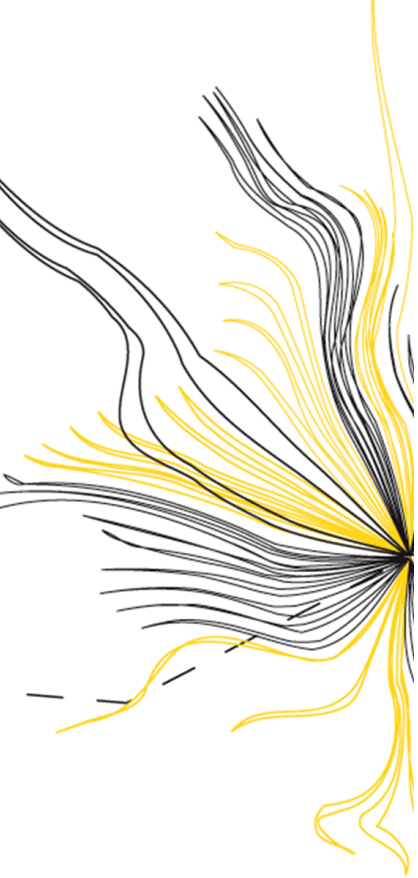


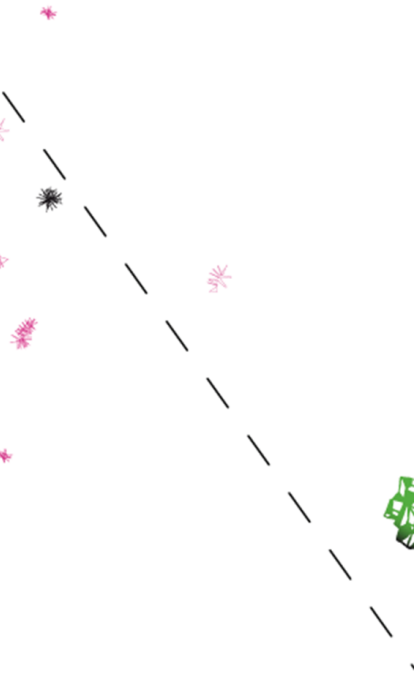
UNIVERSITY OF TWENTE.

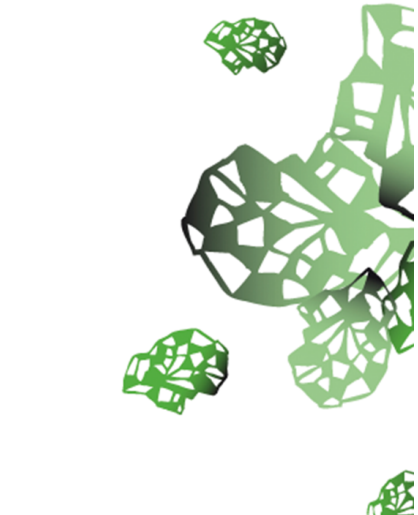
Faculty of Electrical Engineering,
Mathematics & Computer Science

Efficient and Reliable Delivery of VR over 5G Mobile Networks

Hasim Timurcan Evci

MSc. Thesis
January 2020





Examination Committee:
prof. dr. J.L. van den Berg
prof. dr. ir. G.J. Heijenk
dr.ir. A.B.J. Kokkeler

DACS Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Summary

With the introduction of 5G, wireless VR has found a place on the agenda of the VR industry. High data rates and low latency offered by 5G technologies such as mmWave and enhanced mobile edge computing (MEC) provides the opportunity to make a switch from wired to wireless VR. This research focuses on the efficient and reliable delivery of non-interactive VR over 5G mobile networks. The trade-off between efficiency and reliability is the main concern of this research.

The FoV transmission scheme makes it possible to deliver VR over mobile networks but requires FoV prediction. Channel prediction can also be a useful tool, as it can provide information about when to transmit the FoVs. These two prediction algorithms can be computed in the MEC, so the latency will be minimal. The effect of the MEC and these two predictions on the trade-off between efficiency and reliability is investigated in this thesis. Three main delivery strategies, which doesn't include channel prediction, were designed and tested in a Python simulation environment that was built from scratch, namely the reactive, proactive and combined strategies. Four different simulation scenarios were used to test these strategies, namely different users, different video datasets, different channel qualities and different probability thresholds. Later on the research, two enhanced delivery strategies with channel prediction were created to see the effect on the trade-off.

From the experiments, it is concluded that the combined strategy with a low threshold was the best in terms of the trade-off in reliability and efficiency. One of the enhanced delivery strategies with channel prediction resulted in better reliability with little cost in efficiency. To sum up, it is convincing that the MEC and the algorithms computed in the MEC, FoV prediction and channel prediction, helps us to find the optimal trade-off between reliability and efficiency.

Acknowledgement

First of all, I wish to express my sincere appreciation to my main supervisor, prof. dr. J.L. van den Berg, for his guidance and support throughout the duration of this research. From the moment I entered his room to ask for a thesis topic, he has been nothing but kind and helpful.

Secondly, I would like to thank my other supervisors, prof. dr. ir. G.J. Heijenk and dr.ir. A.B.J. Kokkeler, for their time and patience and, am gratefully indebted to them for their very valuable feedback on this thesis.

Finally, I must express my profound gratitude to my family and friends for providing me with endless support and encouragement throughout my long years of studying. Together we have survived difficult times and cherished good memories. This thesis is dedicated to them.

Hasim Timurcan Evci, Enschede 2020.

Contents

List of Figures

List of Acronyms

VR	Virtual Reality
QoS	Quality of Service
QoE	Quality of Experience
MIMO	Multiple-Input and Multiple-Output
MEC	Mobile Edge Computing
HMD	Head Mounted Device
VOD	Video On Demand
FoV	Field of View
MTP	Motion-to-Photon
CSI	Channel State Information
LOS	Line of Sight
RAN	Radio Access Network
CQI	Channel Quality Indicator
DTMC	Discrete-Time Markov Chain

Chapter 1

Introduction

Wireless communications is one of the most rapidly growing areas within the information technologies. As technology evolves, mobile users of today require faster and more reliable connections. As more users come to use mobile applications, current 4G networks are almost at the limit of what they can provide for users. The capacity, bandwidth and latency requirements for novelty applications are not fulfilled with the 4G networks. The fifth generation of mobile networks, 5G, is going to overcome these limits by offering higher capacity due to the introduction of the unused mmWave spectrum, and consequently high datarate and low latency. This opens up lots of opportunities for a variety of applications that require reliable and low latency connections. This is especially important in the upcoming fourth industrial revolution, Industry 4.0. 5G is expected to be the driver of cyber-physical systems, IoT, wireless Virtual Reality (VR) and many more industries. Several other applications can be seen in Fig. 1.1.

Wireless VR, due to its low latency and high bandwidth requirement to prevent motion sickness, is not possible with the current generation 4G mobile networks or WiFi networks. With the improvements in mobile edge computing (MEC) and the high bandwidth of mmWave, the problems ahead of wireless VR are said to be solved. However, there are some specific challenges around the mmWave technology and the computation latency in the MEC. This research focuses on the efficient and reliable delivery of non-interactive and real-time VR over 5G mobile networks. In this chapter, a problem description will be provided, which is followed by the research questions.

1.1 Problem Description

With the introduction of 5G, VR is expected to be evolved into a wireless technology, due to the low latency and high data rates it is expected to deliver. In literature, MEC is found to be a very important enabler of wireless VR as it enables the use of prediction algorithms with lower latency, along with multi-connectivity that offers reliability in the network. Even though 5G promises technologies such as mmWave and, an improvement

Figure 1.1: 5G in Industry 4.0 [1]

in MEC, there are certain challenges around them. Minimizing the effect of blockages in mmWave and choosing the right FoV prediction algorithms for each application is crucial in wireless VR. Using channel prediction provides a respectable way to minimize the susceptibility of mmWaves, given that these predictions are accurate. Channel prediction can be used to detect low channel quality periods in the future, in order to send (predicted) FoVs in a high channel quality period. Since FoV and channel prediction are expected to bring about computation latency, it is necessary to use MEC to prevent to exceed the MTP-latency.

Reliability and efficiency are two important factors in the Quality of Experience (QoE) in wireless VR, since it is necessary to provide a seamless experience to the user, both in terms of visual quality and a longer battery life. However, reliability and efficiency are "inversely proportional", as high reliability has a cost in more battery usage, thus results in a less efficient experience. Therefore, it would be interesting to investigate the trade-off between reliability and efficiency in wireless VR.

The studies performed in the VR field are dominated by non-interactive VR, and especially by those on Video On Demand (VOD) VR [4] [8] [10] [11]. In VOD VR, saliency-based predictions can be used (as well as motion-based predictions), on contrary to the real-time VR, where only motion-based predictions can be made. For this research, it would be interesting to see the trade-off between reliability and efficiency in non-interactive VR as a whole. In the next section, the research questions will be posed according to the problem that has just been described.

1.2 Research Questions

The main research question followed by its subquestions are as follows,

1. How to deliver non-interactive VR in an efficient and reliable way over 5G radio access network?
   (a) How can the MEC improve the Quality of Service (QoS) in the delivery of non-interactive VR over 5G?
   (b) How can motion-based prediction improve efficiency and QoS in the delivery of non-interactive VR over 5G?
   (c) How can channel quality prediction help the delivery of non-interactive VR over 5G?

In the next section, the structure of this report will be provided.

1.3 Organization of the Report

In Chapter 2, some background regarding 5G and VR will be provided, in order to supply the reader with the necessary information. In Chapter 3, the models related to the research questions and some delivery strategies related to the models will be explained. In Chapter 4, the evaluation methodology, including the motivation to create the proprietary simulator and datasets, the evaluation criteria and simulation scenarios, will be discussed. In Chapter 5, the results of the simulation scenarios will be evaluated and analysed. In the final chapter, the overall discussion of the research and the conclusions with some future recommendations will be provided.

Chapter 2

Background and Related Work

In this chapter, some background information regarding the research will be provided to the reader. Since the starting points of the research were 5G mobile networks and VR, firstly, an initial introduction related to this work will be reviewed. Secondly, some specific challenges around 5G and VR will be pointed out. Finally, certain approaches in the literature to overcome these challenges will be discussed.

2.1 5G Mobile Networks

5G presents a whole new unused spectrum above 30 GHz and up to 300 GHz called the millimeter wave (mmWave) spectrum, which allows for an increased capacity and faster data rates [12]. Due to the short wavelengths of mmWave signals, 1 to 10 mm, they cannot travel through objects and can get dropped easily. Technologies such as small cells, massive Multiple-Input and Multiple-Output (MIMO) and beamforming are implemented in 5G to overcome these problems and make it a more reliable network [13]. MEC is another essential technology in 5G, that is expected to minimize latency and backhaul load by enabling cloud computing capabilities at the edge of a mobile network. Because of the close proximity of the MEC servers to the users, application-specific computations can be performed more efficiently, especially for low latency required applications such as connected cars and content delivery [2]. In Fig. 2.1, an overview of the MEC process (in this case, in the IoT domain) is provided. As it can be seen, instead of taking the longer route through the mobile core, the applications can use the shorter route to MEC to perform computations.

In 5G networks, the softwarization of the network allows for different types of wireless communications, e.g. cellular and device-to-device (D2D), and consequently different verticals to be controlled and contained under one roof. Therefore, 5G is expected to be not only the successor of the previous generation mobile networks but also is deemed as the biggest enabler of the fourth industrial revolution, *Smart Industry* [14]. 5G is also expected to play an important role in the smart mobility and, media

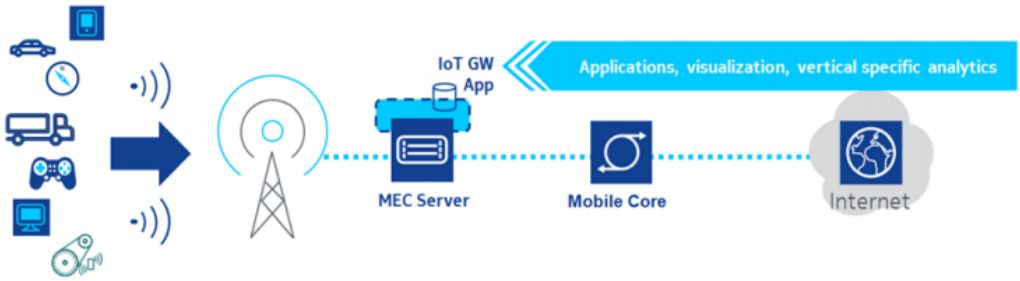


Figure 2.1: Mobile Edge Computing for IoT applications [2]

and entertainment industries, due to the low latency, ultra-reliability and fast data rates it is offering. Applications such as autonomous driving, wireless virtual reality (VR) and augmented reality (AR) become more feasible due to the aforementioned benefits of 5G.

2.2 Virtual Reality

The implementation of wireless VR and AR is among the many applications that are expected to be enabled with the introduction of 5G networks. VR allows for the possibility of experiencing a simulated environment with the aid of a Head Mounted Device (HMD). It is expected to play an important role in a lot of industries, such as the Smart Industry (remote maintenance, virtual factory), health (remote surgery), entertainment (virtual gaming, 360° video). The problem of the current VR HMDs are that most of them require long wires which prevents the movement of the user, and consequently not allowing the user to leave the same location without taking the whole setup with them. Wireless VR opens up the possibility of immersive multimedia and interactive video conferencing, for example while travelling. In the next subsections, the types of VR will be attributed and its delivery over 5G will be shortly introduced.

2.2.1 Types of VR

There are two main types of virtual reality classified upon the level of user interaction. The first type, the weak-interactive VR, provides a passive experience and doesn't grant too much interaction except for the movement of the head. This includes not only 360° VOD, but also the 360° live streaming of a large event such as a concert or a football game.

On the other hand, there is interactive VR, where users interact with a virtual environment with the aid of controllers and sensors. Interactive VR can either be single user or multi-user depending on the application. In multi-user VR, the users interact not only with the virtual environment but also other users. Interactive VR

allows the user to have a tactile experience, respectively by touch. In tactile VR, haptic technology imitates the feeling of touch by using sensors and actuators [15]. With the added dimension of touch, tactile experience allows for applications such as remote maintenance and remote surgery [16].

2.2.2 Delivery of VR over 5G Mobile Network

As of today, most commercial VR HMDs are wired, as the required latency and data rates are not up to standard. Wireless VR, i.e. the delivery of VR over mobile networks, is on the agenda of the VR industry. Wireless VR is expected to bring about a lot of applications, such as interactive video conferencing and looking around immersive multimedia while traveling. In order to achieve the switch from wired to wireless, high data rates and low latency offered by 5G is required [3].

There are two types of transmission schemes to deliver VR over a mobile network, namely omni-perspective and FoV transmission schemes. The omni-perspective transmission scheme is used to transmit a complete 360° view, meaning the view is already processed when the user rotates their head. Even though, this assures that the view is transmitted with identical quality, it wastes a lot of bandwidth for unseen parts of the environment.

In the FoV transmission scheme, the view is divided into multiple visual angles and generates a FoV which contains a high-resolution image within the FoV and a low-resolution image in the surrounding visual angles. A threshold for the head-turning angle is set and when this threshold is exceeded, the next FoV is transmitted at a high resolution. In Fig. 2.2, the FoV transmission scheme and its bandwidth requirement is represented. The bandwidth is almost constant and only exceeds a certain bandwidth when the next FoV needs to be transmitted.

The major advantage of this scheme is that the bandwidth is used more efficiently, which is a key element in the delivery over a mobile network. Nevertheless, this transmission scheme requires accurate and fast prediction algorithms to operate efficiently, since an inaccurate prediction results in a distorted image for the user. Since low latency is a desired quality for an efficient prediction, MEC is a useful technology to compute these predictions among the applications it offers for VR. In Section 2.4.1, this will be explained in more detail.

In order to deliver VR over a mobile network, data rate and latency requirements need to be specified. Motion-to-Photon (MTP) latency, i.e. the time needed for a user movement to be fully reflected on the virtual environment, is required to be lower than a certain threshold to avoid dizziness. In literature, there has been a consensus that the MTP latency should be lower than 20 ms, however, it has been noted that the threshold goes as low as 10 ms [7].

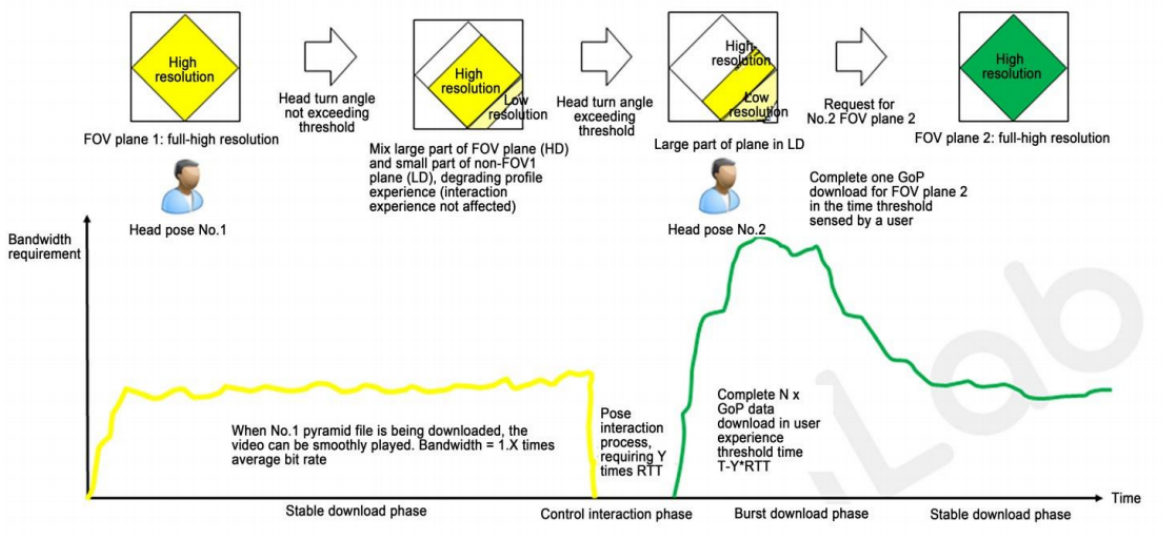


Figure 2.2: FoV Transmission Scheme [3]

The MTP latency sets an upper bound for the transmission and computation latency of the virtual environment, since exceeding this limit may cause health problems for the users. Local VR (delivered over USB or HDMI) applications are below the MTP bound, however when delivered over a 4G-LTE network, a total latency of above 150 ms is observed, due to the increase in transmission and computation latency [8]. In the context of non-interactive and VOD VR, methods such as proactive caching and computing, more elaborated in 2.4.1, are used to minimize the total latency, however, for live VR and interactive VR, this is not an option. With the data rates and technologies promised by 5G, the total latency is expected to be as low as local VR. Several approaches in order to achieve the optimal delivery over mobile networks will be investigated in the upcoming sections.

2.3 Specific Challenges around VR over 5G

As aforementioned, with the introduction of new technologies in 5G, there comes about certain challenges. For VR, low latency transmission and accurate prediction are among the more important challenges. In this section, two specific challenges will be elaborated on, namely the intermittent nature of mmWave channels and prediction algorithms.

2.3.1 Intermittent Nature of mmWave Channels

One of the technologies 5G is expected to bring about is the mmWave radio transmission technology. mmWave signals refer to the 30-300GHz band in the RF spectrum. It has been investigated that the V band (40 to 75 GHz) provides data rates up to 7Gbps for very short ranges (up to 10m), which is ideal for the data rates required for

VR [17]. Due to its short wavelength (1-10 mm), mmWave propagation suffers from blockage as mmWaves do not propagate well through obstacles, including the human body, which inflicts around 20-35 dB of attenuation loss; besides, there are almost no diffractions [7]. Therefore mmWave antennas require to be directional and the propagation needs a Line of Sight (LOS) component in order to have acceptable data rates for VR.

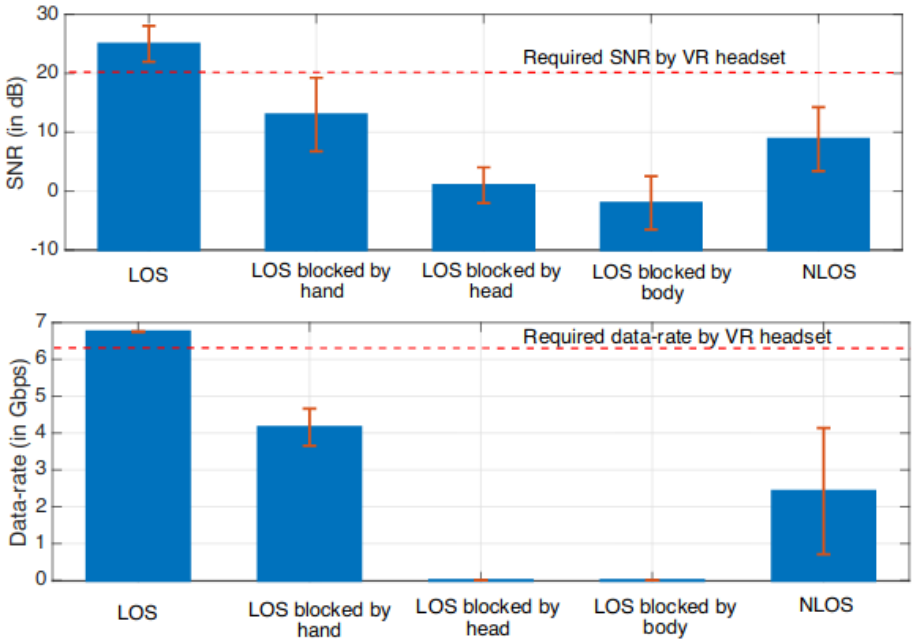


Figure 2.3: mmWave Blockage Impact on Data Rate [4]

In [4], a comparison of SINR and data rates between the unobstructed LOS path, LOS path with blockages such as the user's hand, between the HMD and the mmWave access point, and a different player's body, and the NLOS path has been investigated. The results, depicted in Fig. 2.3, show that only when there is a LOS path with no obstructions, the data rate for VR is acceptable. The LOS blocked by a hand causes almost a drop by half in data rate, and both the LOS blocked by a body and the NLOS path are even lower than the case with the blockage by hand. In Section 2.4.2, an approach to minimize the effect of blockages by using multiple mmWave access points will be elaborated.

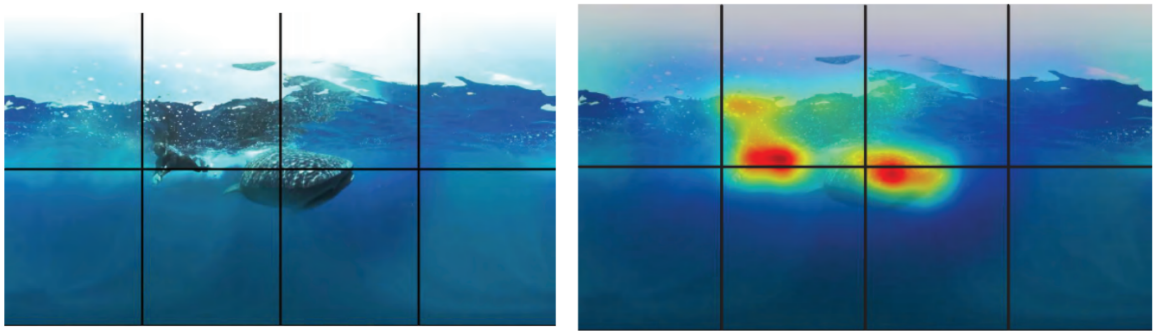
Channel prediction can also be investigated in order to determine the channel state information for hand-offs to other networks. Using the past and current Channel State Information (CSI), a forecast of the future channel state can be made using algorithms such as parametric estimation models [18] or autoregressive models [19] [20]. The trade-off in channel prediction is that if the prediction is too cumbersome, added computing latency will occur, which is not desirable for VR. This seems to be an interesting challenge to investigate in this research.

2.3.2 FoV Prediction Algorithms

The prediction algorithms are key in the FoV transmission scheme. Inaccurate and slow predictions can minimize the experience of users, due to the transmission of incorrect FoVs. There are two types of prediction algorithms, content-based and motion-based prediction respectively.

2.3.2.1 Content-Based Prediction

In this type of prediction algorithm, the factors determining where a user's eyes are fixated play a large role. Therefore it is also called saliency-based prediction. The saliency of a point is the state by which it stands out from its neighbors. When a point in a frame is highly salient, the user is very highly likely to focus on that point. The saliency approach is more likely to be used in the weak-interactive video on-demand VR. In [5], the authors note that the distribution of salient regions in the scene has a significant impact on how viewers explore a scene and that saliency-based predictions not only have an advantage in fixation prediction but also in image compression as less salient regions can be compressed more. In Fig. 2.4, the process of partitioning a 360° video to 8 tiles and determining their saliency weights is shown. According to these weights, the prediction is performed. In [10], the authors argue that the algorithm is time-dependent, as, after the first couple of seconds, user behaviour determines the next FoV. Therefore basing the full prediction on saliency may not seem the optimum solution, however, combined with another prediction algorithm, possibly an algorithm involving user movement, it might result in higher efficiency.



(a) 360°video divided into 8 tiles *(b)* Saliency weights of corresponding tiles

Figure 2.4: Saliency-Based Prediction [5]

2.3.2.2 Motion-Based Prediction

In motion-based prediction, the head movement patterns of users and the type of content play a large role. The prediction algorithms use extrapolation techniques such

as linear regression and weighted linear regression [21] based on the information of the head movement patterns of previous users and previous head movement patterns of the current user. In [11], based on the experimental study of viewer motion, the authors found the viewer motion can be well predicted in 100 to 500ms. This might seem to be well over the latency requirement. However, the authors state that, if viewer motion is perfectly known in advance, the bandwidth consumption and motion prediction time can be reduced by 80%.

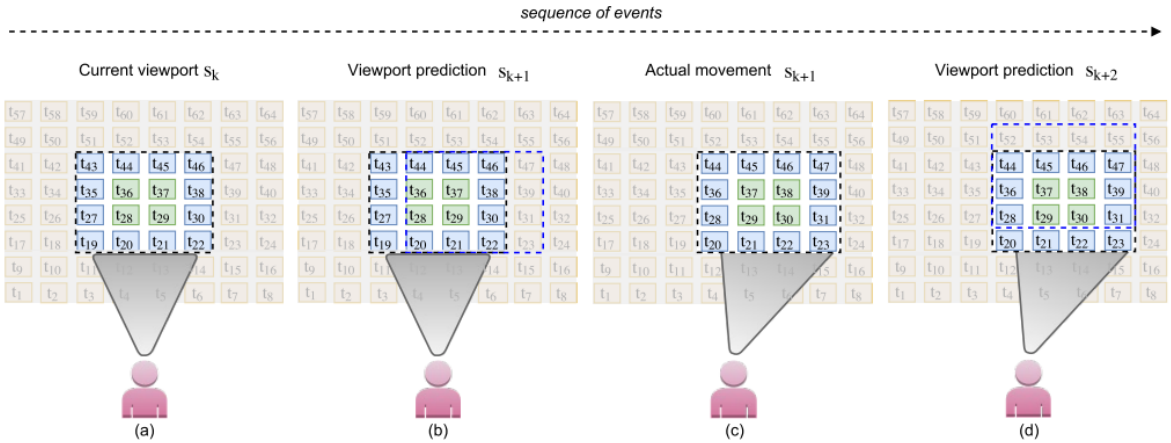


Figure 2.5: Working Principle of Viewport prediction [6]

In [6], using the angular velocity of the user's head, the aforementioned movement patterns, and type of content, the viewport-based transmission is introduced. A viewport is defined as the visible area of the user, so to say the FoV. In Fig. 2.5, the process of the tile-based viewport prediction is depicted. The FoV is partitioned into 16 tiles, such that the middle four tiles are based on where the user currently looks. The neighbouring cells are provided a lower quality than the middle cells, but they are high enough such that if the user moves that direction, the quality doesn't deteriorate that much. This approach is not only be used in VOD and interactive VR, but is also more likely to be used in live events, as it takes into account the movement behaviour of the user.

2.4 Approaches in Literature

In this section, approaches that are being researched in the wireless VR field will be elaborated. In Fig. 2.6, an overview of the requirements and the corresponding technology enablers is portrayed. In this section, the focus of the approaches will be on MEC and multiconnectivity.

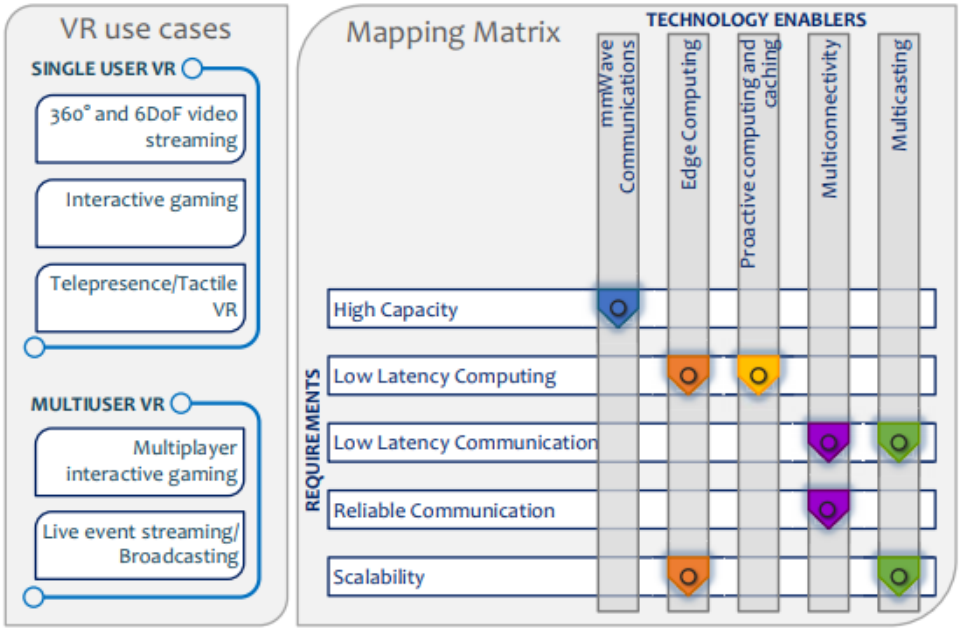


Figure 2.6: VR requirements and technology enablers [7]

2.4.1 MEC Approach

When low latency and high bandwidth are required, one of the first technologies that come into mind is MEC. HMDs need to be lightweight for long usage, meaning that it has a capacity limit and that high computations cannot be performed. MEC, in the context of VR, has been used in proactive caching and computing VR content to avoid a bottleneck in the backhaul link and minimizing the computation performed on the HMDs. In [22], a survey about different caching and mobile computing at different areas of the edge is presented. Caching at the edge is a way to reduce transportation latency, as it minimizes the distance of video content to travel. With the combination of device-to-device caching [23], not only the user can achieve content faster, but also the stress on the backhaul network is reduced.

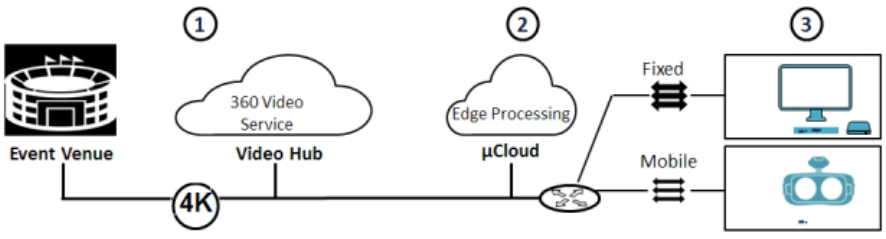


Figure 2.7: MEC used in a FOV rendering flow [8]

The edge not only is effective in caching but also computing, which is a desirable

aspect in order to minimize computation latency. In [8], edge processing is used for a FoV rendering solution for live streaming a VR event. In Fig.2.7, the flow for FoV rendering is depicted. Live footage from a 360° VR camera is sent to the video hub to create a VR environment and sent to the μCloud in order to perform motion prediction, FoV extraction and transcoding. When the processing is completed, it is transmitted to the user, and each time the user angle exceeds a threshold, a new FoV is sent to the user.

In [9], MEC is used in intelligent video acceleration for improved quality of experience and reduced bandwidth consumption. In Fig. 2.8, the flow of the content delivery with video acceleration is depicted. A Radio Access Network (RAN) analytics app in the MEC server provides throughput information to the video content server, in order to reduce TCP congestion and provide better video quality.

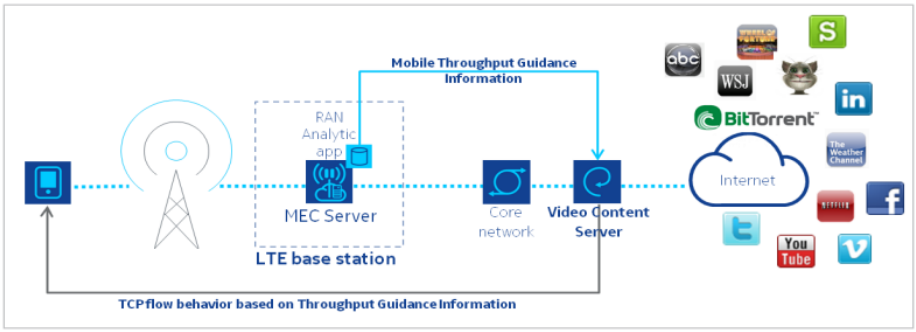


Figure 2.8: MEC used in a Intelligent Video Acceleration Service Scenario [9]

Not only the RAN analytics app can be used in TCP congestion control but also channel-aware adaptive bit-rate streaming. In [24], the MEC is used to implement adaptive bit-rate video streaming using the Channel Quality Indicator (CQI) metric provided from the RAN analytics app. The video quality can therefore be lowered when the channel is below a certain CQI level, which would lower the latency considerably. In VR, latency is more important than quality due to the health effects, so this could be an interesting approach to investigate.

2.4.2 Multiconnectivity Approach

mmWave is highly susceptible to blockages which result in lower SINR and data rates, as aforementioned in Section 2.3.1. In [25], multiple mmWave access points are connected to a MEC network, as depicted in Fig. 2.9. Each square in the VR arcade represents a tracking pod which is mapped into the virtual environment.

Using a time-slotted mode, all mmWave APs sequentially perform beamforming to find the best path between the APs and the HMD. When multiconnectivity with beamforming is allowed, a reduction in communication delay up to 25% percent is

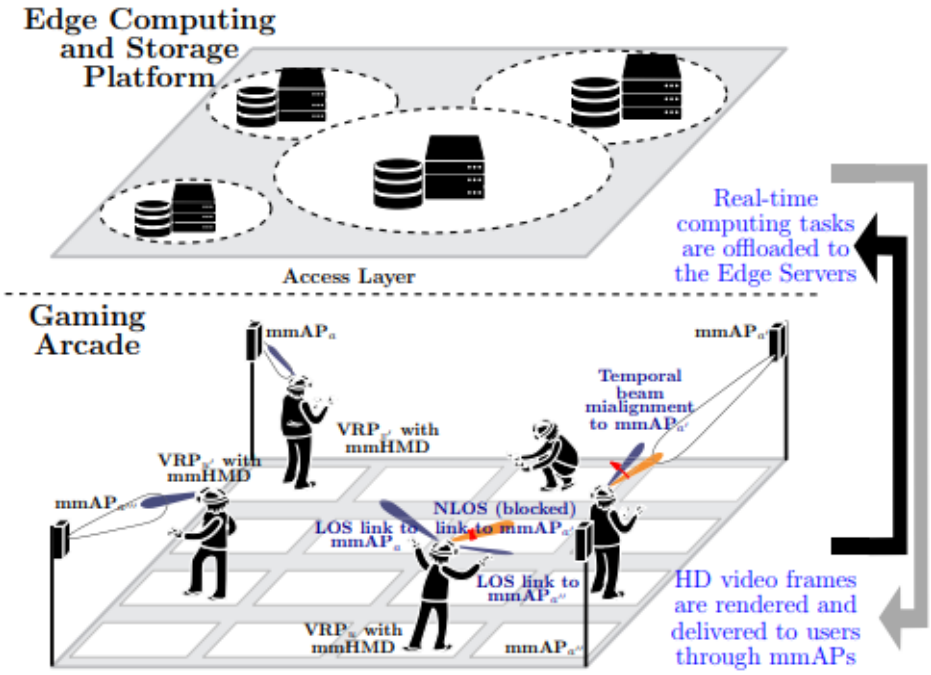


Figure 2.9: Multiple mmWave AP in VR gaming arcade [7]

achieved with an increasing number of players. An increase in mmWave APs, on the other hand, reduced the communication delay up to 30% percent compared to a low number of mmWave APs. Multiconnectivity is an interesting approach to investigate, not only in the mmWave context but also in other forms of wireless communications.

2.4.3 Combining RAN Virtualization and Mobile Edge Computing Approach

One of the advantages of 5G is that it is expected to provide virtualization in the RAN, which means that multiple radio access technologies can be accessed via a centralized base station [26]. In [27], a cloud RAN, or CRAN, paired up with the MEC is regarded as a cost-effective way for the mobile network operators (MNO) to be able to support 5G applications. From the MNO perspective, the cost of providing MEC across an already planned pool of centralized processing points in the CRAN should be significantly lower than a standalone MEC deployment. The benefits of the co-deployment of CRAN and MEC are not limited to reduced CAPEX and OPEX. The communication between the CRAN and MEC can result in more efficient delivery of VR content over 5G. The intelligent video acceleration service or the CQI aware adaptive bit-rate streaming from the previous section can be performed much more efficient due to the MEC and CRAN being in the same location.

Chapter 3

Modelling and Delivery Strategies

In this chapter, the modelling of the 360° video environment and potential delivery strategies will be described. The 360° video environment is modelled based on the points of interests in the research questions. Once the models are described, it is possible to develop delivery strategies, according to the descriptions and assumptions of these models.

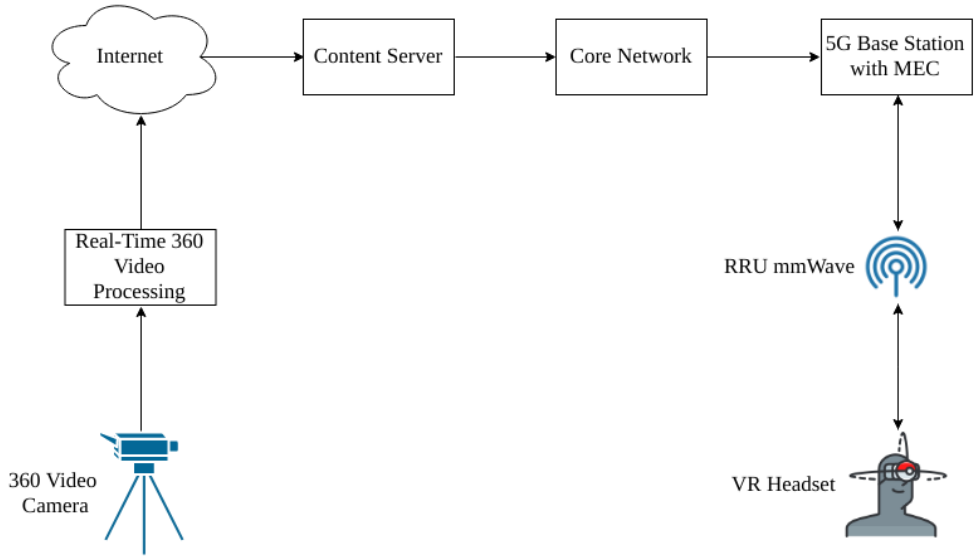


Figure 3.1: High-level Model of VR Delivery over 5G

3.1 Modelling

First, the high-level model will be described, which is depicted in Fig. 3.1. This model consists of the video shooting and processing part, the content delivery part, the MEC, the wireless transmission channel and the user. A choice had been made to focus on non-interactive 360° video, which suggests that the video processing can either be done

in real-time or beforehand. After the video is processed it will be sent to the content server and via the core network to the MEC server. After the necessary computations are completed, the FoVs will be sent to the user using the mmWave channel. There is a duplex communication between the headset and the remote radio unit (RRU) and the base station (BS) with MEC server to feed the channel and FoV information back and forth.

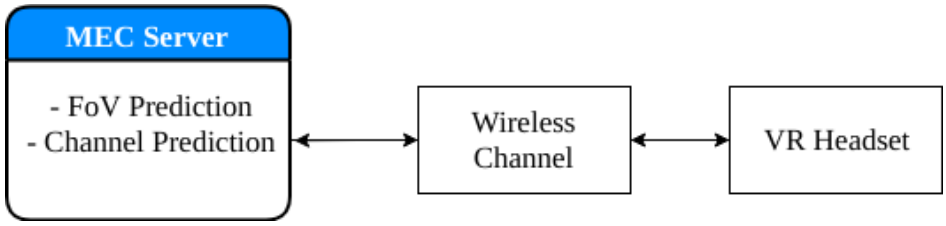


Figure 3.2: Detailed Model Focusing on the MEC Server, Channel and Headset

The video shooting and processing part are not the focus of this research suggesting that they can be abstracted. The core network and the content server are not the main points of interest so they are also removed from the model providing a more detailed to work with as demonstrated in Fig. 3.2. In this detailed model, the main focus is the MEC server located in the 5G base station, where the FoV and channel prediction takes place. The VR headset detects the movement of the user with sensors and transmits the information to the MEC. By doing this the correct FoVs, that are going to be stored in the HMD buffer, are obtained.

3.1.1 Wireless Channel

For the sake of simplicity, the channel quality is assumed to vary between two levels, high (7 Gbps) and low (3 Gbps). Rather than using a deterministic or a completely random fluctuation between these two levels, it would be more reasonable to have a controlled randomness. In [28], this is achieved by modelling the wireless channel as a Discrete-Time Markov Chain (DTMC). The wireless channel can be considered as a two-state DTMC as depicted in Fig. 3.3. The p_h and p_l probabilities refer to the probabilities of staying at high and low quality respectively. The advantage of using a Markov Chain is to be able to create different patterns of channel quality by using different values of p_h and p_l.

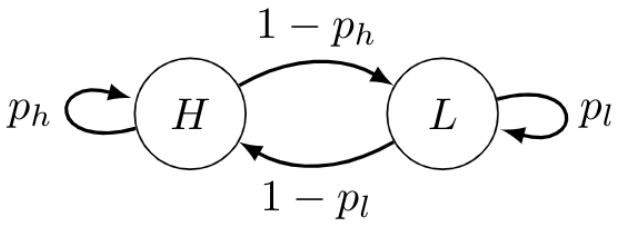


Figure 3.3: Two-state Markov Chain for Channel Quality

A simple way to understand how to achieve different channel quality patterns is to calculate the mean time spent in staying at the same state,

$$T_h = \frac{1}{1 - p_h} \cdot 5\ ms \tag{3.1}$$

$$T_l = \frac{1}{1 - p_l} \cdot 5\ ms \tag{3.2}$$

given that the every 5 ms the channel is updated. The T_h and T_l notation, rather than the p_h and p_l notation, will be used in the results as it gives more of an understanding of how the channel quality pattern looks.

3.1.2 Video Model

One of the most important parts to describe is the 360° video model. As aforementioned, omni-perspective transmission is neither efficient nor possible over wireless channels due to bandwidth constraints, so it is crucial to split each video frame into FoVs, as done in [5]. The way this is done is to convert the spherical video into a rectangular plane. The first decision we make is that a frame is split into 9 FoVs, as depicted in Fig. 3.4. 9 splits allows for 120 by 60 degree FoVs, which is quite realistic. An assumption for this video model is that the user is presumed to watch only one FoV per frame. By making this assumption, we can use statistical viewing behaviour (see values on each FoV depicted in Fig. 3.4). More detailed information about statistical viewing behaviour will be provided in Section 4.3.

Considering the VR delivery, a computation and transmission latency has to be decided. A decision has been made to set the transmission time for sending a FoV in a high channel quality to 5 ms. This suggests that for a low channel quality this would take around 12 ms. The second decision we make is that the frame rate is set to 20 FPS, so essentially every 50 ms a frame is sent. This suggests that there is time to transmit 10 FoVs in a high quality channel.

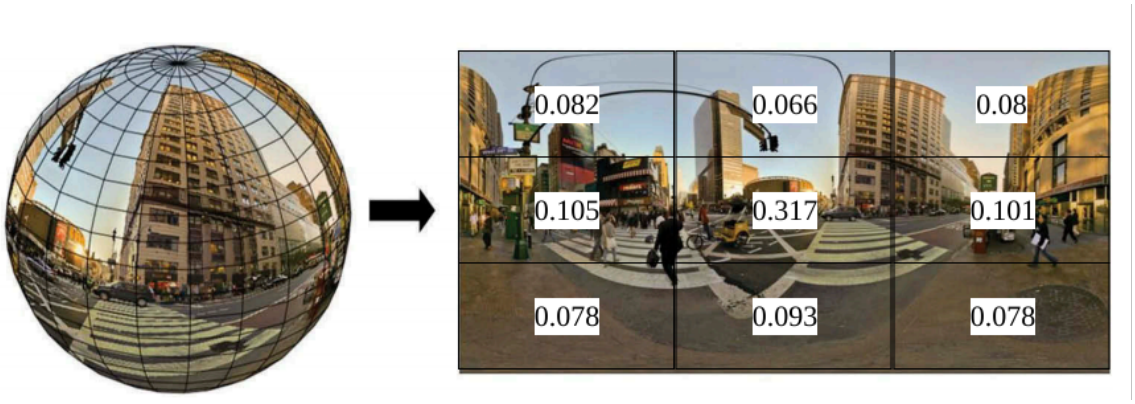


Figure 3.4: Equirectangular Projection of a 360°video

3.1.3 MEC Server

In this research, the MEC server is responsible for calculating the FoV and channel quality prediction, which are two important enablers of making VR wireless, due the capacity restrictions of sending an omni-perspective 360° video. FoV prediction has to be performed with high accuracy and low latency, in order to make sure there are no frames dropped and prevent the MTP latency effecting the user. Channel quality prediction, on the other hand, while not being a necessity, can be a useful feature to improve QoS and efficiency by knowing when to transmit FoVs to the headset beforehand. The modules for the FoV and channel prediction will be explained in detail in Section 4.3. The only assumption we make for the MEC is that all frames with their individual FoVs are directly available at the MEC, as, in this research, we are not considering the video delivery part to the MEC.

3.1.4 VR Headset

The VR headset can be simply seen as a buffer connected to a processing unit that selects the necessary FoVs by using the movement information collected by the headset sensors. Since efficiency in VR delivery is part of this research, the battery on the headset is an important factor. The battery consumption in the headset is directly related to the time the wireless channel is utilized.

3.2 Preliminaries for Definition of Delivery Strategies

Before introducing the strategies, it is crucial to define each individual event, since time-lines are used to explain each strategy. There are three types of events in the delivery of VR, namely FoV transmission, user-movement refresh and frame refresh.

As aforementioned, the *FoV transmission* takes 5 ms on high channel quality. For low channel quality transmission, on the other hand, we would require 15 ms to send a FoV. Since we consider a frame rate of 20 FPS, the *frame refresh* happens every 50 ms. This allows for 10 timeslots to be available for FoV transmission on high channel quality for each frame. There is also the MTP latency that has to be taken into account in wireless VR. Therefore we make a choice to update *user-movement* 20 ms before every frame refresh.

Any event happening before the user-movement refresh is referred to as the *proactive period*, as this is the period where the FoV prediction is going to be used. This is due to the fact that we do not know where the user will move to in the upcoming frame. The events after the user-movement refresh is referred to as the *reactive period*. Since we have information about where the user is going to move, the MEC knows the correct FoV to transmit. Considering there are 10 timeslots, the first 6 are used for the proactive period and the last 4 are used for the reactive period.

To visualize these events in the delivery strategies, time-lines (with arrows pointing to the events), depicted in Fig. 3.5,3.6,3.7, are used. The blue and green arrows refer to user-movement and frame refreshes, while the red and the magenta arrows refer to FoV transmission in the proactive and reactive periods respectively.

3.3 Delivery Strategies

We will propose 2 main strategies that will be investigated; reactive and proactive. The combinations of these strategies with varying thresholds will also be investigated for several reasons that will be explained in the respective subsections. For these strategies, a choice has been made to only send FoVs when the channel quality high, and this can be achieved in one timeslot. The reason for not sending anything in a low channel quality state is due to the fact that 3 consecutive low channel quality timeslots are required to transmit one FoV. Without using channel prediction, it is not possible to know if there will be three consecutive low quality timeslots. This can result in energy overheads. Suppose there is a high quality timeslot after two low quality timeslots and we begin transmission in the low quality timeslots. Energy worth two FoV transmissions is wasted, as the transmission in low quality is not completed, but rather completed at a high quality channel. It would have been more logical to wait two timeslots, and transmit in the high quality state.

3.3.1 Strategy 1: Reactive

The first strategy is purely dependent on the user head-movements. Every 30 ms the movement information is refreshed and within 20 ms (MTP) the frame needs to be

refreshed to avoid motion sickness. We can use this to our advantage, and only send the necessary FOV after the movement is refreshed, i.e. in the 20 ms period. Since one FOV can be sent within one timeslot of 5 ms and nothing is done proactively, the efficiency is at a maximum. The only downside is that, if the 20 ms period is in a state of bad channel quality, the FOV can't be sent to the headset. However, as only one high quality channel timeslot is sufficient for sending the correct FoV, the expectation is that this strategy will yield the high QoS and lowest efficiency.

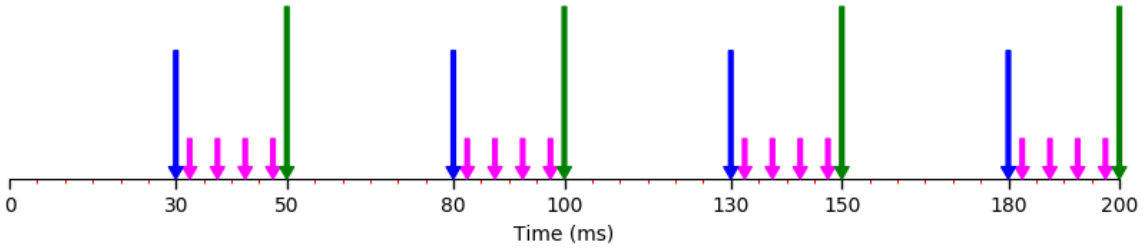


Figure 3.5: Reactive Strategy

3.3.2 Strategy 2: Proactive

The second strategy is to do everything proactively, basing the predictions on statistical data collected from other users. Instead of waiting for the movement refresh at every 30 ms, the refresh point is taken as the checkpoint of whether the necessary FOV is in the headset. This allows to send more than half of all tiles if necessary, as there are 6 timeslots of 5 ms, however setting a probability threshold is more logical due to efficiency requirements. The probability threshold is taken as the sum of the n-largest tiles, where n can be 6 at most. Once this threshold is exceeded, the MEC stops sending tiles to preserve energy.

Since the statistical data is provided with probabilities to which FOV the user is going to look at, a reasonable threshold of 0.9 can be selected for the sum of the FOV probabilities. This makes sure that there is a trade-off between QoS and efficiency. However, for low channel quality, the QoS purely depends on how accurate the predictions are.

3.3.3 Strategy 3: Combine Reactive and Proactive

The third strategy is to combine the reactive and proactive strategies. The way to achieve this is to perform the proactive strategy and if the required FOV is not in the headset, the reactive strategy can be put into force. This way, we make sure that the necessary tile is definitely in the headset. The threshold of 0.9 will be kept the same as in the proactive strategy to make a reasonable comparison. It is expected that in

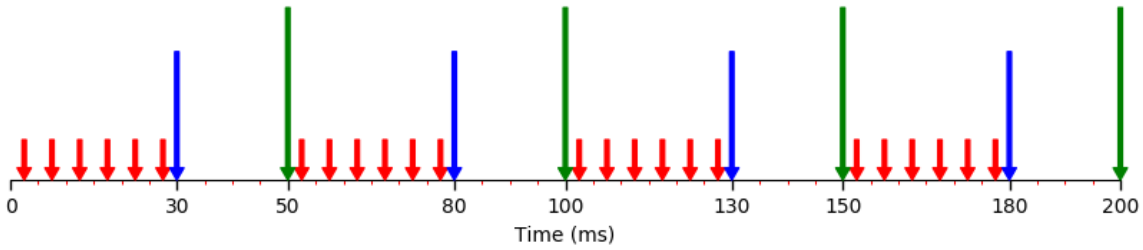


Figure 3.6: Proactive strategy

the QoS department, this strategy will perform both better than the reactive and the proactive, however with a trade-off in efficiency.

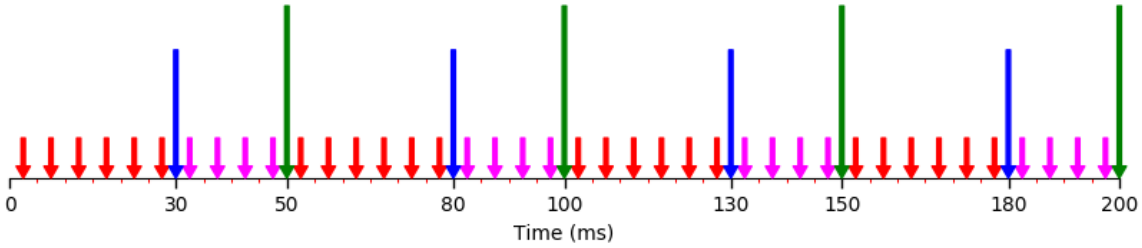


Figure 3.7: Combined Strategy

To see how much effect the threshold has on the QoS and efficiency trade-off in the combined strategy, it would be reasonable to vary the threshold. A higher threshold means more FoVs to be transmitted given the channel quality permits, but in return is less efficient since the channel is used more. A lower threshold might results in a lower QoS, but is more efficient as less FoVs are transmitted. Finding an optimum trade-off between QoS and efficiency is the main objective of this research, so varying the threshold can provide an insight in this matter.

Chapter 4

Evaluation Methodology

In this chapter, the methodology to evaluate the aforementioned models and delivery strategies will be explained. First of all, the evaluation criteria will be introduced, as they will be the reference points to build the simulator. Once they are introduced, the motivation to choose which simulator will be used in this research will be provided. Thirdly, the datasets that are going to be used, and how they are going to be created, will be discussed. The chapter will be concluded by four scenarios to evaluate the simulator.

4.1 Evaluation Criteria

Before designing the simulator, it is necessary to identify the performance indicators. In accordance with the research questions, QoS and efficiency of the simulator are the main points of interest. QoS, per definition, is a very broad term and could mean everything from throughput to latency. In this research, QoS is defined as the percentage of frames where the correct FoV is available in the HMD at the correct moment.

$$QoS\ \% = \frac{\#\ of\ Transmissions\ Resulting\ in\ Correct\ FoVs}{\#\ of\ Frames} \cdot 100 \qquad (4.1)$$

Efficiency, on the other hand, can be measured by the percentage of time when a FoV is transmitted, i.e. the channel usage ratio. It is clear that efficiency and channel usage ratio are reciprocal. The lower the channel usage ratio is, the higher the efficiency becomes.

$$Channel\ Usage\ Ratio\ \% = \frac{\#\ of\ Timeslots\ used\ for\ FoV\ transmission}{Total\#\ of\ Timeslots} \cdot 100 \quad (4.2)$$

In the simulations, we are going to use *cumulative QoS and channel usage ratio*. For each frame, whether the correct FoV is sent or not, and how many timeslots are used for a FoV transmission will be stored and used in the aforementioned equations.

This way we can investigate how the QoS and channel usage ratio vary as each frame appears. The final QoS and channel usage ratio values will also be used to create confidence intervals.

4.2 Development of Simulator

There are two alternatives to start developing a simulator; choosing a proprietary software or to build one from scratch. There are both advantages and disadvantages to use proprietary software. One of the main advantages is that the simulator is already pre-built and it emulates the real-world conditions as accurate as possible. The disadvantage is that there is a learning curve and it is very hard to modify the simulator. Building your own simulator has a very important advantage in having complete comprehension of what is in the code and having the ability the abstract and modify the code.

In the initial two weeks of the implementation phase, NS3 [29] was chosen for building the simulator. NS3 is a discrete-event network simulator that emulates network systems as realistically as possible. It uses C++ as the primary language however Python can be used to visualize the model. After trying to understand and modify a MEC model that was available online, it was apparent that the aforementioned disadvantages would hinder the progress, due to the steep learning curve. It was complicated to build modules that contain the aforementioned evaluation criteria. Therefore the motivation to build a simulator from scratch arose.

Due to its simplicity and diverse library, Python was chosen to be the language that the simulator was built upon. Python has a library called simPy [30] that will be used to emulate the events in the simulation. The simulator mimics the system model described in Section 3, see Fig. 3.2. As specified in the detailed models, the simulator has to consist of three main parts; the MEC, the channel and the user. The MEC needs to be able to predict and send the correct FoV to the HMD. Therefore, a statistical dataset has to be created for the MEC part of the simulator. For the channel part, different traces can be created at the moment of the simulation. Finally, for the user part, a dataset for the user movements needs to be found/created.

4.3 Statistical and User FoV Dataset

In this section, the FoV datasets will be discussed. The FoV prediction data concerning this research is either not that available. The available data is most of the time head movement information which is quite complicated to map to probabilistic data that can be used by the aforementioned delivery strategies, especially the ones requiring the FoV prediction. Therefore, a decision has been made to create the FoV datasets. The

strategies are designed in a such a way that any (statistical) data can be used. The simulator is not built around the data, rather the data is created in a way that it is appropriate for the simulator.

One way to create the statistical dataset is to emulate the motion-based prediction as simple as possible. In the FoV model, the assumption is that there are 9 possible FoVs in every frame. We assume that the video wraps around the horizontal plane and not the vertical plane. If the user is looking at the top or bottom row of the grid, they will have 6 possible movements and for the middle row, they will have 9 possible movements. One way to create the statistical dataset is to assume that the user will keep looking at the same FoV with the highest probability (p_{max}) and starts looking to the neighbouring FoVs with equal probability ((1-p_{max}) /n-1, where n can be 6 or 9). In order to have a reasonable realization, a video of 50 seconds with 20 FPS is considered. This suggests that a dataset of 1000 frames has to be created. During the creation of this dataset, every x *seconds* or every $20 \cdot x$ frames on average, the users return to the middle FoV with a high probability of $p_{return} > 0.8$, which is achieved by creating a uniform distribution between 20(x-1) and 20(x+1) frames, $U(20x - 20, 20x + 20)$, during the dataset creation process. The reason we want to do this is to achieve a more realistic dataset by allowing the user to roam around but eventually return to the middle FoV to prevent too much random movement. It is very likely that a user will look to a certain FoV when a salient image shows up in the 360°video. When considering 1000 users, this dataset creation method has to be run for 1000 times. To obtain the statistical FoV dataset, for each frame the number of occurrences of each FoV is counted and normalized. The user data can then be selected randomly from the dataset containing the runs for 1000 users.

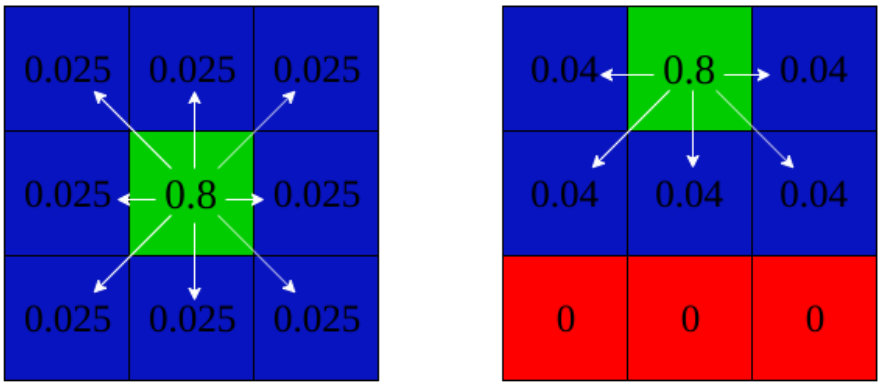


Figure 4.1: Head-movement of a user from the middle FoV to an upper row FoV

In Figure 4.1, a case is illustrated where p_{max} is 0.8. A p_{max} of 0.8 is reasonable, since a movement refresh of 50 ms is considered in this simulation and the user isn't expected to move excessively within this period of time. The green boxes indicate the

FoV the user is actually looking at, while the blue boxes indicate the neighbouring FoVs, where the user has a probability to move to. The arrows indicate the movement of the user from one FoV to a neighbouring FoV. An important thing to emphasize is the fact that when the user moves to the top row (or bottom row), the user can either stay in the same FoV, or move to the surrounding 5 FoVs, rather than all of the 8 FoVs as explained before.

4.4 Simulation Scenarios

It is important to establish simulation scenarios, to investigate the QoS-efficiency trade-off.

1. Different Users: The first scenario is using different users, as this would both indicate whether the simulation is working correctly and whether the datasets are created satisfactorily. Since the users are essentially different runs in creating the statistical dataset, the expectation is that the strategies will perform similarly for different users given that the channel qualities are equal.

2. Different Video Datasets: To see the effect of the video datasets, the p_{return} and x values mentioned in Section 4.3 are going to be varied.

3. Different Channel Qualities: In order to compare the strategies, the channel qualities are varied using the mean times at staying in respective states T_h and T_l.

4. Different Thresholds: For the combined strategy, the threshold determines how much the proactive and reactive strategies are going to be used. This in turn will have an effect on the efficiency.

Chapter 5

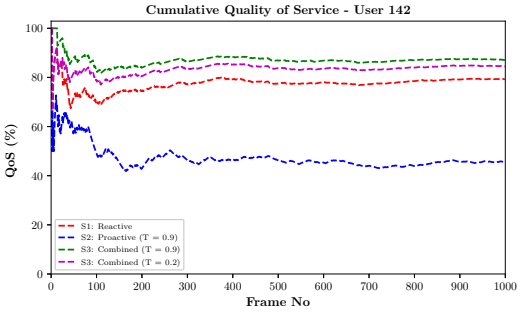
Evaluation and Analysis

In this chapter, the results of the simulations using the aforementioned scenarios will be discussed. First, a scenario where different users with equal channel conditions will be considered. Second, the dataset parameters will be varied to create three datasets and the results will be compared. Third, the channel will be varied for the same user and the effects will be evaluated. Fourth, the threshold of the combined strategy will be varied to see its effects on the trade-off between QoS and efficiency. Finally, the effect of channel quality prediction on the aforementioned delivery strategies will be investigated.

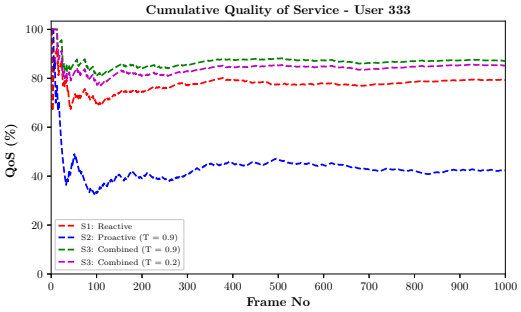
5.1 Different Users

The first scenario is using different users, as this would both indicate whether the simulation is working correctly and whether the user datasets are created satisfactorily. Since the users are different runs in the process of creating the statistical dataset, the expectation is that the strategies will perform similarly for different users given that the channel qualities are equal. From Fig. 5.1, it is clear that when considering four different users, the cumulative QoS approaches to roughly the same value for each strategy, given that the channel quality conditions are equal. In terms of QoS, it appears that the combined strategy with the higher threshold outperforms the rest of the strategies for a fast varying channel. When the channel is slowly varying and has an equal ratio of T_h and T_l, the reactive and the two combined strategies have the same behaviour. This can be explained by the fact that there is barely enough time to transmit one FoV and for 70 % of the frames (upon calculations), this is done in the reactive period of the combined strategy. In terms of the channel usage ratio, the results are expected to be exactly the same for different users, due to the fact that the channel determines when a FoV is sent or not, and the same channel conditions are used for the four users. When investigating the results of Fig. 5.2, it is clear that the results are identical for different users. The combined strategy with the higher threshold and

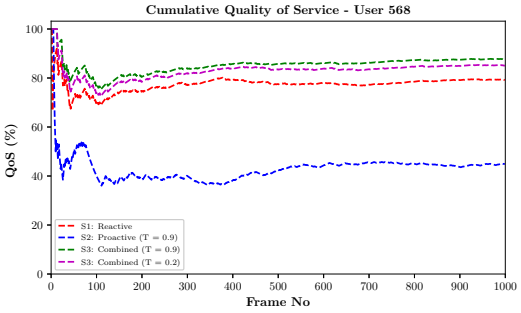
the lower threshold are above and below the line of the proactive strategy in Fig. 5.2.



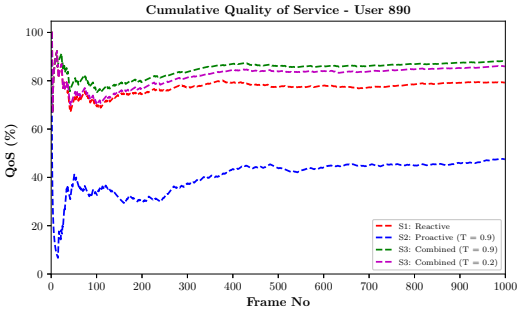
(a) User 142 - $T_h = 20$ ms and $T_l = 20$ ms



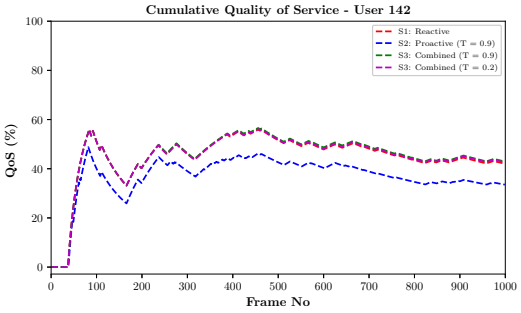
(b) User 333 - $T_h = 20$ ms and $T_l = 20$ ms



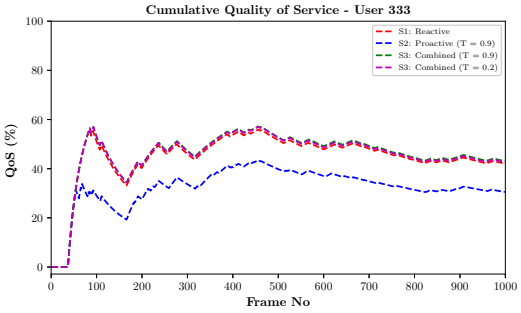
(c) User 568 - $T_h = 20$ ms and $T_l = 20$ ms



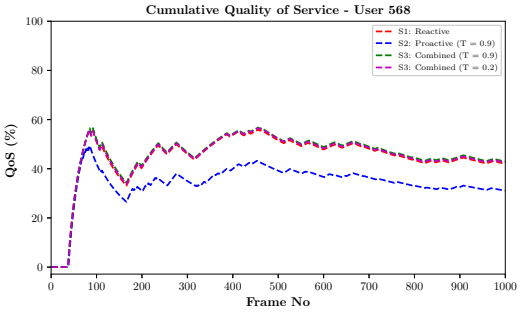
(d) User 890 - $T_h = 20$ ms and $T_l = 20$ ms



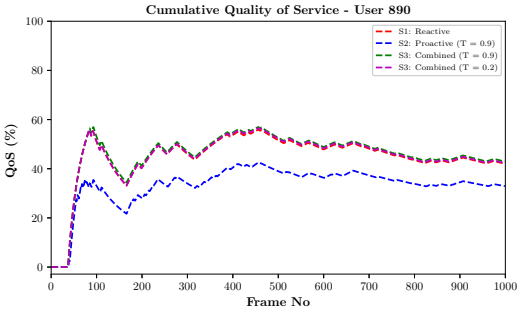
(e) User 142 - $T_h = 1$ s and $T_l = 1$ s



(f) User 333 - $T_h = 1$ s and $T_l = 1$ s

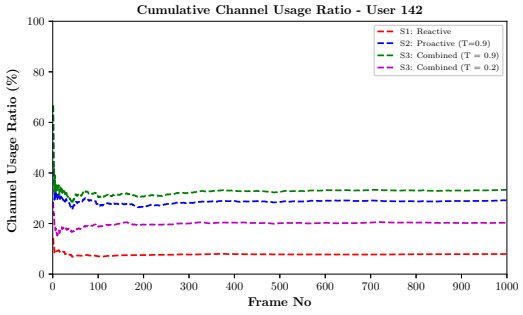


(g) User 568 - $T_h = 1$ s and $T_l = 1$ s

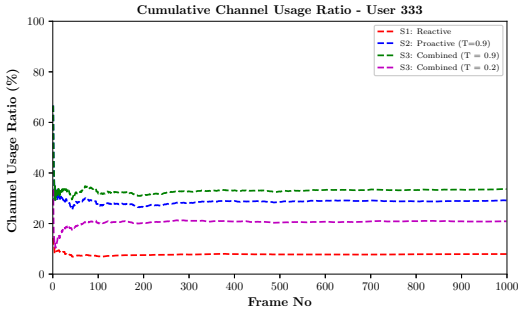


(h) User 890 - $T_h = 1$ s and $T_l = 1$ s

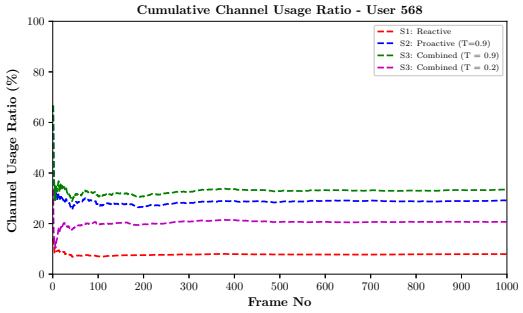
Figure 5.1: Cumulative Quality of Service for Different Users



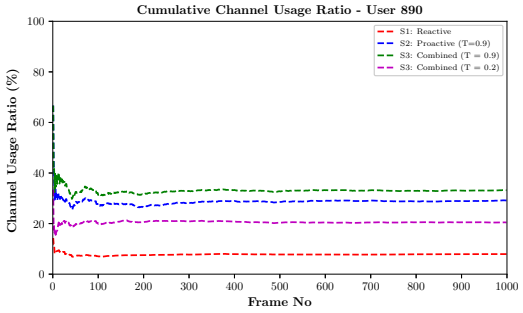
(a) User 142 - $T_h = 20$ ms and $T_l = 20$ ms



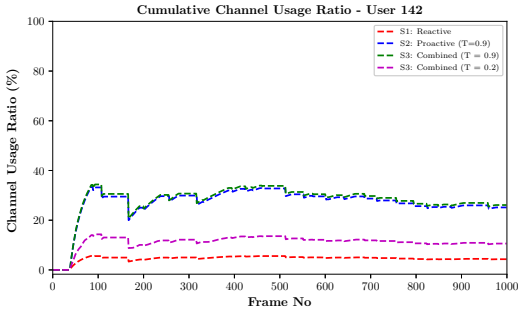
(b) User 333 - $T_h = 20$ ms and $T_l = 20$ ms



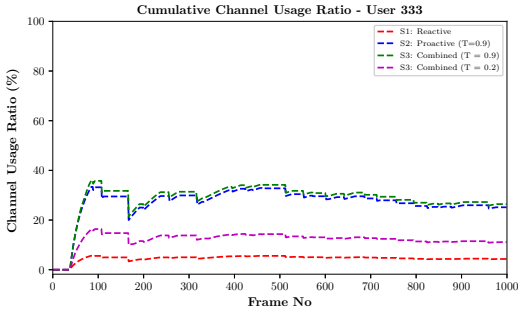
(c) User 568 - $T_h = 20$ ms and $T_l = 20$ ms



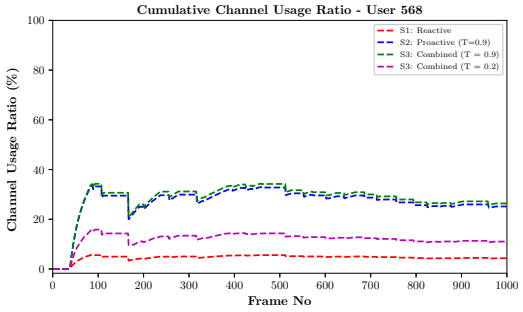
(d) User 890 - $T_h = 20$ ms and $T_l = 20$ ms



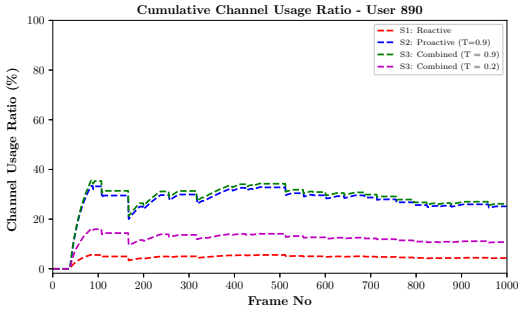
(e) User 142 - $T_h = 1$ s and $T_l = 1$ s



(f) User 333 - $T_h = 1$ s and $T_l = 1$ s



(g) User 568 - $T_h = 1$ s and $T_l = 1$ s



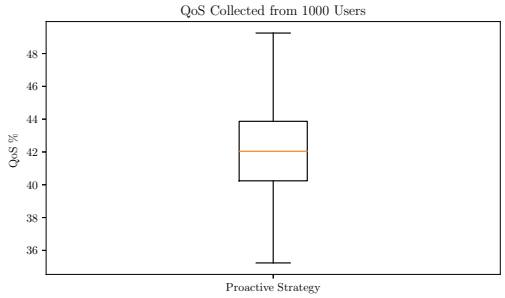
(h) User 890 - $T_h = 1$ s and $T_l = 1$ s

Figure 5.2: Cumulative Channel Usage Ratio for Different Users

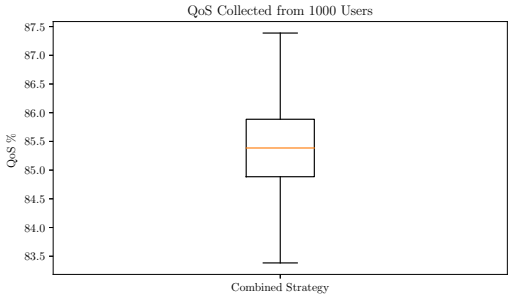
Considering the results from the cumulative QoS graphs were almost equal for both of the combined strategy, it is apparent that using a higher threshold only yields an extra overhead in channel usage ratio. In this test scenario, the reactive strategy is the most energy efficient strategy, as it requires only one timeslot to transmit the correct FoV.

Accuracy of the Simulation

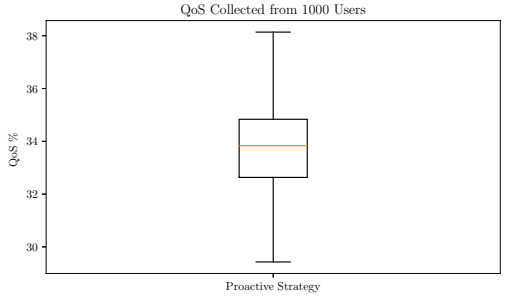
Investigating the results of different users in Fig. 5.1 and Fig. 5.2, it is clear that given equal channel qualities, the QoS is almost equal. However, before a generalization can be made that one user is representative of all users, it would be logical to look at the confidence intervals of each strategy for both channel qualities.



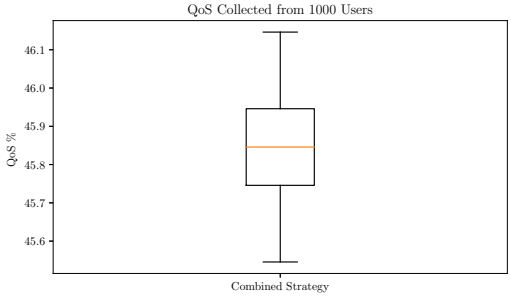
(a) Proactive - $T_h = 20$ ms and $T_l = 20$ ms



(b) Combined - $T_h = 20$ ms and $T_l = 20$ ms



(c) Proactive - $T_h = 1$ s and $T_l = 1$ s



(d) Combined - $T_h = 1$ s and $T_l = 1$ s

Figure 5.3: Box Plots of QoS collected from 1000 Users

We already know that the reactive strategy is independent of the FoV probabilities, so there is no need to calculate the confidence interval. For the proactive and the combined strategies, however, when the channel qualities are equal, the only variables are the FoV probabilities which are different for all users. In order to calculate the confidence intervals, all of the 1000 users are put through the same channel conditions and their QoS is stored. The two channel conditions provided in Fig. 5.1 and Fig. 5.2, are again used, to see whether the confidence intervals change drastically. The box

plots of the QoS collected from 1000 users are provided in Fig. 5.3. It is clear that the QoS for the proactive strategy is a bit more spread out, as the FoV probabilities have more of an effect when compared to the combined strategy, as there is no reactive part. To calculate the confidence interval, we have to check whether the samples satisfy the central limit theorem. From all four box plots in Fig. 5.3, it is clear that the distributions are either symmetrical or not too skewed. Since we have a large sample, we can assume that the samples satisfy the central limit theorem and calculate the confidence interval as follows [31]:

$$CI = \bar{x} \pm z\frac{\sigma}{\sqrt{n}} \tag{5.1}$$

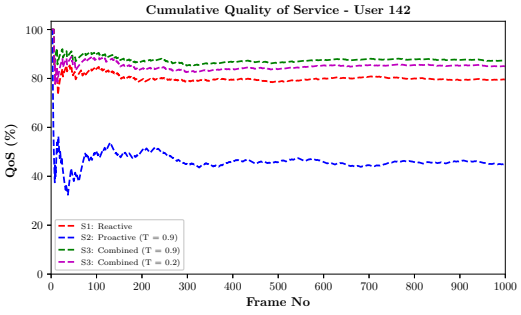
where $\bar{x}$, z, σ,n denote the mean of the sample,z-score,standard deviation and the sample size respectively. Since we are interested in the 95% confidence interval, the z-score is 1.96. The confidence intervals calculated from 1000 users are provided in 5.1. It is clear that the confidence intervals are narrow for the proactive strategy and even narrower for the combined strategy. Therefore it can be concluded that one user can be representative of the rest of the users. From now on, only one user will be tested in the upcoming scenarios.

Table 5.1: Confidence Intervals

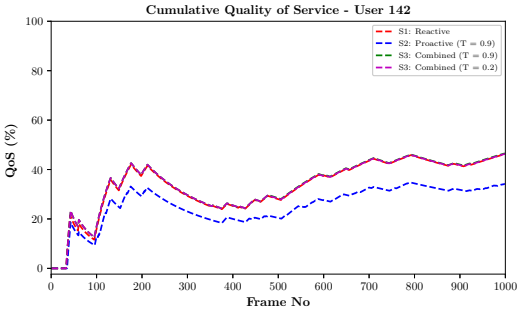
	Proactive ($T_h = T_l = 20ms$)	**Combined** ($T_h = T_l = 20ms$)	**Proactive** ($T_h = T_l = 1s$)	**Combined** ($T_h = T_l = 1s$)
Mean	42.1069	85.3705	33.7368	45.8600
Lower Limit	41.9475	85.3250	33.6321	45.8483
Upper Limit	42.2662	85.4161	33.8415	45.8717

5.2 Different Video Datasets

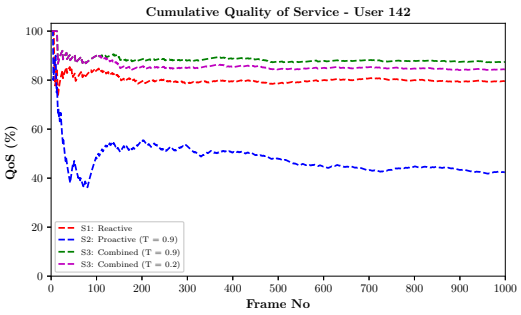
The second scenario is to test different datasets in the simulator. The p_{return} and x values mentioned in Section 4.3 are varied in order to create three datasets. In Dataset 1, it is assumed that the users return to the middle FoV on average every 2.5 seconds with a probability of 1. In Dataset 2, it is assumed that the users return to the middle FoV on average every 5 seconds with a probability 1. In Dataset 3, on the other hand the users return to the middle FoV every 10 seconds, however, with a probability of 0.8. The reason we use a p_{return} of 0.8 in Dataset 3 is that the majority of the users return to the middle FoV, but there are exceptions. The datasets are again subject of two channel qualities, as in Section 5.1. The results of the experiments with different datasets are presented in Fig. 5.4 and 5.5.



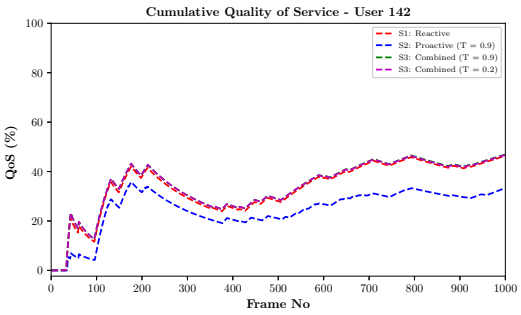
(a) Dataset 1 - $T_h = 20$ ms and $T_l = 20$ ms



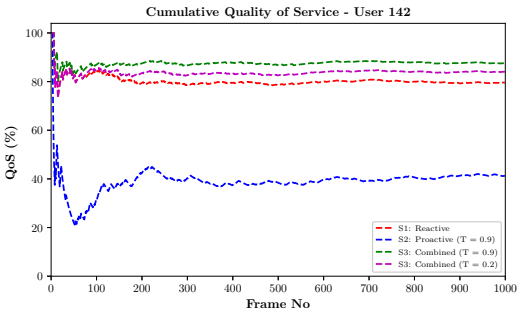
(b) Dataset 1 - $T_h = 1$ s and $T_l = 1$ s



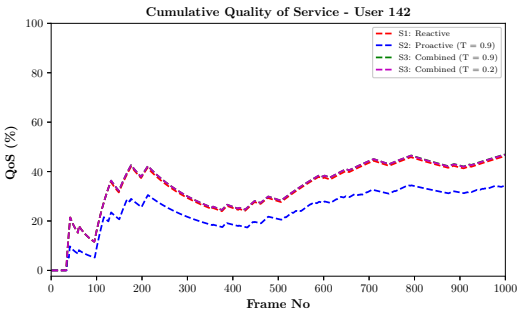
(c) Dataset 2 - $T_h = 20$ ms and $T_l = 20$ ms



(d) Dataset 2 - $T_h = 1$ s and $T_l = 1$ s

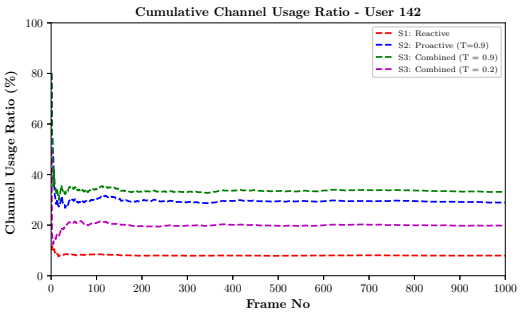


(e) Dataset 3 - $T_h = 20$ ms and $T_l = 20$ ms

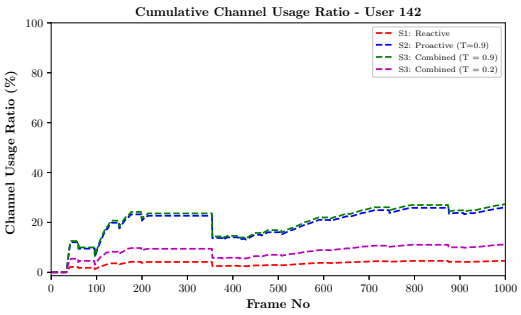


(f) Dataset 3 - $T_h = 1$ s and $T_l = 1$ s

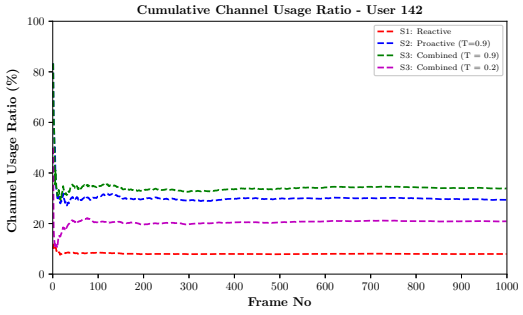
Figure 5.4: Cumulative Quality of Service for Different Datasets



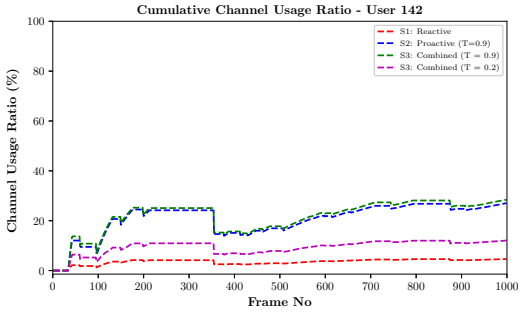
(a) Dataset 1 - $T_h = 20$ ms and $T_l = 20$ ms



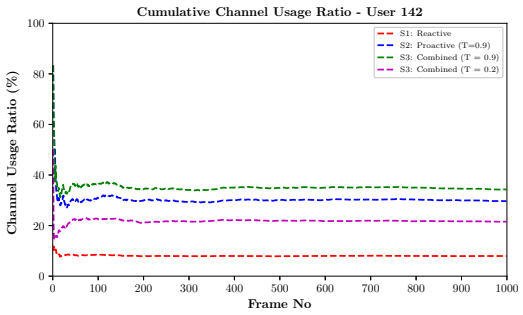
(b) Dataset 1 - $T_h = 1$ s and $T_l = 1$ s



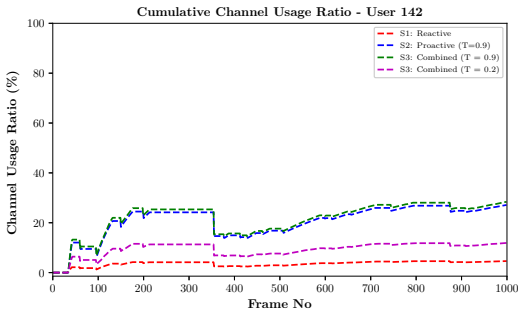
(c) Dataset 2 - $T_h = 20$ ms and $T_l = 20$ ms



(d) Dataset 2 - $T_h = 1$ s and $T_l = 1$ s



(e) Dataset 3 - $T_h = 20$ ms and $T_l = 20$ ms



(f) Dataset 3 - $T_h = 1$ s and $T_l = 1$ s

Figure 5.5: Cumulative Channel Usage Ratio for Different Datasets

It is possible to see that the cumulative channel usage ratio behaviour is very similar for all three datasets in both fast-varying and slow-varying channels. This was already expected as the channel qualities are equal. Since we are changing the way that the FoV probabilities are generated, the expectation for the cumulative QoS is that the longer a user takes to return to the middle FoV, the lower the QoS trace for the proactive strategy will be.

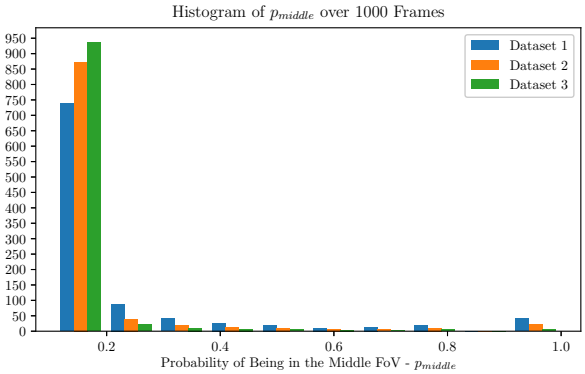


Figure 5.6: Histogram of p_{middle} over 1000 frames for 3 datasets

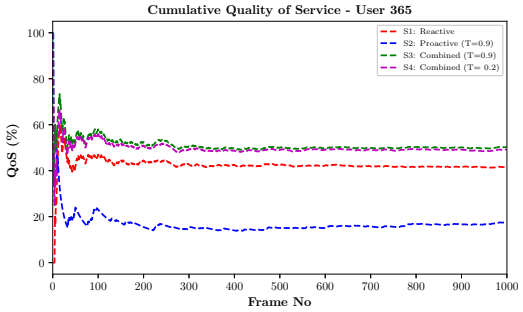
Upon close inspection, this is indeed the case, however, one would think that the difference would be more significant. Since the middle FoV is the only FoV that will

have a chance to have a probability that is higher than 0.2, it would be reasonable to check how the distribution of probability changes over the 1000 frames for the three datasets. The histogram regarding the distribution of probability of being in the middle FoV, p_{middle}, over 1000 frames is provided in Fig. 5.6. It is quite clear from the histogram that the longer it takes to return to the middle FoV, the less and less the p_{middle} exceeds the 0.2 mark. However, the differences between all datasets are quite insignificant, which has also reflected to the results in Fig. 5.4 and 5.5.

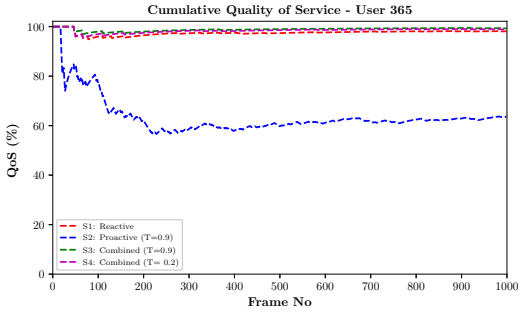
5.3 Different Channel Qualities

In order to compare the strategies in terms of the QoS-efficiency trade-off, the channel qualities are varied using the mean times at staying in respective states T_h and T_l. The channel qualities are created as a combination of either fast or slow varying, and either predominantly low or high channels. In a predominantly high quality channel, the reactive strategy is expected to perform the best in terms of the QoS-efficiency trade-off, since only one timeslot is required to transmit the correct FoV. In a predominantly low quality channel, the combined strategy is expected to perform better in QoS, with a small trade-off in efficiency. The proactive strategy, in its nature, is not a very efficient strategy, as it requires a minimum of one FoV to acquire the correct FoV given that the prediction is correct. Therefore, the combined strategy is expected to outperform the proactive strategy, as there is a margin of recovery with the reactive strategy.

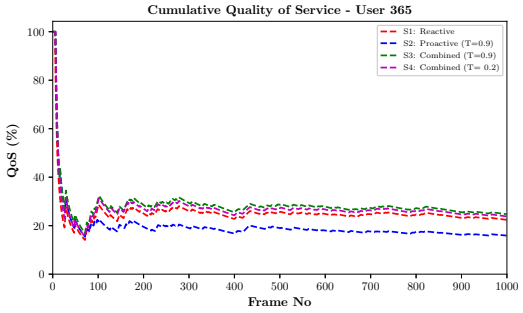
In Fig. 5.7 and Fig. 5.8, the results of the simulation for four different channel qualities is depicted. Looking at both graphs, it is clear that the proactive is the least efficient and has the least QoS in all of the channel scenarios. For the cumulative QoS illustrated in Fig. 5.7, it is clear that in a predominantly high quality channel, regardless of the fast or slow varying feature, the reactive and both of the combined strategies approach the same percentage. This is due to the fact that when the channel quality is high, the added proactive FoVs don't add value to the QoS, but instead result in an overhead in energy as seen in the cumulative channel usage ratio graph in Fig. 5.8. When considering a predominantly low quality channel, the situation is a bit different. For a fast varying and predominantly low quality channel, both of the combined strategies have an added buffer with their respective proactive periods and they outperform the pure reactive and proactive strategies in the cumulative QoS department by 10 percent, with a mere 10 percent addition in cumulative channel usage ratio. For a slow varying and predominantly low quality channel, all of the strategies perform very poorly in terms of cumulative QoS, with a maximum of 30 percent. The cumulative channel usage ratio gave a similar result to the fast varying and predominantly low channel.



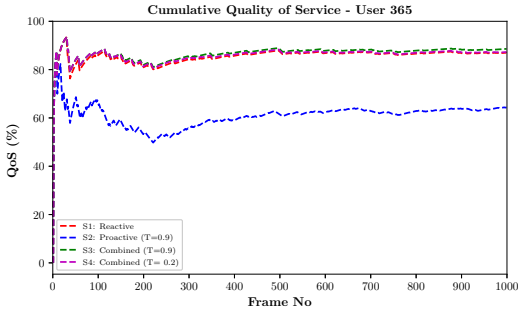
(a) $T_h = 10$ ms and $T_l = 50$ ms



(b) $T_h = 50$ ms and $T_l = 10$ ms

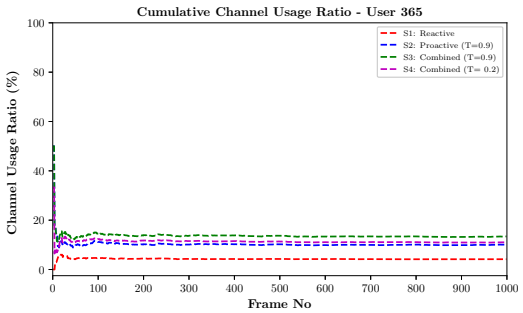


(c) $T_h = 100$ ms and $T_l = 500$ ms

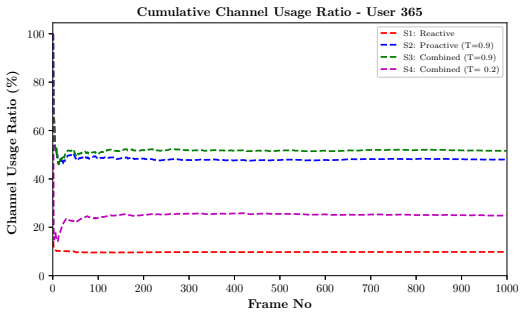


(d) $T_h = 500$ ms and $T_l = 100$ ms

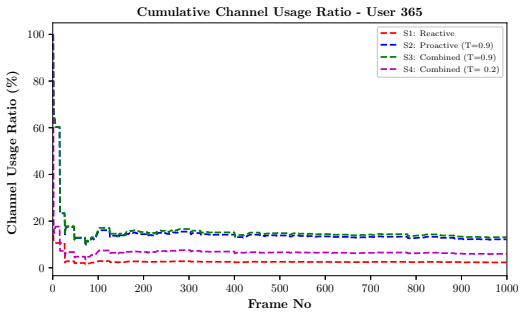
Figure 5.7: Cumulative Quality of Service for Different Channel Qualities



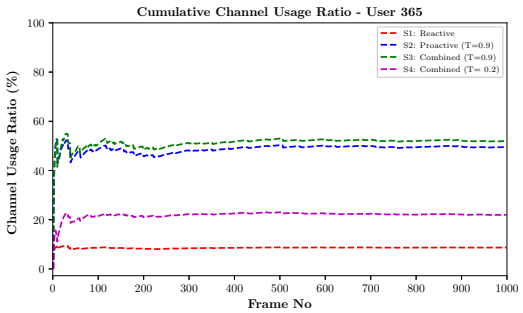
(a) $T_h = 10$ ms and $T_l = 50$ ms



(b) $T_h = 50$ ms and $T_l = 10$ ms



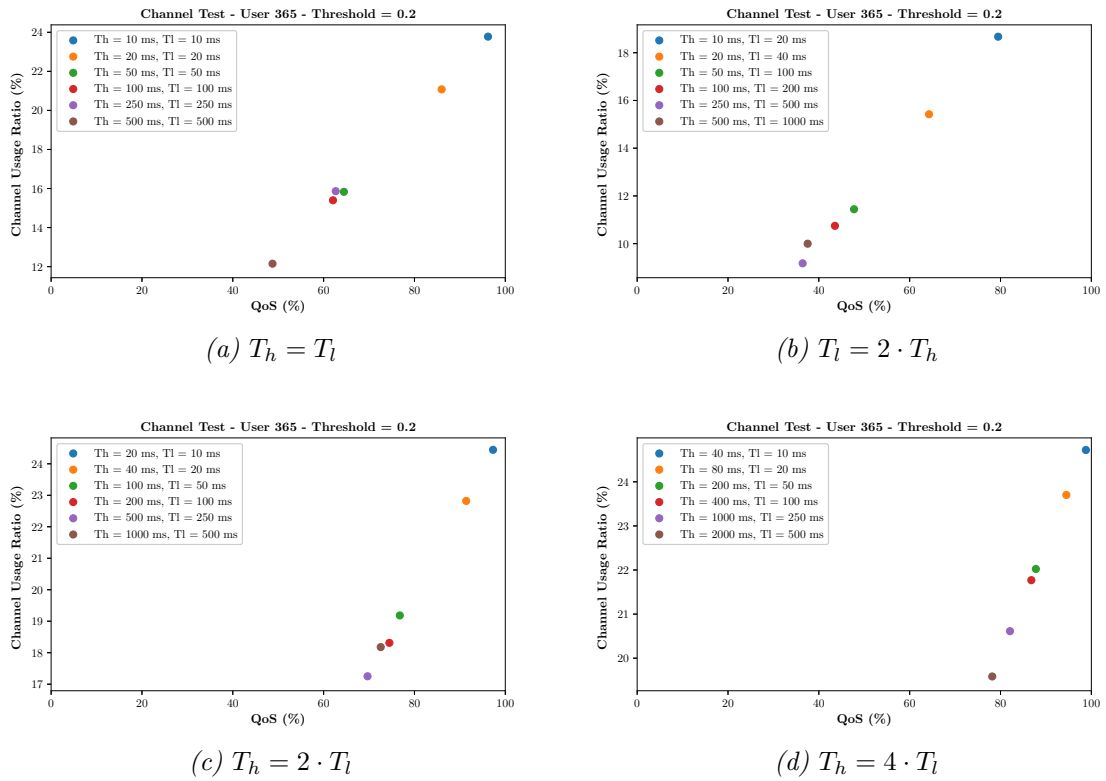
(c) $T_h = 100$ ms and $T_l = 500$ ms



(d) $T_h = 500$ ms and $T_l = 100$ ms

Figure 5.8: Cumulative Channel Usage Ratio for Different Channel Qualities

To investigate the effect of the channel quality on the combined strategy, another test is performed where the ratio between T_h and T_l is held constant but their respective values are increased. The reason to do this is to see the effect of the rate of variation of the channel on the QoS-efficiency trade-off. This can be seen in Fig. 5.9. It is important to note that the x-axis is now the cumulative QoS and the y-axis is the cumulative channel usage ratio. It is natural to expect that a fast varying channel results in a better QoS, as such channel allows to transmit the FoV in the reactive period, most of the time. The results in Fig. 5.9 acknowledges the expectation for all four ratios. Naturally, the higher the ratio between high and low gets, the better the QoS is. However, it is very interesting to see that there is a decreasing linear trend from fast varying to slow varying. When the channel varies slowly, there are longer periods where the channel quality is low, meaning that nothing can be transmitted. The fact that there are longer high quality periods doesn't mean that the QoS will improve, but instead means that more FoVs will be transmitted. Since the combined strategy is considered in this scenario, only one high quality timeslot in the reactive period is sufficient to get the correct FoV.



(a) $T_h = T_l$ *(b)* $T_l = 2 \cdot T_h$

(c) $T_h = 2 \cdot T_l$ *(d)* $T_h = 4 \cdot T_l$

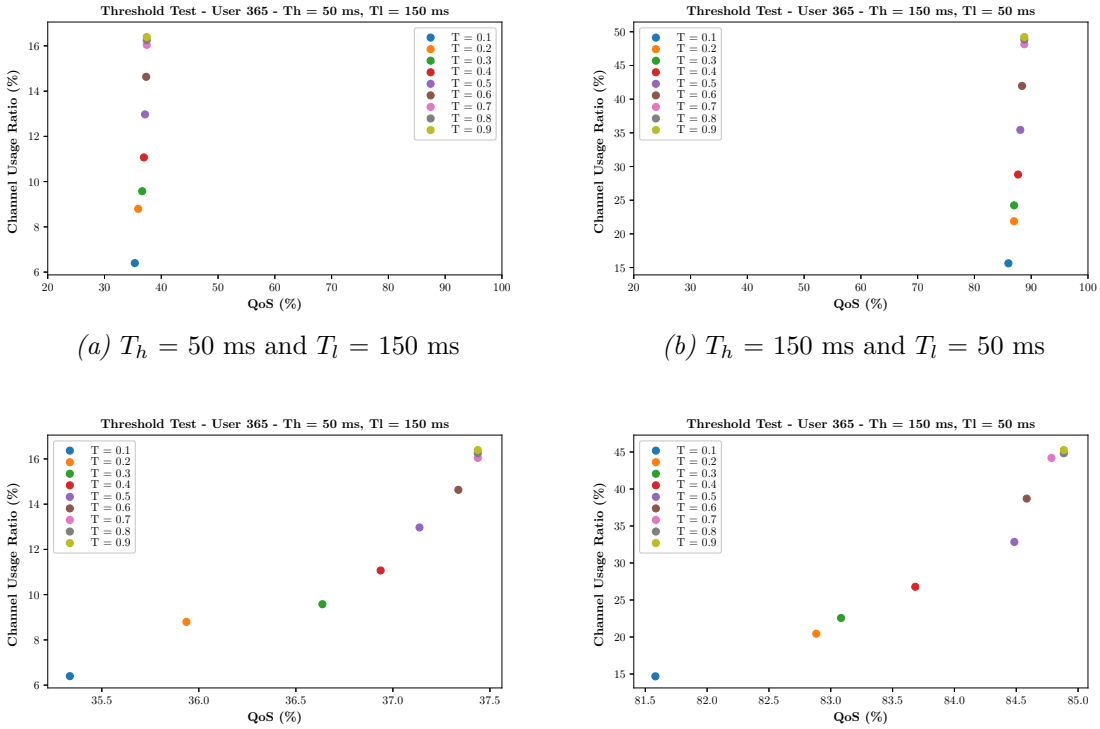
Figure 5.9: Combined Strategy for Four Different Channel Ratios

There are some outliers on the slow varying side of the spectrum, where the higher T_h and T_l results in a better QoS-efficiency trade-off compared to its predecessor with a lower T_h and T_l. This can be attributed to the fact that the T_h and T_l values are

not deterministic but probabilistic. The durations of T_h and T_l are determined by the probabilities p_h and p_l. The higher the p_h and p_l becomes the higher T_h and T_l gets, however consequently the probability to stay in the state longer than T_h and T_l also increases.

5.4 Different Thresholds

When considering the combined strategy, it is reasonable to think that for a predominantly high quality channel, the lower the threshold, the more efficient the transmission becomes. This is due to the fact that, with a lower threshold, the reactive part overtakes the proactive part and less energy is consumed. The higher the threshold gets, the less the efficiency, as more and more FoVs are sent with little to none improvement in quality. However, for a predominantly low quality channel, it is expected that a higher threshold might come in useful with the added buffer as long as the channel allows it.



(a) $T_h = 50$ ms and $T_l = 150$ ms

(b) $T_h = 150$ ms and $T_l = 50$ ms

(c) A closer look to $T_h = 50$ ms and $T_l = 150$ ms

(d) A closer look to $T_h = 150$ ms and $T_l = 50$ ms

Figure 5.10: Threshold Test for Two Different Channels

In Fig. 5.10 (a) and (b), threshold tests for two specific channel quality scenarios are illustrated and in Fig. 5.10 (c) and (d), magnified versions of Fig. 5.10 (a) and (b) are depicted for a closer look. It is quite clear that the threshold has very little impact on the cumulative QoS in both situations. The higher the threshold, the more

the channel is used, with a very little improvement (in the order of 0.5% per threshold step as seen in Fig. 5.10 (c) and (d)) in QoS. This is due to the fact that the first FoV that has been transmitted is the FoV with statistically the highest probability and the rest of the FoVs are essentially an insurance policy given that the 4 timeslots in the reactive period of a frame is on low channel quality.

Chapter 6

Enhanced Delivery Strategies using Channel Prediction

In this chapter, two new delivery strategies that utilize channel prediction will be introduced. It is important to emphasize that we are not going to create a channel prediction algorithm, but rather investigate the potential effect of the channel prediction. To explain the so called enhanced strategies, first, we have to describe the channel prediction. Since a Discrete-Time Markov Chain (DTMC) is created to model the channel quality, the Markov property can be used to perform the channel prediction. In reality, the probability of staying in low and high states are not really known, but for the purpose of this experiment, using this prediction method is sufficient. Given that $p(0)$ is the initial state vector and P is the transition matrix, the probability distribution over states after n-steps in time is,

$$p(n) = p(0) \cdot \mathbf{P^n} = [p(0)_l \; p(0)_h] \cdot \begin{bmatrix} p_l & 1-p_l \\ 1-p_h & p_h \end{bmatrix}^n \tag{6.1}$$

Since the Markov property is memory-less, the expected number of visits to a state is independent of the preceding events,

$$E(N_j | X_0 = j) = \frac{1}{1-p_{jj}} \tag{6.2}$$

where j can be the respective states H and L.

The Markov property also allows to calculate the probabilities for different patterns occurring in the channel quality. This would be useful for the transmission in low quality, as 3 consecutive low quality states are required to transmit a single FoV. The probability of having 3 or more consecutive low quality states can simply be calculated by,

$$\begin{aligned} Pr[\#\ consecutive\ low\ quality \geq 3] &= 1 - Pr[\#\ consecutive\ low\ quality < 3] \\ &= 1 - p_l \cdot (1 - p_l) - p_l^2 \cdot (1 - p_l) \\ &= 1 - p_l + p_l^3 \end{aligned} \tag{6.3}$$

It is clear that in real life conditions, the wireless channel can not be reduced to two levels and the p_l and p_h parameters are not known beforehand. Therefore, it is important to note that the purpose of these experiments are not to develop new channel prediction algorithms, but rather to investigate how channel prediction effects the trade-off between QoS and efficiency on the developed strategies.

6.1 Enhanced Delivery Strategies

6.1.1 Enhanced Delivery Strategy 1

Up until this point, when the channel quality was low, there was no transmission. This is due to the fact that three consecutive low quality channel timeslots are required to transmit one FoV. Not only this is inefficient, but also if there are two consecutive bad quality timeslots followed by a good quality timeslot, the energy consumed will be redundant, as it would have been much more logical to wait for a high quality timeslot. This is when the channel prediction comes into play. Considering the combined strategy as a base for this strategy, if the channel prediction algorithm detects three consecutive timeslots where the channel quality is low, the transmission can be completed, and an extra FoV can be sent rather than just waiting for a good quality timeslot. Using 6.1, the probabilities staying in high or low quality and the number of timeslots that are expected to have high and low quality can be calculated using Eq. 6.2. If it is predicted to achieve the FoV probability threshold given the number of high quality timeslots, nothing will be transmitted. Otherwise, another decision maker to transmit in low quality will be introduced, as given in Eq. 6.3. The probability $Pr[\#\ consecutive\ low\ quality \geq 3]$ has to be higher than a certain threshold, or else there might be a situation where the transmission in low quality starts but can not be completed as there won't be three consecutive low quality timeslots. This threshold will be referred to as p_{thresh} from now on.

$$p_{thresh} < 1 - p_l + p_l^3 \tag{6.4}$$

Since the objective is to make sure that there is a very high probability of having 3 or more consecutive low quality channel timeslots, a choice has been made to take a value

of 0.9. This would give the following relation,

$$0.9 < 1 - p_l + p_l^3$$

This suggests that the probability p_l must be large enough, i.e. the time to stay at the low state must be long enough to satisfy the condition. In a fast varying channel, this strategy will follow the combined strategy, while in a slow varying channel, it will transmit in the low quality state given that the other condition is also satisfied.

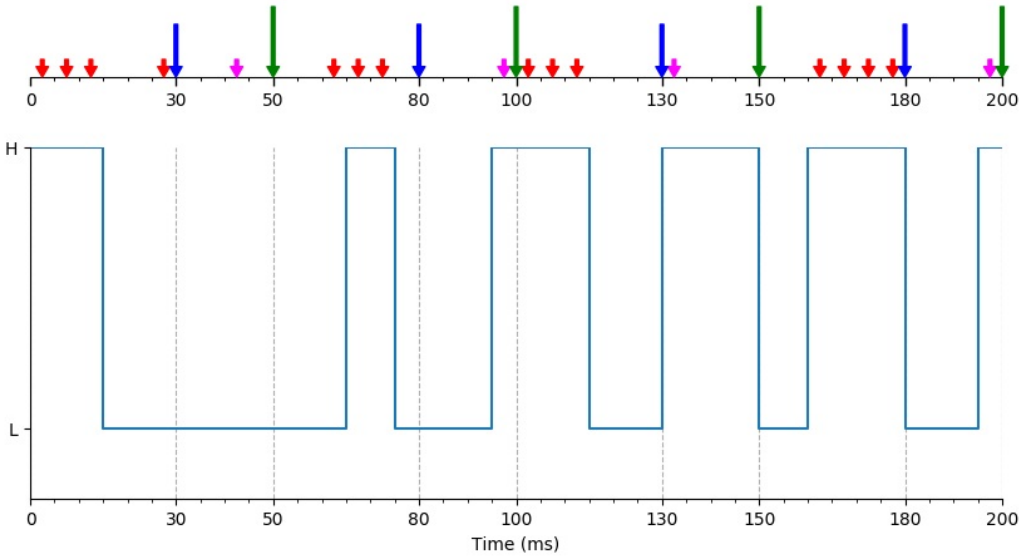


Figure 6.1: Enhanced Delivery Strategy 1

6.1.2 Enhanced Delivery Strategy 2

In the combined strategy, there are 4 timeslots in the reactive period for transmitting a FoV but only one of these timeslots is used to transmit the correct FoV of the current frame. A second way to utilize channel prediction is to start looking ahead into the proactive period of the next frame, in order to start transmission in the reactive period of the current frame, if the FoV probability threshold is not expected to be exceeded. If the channel quality is low for a period of time such that the FoV probability threshold is not surpassed, the MEC can make a decision to start transmitting the FoVs of the next frame in the reactive period of the current frame once the FoV of the current frame is transmitted. The number of high quality timeslots in the reactive period of the current frame and the proactive period of the next frame can again be calculated using Eq. 6.2. If the FoV probability threshold is expected to be surpassed, the strategy will basically behave as the combined strategy.

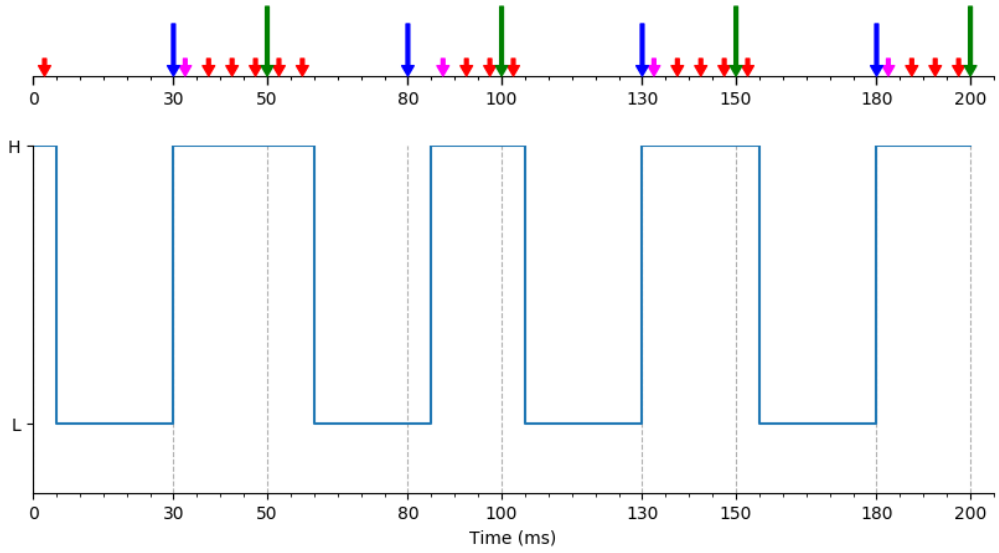


Figure 6.2: Enhanced Delivery Strategy 2

6.2 Evaluation of Enhanced Delivery Strategies

In these tests, the enhanced delivery strategies will be investigated. In Strategy 5, firstly the comparison between transmission in the combined strategy at low channel quality using channel prediction only in the proactive and only in the reactive periods will be tested. The reason to not perform the channel prediction on both the proactive and reactive periods simultaneously is to prevent unnecessary channel usage. In the reactive period of the combined strategy, it is guaranteed to get the correct FoV, regardless of it being transmitted in the low or high quality states.

In Strategy 6, the reactive period of the current frame is used to transmit the FoVs for the proactive period of the next frame (if necessary) given that the channel quality permits. This strategy isn't expected to result in a huge improvement to the combined scenario, since an improvement in the proactive period is considered. Therefore, again, the results are heavily dependent on the FoV prediction accuracy and the channel quality. In Fig. 6.3, the cumulative QoS of all the strategies are depicted for two different channel qualities. One thing to note is that in Fig. 6.3 (a) and (c) channel prediction is only applied in the proactive part for Strategy 5, while in Fig. 6.3 (b) and (d) it is only applied in the reactive parts of Strategy 5. The value of p_{thresh} is considered to be 0.9. In a fast varying channel, i.e. where T_h and T_l are 20 ms, for Strategy 5, it is clear that p_{thresh} is not exceeded for the fast varying channel, as it behaves completely like the combined strategy both in (a) and (b). Looking at Fig. 6.4, this becomes clear, as the results of the combined strategy and the transmission on low quality strategy overlap. Even though Strategy 6 uses more resources, the end

results in terms of the QoS is identical to those of Strategy 5.

When investigating a slow varying channel however, it is clear that Strategy 5 results in better QoS in both only proactive and only reactive. However, it is clear that the only reactive low quality transmission (in the combined strategy) outperforms the only proactive one, as it achieves 100 % QoS with the same percentage of channel usage around 40 to 50 %. In the reactive period either three consecutive low quality channel timeslots or one high quality channel timeslot is sufficient to achieve a successful transmission. For Strategy 6, on the other hand, the QoS is exactly the same as the combined strategy while again using more resources. Because of this, it can be concluded that Strategy 6 is not bringing an improvement in QoS, but instead is resulting in inefficient channel usage.

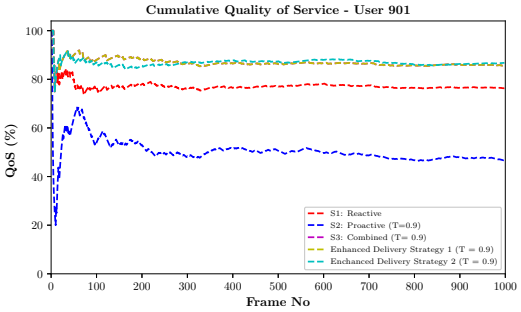
In order to get an overview of the effect of the channel quality on the QoS-efficiency trade-off, the channel ratio test performed in Section 5.2 is repeated for both the only proactive and the only reactive cases, illustrated in Fig. 6.5 and 6.6 respectively. It is clear that for a fast varying channel, the channel prediction is not applied as the p_{thresh} is not exceeded. Once the threshold is exceeded, the trade-off becomes worse for the only proactive period low quality transmission. The channel is used more but the QoS does not improve. For the only reactive period low quality transmission, however, the QoS converges to 100% with a minor increase in channel quality usage. Needless to say, the combined strategy with the only reactive period channel prediction performs the best among all of the aforementioned strategies.

Up until this point, the channel level values were held constant for the high (7 Gbps) and low (3 Gbps) quality channels. For the sake of this experiment, it doesn't seem too necessary to change the high quality level. However, when the low quality transmission with the aid of channel prediction is considered, it seems interesting to reduce the low quality level such that it becomes too low to complete a transmission in the reactive period, i.e. it takes 25 ms to transmit one FoV. This would suggest a minor alteration in the notation of p_{thresh} which was defined in Eq. 6.4, as for low quality transmission 5 consecutive timeslots would be required. p_{thresh}, would now refer to the threshold of the value for $Pr[\#\ consecutive\ low\ quality \geq 5]$:

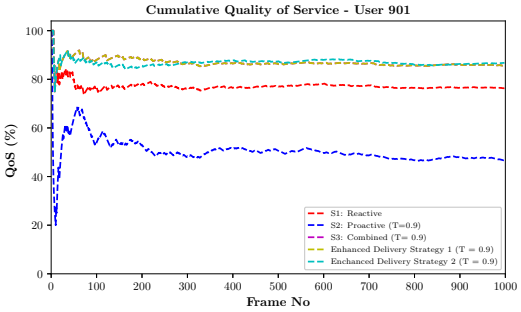
$$\begin{aligned} p_{thresh} &< 1 - p_l + p_l^5 \\ 0.9 &< 1 - p_l + p_l^5 \end{aligned}$$

Since 5 consecutive timeslots with low channel quality are required to transmit a FoV, the low quality transmission in the reactive period is ruled out. The only improvement in QoS could be provided by the added FoV in the proactive period. The QoS-efficiency graph for this scenario is provided in Fig. 6.7. The results of the combined strategy with identical channel conditions are provided in Fig. 6.8 for

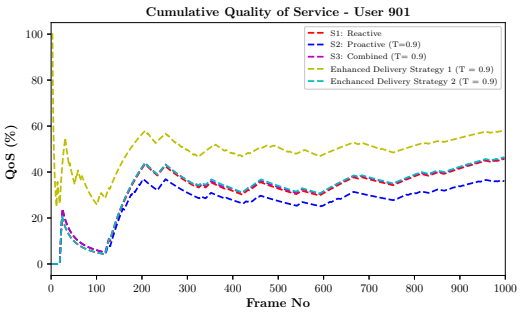
reference. From Fig. 6.7, it is clear that the p_{thresh} is not exceeded for the first three channel conditions indicated in blue, orange and green, as they behave exactly the same as their reference points in the combined scenario depicted in all the subfigures in Fig. 6.8. In Fig. 6.7, there is an observable increase in channel usage ratio for the remainder of the three channel conditions indicated in red, magenta and brown. The QoS for these channels get to the level of the first three channels that do not exceed the threshold. These are linked to the respective lengths of the low quality period, as the lower the quality stays, the more channel resources will be used but higher the QoS gets.



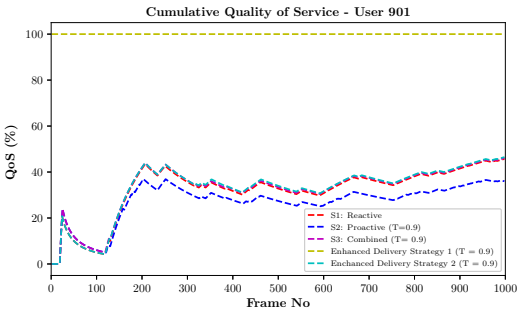
(a) User 901 - $T_h = 20$ ms and $T_l = 20$ ms - Only Proactive Low Quality Transmission with $p_{thresh} = 0.9$



(b) User 901 - $T_h = 20$ ms and $T_l = 20$ ms - Only Reactive Low Quality Transmission with $p_{thresh} = 0.9$

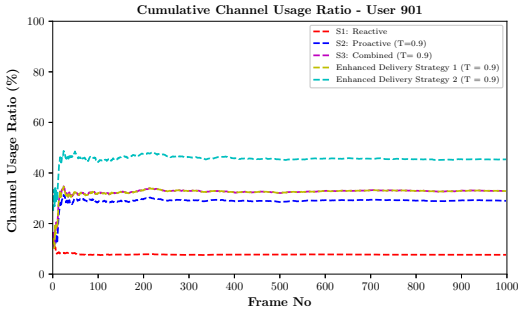


(c) User 901 - $T_h = 1$ s and $T_l = 1$ s - Only Proactive Low Quality Transmission with $p_{thresh} = 0.9$

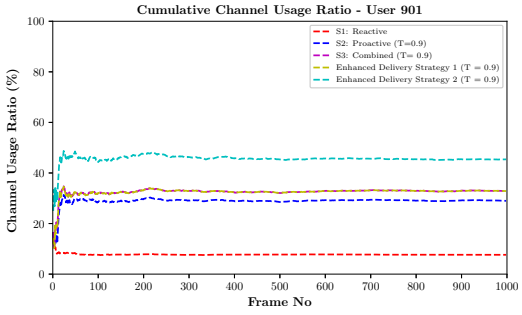


(d) User 901 - $T_h = 1$ s and $T_l = 1$ s - Only Reactive Low Quality Transmission with $p_{thresh} = 0.9$

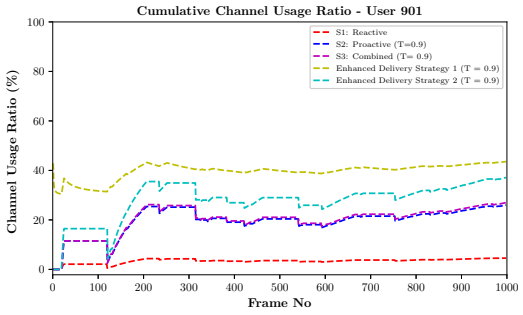
Figure 6.3: Cumulative Quality of Service using Channel Prediction



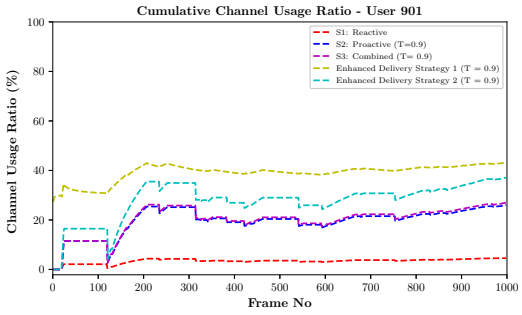
(a) User 901 - $T_h = 20$ ms and $T_l = 20$ ms - Only Proactive Low Quality Transmission with $p_{thresh} = 0.9$



(b) User 901 - $T_h = 20$ ms and $T_l = 20$ ms - Only Reactive Low Quality Transmission with $p_{thresh} = 0.9$

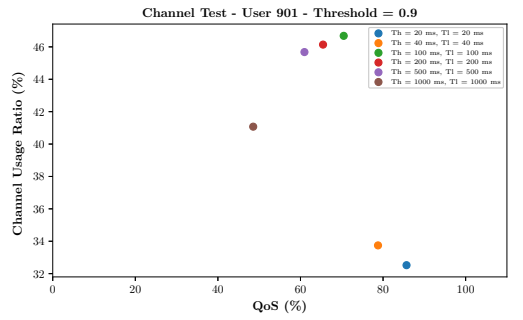


(c) User 901 - $T_h = 1$ s and $T_l = 1$ s - Only Proactive Low Quality Transmission with $p_{thresh} = 0.9$

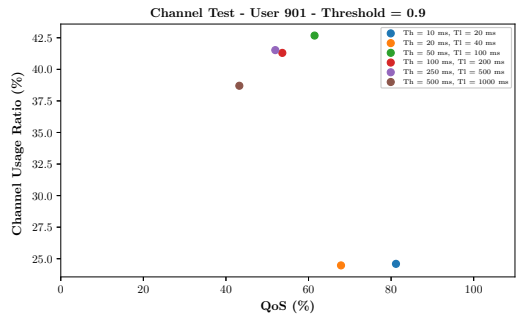


(d) User 901 - $T_h = 1$ s and $T_l = 1$ s - Only Reactive Low Quality Transmission with $p_{thresh} = 0.9$

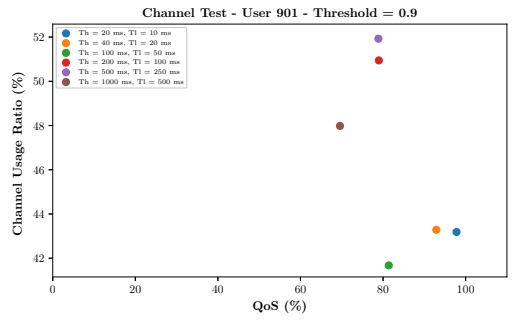
Figure 6.4: Cumulative Channel Usage Ratio using Channel Prediction



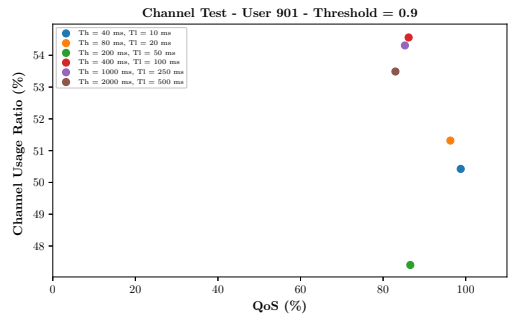
(a) $T_h = T_l$



(b) $T_l = 2 \cdot T_h$

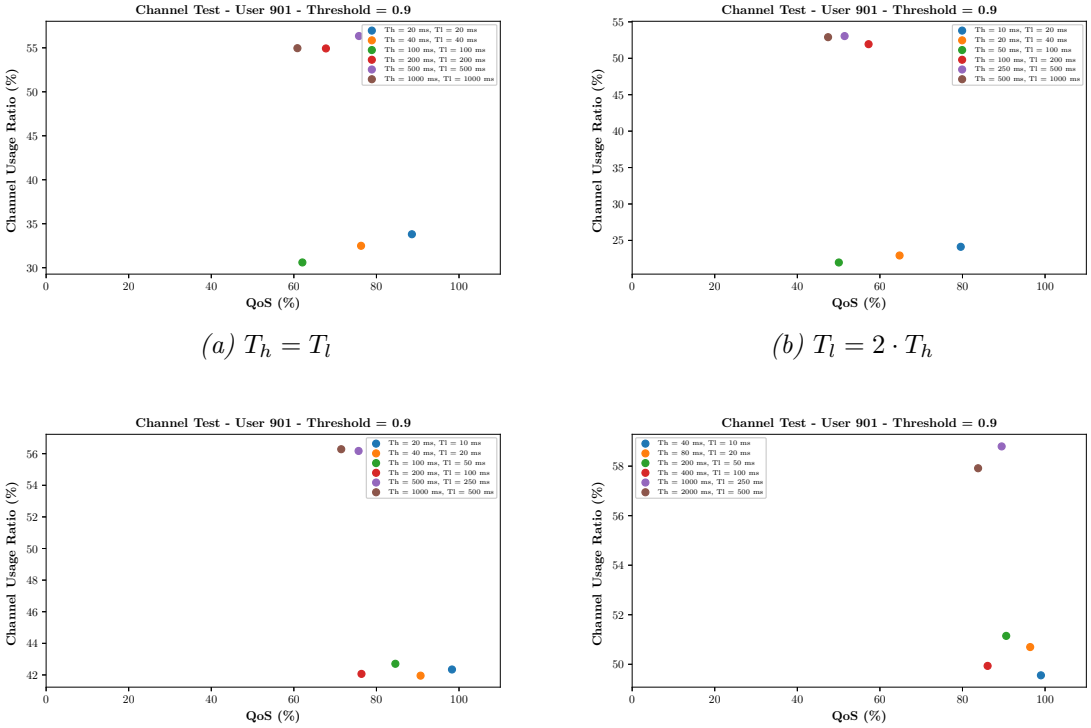


(c) $T_h = 2 \cdot T_l$



(d) $T_h = 4 \cdot T_l$

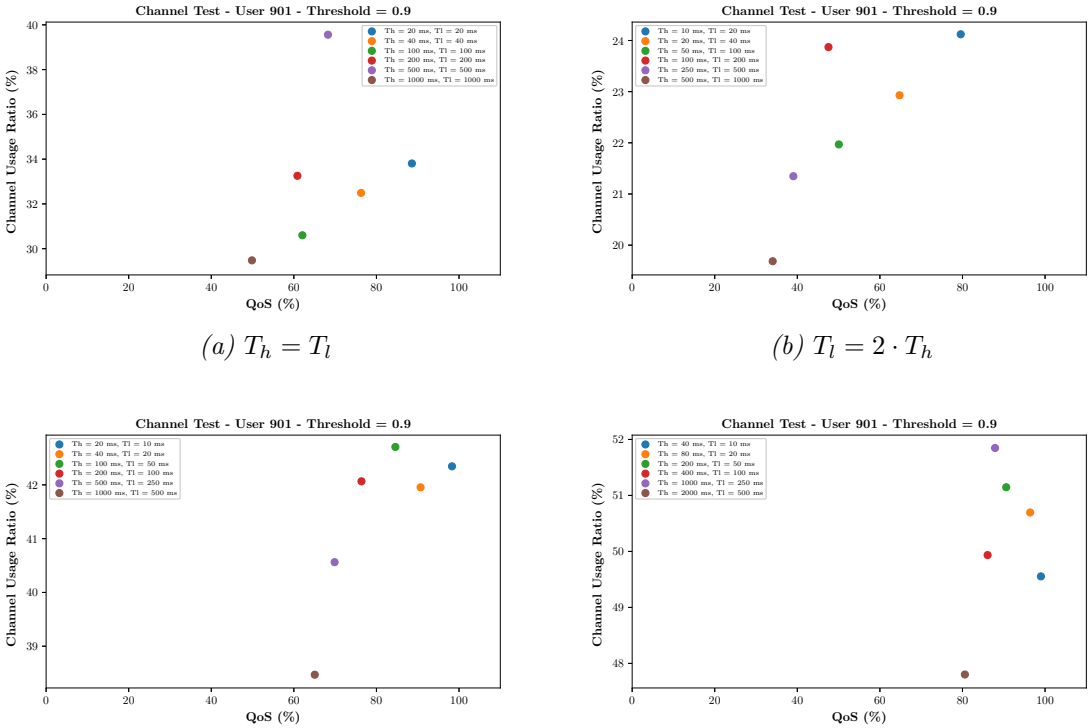
Figure 6.5: Channel Test for Only Proactive Low Quality Transmission - Enhanced Delivery Strategy 1



(a) $T_h = T_l$ *(b)* $T_l = 2 \cdot T_h$

(c) $T_h = 2 \cdot T_l$ *(d)* $T_h = 4 \cdot T_l$

Figure 6.7: Channel Test when Low Quality Level is 1 Gbps - Enhanced Delivery Strategy 1



(a) $T_h = T_l$ *(b)* $T_l = 2 \cdot T_h$

(c) $T_h = 2 \cdot T_l$ *(d)* $T_h = 4 \cdot T_l$

Figure 6.8: Channel Test for Combined Scenario (Reference)

Chapter 7

Conclusion and Recommendations

7.1 Discussion and Conclusion

In this chapter, a general overview of the results will be discussed and the research will be concluded by answering the research questions. In this research, efficient and reliable strategies to deliver VR over 5G is investigated. The purpose of this research was to answer the main research question and subquestions,

- How to deliver non-interactive VR in an efficient and reliable way over 5G radio access network?
    1. How can the MEC improve the quality of service (QoS) in the delivery of non-interactive VR over 5G?
    2. How can motion-based prediction improve efficiency and QoS in the delivery of non-interactive VR over 5G?
    3. How can channel quality prediction help the delivery of non-interactive VR over 5G?

To answer the main research question, three main VR delivery strategies were created, namely the reactive, proactive and the combined strategies. In the reactive strategy, we wait for the user head-movements to refresh and send the correct Field of View (FoV) according to the collected head-movements. In the proactive strategy, we use the probabilities of moving to a certain FoV to send the FoVs before the user head-movement information is refreshed. The FoVs with the highest probabilities are sent to the user, but is limited by a probability threshold which is implemented to prevent using too much energy. The combined strategy is the combination of the aforementioned strategies. Later on the research, two enhanced delivery strategies were considered that included the channel prediction aspect. The delivery strategies were implemented and tested in a Python simulator that is built from scratch with four different simulation scenarios, namely, different users, different video datasets,

different wireless channels and different thresholds. Before presenting the conclusions, it is important to note that the results are closely related to the assumptions and datasets that were used in the simulation. For example, for modelling variations in channel speed we used a two-state Markov Chain. In reality, these variations are much more complex. Therefore, more research is necessary to derive models for these more realistic settings and draw general conclusions.

It has been concluded that the scenario with different users gave similar to identical results for both the QoS and the channel usage ratio, as they were provided equal channel conditions. Upon investigating confidence intervals for the QoS over 1000 users, a conclusion has been made that one user can be representative of the rest of the users. For the scenario with different datasets, three datasets were created where an assumption has been made that the user returns to the middle FoV at different times and with different probabilities. It was clear from the results were quite similar QoS between the 3 datasets. The channel usage ratio was already expected to be identical as the channel conditions were equal for the tests. For different wireless channel qualities, it has been found that the combined strategy is the most reliable option, although the reactive strategy being the most efficient one. When the channel quality is predominantly low and slowly varying, the reactive strategy doesn't have a buffer and suffers from a significant decrease in QoS. For different thresholds in the combined strategy, there is little difference in QoS with a considerable increase in channel usage. The higher the threshold gets, the smaller the increases get in the QoS. Since, wireless VR is the topic of the trade-off question, it would be more reasonable to select a lower threshold, as efficiency is a concern. However, the value of the threshold also depends on how accurate the FoV predictions are. To investigate the effect of Mobile Edge Computing (MEC) on the QoS and efficiency trade-off, motion-based prediction and channel quality prediction is emulated in simple fashion, as the data was not available to the author. In this research, statistical data is created by assuming that the user will mostly stay at the same FoV and the rest of the FoVs have equal but small probabilities. The statistics would suggest that the first FoV that will be extracted from the MEC would be the FoV where the user was looking at in the previous frame and the following FoVs will be random. If there were a more realistic dataset, choosing a higher threshold might make more sense.

There are two enhanced delivery strategies using channel predictions that were considered. The first enhanced delivery strategy uses channel prediction to make a decision to transmit FoVs in low channel quality. The channel prediction for this strategy is either performed in the proactive period or the reactive period. The second enhanced delivery strategy uses channel prediction to look one frame ahead to see if there is a low channel quality period that prevents us to send enough FoVs to reach the threshold and if it is necessary to start building a buffer for the next frame. For

the first enhanced delivery strategy, the distinction between the proactive and reactive periods in the combined strategy were made and the case were the channel prediction is applied for both periods is left out due to efficiency purposes. The expectation was that the reactive period channel prediction would outperform the proactive period channel prediction as the reactive period would get the correct FoV regardless of it being in a low quality or a high quality channel state. The simulations performed for different channel qualities acknowledged the fact that this expectation is fulfilled. From this the question was that if the combined strategy was a necessity, if the reactive strategy with the channel prediction can perform this well. To answer this question, another scenario was created for the low quality transmission using channel prediction to change the channel level of the low quality state, such that the reactive transmission is not possible in low quality. This is a realistic scenario, as the blockages can prevent the mmWave to get up to a certain datarate. The combined scenario on its own already provides a buffer, as opposed to the reactive period. The results show that the extra FoV obtained using channel prediction in the proactive period for low quality transmission results in a significant increase in the QoS. The second enhanced delivery strategy, on the other hand, didn't perform well, which can be attributed to the fact that the datasets were created on a statistical basis. It didn't improve the QoS but rather reduced the efficiency.

With all the conclusions mentioned above, the subquestions of the main research question can be answered. The MEC, by computing the motion-based FoV predictions and the channel predictions, allows to help us determine the best delivery strategy. We used statistical datasets that were created by mimicking a FoV prediction. Although the proactive strategy is not very successful on its own, in the combined scenario it acts as a buffer when there is a low quality period approaching, and results in a significant improvement in the QoS department with a reasonable decrease in efficiency. When the channel prediction comes into play, transmission in low quality becomes possible and considering it being performed only in the reactive period, it results in maximum QoS with reasonable channel usage. To answer the main question, upon considering different simulation scenarios, we determined that the reactive strategy combined with a low threshold proactive strategy is the most reliable and efficient option, considering different quality channels. When combined with the low quality transmission using channel prediction in the reactive period, the performance is even better in terms of QoS with reasonable expense in the channel usage ratio.

7.2 Recommendations for Further Research

This research offers various directions for further research. There are several ways to improve the proposed delivery strategies. Also, it is possible to get more realistic

results, if real user-data is used. At the end of this research, a conclusion can be made that the delivery strategies need to be tested with real data obtained by measuring user movements. Therefore the first recommendation would be to test the simulation with real data. The data we used is generated in a statistical manner. The assumption that was made when creating the dataset, was that the user would stay at the same FoV with a really high probability and move to the other FoVs with equal but small probabilities. After a certain amount of frames (depending on which dataset it belongs to), the probability to look at any frame, converges more or less to the same value. This in return gives a random movement pattern for most users, when they don't stay at the same FoV. A final recommendation would be to use real data in the simulation in order to take the randomness out of the equation and get more realistic results.

The second recommendation would be to improve the proactive strategy (and hence the combined strategy) by first transmitting the FoV where the user previously was looking at and then the FoVs that have the largest probability according to the statistical viewing behaviour. The reason to first transmit the FoV where the user was previously looking at is that the user isn't expected to move a lot within the 50 ms period. This is a way to combine the user-movement information and the statistical viewing behaviour, and is expected to result in better QoS and potentially better efficiency.

Another recommendation would be to vary the length of the proactive and reactive periods, in order to be able to see what happens in a different frame rate scenario. In this study we used 6 timeslots for the proactive period and 4 timeslots for the reactive period to transmit a FoV (if necessary). A conclusion has been made that the proactive strategy on its own is neither efficient nor reliable with the current dataset, so there is no reason not to shorten this period by half or even more. It would be interesting to see what happens to the trade-off in QoS and efficiency for the delivery strategies, if the length of the reactive period is also varied.

The fourth and final recommendation would be to investigate the created strategies in real-time non-interactive VR conditions. Due to time constraints and the unavailability of resources, it was not possible to test the delivery strategies in a real-time non-interactive VR setting. The reactive strategy used in this study would work on a real-time setting, as it doesn't require FoV prediction or statistical datasets to operate. The proactive and combined strategies used in this study, on the other hand, rely on the statistical data to transmit FoVs. In real-time VR, it is not possible to make a complete statistical analysis as the events are broadcasted live. However, there is a way to improve the QoS and efficiency trade-off for a certain amount of users. Transmitting the 360° video in different times to different amount of users can help us create a statistical profile and allow to use the proactive and combined strategies for a certain amount of users.

Bibliography

[1] M. Lisi, ""A Digital "New World": the Big Fusion between Ubiquitous Localization (GNSS), Sensing (IOT) and Communications (5G)"," 10 2016.

[2] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile Edge Computing - A key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[3] "Whitepaper on the VR-Oriented Bearer Network Requirement(2016)," Sep 2016. [Online]. Available: https://www-file.huawei.com/-/media/corporate/pdf/whitepaper/whitepaper-on-the-vr-oriented-bearer-network-requirement-en.pdf

[4] O. Abari, D. Bharadia, A. Duffield, and D. Katabi, "Cutting the Cord in Virtual Reality," *Proceedings of the 15th ACM Workshop on Hot Topics in Networks - HotNets 16*, 2016. [Online]. Available: https://www.acm.org/binaries/content/assets/src/2016/omidabari.pdf

[5] W. Huang, L. Ding, H.-Y. Wei, J.-N. Hwang, Y. Xu, and W. Zhang, "QoE-Oriented Resource Allocation for 360-degree Video Transmission over Heterogeneous Networks," *arXiv preprint arXiv:1803.07789*, 2018. [Online]. Available: https://arxiv.org/pdf/1803.07789.pdf

[6] R. I. T. da Costa Filho, M. C. Luizelli, M. T. Vega, J. van der Hooft, S. Petrangeli, T. Wauters, F. De Turck, and L. P. Gaspary, "Predicting the performance of virtual reality video streaming in mobile networks," in *Proceedings of the 9th ACM Multimedia Systems Conference*. ACM, 2018, pp. 270–283. [Online]. Available: https://biblio.ugent.be/publication/8565900/file/8565902.pdf

[7] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler, "Toward Low-Latency and Ultra-Reliable Virtual Reality," *IEEE Network*, vol. 32, no. 2, p. 78–84, 2018. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8329628&tag=1

[8] S. Mangiante, G. Klas, A. Navon, Z. GuanHua, J. Ran, and M. D. Silva, "VR is on the Edge: How to Deliver 360 Videos in Mobile Networks," in *Proceedings of*

the Workshop on Virtual Reality and Augmented Reality Network. ACM, 2017, pp. 30–35. [Online]. Available: https://dl.acm.org/citation.cfm?id=3097901

[9] Y. Chao Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile Edge Computing A key technology towards 5G," Sep 2015. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf

[10] V. Sitzmann, A. Serrano, A. Pavel, M. Agrawala, D. Gutierrez, B. Masia, and G. Wetzstein, "Saliency in VR: How do people explore virtual environments?" *IEEE transactions on visualization and computer graphics*, vol. 24, no. 4, pp. 1633–1642, 2018. [Online]. Available: http://webdiis.unizar.es/~aserrano/docs/Sitzmann_TVCG2018_VRsaliency.pdf

[11] Y. Bao, H. Wu, T. Zhang, A. A. Ramli, and X. Liu, "Shooting a moving target: Motion-prediction-based transmission for 360-degree videos." in *BigData*, 2016, pp. 1161–1170. [Online]. Available: http://web.cs.ucdavis.edu/~liu/paper/BigData16.pdf

[12] E. Hossain and M. Hasan, "5G cellular: key enabling technologies and research challenges," *IEEE Instrumentation & Measurement Magazine*, vol. 18, no. 3, pp. 11–21, 2015. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7108393

[13] T. E. Bogale and L. B. Le, "Massive MIMO and mmWave for 5G wireless HetNet: Potential benefits and challenges," *IEEE Vehicular Technology Magazine*, vol. 11, no. 1, pp. 64–75, 2016.

[14] "Smart Industry Implementation Agenda 2018-2021," Feb 2018. [Online]. Available: https://www.smartindustry.nl/wp-content/uploads/2018/03/SI-Implementation-Agenda-2018-English.compressed.pdf

[15] I. Ahmed, V. Harjunen, G. Jacucci, E. Hoggan, N. Ravaja, and M. M. Spapé, "Reach out and touch me: effects of four distinct haptic technologies on affective touch in virtual reality," in *Proceedings of the 18th ACM International Conference on Multimodal Interaction*. ACM, 2016, pp. 341–348.

[16] Y. Kim, H. Kim, and Y. O. Kim, "Virtual reality and augmented reality in plastic surgery: a review," *Archives of plastic surgery*, vol. 44, no. 3, p. 179, 2017.

[17] T. S. Rappaport, Y. Xing, G. R. Maccartney, A. F. Molisch, E. Mellios, and J. Zhang, "Overview of Millimeter Wave Communications for Fifth-Generation (5G) Wireless Networks—With a Focus on Propagation Models," *IEEE*

Transactions on Antennas and Propagation, vol. 65, no. 12, p. 6213–6230, 2017. [Online]. Available: https://arxiv.org/pdf/1708.02557.pdf

[18] K. Liu, C. Tao, L. Liu, Y. Lu, T. Zhou, and J. Qiu, "Analysis of Downlink Channel Estimation Based on Parametric Model in Massive MIMO Systems," in *2018 12th International Symposium on Antennas, Propagation and EM Theory (ISAPE)*. IEEE, 2018, pp. 1–4.

[19] Y. Zhao, X. Wang, G. Wang, R. He, Y. Zou, and Z. Zhao, "Channel estimation and throughput evaluation for 5G wireless communication systems in various scenarios on high speed railways," *China Communications*, vol. 15, no. 4, pp. 86–97, 2018.

[20] M. Mehrabi, M. Mohammadkarimi, M. Ardakani, and Y. Jing, "Decision Directed Channel Estimation Based on Deep Neural Network k-step Predictor for MIMO Communications in 5G," *arXiv preprint arXiv:1901.03435*, 2019.

[21] Y. Ban, L. Xie, Z. Xu, X. Zhang, Z. Guo, and Y. Wang, "Cub360: Exploiting cross-users behaviors for viewport prediction in 360 video adaptive streaming," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2018, pp. 1–6.

[22] M. Erol Kantarci and S. Sukhmani, *Caching and Computing at the Edge for Mobile Augmented Reality and Virtual Reality (AR/VR) in 5G*, 01 2018, pp. 169–177.

[23] M. Ji, G. Caire, and A. F. Molisch, "Wireless Device-to-Device Caching Networks: Basic Principles and System Performance," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 1, pp. 176–189, Jan 2016.

[24] C. Wang, Z. Lin, S. Yang, and P. Lin, "Mobile edge computing-enabled channel-aware video streaming for 4G LTE," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, June 2017, pp. 564–569.

[25] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler, "Edge computing meets millimeter-wave enabled VR: Paving the way to cutting the cord," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2018, pp. 1–6.

[26] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7926923

[27] A. Reznik, L. M. C. Murillo, and Y. Fang, "Cloud RAN and MEC: A Perfect Pairing," Feb 2018. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp23_MEC_and_CRAN_ed1_FINAL.pdf

[28] D. Sánchez-Salas, J. L. Cuevas-Riz, and M. Gonzalez-Mendoza, *Wireless Channel Model with Markov Chains Using MATLAB*, 06 2011.

[29] G. Carneiro, "NS-3: Network simulator 3." [Online]. Available: https://www.nsnam.org

[30] S. Luensdorf, Ontje & Scherfke, "SimPy - Discrete event simulation for Python." [Online]. Available: https://simpy.readthedocs.io/

[31] B. R. Haverkort, *Performance of Computer Communication Systems: A Model-Based Approach*. New York, NY, USA: John Wiley & Sons, Inc., 1998.

Appendix A

Appendix: Simulation Code

A.1 Dataset Creation

In this code, the procedure explained in Section 4.3 is emulated and the results have been stored in two separate csv files for the user and the statistical datasets respectively.

```
import numpy as np
import pandas as pd
import random

z= 0;
q = []
while(z <1000):
    q.append(np.random.choice(np.arange(30,170),1))
    z = np.cumsum(q)[-1]
w = np.delete(np.cumsum(q),-1)
a = [[[0,0,0,0,0,0,0,0,0] for i in range(1000)] for r
    in range(1000)]
for i in range(0,1000):
    a[i][0] = [0.025, 0.025, 0.025, 0.025, 0.8,
   0.025, 0.025, 0.025, 0.025]
    for k in range(1,1000):
        if(k in w):
            a[i][k] =
   [0.00125,0.00125,0.00125,0.00125,

   0.99,0.00125,0.00125,0.00125,0.00125];
        else:
            b = int(np.random.choice(9, 1, p=a[i][k
   -1]))
            if(b==0):
                a[i][k][0]=0.8
                a[i][k][1]=0.04
                a[i][k][2]=0.04
                a[i][k][3]=0.04
                a[i][k][4]=0.04
                a[i][k][5]=0.04
                a[i][k][6]=0
                a[i][k][7]=0
                a[i][k][8]=0

            if(b==1):
                a[i][k][0]=0.04
                a[i][k][1]=0.8
```

```
        a[i][k][2]=0.04
        a[i][k][3]=0.04
        a[i][k][4]=0.04
        a[i][k][5]=0.04
        a[i][k][6]=0
        a[i][k][7]=0
        a[i][k][8]=0

    if(b==2):
        a[i][k][0]=0.04
        a[i][k][1]=0.04
        a[i][k][2]=0.8
        a[i][k][3]=0.04
        a[i][k][4]=0.04
        a[i][k][5]=0.04
        a[i][k][6]=0
        a[i][k][7]=0
        a[i][k][8]=0

    if(b==3):
        a[i][k][0]=0.025
        a[i][k][1]=0.025
        a[i][k][2]=0.025
        a[i][k][3]=0.8
        a[i][k][4]=0.025
        a[i][k][5]=0.025
        a[i][k][6]=0.025
        a[i][k][7]=0.025
        a[i][k][8]=0.025
```

```
if(b==4):
        a[i][k][0]=0.025
        a[i][k][1]=0.025
        a[i][k][2]=0.025
        a[i][k][3]=0.025
        a[i][k][4]=0.8
        a[i][k][5]=0.025
        a[i][k][6]=0.025
        a[i][k][7]=0.025
        a[i][k][8]=0.025

if(b==5):
        a[i][k][0]=0.025
        a[i][k][1]=0.025
        a[i][k][2]=0.025
        a[i][k][3]=0.025
        a[i][k][4]=0.025
        a[i][k][5]=0.8
        a[i][k][6]=0.025
        a[i][k][7]=0.025
        a[i][k][8]=0.025

if(b==6):
        a[i][k][0]=0
        a[i][k][1]=0
        a[i][k][2]=0
        a[i][k][3]=0.04
        a[i][k][4]=0.04
        a[i][k][5]=0.04
        a[i][k][6]=0.025
        a[i][k][7]=0.025
        a[i][k][8]=0.8
```

```
                a[i][k][7]=0.04
                a[i][k][8]=0.04

            if(b==7):
                a[i][k][0]=0
                a[i][k][1]=0
                a[i][k][2]=0
                a[i][k][3]=0.04
                a[i][k][4]=0.04
                a[i][k][5]=0.04
                a[i][k][6]=0.04
                a[i][k][7]=0.8
                a[i][k][8]=0.04

            if(b==8):
                a[i][k][0]=0
                a[i][k][1]=0
                a[i][k][2]=0
                a[i][k][3]=0.04
                a[i][k][4]=0.04
                a[i][k][5]=0.04
                a[i][k][6]=0.04
                a[i][k][7]=0.04
                a[i][k][8]=0.8

            else:
                pass

x = [[0 for p in range(1000)] for z in range(1000)]
for i in range(1000):
    for k in range(1000):
        x[i][k] = a[i][k].index(max(a[i][k]))+1

us = pd.DataFrame(x)
stat = [[0,0,0,0,1,0,0,0,0] for i in range(1000)]
for q in range(1,1000):
    if(q in w):
        stat[q] = [0,0,0,0,1,0,0,0,0]
    elif(q in w+1):
        stat[q] = [0.00125,0.00125,0.00125,0.00125,

   0.99,0.00125,0.00125,0.00125,0.00125]
    else:
        sorted1 = us.sort_values(by=q,axis=0)
        stat[q]=(sorted1[q].value_counts(sort=False,
   normalize=True)).tolist()

us.to_csv("New08User.csv",index=False)
statistics = pd.DataFrame(stat)
statistics.to_csv("New08Stat.csv",index=False)
```

A.2 Functions

The functions that are utilized in the simulation are defined in this code. The Markov Chain function is used to create the channel quality dataset. The *idlargest* function is used to sort and store the indexes of the highest probability FoVs for each frame. The indexes of transmitted FoVs and the actual tile the user is look-

ing at is then stored using the *findid* function. Using the *checkid* function the success or failure of the correct FoV transmission can be determined. Finally using the *qos* function the QoS percentage can be calculated.

```
import numpy as np
import pandas as pd
import random

#Markov Chain for Datarate dataset creation
class MarkovChain(object):
    def __init__(self, transition_prob):
        self.transition_prob = transition_prob
        self.states = list(transition_prob.keys())

    def next_state(self, current_state):
        return np.random.choice(
            self.states,
            p=[self.transition_prob[current_state][
   next_state]
               for next_state in self.states]
        )

    def generate_states(self, current_state, no=10):
        future_states = []
        for i in range(no):
            next_state = self.next_state(
   current_state)
            future_states.append(next_state)
            current_state = next_state
        return future_states

#Sort and store index
def idlargest(x,k):
    a = np.sort(x[k])[::-1]
    b = np.argsort(x[k])[::-1]
    t = 0;
    n = 0;
    while(t<0.9):
        t = sum(a[:n]);
        n = n+1;
    return [n-1,b]

#Modified version of idlargest
def idlargestalt(x,k,thres):
    a = np.sort(x[k])[::-1]
    b = np.argsort(x[k])[::-1]
    t = 0;
    n = 0;
    while(t<thres):
        t = sum(a[:n]);
        n = n+1;
    return [n-1,b]

#Find index of predicted and actual tiles
def findid(b,n,u,k):
    pred_index = b[:(n)]+1
    user_index = int(u[k])
    return [pred_index, user_index]
```

```
#Check if actual id in predicted id list
def checkid(pred_index,user_index,succount,failcount)
    :
    if user_index in pred_index:
        succount = succount+1;
    else:
        failcount = failcount +1;
    return[succount,failcount]

#Calculate QoS and print
def qos(failcount,succount):
    qosper = 100* float(succount)/float(failcount+
    succount)
    return qosper
```

A.3 Simulation

Using the datasets and functions created in the previous section, the three basis and two enhanced strategies can be emulated. Using the *dataplot,qosplot,enerplot and datatestplot* functions the results can be plotted.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import simpy
import random
from allfunc import MarkovChain,idlargest,checkid,qos
    ,findid,idlargestalt
```

```
from numpy.linalg import matrix_power

#Reactive Strategy
def reac(env):
    u = 0; #Timeslot Number
    k = 0; #Frame Number
    failcount = 0; #Number of unsuccessful
    transmissions
    succount = 0; #Number of successful transmissions
    chan_usage = 0; #Channel usage counter
    ieff = 0; #Instantenous efficiency
    global chan_quality; #Channel Quality Dataset
    global stat; #Statistical Dataset
    while True:
        while (k<len(stat)):
            u = u+6;
            yield env.timeout(30)
            bad = 0;
            while(bad<4):
                if(chan_quality[u]==7):
                    send = 5;
                    yield env.timeout(send)
                    succount = succount +1
                    chan_usage = chan_usage +1;
                    ieff = 100*(chan_usage/u)
                    qosper = qos(failcount,succount)
                    qos1.append(qosper)
                    eff1.append(ieff)
                    fratit = 3-bad;
                    u = u + fratit +1;
```

```
                    k = k+1;
                    yield env.timeout(fratit*5)
                    break
                else:
                    wait = 5;
                    yield env.timeout(wait)
                    u = u+1;
                    bad = bad+1;
            else:
                failcount = failcount +1
                qosper = qos(failcount,succount)
                qos1.append(qosper)
                eff1.append(ieff)
                k = k+1;
        env.exit(k == len(stat))

#Proactive Strategy
def pro(env):
    u = 0; #Timeslot Number
    k = 0; #Frame Number
    failcount = 0; #Number of unsuccessful
   transmissions
    succount = 0; #Number of successful transmissions
    chan_usage = 0; #Channel usage counter
    ieff = 0; #Instantenous efficiency
    global chan_quality; #Channel Quality Dataset
    global stat; #Statistical Dataset
    global user; #User dataset
    while True:
        while (k<len(stat)):
            [w,q] = idlargest(stat,k);
            framestart = env.now
            z = 0;
            bad =0;
            while(z<w and z+bad<6):
                if(chan_quality[u]==3):
                    wait = 5;
                    yield env.timeout(wait)
                    u = u+1;
                    bad =bad+1;
                else:
                    tile_sent = 5;
                    yield env.timeout(tile_sent)
                    chan_usage = chan_usage+1;
                    ieff = 100*(chan_usage/u)
                    u = u+1;
                    z = z+1;
            else:
                frametimeout = 6-bad-z;
                yield env.timeout(frametimeout*5)
                u =u+frametimeout;
            [pre,use] = findid(q,z,user,k);
            [succount,failcount] = checkid(pre,use,
succount,failcount);
            qosper = qos(failcount,succount)
            qos2.append(qosper)
            eff2.append(ieff)
            yield env.timeout(20)
            u = u+4;
            k = k+1;
```

```
        env.exit(k == len(stat))


#Combined Strategy
def combi(env):
    u = 0; #Timeslot Number
    k = 0; #Frame Number
    failcount = 0; #Number of unsuccessful
   transmissions
    succount = 0; #Number of successful transmissions
    chan_usage = 0; #Channel usage counter
    ieff = 0; #Instantenous efficiency
    global chan_quality; #Channel Quality Dataset
    global stat; #Statistical Dataset
    global user; #User dataset
    global thresh; #Threshold value
    while True:
        while (k<len(stat)):
            [w,q] = idlargestalt(stat,k,thres);
            framestart = env.now
            z = 0;
            bad =0;
            bad1 = 0;
            while(z<w and z+bad<6):
                if(chan_quality[u]==3):
                    wait = 5;
                    yield env.timeout(wait)
                    u = u+1;
                    bad =bad+1;
                else:
                    tile_sent = 5;
                    yield env.timeout(tile_sent)
                    chan_usage = chan_usage+1;
                    ieff = 100*(chan_usage/u)
                    u = u+1;
                    z = z+1;
            else:
                framefin = 6 - z -bad;
                yield env.timeout(framefin*5)
                u =u+framefin;
            [pre,use] = findid(q,z,user,k);
            if use in pre:
                succount = succount+1
                qosper = qos(failcount,succount)
                qos3.append(qosper)
                eff3.append(ieff)
                yield env.timeout(20)
                k = k+1;
                u = u+4;
            else:
                bad1 = 0;
                while(bad1<4):
                    if(chan_quality[u]==7):
                        send = 5;
                        yield env.timeout(send)
                        succount = succount +1
                        chan_usage = chan_usage +1;
                        ieff = 100*(chan_usage/u)
                        pre = np.append(pre,use);
```

```
                        qosper = qos(failcount,
   succount)
                        qos3.append(qosper)
                        eff3.append(ieff)
                        fratit = 3-bad1;
                        u = u + fratit +1;
                        k = k+1;
                        yield env.timeout(fratit*5)
                        break
                    else:
                        wait = 5;
                        yield env.timeout(wait)
                        u = u+1;
                        bad1 = bad1+1;
                else:
                    failcount = failcount +1
                    qosper = qos(failcount,succount)
                    k = k+1;
                    qos3.append(qosper)
                    eff3.append(ieff)
        qostest.append(qosper)
        efftest.append(ieff)
        env.exit(k == len(stat))

#Enhanced Strategy 1
def enhstr1(env):
    u = 0; #Timeslot Number
    k = 0; #Frame Number
    failcount = 0; #Number of unsuccessful
   transmissions
    succount = 0; #Number of successful transmissions
    chan_usage = 0; #Channel usage counter
    ieff = 0; #Instantenous efficiency
    global chan_quality; #Channel Quality Dataset
    global stat; #Statistical Dataset
    global user; #User dataset
    global thresh; #Threshold value
    while True:
        while (k<len(stat)):
            a = np.sort(stat[k])[::-1]
            [w,q] = idlargestalt(stat,k,thres);
            framestart = env.now
            z = 0;
            bad =0;
            bad1 = 0;
            counter = 0;
            #Predictions according to equations in
   Chapter 6, progoodcnt and regoodcnt are the number
    of
            #timeslots where it is expected to be
   good quality for proactive and reactive
   respectively
            if(chan_quality[u]==3 and 4/5>=pll):
                progoodcnt = 5-int(1/(1-pll))
            if(chan_quality[u]==3 and 4/5<pll):
                progoodcnt = 0
            if(chan_quality[u]==7 and phh<=5/6):
                progoodcnt = int(1/(1-phh))
            if(chan_quality[u]==7 and phh>5/6):
                progoodcnt = 6
```

```
            while(z+bad+counter<6 and z<w):
                if(chan_quality[u]==3):
                    if(sum(a[:progoodcnt])<thres and
prothres>pll-np.power(pll,3)):
                        while(chan_quality[u]==3 and
counter <2 and z+bad+counter<5):
                            counter = counter + 1
                            u = u + 1
                            yield env.timeout(5)
                        else:
                            wait = 5;
                            yield env.timeout(wait)
                            bad = bad + 1
                            u = u + 1
                    else:
                        wait = 5;
                        yield env.timeout(wait)
                        u = u+1;
                        bad =bad+1;
                else:
                    tile_sent = 5;
                    yield env.timeout(tile_sent)
                    chan_usage = chan_usage+1;
                    u = u+1;
                    z = z+1;
            else:
                frametimeout = 6-z-bad-counter
                yield env.timeout(frametimeout*5)
                u =u+frametimeout;
                if(counter == 2):
                    z = z+1;
                    chan_usage = chan_usage+3;
                    ieff = 100*(chan_usage/u)
                else:
                    ieff = 100*(chan_usage/u)
            [pre,use] = findid(q,z,user,k);
            counter = 0;
            if(chan_quality[u]==3 and 2/3>=pll):
                regoodcnt = 3-int(1/(1-pll))
            if(chan_quality[u]==3 and 2/3<pll):
                regoodcnt = 0
            if(chan_quality[u]==7 and phh<=3/4):
                regoodcnt = int(1/(1-phh))
            if(chan_quality[u]==7 and phh>3/4):
                regoodcnt = 4
            if use not in pre:
                while(bad1+counter<4):
                    if(chan_quality[u]==7):
                        send = 5;
                        yield env.timeout(send)
                        succount = succount +1
                        chan_usage = chan_usage +1;
                        ieff = 100*(chan_usage/u)
                        pre = np.append(pre,use);
                        qosper = qos(failcount,
succount)
                        chan1.append(qosper)
                        cheff1.append(ieff)
                        fratit = 3-bad1 - counter;
                        u = u + fratit +1;
```

```
                        yield env.timeout(fratit*5)
                        k = k+1
                        break
                    else:
                        if(regoodcnt == 0 and rethres
>pll-np.power(pll,3)):
                            if(bad1<1 and counter<2):
                                counter = counter + 1
                                u = u + 1
                                yield env.timeout(5)
                            else:
                                wait = 5;
                                yield env.timeout(
wait)
                                u = u+1;
                                bad1 =bad1+1
                        else:
                            wait = 5;
                            yield env.timeout(wait)
                            u = u+1;
                            bad1 =bad1+1;
                else:
                    if(counter == 2):
                        pre = np.append(pre,use);
                        succount = succount + 1;
                        qosper = qos(failcount,
succount)
                        chan_usage = chan_usage + 3;
                        ieff = 100*(chan_usage/u)
                        k = k+1;
                        chan1.append(qosper)
                        cheff1.append(ieff)
                    else:
                        failcount = failcount +1
                        qosper = qos(failcount,
succount)
                        k = k+1;
                        chan1.append(qosper)
                        cheff1.append(ieff)
                else:
                    [succount,failcount] = checkid(pre,
use,succount,failcount);
                    qosper = qos(failcount,succount)
                    chan1.append(qosper)
                    cheff1.append(ieff)
                    yield env.timeout(20)
                    u = u+4;
                    k = k+1;
            qostestc.append(qosper)
            efftestc.append(ieff)
            env.exit(k == len(stat))


#Enhanced Strategy 2
def enhstr2(env):
    u = 0; #Timeslot Number
    k = 0; #Frame Number
    failcount = 0; #Number of unsuccessful
   transmissions
    succount = 0; #Number of successful transmissions
```

```
 chan_usage = 0; #Channel usage counter
 ieff = 0; #Instantenous efficiency
 global chan_quality; #Channel Quality Dataset
 global stat; #Statistical Dataset
 global user; #User dataset
 global thresh; #Threshold value
 startp = 0; #Starting position for proactive
strategy, if 0 start from the proactive part, if 1
 start from reactive part of current frame
 while True:
     while (k<len(stat)):
         if (startp == 0):
             react = []
             [w,q] = idlargestalt(stat,k,thres);
         if (startp == 1):
             react = m[:regoodcnt]
             [w,q] = [len(m),m]
         if(chan_quality[u+6]==3 and 2/3>=pll):
             regoodcnt = 3-int(1/(1-pll))
         if(chan_quality[u+6]==3 and 2/3<pll):
             regoodcnt = 0
         if(chan_quality[u+6]==7 and phh<=3/4):
             regoodcnt = int(1/(1-phh))
         if(chan_quality[u+6]==7 and phh>3/4):
             regoodcnt = 4
         if(chan_quality[u+10]==3 and 4/5>=pll):
             progoodcnt = 5-int(1/(1-pll))
         if(chan_quality[u+10]==3 and 4/5<pll):
             progoodcnt = 0
         if(chan_quality[u+10]==7 and phh<=5/6):
             progoodcnt = int(1/(1-phh))
         if(chan_quality[u+10]==7 and phh>5/6):
             progoodcnt = 6
         framestart = env.now
         z = 0; #Transmitted FoV Counter
         bad = 0; #Bad Quality Counter
         bad1 = 0;
         while(z<w and z+bad<6):
             if(chan_quality[u]==3):
                 wait = 5;
                 yield env.timeout(wait)
                 u = u+1;
                 bad = bad+1;
             else:
                 tile_sent = 5;
                 yield env.timeout(tile_sent)
                 chan_usage = chan_usage+1;
                 ieff = 100*(chan_usage/u)
                 u = u+1;
                 z = z+1;
         else:
             framefin = (6 - z -bad);
             yield env.timeout(5*framefin)
             u = u + framefin;
         [pre,use] = findid(q,z,user,k)
         pre = np.append(react,pre)
         if use in pre:
             succount = succount+1
             qosper = qos(failcount,succount)
             if(k==len(stat)-1):
```

```
                    qostest.append(qosper)
                    efftest.append(ieff)
                    env.exit()
                if(regoodcnt == 0):
                    startp = 0;
                    yield env.timeout(20)
                    u = u + 4;
                    chan2.append(qosper)
                    cheff2.append(ieff)
                    k = k + 1;
                else:
                    x = 0;
                    cntr = 0;
                    while(regoodcnt>0 and cntr<4):
                        if(chan_quality[u]==7):
                            x = x + 1;
                            u = u + 1;
                            yield env.timeout(5)
                            cntr = cntr +1;
                            regoodcnt = regoodcnt - 1
                            chan_usage = chan_usage
+1;
                            ieff = 100*(chan_usage/u)
;
                            continue
                        else:
                            cntr = cntr + 1;
                            u = u + 1;
                            yield env.timeout(5)
                            continue
                    a = np.sort(stat[k+1])[::-1]
                    if(sum(a[:progoodcnt])<thres):
                        startp = 1
                        [n, m] = idlargestalt(stat,k
+1,thres);
                        regoodcnt = x;
                        m = m[regoodcnt:n];
                        chan2.append(qosper)
                        cheff2.append(ieff)
                        k = k + 1 ;
                        break
                    else:
                        startp=0
                        chan2.append(qosper)
                        cheff2.append(ieff)
                        k = k + 1 ;
                        break
            else:
                if(k==len(stat)-1):
                    qostest.append(qosper)
                    efftest.append(ieff)
                    env.exit()
                if(regoodcnt == 0):
                    startp = 0;
                    bad1 = 0;
                    while(bad1<4):
                        if(chan_quality[u]==7):
                            succount = succount +1;
                            chan_usage = chan_usage +
1 ;
```

```
                        u = u + 1;
                        yield env.timeout(5)
                        qosper = qos(failcount,
succount)
                        chan2.append(qosper)
                        cheff2.append(ieff)
                        k = k + 1;
                        break
                    else:
                        bad1 = bad1 + 1;
                        u = u + 1;
                        yield env.timeout(5)
                else:
                    failcount = failcount +1
                    qosper = qos(failcount,
succount)
                    chan2.append(qosper)
                    cheff2.append(ieff)
                    k = k + 1;
            else:
                x = 0;
                y = 0;
                reactgood = regoodcnt;
                for y in range(0,4):
                    if(reactgood == regoodcnt):
                        if(chan_quality[u]==7):
                            reactgood = reactgood
 - 1;
                            succount = succount
+1
                            qosper = qos(
failcount,succount)
                            u = u + 1;
                            chan_usage =
chan_usage+1;
                            ieff = 100*(
chan_usage/u);
                            continue
                        else:
                            u = u + 1;
                            wait = 5;
                            yield env.timeout(
wait)
                            continue
                    elif(reactgood>0):
                        if(chan_quality[u]==7):
                            reactgood = reactgood
 - 1;
                            x = x + 1;
                            u = u + 1;
                            chan_usage =
chan_usage+1;
                            ieff = 100*(
chan_usage/u);
                            continue
                        else:
                            u = u + 1;
                            wait = 5;
                            yield env.timeout(
wait)
```

```
                                continue
                            else:
                                u = u + 1;
                                wait = 5;
                                yield env.timeout(wait)
                                continue
                        a = np.sort(stat[k+1])[::-1]
                        if(sum(a[:progoodcnt])<thres):
                            startp = 1
                            [n, m] = idlargestalt(stat,k
    +1,thres);
                            regoodcnt = x;
                            m = m[regoodcnt:n];
                            chan2.append(qosper)
                            cheff2.append(ieff)
                            k = k + 1 ;
                            break
                        else:
                            startp=0
                            chan2.append(qosper)
                            cheff2.append(ieff)
                            k = k + 1 ;
                            break

#Definition of Channel Quality Plot
def dataplot(chan_quality):
    plt.rcParams['figure.figsize'] = [7, 4]
    plt.rc('text', usetex=True)
    plt.rc('font', family='serif')
    pn,rn,sn = str(prothres).partition('.')
    prothresh = str(pn+ rn.replace('.','') + sn)
    wn,un,on = str(rethres).partition('.')
    rethresh = str(wn+ un.replace('.','') + on)

    datx = 5*np.arange(0,len(chan_quality[0:5*th+5*tl
    ]))
    plt.plot(datx,chan_quality[0:5*th+5*tl],drawstyle
    ='steps-post')
    plt.title(r'\textbf{chan_quality per Timeslot}')
    plt.xlabel(r'\textbf{Time (ms)}')
    plt.ylabel(r'\textbf{Channel Quality(Gbps)}')
    plt.xlim(0,5*th+5*tl);
    plt.xticks(np.arange(0,5*th+5*tl+5,step=th/2+tl
    /2))
    plt.yticks([3,7])
    plt.savefig('#Some Directory'+str(strings)+ '
    _User'+ str(a)+'_Th'+str(th)+'_Tl'+str(tl)+'_pt_'+
    str(prothresh)+'_rt_'+str(rethresh)+ '_Datarate.
    eps', format='eps')
    plt.close()

#Definition of Cumulative QoS Plot
def qosplot(qos1,qos2,qos3):
    get_ipython().run_line_magic('matplotlib', '
    inline')
    plt.rcParams['figure.figsize'] = [7, 4]
    plt.rc('text', usetex=True)
    plt.rc('font', family='serif')
    qn,yn,zn = str(thres).partition('.')
    thresh = str(qn+ yn.replace('.','') + zn)
```

```
 pn,rn,sn = str(prothres).partition('.')
 prothresh = str(pn+ rn.replace('.','') + sn)
 wn,un,on = str(rethres).partition('.')
 rethresh = str(wn+ un.replace('.','') + on)

 x11 = np.arange(1,len(qos1)+1)
 scen1, = plt.plot(x11,qos1, 'r--',label='S1:
Reactive')
 scen2, = plt.plot(x11,qos2, 'b--',label='S2:
Proactive (T=0.9)')
 scen4, = plt.plot(x11,qos3, 'm--',label='S3:
Combined (T= %.1f)' %thres)
 chane1, = plt.plot(x11,chan1, 'y--',label='
Enhanced Delivery Strategy 1 (T = %.1f)' %thres)
 chane2, = plt.plot(x11,chan2, 'c--',label='
Enchanced Delivery Strategy 2 (T = %.1f)' %thres)

 plt.title(r'\textbf{Cumulative Quality of Service
 - User %i}' %a,fontsize=11)
 plt.xlabel(r'\textbf{Frame No}',fontsize=11)
 plt.ylabel(r'\textbf{QoS ($\textbf{\%}$)}',
fontsize=11)
 plt.xlim(0,1000)
 plt.xticks(np.arange(0,1100,step=100));
 plt.yticks(np.arange(0,110,step=20));
 plt.legend(handles=[scen1,scen2,scen4,chane1,
chane2], loc='best',prop={'size': 7});
 plt.savefig('#Any_Directory'+str(strings)+ '_User
'+ str(a)+'_Th'+str(th)+'_Tl'+str(tl)+'_QoS' +'
_Thresh_'+str(thresh)+'_pt_'+str(prothresh)+'_rt_'
   +str(rethresh)+'.eps', format='eps')
    plt.close()


#Definition of Cumulative Channel Usage Ratio Plot
def  enerplot(eff1,eff2,eff3):
    get_ipython().run_line_magic('matplotlib', '
   inline')
    plt.rcParams['figure.figsize'] = [7, 4]
    qn,yn,zn = str(thres).partition('.')
    thresh = str(qn+ yn.replace('.','') + zn)

    pn,rn,sn = str(prothres).partition('.')
    prothresh = str(pn+ rn.replace('.','') + sn)
    wn,un,on = str(rethres).partition('.')
    rethresh = str(wn+ un.replace('.','') + on)

    x11 = np.arange(1,len(eff1)+1)
    scen1, = plt.plot(x11,eff1, 'r--',label='S1:
   Reactive')
    scen2, = plt.plot(x11,eff2, 'b--',label='S2:
   Proactive  (T=0.9)')
    scen4, = plt.plot(x11,eff3, 'm--',label='S3:
   Combined  (T= %.1f)' %thres)
    chane1, = plt.plot(x11,cheff1, 'y--',label='
   Enhanced Delivery Strategy 1 (T = %.1f)' %thres)
    chane2, = plt.plot(x11,cheff2, 'c--',label='
   Enhanced Delivery Strategy 2 (T = %.1f)' %thres)
```

```
    plt.title(r'\textbf{Cumulative Channel Usage
   Ratio - User %i}' %a,fontsize=11)
    plt.xlabel(r'\textbf{Frame No}',fontsize=11)
    plt.ylabel(r'\textbf{Channel Usage Ratio ($\
   textbf{\%}$)}',fontsize=11)
    plt.xlim(0,1000)
    plt.xticks(np.arange(1100,step=100));
    plt.yticks(np.arange(0,110,step=20));
    plt.legend(handles=[scen1,scen2,scen4,chane1,
   chane2], loc='best',prop={'size': 7});
    plt.savefig('#Any_Directory'+str(strings)+ '_User
   '+ str(a) +'_Th'+str(th)+'_Tl'+str(tl)+'_Energy'+
   '_Thresh_'+str(thresh)+'_pt_'+str(prothresh)+'_rt_
   '+str(rethresh)+'.eps', format='eps')
    plt.close()

#Definition of Cumulative QoS vs Channel Usage Ratio
   Plot
def datatestplot(qostest,efftest,thres):
    get_ipython().run_line_magic('matplotlib', '
   inline')
    onhigh = round(th/(th+tl),2)
    an,bn,cn = str(onhigh).partition('.')
    pstring = str(an+ bn.replace('.','') + cn)
    qn,yn,zn = str(thres).partition('.')
    thresh = str(qn+ yn.replace('.','') + zn)
    pn,rn,sn = str(prothres).partition('.')
    prothresh = str(pn+ rn.replace('.','') + sn)
    wn,un,on = str(rethres).partition('.')
    rethresh = str(wn+ un.replace('.','') + on)
    for q in range(0,len(Thh)):
        plt.scatter(qostest[q],efftest[q], label ="Th
    = %d ms,   Tl = %d ms" %(Thh[q],Tll[q]))

    plt.rcParams['figure.figsize'] = [7, 4]
    plt.rc('text', usetex=True)
    plt.rc('font', family='serif')
    plt.title(r'\textbf{Channel Test - User %i -
   Threshold = %.1f}' %(a,thres),fontsize=11)
    plt.xlabel(r'\textbf{QoS ($\textbf{\%}$)}',
   fontsize=11)
    plt.ylabel(r'\textbf{Channel Usage Ratio ($\
   textbf{\%}$)}',fontsize=11)
    plt.xlim(0,110)
    plt.xticks(np.arange(0,110,step=20));
    plt.legend(loc="best",prop={'size': 6})
    plt.savefig('#Any_Directory'+str(strings)+'_User_
   '+ str(a)+'_Ratio_'+str(pstring)+'_Thresh_'+str(
   thresh)+'_pt_'+str(prothresh)+'_rt_'+str(rethresh)
   + '.eps', format='eps')
    plt.close()

#Loading Datasets
strings = str('New');
stat = pd.read_csv(str(strings)+'Stat.csv').fillna(0)
stat = np.array(stat.values)
users = pd.read_csv(str(strings)+'User.csv')
users.columns = users.columns.astype(int)
```

```
#Running the simulation
qostest = []
efftest = []
qostestc = []
efftestc = []
Thh =[]
Tll =[]
a = 901
for i in [2,4,10,20,50,100]:
    th = 10*i;
    tl = 10*i;
    Thh= np.append(Thh,th)
    Tll=np.append(Tll,tl)
    phh = 1-(5/th)
    pll = 1-(5/tl)
    transition_prob = {3: {3: pll, 7:1-pll},7: {3:1-
   phh, 7:phh}}
    dr1 = MarkovChain(transition_prob=transition_prob
   )
    current_state = int(np.random.choice(([3,7]),1))
    chan_quality = dr1.generate_states(current_state=
   current_state, no=10000)
    for x in [1,2]:
        if(x==1):
            prothres = 0.1;
            rethres= 0.0;
        if(x==2):
            prothres = 0.0;
            rethres= 0.1;
        global thres;
        thres = 0.9
        user=[]
        qos1 =[]
        qos2 = []
        qos3 = []
        eff1 =[]
        eff2 =[]
        eff3 =[]
        chan1 =[]
        cheff1 = []
        chan2 =[]
        cheff2 = []
        user = np.array(users.loc[a])
        simduration = 1000000
        env = simpy.Environment();
        env.process(pro(env))
        env.process(reac(env))
        env.process(combi(env))
        env.process(enhstr1(env))
        env.process(enhstr2(env))
        env.run(until=simduration)
        qos1 = qos1[:-1]
        eff1 = eff1[:-1]
        qos2 = qos2[:-1]
        eff2 = eff2[:-1]
        qos3 = qos3[:-1]
        eff3 = eff3[:-1]
        chan1 = chan1[:-1]
        cheff1 = cheff1[:-1]
        qosplot(qos1,qos2,qos3)
```

```
    enerplot(eff1,eff2,eff3)
qostest1 = list(qostestc[i] for i in [0, 2, 4, 6,
    8,10])
efftest1 = list(efftestc[i] for i in [0, 2, 4, 6,
    8,10])
qostest2 = list(qostestc[i] for i in [1, 3, 5, 7,
    9,11])
```

```
efftest2 = list(efftestc[i] for i in [1, 3, 5, 7,
    9,11])
prothres = 0.1;
rethres= 0.0;
datatestplot(qostest1,efftest1,thres)
prothres = 0.0;
rethres= 0.1;
datatestplot(qostest2,efftest2,thres)
```