Expertise
and insight
for the future

Timo Blomqvist

Video latency in a vehicle's remote operating system

Metropolia University of Applied Sciences

Bachelor of Engineering

Information and Communications Technology

Bachelor's Thesis

8 November 2018

Author Title Number of Pages Date	Timo Blomqvist Video latency in a vehicle's remote operation system 37 pages 8 November 2018
Degree	Bachelor of Engineering
Degree Programme	Information and Communications Technology
Professional Major	Media Engineering
Instructors	Toni Spännäri, Senior Lecturer

This thesis report contains information about live video streaming and video latency. Various components of a live streaming system can significantly affect the delay between capturing video on a capture device such as a video camera and transmitting it to a remote display. Research about what these factors are, what kinds of standards and protocols are used with them and how they behave together was conducted. In addition, latency measurement techniques were also researched to determine a method for effective live monitoring and measuring of video latency.

The purpose of this thesis was to determine the liability of modern live streaming techniques in remote operation of a vehicle. For this reason, tests regarding video latency were conducted to find out how much end-to-end latency occurred in a remote operation system and how much each component of said system contributed to the latency.

The thesis was carried out as part of one of Metropolia's projects, ROBUSTA. The purpose of ROBUSTA is to research technologies that would enable full remote operation of autonomous buses in the future. As development of traffic laws to introduce automation of vehicles to public roads moves further, safety and functionality of live streaming systems in a remote operation environment will become crucial in developing safe and stable remote operation systems.

Keywords	remote operation, latency, video, streaming, GStreamer

Tekijä Otsikko Sivumäärä Aika	Timo Blomqvist Videolatenssi etäohjattavan ajoneuvon hallintajärjestelmässä 37 sivua 8.11.2018
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	tieto- ja viestintätekniikka
Ammatillinen pääaine	mediatekniikka
Ohjaaja	lehtori Toni Spännäri

Insinöörityössä tutkittiin livestreamausta ja siinä ilmenevää videolatenssia. Videolatenssi tarkoittaa viivettä, joka alkaa kameran videokuvan kaappauksesta ja päättyy videokuvan näkymiseen näyttöpäätteessä reaaliajassa. Videolatenssi aiheutuu useimmiten videon prosessoinnista, ja livestreamausjärjestelmissä useat eri komponentit voivat vaikuttaa videolatenssiin merkittävästi. Työssä suoritettiin tutkimustyötä näihin komponentteihin liittyen.

Työn tarkoituksena oli myös selvittää, miten nykypäivän livestreamausteknologia sopi ajoneuvon etähallintaan. Tätä varten tehtiin videolatenssimittauksia, joista selvisivät livestreamausjärjestelmän kokonaislatenssi ja järjestelmän komponenttien yksilölliset osuudet kokonaislatenssista. Työssä ilmeni sekä livestreamauksessa käytettävän laitteiston aiheuttama latenssi että ajoneuvon etähallintaa ja livestreamausta varten käytetyn 4G-mobiiliverkon vaikutus kokonaislatenssiin. Tämän lisäksi työssä tutkittiin eri latenssinmittaustekniikoita, joilla voitaisiin suorittaa mittauksia helposti ja tarkasti. Näitä tekniikoita käytettiin työn aikana latenssimittauksissa. Mittauksissa selvisi, että laitteiston videoprosessointi aiheutti lähes yhtä suuren määrän latenssia kuin mobiiliverkko. Molemmat tekijät kattoivat kokonaislatenssista noin puolet. Koska työssä käytetyn laitteiston prosessointikapasiteetti oli kohtalaisen pieni, mobiiliverkon osuus livestreamausjärjestelmän latenssissa voitiin päätellä merkittävimmäksi, koska mobiiliverkkojen toimintaan on haasteellisempi vaikuttaa.

Työ tehtiin osana ammattikorkeakoulun hanketta, jossa tutkitaan ja testataan teknologioita, joita voitaisiin tulevaisuudessa hyödyntää itseohjautuvien bussien etähallintajärjestelmissä. Tieliikennelakeja kehitetään jatkuvasti modernimpaan suuntaan mikä voi mahdollistaa vähintään osittaisen liikenteen automatisoitumisen tulevaisuudessa. Työn tulokset antoivat hankkeelle enemmän tietoa siitä, mihin tekijöihin on kiinnitettävä huomiota, kun on kyse videolatenssista. Hankkeessa on tarkoitus tulevaisuudessa tutkia mobiiliverkkoteknologian toimintaa ja käyttäytymistä etähallintaskenaarioissa. Livestreamausjärjestelmien toimivuus on siis tärkeä osa toimivan ja turvallisen etäohjausjärjestelmän kehittämistä.

Avainsanat	etäohjaus, latenssi, viive, video, streamaus, GStreamer

Contents

List of Abbreviations

Bitrate	The number of bits of data that is transferred per unit of time.
H.264	Video coding standard, also known as Advanced Video Coding (AVC).
H.265	Video coding standard, also known as High Efficiency Video Coding (HEVC).
HLS	HTTP Live Streaming, media streaming protocol.
MJPEG	Motion JPEG, video coding standard.
NAL	Network Abstraction Layer, part of H.264 and H.265.
RTMP	Real-Time Messaging Protocol, a media streaming protocol.
RTP	Real-time Transport Protocol, a media streaming protocol.
TCP	Transmission Control Protocol, an internet transport protocol.
UDP	User Datagram Protocol, an internet transport protocol.
VCL	Video Coding Layer, part of the H.264 and H.265 standards' architecture.
Wi-Fi	Wireless local area network.

1 Introduction

The thesis was carried out to determine the different factors of live video streaming latency in a remote monitoring environment and how wireless networking behaved while utilizing multiple streams. The environment consisted of remote controlling a small vehicle in a place where the vehicle could not be seen. To give the operator of the vehicle visibility while driving, it was rigged with several cameras that were used to stream live video to the control center screens. This testing environment was used in the case project to research user experience in remote operating a vehicle. The project that this thesis was worked on also consisted of conducting research regarding technologies related to live video streaming, networking and video coding. This research was conducted to realize the different aspects of a basic live streaming system, how they relate to each other and how much they typically contribute to the video latency.

In general, live video streaming means taking a video source, e.g. a video camera, and showing it on a remotely located display in real time, meaning that the delay between capturing the video and viewing the result remotely needs to be minimal. This delay is also referred to as end-to-end latency. In live streaming the video signal is transported through a network. To send the signal through said network and view it in the relatively or precisely the same quality as it was in the capture, it needs to be processed which can all be responsible for additional latency. There are also techniques that are often required to ensure a stable viewing experience without unnecessary interruptions in the video feed such as temporary data storage, also known as buffering. It is also important to note that the latency caused by some factors can be dependent on other factors in the streaming system. An example of this is the buffer's dependency on the stream's bitrate. These issues and more are further discussed in section 3 of this report.

2 Background

2.1 Legislation of remote operation

The current traffic legislation of Finland states that a mobile vehicle is not initially required to have its operator controlling it. The only requirement is that the vehicle has a person present who can take direct control of the vehicle if needed and pays constant attention to the vehicle's surroundings while in motion. [1.]

As of now, the traffic laws prohibit any vehicle to operate without a driver. This is based on the international road traffic treaty signed in Vienna in 1968 which states that "the driver must be physically and mentally capable of fulfilling their task and they must be in a physical and mental state that is required to drive a vehicle". [1.] However, Finland's traffic legislation is currently being developed further towards a state which would allow almost total automation of vehicles, meaning that the driver's requirement to be physically present would be removed. The Finnish Transport Agency has been conducting research related to the subject and predicts that by the year 2020 the development would have reached a state where the driver of the automated vehicle would not be physically required to watch their surroundings but instead concentrate on other activities if they so desired. [2.] They would still be required to be present in the vehicle, but as the development of traffic laws in Finland move further it is not impossible to assume the driver's physical presence be removed. For this reason, traffic automation technology development in Finland and other countries is moving forward with safety in mind. Several different projects related to the subject are underway, one of them being the ROBUSTA project.

2.2 Importance of video latency

In an environment where the operator of a vehicle of any kind is not physically present inside the vehicle, controlling the vehicle and visibility of the vehicle's surroundings both inside and outside becomes limited. This can be considered a significant safety hazard, especially if the vehicle is designed to move at high speeds and/or transport people. To

make seeing the vehicle and its environment easier for the remote operator, surveillance cameras in addition to wireless live streaming technology must be applied.

Since video latency is a common occurrence in live streaming, it can introduce additional safety risks to the vehicle and everything around or inside it. Generally, the higher the end-to-end latency of a vehicle surveillance system is, the longer the operator must wait until they can see any events occurring near the vehicle. This waiting period can be considered a visual blind spot for the operator because while they will be able to see the event, they will see it significantly later than when it occurred.

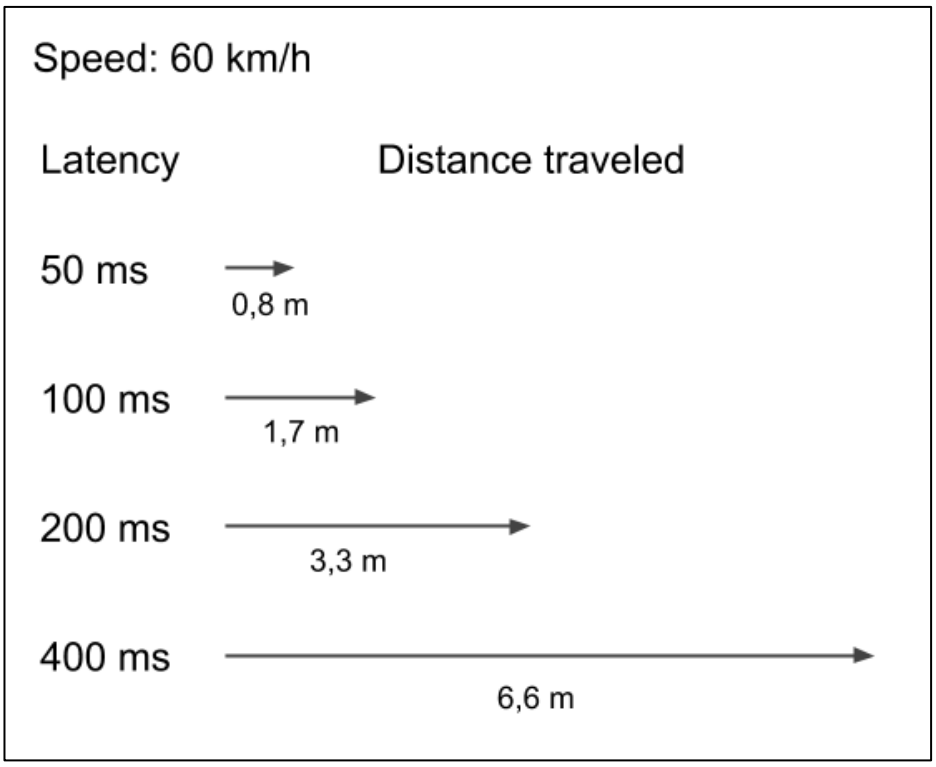

Image 1. A diagram of video latency's correlation to vehicle's travelled distance during the waiting period with an even driving speed of 60km/h.

With a video latency of more than 200ms and high vehicle speed the distance the vehicle travels before the necessary visual information is delivered to the operator, it will have moved forward for over 3 meters. Considering that the operator will react according to what is presented to them on screen the latency correlates with human reaction times.

This means that video latency can indirectly increase the reaction time to levels that can be considered hazardous in an emergency.

It is also important to note that the distance traveled during the latency period is directly dependent on the vehicle's speed as illustrated in the diagram below.

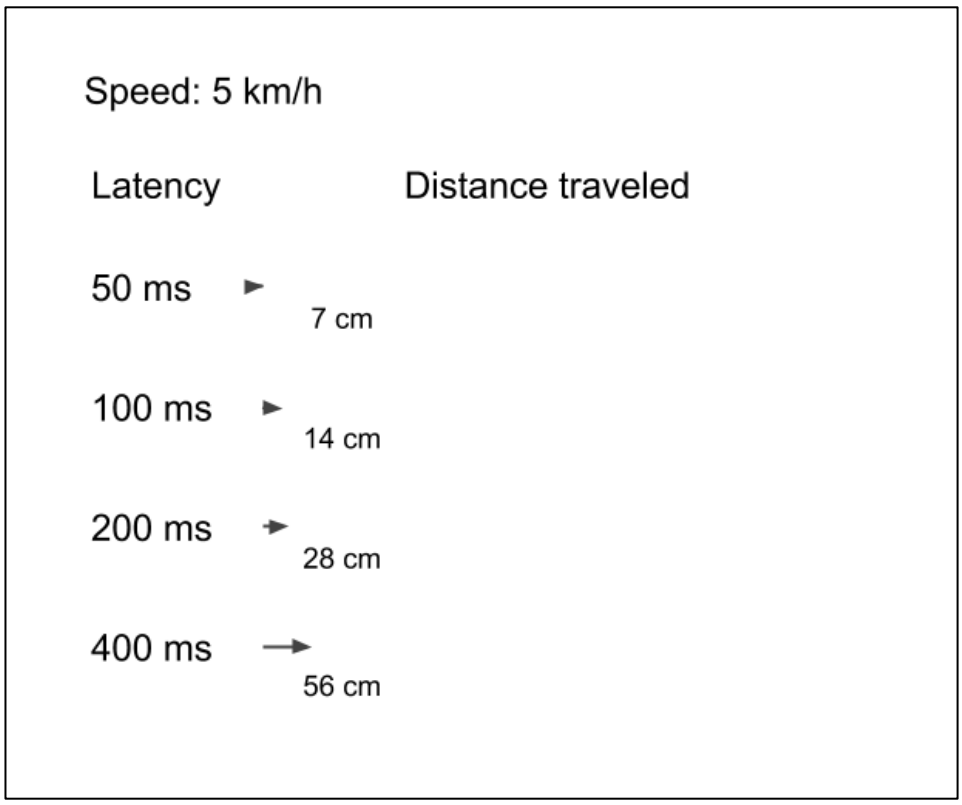

Image 2. Traveled distance of a vehicle during latency with a speed of 5km/h.

With lower speeds video latency becomes considerably less significant with the vehicle moving less than a hundred centimeters with a latency of 400ms. However, driving speeds of 5-20km/h is not as common in modern traffic since speed limits of 30km/h or higher are used in all public roads. Speeds lower than 30km/h are mostly present in parking areas. This makes video latency extremely important in terms of safety in remote operation especially when considering future development of automated vehicles. Being able to determine the factors that cause end-to-end latency and how much each part of a streaming system affects, it is crucial in creating a safe operating environment for the driver of the vehicle, passengers and the surrounding traffic.

3 Live video streaming components

Live video streaming means streaming content captured and output by a video camera or other capture hardware or software that has real time output capabilities such as live screen capture on a computer. After capture the video is sent to a processing component called an encoder where the video signal is converted to a data form, so it can be sent through an IP network with data packets. The video might also be compressed to a smaller size to compensate for the available upload bandwidth of the network. The methods used in the encoding and compressing process are defined by standards developed for coding media and depend on the hardware or software that is used in the stream. [3.]

The video data is sent through the network with a media transfer protocol using packets. The stream is typically sent to a content delivery network (CDN) for distribution to one or several clients. [3.] When receiving the stream, the client must convert it to a viewable form before it can be shown on a video player. This is called decoding and the methods used to decode video data are also dependent on the earlier mentioned coding standards and can be achieved with currently available hardware or software.

3.1 Video encoding and compression

In compression, the data that is handled is reduced to a smaller number of bits than its original size. Data compression is achieved by removing data that is unnecessary for creating an accurate representation of said data (redundancy). In live video streaming this means removing parts of a video sequence before sending it over a network. Some types of data can be converted using lossless compression, mirroring the quality of the source data perfectly. A disadvantage of implementing lossless compression is that it requires a significant amount of bandwidth. Lossy compression on the other hand uses a much higher level of compression with some reduction in visual quality. In lossy compression, some video sequences can contain elements that can be removed without significantly affecting quality. [4.]

If not compressed, the video signal can require an immense amount of bandwidth to be transported which most data links do not provide, making compression important for

streaming. Compression can be achieved with either media software or physical hardware made specifically for encoding and decoding video. [4.]

Most video coding methods use spatial and temporal coding algorithms to exploit redundancy. Temporal coding makes use of comparing adjacent frames in a timeline. Especially with high frame rates, there is often high correlation between frames. In spatial coding, high correlation can be observed in adjacent pixels in a frame. [4.]

Image 3. Difference between temporal and spatial correlation [3].

Most modern encoding standards are based on a codec model containing predictive coding, motion compensation, transform, quantization and entropy coding. Typically, the encoder consists of three units: a prediction model, a spatial model and an entropy encoder. The video signal to be compressed is first processed in the prediction model where redundancy is removed by comparing either differences in frames (inter prediction) or neighboring samples in a single frame (intra prediction). The prediction model

outputs a residual frame that consists of a frame that has had its prediction samples removed and parameters indicating either intra or inter prediction. [4.]

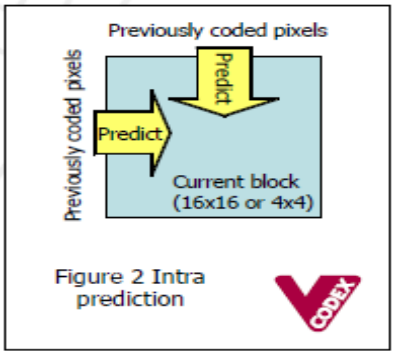

Image 4. In intra prediction, the coding unit is formed based on the block's surrounding, previously-coded pixels in the I-frame. [5.]

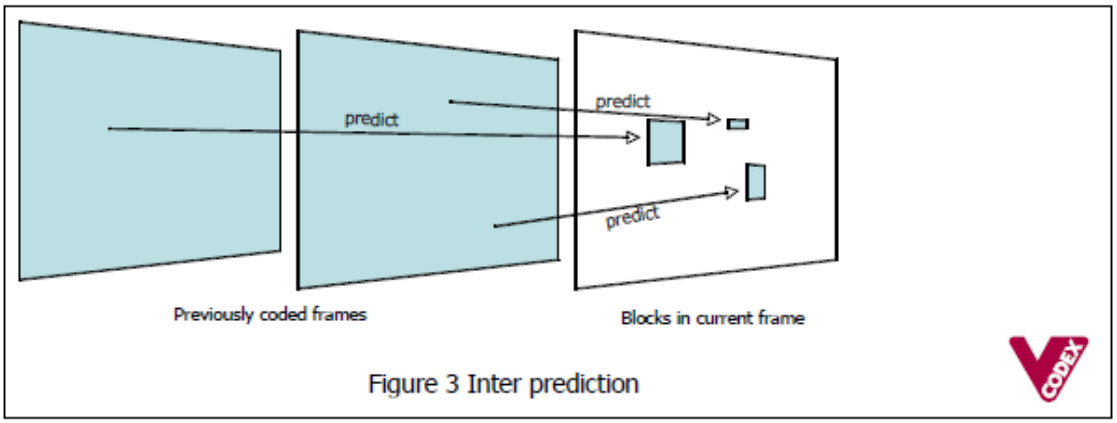

Image 5. In inter-prediction, blocks of pixels are predicted from similar regions in previous frames [5].

The residual frame is transferred to the spatial model that removes spatial redundancy. The residual samples are turned into transform coefficients and are then quantized. The parameters of both the prediction model and spatial model are sent to the entropy encoder where the frame is compressed by removing more redundancy. The compressed bitstream is then transported to a decoder where the video frames are reconstructed by decoding the parameters for the prediction and spatial models. [4.]

While the basic principle for temporal and spatial coding is the same for all encoders, the standards that have been developed to define ways of compressing video differ in their methods. This makes some standards more viable than other in terms of efficiency and accuracy.

3.2 Encoding and compression standards

In live video streaming, the video signal that the capture device (e.g. a video camera) outputs needs to be converted to a format that can be transferred through a network. This method is known as encoding. After the signal has been encoded into data it is transmitted through a link into a decoder. The decoder then converts the signal back into its original form before it was encoded. There are several different standards that define ways of how the encoding and compressing video is achieved. These standards go through several versions after they have been introduced because of the available technology constantly changing in the media industry. This section contains information on coding standards that have been observed to be the most commonly utilized as well as a new standard that is bound to replace its predecessors in the future.

3.2.1 H.264

The H.264 codec, also known as AVC (Advanced Video Coding) or MPEG-4 Part 10, is a part of the MPEG-4 standard for encoding and playing back time-based media. It is currently one of the most common encoding formats used today because of its efficiency, requiring a smaller bitrate for same or better-quality compression with the same bitrate as its predecessors such as H.263. The difference in H.264's bitrate usage compared to H.263 can be seen in figure 1 below.

	Average bit-rate savings relative to:		
Coder	MPEG-4 ASP	H.263 HLP	MPEG-2
H.264/AVC MP	37.44%	47.58%	63.57%
MPEG-4 ASP	-	16.65%	42.95%
H.263 HLP	-	-	30.61%

Figure 1. The comparison of bitrate savings from a test regarding codec performance [6].

As can be seen from figure 1, H.264 requires almost 50% less bitrate to achieve the same quality of video than H.263.

In encoding, the frame of a video is processed in units called macroblocks that are generally 16x16 displayed pixels in size. A prediction of a macroblock is formed based on already coded data. This can be either in the form of the current frame or frames that have been coded after it. The prediction is then subtracted from the current macroblock to form a residual. H.264 makes the prediction more accurate, using block sizes from 16x16 to 4x4 pixels to predict the macroblock. [5.] Handling of video data in the H.264 standard is done in different layers called the Video Coding Layer (VCL) and Network Abstraction Layer (NAL). VCL consists of all the elements related to compressing the video such as prediction and motion compensation. NAL contains the coded video sequence converted into NAL units, otherwise known as “packets”. This allows for them to be mapped to the transport protocols (e.g. RTP). There are also some transport protocols that do not use packets to deliver content, which is why H.264 has also defined a byte-stream format to transport NAL units as ordered streams of bytes. [6.]

H.264 also has a set of profiles that determine its behavior in applications. These profiles contain a profile code and a set of constraints applied to the encoder: [7].

- **Baseline Profile**: Used in low-cost applications that require additional data loss robustness, most notably video conferencing and mobile.
- **Main Profile**: Used in standard-definition digital TV broadcasting.
- **Extended Profile**: For streaming video, this profile introduces high compression capability and data loss robustness.
- **High Profile**: Used in high-definition broadcasting and data storage (Blu-Ray and HD DVD).
- **High 10 Profile**: Enhanced version of the High Profile, this profile is meant for applications beyond consumer product capabilities. Supports up to 10 bits per sample of decoded picture precision.
- **High 4:2:2 Profile**: Enhanced version of the High 10 Profile, targeted for professional applications that use interlaced video, enabling 4:2:2 chroma subsampling.
- **High 4:4:4 Predictive Profile**: Enhanced version of the High 4:2:2 Profile, enables 4:4:4 chroma subsampling up to 14 bits per sample and supports efficient lossless region coding and the coding of each picture as three separate color planes.

Additionally, the standard has been given “all-Intra profiles” which are subsets of other already existing profiles. These subsets are mostly meant for professional applications. [5.] There are also scalable versions of some profiles enabled by the SVC (Scalable Video Coding) extension that was finalized in 2007. The idea of SVC is to allow multiple bitstreams of a video signal that have different resolution scales and rate levels. Scalability is available in the following categories: [8.]

- **Temporal**: change of frame rate
- **Spatial**: change of frame size
- **Fidelity**: change of overall quality of video.

Currently, H.264 is considered one of the most versatile codecs available to the public, which is why it is implemented in most of modern encoding software and hardware today. However, other options for encoding methods have risen over the years as technology has developed to a point where video compression efficiency can be increased by a wider margin.

3.2.2 H.265

H.265, or HEVC (High Efficiency Video Coding), is considered as a successor to H.264. Reducing bit rate by half while achieving the same video quality as H.264, HEVC, while being quite new (introduced in 2013), has already been implemented to several modern encoding software and hardware. However, given the larger scale of its efficiency compared to H.264, it requires significantly more processing power to work properly, which is why most HEVC encoding hardware (that have been observed) are of professional grade and can be quite expensive. That said, these encoders not only offer high quality streaming with a smaller requirement for bandwidth but also extremely low end-to-end video latencies (~150-200ms). Examples of this are the streaming solutions company VITEC's products, some of which advertise end-to-end latencies of ~160ms. [9.] It should be noted however that the product pages do not specify the capture equipment, or any method of data transmission used to achieve these latency levels meaning that this low amount should be considered as the best-case scenario where the capture and data transport delays are close to insignificant.

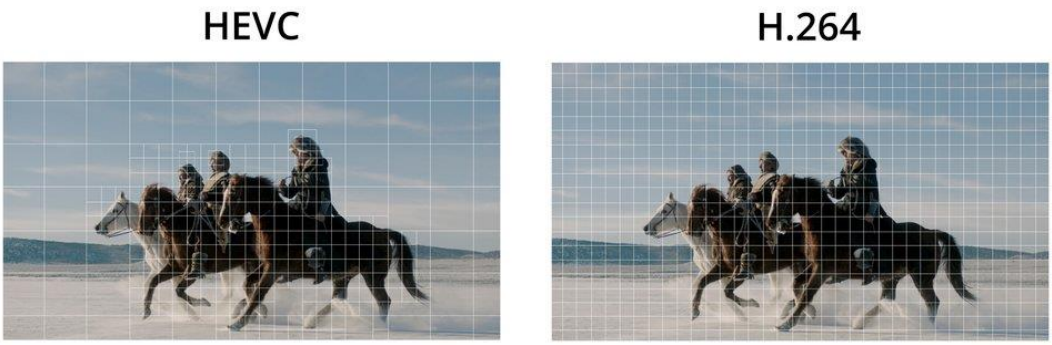

Image 6. Difference in coding blocks distribution between HEVC and H.264 [10].

While H.264 uses macroblock sizes up to 16x16 pixels, H.265 can use block sizes up to 64x64. Instead of macroblocks, H.265 uses coding tree units (CTU) that are made up of coding tree blocks (CTB). The units have either luma (brightness) or chroma (color) characteristics and a CTU consists of a luma CTB and two chroma CTBs. The CTU contains a quad-tree syntax enabling variable coding block sizes depending on the region covered by the CTB. An example of different block sizes is illustrated in image 4 above. The

coding tree blocks can be partitioned into coding blocks (CB). A coding unit (CU) is formed by luma CBs and chroma CBs. Each CU is given a prediction mode, either inter or intra. [11.]

With the significant increase in accuracy and efficiency of compression and encoding, H.265 can be considered vastly superior to H.264. Usability of H.265 is still quite limited, however, and is not applicable in all live streaming environments due to its requirement of higher-end hardware compared to H.264.

3.2.3 MJPEG

MJPEG encoding can be considered as a simpler version of H.264 and HEVC. Instead of using predictive methods to create images, MJPEG treats each frame as an individual JPEG image, meaning that none of the frames are dependent of each other in any way. This ensures that an MJPEG video can retain its quality without visual artifacts even if frames are dropped during the streaming process. [12.]

There are some disadvantages that come from using MJPEG. Since the frames aren't exposed to any compression techniques to reduce data size, the bitrate requirement for streaming stable high-quality video is quite high. It is also important to note that the end quality of a video frame is directly dependent of the complexity of it, meaning that frames that are made up of simple straight lines and smooth surfaces do not require as much bandwidth or storage space as complex scenes with more details do. [12.]

3.3 Internet transport protocols

Internet transport protocols are essential in providing end-to-end communication services to applications. They are responsible for delivering data to and receive from the host computers in a network which involves forming appropriate data packets for the protocol and adding necessary information to the packet header such as the destination port number. Currently, there are two protocols on the transport layer that are used most commonly called TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). In terms of handling and delivering data their workflow differs significantly which is

why both of their services aren't usually implemented in the same form of application. [13.]

3.3.1 Transport Control Protocol

The Transport Control Protocol (TCP) is a connection-oriented protocol, meaning that the hosts that TCP has established a connection between are aware of each other. The connection is maintained until all the information between the two hosts has been exchanged. The protocol can be considered reliable because delivery of packets is guaranteed with acknowledgements of sent data. In addition, if packets are lost during transmission they are sent again until they arrive to their destination. Essentially, the sending host must wait for the confirmation that the packets have been delivered before sending more. [14.] In a situation where the network is congested, TCP utilizes data buffers at both ends of the connection. Initially, the receiver specifies a maximum number of bytes that it's ready to accept in its buffer. The sender then sends the amount of data up to the specified size and waits for acknowledgement that all the data has been received. The sender buffers all its own sent data until that happens. [15.]

3.3.2 User Datagram Protocol

As opposed to TCP, User Datagram Protocol is considered a connectionless protocol. Unlike TCP, UDP does not establish a direct connection between two hosts and doesn't use acknowledgement messages, making it an unreliable protocol. It doesn't wait for confirmation that the data has been received and just keeps sending packets without knowing if they arrive to their destination. Checking for dropped packets or other errors are left as the application's responsibility. [14.]

While UDP is not considered as reliable as TCP in terms of keeping received data intact it can be more beneficial in certain scenarios. Since UDP does not perform regular checks on the data it sends and receives and doesn't retransmit any data lost on the way, its transmission rate is faster than TCP. In a situation where packet loss is not considered crucial to the application and the data needs to arrive quickly it is better to utilize UDP as a transport method. As an example, in multimedia streaming it is common

for packet loss to go unnoticed on the client's side and waiting to retransmit said packets would cause noticeable disruptions in the feed, making receiving the stream arduous and high in latency. [16.]

3.4 Media transfer protocols

Streaming media requires it to be transferred through the internet to its destination. The rising popularity of different streaming services has led to modifications of existing internet protocols to better support real-time applications. For this purpose, several different protocols have been defined with each having unique methods of handling packets of data and some enabling additional properties for controlling the stream and showing detailed information about it. [17.]

Live streaming quality of service is often determined by the quality of data flow. For example, the Transport Control Protocol (TCP) requires retransmission of a packet if it's lost during transmission. While it ensures all packets arrive to their destination, it can cause considerable delay in the stream and possible connection outages. Protocols like User Datagram Protocol (UDP) however allow continuous transmission of new packets even if some are lost on the way. The protocols detailed in this section are the ones that are most commonly used in media streaming applications.

3.4.1 Real-Time Transport Protocol

Real Time Transport Protocol (RTP) is specifically designed for delivering media data over the internet in real time. It is built upon the User Datagram Protocol (UDP). UDP doesn't support timestamping on its own which is partially why RTP was introduced. RTP does not guarantee keeping packets of data in a sequence, leaving it to the application itself. [17.] In live video streaming, this is usually left to the decoder [18]. Recovery of any lost data segments and reconstruction of the sequences is also left for the application layer to handle. RTP provides the type of the payloads, identification of the source, sequence numbering and timestamping. [17.]

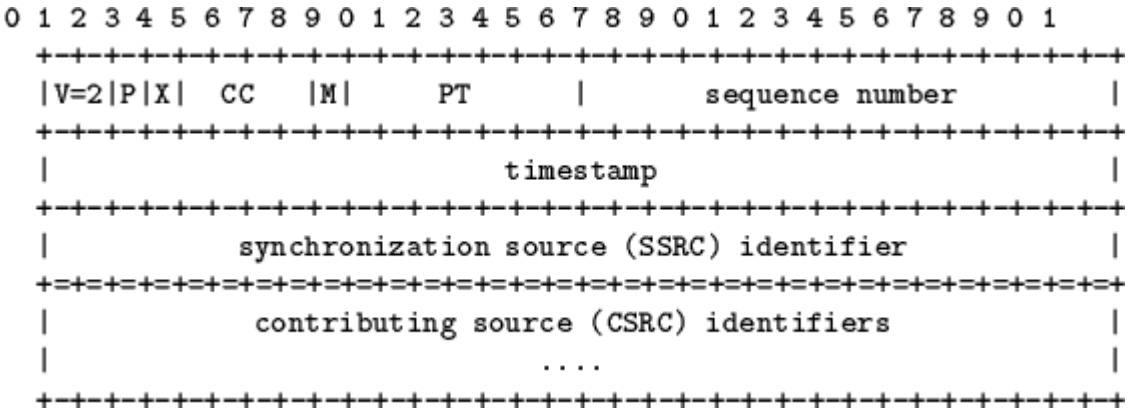

Image 7. RTP packet header structure [19]. The payload portion of the packet is determined by the application and contains the initial media data (e.g. encoded video).

RTP is paired with the RTCP (RTP Control Protocol) which provides information about the quality of the stream to users. It uses separate packets, which are similar in structure to RTP packets, to deliver the relevant information, e.g. packet loss and delay. [17.]

3.4.2 Real-Time Messaging Protocol

Real-Time Messaging Protocol (RTMP) is an application-level media streaming protocol originally developed by Macromedia for their own use with the intention of streaming Flash-video over the internet. After Macromedia was bought by Adobe, an unfinished version of the protocol was released to the public. Unlike RTP, RTMP is based on TCP. [20.] However, there exists a variation of RTMP called RTMFP (Real-Time Media Flow Protocol) which has been developed to use UDP in P2P (peer-to-peer) connections, enabling greater tolerance for dropped packets in the dataflow and smaller delay. This is because UDP does not require retransmission of lost packets. [21.]

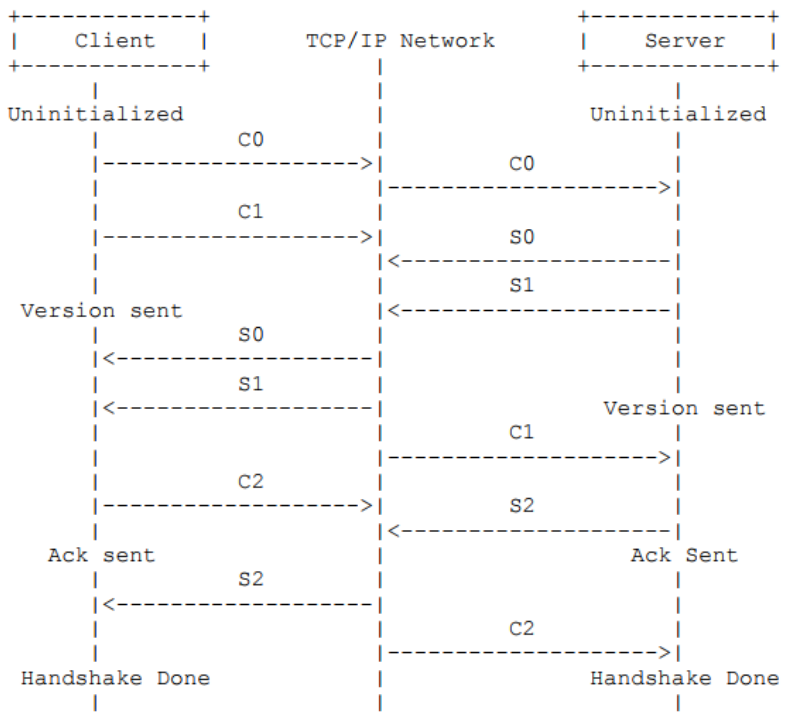

Image 8. RTMP handshake diagram. The client's chunks are represented by C0-C2 and the server's by S0-S2. [22.]

To establish a connection, RTMP requires a "handshake". In the handshake, three data chunks of the same size are sent by both the client establishing the connection and the server that the client is communicating with. The client starts by sending 2 chunks to the server. After the server has received the first chunk and, in some cases, the second chunk it may send its own 2 chunks to the client. However, it MUST wait for the second one to send its third chunk. After receiving the server's second chunk, the client sends its own third chunk to the server and must wait for the third chunk before it can send any more data. The same applies to the server. [22.]

3.4.3 HTTP Live Streaming

HTTP Live Streaming (HLS) is a streaming protocol that has been developed by Apple to be able to stream content to Apple devices. HLS utilizes adaptive streaming, meaning that the player showing the stream is distributed several files with varying bit rates to optimize playback for different devices. [23.]

The source is encoded into multiple files and divided into chunks which are usually a few seconds long. The files are sent to the server alongside a manifest file that directs the player to the streams. During playback, the player monitors bandwidth availability. If a change in bandwidth is large enough to require a different data rate, the player checks the manifest file for the location of the other streams. It then fetches the URL of the next video data chunk from the stream file. [23.]

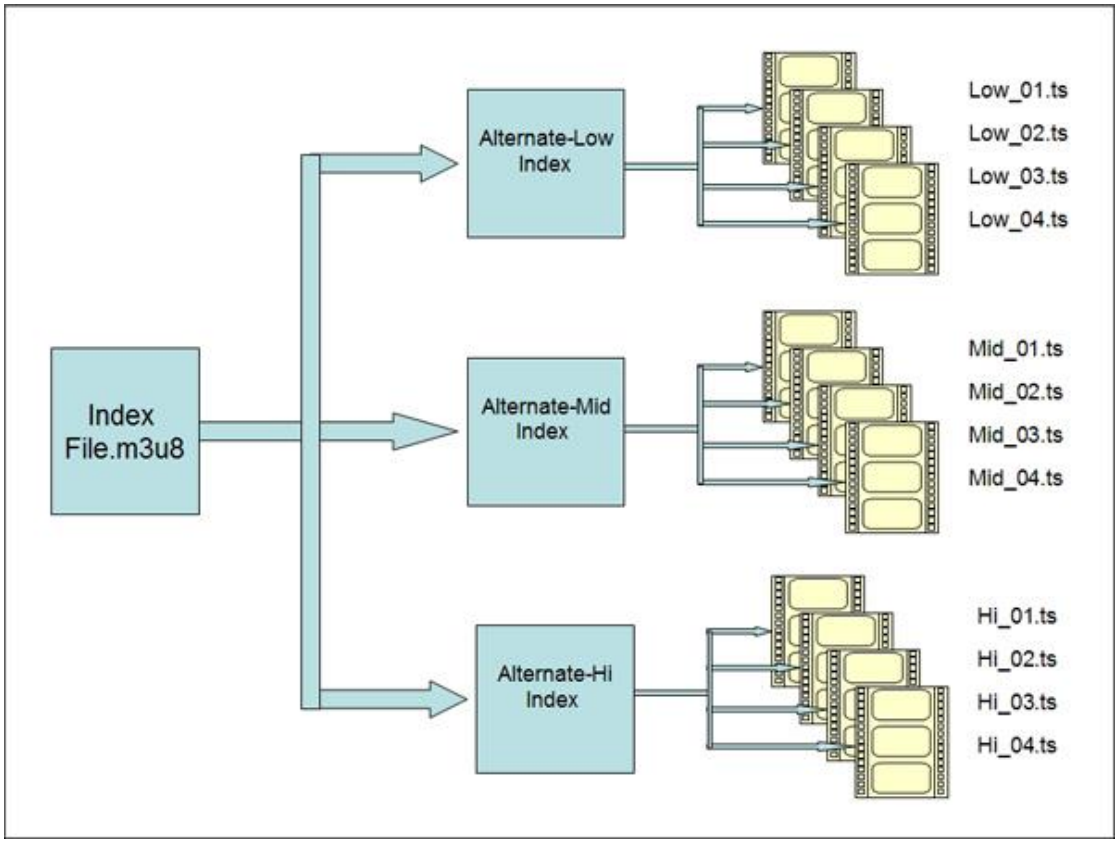

Image 9. HLS stream flow. The manifest file directs the player to index files of different streams which in turn direct the player to the video chunks which are indicated by the .ts extension. [23.]

HLS operates on HTTP (Hypertext Transfer Protocol) which is built upon TCP as its transport protocol. The encoded media that needs to be transmitted is segmented into containers (previously mentioned as chunks) in MPEG-TS (MPEG transport stream) format [23]. Essentially, the containers hold a sequence of packets which in turn hold the elementary streams that can consist of either MPEG-based encoded video data,

metadata like captions and information about the stream or non-MPEG encoded media data. [24.]

3.5 Capture devices

Depending on the devices used for capturing video that needs to be livestreamed, it is possible for them to introduce latency during the capture and output sequence. Considering that live streaming requires the video frames to be output in digital form the capture device must process them before they are sent out. The time is takes for a video frame to be processed depends on the capture device's capabilities. For example, some action cameras are not as well suited for real-time video output as video cameras used in studios and as such require more time to process the video frames. A video latency test that featured an action camera was carried out and is discussed in section 4.1 of this report.

In a video camera, frames are formed by pixels captured by the camera sensor. The latency that occurs from this is determined by the frequency at which the sensor captures frames, known as framerate which is measured in frames per second (fps). For example, if the framerate of the capture is set to 30 fps, the sensor captures a frame every 1/30th of a second resulting in a capture latency of about 33ms per frame. If the capture rate is set higher, for example 60fps, the latency is significantly lower with only about 16ms. [18.]

3.6 Video decoding

When the video stream arrives to its destination through the network it needs to be reordered, converted and uncompressed to be able to view it. In addition, depending on the network conditions, the decoder might apply a buffer. Buffering is temporarily storing data and it can be considered as one of the top contributors to end-to-end latency in a video streaming system. This ensures that the data packets applied by the transport protocols arrive in order and that the decoder has enough data to continue showing the stream with an even frame rate even with network congestion. [18.]

The size of the buffer is determined by the bitrate of the video stream. Depending on the bandwidth available for the video stream it might be necessary to utilize bitrate regulation during transmission. It is highly common for the bitrate to not remain constant even if it has been set to be so in the encoding process which is why the buffer is required. In a compressed video stream, the bitrate is often natively a variable bitrate (VBR). Video sequences that contain more movement than others require more bits to transmit them with the same quality. In contrast, sequences that have only a moderate amount of motion or not at all require less bits. If bitrate regulation is used, the compression is forced to constantly produce the same number of bits. This is called constant bitrate (CBR) and it's defined by the averaging period. Since the bitrate is not constant, it fluctuates. [25.]

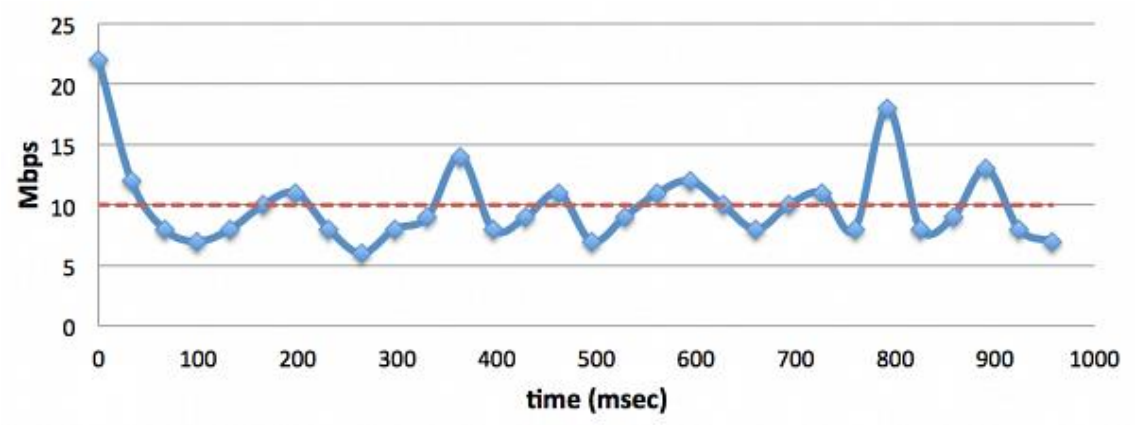

Image 10. A graph of a video stream with a CBR of 10 Mb/s and an averaging period of 10 frames [25].

For the decoder to keep showing data on a steady rate during playback, the buffer needs to store the amount of data corresponding to an averaging period, meaning that the averaging period determines the latency of the buffer [25].

4 Latency measurement methods

4.1 Two display method

Several methods can be applied to determine the amount of end-to-end latency in a live streaming system. One of the easiest and simplest methods is to use a stopwatch to calculate the difference between the stopwatch captured by a camera and the stopwatch shown on a display. The camera is pointed towards the stopwatch and the video is projected to a display. Both timers are shown side by side to allow sampling their values simultaneously. After the values are captured the value of the delayed stopwatch is subtracted from the original stopwatch to determine the end-to-end latency. In addition to a stopwatch it is also reliable to use a counter that increases by 1 every millisecond as shown in the figure below. [26.]

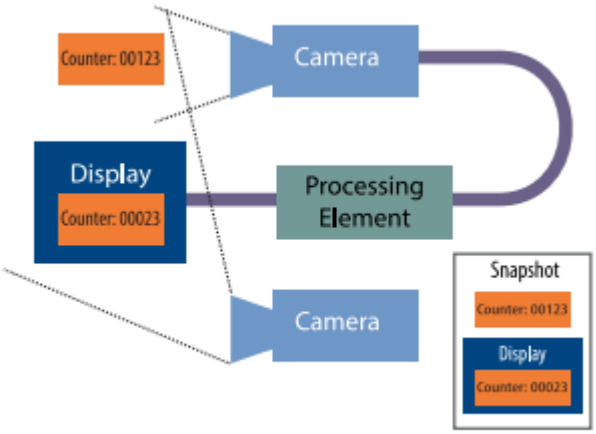

Figure 2. Setup for calculating latency based on counter values shown on two displays [26].

To test the integrity of this method, a practical experiment was carried out. Even though clearly visible video latency is mostly observed in live streaming systems it is possible for it to be noticeable in systems that have a camera outputting a video signal directly to a display without any additional processing. In the test environment a GoPro Hero 4 action camera was used to capture a running internet browser stopwatch on a computer display. The video signal was then output in real time with an HDMI connection cable to another computer display.

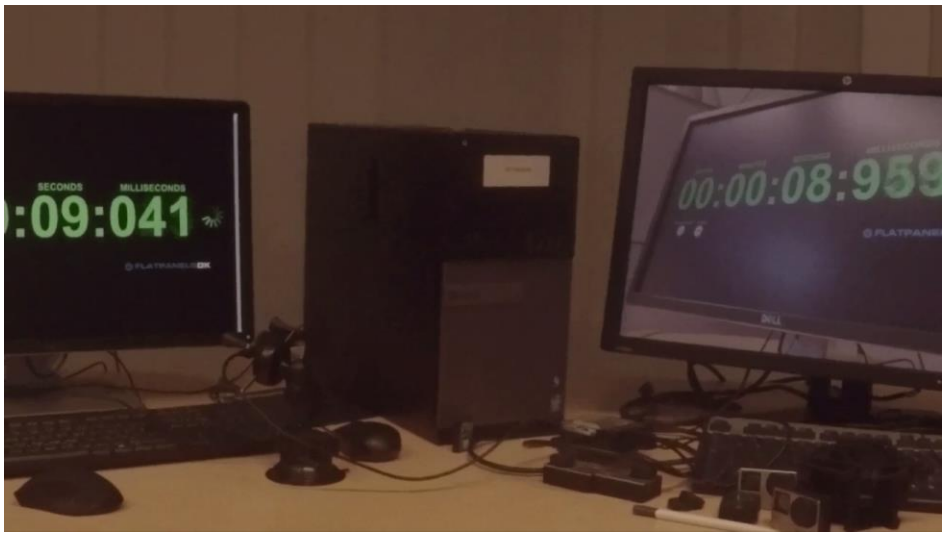

Image 11. A snapshot of 120 fps video taken during the test to determine the values of the timers.

To determine the accuracy of the timer and to get multiple samples of the latency another GoPro Hero 4 camera was used to record both timers on a video with 120 fps. The video was the reviewed to get several values of the timers and calculate the differences between them. The average of the differences was calculated to determine the overall latency of the system.

Three tests were carried out with each test using a different FPS (frames-per-second) value in the output of the camera that was capturing the original timer. This was to determine how much using different FPS values affected latency in a simple system. The results of the tests can be seen in table 1.

Resolution and fps	Latency(ms)
1080p24	115
1080p30	95
1080p60	83

Table 1. The results of the latency tests of the GoPro Hero 4 camera.

Latency in this system did not scale down with the fps values in the same way as is assumed. In section 3.5 of this report the latency of capturing with 30 fps was mentioned

to be about 33ms and 60 fps to be about 0.016ms, the difference between the two being about 32,98ms. However, the difference between the 30 fps and 60 fps latencies from the experiment was only 12ms.

As can be determined from the values received from the test, the overall latency of the HDMI connection alone was quite high. Given that the GoPro Hero 4 camera does not utilize content encryption in its HDMI connections [27] it could be assumed that the processing of the video signal inside the camera is slower because of its hardware. Since action cameras are not commonly used in live streaming applications their real-time output capabilities are not assumed to be of high quality.

Based on the tests carried out with the two-display method, it can be assumed to hold some accuracy in measuring video latency. However, it should be noted that some stopwatches or timers do not refresh their displayed values in regular intervals. For example, the stopwatch used in the above tests had irregular periods of 3-4 frames where the time was not refreshed. In a 120fps video this means that in a period of 3,6-4,8ms the value of the timer did not change. Considering that the timer itself did not stop but only failed to refresh the value it did not seem to negatively affect the measuring. This was because utilizing video capture allowed the gathering of several samples of timer values from different parts of the measuring sequence. Comparing these samples confirmed the functionality of the timer and consistency of the latency.

4.2 Synchronized timers

In a situation where latency needs to be measured on a remote system where both endpoints have a considerable amount of distance between each other, using the measuring method mentioned in section 3.1 to accurately determine latency is practically impossible. This is simply because in a remote operating scenario the two displays cannot be viewed side by side so comparing their values is not possible. To this end, an additional timer or a stopwatch can be utilized to help with the measuring. To ensure that the measurement is accurate, both clocks should at the very least use the same hardware to function. This is because using two clocks with different system specifications can cause them to not be fully synchronized because of possible differences in frequency.

The idea of using two clocks in a remote testing environment is to start them both at the exact same time, either manually or with some form of automation. The latter option would have the factor of human error not present in the synchronization of the clocks provided that the automation method is precise. After the clocks have been synchronized one of them is placed in front of the camera that is on the other end of the streaming system. To measure the latency the second clock will be placed next to the display that is viewing the incoming video stream. A snapshot of the two clocks is taken to make calculating the values easier and the difference in values of the two clocks is calculated to determine the end-to-end latency. This method was applied to the latency measurement of the remote operation system.

4.3 Optical character recognition

It is possible to make measuring end-to-end latency faster by applying pattern recognition technology. In a testing environment where latency needs to be measured with multiple different components and with several combinations of the streaming system's settings, automation of the latency calculation can make testing significantly easier because of the shortened measuring time.

Optical character recognition (OCR) is a type of pattern recognition technology that is used to convert images of text into electronic form. This includes text that is either typed, handwritten or printed, and it is not restricted to what type of image OCR is targeted by. In some applications, OCR does not require an image to be captured to scan it but is able to recognize text patterns in real-time and show the desired result immediately. An example of this is Google's Translate mobile application which enables the usage of a mobile device's camera to directly translate text to another language in real-time (image 12).

Image 12. Google Translate application's real-time OCR in use [28].

Real-time OCR technology could be applied for almost full automation of measuring latency. The application would only need to recognize the numbers from the timer that the camera captures and the timer that and separate them to two variables and then calculate the difference between those variables. The result would then be the current end-to-end latency. However, accuracy of utilizing OCR in latency measurement can be argued as being inconsistent in some scenarios. In the case of timers that show times with 1ms accuracy, it is possible for them to display overlapping values when captured as can be seen from the image below.

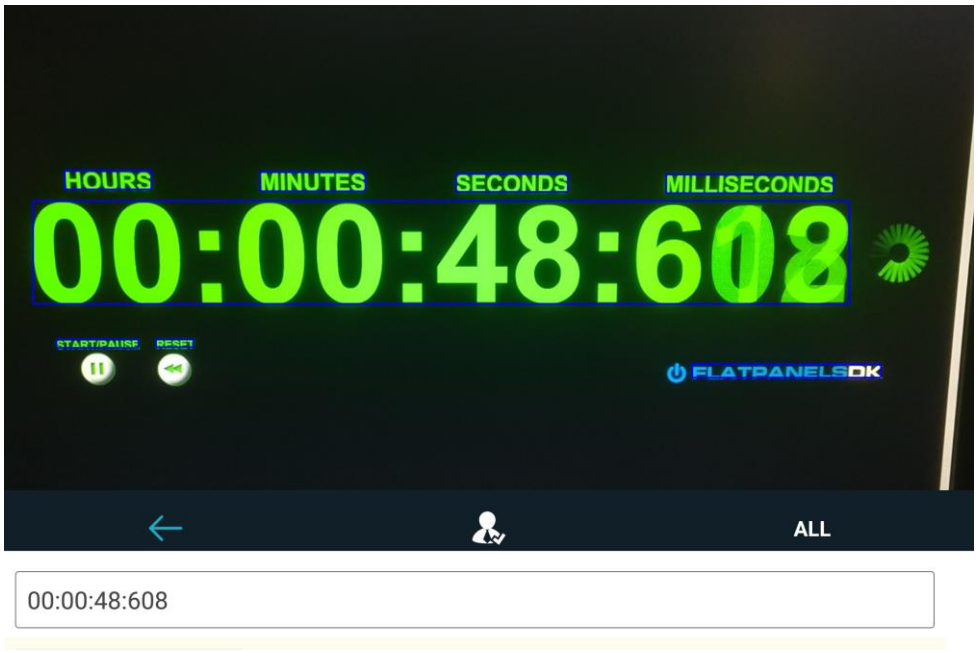

Image 13. Snapshot of the OCR mobile application Smart Lens capturing a timer. The below number is the value of the timer that the application recognized.

The reason for timer values to display overlapping is because of the refresh rate of the display. The display used in the above image had a refresh rate of 144Hz, meaning that the display updates every 144th of a second which is about 7ms. This causes the timer to only show values every 7ms or in this case two values showing at the moment of change. At high refresh rates, this should not be an issue when considering the accuracy of latency measurement but on lower refresh rates the error margin can increase. In the case of computer displays, the refresh rate of 60hz is more likely to be observed considering the higher cost of displays with refresh rates of 120Hz or higher. With 60Hz the display can only show timer values every 16-17ms which can result in small inaccuracy in latency calculations as it is unknown which of the overlapping values represents true time. For example: A timer capture shows two overlapping millisecond values of “008” and “024”. The OCR software recognizes the value as “004”, meaning that it is showing the result with an error margin of 20ms.

Overall, utilizing OCR technology to automate video latency measurement can be considered recommendable. In situations where multiple tests must be conducted in a relatively short time it can be used to shorten the time needed to obtain the measurement

results, which would normally include capturing the timers to an image manually, determining the values and calculating the results. If paired with an application that would automatically calculate those results from OCR's values the measuring sequence could be shortened even further.

5 Remote operation testing

The testing environment used for researching user experience in remote operation consisted of a battery powered RC vehicle and a control center with multiple screens. The control center was equipped with a steering wheel and foot pedals for controlling the vehicle. To send the control signal to the vehicle and to stream live video from the vehicle back to the control center, four Raspberry Pi computers were utilized. Raspberry Pi computers are small, extremely versatile for programming and are equipped with a minimum amount of input and output ports. The computers come with a Linux based operating system Raspbian. One of the computers, along with a microcontroller chip, was used to receive the control signal for the vehicle. The base remote-control testing environment and streaming system was built and programmed by Metropolia lecturer Matti Peltoniemi and student Anton Dementev.

5.1 Live streaming system

The remote operating system utilized two different types of Raspberry Pi computers. As can be seen from image 14 below, the front and rear computers were of the Model 2 B variety. The computers positioned on the vehicle's sides were two smaller Zero W models. The main reason for using two different Raspberry Pi models was to minimize weight and the use of space on the vehicle. Each of the Raspberry Pis was equipped with a camera module for capturing video. In addition, the rear camera utilized a wide-angle lens for increased coverage. A special casing was used to attach the lens on top of the camera. The front computer used a microcontroller that required USB connection which the Zero W model does not provide. The two side computers did not require any additional modifications aside from the camera module, so they were chosen to use the

smaller Raspberry Pi model. Power to two of the computers was distributed from the RC car batteries and the other two used a USB power bank.

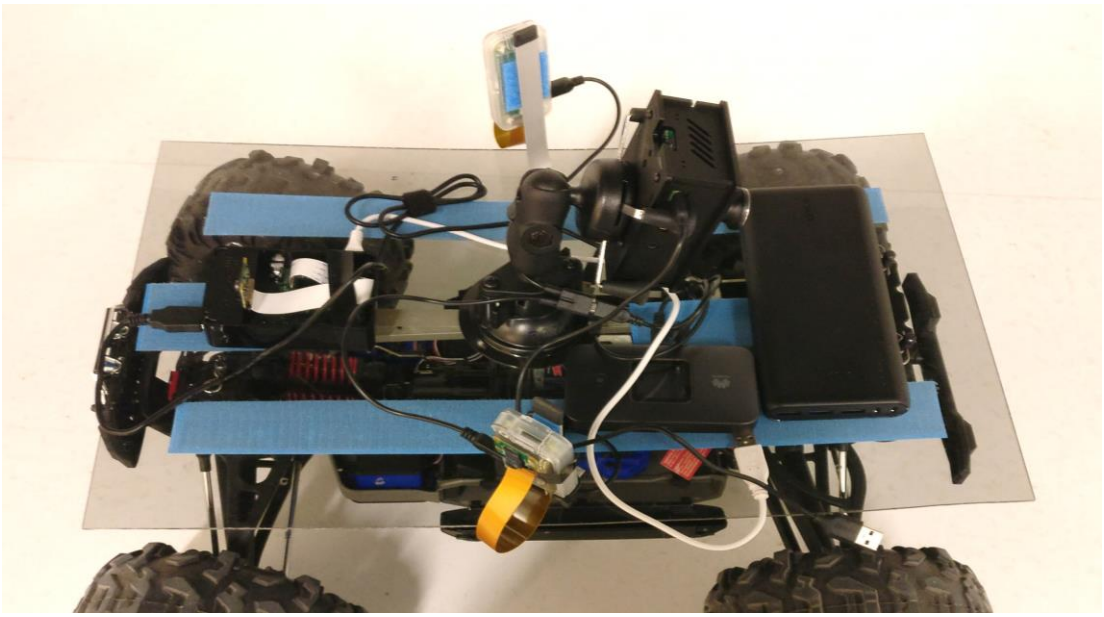

Image 14. Distribution of the Raspberry Pi computers on the RC vehicle.

To stream the video feed from the cameras to the control center, the testing environment utilized GStreamer which is an open-source framework for creating multimedia applications. GStreamer is lightweight and utilizes plugins for constructing arbitrary pipelines and it can be used for both sending and receiving video streams. Initially, GStreamer requires a video source element and a sink. Between these two, elements that are used to process the source video before it reaches the sink can be applied. After the video is processed, the sink outputs the stream. [29.] Various elements can be applied with the help of available plugins for GStreamer. For example, x264enc is an element that allows for H.264 encoding of raw video. To send the encoded video through the network with RTP, a payload element is used in the same pipeline. The element embeds H.264 encoded video data into the RTP packets' payload for transport. The streaming pipeline used for sending the streams in the remote operation system can be seen in the figure below.

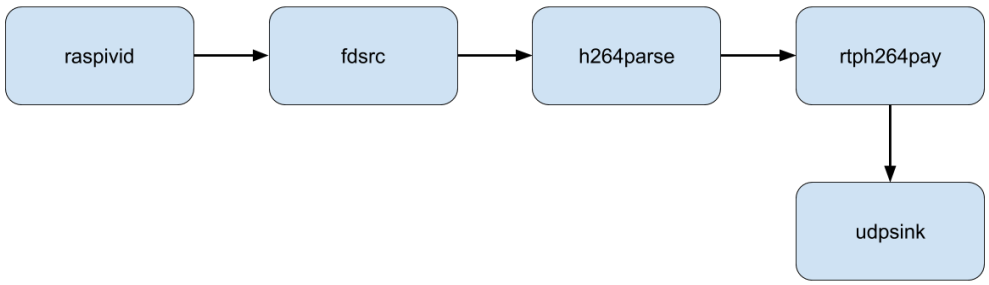

Figure 3. The GStreamer pipeline for sending the streams.

The primary functions of each element in the above pipeline are as follows:

- **raspivid**: The command line tool for capturing video with the camera module
- **fdsrc**: Reads data from the initial input in the pipeline (raspivid in this case)
- **h264parse**: Analyzes the incoming H.264 format and signal the information forward
- **rtph264pay**: Embeds H.264 encoded video to RTP formatted data packets as a payload
- **udpsink**: Sends the stream forward as UDP packets to the specified host address and/or port.

The resolution and framerate for most of the camera modules was set to 1640x1232 and 25 fps. This resolution allowed a maximal field of view that the cameras were able to put out. As an exception, the rear camera's height resolution was set lower than the others to simulate the look of a rear-view mirror for user experience testing. The source element (fdsrc) was used to pipeline the camera module's (raspivid) input and the sink (udpsink) was used to send UDP packets to the network. Since the Raspberry Pi camera module captures and processes video in H.264 format by itself, no elements for encoding raw video was required. Decoding and viewing the stream in the control center was achieved with elements designed for decoding and buffering video data. A buffer was also applied to keep the video stream stable in the case of network disturbances such as interrupted

connections. Using a buffer ensured the integrity of the video stream. The pipeline used for receiving the stream is illustrated in the figure below.

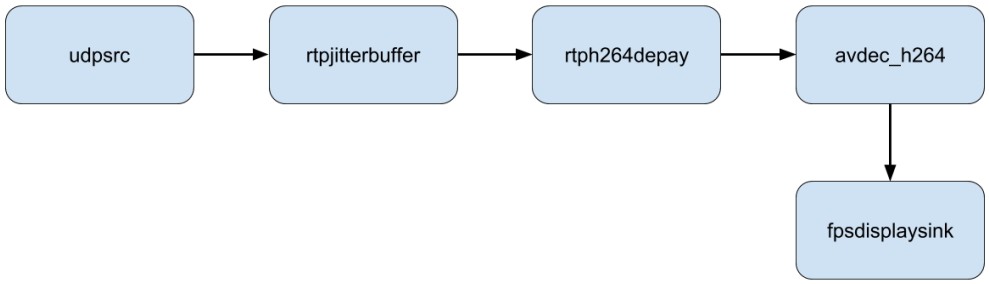

Figure 4. The GStreamer pipeline for receiving the streams.

The primary functions of each element in the receiver pipeline are as follows:

- **udpsrc**: Reads UDP packets from the network.
- **rtpjitterbuffer**: Reorders and removes duplicate RTP packets.
- **rtph264depay**: Extracts H.264 encoded video data from RTP packets.
- **avdec_h264**: Decodes H.264 video.
- **fpsdisplaysink**: Displays the decoded video with additional information such as framerate and the total amount of rendered frames.

The receiver pipeline of the streaming system utilized a buffer element (rtpjitterbuffer) to help maintain stable streams. The element reorders RTP packets as they arrived and removes any duplicate packets if they occur. The buffer waits for missing packets in the sequence for a set amount of time. If they arrive late or not at all, they will be considered lost and the packets stored in the buffer are transferred to the pipeline's next element. [30.]

Since the stream needed to be sent out wirelessly, mobile networking was used. To achieve this, a Huawei 4G/LTE modem equipped with a SIM card was used to connect

the Raspberry Pi computers to the internet. The computers used Wi-Fi technology to connect to the modem. The final workflow of the streams is illustrated in the figure below.

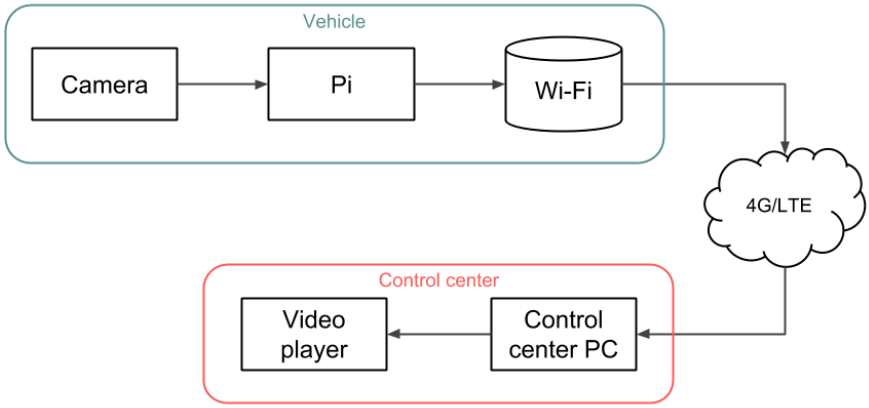

Figure 5. Illustration of the flow of the live video feed from the camera modules to the control center.

Initially, this streaming setup was used throughout the testing process with little changes applied to it during the project. The main goal for researching the latency of this setup was to determine the integrity of wireless networks when streaming from multiple sources simultaneously. In addition, behavior of the video streams while controlling the vehicle was observed with the intention to determine whether it had any effect on the end-to-end video latency.

5.2 Performance and latency

During the first weeks of testing, the live streaming system proved to be mostly stable. The RC vehicle was situated in an open indoor area with a suitable mobile network coverage. No significant loss of stream quality was observed when the vehicle was staying still or when it was being controlled. However, irregular intervals of packet loss were observed when the driving area was occupied by more than two people during the vehicle's movement. This was assumed to occur because of the nature of mobile networks. When more devices that use the same 4G network enter the area, there is a possibility of that network to become congested. This can cause a decrease in the network's performance and packet drops. This was never confirmed, however. It is unknown whether

the connection issues occurred because of more physical objects larger than the vehicle in the room which could have made maintaining a stable connection more difficult.

In the case of video latency, an average of 250ms delay was observed in the system when all four streams and the control signal were active. To determine if having multiple streams active and transmitting to the same destination affected it, a test with a single Raspberry Pi active only was conducted. The results indicated no significant change to the latency with the amount of delay remaining as 250ms, meaning that the network was deemed to be efficient enough to not experience loss of quality with four simultaneous streams.

During the later stages of the project, a significant change in video latency was observed. Without making any notable changes to the remote operation system, the latency had suddenly increased to 1 second. Because of this, additional latency testing was conducted to try to determine the possible cause. It was theorized that the reason for the latency increase could have been a modification made to the remote-control user interface since it was the only change made before the increase was observed. For this reason, the Raspberry Pi that was used for receiving the control signal was first disabled during the measuring. With only three video streams sending data over the network, the latency was measured to be 250ms. When the fourth Pi was brought back online, and the vehicle controls were enabled, the latency increased back to 1s. The initial cause for the increase remained to be unknown because no changes to the controls themselves were made during the project.

After several weeks, new latency measurements were conducted to attempt to research the increased delay again. During these tests it was observed, however, that the 1s video latency with the vehicle controls enabled had reverted to 250ms. No additional changes had been made to the system prior to this point which made determining the cause of the increased latency significantly more difficult. The only known variable factors, with the nature of wireless networks being slightly unpredictable in terms of liability, were the Wi-Fi connection and the 4G network. With the 4G network in the same state as it was during its congestion tests it was not deemed necessary to be tested again. The Wi-Fi connection’s behavior, however, needed to be assessed.

Since Wi-Fi is a wireless technology and with several devices can be connected to one Wi-Fi network simultaneously, there is a possibility for it to become congested as well. This was tested with a ping to the Wi-Fi default gateway during various scenarios. The results of this test are described in the table below.

Scenario	Latency (ms)
No active streams or control signal	1-2
All four streams active, no control signal	1-4
All four streams and the control signal active	2-7

Table 2. The results of the Wi-Fi latency test with different scenarios.

The results of this latency test indicate that the delay caused by the Wi-Fi connection in the remote operation system should not be considered significant as latencies of this magnitude contribute to an end-to-end latency of 250ms by less than 3%. Because of this, the reason for the 1s latency increase could not have been caused by the Wi-Fi connection, leading to believe that the main cause of the sudden spike in latency was the 4G connection.

5.3 Latency distribution

To calculate each streaming system component's percentile contribution to the overall latency, the 4G connection's own latency had to be assessed. A test was conducted to determine the 4G network's latency contribution. This was achieved by connecting one of the Raspberry Pi computers to the LAN (Local Area Network) of the control center and use that to send the stream to the receiver. Network latency in a local area network in most cases is less than 5ms so its contribution to the end-to-end latency could be considered near-nonexistent. The end-to-end latency measured in this test was 110ms. When subtracted from the end-to-end latency that utilized 4G networking (250ms), the approximate contribution of the mobile network to the latency could be determined to be 140ms. Now the latency contribution of the streaming hardware (video capture, encoding

and decoding) could be calculated by subtracting both the Wi-Fi connection latency and 4G connection latency from the end-to-end latency value.

With all three parts of the live streaming system now measured individually and having a regular value to the end-to-end latency, it was possible to determine each part's percentile contribution to the overall latency. The contribution percentage is illustrated in the figure below.

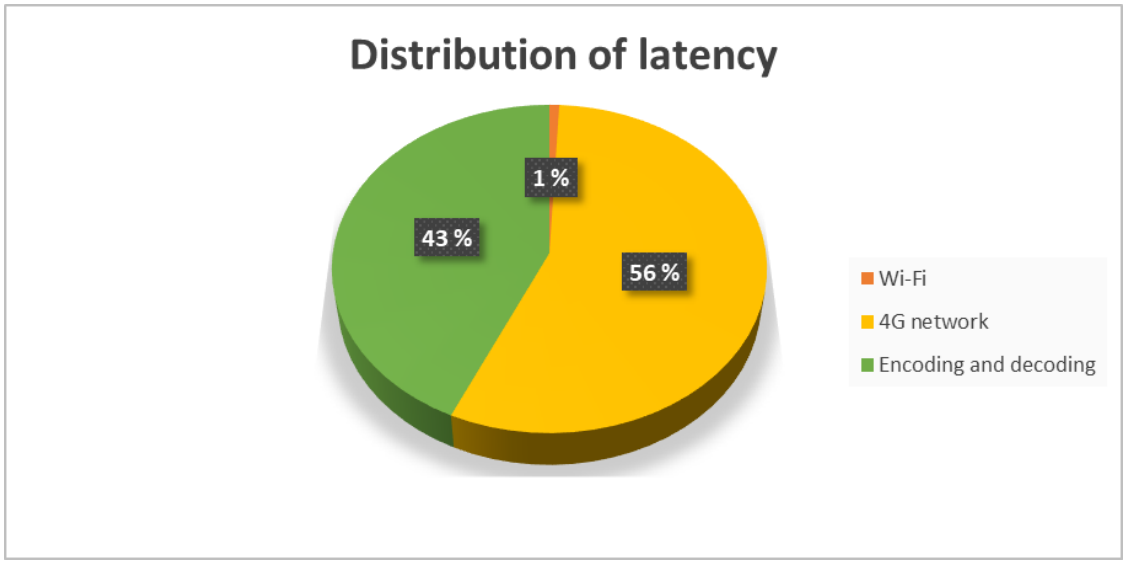

Figure 6. The distribution of latency of all factors in the live streaming system by percentage.

As can be determined from figure 6, the 4G mobile network contributed to the overall latency the most by over 50% with the encoding and decoding processes also having a significant impact on it. Because of mobile networking remaining to be unstable in some scenarios and having the largest latency contribution percentage in the remote operation system, it should not be considered a liable connection method in its current state. It should also be noted that the processing capabilities of the Raspberry Pi computers are far more limited than hardware that are specifically designed for live streaming media. This could explain the large amount of latency that the required processes caused in the system. With Raspberry Pi computers not being designed for processing high-quality media with the latest codecs such as H.265, encoding and decoding a single video frame with this hardware can cause latencies that could be deemed unsafe when considering remote operation of vehicles, as mentioned in section 2 of this report.

5.4 Additional testing

To determine any additional contributors to the end-to-end latency regarding hardware performance, several measurement tests with different video resolutions and framerates were conducted. To make the testing more efficient and to have the streaming system be more accessible for making changes, the testing environment was moved from the control center to a single room. As stated in section 5.2 the control signal for the RC vehicle did not have any significant effect on the end-to-end latency so its presence was deemed unnecessary for these tests. It is also important to note that a significant change in the end-to-end latency was observed when switching testing spaces. As mentioned in an earlier section of this report the end-to-end latency in the control center was 250ms. When measured in the new testing environment, however, it had been reduced to an average of 200ms. The only possible explanation to this was the behavior of the wireless 4G network, further enforcing the fact that this type of connection should be considered unpredictable in a real-life scenario.

The following table shows the different resolution and framerate settings that were tested with GStreamer and the resulting latency and performance changes. The method used for measuring latency was done with two synchronized timers. The difference in latency was deduced from comparing the new results to the original latency with the resolution of 1640x1232 and a framerate of 25 fps.

Settings	Difference in latency and/or performance
Base resolution (1640x1232), framerate 50	-30ms
Base resolution (1640x1232), framerate 60	+50ms, unstable framerate
Resolution 1280x720, framerate 60	-60ms
Resolution 640x480, framerate 60	-100ms
Resolution 640x480, framerate 25	-30ms
Resolution 640x480, framerate 15	+30ms, inaccurate measuring
Resolution 1640x1232, framerate 15	+30ms, inaccurate measuring

Table 3. Latency measurement results of different resolution and framerate settings with GStreamer.

The results of the tests gave a significant amount of insight on the streaming hardware's behavior with different configurations. The most notable observations that was made from the results is the significance of framerate. The increase and decrease of framerate can affect two things: video capture latency and processing time. In the case of utilizing Raspberry Pi computers as processing units, hardware limitations regarding encoding and decoding are more apparent because of their low processing power. The more frames are captured the more processing power is needed when dealing with higher resolutions. This was apparent in the second test where the framerate was increased to 60fps. With the minicomputer not being able to process all the frames accordingly the latency was increased by 50ms and the framerate became noticeably unstable. However, in the case of processing power not being limited and allowing increased framerates it can be beneficial for the latency. As stated in section 3.5 of this report, the latency caused by video capture depends on the framerate. The original framerate used for the tests' comparison was 25 fps, meaning that the capture latency was approximately 40ms. As can be seen from table 3 the first test had the framerate at 50 fps and the latency was improved by 30ms. The capture latency in this setting was calculated to be only 20ms. In comparison, using lower framerates can also result in an increase in latency. In the two final tests, a framerate of 15fps was used with two separate resolution values and both resulted in the same amount of latency increase of 30ms. In a scenario where the end-to-end latency needs to be lowered to a minimum, utilizing hardware that is capable of streaming with framerates of 60fps or more with stability can be crucial.

6 Conclusion

The purpose of this thesis was to determine the different factors of live video streaming latency in a remote operating environment. The research that was conducted regarding different live streaming elements helped understand the process of sending and receiving streams which in turn made it significantly easier to prioritize different aspects of the streaming system that contributed to the end-to-end latency more than others.

It is important to note that while it is possible to lower the latency to a value that makes it more difficult to notice with the human eye, with current technology it is simply not possible to lower it to a level where it is practically nonexistent. Also, live streaming applications where maintaining high video quality to the viewer is a priority could be even more difficult, since keeping the quality from the moment of frame capture to the viewer's screen means more data for the network to transport. Live streaming system designers that want to maintain a low latency need to prioritize the different aspects of the stream depending on what the stream is used for. For example, in a situation where the video quality needs to be high and the stream relatively stable the encoding and decoding process must be fast, and the network must have enough capacity to transfer large amounts of data simultaneously. The larger the resolution and framerate values the video has the more data the encoder and decoder need to process, and the more data the network needs to handle.

As can be seen from this report, the largest contributor to the end-to-end latency in the remote operating scenario is mobile networking. When considering applying remote operation to a large-scale fleet of public transportation the scalability of modern mobile networks becomes an issue. While encoding and decoding latencies can be reduced by utilizing more versatile hardware, with every vehicle in the fleet utilizing several cameras and with every camera requiring its own stream the requirement of network bandwidth will be colossal. Considering that the remote operators would be required to have constant visibility on the vehicles, some form of a prioritization system would need to be applied to the mobile network or have a network dedicated entirely to the video streams and the vehicle controls. With 4G networks being too unreliable for such tasks it is also unknown if 5G networking that is currently under development will be able to answer to these demands. For this reason, further research of mobile networking as a latency

contributor should be considered a priority. It is also important to consider the current speed of live streaming technology development. For example, it is safe to assume that the HEVC codec will eventually replace H.264 which is currently a global standard for encoding and compression. Given that H.264's first version was introduced in 2003 and has gone through multiple revisions since then, it is obvious that a codec's "lifespan" can last for more than a decade. And since HEVC is still considered new it must wait for hardware technologies to become versatile enough to answer to its demands regarding processing hardware. Implementing HEVC into equipment that have used H.264 until now would mean that the standard would most likely have to work with smaller resources, limiting its current capabilities. In addition, HEVC is not yet fully compatible with other live streaming protocols. As an example, RTP does not have a supported payload format to transfer HEVC encoded data over a network yet. This is simply because the format is still in development with the latest RFC document having been updated in May 2016. It is also safe to assume that HEVC will have additional versions adding more profiles and levels to support different scenarios with the same variety as H.264.

References

1 Vienna Convention on Road Traffic. 1986. 30/01.04.1986.

2 Lumiaho, Aki & Malin, Fanny. 2016. Tieliikenteen automatisoinnin etenemissuunnitelma ja toimenpideohjelma 2016–2020. <https://julkaisut.liikennevirasto.fi/pdf8/lts_2016-19_tieliikenteen_automatisoinnin_web.pdf>. Accessed 20 January 2017.

3 Streaming media. Wikipedia. <https://en.wikipedia.org/wiki/Streaming_media>. Accessed 10 January 2018.

4 Anthony, Paul & Richardson, Iain. 2010. The H. 264 Advanced Video Compression Standard. John Wiley & Sons, Incorporated. Ebook Central. <https://ebookcentral.proquest.com/lib/metropolia-ebooks/detail.action?docID=540113>. Accessed 10 January 2018.

5 An Overview of H.264 Advanced Video Coding. Vcodex Ltd. <https://www.vcodex.com/an-overview-of-h264-advanced-video-coding/>. Accessed 10 January 2018.

6 Wiegand, Thomas. H.264/AVC Video Coding Standard. <http://iphome.hhi.de/wiegand/assets/pdfs/DIC_H264_07.pdf>. Accessed 10 January 2018.

7 Huang, Stanley. 2008. H.264 profiles and levels. Mediatronic Ltd. <http://blog.mediacoderhq.com/h264-profiles-and-levels/>. Accessed 10 January 2018.

8 Wiegand, Thomas. Scalable Video Coding in H.264/AVC. <http://iphome.hhi.de/wiegand/assets/pdfs/DIC_SVC_07.pdf>. Accessed 10 January 2018.

9 MGW Ace Encoder. VITEC. <http://vitec.com/fileadmin/downloads/PRODUCTS/Encoders/Portable/ACE/MGW_Ace_Datasheet_Web_RevT.pdf>. Accessed 10 January 2018.

10 2017. 4 Reasons Why HEVC (H.265) Matters, and How You Can Start Using It Now. Teradek. https://teradek.com/blogs/articles/3-reasons-why-hevc-x265-matters-and-how-you-can-start-using-it-now>. Accessed 10 January 2018.

11 Ohm, Jens-Rainer; Schwartz, Heiko; Sullivan, Gary J.; Tan, Thiow Keng & Wiegand, Thomas. 2012. Comparison of the Coding Efficiency of Video Coding Standards—Including High Efficiency Video Coding (HEVC). IEEE Xplore.

<http://ieeexplore.ieee.org/abstract/document/6317156/>. Accessed 10 January 2018.

12 Technical guide to network video. Axis. <https://www.axis.com/fi/en/learning/web-articles/technical-guide-to-network-video/compression-formats>. Accessed 10 January 2018.

13 Transport Layer. Wikiversity. Last update 19 November 2016. <https://en.wikiversity.org/wiki/Transport_Layer>. Accessed 10 January 2018.

14 Blank, Andrew. 2006. TCP/IP JumpStart: Internet Protocol Basics. Sybex, Incorporated. Ebook Central. <https://ebookcentral.proquest.com/lib/metropolia-ebooks/detail.action?docID=3056693>. Accessed 10 January 2018.

15 Haden, Rhys. TCP. <http://www.rhyshaden.com/tcp.htm>. Accessed 22 January 2018.

16 DCN - User Datagram Protocol. Tutorialspoint. <https://www.tutorialspoint.com/data_communication_computer_network/user_datagram_protocol.htm>. Accessed 22 January 2018.

17 Koistinen, Tommi. Protocol overview: RTP and RTCP. <https://www.netlab.tkk.fi/opetus/s38130/k99/presentations/4.pdf>. Accessed 22 January 2018.

18 Latency in live network video surveillance. Axis. <https://www.axis.com/files/whitepaper/wp_latency_live_netvid_63380_external_en_1504_lo.pdf>.

19 RTP Packet Format. <https://www.cl.cam.ac.uk/~jac22/books/mm/book/node159.html>.

20 Real-Time Messaging Protocol. Wikipedia. <https://en.wikipedia.org/wiki/Real-Time_Messaging_Protocol>. Accessed 10 January 2018.

21 Real-Time Media Flow Protocol. Wikipedia. <https://en.wikipedia.org/wiki/Real-Time_Media_Flow_Protocol>. Accessed 10 January 2018.

22 Parmar, Hardeep & Thornburgh, Michael. 2012. Adobe's Real Time Messaging Protocol. Adobe Systems Incorporated. <http://wwwimages.adobe.com/www.adobe.com/content/dam/acom/en/devnet/rtmp/pdf/rtmp_specification_1.0.pdf>. Accessed 22 January 2018.

23 Ozer, Jan. 2011. What is HLS (HTTP Live Streaming). Streaming Media Magazine. <http://www.streamingmedia.com/Articles/Editorial/What-Is-.../What-is-HLS-(HTTP-Live-Streaming)-78221.aspx>. Accessed. 22 January 2018.

24 MPEG transport stream. Wikipedia. <https://en.wikipedia.org/wiki/MPEG_transport_stream>. Accessed 22 January 2018.

25 Understanding - and Reducing - Latency in Video Compression Systems. Design & Reuse. CAST, Inc. <https://www.design-reuse.com/articles/33005/understanding-latency-in-video-compression-systems.html>. Accessed 22 January 2018.

26 Eberlein, Peter. 2015. Video Latency - What It Is and Why It's Important. Sensoray. <http://www.sensoray.com/downloads/whitepaper_latency.pdf>.

27 Is HDCP Enabled in HERO4? GoPro. <https://gopro.com/help/articles/question_answer/Is-HDCP-Enabled-in-HERO4>. Accessed 22 January 2018.

28 How to use the Google Translate app in the classroom. 2016. Daily Genius. <https://dailygenius.com/use-google-translate-app-classroom/>. Accessed 5 April 2018.

29 Laine, Joona. 2016. Low latency high-definition video streaming for real-time teleoperation platform. Master of Science thesis. Tampere University of Technology. <https://dspace.cc.tut.fi/dpub/bitstream/handle/123456789/24843/Laine.pdf>. Accessed 5 April 2018.

30 Rtpjitterbuffer: GStreamer Good Plugins 1.0 Plugins Reference Manual. GStreamer. <https://gstreamer.freedesktop.org/data/doc/gstreamer/head/gst-plugins-good/html/gst-plugins-good-plugins-rtpjitterbuffer.html>.