream was sent to would not be in the same room. The receiving end was setup at a remote location and controlled remotely through a program called nomachine. This extra link in the system introduces some extra latency but it can easily be accounted for.

The receiving computer was connected to the fixed network, while the transmitting computer was connected to the LTE network. The laptop controlling the receiving computer was also connected through the LTE network.

The receiving computer was controlled from a laptop at the test location, which the camera recorded, the captured frames were encoded on a separate computer and transmitted via LTE over the Internet to the receiving computer, thus completing the loop.

On the test site only traffic from the transmitting computer was analyzed with Wireshark as the returned traffic to the laptop was not of interest. The traffic was also analyzed on the receiving computer.

3.3 Data Analysis

This section covers the analysis of the data gathered in the previous section.

3.3.1 Latency

The latency is measured in milliseconds, from the time the image is captured by the camera to the time when it is displayed on the remote screen, i.e. the time it takes to capture, encode, package, transmit, receive, unpack, decode and display each frame. From each frame in the stream where the timer is visible and readable, the difference is recorded and subsequently collected in a data set that is used to determine the average latency over time. The standard deviation is then calculated from the same data set.

3.3.2 Packet Loss

From the logs of a packet analyzer the packet loss can be deducted quite easily as this feature is often supplied with the packet analyzer. Given the data capture of the sending and receiving ends of the UDP connection it is possible to determine the number of packets sent and received. The packet loss rate is then discernible by the following formula:

$$P_l = 1 - \frac{P_r}{P_s}$$

Where P_l is the packet loss rate, P_r is the amount of packets received and P_s is the amount of packets sent.

To verify the results from the packet analyzer, the timestamps and sequence numbers of the captured traffic were exported to a comma separated values (csv) file for each captured stream. The csv files were then segmented into parts of 65536 entries each. This is due to the fact that the sequence numbers go to 65535 and then wrap to 0. Each segment is paired in transmitter receiver pairs, and subsequently joined on sequence numbers. The sequence numbers that do not appear in both files are assumed to have been lost during transmission and tallied towards a total number of lost packets.

Chapter 4

Implementation

This chapter explains the method used in this project. The chapter is divided into the following parts:

Overview An overview of the implementation

Hardware A deep dive into the hardware oriented parts of the implementation.

Software Describes the software aspect of the implementation.

4.1 Overview

During our project we have implemented a platform as a solution which provides low latency video streaming. This was constructed to be able to work with different network mediums and coding standards. It was an important to be able to gather measurements from the platform that would make it possible to compare different solutions. It was also important to make sure that this implementation could be used in the car for the demo presentation.

This implementation is constructed using several hardware and software components. These different components interact and work with each other in different ways and have different dependencies.

Originally our intentions were to implement our platform to work with the demo setup. This was constructed with additional features which were not fulfilled as a different solution was later found to be easier to apply. The original implementation setup is presented in figure 4.2.

In an earlier step in our project we developed a test setup which was supposed to be used to test the different codecs and wireless mediums. This was ultimately the model we finished with as it was close enough to the intended implementation. This implementation was also used to execute tests with the different codec and medium combinations.

In both of these setups the routers and coders/decoders are intended to represent the component used in the specific test case, like for example the router used is a WiFi router in the WiFi test and the codec is JPEG in the JPEG test

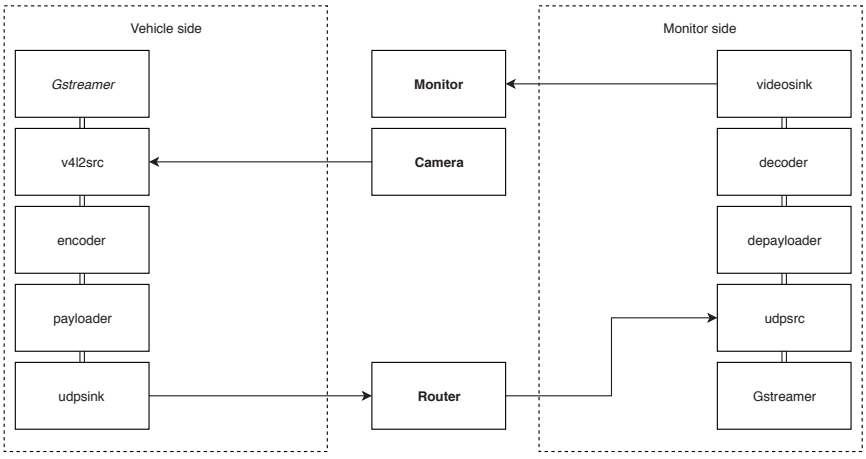

FIGURE 4.1: Test Setup

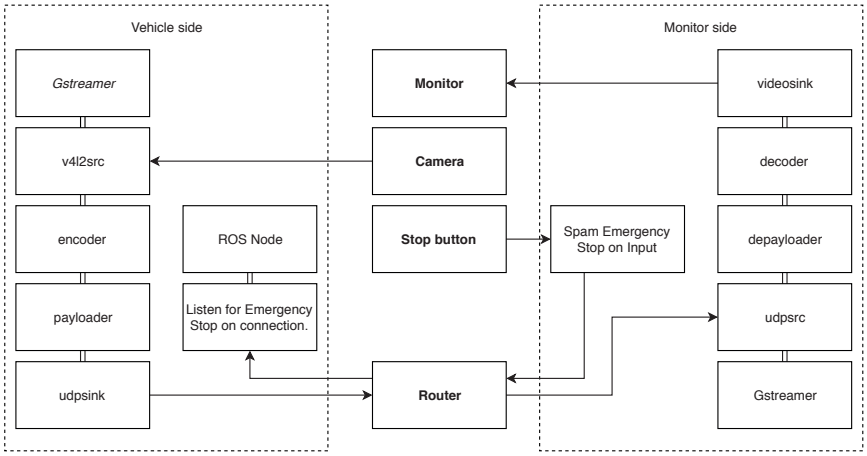

FIGURE 4.2: Planned Setup

4.2 Hardware

To implement this project in the car, the following hardware was required.

Control computer This computer is responsible for receiving a video feed from the vehicle, decoding it and displaying it. This computer would also during a real life demo be responsible for sending stop signals to the vehicle.

Vehicle computer During the demo this computer is supposed to be a Nvidia Drive PX2, as it was supposed to serve as the vehicles main computer. However during the tests as this computer was not as readily available we used a custom made stationary computer running an AMD Ryzen 5 and a Nvidia GTX 1060. This computer serves the same purpose in both of the scenarios, it is intended to capture video footage with the help of the camera and then encode and transmit it.

Router For Wi-Fi a router implementing IEEE 802.11n was utilized as this supported up to 288.8Mbit/s. For LTE we used a LTE router supplied by Tele2 IoT. The router serves the same purpose in all scenarios, it relays data between the vehicle and control computer.

Camera The camera used in this implementation was a USB camera from ELP which is pretty bare-bone as to provide as low latency as possible. In the demo this would have been replaced by an Nvidia camera which accompanied the PX2. During the demo the camera's purpose was to capture the vehicles surroundings and during the testing scenario its purpose was to measure latency.

Monitor Used to display video feed from the vehicle during the demo. Used to measure latency during latency testing.

4.3 Software

For the software part of the implementation, different solutions were examined. The final solution needed to incorporate a way to fetch image data from the camera, be able to encode the video stream, package the encoded video in packets, and to transmit the resulting packets. Likewise, the receiving end needed to be able to receive, unpack the packets, decode, and play back the video stream.

Although several solutions were examined at the start, our implementation eventually ended up relying most heavily on the Gstreamer framework. This framework incorporates modules that easily lets you encode, decode, pack, and unpack a video stream. Gstreamer also turned out to be one of the more latency efficient solutions we would investigate, which is why we chose to work with it.

Since Gstreamer works out to do most of our work for us our software implementation is mostly about using scripts to set up Gstreamer to establish the connection properly.

As the setup of Gstreamer can be a bit tricky, the setup was automated with a script that set up the entire process depending on a few input parameters. This made it easy to set up different encoders and IP addresses during testing. The test scripts can be found in appendix A.1 and A.2.

For the implementation on the vehicle Gstreamer was setup to run on startup on the vehicle computer. This script connects to the camera and sets up the video streaming to a specified IP. The startup script was similar to the test scripts, but with more error handling to ensure that the script could be started correctly.

Chapter 5

Results

This chapter will summarize the results from our tests developed for the goal of this project. This chapter is divided into four sections.

Metrics This section describes the different metrics used in the test results.

LTE Tests This section displays the LTE test results

Wi-Fi Tests This section displays the Wi-Fi test results

Distributions This sections displays and compares the different distributions

5.1 Presentation

From the requirements for this project a set of metrics were identified to be of significance to be able to objectively determine the suitability of the solutions tested. The metrics selected for this were latency, standard deviation and packet-loss.

The measurements for all test cases are presented in a few different ways. Packet loss and standard deviation are presented in tables, while latency is displayed in different forms of graphs. Latency is presented in discrete graphs as well as histograms, further on distributions of the latency for the different test cases are also displayed.

For our latency measurements the aim was to provide a visual representation of 1000 data points for each test case, this was chosen as it seemed as an ample amount to display. A data point in this case represents a chain of frames that could be observed at a single moment according to the description in 3.2.1. The difficulty in procuring each data point is that a chain of frames often can make it hard to read the timestamps between frames, this results in a large amount of unusable frames. The nature of the testing methodology also makes it so that all data points measured have random time intervals between them. With all these things in mind it was decided that 1000 data points which were taken at semi-random intervals from each test case would be adequate as this would both give a large number of data points to observe as well as a bit of an overhead of trends in latency, since all data points in the discrete graphs are ordered by sequence.

5.2 LTE Tests

This section provides the results from each of the measurements done over LTE.

5.2.1 Latency

Protocol	Mean	Std. Dev.	Min	Max
H.264	300.694ms	109.766ms	132ms	808ms
JPEG	254.568ms	68.903ms	100ms	636ms

TABLE 5.1: LTE Results

A table that presents the standard deviation as well as mean, min- and max values is presented in 5.1. The mean for both encodings seem to not differ by too much, but there is quite a difference in standard deviation.

A graph comparing 1000 data points of latency for H.264 is presented in figure 5.1. However for JPEG there wasn't enough data to process 1000 clean data points, which is why there is only 760 data points for JPEG, the latency for JPEG is shown in figure 5.3.

Both graphs show values that can be considered unstable with some visible trends. Overall JPEG does seem to be a bit more stable which is the same conclusion that can be drawn from table 5.1. A histogram is also presented for both H.264 and JPEG in figure 5.2 and 5.4 to give a picture of the frequency of specific values. These graphs both seem to suggest similar distributions for the two encodings, although lower values seem more frequent for JPEG.

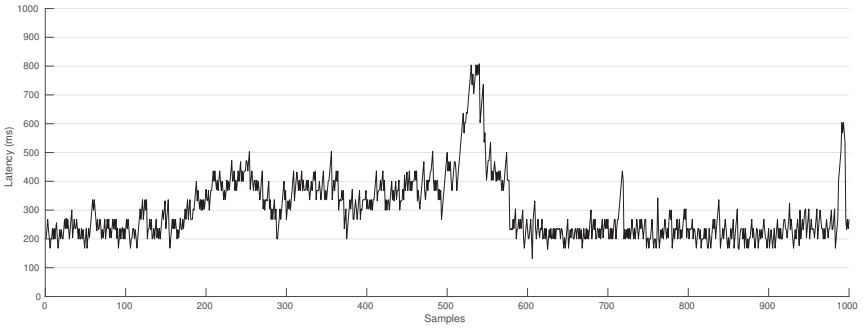

FIGURE 5.1: Latency of the H.264-LTE Stream

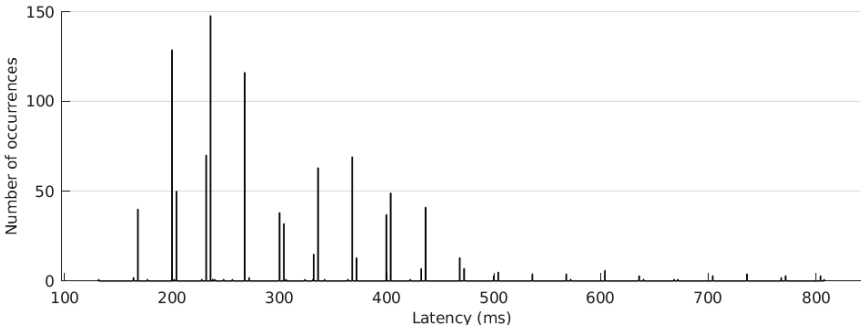

FIGURE 5.2: Distribution of Data Points for H.264 over LTE

5.2.2 Packet Loss

The result from analyzing the sent and received packets are presented in table 5.2. It is worth noting that the amount of packets sent during the JPEG measurement is significantly lower, however this has nothing to do with the efficiency of the encoding, but rather that the measurement was shorter. During the data gathering for JPEG over LTE the first stream capture got corrupted and no useful data could be produced from it. The corruption was not noted

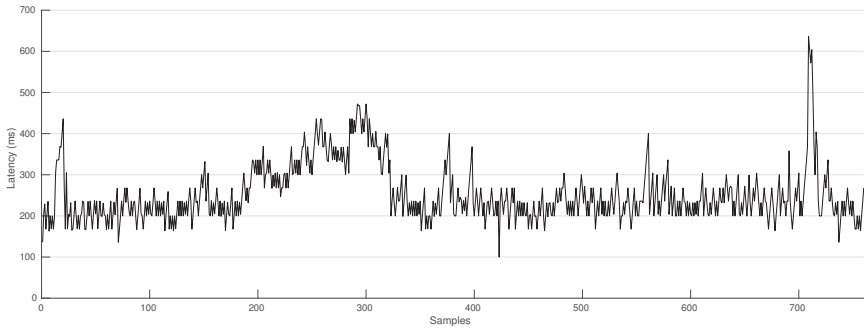

FIGURE 5.3: Latency of the JPEG-LTE Stream

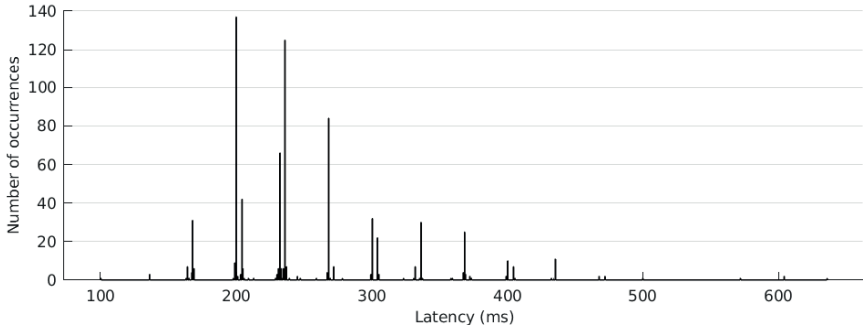

FIGURE 5.4: Distribution of Data Points for JPEG over LTE

until after when the data was being analyzed. During the gathering inconsistent behavior was identified in the stream, odd freezes and inconsistent latencies, and thus another capture was done to serve as a verification of the first stream. The second stream was significantly shorter as it was only meant to serve as validation of the first stream, however as no useful data could be produced from the original capture, the second was used instead.

Protocol	Packets sent	Packets received	Packets lost	% dropped
H.264	739833	739496	337	0.05
JPEG	194751	194572	179	0.09

TABLE 5.2: LTE Packet Loss

5.3 Wi-Fi Tests

This sections provides the results from each of the measurements done over Wi-Fi.

5.3.1 Latency

The results from the latency measurements are summarized in table 5.3, and covers the mean, standard deviation, min and max values.

The latency measured for H.264 over Wi-Fi is depicted in figure 5.5 while the measured latency for JPEG is depicted in figure 5.7. The distribution of the measurements for H.264 and JPEG over Wi-Fi are depicted in figure 5.6 and 5.8 respectively. Here each data set consists of 1000 data points.

By looking at the histograms for both H.264 and JPEG, the measurements show a rather stable video stream, with some occasional latency spikes. However most of the measurements are located at 100ms for both H.264 and JPEG.

By looking only at the graphs not much can be discerned from them, however by looking at the table 5.3 it is apparent that both the mean and standard deviation for JPEG are lower than that of H.264 by 4.559ms and 17.126ms respectively.

Protocol	Mean	Std.Dev.	Min	Max
H.264	110.091ms	37.080ms	64ms	472ms
JPEG	105.532ms	19.954ms	32ms	304ms

TABLE 5.3: Wi-Fi Results

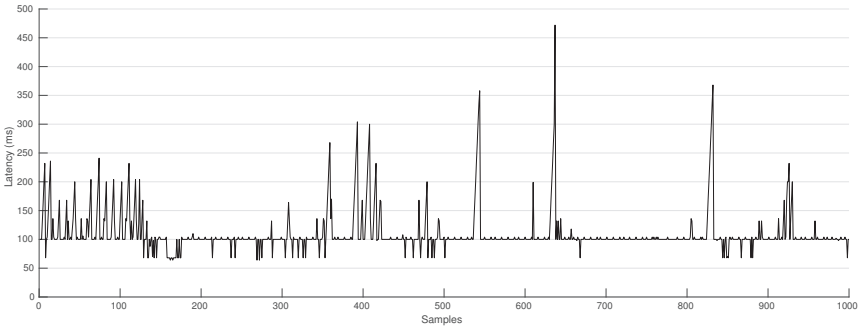

FIGURE 5.5: Latency of the H.264-Wi-Fi Stream

5.3.2 Packet Loss

The result from analyzing the sent and received packets are presented in table 5.4. It covers the amount of packets sent, packets received, packets dropped, and drop percentage.

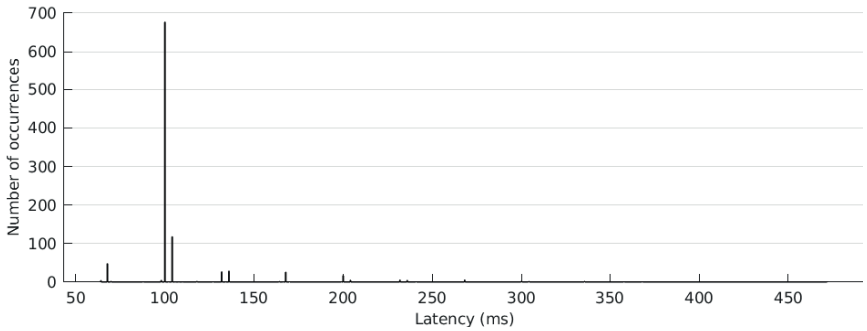

FIGURE 5.6: Distribution of Data Points for H.264 over Wi-Fi

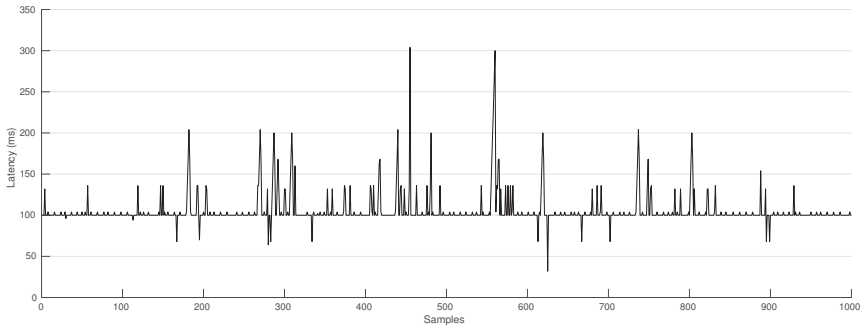

FIGURE 5.7: Latency of the JPEG-Wi-Fi Stream

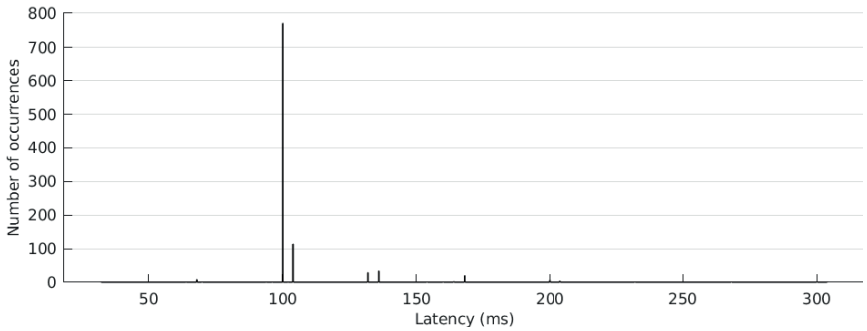

FIGURE 5.8: Distribution of Data Points for JPEG over Wi-Fi

Protocol	Packets sent	Packets received	Packets lost	% dropped
H.264	1281697	1281689	8	6.24×10^{-4}
JPEG	4380275	4379886	389	8.88×10^{-3}

TABLE 5.4: Wi-Fi Packet Loss

5.4 Distribution

This section covers the distribution and the visualization of the distribution of the data, as well as the confidence intervals for each of the distributions.

5.4.1 Cumulative Distributions

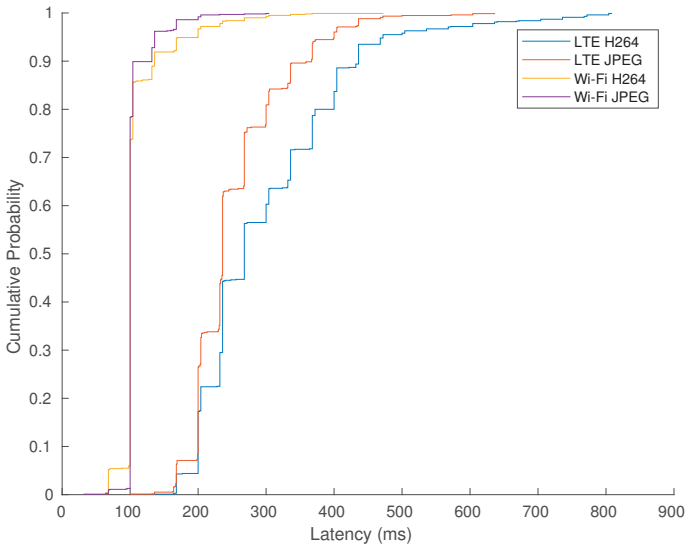

FIGURE 5.9: ECDF Comparison

In figure 5.10,5.11,5.12, and 5.13, the empirical cumulative distribution functions (ECDF) for each test case is visualized (the red line) with a generated cumulative distribution function (CDF) for a normal distribution given the calculated μ and s for each data set (the blue line). Examining the figures it is safe to say that the distribution is close enough to normal distribution, such that normal approximation can be utilized in the approximation of the confidence intervals.

5.4.2 Confidence Intervals

The confidence intervals indicate with a given accuracy, within which range the measurements are expected to appear. For this experiment, an accuracy of 99.9% was selected. This means that $\lambda_{\alpha/2} = 3.2905$. The mean was calculated as the arithmetic mean value of the measured data, and the standard deviation (s) was calculated as the square root of the sum of the squared differences divided by the number of elements. These results are presented in table 5.5.

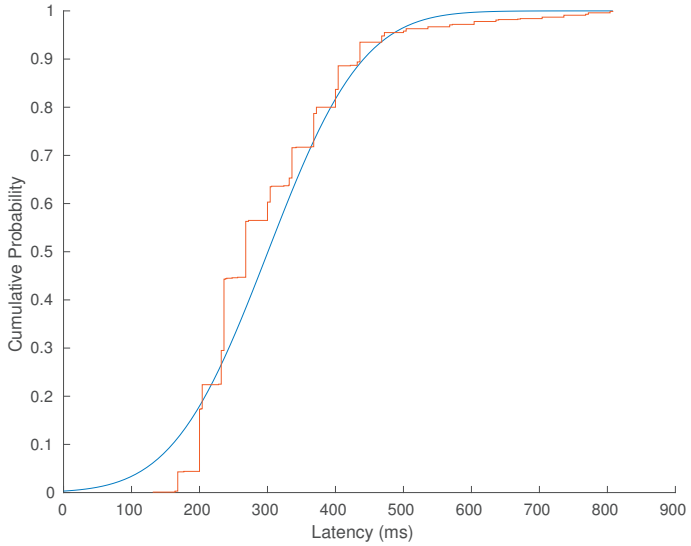

FIGURE 5.10: ECDF LTE H264

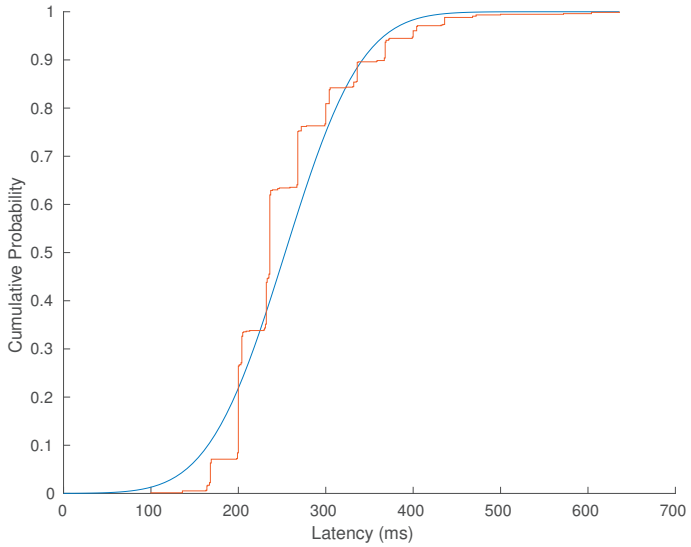

FIGURE 5.11: ECDF LTE JPEG

Medium	**Protocol**	I_θ
LTE	H.264	(289.2623267ms, 312.1156733ms)
	JPEG	(246.3438082ms, 262.7921918ms)
Wi-Fi	H.264	(106.23265ms, 113.94935ms)
	JPEG	(103.4556916ms, 107.6083084ms)

TABLE 5.5: Confidence Intervals

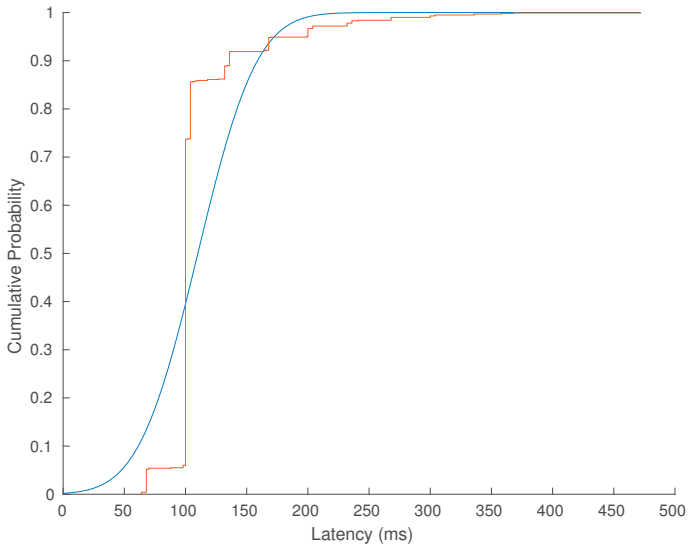

FIGURE 5.12: ECDF Wi-Fi H264

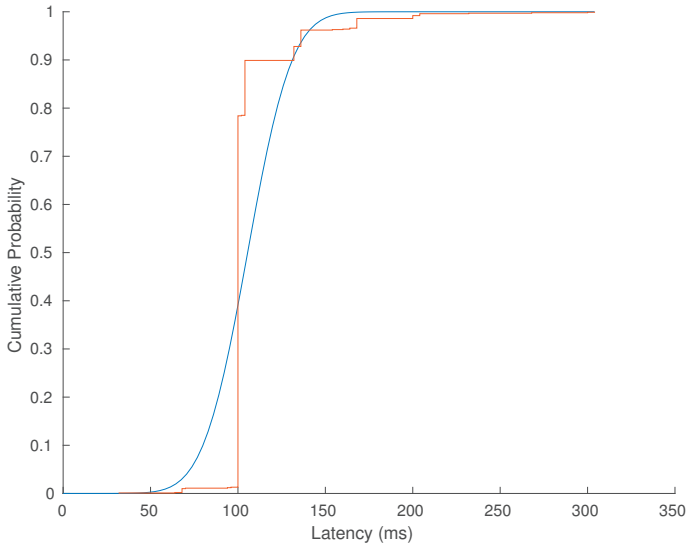

FIGURE 5.13: ECDF Wi-Fi JPEG

Chapter 6

Conclusion

6.1 Evaluation of Results

6.1.1 Transmission Method

Looking at the results in chapter 5 in regards to the different metrics there is a clear correlation between performance and the choice of transmission medium. Looking at LTE the mean value of H.264 is 300.694ms as well as the mean value of JPEG is 253.568. This is interesting in contrast to Wi-Fi where the mean for H.264 is 110.091ms and 105.532 for JPEG. This shows a clear favor in latency for Wi-Fi.

When looking at packet loss LTE performed worse than Wi-Fi. LTE dropped 0.05-0.09% of the packets while Wi-Fi dropped 0.000624-0.00888%. There was no significant difference between the encodings in terms of packet loss, which is to be expected as they should not affect this metric.

Looking at figure 5.9 you can distinguish the two different mediums as their cumulative probability function curves follow similar patterns. In this context Wi-Fi performs quite a bit better then LTE. Looking at a 95% case for both Wi-Fi cases you can see that neither of these cases have a latency of above 200ms. However looking at LTE the same analysis yields results of up to 400-500ms. This also gives Wi-Fi a clear advantage.

Generally speaking this makes Wi-Fi seem like the clear cut choice as a medium, however this is probably affected by the setup of the two testing environments. In the Wi-Fi setup the entire system is constructed in a closed network, in the LTE setup however the network is connected to the internet and data is relayed through a remote host outside. This is however a realistic setup since we would need to build our own base station to transmit our own LTE transmission to keep everything in a closed network. In the Uniti demo we wouldn't build our own base station either as it would make this network solution far too expensive to be viable. Since we can't keep the LTE solution in a closed network it will introduce the latency of sending our network traffic through the internet. This will always give Wi-Fi a certain advantage when comparing latency.

Another thing to discuss is the stability of the different mediums and looking at figure 5.9 again it can easily be spotted that around 70% of the probability is on the exact same value for Wi-Fi. LTE on the other hand varies a bit more.

6.1.2 Encoder

When comparing the time efficiency of the H.264 and JPEG encoder there is an overlap in the confidence intervals when measured over Wi-Fi, so we can't say with certainty that one is better than the other. For LTE however, the confidence intervals are completely separated even at a 99.9% percentile. This is somewhat counter intuitive as JPEG in general is less efficiently encoded in terms of bit usage. The MJPEG encoding is however faster than H.264 in the general case. There are some exceptions to this, however, for instance if you only allow for intraframe encoding using H.264 it could possibly be faster. This does somewhat defeats the purpose of the encoder.

The bit rate needed to transmit either a H.264 or a JPEG video stream are quite different as H.264 is quite a bit more efficient in terms of bit rate. On average a low resolution (704x480) video stream running at 30fps would require roughly $1.4\mathrm{Mbit/s}$ to be transmitted, while a MJPEG with the same specification would require roughly $7.2\mathrm{Mbit/s}$. If we increase the resolution, the requirements on the bit rate increases as well. For instance for H.264 at 720p resolution at 30fps a bit rate of roughly $3.8\mathrm{Mbit/s}$ is required while for MJPEG it would be $19.2\mathrm{Mbit/s}$. This is definitely worth taking into account when deciding on which solution to use.

Depending on which MCS index is available the bit rate might be an issue. For instance, if the MCS index is zero for IEEE 802.11n and the channel width is $20\mathrm{MHz}$, then available bit rate would be $6.5\mathrm{Mbits/s}$. This means that for a low resolution video, the bit rate is sufficient for H.264, but might not be for JPEG. With a moving unit the MCS might change quite a bit and therefore affect the quality of the connection. LTE doesn't seem to suffer as much when the unit is moved, the conditions for whether the LTE connection is stable or not are more dependent on general range to the nearest base station as well as the overall number of units connected to it.

The standard deviation for JPEG is consistently lower than the standard variation for H.264, suggesting a more stable video stream for JPEG under the conditions of this project.

6.2 Suggested Solution

Motivation of Solution

Looking back at the problem description, we were asked to look for a solution that could supply a video stream with a latency of at most $200\mathrm{ms}$ and

preferably around 100ms. We did after a while feel like these limits were perhaps unreasonable for any kind of wireless connectivity as we in the earlier stages had around 200ms when working between two computers on a wired network, however finding some better software and the use of better hardware easily reduced this time to within limits which could be handled.

Looking at the results of our different tests does give some interesting insights. If 200ms is to be considered a hard limit then both of the LTE solutions are nonviable, even if you account for certain delays with relaying data through external hosts. It would perhaps be possible to make LTE work with the JPEG encoding with some tweaks but even then it would rely heavily on the connection being stable.

Wi-Fi on the other hand has acceptable results with both coding standards. Even with the standard deviations the results seem well within limits. When looking at the optimal limit for the solution of 100ms it looks like both JPEG and H.264 are fairly close and both could therefore be suitable solutions. Since JPEG transmits compressed whole images in its stream it requires a higher bit rate to work, however H.264 can work with substantially lower bit rate which would be desirable with the application in mind. There are however other problems which Wi-Fi bring to the solution, which is mainly limited range. In general a vehicle is used to move large distances and this would take you out of range from an access point rather quickly. This is why LTE would be desirable if it could work in acceptable ranges of latency for remote controlled steering over a video stream. For the purpose of the demo event, the vehicle will not move much, at most 20m on a stage and thus Wi-Fi is still a suitable solution.

The standard deviation of the latency gives an indication of the stability of the connection. Most values will be within the range from the expected value. From the measurements of the standard deviation we find that for both LTE and Wi-Fi, the deviation is lower for JPEG. This suggests that the time to encode each frame with JPEG is less variable than the time to encode each frame using H.264.

Further insight can be derived from the ECDF of each measurement. For Wi-Fi the results are heavily weighted towards 100ms, while for LTE the distribution is wider and thus would suggest higher instability in the connection.

Another aspect which we haven't investigated as much in this project is the effect that different bit rates have on the different kinds of connections. The most important concept which wasn't tested in this project is throttling the bit rate of connections to see how much it compromised the connections. In the LTE case this could complicate things a bit as our bit rate with our LTE solution both have to depend on our 4G router as well as our 4G Internet service provider. However in our Wi-Fi tests the only access point between the test computers was a single Wi-Fi router which provides the highest bitrate it supports. This may cause the LTE to be a far less viable solution than it may seem, but we do however doubt that it actually is, seeing as the resolution we choose was 480x640 at 30fps it would not require that high of a

bit rate to work, in comparison with modern streaming services which can deliver up to 1080x1920 resolution over a megabit connection. Furthermore we do not really believe the bit rate to influence the results that much seeing as H.264 in the LTE case has a way higher latency than JPEG, which kind of contradicts the facts. Since the H.264 coding standard is implemented to reduce the required bit rate of the connection it feels weird that it should have more than 15% higher average latency than JPEG if bit rate was the problem.

When using both LTE and Wi-Fi there is a varying degree of packet-loss, this metric is however highly volatile and is very prone to being affected by external interference. Looking at table 5.2 and 5.4 it is easy to see that the packet-loss rate in the LTE cases is noticeably higher. This could be because the LTE connection goes through more network nodes and therefore has that many more occasions where a packet can be lost. As LTE could not guarantee that at least 99.9% of the packets sent were received it is fair to assume that it could affect key frames and thus create artifacts in the stream. This is undesirable behavior for the application during a demo event with a high number of people near by. It could however be used for monitoring the vehicles positioning during testing on a test track where Wi-Fi would not be a suitable solution.

Suggestion

With all this in mind, the solution proposed for the Uniti demo would be to utilize Wi-Fi. This would allow flexibility in the choice of encoder as there is inconclusiveness for which is definitely better in terms of latency. This would be a suitable solution for the event as it gives more control over the situation. For instance, a quick measurement to see which channels are heavily populated and make sure to avoid these. The monitoring computer can be set up such that the Wi-Fi receiver always is within range of the vehicle.

It could also be a factor that a lot of people are present at the demo, which could load the network on which the system operates. This is however not a big problem as all the tests have been constructed in environments under varying loads.

Considering other factors, for the demo we would suggest using MJPEG due to the characteristics of encodings that were taken into consideration. If when using H.264, an I-frame were to be dropped or otherwise lost, artifacts would be introduced to the video stream, thus making it difficult to discern the state of the area ahead of the vehicle. This could make the decision whether or not to stop the vehicle more difficult. In comparison if a frame were to be lost in an MJPEG stream, a slight stutter might be visible, but the next frame would quickly mend that. This is also a cause for concern, as a stutter at 30fps adds roughly 33ms of latency. Consecutively dropped frames could add to the latency. This seems unlikely however, based on the measurements taken during testing of the system.

This solution is however intended for the demo and might not be a viable solution when used in a vehicle under regular traffic conditions, that is when the vehicle is actually out on the road. This is because this would require coverage over a large area, which LTE might provide, but when creating our solution the priority was to make it work for the Uniti demo. This solution assumes that the vehicle would drive in a confined and controlled environment.

6.3 Future Work

In this project, only software codecs have been considered, so for further improvements hardware encoders could be used if possible. There are also a lot of other coding standards which could be used as well to expand on the tests, for example there are newer versions of H.264 which compresses frames further.

Another part that would have been of interest to look further into would be the visual fidelity that different encoders yield in different environments, it is something that could have a large impact on the feasibility of different encoders for the application of this project.

Quick tests with a camera providing MPEG video yielded a resulting latency of around 58ms, and this solution was thusly selected for cameras streaming internally in the vehicle to the infotainment system.

It would have been interesting to have a section in the project about bit rate and what effect throttling the bit rate could have on the quality and performance of a video stream.

As 5G networking is approaching and is being discussed to become the standard for vehicle-to-everything (V2X) communication [23], the viability of using 5G as a transmission medium for the purpose of remote monitoring of vehicles could be an interesting way to take this project further.

Chapter A

Test Script

A.1 Camera Side Test Script

```
#!/bin/bash

HOST=192.168.1.12
PORT=5004

if [ "$1" == "" ];
then
        echo "usage: $0 [options]";
        exit 1;
fi;

WIDTH=640;
HEIGHT=480;
ENCOPT="";
ENC="jpegenc";
PAYLOADER="rtpjpegpay";
IFRAMES="";

while getopts w:h:e:d:p:i: option
do
        case "${option}"
        in
        h) HEIGHT=${OPTARG};;
        w) WIDTH=${OPTARG};;
        e) ENCOPT=${OPTARG};;
        d) HOST=${OPTARG};;
        p) PORT=${OPTARG};;
        i) IFRAMES="key-int-max=${OPTARG}";;
        esac
done


if [ "$ENCOPT" == "jpeg" ];
```

```
then
        ENC="jpegenc";
        PAYLOADER="rtpjpegpay";
elif [ "$ENCOPT" == "h264" ];
then
        ENC="x264enc
                pass=qual
                bitrate=8192
                $IFRAMES
                tune=zerolatency"
        PAYLOADER="rtph264pay";
else
        echo "Unsupported encoder option: $ENCOPT";
        echo "Supported options are:";
        echo -e "  * jpeg";
        echo -e "  * h264";
        exit 1;
fi

echo "starting camera stream with enc $ENCOPT";
echo "Target host:      $HOST";
echo "Target port:      $PORT";
echo "Encoder:  $ENC";
echo "";

gst-launch-1.0 v4l2src ! \
        video/x-raw,width=$WIDTH,height=$HEIGHT ! \
        timeoverlay ! \
        videoconvert ! \
        $ENC ! \
        $PAYLOADER ! \
        udpsink host=$HOST port=$PORT

exit 0;
```

A.2 Monitor Side Test Script

```
#!/bin/bash

PORT=5004

while getopts e:p: option
do
        case "${option}"
        in
        e) ENC=${OPTARG};;
        p) PORT=${OPTARG};;
```

```
        esac
done

if [ "$ENC" == "jpeg" ];
then
        gst-launch-1.0 udpsrc port=$PORT ! \
                "application/x-rtp, payload=26" ! \
                rtpjpegdepay ! \
                jpegdec ! \
                videoconvert ! \
                xvimagesink
elif [ "$ENC" == "h264" ];
then
        gst-launch-1.0 udpsrc port=$PORT ! \
                "application/x-rtp, payload=127" ! \
                rtph264depay ! \
                avdec_h264 ! \
                videoconvert ! \
                xvimagesink #sync=true -v
else
        echo "Unsupported decoder...";
fi

exit 0;
```

Bibliography

[1] *802.11ac: The Fifth Generation of Wi-Fi - Accessed 2018-07-23*. Technical White Paper. Jan. 2018. URL: https://www.cisco.com/c/dam/en/us/products/collateral/wireless/aironet-3600-series/white-paper-c11-713103.pdf.

[2] Zohreh Azimifar Amir Sodagaran Narjes Zarei. "Intelligent Traffic Information System a Real-Time Traffic Information System on the Shiraz Bypass - Accessed 2018-07-23". In: (July 2016). URL: https://www.matec-conferences.org/articles/matecconf/pdf/2016/44/matecconf_ictte2016_04003.pdf.

[3] L. Berc et al. "RTP Payload Format for JPEG-compressed Video - Accessed 2018-07-23". In: (Aug. 1998). URL: https://tools.ietf.org/html/rfc2435.

[4] Fredrik Björkman. *Efter vändorna: Volvo får grönt ljus för tester av självkörande bilar - Accessed 2018-12-17*. URL: https://digital.di.se/artikel/efter-vandorna-volvo-far-gront-ljus-for-tester-av-sjalvkorande-bilar.

[5] Axis Communications. "Latency in live network video surveillance - Accessed 2018-07-23". In: (2015).

[6] Behrouz A. Forouzan. *Data Communications and Networking*. Fifth Edition, p. 699. ISBN: 978-0-07-131586-9.

[7] Behrouz A. Forouzan. *Data Communications and Networking*. Fifth Edition, pp. 979–981. ISBN: 978-0-07-131586-9.

[8] Wireshark Foundation. *Wireshark Features - Accessed 2018-07-23*. URL: https://www.wireshark.org/about.html.

[9] Mobile Network Guide. *What is a Mobile Base Station? - Accessed 2018-07-23*. URL: http://mobilenetworkguide.com.au/mobile_base_stations.html.

[10] R. Frederick V. Jacobson H. Schulzrinne S. Casner. "RTP: A Transport Protocol for Real-Time Applications - Accessed 2018-07-23". In: (July 2003). URL: https://tools.ietf.org/html/rfc3550.

[11] Wilhelm Ochsenreiter Hermann Kopetz. "Clock Synchronization in Distributed Real-Time Systems - Accessed 2018-07-23". In: (Aug. 1987)). URL: https://ieeexplore.ieee.org/abstract/document/5009516/.

[12] "IEEE 802.11ac: What Does it Mean for Test? - Accessed 2018-12-17". In: (2013). URL: http://litepoint.com/whitepaper/80211ac_Whitepaper.pdf.

[13] Telecommunication Standardization Sector of ITU. "H.264 standard - Accessed 2018-07-23". In: (Apr. 2017)). URL: https://www.itu.

int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-201704-I!!PDF-E&type=items.

[14] Chris Paukert Kyle Hyatt. *Self-driving cars: A level-by-level explainer of autonomous vehicles - Accessed 2018-07-23*. URL: https://www.cnet.com/roadshow/news/self-driving-car-guide-autonomous-explanation/.

[15] Pearson IT Certification Mike Harwood. *802.11 Wireless Standards - Accessed 2018-07-23*. URL: http://www.pearsonitcertification.com/articles/article.aspxp=1329709&seqNum=4.

[16] J. Postel. "User Datagram Protocol RFC - Accessed 2018-07-23". In: (Aug. 1980). URL: https://tools.ietf.org/html/rfc768.

[17] *Renault SYMBIOZ Concept - Accessed 2018-12-17*. URL: https://www.renault.co.uk/vehicles/concept-cars/symbioz-concept.html.

[18] Information Sciences Institute University of Southern California. "Transmission Control Protocol RFC - Accessed 2018-07-23". In: (Sept. 1981). URL: https://tools.ietf.org/html/rfc793.

[19] *State of LTE - Accessed 2018-07-23*. URL: https://opensignal.com/reports/2018/02/state-of-lte.

[20] The Tcpdump team. URL: http://www.tcpdump.org/manpages/tcpdump.1.html.

[21] *Tesla Model S Owner's Manual - Accessed 2018-07-23*. Aug. 2018. URL: https://www.tesla.com/sites/default/files/model_s_owners_manual_north_america_en_us.pdf.

[22] Gisle Bjøntegaard Ajay Luthra Thomas Wiegard Gary J. Sullivan. "Overview of the H.264/AVC Video Coding Standard - Accessed 2018-07-23". In: (July 2003). URL: http://ip.hhi.de/imagecom_G1/assets/pdfs/csvt_overview_0305.pdf.

[23] Hugues-Antoine Lacour Andrew Killeen David McClure Alain Dunoyer Tom Rebbeck Janette Stewart. *5GAA Study | The cost-benefit analysis on cellular vehicle-to-everything (C-V2X) technology and its evolution to 5G-V2X - Accessed 2018-12-17, Url = http://5gaa.org/news/5gaa-study-the-cost-benefit-analysis-on-cellular-vehicle-to-everything-c-v2x-technology-and-its-evolution-to-5g-v2x/.*

[24] *Waymo Journey - Accessed 2018-07-23*. URL: https://waymo.com/journey/.

Printed by Tryckeriet i E-huset, Lund 2018

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2018-679
http://www.eit.lth.se