THE CATHOLIC UNIVERSITY OF AMERICA

An Analytical Model and Protocol for Optimizing Quality of Experience in Real-Time Communications

A DISSERTATION

Submitted to the Faculty of the

Department of Electrical Engineering and Computer Science
School of Engineering

Of The Catholic University of America

In Partial Fulfillment of the Requirements

For the Degree

Doctor of Philosophy

By

Kristofer R. Smith

Washington, D.C.

2016

An Analytical Model and Protocol for Optimizing Quality of Experience in Real-Time Communications

Kristofer R. Smith, Ph.D.

Director: Hang Liu, Ph.D.

This dissertation evaluates the analytical correlations between the quantitative quality of service parameters and qualitative quality of experience; and defines the desired quality of experience and realized quality of experience, to aid in optimizing the quality of experience. Next, this dissertation proposes a cloud-enabled wireless access network architecture that implements software defined networking for control and optimization. And lastly, this dissertation evaluates the benefits of the proposed architecture, utilizing the desired quality of experience.

The proliferation of mobile devices and broadband applications has placed tremendous demands on wireless network services. Demands for network accessible multimedia content, especially video, has been growing at a rapid pace. When accessed using mobile devices via wireless or mobile networks, a high demand is placed on these resource constrained dynamic environments. Optimizing the performance of wireless edge networks to ensure a high quality of experience for all connected users requires employing new capabilities on the edge network. This dissertation introduces the

concepts of desired and realized quality of experience, which can be used to normalize the quality that users perceive in order to make more accurate comparisons across a wide range of devices and scenarios.

The trend of combining advanced communications and information technologies has created unprecedented opportunities for innovation in network-centric services. The rapid growth in cloud computing and middlebox deployment is an outcome of such integration. A similar level of success should be expected if this paradigm is adopted by access networks. This dissertation presents a computation-capable and programmable wireless access network architecture to enable more efficient and robust content delivery. The proposed architecture integrates cloud computing technology to support in-network processing and caching, and software defined networking for flexible management and control of network resources. Finally, this dissertation proposes the framework and algorithms for optimizing the quality of experience of multiple video streams in real-time, subject to wireless transmission capacity and in-network computational power constraints. The framework and algorithms address the multiple resource management challenges that arise in exploiting such integration. The evaluation results show the proposed algorithms significantly improve the average quality of experience of wireless users, especially in congested environments.

This dissertation by Kristofer R. Smith fulfills the dissertation requirement for the doctoral degree in Electrical Engineering approved by Hang Liu, Ph.D., as Director, and by Lin-Ching Chang, D.Sc., and Nader Namazi, Ph.D., as Readers.

_____

Hang Liu, Ph.D., Director

_____

Lin-Ching Chang, D.Sc., Reader

_____

Nader Namazi, Ph.D., Reader

Dedication

My dissertation is dedicated in loving memory to my grandparents, Ralph & Madeline Smith, whose regard for education and hard work put me on the path toward higher education. Without the loving encouragement, support, and sacrifice of my parents, Stan & Nora Smith, my education would not have been possible. I thank my beautiful wife, Nina, for all her support, standing by me, and encouraging me throughout this endeavor; my sons, Kameron and Keegan; and all of my family and friends for their support as well.

Acknowledgements

I offer, regrettably posthumously, my profound gratitude to my professor, my advisor, and my friend, Dr. Arozullah; who guided my education for many years. I give my sincere gratitude to Dr. Liu, for picking up where Dr. Arozullah had to leave off, challenging me, supporting me, and advising my work through to completion. I also wish to thank Dr. Namazi and Dr. Chang for their commitment of time and support throughout this endeavor.

"The present is theirs; the future, for which I really worked, is mine." --Nikola Tesla,

Quoted in "A Visit to Nikola Tesla" by Dragislav L. Petković in Politika, April 1927

Table of Contents

Table of Figures

Table of Tables

Chapter 1 Introduction

Each day, more people access greater quantities of Internet based content using mobile devices connected to wireless and mobile networks, which are resource constrained dynamic environments, while higher and higher resolutions of multimedia content require ever greater amounts of bandwidth for transmission. This means the need to optimize the underlying network Quality of Service (QoS) [1] parameters to deliver the optimum Quality of Experience (QoE) [2] for these users is becoming increasingly important. One of the reasons for this, is that although many users have access to high definition (HD) data (e.g., streaming videos) they often lack a component for playing back HD content, typically because either the device is incapable of displaying an HD picture or the underlying network is incapable of streaming at a high enough bit rate. Therefore, when a user requests a higher than useable data rate, it places an extraneous load on the end-to-end network; especially on bandwidth limited and spectrum constrained wireless and mobile networks, where this extra load may also negatively affect other mobile network users. This led to the development of the concept that a user has a Desired Quality of Experience (DeQoE) which becomes the maximum QoE goal for a given scenario, and the Realized Quality of Experience (ReQoE) which is the final QoE a user receives, after establishing their DeQoE. The success of a system can

be measured as the ratio of the ReQoE to the DeQoE, which may not require the best available QoS parameters.

The proliferation of mobile devices and new broadband applications has placed a high demand on mobile and wireless access networks. Multimedia content, especially video traffic, has been growing at a rapid pace. It is anticipated that mobile video traffic will increase 13-fold from 2014 to 2019, reaching 17.4 Exabyte's per month and accounting for over 70% of the world's mobile data traffic by 2019 [3]. However, today's IP networks, fixed Internet core, and wireless access networks, were designed based on the end-to-end principle of IP addresses. To retrieve content, a client must first find the address of the server, and establish a relationship with the server before data transport occurs. Then, data packets are forwarded by IP routers in a best-effort fashion from the server to the client, which means significant delay and loss of data may occur within the network. It is challenging to provide QoS and add new functions to the network layer while maintaining network efficiency.

Recent years have shown a consistent trend in the integration of information technologies and communications technologies. The rapid adoption of cloud computing is an example of such integration. In addition, users and applications increasingly demand new in-network services. Middleboxes are widely deployed in today's networks

to meet different requirements such as proxies, firewalls, intrusion detection systems, wide area network (WAN) optimizers, etc. However, these solutions are often aimed at addressing general Internet problems instead of solving specific problems related to wireless networks or mobile devices. For example, a mobile device may experience varying wireless connectivity due to location, interference, and/or traffic load. The users' responsible content provider may be far away, making it difficult to adapt to dynamic wireless network conditions.

The ability to dynamically program and reconfigure networks is essential to maintain a high QoS and efficient content delivery. Software-defined networking (SDN) and Network Functions Virtualization (NFV), are envisioned to be key technological enablers for meeting these requirements [4][5][6][7]. The ability to decouple the data plane and control plane in SDN, along with its open interface, allows network administrators to have real-time programmable central control of network traffic, and more flexibly and predictably for routing specific traffic flows. In recent years, the SDN paradigm has been increasingly adopted by industry, and many commercial switches and routers now support the OpenFlow standard [4] for programmability. Researchers are developing programmable base stations, access points, and wireless access network infrastructure [8][9][10], which will provide flexibility in controlling radio physical layer, media access

control (MAC) layer, and network layer parameters. SDN is currently only used for controlling data forwarding. However, it seems to be a natural technological evolution to incorporate in-network processing and caching with SDN and NFV into edge networks.

1.1 Purpose

This dissertation proposes a method for evaluating a user's QoE based on that user's unique set of parameters, and the ability of the network to meet what is considered to be that user's DeQoE. In order to accomplish this, an analytical review is conducted to determine how the QoE is linked to the various QoS parameters [11] of the network elements: bandwidth (transmission capacity), input generating content providers (data rate), mobile communication devices (data handling capability), and display devices (resolution and frame rate). Then, this analytical model is to be used to develop and evaluate a protocol for designing such networks to provide the optimum QoE while minimizing the utilization of bandwidth [12] in order to support additional users.

Next, this dissertation presents a computation-capable and programmable wireless access network architecture, called CloudEdge, engineered to provide more efficient and

robust content delivery to end users. CloudEdge turns a wireless access network into a programmable micro-cloud, integrating in-network processing and storage capabilities with wireless access using SDN techniques with an open interface and NFV. Computing and caching nodes can be deployed in an access network, and the various physical resources are then controlled by a unified orchestration platform, to fully exploit the potential of virtualization to enhance user experience and optimize traffic in the network. For example, if a user is watching streaming video on a mobile device, and the wireless link deteriorates due to congestion or mobility, the subsequent video objects would be automatically rerouted to a virtual server instance in the access network that transcodes the video objects to match the current channel conditions. Then, the transcoded video objects are forwarded to the mobile user. This ensures that the mobile user will get the best available QoE given the existing conditions. In addition, the open interface allows a third party to request a slice of resources (computing, storage, and bandwidth) to deploy its own services.

1.2 Contribution and Originality

This dissertation makes two distinct contributions. The first being the development of the quantities defined as the DeQoE and ReQoE, which enables the comparison of QoE with respect to its given scenario. The DeQoE success ratio can then be used to compare how successful disparate systems were in meeting the quality expectations of their users. This is accomplished by defining an initial QoE value based on the quality of the content requested and the quality of content available, and then comparing the final QoE relative to the initial QoE. This is necessary because technology is rapidly advancing and often times either the requested content is higher or lower quality than what is consumable. This makes it nearly impossible to assess and compare differences in QoE. As part of the QoE analysis, queueing theory is used to map the interrelationships between bandwidth and other QoS variables, the resulting equations are used to optimize the QoE. Also, no work was found that gave a mathematical function for directly relating latency to a QoE score; so a latency to QoE mapping function was generated by utilizing data gathered from the ITU E-Model [13] and curve fitting it.

The value in optimizing QoE, even slightly, is potentially exponential. When combined with the rapid growth in Internet enabled mobile devices and the high cost of

mobile data plans, this work should easily result in significant savings for many users, and improved user satisfaction for service providers and content providers. It will result in improved wireless and mobile bandwidth efficiency, which directly relates to spectrum efficiency. The results are achieved in many ways: first, for users with limited data plans; second, for mobile and wireless providers who are consistently building out additional infrastructure and are constrained by their available spectrum. This resulting mathematical protocol will enable implementers to have a better understanding of the impacts quantitative network QoS factors have on the qualitative user-perceived QoE. The model developed, mathematically maps the impacts and interactions of different QoS factors, both on each other, and on the final QoE. The reasoning for basing this protocol on the fundamental network QoS factors and rooting it in queueing theory, was to allow the protocol to be as independent and beneficial as possible, whatever future technology innovations arise.

The second contribution is in wireless and mobile edge computing, where this dissertation proposes a new model to address the limitations that exist because cloud services are distant from wireless and mobile users. To address these limitations, a new model that integrates computing and storage capabilities at the edge of the network, closer to the rapidly growing number of mobile devices, is needed.

The devices in wireless access networks such as routers, switches, base stations, in-network computing servers, etc. provide computing, storage, and networking services. These network devices carry out a substantial amount of data processing, analysis, caching, and control tasks. In the current model, if wireless access networks wanted to take advantage of cloud resources, all the data would have to be routed back and forth over the Internet backbone, between end devices and traditional cloud data centers.

CloudEdge will minimize latency, conserve network bandwidth, and have the ability to make better location based and context aware decisions, as well as alleviating some security and privacy concerns. In recent future Internet architecture projects, such as MobilityFirst, led by Rutgers University [14], routers and other networking devices were equipped with an optional compute plane to support future extensions to the network protocols and in-network services. Edge network computing is also attracting industry interests as can be seen in the recent fog computing initiative by Cisco [15]. However, despite the promises and advantages, the realization of the edge network computing faces many challenges. One challenge addressed in this dissertation is how to enable the multi-dimensional resources (bandwidth and computational power) within a wireless access network to optimize the overall user experience. The proposed method accomplishes this through the combined use of software defined networking, network

functions virtualization, and a micro-cloud server that orchestrates the traffic within the access network.

To fully exploit the benefits of integrating in-network processing, caching, and wireless delivery, the resources in the network (bandwidth, CPU cycles, and storage) need to be carefully managed since they are shared by multiple applications and data flows. Therefore, it was necessary to formulate a protocol to optimize the ReQoE of multiple video streams subject to each user's wireless channel bandwidth and the constraints of the in-network video transcoder. The evaluation results demonstrate the benefits of deploying a CloudEdge network. Although the focus of this dissertation is on wireless access networks with video content delivery as a targeted application, the architecture and techniques developed for this dissertation are expected to be useful in the design of many types of networks and applications (e.g., commercial cellular networks, wired and wireless enterprise networks and home networks) through the integration of computational power, storage, and communications in an SDN-enabled platform. Such a computation-capable and programmable networking paradigm should enable a whole range of yet to be considered applications as well. This work will also have broader impacts on how engineers think about and design small and large network infrastructures.

Chapter 2 Background

This chapter provides technical background information and context applicable to the work performed in the rest of this dissertation. The following subjects will be covered briefly but substantively: Internet Protocol (IP), Quality of Service (QoS), Quality of Experience (QoE), Software Defined Networking (SDN), Network Functions Virtualization (NFV), and Cloud Computing.

2.1 Internet Protocol

Internet Protocol (IP) is used for sending data between networked devices, which have IP addresses assigned to them. IP datagrams are sent over the Internet using either the connection oriented Transmission Control Protocol (TCP) or the connectionless User Datagram Protocol (UDP). Both have many similarities, but also some key differences. TCP is designed for reliable data transfer, it requires a handshake to take place between the sender and the receiver before the actual data transfer begins, and this allows both parties to set several parameters necessary to establish a connection between two end systems [16]. After each data packet is sent, its receipt is acknowledged by the receiver; if a packet is not received, the receiver requests retransmission of the missing packets. TCP provides applications with reliable transport, flow control, and congestion control.

UDP does not have a handshake process, the sending application simply starts sending packets to the intended recipient; and there is no acknowledgement sent back to the sender, so the sender does not know if the intended recipient successfully received the data. Additionally, UDP does not provide any support for flow control or congestion control [16]. UDP is useful for data that is time-sensitive or data that is streamed and immediately consumed, such as Voice or Video Conferencing, while information that must be reliably transmitted and verified upon receipt should be sent using TCP.

2.2 Quality of Service

Quality of Service is a term that is applied to many factors that impact the overall ability of a network to meet the requirements of its users. They are metrics often used in contracts with services providers for measuring the provider's ability to deliver the services that have been promised. The following are the Network QoS Factors that will be considered for calculating the necessary conditions to achieve a high QoE, as well as determining what resources are available, that may be used to optimize the network. Many of the QoE factors impact live media (e.g., VoIP) streams significantly differently than recorded media streams.

1. **Available Bandwidth (Data Rate requested over Available Capacity):** Is defined as the throughput or bits per second available for a new connection [16]. The network bandwidth will be calculated as a ratio of data rate and available capacity. These calculations will be based on Queueing Theory, and if the probability of blocking is high, then the QoE will be negatively impacted due to the requested data service (Voice/Video) having to be scaled to a lower resolution/quality in order to be transmitted within the allotted bandwidth.

2. **End-to-End Transmission Delay (Latency):** Is defined as the time (typically measured in milliseconds) required to transmit an object from the sender to the receiver [16]. This time required for a sender's data packet to be received by the receiver is affected by network congestion, queueing, the number of hops on a network, and the physical distance (due to the speed of light being a limiting factor) between the sender and the receiver. Latency is a major factor for interactive live media: if the transmission delay is long, then QoE will suffer greatly; but in most cases it has a negligible impact on recorded media, because a portion of data is buffered to help improve the QoE if disruptive network

conditions occur, and filling this buffer typically adds seconds of startup delay on top of the milliseconds of latency.

3. **Startup Delay (Buffering time + Transmission Delay):** Is defined as the delay between the time when service is requested and before service consumption begins [17]. That is, the time it takes from the initial request for data until the receipt and display of the data. This delay is most often perceived when attempting to stream stored content that is buffered before display. Startup Delay has a high impact for recorded media: if startup delay is long (measured in seconds), then QoE will suffer. Typically, for live media, this is not a major factor because connections are based on the connectionless UDP.

4. **Packet Delay Variation (i.e., Jitter):** Is defined as the end-to-end delay fluctuation from one pack to the next packet [16]. This can also result in the receiver receiving Packets out-of-order, which then need to be reordered. Packet Delay Variation (PDV) typically occurs when packets traverse different routes from the sender to the receiver, resulting in variations in the arrival time. It has about the same level of impact in both the case of live media and recorded media. For live media, out

of order packets are dropped if they are not received before their scheduled playout window usually less than 100 ms, but a small number of infrequently dropped packets may go unnoticed by most users. Most consumers of recorded media have large buffers that last for seconds to smooth even long PDV's. If sufficiently long enough nearly all out-of-order packets can be received and reordered in time to be utilized.

5. **Packet Loss (Lost/Dropped Packets):** Is defined as a packet that is dropped or discarded at any time along a network path. From a system perspective, it will look like a packet that was transmitted into the network that never arrived at its destination [16]. Packet Loss may be the result of various issues anywhere along the network path from a user's device to the host server. The primary cause being queue overflows, which occur when packets arrive faster than they can be serviced; and the secondary is caused by errors due to poor link quality. Packet Loss has about the same level of impact in both cases. For live media, a small number and low frequency of dropped packets maybe acceptable. For recorded media, a small number of retransmission requests can be used to offset dropped

packets, given a sufficient buffer at the end point; the packets may subsequently be reordered before being utilized, without impacting the QoE.

Packet Loss affects both TCP and UDP connections. In UDP, it appears as lost data; for example, a missing piece of a VoIP call, or missing or garbled frames in a video stream. The impact of Packet Loss on TCP is more complicated because missing packets lead to retransmission requests, which require additional bandwidth and add delay to the overall transmission. The impact of Packet Loss on TCP is further increased by the end-to-end delay. Packet Loss in TCP negatively impacts the overall network throughput. The equation below, often referenced as the Mathis equation [18], calculates how packet loss degrades network performance:

$$Maximum\ TCP\ Rate = \frac{Maximum\ Segment\ Size}{Round\ Trip\ Time} * \frac{1}{\sqrt{Packet\ Loss\ Rate}}$$ [18]

The Maximum Segment Size (MSS) = Maximum Transmission Unit (MTU) - (20 bytes for TCP + 20 bytes IP). Networks commonly implement a minimum and maximum Path MTU of 1500 bytes; this means that, typically, the MSS will equal 1460 bytes. The round

trip time is in milliseconds and the packet loss rate is a percentage. According to the Mathis equation [18] and as shown in Figure 1, if there is a round trip time of 80 ms and 1% packet loss, the maximum TCP throughput that can be achieved, regardless of the link capacity, is 1.46 Mbps.

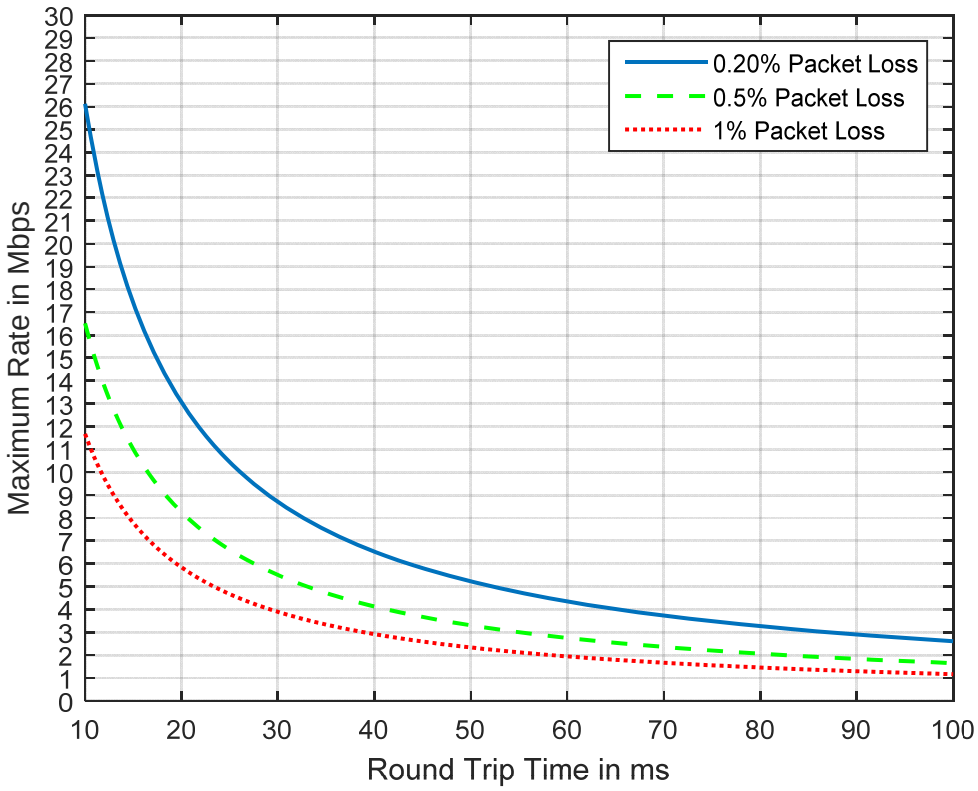

Figure 1: Maximum potential TCP throughput based on Packet Loss

2.3 Quality of Experience

The International Telecommunications Union (ITU) TD109rev2 (PLEN/12) [2], defines QoE as "the overall acceptability of an application or service, as perceived subjectively by the end-user." They note that the "Quality of Experience includes the complete end-to-end system effects (client, terminal, network, services infrastructure, etc)." and that the "overall acceptability may be influenced by user expectations and context."

There are many factors that play a role in forming end-users' perceived quality of experience. The factors that are focused on in this dissertation include network QoS, bandwidth, the available content from a provider, the output/display properties of the users' device, and signal strength (for wirelessly devices).

There are two methods that are commonly used to measure QoE: the Mean Opinion Score (MOS) and the R-Factor. The methods for determining MOSs are specified by ITU-T in recommendation P.800 [16]. Another method for scoring is set forth by the ITU-T in Recommendation G.107 [13], the E-model, which sets the R-Factor between 0 and 100, with higher values indicating higher quality. Based on ITU G.107, a MOS can be calculated given an R-Factor using the following equation [2]:

$$MOS = \begin{cases} R \leq 6.5 & 1 \\ 6.5 \leq R \leq 100 & 1 + .035R + R(R-60)(100-R)7 * 10^{-6} \\ R \leq 100 & 4.5 \end{cases}$$

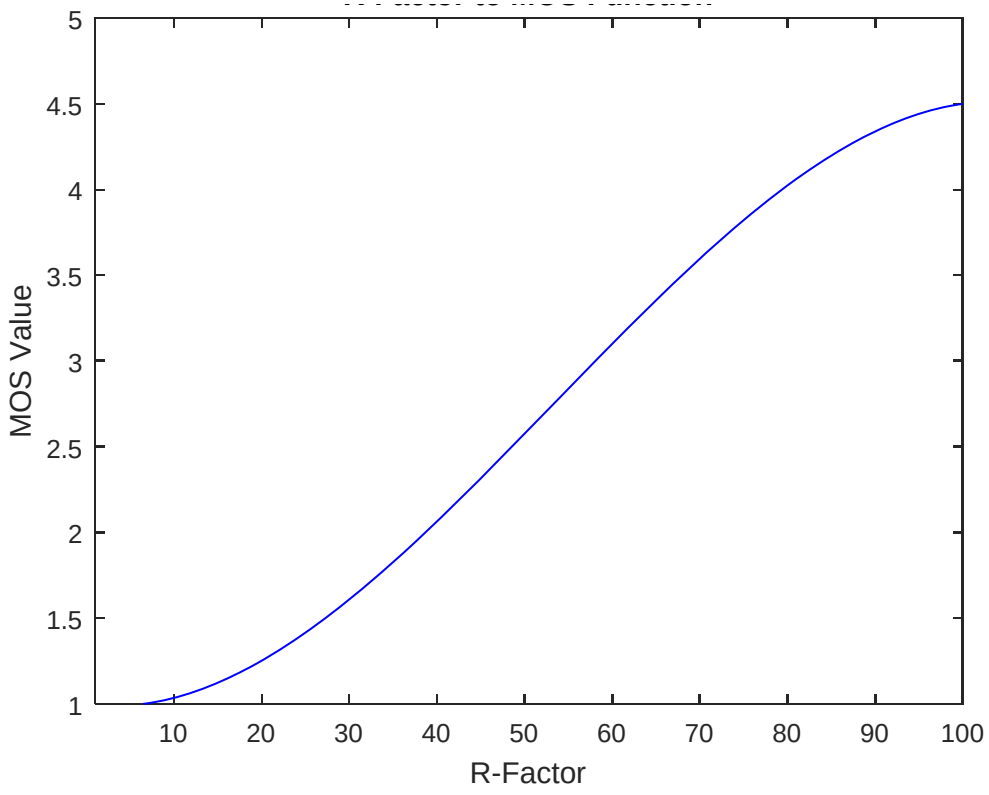

Figure 2: R-Factor to MOS mapping function

The R-Factor can also be calculated given a MOS using the following equation [2]:

$$R = \frac{20}{3}\left(8 - \sqrt{226} * \cos\left(h + \frac{\pi}{3}\right)\right)$$

$$h = \frac{1}{3} arctan2(a, b)$$

$$\boldsymbol{a = (18566 - 6750 * MOS)}$$

$$\boldsymbol{b = \left(15\sqrt{-903522 + 1113960 * MOS - 202500 * MOS^2}\right)}$$

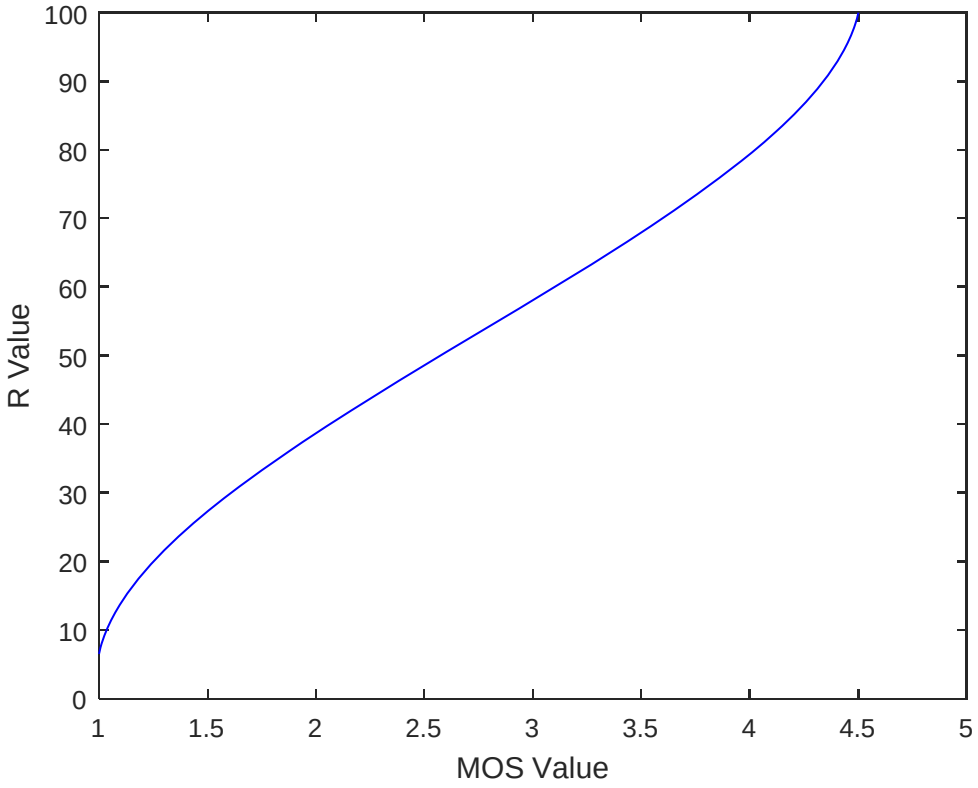

Figure 3: MOS to R-Factor mapping function

Special Note: In MATLAB, this equation is executed as follows. Note that the function ***`atan2`*** is used instead of ***`arctan2;`*** this is how it is implemented in MATLAB, but there

is a critical difference in the operations of these two functions: $\boldsymbol{arctan2(X,Y)} = \boldsymbol{atan2(Y,X)}$. This difference should be noted in the function below:

```
R = (20/3)*((8-sqrt(226)*cos([(1/3)*(atan2(15*sqrt(-
903522+1113960*MOS-202500*(MOS^2)), 18566-6750*MOS))]+(pi/3))))
```

User Rating	MOS	R-Factor
Very satisfied	4.3-4.5	90-100
Satisfied	4.0-4.3	80-90
Some users satisfied	3.6-4.0	70-80
Many users dissatisfied	3.1-3.6	60-70
Nearly all users dissatisfied	2.6-3.1	50-60
Not recommended	1.0-2.6	0-50

Table 1: MOS & R-Factor based on R-Factors given in ITU-T Rec. G.107 Annex B [13]

This dissertation focuses on utilizing the Mean Opinion Score (MOS) for rating QoE. The MOS was developed for categorizing user satisfaction with telecommunication services. When factoring the QoE, bandwidth is a major concern, but when calculating the final QoE, queueing theory is used to translate the impact of insufficient bandwidth into packet loss. Therefore the QoE MOS can be given as the function for mapping QoE to different QoS levels [20].

$$\boldsymbol{QoE = f\ (Latency, Packet\ Delay\ Variaiton, Packet\ Loss)}$$

Table 2 shows how latency, packet delay variation, and packet loss are measured very differently, and gives a basic concept of how each relate to a general MOS.

QoS	Ideal	Normal	Poor
Latency	< 150 ms	150 - 200 ms	> 200 ms
Packet Delay Variation	0 - 20 ms	20 - 50 ms	> 50 ms
Packet Loss	0 - 0.1 %	0.1 - 1 %	> 1%
MOS	5 - 4	3.5 - 4	< 3.5

Table 2: MOS based on QoS network conditions [20]

Mean Opinion Scores were originally standardized for defining the QoE for voice service, but are often used for quantifying multimedia QoE today. The MOS values for each variable have a curve fit equation which will be discussed later that is based upon the results of various qualitative studies. This curve equation can be applied to determine the quantitative MOS value given known network conditions.

2.4 Software Defined Networking and OpenFlow

Software Defined Networking (SDN) [21] provides the ability to separate the control plane and data plane of the network and to centrally manage network devices. The initial focus of SDN was to compliment the wide-scale deployment of virtualization and allow administrators to easily and remotely configure network flows associated with virtualized services. This level of management allows for more efficient management of the control plane and the ability to direct the flow of traffic at the data plane. Because the network state is centrally managed, it can rapidly adapt to changes in network conditions and requirements [21].

OpenFlow [4] is the protocol that provides access to and manipulation of the forwarding plane of a router or switch over the network in SDN enabled devices that are both physical and virtual. Also, OpenFlow enables programmatic control of packet forwarding on a per flow basis [21], and programmable remote control of network traffic manipulation, without the need for physical access to network devices, which themselves may be virtualized. The level of control enabled by OpenFlow allows the network to adapt in real-time to changes that occur at the user, session, or application levels [21].

2.5 Network Functions Virtualization

The European Telecommunications Standards Institute (ETSI) is the standards body responsible for Network Functions Virtualization (NFV). They define the primary differences between deploying physical non-virtualized network appliances and NFV as the ability to remove the requirement for specialized hardware to run specific software, enable the flexible and scalable deployments of network function, and allow for dynamic operations [22].

NFV enables cloud infrastructure to support network appliances, just as the cloud supports the virtualization of servers. Virtualization and cloud computing now afford network appliances and network services: the ability to be hardware agnostic, rapid scalability, remote management and control, and in many instances high availability and continuity of operations in the event of a software or hardware failure.

Because NFV supports the dynamic creation of scalable virtual network appliances, just like cloud servers, each virtual network appliance can be sized to meet immediate requirements and reconfigured based on service demands. Because NFV appliances are hardware agnostic, when a service is no longer needed, virtual appliances can be shut down and their physical resources can be reallocated as needed.

2.6 Cloud Computing

The National Institute of Standards and Technology (NIST) defines cloud computing in NIST SP800-145 [23] as being "ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."[23] The NIST definition defines, in detail, five essential characteristics, three service models, and four deployment models. The characteristics it defines are: on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. The service models given are: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). And finally, the deployment models given are: private cloud, community cloud, public cloud, and hybrid cloud [23].

One of the key business drivers for utilizing cloud resources is the fact that the physical servers can be virtualized and run on shared consolidated computational resources that can be quickly reallocated based on current service requirements. The pools of resources are often very large and can scale to meet nearly any demand. Virtual instances are often routinely backed up so that, if a failure occurs, a new virtual instance

can be brought online quickly to quickly restore lost services. NFV relies on cloud resources, and exists in part due to the fact that many virtual servers running on a single platform desire to utilize network resources that they would otherwise have to traverse the network to use. Enabling clouds with virtualized network functions can greatly reduce the number of physical ports required within data centers, and greatly reduce intra-datacenter latency. SDN works hand in hand with cloud computing. Although cloud resources are not required for enabling SDN, SDN offers tremendous benefits for cloud implementations. This is because virtualized services in cloud environments can be enabled or disabled quickly in software; but without SDN, their network connections may require physical changes to the network before they can provide their desired function.

Chapter 3 Calculating the Quality of Experience

This chapter considers some of the parameters that may impact a user's QoE, this work is primarily focused on wireless and mobile device users. It provides detailed methods for calculating the QoE based on QoS parameters. This chapter also explores how to utilize network queueing theory to calculate the interdependencies of several of the QoS variables and the impact they have on the QoE. The first step will be to analyze and map the quantitative QoS parameters to the qualitative QoE [24] so that an accurate model can be constructed. These include the QoS parameters from the content provider all the way to the end user [25].

There are many factors that contribute to forming end-users' perceived QoE. Table 3 shows some of the potential variables that may impact a mobile device user's QoE.

Variables	Impacts on QoE
Device Resolution	Defines the maximum displayable resolution
Content Resolution	Defines the maximum resolution a user may receive
End-to-End Bandwidth	Affects the maximum resolution that can be streamed
Available Power	Defines the maximum run time
Wireless Signal Strength	Affects the bandwidth and battery life
Data Consumption Rate	Cost of cellular data and battery power
Buffer Size	Helps to compensate for minor network QoS issues Increases startup delay to fill and excess bandwidth to refill

Table 3: Variables that may impact the QoE of mobile device users

Figure 4 provides a generalized scale of how changes in the QoS impact the perceived QoE. This graph is broken in to three discrete regions, each of which represents a different QoE performance range. The final QoE value is based on the network conditions as a function of QoS disturbance, $\boldsymbol{x}$, and is separated into three distinct categories using the thresholds $\boldsymbol{a}$ and $\boldsymbol{b}$ [24].

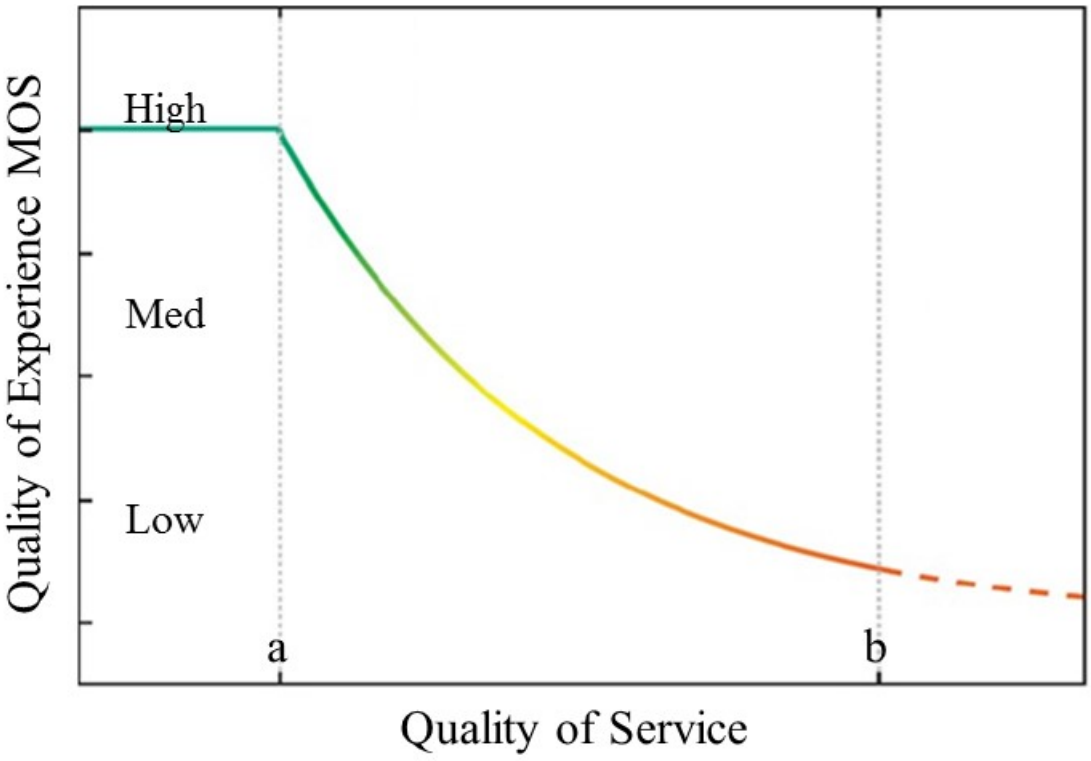

Figure 4: Representation of the relationship of QoS to QoE based on [24]

$\boldsymbol{If\ 0 < x < b} == \mathbf{Excellent\ QoE}$ — A small QoS disturbance (up until $\boldsymbol{a}$ will not affect the QoE at all. (e.g., a video is delayed up to quarter of a second; even if it loaded sooner, the QoE of the average user would not change) [24].

$\boldsymbol{If\ a < x < b == \textbf{Degrading QoE}}$ — When a QoS disturbance exceeds that of $\boldsymbol{a}$, the ideal QoE can no longer be provided to the user, and as the disturbance(s) in QoS increases, the QoE continues to drop [24].

$\boldsymbol{If\ x > b == \textbf{Failed QoE}}$ — When QoS disturbance reaches $\boldsymbol{b}$, the quality is no longer acceptable [24].

Because QoE is influenced by many factors, including itself [24], in this dissertation, the following factors will be considered: the users' device capabilities and network connection, the carriers' network connection, and the providers' content and network connection. For this work, network connection will be defined as bandwidth, latency, packet delay variation, and packet loss.

3.1 Review of Literature: Quality of Experience

The Third Generation Partnership Project (3GPP) has defined several QoE metrics [26] for different scenarios, and the International Telecommunications Union (ITU) has released a formal definition of QoE [2]. Some subjective evaluations of QoE for streaming

video [25] have been performed. Overall, most work is limited in scope and only consider the impact of a single Quality of Service (QoS) parameters at a time. Much of the work related to QoE is focused on voice traffic rather than multimedia traffic, and does not take into account the interdependencies of the QoS parameters [27]. One work discusses the subject of QoE Hysteresis [28], which provides interesting insights into the complexity and depth of analyzing QoE. This work differs because what has been accomplished so far does not take into account the concept of a calculable DeQoE and ReQoE, which are proposed, in order to establish a goal for a system's QoE to attempt to achieve, based on user device capabilities and context. The DeQoE becomes a base line for comparison of disparate scenarios and may potentially enables the conservation of network resources and support of additional users within congested environments.

3.2 Methodology

The approach taken to calculate the QoE is broken down in to several parts, each of which will be discussed in detail in the subsequent sections. First, a method for mapping the impact of the network QoS parameters and device parameters to QoE scores is established. This describes the relationship between each network QoS parameter and

how the QoE score is related to it. The qualitative QoE parameters will be closely related to analog or fuzzy metrics, while the QoS parameters are discrete quantitative metrics. Next a short discussion is presented on the concept of hysteresis and how it may impact a user's perceived QoE. Then, queueing theory is used to calculate the interdependencies of bandwidth and buffers on the QoS parameters; and how they, along with the QoS factors, impact the final QoE. Next, the concepts of the DeQoE and ReQoE are defined. Then their purpose and potential benefits are presented.

Further, the basis of the model is described; including how both live and recorded streaming models are affected by the QoS variables, and how in some cases the QoE can be greatly improved given each model's particular requirements. Finally, a detailed review of the initial calculations and considerations necessary to establish the baseline requirements and QoE of a given system is presented. These are the initial calculations that the protocol performs in order to begin its evaluation of the known system. They are used to generate a QoE goal and show how the final QoE value is determined, based on the numerous network QoS variables and the device and content constraints. As part of the model, a graphical flow chart of the protocol and considerations is displayed. This serves as a visual representation of the calculations that must take place to evaluate a system and determine how to best optimize the QoE given the initial conditions.

The reasoning for basing this protocol on the fundamental network QoS parameters and queueing theory is to create an analytical protocol that is capable of modeling a very complex environment in a general and flexible manner so that it may be as technology independent as possible and remain applicable as future technology innovations arise.

3.3 Mapping Quantitative QoS to Qualitative QoE

The following formulas will be the basis for relating QoE to QoS and building a dynamic model that will be able to relate all of the QoS factors to provide a single QoE of a given end-to-end system. The expectation is that this model will be applicable to all network communication systems, as it takes in to account all of the system properties that affect current and future data communication systems.

3.3.1 Startup Delay

In [29], the authors collected user feedback and generated a function for the impact of startup delay on QoE. The curve of the resulting function gives the probability that the user will find the startup delay to be unacceptable:

$$\frac{e^{-2.4437+0.0626*Delay}}{1+e^{-2.4437+0.0626*Delay}} \quad [29]$$

This equation was inverted and scaled so that could be correlated to diminishing QoE over time to produce eq. 3.1; the results of which are shown in Figure 5.

$$\mathbf{100} * \left[\mathbf{1} - \frac{e^{-2.4437+0.0626*Delay}}{1+e^{-2.4437+0.0626*Delay}}\right] \quad \text{eq. 3.1}$$

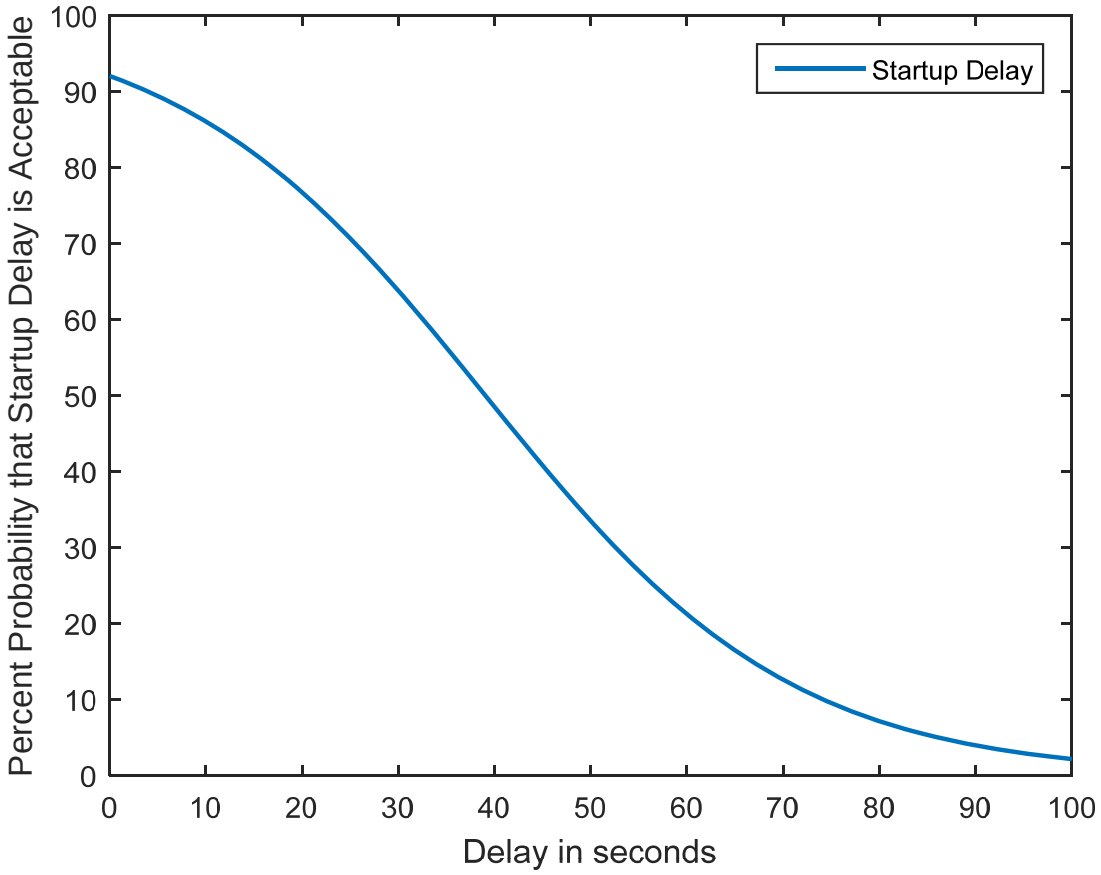

Figure 5: Percent probability that Startup Delay is acceptable

3.3.2 Latency

Upon review of the literature, no work was found that defined an analytical formula for mapping Latency to QoE. Therefore, a function based on the output of the ITU E-Model [13] was constructed. In order to build an accurate latency function, the data points collected from the ITU E-Model, shown in Table 4, was used. This data was generated by checking the values for ***Ta,*** which is defined as the mouth to ear delay for a voice call. The values were checked in 10 ms increments from 150 ms to 500 ms, because all values below 150 ms the resulted in an R-Factor of 93, and MOS of 4.41.

***Ta* (ms)**	**R-Factor**	**MOS**
0-150	93	4.41
160	92.8	4.4
170	92.4	4.39
180	91.9	4.38
190	91.1	4.36
200	90.2	4.34
210	89.1	4.32
220	87.9	4.29
230	86.7	4.25
240	85.5	4.21
250	84.3	4.18
260	83.1	4.13
270	81.9	4.09

280	80.7	4.05
290	79.6	4.01
300	78.4	3.96
310	77.4	3.92
320	76.3	3.88
330	75.3	3.84
340	74.3	3.79
350	73.4	3.75
360	72.5	3.71
370	71.6	3.67
380	70.8	3.63
390	69.9	3.59
400	69.1	3.56
425	67.3	3.47
450	65.6	3.38
475	64	3.3
500	62.6	3.23

Table 4: Latency data points collected from the ITU E-model

Given this data set, in order to design a function for the Latency, or mouth to ear delay, it was first necessary to exclude the values lower than 150 ms, which greatly improved the accuracy of the polynomial curve. The simplest curve fit function with a very degree of accuracy was the 4th degree polynomial function. The result is shown in eq. 3.2 which resulted in a Norm of Residuals value of 0.020317.

$$\boldsymbol{MOS = -1.11e-10 * L^4 + 1.75e-07 * L^3 - 9.825e-05 * L^2 + 0.0193 * L + 3.18}$$

eq. 3.2

Figure 6 illustrates the work described above, including a plot of the data points extracted from the E-Model, a plot of the 4th degree polynomial. Figure 7 is the plot of the Norm of Residuals.

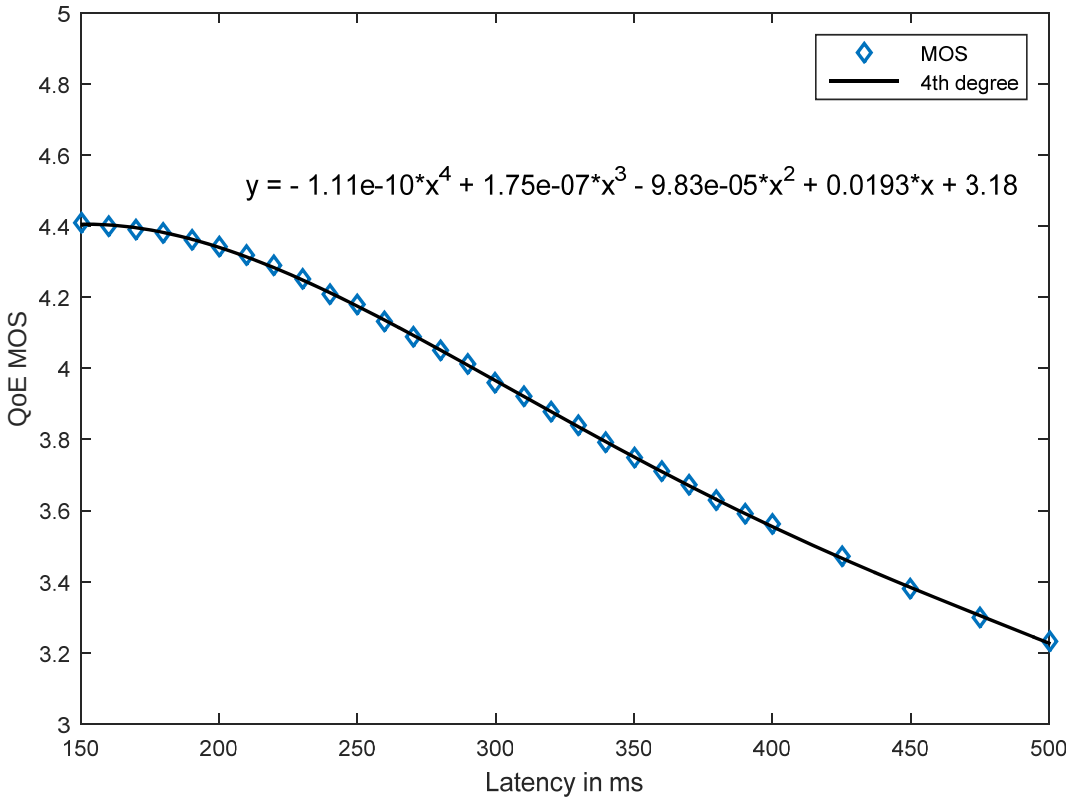

Figure 6: Curve fit of Latency data set to 4th degree polynomial function

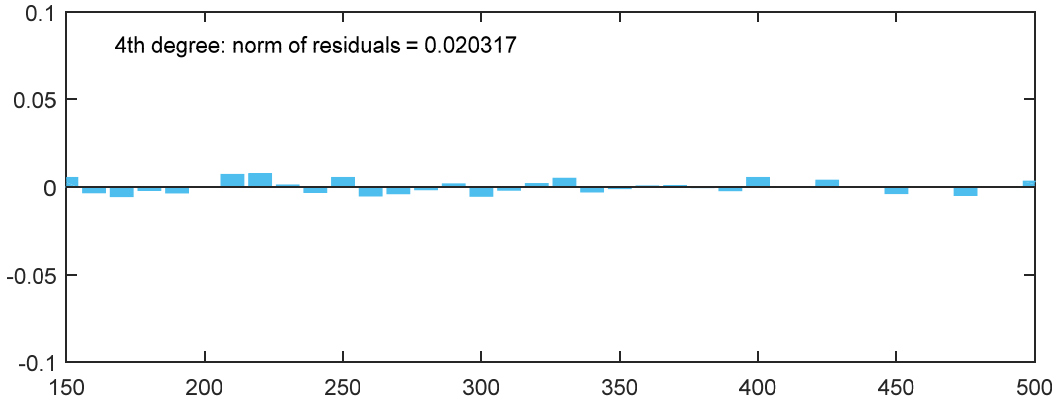

Figure 7: Norm of residuals of the 4th degree polynomial function

Once the 4th degree polynomial function was calculated, it was possible to generate the graph of the full Latency curve, shown in Figure 8. This was accomplished using a fixed MOS value of 4.41 for Latency values less than 150 ms, per the results of the E-Model, and eq. 3.2, as the Latency function for 150 ms to 500 ms.

$$\boldsymbol{If\ Latency \leq 150ms} \qquad \boldsymbol{MOS = 4.41}$$

$$\boldsymbol{If\ Latency > 151ms}$$

$$\boldsymbol{MOS = -1.11e-10 * L^4 + 1.75e-07 * L^3 - 9.825e-05 * L^2 + 0.0193 * L + 3.18}$$

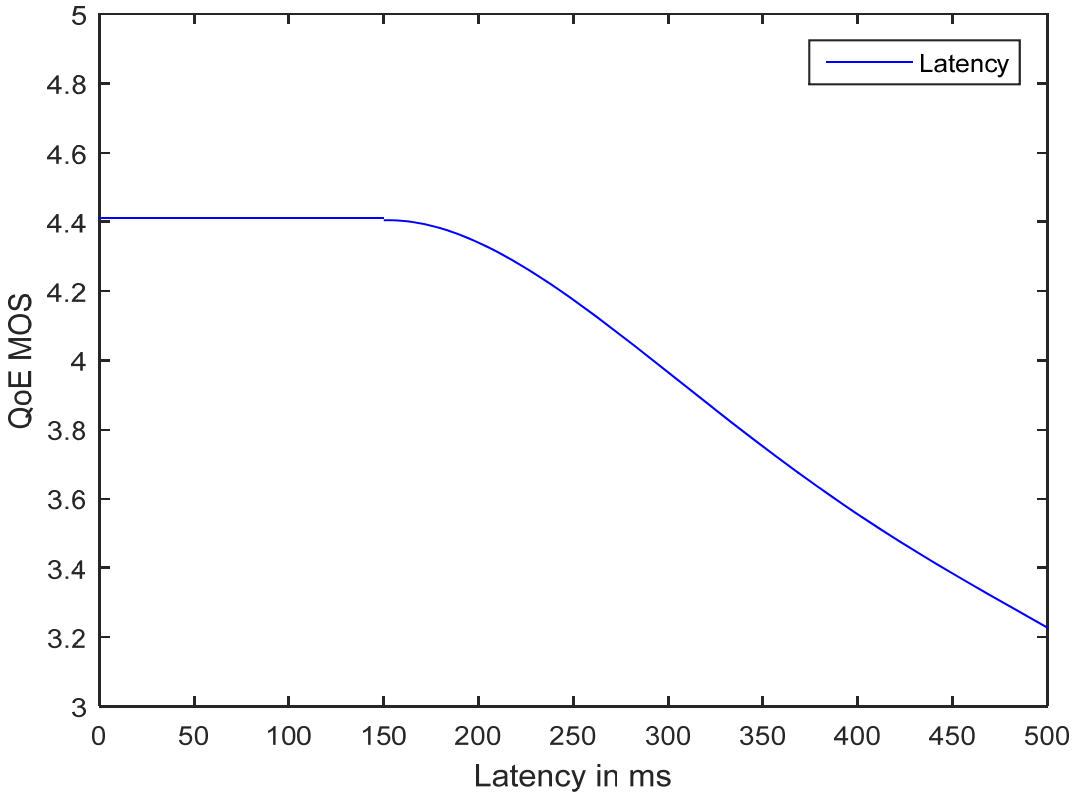

Figure 8: MOS versus Latency based on data collected from the ITU E-model

3.3.3 Packet Loss

The authors of [24] assumed a linear dependence on the QoE level, and arrived at the following differential equation:

$$\boldsymbol{QoE} = \boldsymbol{\alpha} * \boldsymbol{e}^{-\boldsymbol{\beta} * \boldsymbol{QoS}} + \boldsymbol{\gamma}$$ [24]

The authors then curve fit this equation to a MOS dataset based on user feedback of packet loss between 0 and 100%, and their result was the following function:

$$\boldsymbol{QoE = 3.010 * e^{-4.473 * Packet\ Loss} + 1.065},$$ with an accuracy of 99.8% [24]

3.3.4 Packet Delay Variation also commonly called Jitter

PDV leads directly to packets out of order, that must be reordered before the data can be used or discarded if they arrive too late. If the packets are discarded, the effect on QoE is calculated by the packet loss equation. If the packets require reordering, their impact on QoE is calculated by the packet reordering ratio. The authors of [24] applied the same method for relating the packet reordering ratio to QoE as was done for Packet Loss in the previous section. The result was:

$$\boldsymbol{QoE = 2.482 * e^{-10.453 * Reordering\ Ratio} + 1.141},$$ with an accuracy of 99.3% [24]

This formula will be used to calculate the QoE, based on the real-time QoS parameters of a given system.

3.3.5 Graph of Packet Loss & Packet Reordering Ratio

Figure 9 shows a merged graph of the equation for packet loss and the equation for the packet reordering ratio described in [24]. This presents a direct comparison of the two QoS to QoE mapping functions, and clearly shows that although packet loss has a negative effect on the QoE, the packet reordering ratio has an immediate and drastic impact on the QoE.

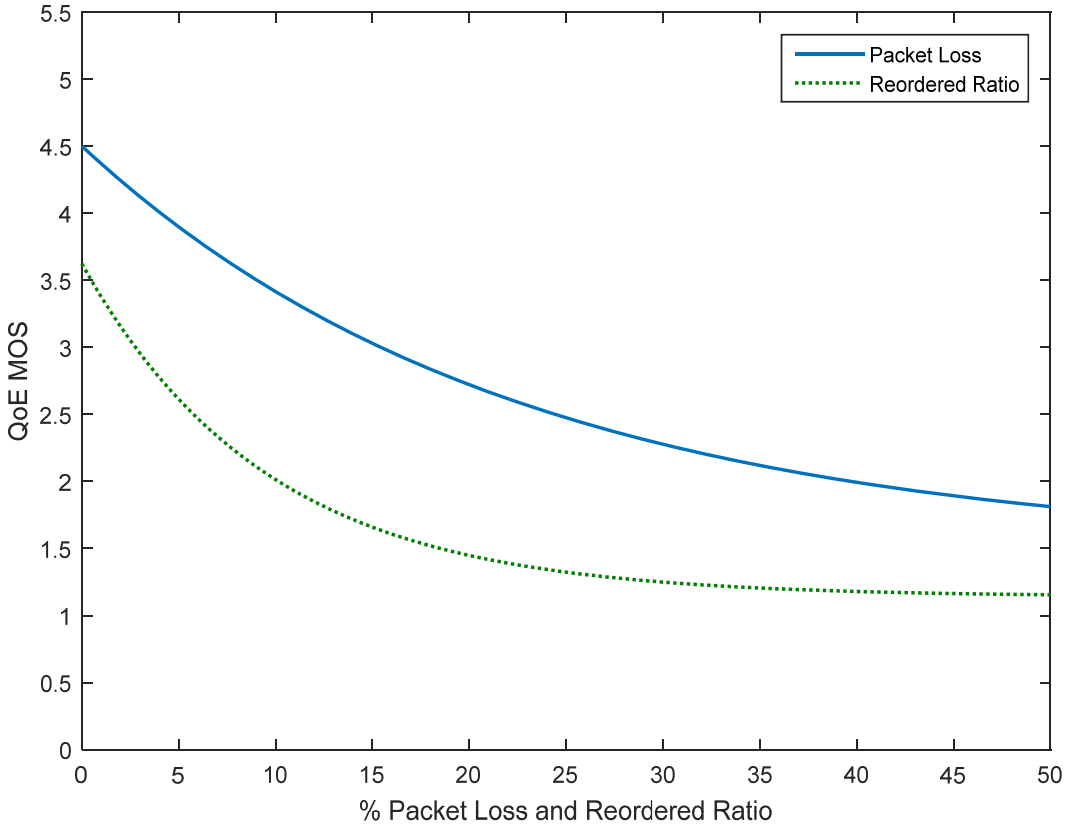

Figure 9: MOS versus Packet Loss & Reordered Ratio based on [24]

3.3.6 Graph of Packet Loss & Packet Reordering Ratio with Buffering

One example of how QoS factors interact would be that if an end device had a buffer, it could offset some of the negative impacts of PDV and packet loss. The impacts that can be offset are dependent upon many factors including the size of the buffer, the excess bandwidth available to fill and refill the buffer, and most importantly, the amount of packet delay variation and packet loss the user is experiencing. With this data, the modified functions can be used to map the QoS to the appropriate QoE in order to account for this offset. In short, the negative impact from some network QoS factors can be mitigated by a buffer, allowing for the maintenance of a higher level of QoE. If a device has an appropriate sized buffer, offsetting the packet loss or PDV still comes at a price, most often a few seconds of startup delay to fill the buffer initially; but as was shown in Figure 5, this has almost no impact on QoE. While as shown here, even a small percentage of packet loss or PDV has a very significant impact on QoE.

Once a buffer is incorporated into the system, the results would look something like Figure 10 (note that the benefit of the buffer would be impacted by the amount of data in the buffer, the rate at which the data is consumed, the amount of bandwidth available to refill the buffer, the amount of PDV and packet loss, and the impact of the startup delay accepted to pre-fill the buffer). The mark of 10% was chosen simply as a notion of what

a buffer may be able to overcome; actuality may be much higher or lower, depending on the real-time network factors of a given scenario.

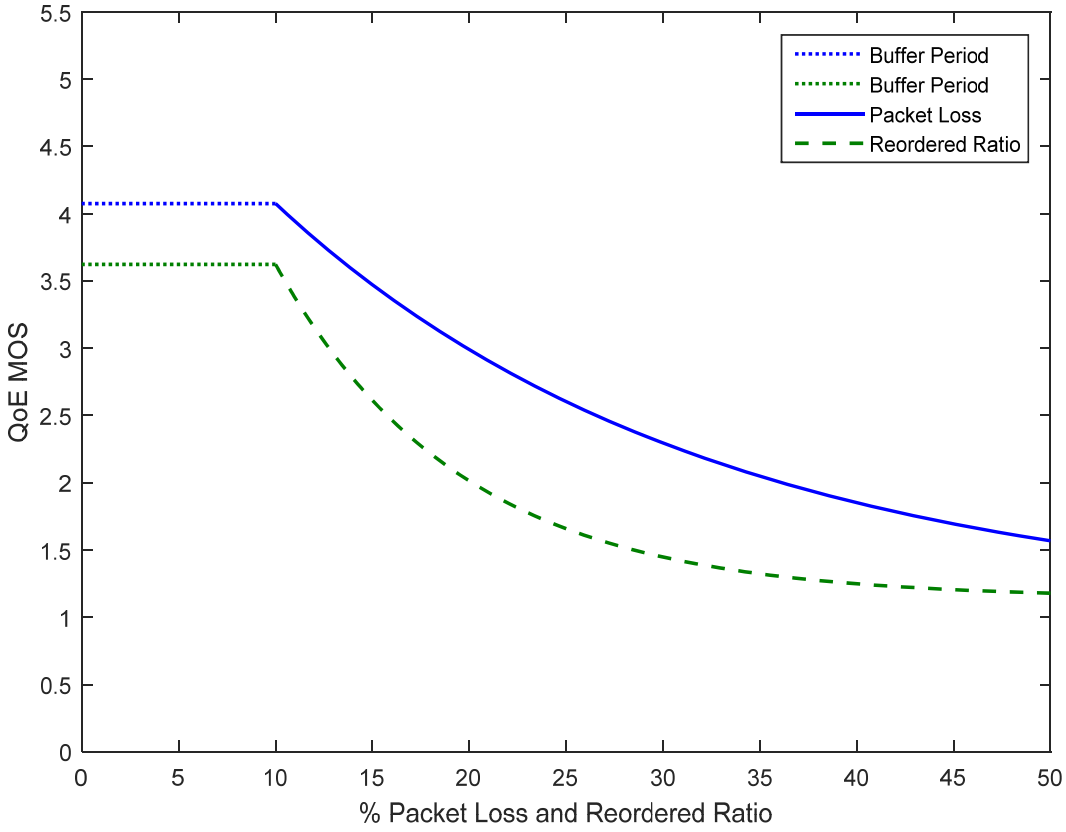

Figure 10: Impact of buffer on QoE for Packet Loss & Reordered Ratio based on [24]

3.3.7 Bandwidth

Unlike the rest of the network QoS factors that impact the QoE, bandwidth is not a smooth function. Because different resolutions of content have discrete levels, the

bandwidth requirement should follow near these discrete levels. The bandwidth will always need to be higher than the minimum required to support a given resolution because of transport overhead and the potential need to fill and maintain device buffers. Once a user's requested bandwidth exceeds their available bandwidth, the resolution of the transmission must be reduced, and this will in turn lower the QoE. Because this reduction in resolution is by a discrete factor, once a resolution that conforms to the bandwidth limitations is identified, there may be excess bandwidth available. This excess bandwidth may potentially be allocated to another network user to enhance their QoE. Therefore a graph of the bandwidth would appear as a step function because a decrease in QoE would also result in a decrease of required bandwidth, both would be abrupt changes. Further decreases in available bandwidth will not impact QoE again until the bandwidth reaches another point at which it is insufficient to support the currently transmitted resolution. At that time, the user would encounter another sudden decrease in QoE due to the audio/video quality dropping another level.

3.3.8 The Impact of Hysteresis on QoE

The concept of QoE hysteresis [28] is that if a user who is accustomed to a high QoE, experiences a small negative change, it will be more obvious than if a user with a low QoE experiences the equal and opposite change for the better. This is because improvements are always welcome but may be expected and potentially unobserved, while a negative change is not appreciated and is more likely to be quickly noticed. Therefore a small negative change in QoS may have a large negative impact on the QoE, while a positive change in QoS may have very little positive impact on the QoE.

From this, it is observed that the QoE a user perceives in a new scenario may depend greatly on QoE they had with other recent scenarios. QoE hysteresis is believed to exist because users expect equal or better quality in the future, based on the quality they received in the past.

3.4 Queueing Theory

Queueing theory is used to determine the desired network capacity needed to realize the desired QoS parameters and the impact on the final QoE if the network capacity is insufficient. An M/M/1/K queueing model [30] is used, which is similar to the classic

M/M/1 queue except that instead of assuming an infinite buffer or queue, there is a finite system size of K. In an M/M/1/K queue, the M is short for "Markov" which means an exponential distribution. The first "M" represents the arrival rate, the second "M" represents the service rate, the "1" denotes the number of servers, and the K defines the size of the system.

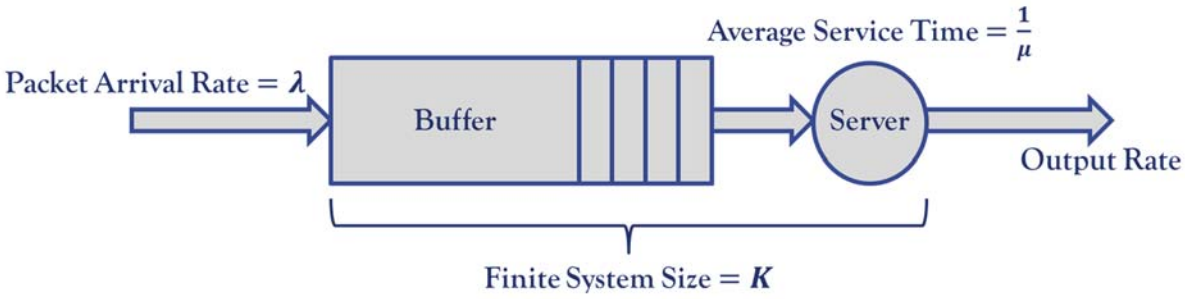

Figure 11: Diagram of M/M/1/K Queueing model

The following are the Queueing variables used in this analytical model, and the definition of each. Although many of these symbols appear to be standard, many authors use different symbols for the different variables and with vastly different units of measure. The following appeared to be the most common [30][31][32], but due to the potential for confusion, they are included here with the proper units for network queueing in the description for each variable.

$\boldsymbol{\lambda_i} = \frac{N}{T}$ = Average Packet Arrival Rate, in packets per second

$\boldsymbol{C_i}$ = Capacity of the Link, in bits per second

$\boldsymbol{L_i}$ = Average Packet Size, in bits per packet

$\boldsymbol{\mu = \frac{C}{L}}$ = Average Service/Transmission Rate, in packets per second

$\boldsymbol{\frac{1}{\mu} = \frac{L}{C}}$ = Average Service/Transmission Time, in seconds per packet

$\boldsymbol{\rho = \frac{\lambda}{\mu}}$ = Link Utilization Factor, as a percentage

$\boldsymbol{\gamma}$ = Throughput or rate of service, in packets per second

$\boldsymbol{B}$ = Buffer, in bits

$\boldsymbol{\gamma = \lambda(1 - P_b)}$ = Throughput or rate of service, in packets per second

$\boldsymbol{T = \frac{N}{\lambda} = \frac{\rho}{\lambda(1-\rho)} = \frac{1}{\mu-\lambda}}$ = Average Packet Transfer Delay (including time in the Queue), in seconds

$\boldsymbol{N = \lambda T = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu-\lambda}}$ = Average Number of Packets in the System, in packets

$\boldsymbol{W = T - \frac{1}{\mu} = \frac{\rho}{\mu-\lambda}}$ = Average Waiting Time in the Queue, in seconds

The percentage of Packet Loss is calculated based on the probability of blocking $\boldsymbol{P_b}$, where $\boldsymbol{B}$ is buffer size that will ensure the percent packet loss is less than the $\boldsymbol{P_b}$ using:

$$\boldsymbol{\gamma = \lambda(1 - P_b)} \text{ [[31] } \mathbf{eq.2-22}\text{], and}$$

$$\boldsymbol{P_b = \frac{(1-\rho)\rho^B}{1-\rho^{(B+1)}}} \text{ [[31] } \mathbf{eq.2-24}\text{].}$$

The throughput of the system in terms of $\boldsymbol{P_b}$ is $\boldsymbol{\gamma = \lambda(1 - P_b)}$ [31] eq. 2-22, in packets per second. And the normalized throughput of the system as a function of the normalized load is:

$$\frac{\boldsymbol{\gamma}}{\boldsymbol{\mu}} = \frac{\boldsymbol{\rho(1 - \rho^B)}}{\boldsymbol{(1 - \rho^{B+1})}} \text{ [[31] } \mathbf{eq.\ 2 - 26}\text{]}.$$

The Transmission Delay, also known as Latency, consists of both the queueing delay and the actual service time, is defined as:

$$\boldsymbol{T} = \frac{\mathbf{1}}{\boldsymbol{\mu C_i - \lambda_i}} \text{ [[32] } \mathbf{eq.\ 5.18}\text{]}.$$

The formula above is how the Transmission Delay equation is nearly always written, but in order to incorporate this equation into the QoS to QoE mapping model, it is necessary to redefine the value of μ based on the following information.

In chapter 5, section 6 of [32], Kleinerock defines $\boldsymbol{1/\mu C}$ as the "mean service time in seconds" and earlier in chapter 5, section 1, Kleinerock gives a footnote that says "*In this chapter, we use a different definition for $\boldsymbol{1/\mu}$ than we had used earlier. We do this

specifically to introduce the processor (or channel) capacity ***C***." The text in chapter 5, section 1, that this footnote linked to, Kleinerock writes:

> *"We let **C** denote the capacity of the resources in operations per second. We consider an orderly situation in which waiting jobs form a queue. Furthermore, we let **1/μ** represent the average number of operations required by a job. Thus we see that the average number of seconds a job requires from the resource is simple* **1/μC**."*

When attempting to mix multiple formulas in order to create a protocol in which several different parameters are calculated and recalculated based on their interactions, redefining variables is not an acceptable solution. Because of this situation, it was necessary to change ***1/μC*** to be ***1/xC***, and solve for x. Doing this allowed for consistent variables throughout all of the queueing equations within the model. The result was that ***x*** would simply be equal to ***1/L*** where ***L*** is the mean packet size in bits. This seemed much simpler then attempting to redefine ***μ*** in the first place, which is used as a fundamental variable throughout queueing theory.

In order to account for this significant deviation from Queueing Theory's standard definition of ***μ***, the Transmission Delay function above is rewritten, eq. 3.3, in order for

the variables to be consistent and relatable between multiple queueing equations. This proposed change maintains the variable $\boldsymbol{C}$ in the original equation, which is needed because the capacity of the link is the primary variable in the proposed protocol. Kleinerocks' equation 5.18 is changed to:

$$\boldsymbol{T} = \frac{\mathbf{1}}{\left(\frac{\boldsymbol{C_i}}{\boldsymbol{L_i}}\right) - \boldsymbol{\lambda_i}} \quad \text{eq. 3.3}$$

Incorporating these equations into the protocol's calculations will provide insight into the impact of network capacity, network congestion, device buffers, and end-to-end bandwidth on the QoE.

3.5 Desired Quality of Experience

The novel concept of DeQoE proposed in this dissertation arose from the fact that Internet users connect to other Internet users and services through various wired or wireless network connections and a vast array of heterogeneous devices (desktop computers, laptops, mobile devices, etc.) with different audio and video processing and playback capabilities. The vast difference in device capabilities, and the qualities of

available content, make calculating a general QoE for a specific user of a given scenario incredibly difficult and nearly useless in terms of providing a metric of success. Furthermore, QoE for video has become a moving scale as technology advances, e.g., what was considered great video quality a few years ago, is considered second rate today, and new video standards are being rapidly deployed.

The concept of DeQoE is to determine the QoS values required to meet a user's desires or expectations, based on that individual user's circumstances, and setting that as being the highest potential QoE of the system. The actual QoE the user experiences in a given scenario is then called the Realized QoE (ReQoE), which is based on how well a scenario comes to meeting the users DeQoE. This makes it possible to compare a user's ReQoE to their DeQoE, to measure how successful the system was in meeting the user's expectations. This success metric may be used to make more accurate comparisons of user experiences across a wide range of devices, contexts, and scenarios. In addition, the information available, based on the DeQoE, provides network service providers with the ability to allocate resources for performance optimization.

The DeQoE can also be understood as the QoE goal that a system should attempt to achieve. This is because in many cases, one or more constraints will preclude maximizing all QoS parameters (bandwidth, delay and PDV, packet loss) necessary in order to

achieve the optimal QoE. Again, once the final ReQoE is determined, the system's level of success can be measured against the DeQoE. This is a crucial concept because DeQoE inherently allows for the performance of heterogeneous systems and scenarios to be normalized and compared.

In order to optimize the QoE of a given scenario, the highest potential QoE must be determined, based on the primary non-network limiting factors of that scenario it is calculated as follows:

$$\boldsymbol{DeQoE = f\left(Min\ (Displayable\ Resolution\ and\ Available\ Resolution)\right)} \quad \text{eq. 3.4}$$

The displayable resolution is a fixed limiting factor, and the available resolution of the source material is another fixed limiting factor. Therefore, the notion of DeQoE, is that a reasonable user will not expect the QoE of a particular scenario to exceed the minimum of these fixed non-network limiting factors. In other words, if a user's device can only display a video of a particular resolution, and the source resolution is higher, the user can at best expect the resolution that their device can display, and in the case the source resolution is lower, the highest resolution the user can expect is the resolution of the

source (there is the possibility for the user device to implement some method of up conversion of media, but the impact of such a feature is not being considered at this time).

Case* 1**: ***If Displayable Resolution = Available Resolution

Then you have an ideal system

Case* 2**: ***If Displayable Resolution ≠ Available Resolution

Then the DeQoE will be based on the lower of the two resolutions

The next step in determining if a particular level of QoE can be achieved, is to find out if the system can meet the demands of the DeQoE, including the latency, packet delay variation, packet loss, startup delay, and bandwidth. In the second case, because the highest potential DeQoE is based on the lower of the two constraints, the system therefore requires lower overall QoS parameters to achieve the highest DeQoE; knowing this can conserve users' and providers' resources (which likely saves money) and any unused resources may potentially improve the QoE of other users. This is especially valuable for bandwidth limited and spectrum constrained wireless and mobile networks, where this extra load may even negatively affect other users. Because the ReQoE is the final QoE a

user receives, in an ideal scenario $\boldsymbol{DeQoE = ReQoE}$, but typically $\boldsymbol{DeQoE > ReQoE}$ due to any number of network factors that also impact the QoE, even if you have an ideal system as described in case 1.

3.6 An Analytical Model for improving DeQoE

There are many parameters that will be analyzed to find the QoE of a system and determine how best to optimize network resource utilization and improve the QoE. Many of these factors are not only required input to the calculation, but will also require a degree of recursive calculation due to the interdependencies that exist between the device, data, and network variables. This is the case if one is changed to accommodate the others; then the entire model will need to be executed again, utilizing the updated values. This is especially true for multi-user scenarios, where negatively impacting the QoE of one user may enhance the QoE of one or more other users.

In order to consider the interactions of the various QoS parameters on an end user's QoE, it is necessary to understand the application. There are essentially two different network models that are considered in terms of the QoE. The first is live applications such as a Voice over IP call or conference, and a Video call or conference over IP. The

second is recorded media that can be either audio or video in nature. The QoE of these communication types are sensitive in different ways to the various QoS factors. Ensuring sufficient bandwidth is critical for both applications and, based on queueing theory, the lack of it can be detrimental in terms of higher packet loss.

For live data applications, if latency, packet delay variation, packet loss, and startup delay are high, then the QoE will be low. If only latency and/or startup delay are high, then this means initiating a streaming connection may take a bit longer, but once it has begun, the session should run fine.

For recorded media, if packet delay variation and packet loss are high, then a large enough buffer may be able to maintain a high QoE. There are several conditions for this: the buffer must be pre-filled; the device must have sufficient bandwidth to replenish the buffer if it is necessary to reorder any out of order packets; and the device buffer must allow for enough time to re-request any lost packets, before they are required, to ensure the continuous display of content to the user. For recorded media, increasing the buffer can improve the QoE while only adding a small initial queueing/startup delay which, as shown in [29], only negligibly impacts the QoE. This allows the system to then maintain a high QoE throughout the rest of the session.

If there is a packet reordering issue due to severe packet delay variation in a live application, then in most cases, the out of order packets (likely User Datagram Protocol (UDP)) will be discarded. Some devices may have a small buffer that can overcome minor packet reordering issues for live applications, but this is heavily dependent upon the packet delay experienced by the out of order packet. If the packet arrives after its contents were intended to be consumed, then it is too late to be useable and must be considered packet loss.

Although these two communications models differ greatly, they also share many requirements in common. In both models, bandwidth is a critical component and both can be optimized in real-time to improve the users' QoE.

3.6.1 Initial Calculations and Considerations

Variables:

dr = Max Device Resolution (converted to a data rate)

db = Max Device data rate (user's available bandwidth)

sr = Max resolution of source content (as a data rate)

sb = Max source bandwidth (provider's available bandwidth)

eb = End-to-end available network bandwidth

The first step is to determine the resolution that the content provider should transmit to the end user, this also becomes the basis for the users' DeQoE:

1. Find the Maximum resolution (`maxr`) to request:

```
If dr < sr
     Then dr = maxr
Else sr = maxr
```

2. Find the maximum bandwidth (`maxb`) to support transmission:

```
If eb < db && eb < sb
     Then eb = maxb
Else If db < eb
     Then db = maxb
Else sb = maxb
```

3. Determine which resolution should be transmitted:

```
If maxb > maxr then maxr is achievable
Else maxr is limited by maxb
```

Next, a number of basic end-to-end calculations are performed to determine the end-to-end throughput available, prior to calculating the QoE.

RD = Requested Data in Mbits = [Data Rate of Requested Audio + Requested Video + IP overhead]

ADB = Device Bandwidth in Mbits = [Current Available Device

Throughput – Current Utilization]

$\boldsymbol{PB}$ = Total Provider Bandwidth

$\boldsymbol{PU}$ = Provider Utilization

$\boldsymbol{NB}$ = End-to-End Bandwidth Available = [Network Capacity – Network Utilization]

$\boldsymbol{Tx}$ = Transmission Rate

$$\boldsymbol{If}\ \frac{\boldsymbol{RD}}{\boldsymbol{ADB}} \leq \mathbf{1}\ \textit{Then User's Device is capable of receive at the requested rate}$$

$$\boldsymbol{Then\ If}\ \frac{\boldsymbol{PU + RD}}{\boldsymbol{PB}} \leq \mathbf{1}\ \textit{Then Provider has available bandwidth}$$

$$\boldsymbol{Finally\ If\ Tx} = \frac{\boldsymbol{PU + RD}}{\boldsymbol{NB}} \leq \mathbf{1}$$

$$\textit{Then Provider can transmit at the requested rate}$$

If $\boldsymbol{Tx}$ is greater than 1, then a long delay is necessary to pre-fill available buffer, and if the buffer runs out, then streaming will be paused (delayed). This would lead to a very poor QoE, especially in situations with long running instances that require several pauses of the data stream.

3.6.2 DeQoE Protocol Decision Diagram

The flow diagram shown in Figure 12 provides a visualization of the process the DeQoE protocol runs through and the variables that are required (or must be assumed)

to complete the initial optimization calculation. There are parts of this that are purposefully recursive; for example, if the bandwidth is initially found to be sufficient, but due to poor latency/PDV/packet loss, a buffer is required, filling the required buffer could raise the bandwidth requirements of the system past that of the available bandwidth. This would necessitate a drop in the QoE, and the variables would then be recalculated based on the new lower QoE being the new goal of the system.

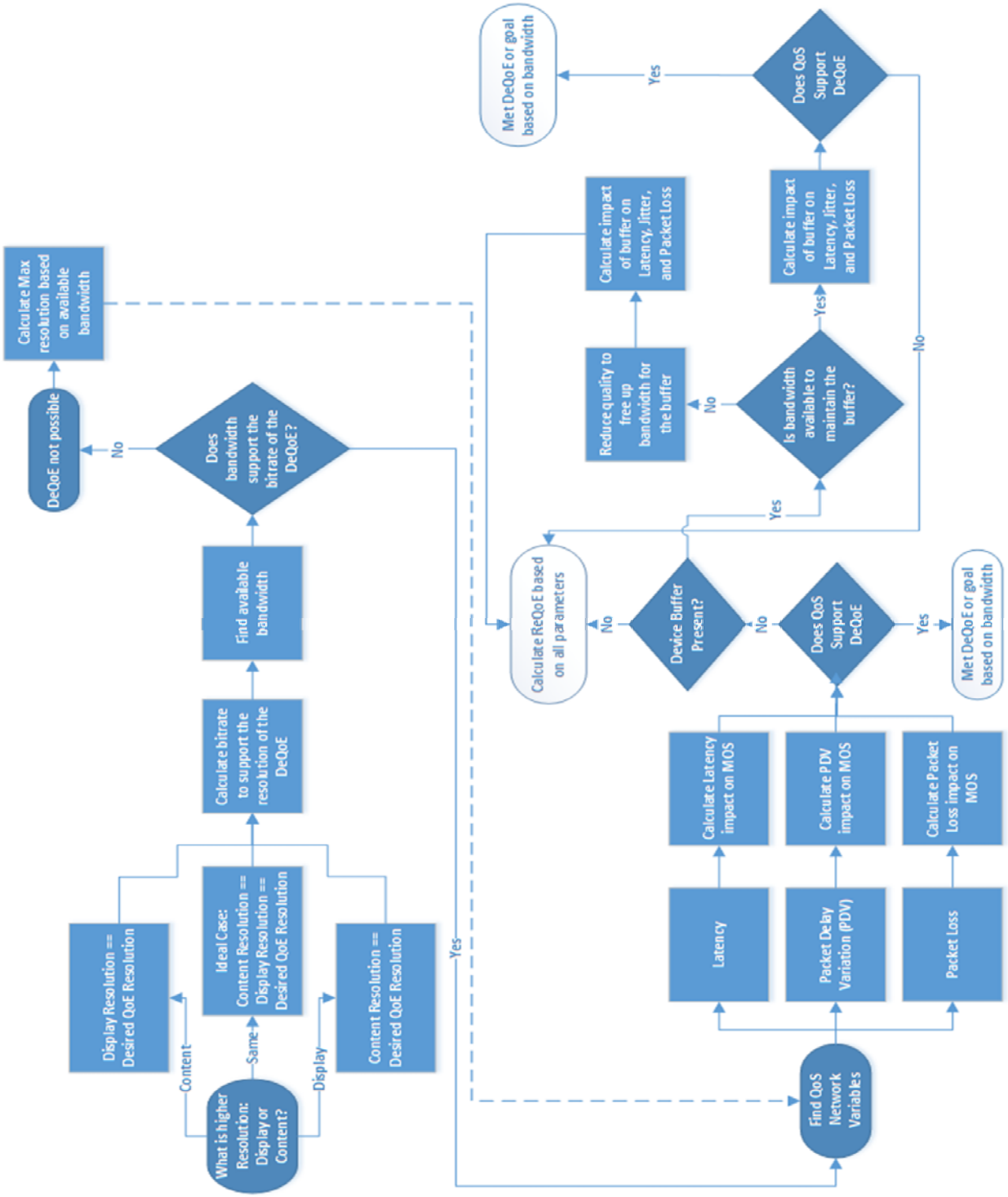

Figure 12: DeQoE Protocol Flow Diagram

Chapter 4 CloudEdge

CloudEdge enables a wireless access network with computational power, storage, and SDN control. CloudEdge will actively evaluate network conditions and user requirements in real-time and make adjustments, if necessary, to improve the average ReQoE of its connected users. CloudEdge can be implemented as an overlay on conventional IP networks and it is intended to be compatible with future potential Internet architectures such as CCN and MobilityFirst. This dissertation focuses on a specific use case, in-network rate adaptation of streaming video, to further describe the CloudEdge design in more detail and demonstrate the benefits. The goal in this scenario is to enable the optimization of multiple variables with multiple constraints to improve the average ReQoE of a wireless access network using in-network transcoding. Figure 13 provides an overview of the CloudEdge architecture and data paths through the network.

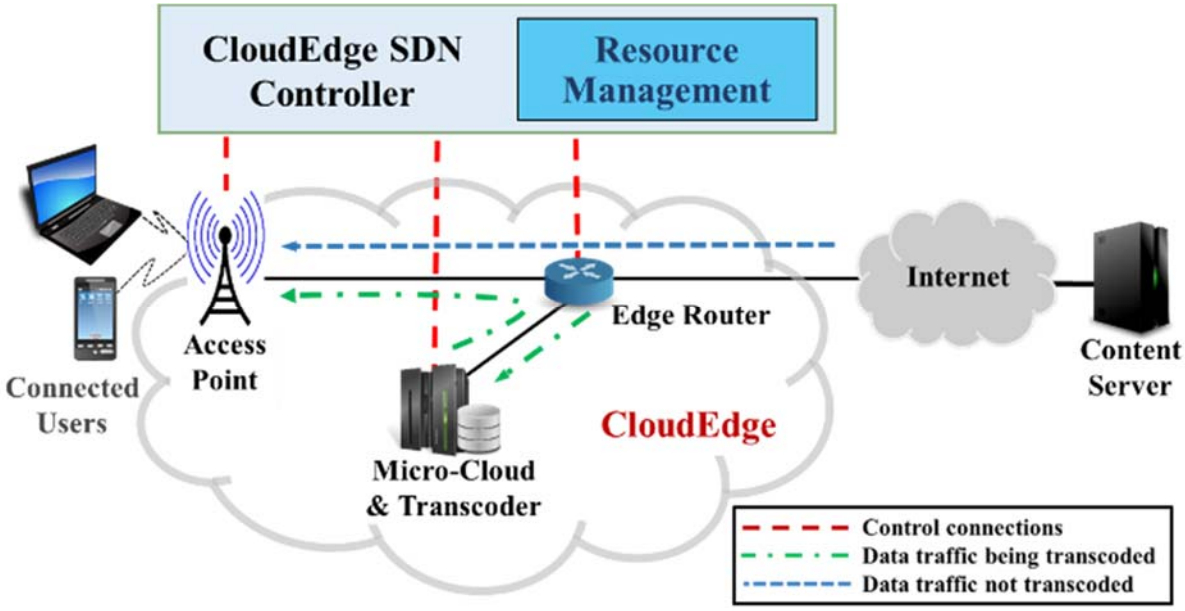

Figure 13: Diagram of CloudEdge data flows [33]

4.1 Review of Literature

With regard to related works, content-centric networking (CCN) [14][34][35][36][37], also referred to as information-centric networking (ICN), has been proposed as a new paradigm to address the challenges posed for content retrieval. In CCN networks, content routers (CRs) have integrated storage to cache content in the networks. A user requests a content object (CO) by sending its Interest packet containing the requested content name. The CR routes the Interest packet to the best cache or source using name-

based routing, which responds with the requested content data. Although CCN enables in-network caching and delivery from the best content location, it typically does not process and change data representation in the networks. Content distribution networks have been widely deployed to improve content delivery performance by strategically placing overlay cache servers close to users. CCN [34] provides a network layer solution by integrating the caching function in the content routers with name-based routing to deliver the content from the best source. However, these solutions do not change the content representation (e.g., they do not perform transcoding). In [38], a scheme was proposed to use the public clouds to perform in-network processing tasks for enterprise networks. However, it requires the traffic to be routed to the public cloud and may generate unwanted inter-domain traffic [39]. In [40], an architecture was proposed to consolidate middleboxes in enterprise networks and enable the middlebox applications to run on a consolidated hardware platform. However, this work was mainly focused on the management of general enterprise network appliance functions such as intrusion detection systems, firewalls, HTTP proxies, VPNs, and WAN optimizers, instead of video content delivery. In addition, none of these consider the impact of varying wireless channel conditions to the middleboxes, nor allow third-party content providers to use the resources.

A cloudlet based on an open platform architecture was proposed in [41] to host in-network services, and to help Internet Service Providers (ISPs) and application providers deploy their services to strategic network locations. A video transcoding service was implemented based on the cloudlet in [42]. However, the designs in these two works are based on MobilityFirst, a clean-slate internet architecture [14], which uses a globally unique identifier (GUID) for content, service and other objects, as well as a distributed GUID resolution mechanism to resolve GUIDs to their locators or network addresses. This work differs from the aforementioned two works in two main ways: first, neither of these papers integrates SDN into the architecture and second, the resource management and optimization challenges in the open cloudlet are not addressed. In this architecture, the SDN controller is extended to manage not only data delivery but also content processing and caching, which allows for unified resource management and optimization based on varying wireless network conditions. In addition, although SDN has been used in data center networks [6][7], there has been little work on quantifying the benefits of SDN and in-network processing for wireless content delivery; nor any in-depth attempt to re-architect wireless access networks to exploit integrating these techniques. Based on a thorough review of the literature, it is believed that CloudEdge is the first design of an

SDN-based wireless access network architecture with integrated processing and storage capabilities.

4.2 Methodology

This dissertation is focused on video because it is the predominant traffic in networks, and it will greatly benefit from in-network adaptation services. The bottleneck in content delivery is usually at the wireless access network, and the network performance experienced by wireless users is highly variable due to mobility, channel fading, and traffic load. Dynamic Adaptive Streaming over HTTP (DASH) [26][43] is a standard state-of-the-art solution for video streaming, in which the client estimates the available bandwidth, and selects a representation of the video segment based on current bandwidth constraints. It is well known that obtaining accurate estimates of available bandwidth is problematic for clients, especially in wireless networks. In addition, this client-side approach could cause degradation and instability in content delivery performance because each client performs its own estimation and rate adaptation [44][45]. CloudEdge provides a network-side solution for rate adaptation. Deployment of the video adaptation service within the wireless access network would greatly improve

the accuracy of the estimation of the wireless network state. Furthermore, the wireless network operator can take advantage of in-network processing to manage multiple video flows in an efficient and fair way to improve the overall QoS.

In the remaining sections of this chapter, there is a presentation of the system architecture including the variables and constraints in the proposed CloudEdge video transcoding scenario. Then, the analytical methods of this analysis are presented and described. Lastly, a review of the results obtained through the different analytical methods for multiple types of scenarios is offered, including a description of the results and initial conclusions.

4.3 System Architecture

The architectural design of CloudEdge is that of a network service centric model. Wireless access network operators, such as cellular or hotspot operators, can deploy a distributed pool of computing and storage resources in their networks, which can host virtual instances to dynamically perform customized computations on their users' data as needed, as well as cache processed data while delivering data to end users. The computation and storage hardware may be integrated into or co-located with networking

nodes such as backhaul routers, switches, and access points (AP)/base stations (BS). The following are the principles of the CloudEdge design:

- *Decoupling of the data plane and control plane, as well as the software from the hardware*. This SDN technique allows for traffic flows to be routed with greater flexibility. NFV allows for software-based implementations of in-network services to run on general-purpose or specialized hardware platforms.
- *Unified control*. Wireless transmission, traffic routing, in-network processing, and caching are all managed by a single (logically) centralized controller. Extending the SDN concept, the controller takes a unified, network-wide view to generate configurations and policy rules for all traffic as well as in-network services. The bandwidth, computation, and storage resources can be abstracted and shared by multiple applications [6][39]. This is different from standalone middleboxes, where each box or network application must be managed independently. Given the power of modern computing and storage devices, as well as the potential scale of a wireless access network, redundant clusters may be used as the controller platform [5][6] to address single-point failure and scalability concerns.
- *Open interface*. The resources in the access network can be used by network operators to provide in-network services. Such services could be user requested

services and/or transparent services that are activated by the network operator. In addition, like a cloud, the network infrastructure can be virtualized, and the resources can be used to host the third-party services, as virtual instances, through an open interface. This can simplify new service deployments for the third-parties, and also be a means of generating extra revenue for the network operator.

- *Name-based network management*. Each device, service, content item, resource, and interface is given a unique name that is used for reachability and management.

Figure 13 illustrates the high-level system diagram of a CloudEdge wireless access network. Mobile users connect to the wireless access network through WiFi or 4G radios. The AP/BS is enhanced to report measured radio parameters, e.g., the link data rate of each mobile user and current bandwidth usage to the CloudEdge controller. The AP/BS also receives configuration commands from the controller. To reduce the controller load, the MAC function is split between the controller and the AP/BS. The controller provides the high-level policy such as the maximum share of channel time that can be used to transmit a particular data flow. The AP/BS will implement the scheduling to transmit data using standard protocols, such as IEEE 802.11 or Long-Term Evolution (LTE), based on the resource allocation policy.

The bandwidth, computation, and storage resources can be virtualized and dynamically shared by different applications and services. The CloudEdge controller will monitor and manage all the resources in the access network. All of the devices, such as routers, switches, APs/BSs, computers, and networked disks, have separate control and data interfaces. The control interfaces are used to communicate with the controller, reporting the resource usage and service status, and receiving commands from the controller. The controller also has an open interface for external applications that allows third-party application/content providers to lease resources and deploy their own services within the access network.

Just as in a CCN based network, where each content object (CO) is given a unique identifier (COID), in the CloudEdge network, every deployed service is also identified by a unique and routable service identifier (SID). More so, each of the network-attached objects, including devices, contents, services, applications, interfaces, are assigned a unique name or identifier, like several other future Internet architecture designs [14][34][35][36][37]. A hierarchical naming scheme can be used [34]. The COID of a video segment in a binary-coded format is as follows: /Publisher.Com/MovieTitle/SegmentID [34]. An example of a SID for may look like this: /WirelessNetID/TranscodingServiceID. The CloudEdge controller would be responsible for maintaining the mappings between

the SID's and the virtual/physical resources. The CloudEdge controller also manages the in-network processing and caching, as well as the role of an SDN controller responsible for establishing the data routing paths.

When the edge router in CloudEdge receives a CO, it verifies whether the rules have been set up to forward this CO in its forwarding table based on the CO header. If yes, it will apply the rules to forward the CO to the next hop. Otherwise, it will buffer the CO, and forward only the CO header to the controller through its control interface. The controller will then make a decision based on the resources required by this CO and the resources available in the access network If there is insufficient wireless bandwidth to transmit the CO to the user, or an in-network service, e.g., video coding format change has been requested by the original sender (a requested SID included in the CO header), the controller will instruct the edge router to send the CO to the transcoding service for processing by adding a command field in the CO header and sending the header back to the edge router. The command field contains the routing rules from the edge router to the transcoder, and the rules for processing the CO, including resolution, format, and data rate that the content needs to be transcoded to, as well as the route from the transcoder to the user. The CO is forwarded to the transcoder, the transcoder reduces the video resolution per the controller's instruction, and the transcoded video is sent to the

AP/BS. If there is enough wireless bandwidth to forward the CO to the user and no in-network service is requested for this CO, the controller will instruct the edge router to directly forward the CO data to the user through the AP/BS by adding a source route in the CO header before sending the modified CO header to the edge router over the control interface. It is also possible for the controller to instruct the edge router to handle additional COs belonging to the same content or user with the same rules until such rules are modified or expired. Then the edge router does not have to send the CO headers of the related incoming COs to the controller.

4.3.1 List of Variables and Equations

The variables and equations defined here include the objectives and constraints of the CloudEdge architecture that is modeled.

- A video stream $\boldsymbol{i}$ is sent to user $\boldsymbol{i}$ at a data rate of $\boldsymbol{r_i}$
- The AP channel utilization is $\boldsymbol{\alpha_i}$
- The throughput of stream $\boldsymbol{i}$ is: $\boldsymbol{T_i = \alpha_i r_i}$ eq. 4.1
- The data transmission must meet the wireless channel utilization constraint:

$$\sum_{i=1}^{N} \boldsymbol{\alpha_i} \leq \mathbf{1} \quad \text{eq. 4.2}$$

- The number of processing cycles needed to transcode a stream: $\boldsymbol{c_i}$
- The total available processing cycles is: $\boldsymbol{C}$
- x_i is used to indicate if a stream is transcoded ($\boldsymbol{x_i = 1}$) or not ($\boldsymbol{x_i = 0}$)
- Then, the transcoding constraint is:

$$\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{c_i x_i \leq C} \quad \text{eq. 4.3}$$

- The desired video rate is: $\boldsymbol{S_i^d}$
- The final transmitted video rate (after transcoding if necessary) is: $\boldsymbol{S_i^t}$
    - The transcoder can only reduce the video resolution, Therefore:

$$\boldsymbol{S_i^t < S_i^d \; if \; x_i = 1} \; (transcoding) \quad \text{eq. 4.4,}$$

$$\text{and,} \boldsymbol{S_i^t = S_i^d \; if \; x_i = 0} \; (no \; transcoding) \quad \text{eq. 4.5.}$$

- If $\boldsymbol{\delta}$ denotes the protocol overhead, which includes the lower layer headers and overhead to transmit video data, then the required bandwidth to transmit the video stream is:

$$\boldsymbol{T_i = \alpha_i r_i = \delta S_i^t} \quad \text{eq. 4.6}$$

The objective of the CloudEdge implementing transcoding in access networks is to:

$$\boldsymbol{Maximize} \sum_{\boldsymbol{i}} \boldsymbol{ReQoE_i(\alpha_i, S_i^t)} \quad \text{eq. 4.7}$$

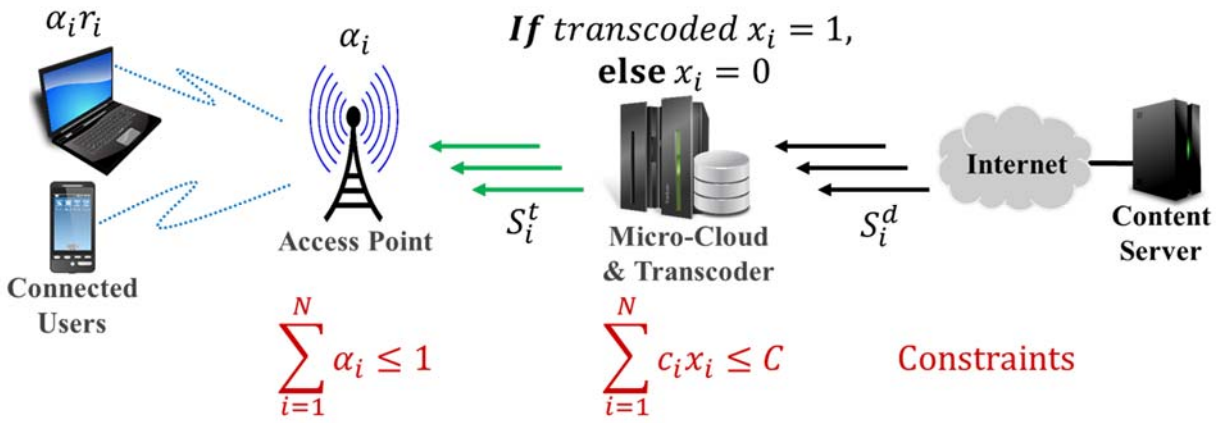

Figure 14: CloudEdge network data flow including variables and constraints [33]

4.4 Enabling Video Transcoding on Wireless Edge Networks

There are several assumptions and pre-defined variables that need to be established to setup the simulation before any calculations can be made. These include the number of supported video resolutions, the number of users, the number of access point data rates, the packet overhead rate, and the number of transcoders available to the model. The following sections describe how many of these are arrived at and why they are the values utilized here. The model was engineered in an elastic fashion so that in the future, most of these variables, including the number and data rates of the resolutions and the data rates of the access point, can be easily adjusted to simulate different scenarios.

4.4.1 Access Point Data Rates

The first step is to determine the wireless data rates at which users will be connecting to the access point. For this scenario, 802.11g access point data rates were used. These rates are derived from each user's wireless Signal to Noise Ratio (SNR); the ratio of wireless signal power to background wireless noise power between a user and an access point (AP) [46]. This is because the SNR dictates the number of coded bits per orthogonal frequency-division multiplexing (OFDM) symbol, as well as the level of Forward Error Correction (FEC) required to ensure reliable communications. Which in turn impacts the goodput data rate perceived by the user once connected. The total throughput of an AP is divided amongst its users and the total AP utilization is the sum of the percent utilization of each user's connection rate that is being used. For example, an 802.11g access point has a theoretical maximum throughput of 54 Mbps, but this is shared amongst all connected users. Therefore, if two users are connected at 54 Mbps and each user is given 50% of the AP utilization, then each user would have a maximum data rate of 27 Mbps. If instead, one of the two users was connected at 24 Mbps, then with 50% of the utilization, that user would have a maximum data rate of only 12 Mbps. Table 5 shows the mapping of various SNR's to IEEE 802.11g data rates. The fact being that a high SNR means a user can achieve a much higher data rate, while a low SNR requires

different modulation, coding rates, and more forward error correction to overcome, which results in a great deal of overhead traffic and a much lower goodput data rate.

Modulation	FEC Code Rate	Coded bits per Subcarrier	Coded bits per OFDM Symbol	Data bits per OFDM Symbol	Data Rate in Mbps, 250K Symbols per sec	Minimum SNR in dB
BPSK	½	1	48	24	6	4
BPSK	¾	1	48	36	9	5
QPSK	½	2	96	48	12	7
QPSK	¾	2	96	72	18	9
16 QAM	½	4	192	96	24	12
16 QAM	¾	4	192	144	36	16
64 QAM	⅔	6	288	192	48	20
64 QAM	¾	6	288	216	54	21

Table 5: Characteristics of IEEE 802.11g data rates [46]

4.4.2 Bandwidth requirements based on Video Resolutions

To determine the required bandwidth per user, two primary factors are accounted for in this model. The first factor is the data rate of the video stream. The second is a percent network overhead factor that is added to that data rate. In order to calculate the required bandwidth to transmit different video resolutions and what the change in bandwidth would be if a video resolution was lowered, it was necessary to establish a standard set of resolutions and data rates. Table 6 lists the standard bitrates as recommended by YouTube.com that were chosen to be utilized in this model.

Type	Video Bitrate	Audio Bitrate	Resolution
1080p	8,000 kbps	384 kbps	1920x1080
720p	5,000 kbps	384 kbps	1280x720
480p	2,500 kbps	128 kbps	854x480
360p	1,000 kbps	128 kbps	640x360

Table 6: Standard recommended bitrates for YouTube.com [47]

4.4.3 Mapping Discrete Changes in Video Resolution to ReQoE MOS

In order to quantify the impact of transcoding the video resolution on the ReQoE, the ratios of the qualitative relationship shown in Table 7, as defined by International Telecommunications Union (ITU) [13], are considered and it is postulated that for each drop in resolution (1080p to 720p to 480p to 360p), the qualitative "User Rate" would also drop one level.

User Rating	MOS
Very satisfied	4.3-4.5
Satisfied	4.0-4.3
Some users satisfied	3.6-4.0
Many users dissatisfied	3.1-3.6
Nearly all users dissatisfied	2.6-3.1
Not recommended	1.0-2.6

Table 7: MOS as defined in ITU-T Rec. G.107 Annex B [13]

This conjecture resulted in the three data sets shown Table 8, Table 9, and Table 10, which show the discrete changes in the QoE based solely on the difference between the

desired resolution and the received resolution. Because the ITU defines a range for User Satisfaction, three tables were generated: one based on the lower limit, one on the upper limit, and one on the mean of the MOS range. For this simulation, calculations for the upper limit, shown in Table 9, were used because many other factors may also lower the QoE and the objective here was to find the highest possible QoE given a user's requested resolution. As an example, if a user requests a streaming video at a resolution of 1080p and receives 1080p video, the user has a ReQoE of 4.5 or Very Satisfied. If a user receives 720p when his desired rate is 1080p, that user has a ReQoE of 4.3 or Satisfied. The ReQoE is based on the change in received vs. requested resolutions. Here, it is assumed that there is no packet loss. The impact of packet loss will be discussed later. Also note that if the user requests 360p and receives 360p, the user's ReQoE is the same as if the user requests 1080p and receives 1080p. This is based off the assumption that users will be very satisfied if they receive content at the resolution they requested.

ReQoE based on Received vs. Desired Video Resolution					
Based on the MOS Lower Limit		Received			
		1080p	720p	480p	360p
Desired	1080p	4.3	4.0	3.6	3.1
	720p	-	4.3	4.0	3.6
	480p	-	-	4.3	4.0
	360p	-	-	-	4.3

Table 8: ReQoE based on Received vs. Desired Video Resolution: Lower Limit

ReQoE based on Received vs. Desired Video Resolution					
Using the MOS Upper Limit		Received			
		1080p	720p	480p	360p
Desired	1080p	4.5	4.3	4.0	3.6
	720p	-	4.5	4.3	4.0
	480p	-	-	4.5	4.3
	360p	-	-	-	4.5

Table 9: ReQoE based on Received vs. Desired Video Resolution: Upper Limit

ReQoE based on Received vs. Desired Video Resolution					
Based on the Average MOS		Received			
		1080p	720p	480p	360p
Desired	1080p	4.4	4.2	3.8	3.3
	720p	-	4.4	4.2	3.8
	480p	-	-	4.4	4.2
	360p	-	-	-	4.4

Table 10: ReQoE based on Received vs. Desired Video Resolution: Average

4.5 Optimizing a Wireless Access Network through Transcoding

We consider a scenario that N video streams are sent to the users of an access point, and optimize the ReQoE of the video streams. If there is insufficient wireless bandwidth to transmit the N video streams at the quality that the users request, then one or more of the video streams will be rerouted to the transcoder for rate adaptation. Due to the design of the system, there are two primary constraints. First is the wireless channel bandwidth which limits the maximum throughput of the access point. The second constraint is the computational power available to the transcoder. If a virtual instance of a transcoder is deployed within the CloudEdge, this would be a variable limit, dependent upon the shared resources available. If, on the other hand, a dedicated transcoder is deployed within the access network, there would be a fixed limit.

Adaptive modulation and forward error correction are typically used by the access point, to achieve reliable throughput even under less than ideal wireless conditions. The transmission rate that a user gets depends on their SNR. For example, for 802.11g, the user data rate will be 6, 9, 12, 18, 24, 36, 48, or 54 Mbps, depending on the channel SNR.

4.5.1 Transcoding Parameters

Assume that video stream $\boldsymbol{i}$ is sent to user $\boldsymbol{i}$ at data rate $\boldsymbol{r_i}$. Let $\boldsymbol{\alpha_i}$ denote the channel utilization, i.e., the share of time that the AP/BS uses the wireless channel to transmit video stream $\boldsymbol{i}$. Then the throughput of stream $\boldsymbol{i}$ is $\boldsymbol{T_i = \alpha_i r_i}$. The data transmission should meet the wireless channel utilization constraint, that is:

$$\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{\alpha_i \leq 1} \quad \text{eq. 4.2.}$$

Assume $\boldsymbol{c_i}$ units of processing cycle are needed if stream $\boldsymbol{i}$ is transcoded. $\boldsymbol{x_i}$ is defined as a variable to indicate whether stream $\boldsymbol{i}$ is transcoded ($\boldsymbol{x_i = 1}$) or not ($\boldsymbol{x_i = 0}$). Then the transcoding constraint is:

$$\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{c_i x_i \leq C} \quad \text{eq. 4.3,}$$

where C is the total available processing cycles.

The desired video rate, i.e., the user requested rate for video stream $\boldsymbol{i}$ is $\boldsymbol{S_i^d}$, and the video rate after transcoding is $\boldsymbol{S_i^t}$. The transcoder can only reduce the video resolution, i.e., decreasing the data rate. Which means that:

$$\boldsymbol{S_i^t < S_i^d} \qquad \boldsymbol{if\ x_i = 1}\ (transcoding) \quad \text{eq. 4.4,}$$

and $\boldsymbol{S_i^t = S_i^d} \quad \boldsymbol{if\ x_i = 0}\ (no\ transcoding)$ eq. 4.5.

Let δ denote the protocol overhead, which includes the CO header, and the lower layer headers and overhead to transmit video data, then, the required bandwidth to transmit the video stream is:

$$\boldsymbol{T_i = \alpha_i r_i = \delta S_i^t} \text{ eq. 4.6.}$$

The objective is to optimize the total ReQoE for all video streams transmitted by the access point. Because the ReQoE is based on the received vs. desired resolution, and the resolution translates to required bandwidth, the objective is as was stated above in eq. 4.7 to:

$$\boldsymbol{Maximize} \sum_{\boldsymbol{i}} \boldsymbol{ReQoE_i(\alpha_i, S_i^t)} \text{ eq. 4.7,}$$

subject to wireless channel constraint eq. 4.2, transcoding constraint eq. 4.3, as well as constraints shown in eq. 4.4, eq. 4.5, and eq. 4.6.

In general, ReQoE is a nonlinear function of S_i^t or α_i. It is thus a nonlinear programing problem, which is Non-deterministic Polynomial-time (NP) hard. This dissertation considers a set of the standard video resolutions and for each video resolution, there is a targeted streaming data rate as recommended by YouTube [47], which is shown in Table

6. To calculate the optimized variables, a linear programing is used to determine the allocation of wireless channel utilization and data rate for each video stream. Then, if there is enough wireless bandwidth to meet the requirement of the video streams at the quality desired by the users, that is, constraints eq. 4.2, eq. 4.5, and eq. 4.6 are satisfied, no transcoding is necessary. The incoming video contents are directly sent to the users via the access point. Otherwise, a search is performed with one stream at a time assigned a lower resolution until the optimal solution. Note that certain video streams may be dropped if the access point is unable to support the data rate required for the minimum video resolution (360p).

4.5.2 CloudEdge Transcoding Process

Figure 15 shows the steps taken by the CloudEdge network, through the data collection, calculations, and configurations that the CloudEdge SDN control performs in real-time, whenever a change occurs (e.g., add/drop video stream or a significant change to a user's SNR) to optimize the average ReQoE of the access network it is controlling. First, $\boldsymbol{N}$ users request video streams, then the controller queries the AP for the link data rate of each user while simultaneously querying the transcoder for its available resources.

Next, the controller uses the queried information to calculate the settings to optimize the average ReQoE of all the users. Finally, the controller redirects the traffic flows that are to be transcoded to the transcoder, configures the transcoder with the video transcoding parameters of each stream, and configures the AP channel utilization for each user.

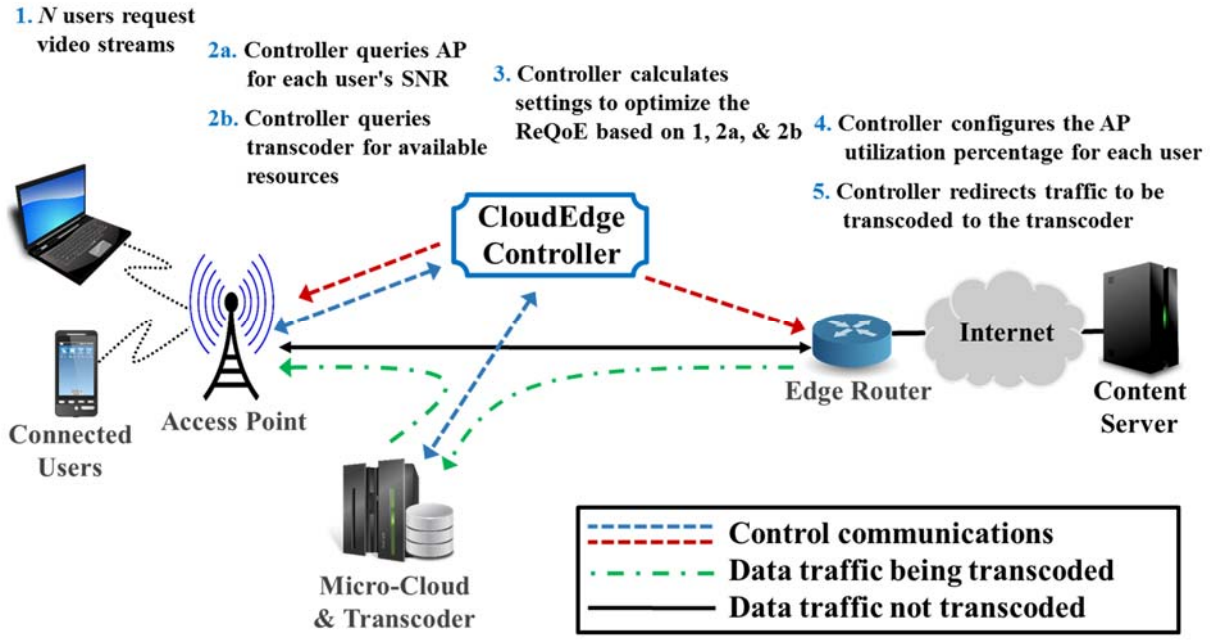

Figure 15: CloudEdge Operational Diagram [33]

4.6 Analytical Methods

The different methods used to analyze the access network model designed here are presented the following sections. The first method was generated to present a baseline and show the impact of users requesting higher than available data rates on ReQoE. The

second is a basic attempt at optimizing the throughput of the access point with in-network transcoding to observe the initial benefits this simple calculation would provide. The third method presented is a heuristic algorithm that iterates itself whenever making a change to the resolution of a data stream so that any residual resources resulting from that change may be given to another user, if possible, to improve the average ReQoE of the access network. Then, the method of exhaustive searching is used to validate the results of the other optimization methods. While it is very computationally expensive, in some cases it does produce significantly higher average ReQoEs. However, this method is not without draw backs, which can be observed in the results of the last method. The final analytical method, used to evaluate this model and compare the results of the different optimization approximation techniques, is to calculate the Jain's Fairness Index [48] of each of the three optimization approximation techniques.

4.6.1 Baseline ReQoE (i.e., No Transcoding)

In order to establish a baseline for the comparison of a CloudEdge network that can transcode users' video streams and a basic edge network that cannot, the effect of the packet loss on the average ReQoE of a video stream is considered. If there is no

transcoding to adjust the video rate, the data that is above the AP/BS wireless channel capacity would be dropped. In [24], the authors present a function for mapping the quantitative relationship between QoE score and packet loss, and give the parameters based on the curve fitting of actual user provided data points. The function described in [24] was slightly modified in order to raise the initial QoE value to 4.5, which is the highest potential QoE MOS based on the ITU definition shown in Table 7. This modified packet loss equation is then used to calculate the impact of packet loss on the QoE value of each data stream.

Original Packet Loss Equation: $\boldsymbol{QoE = 3.010 * e^{-4.473 * Packet\ Loss} + 1.065}$ [24]

Modified Packet Loss Equation: $\boldsymbol{QoE = 3.010 * e^{-4.473 * Packet\ Loss} + 1.49}$ eq. 4.8

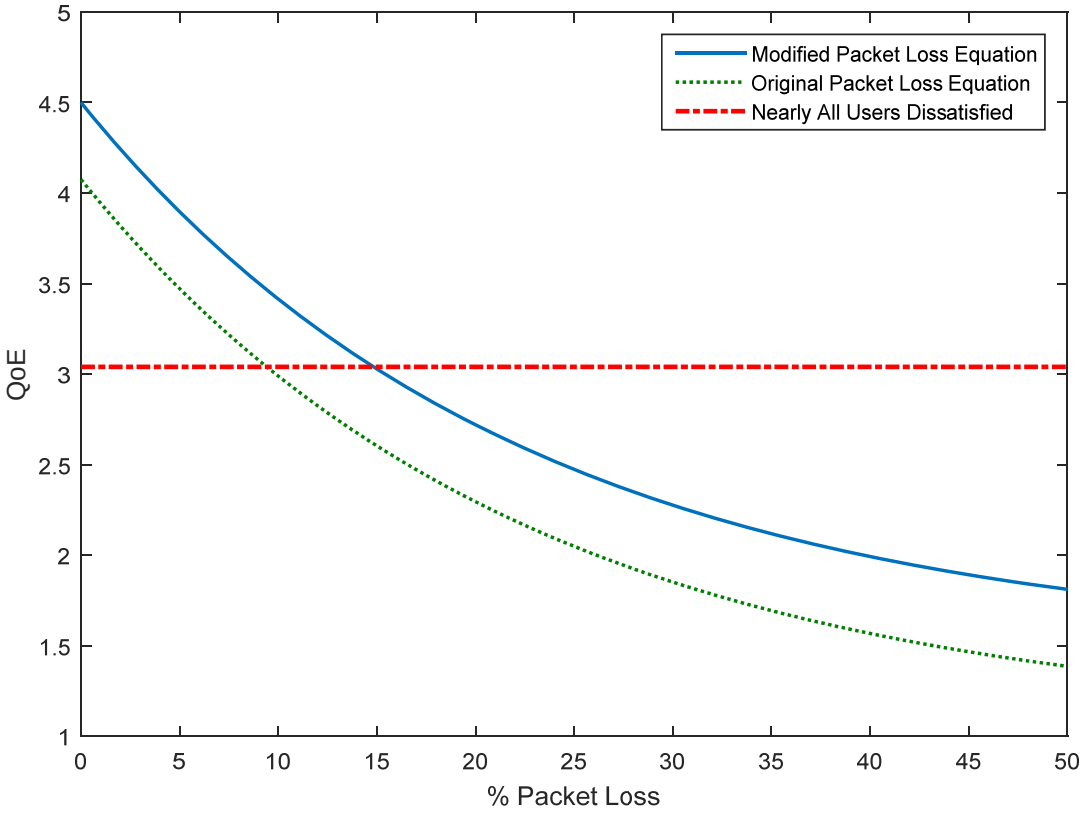

Figure 16: Original Packet Loss Equation and Modified Packet Loss Equation

4.6.2 Optimizing the Total Access Point Throughput

Optimizing the total access point throughput is a computationally simple approach to solving the problem presented in this model. This method was implemented to determine how much it could be improved upon, if at all. It also provided a first glimpse at the benefits of implementing the proposed CloudEdge architecture in terms of enabling in-network transcoding. Although the goal is to optimize the average ReQoE of

the access network, rather than the throughput, this method was considered because the resolution the users receive vs. the resolution requested, is directly related to the bandwidth each user is allocated and their SNR. The throughput maximization problem can be formulated as:

$$\boldsymbol{Maximize} \sum_{i} \boldsymbol{\alpha_i r_i} \quad \text{eq. 4.9}$$

$$\boldsymbol{Subject\ to\ the\ constraints\ of\ equations\ 4.2, 4.3, 4.4, 4.5, and\ 4.6.}$$

The throughput maximization calculations were performed using Linear Programing, with the following characteristics:

Objective: $\boldsymbol{Maximize\ T_i = r_A * \alpha_1 + r_B * \alpha_2 + r_C * \alpha_3 \ldots + r_i * \alpha_i}$

Constraints:

$\boldsymbol{\alpha_1 + \alpha_2 + \alpha_3 \ldots + \alpha_i \leq 1}$ — Limit maximum utilization to 100%

$\boldsymbol{\alpha_1 \geq 0, \alpha_2 \geq 0, \alpha_3 \geq 0, \ldots \alpha_i \geq 0}$ — Restrict utilization to only positive values

$$\begin{aligned} r_A * \alpha_1 &\geq S_A^t \\ r_B * \alpha_2 &\geq S_B^t \\ r_C * \alpha_3 &\geq S_C^t \\ &\vdots \\ r_i * \alpha_i &\geq S_i^t \end{aligned}$$

Calculate utilization rates to support each desired data rate

$\boldsymbol{S_i^d}$ is the desired rate, $\boldsymbol{S_i^t}$ is the transmitted rate, $\boldsymbol{S_i^t \leq S_i^d}$

This Linear Program attempts to find an $\boldsymbol{\alpha_i}$ (e.g., utilization percentage) for each user such that $\boldsymbol{r_i * \alpha_i \geq S_i^d}$. Once complete, each $\boldsymbol{S_i^t}$ is compared with each initial $\boldsymbol{S_i^d}$ and if

$S_i^d < S_i^t$ then the model knows that user must be transcoded in order to be successfully transmitted. Finally, the ReQoE score for each requested stream, those that require transcoding, those that do not, and those which are dropped, is calculated and used to determine the average ReQoE of the access network. The pseudocode for this algorithm is presented below:

```
Algorithm 1: Throughput Maximization (3 stream/user case)
%APRate = AP Data Rate (based on each user's SNR)
%DDRate = Desired Data Rate (of Request Video)
%RDRate = Requested Data Rate (changeable rate)
%FDRate = The Final Data Rate after optimization

1. Run the LP Function to find alpha:
fn = [-User1APRate; -User2APRate; -User3APRate; 0]
A = [1 1 1 1; User1APRate 0 0 0; 0 User2APRate 0 0;
     0 0 User3APRate 0]
b = [1; User1RDRate; User2RDRate; User3RDRate]
lb = [0; 0; 0; 0;]
[alpha,fmax] = linprog(fn,A,b,[],[],lb)
```

4.6.3 A Heuristic Model for Optimizing the Average ReQoE

In order to improve upon the basic throughput maximization, a heuristic algorithm was engineered to attempt to optimize the average ReQoE. It was designed to calculate an alpha for each user based upon the constraints of the system and iterating, after changing the requested rate of one of the users, if necessary, to improve the average ReQoE. Each time the function is iterated, the results are analyzed for specific conditions

that could be manipulated in order to optimize the final outcome. Only one variable is ever changed at a time, then each user's alpha is recalculated, and the process begins again by analyzing the new results. This process continues until the model can no longer improve the average ReQoE.

The heuristic algorithm attempts to fairly distribute the available AP utilization by only lowering one value at a time before reassessing the average ReQoE of the AP. It was designed so that it would not transcode any user more than once, until all other users had been transcoded once. Since users with lower SNRs require higher AP utilization to receive the same bandwidth as users with higher SNRs, the user with the lowest SNR, that has not yet been transcoded, is the user who is transcoded. In the case a user must be transcoded and multiple users have the same parameters, then the last user to join the AP, that has yet to be transcoded, is the user who is transcoded. Because the QoE rating drops very sharply when no video is received, this algorithm's primary goal is to ensure as many users receive video as possible. Its secondary goal is to provide the best video resolution possible, within the confines of the systems constraints. The pseudocode for this algorithm is presented below:

Algorithm 2: Heuristic Iterative ReQoE Optimization

```
% N: total number of video streams
% S_i^d: Desired video rate, i.e., the user requested
%   rate for video stream i
% S_i^t: Data rate after optimization or transcoded
%   video rate (0 means video stream i is dropped)
% r_i: Link data rate between AP and user i
% δ: Protocol and header overhead
% α_i: Channel utilization of stream i
% c_i: Unit of processing cycle needed for transcoding
% x_i: transcoded (x_i = 1) or not (x_i = 0)
% C: the total available processing cycles

1 for i = 1:N % Initialization
2     α_i = δ S_i^d / r_i;; S_i^t = S_i^d;
3 end
4 If Σ_{i=1}^{N} α_i ≤ 1 then
5   break;      % no transcoding
6 else
7   sort the users' link data rate, i.e., r_i ≤ r_j, if i < j;
8   i = 1;
9   while (1)
10    if Σ_{i=1}^{N} c_i x_i ≤ C      % available transcoding cycles
11      reduce stream i resolution by one level, S_i^t = S_i^t − 1;
12    else
13      drop stream i, S_i^t = 0;
14    α_i = δ S_i^t / r_i;
15    If Σ_{i=1}^{N} α_i ≤ 1 then
16      break;
17    else i++;
18    if i > N then i = 1;
19    end
20  end
```

Note: This lowers all users' data rates once before lowering a single user's rate twice.

4.6.4 Optimal ReQoE found through Exhaustive Searching

The problem of optimizing the ReQoE for a wireless access network is a non-linear programing problem with multiple variables and multiple constraints. Therefore, finding the maximum achievable average ReQoE for a given scenario is very computationally expensive. However, it was necessary to determine how well the heuristic algorithm performed. This was accomplished without searching based on percent utilization because there are too many possibilities, instead the fact that there are fixed resolutions available and known fixed AP data rates was exploited to greatly limit the scope of the search. Because of these discrete data rates and discrete resolutions, it was possible to perform an exhaustive search of all possible combinations of variables based on the constraints of a given scenario. Each user has a discrete access point data rate based on their SNR and this model only considered four possible data rates for video streams. This still resulted in a complex mathematical model which is exponentially computationally expensive and time consuming to run. The models presented in the results only go up to the case of 10 users requesting service, because in the implementation used to generate these results, simulating 10 users requesting 1080p content required over 12 GB of memory, simulating 11 users required over 49 GB or memory, and simulating 12 users required over 260 GB of memory.

In this model, there are only four possible video resolutions, plus the potential for a stream to be dropped, resulting in five possible data rates for each user. For example, with 3 data streams there are 153 possible configurations of the variables, for 4 data streams there are 777 configurations, and for 5 data streams there are 3901 configurations. The number of searches required for $\boldsymbol{N}$ users is:

$$\left(\sum_{i}^{N} \mathbf{5}^{\boldsymbol{N}} - \mathbf{1}\right) + \mathbf{5}^{\mathbf{3}} + \mathbf{3}^{\mathbf{3}} + \mathbf{1}, \qquad \boldsymbol{For\ N} \geq \mathbf{4}$$

eq. 4.10

$$\mathbf{5}^{\boldsymbol{N}} + \mathbf{3}^{\mathbf{3}} + \mathbf{1}, \qquad \boldsymbol{For\ N} \leq \mathbf{3}$$

This method has the potential to produce different data sets with the same average ReQoE, because the exhaustive search function tries every possible combination of variables, then, out of the valid scenarios where $\sum_{i} \boldsymbol{\alpha_i} \leq \mathbf{1}$, it finds the highest resulting average ReQoE as its result. The result of this exhaustive search will be referred to as the Optimal ReQoE.

4.6.5 Jain's Fairness Index

The Jain's Fairness Index [48] is a mathematical tool for calculating how evenly different users or processes are treated within a particular scenario. The Jain's Fairness index was developed to be metric and scale independent, continuous, and bounded between 0 (0%) and 1 (100%). It is calculated as follows:

$$\mathcal{J}(\boldsymbol{x_1}, \boldsymbol{x_1}, \ldots \boldsymbol{x_N}) = \frac{\left(\sum_{i=1}^{N} x_i\right)^2}{N * \sum_{i=1}^{N} {x_i}^2} \quad [48]$$

Earlier in this dissertation, it was established a drop in resolution was directly related to a drop in QoE user satisfaction rating, therefore Jain's Fairness Index can be directly calculated based on the individual users' ReQoE MOSs. For an added level of detail and for comparison, the Jain's Fairness is also calculated based on the final received data rate for each user.

4.7 Results

This section evaluates the performance of the proposed resource optimization algorithms and presents simulation results to demonstrate the impact of in-network

processing on video delivery quality. ***N*** video streams are sent to the users through a CloudEdge wireless access network. The radio access protocol is assumed to be IEEE 802.11g, and the link data rate for transmitting a video stream is determined using a practical link adaptation algorithm [46] based on the end user's channel SNR. As shown previously in Table 5, AP's operate at a limited number of discrete data rates, each corresponding to a range of SNR's. This means it is likely that several users will be connected at the exact same data rate. It is assumed that users will request a desired video resolution based on their device capability and that the sender is capable of sending the desired resolution.

First, each user's DeQoE is calculated, then, the CloudEdge attempts to fulfill that user's DeQoE. However, in cases where transcoding of one or more users is required to improve the average ReQoE of all users of the AP, the CloudEdge controller will send a lower resolution stream to some or all users. The QoE that each user receives is then the Realized QoE (ReQoE) that was discussed in Chapter 3.

4.7.1 Initial Findings Given Unlimited Transcoding Capability

Figure 17 shows the average ReQoE score versus the number of video streams with in-network transcoding and the heuristic algorithm, as well as without transcoding. In all four cases, the initial video resolutions, i.e., the resolutions desired by the users, are either all 1080p or all 720p. The assumption in this simulation is that the transcoder is able to process as many of the streams as necessary to achieve the highest average ReQoE. The impact of limited transcoding capacity will be shown later. In Figure 17, the simulation results show that when the number of video streams increases and there is not enough wireless capacity to transmit the video streams at their original quality/data rate, employing in-network transcoding to lower the video quality, which in turn lowers the data rate sent over the wireless channel, can greatly improve the average ReQoE. Otherwise, when video streams suffer from packet loss, the ReQoE and the video quality degrade quickly. This clearly demonstrates one of the potential benefits of implementing CloudEdge, with the ability to perform in-network transcoding and resource management within a wireless access network.

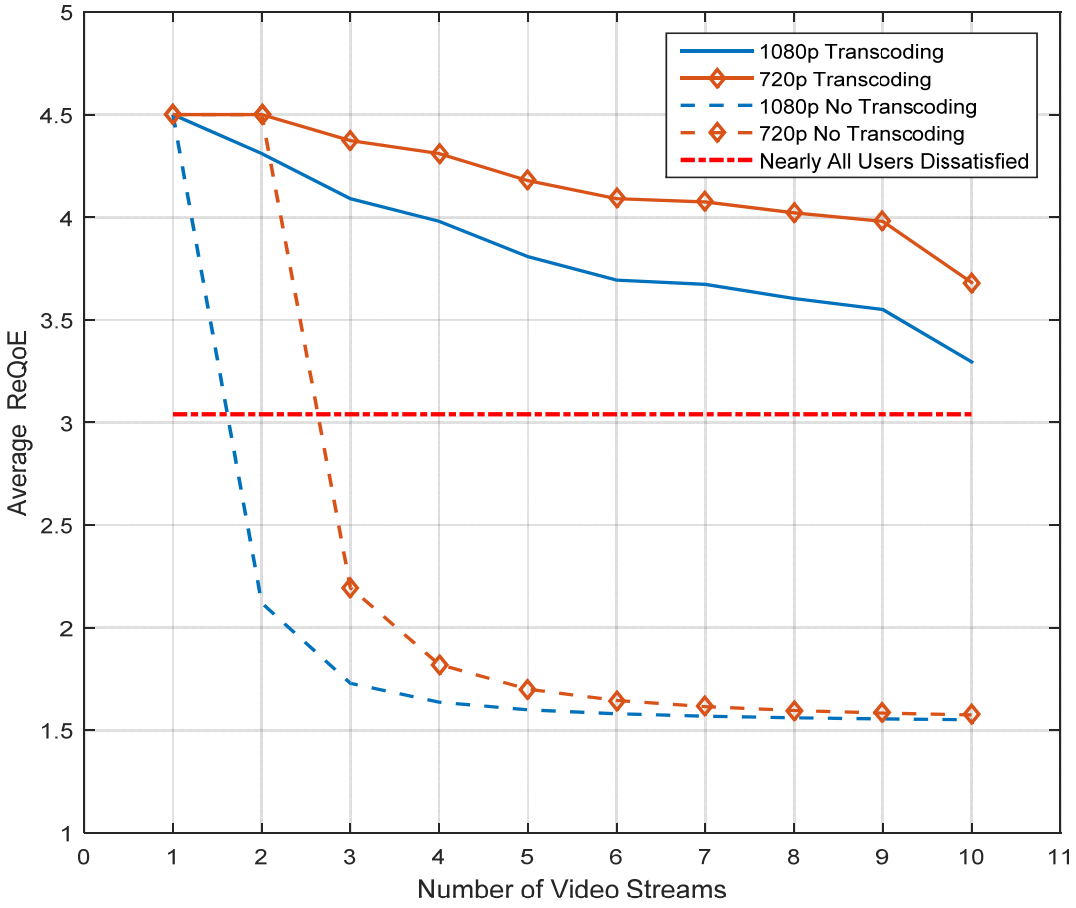

Figure 17: Average ReQoE of AP users connecting at the same data rate

Figure 18 is was generated using the same mathematical model as Figure 17, but in this case, the users are connecting to the AP at different data rates. This is a more realistic scenario, as it is unlikely that all users will all be connecting to an AP at the same data rate. One particularly interesting data point to note is in the 720p heuristic algorithm transcoding case, when user six connects to the AP; the average ReQoE drops sharply, but

it goes up when user seven joins. This phenomenon occurs because user six has a very low data rate and user seven has a very high data rate, therefore, when user seven joins and is able to receive a high ReQoE while requiring very low utilization, this offsets the high utilization required by user six to receive even a low ReQoE.

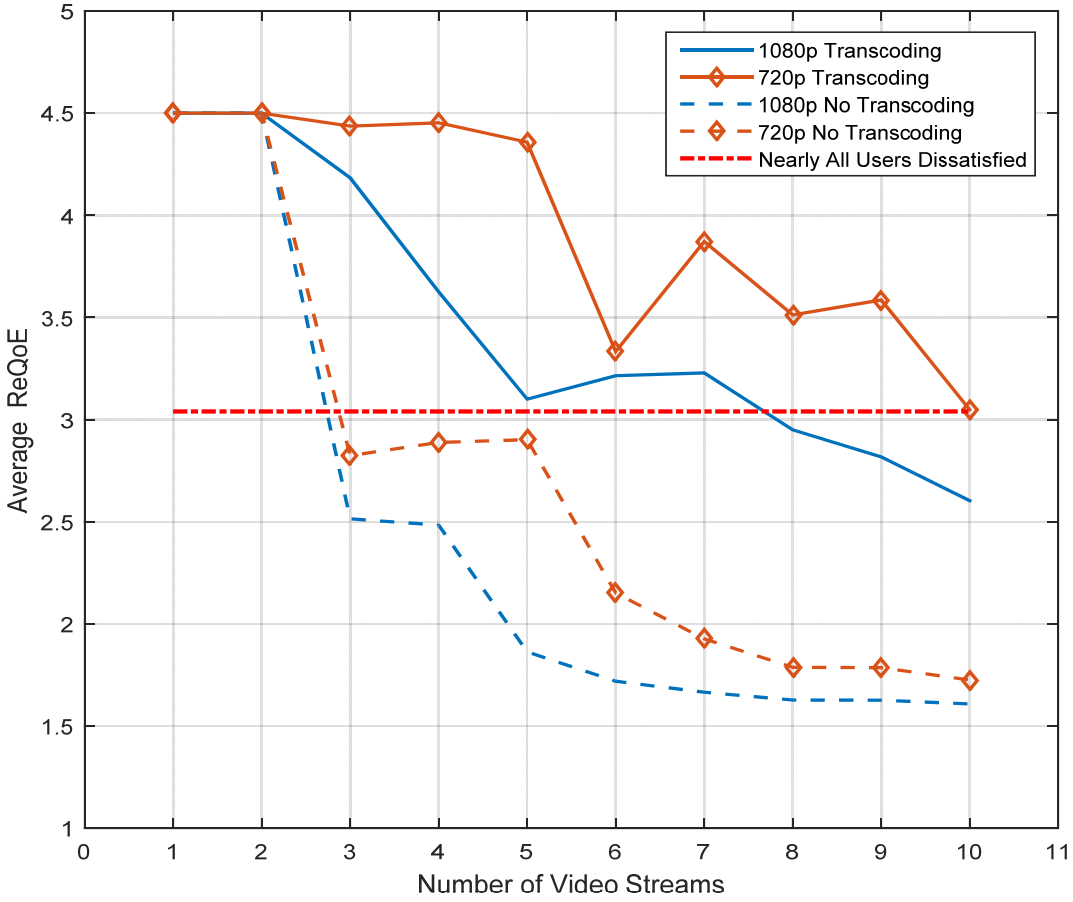

Figure 18: Average ReQoE of AP users connecting at various data rates

4.7.2 Limited Transcoding Capacity

Next is an investigation of the impact of limiting the transcoder computational power on the Average ReQoE when using the heuristic algorithm. Figure 19 and Figure 20 shows the average ReQoE vs. the maximum number of video streams the in-network transcoding service can process. In both cases there are total 12 video streams attempting to be received in this simulation, and the video resolutions being requested by all users are 1080p in the first scenario and 720p in the second. Figure 19 shows the results when using the throughput maximization algorithm. Because of how the algorithm works in the special case that all users are connecting at the same data rate, it evenly distributes the available bandwidth amongst all users. For this reason, each user, in both the 1080p and 720p simulation must be transcoded to achieve the highest average ReQoE, and any users who are not transcoded are dropped. Next, Figure 20 shows the results based on the heuristic algorithm. Here, in the 720p scenario, 8 streams must be transcoded in order for the average ReQoE to achieve its maximum potential. For the 1080p scenario, the optimal ReQoE is not achieved until all 12 streams are able to be transcoded.

Limitations on the transcoder can exist for many reasons. In the situation where a virtual transcoder is employed that is part of a cloud system, the total computational power of the cloud would be the absolute maximum limit, but the other services the cloud

is responsible for will also directly decrease the number of streams that can be transcoded. Although supporting a dynamic number of streams may seem like a significant disadvantage, the ability for the cloud to support additional services either temporarily or permanently, and also to support future applications, may prove it a valuable network resource.

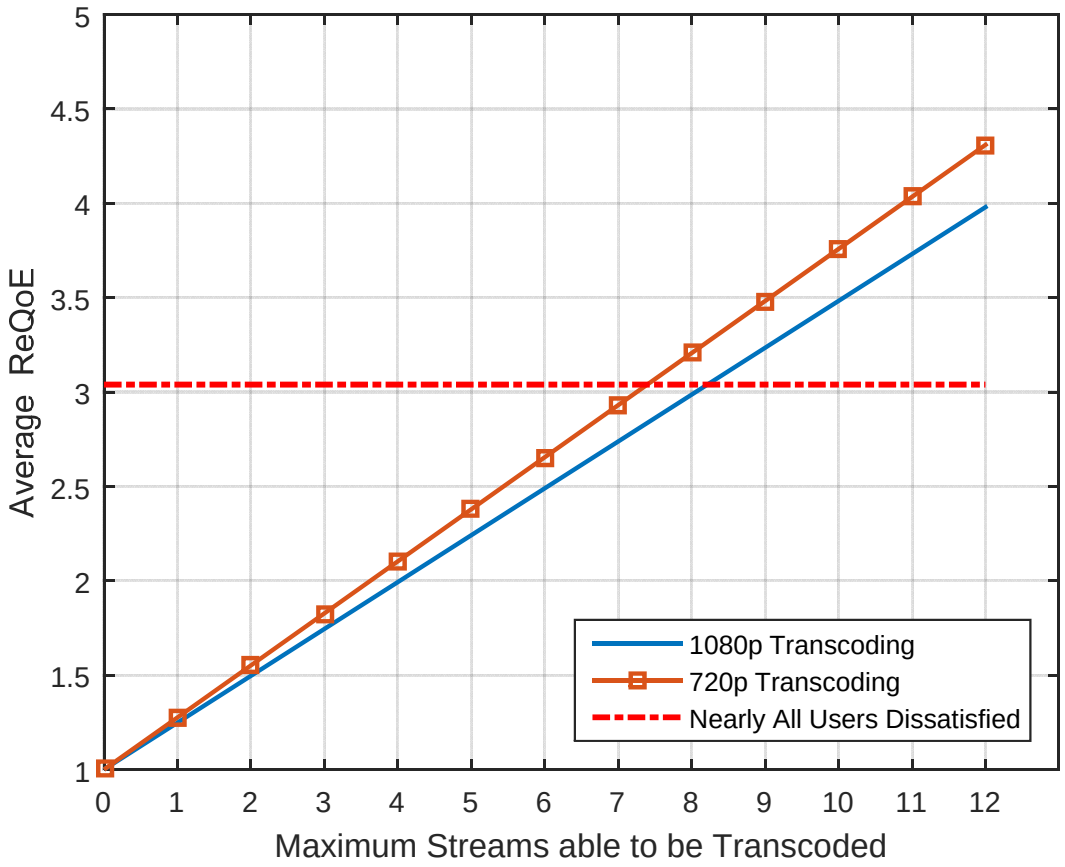

Figure 19: Average ReQoE of 12 AP users vs. the number of transcoders, using the Throughput Maximization

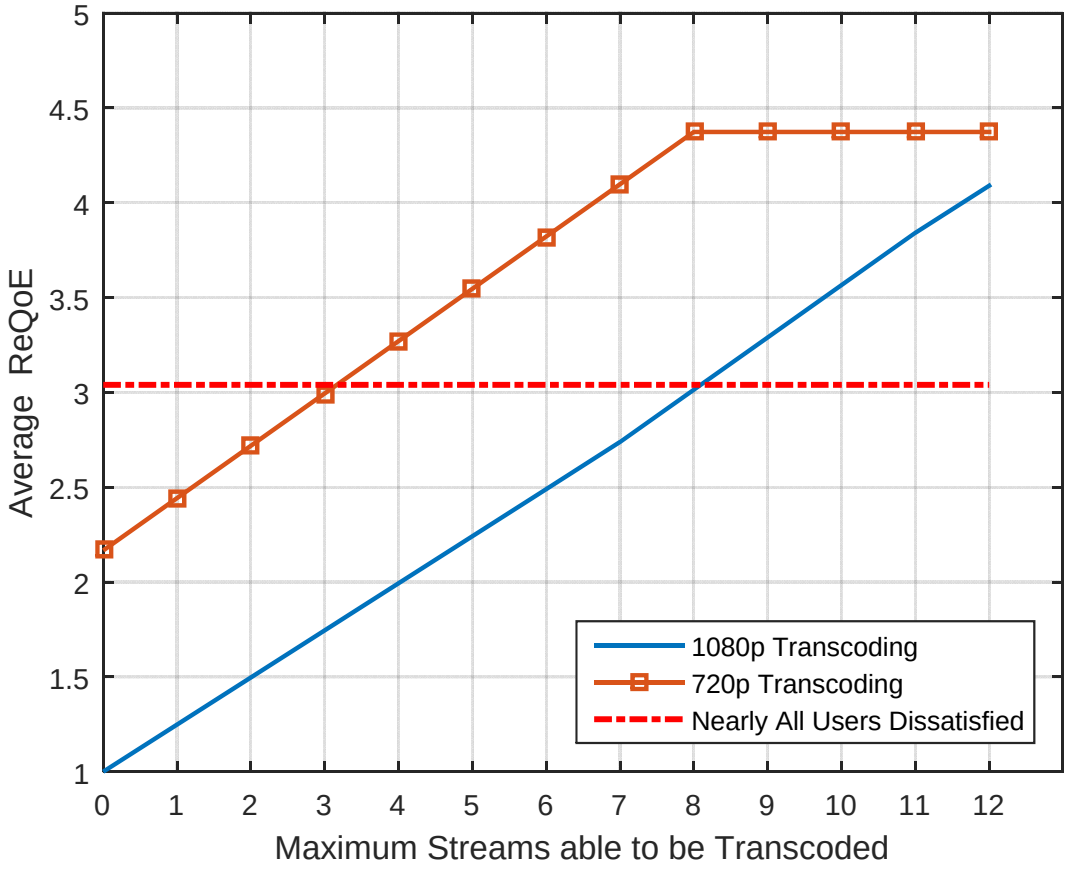

Figure 20: Average ReQoE of 12 AP users vs. the number of transcoders, using the Heuristic Algorithm

4.7.3 Comparison of Optimization Methods

Figure 21 and Figure 22 are both generated using the same model. The difference is that in Figure 21, all of the users connect to the AP at the same data rate, and in Figure 22, the users connect at various data rates. Each figure shows six different scenarios at the same time to provide a deeper insight into the inner workings of the model and the

potential benefits of implementing CloudEdge for in network transcoding. One can quickly observe that the lack of transcoding provides poor ReQoE for the users. And that the heuristic algorithm developed in this dissertation either performs equal to or better than the throughput maximization technique. In a few cases, the heuristic algorithm greatly surpasses the throughput maximization.

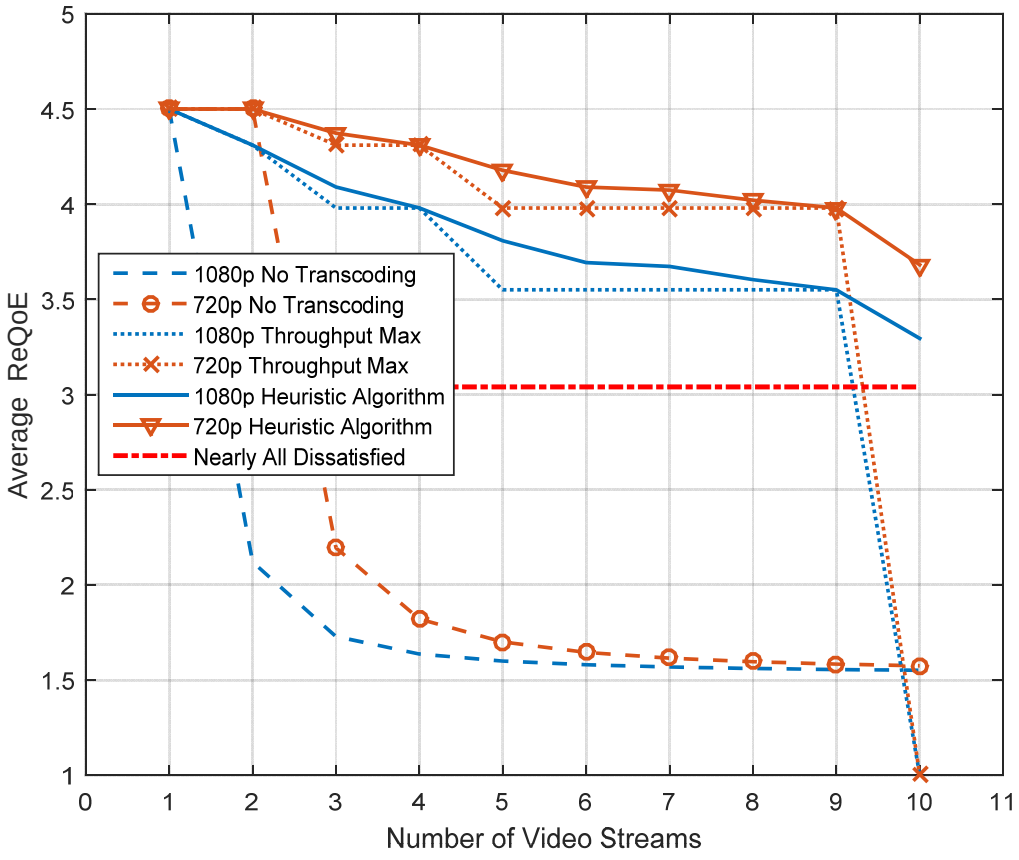

Figure 21: Average ReQoE of AP users connecting at the same data rate

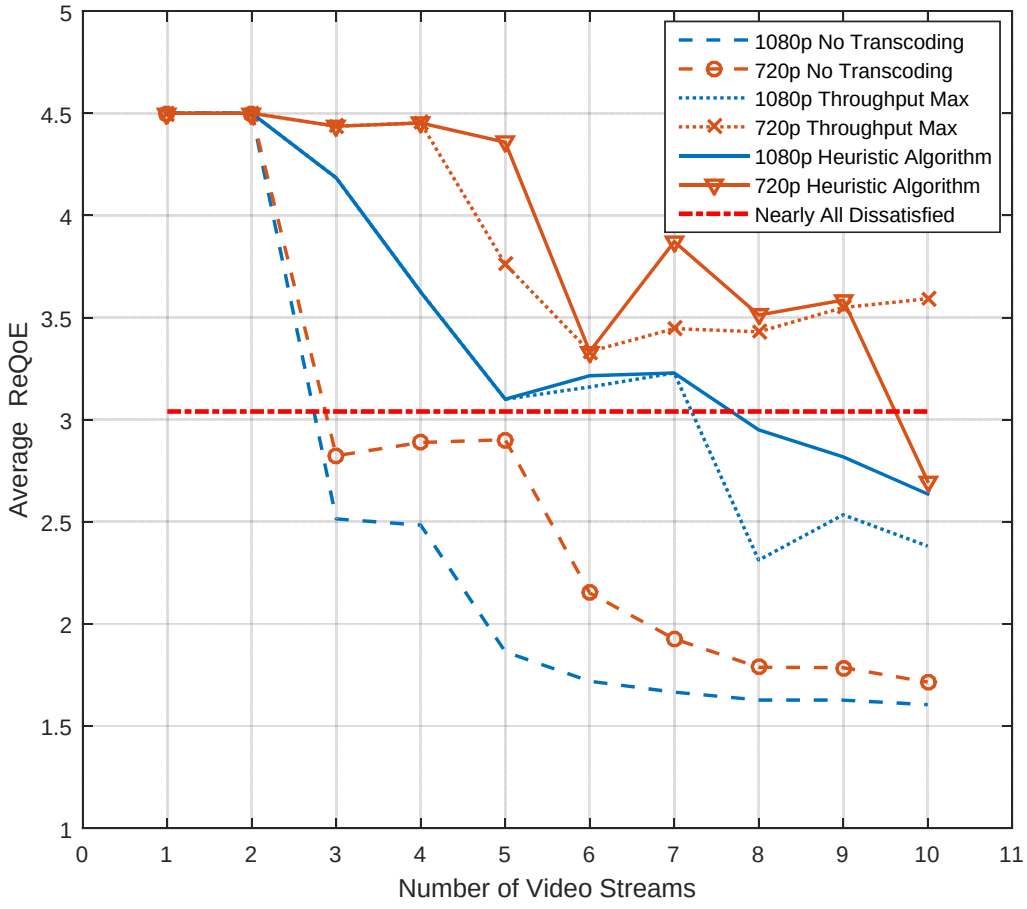

Figure 22: Average ReQoE of AP users connecting at various data rates

4.7.4 Comparison of Approximation Algorithms and the Optimal ReQoE

By performing an exhaustive search of all potential transcoding rates each user may receive, including the potential for requesting but receiving no video, the maximum potential or Optimal ReQoE can be found. Figure 23 and Figure 24 present the results of the Optimal ReQoE versus the two approximation algorithms, and no transcoding. An

interesting fact to note here is the plot lines in Figure 21 and Figure 23 appear to be the same, because the Optimal ReQoE matches up exactly with the results of the heuristic algorithm, in the special case that all users are connecting at the same data rate. This result was expected, as there are only a few discrete values for each user's ReQoE based on the requested vs. received video resolution, which can be directly related to each user's actual received data rate, and because receiving a video that has been transcoded several resolutions lower, results in a much higher QoE then receiving no video at all.

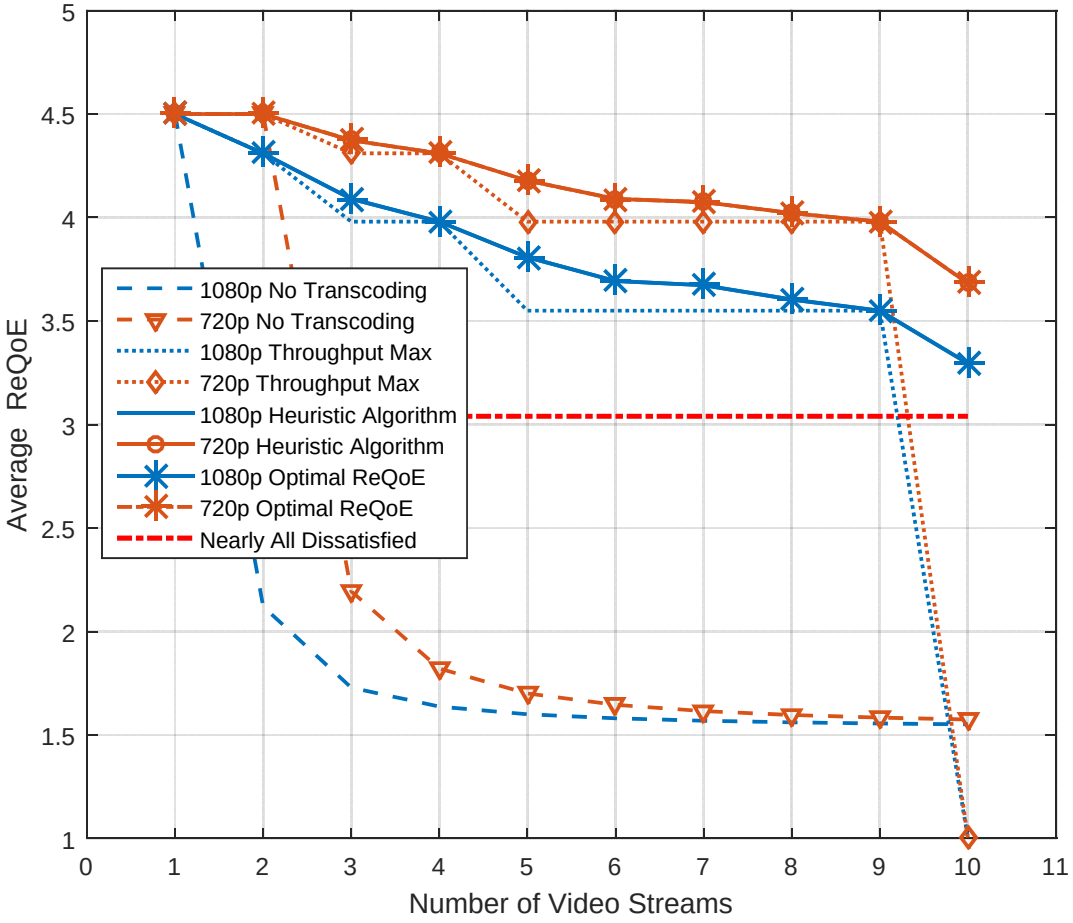

Figure 23: Average ReQoE of AP users connecting at the same data rate

The results are significantly different however when users are connecting at different date rates, as can be observed when comparing Figure 22 to Figure 24. The resulting Optimal ReQoE is initially aligned with the heuristic algorithm, but as more users join the AP the results diverge and the Optimal ReQoE is a higher average ReQoE in both the 1080p and 720p case. The Optimal ReQoE is higher because the exhaustive search is able

to find higher average ReQoEs because it looks at all potential combinations of variables for a given scenario, and often, the best results are achieved if one or more user's video streams are transcoded down several levels or even dropped, while other users are not transcoded at all. This does not occur with the heuristic algorithm, because its goal is to optimize the ReQoE while ensuring that the even the users connecting at the lowest data rates receives the best possible video, as long as doing so does not prevent another user from receiving the same or better quality video.

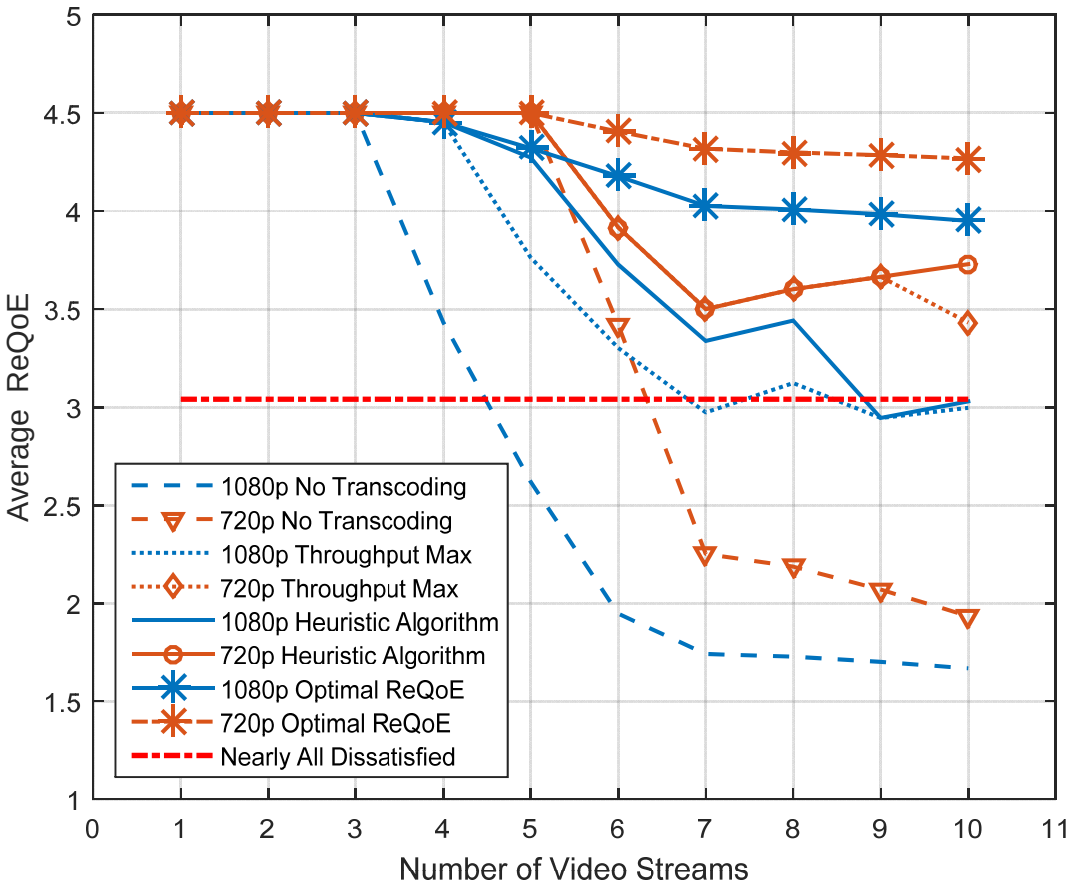

Figure 24: Average ReQoE of AP users connecting at various data rates

Figure 25 and Figure 26 show additional results for the Average ReQoE of an AP with all users connecting with a different set of various data rates. Here, the users requesting 720p content are shown in Figure 25 and the users requesting 1080p content are shown in Figure 26. These results are interesting; even the no transcoding case sees an improvement in average ReQoE when users 4 and 5 join the AP. This is because both

users are connected with a high data rate and are therefore able to achieve a high level of ReQoE even though there is a great deal of packet loss in this highly resource constrained environment. The Optimal ReQoE finds the highest average ReQoE, and the Heuristic Algorithm just slightly outperforms the Throughput Maximization, but it does so in all cases.

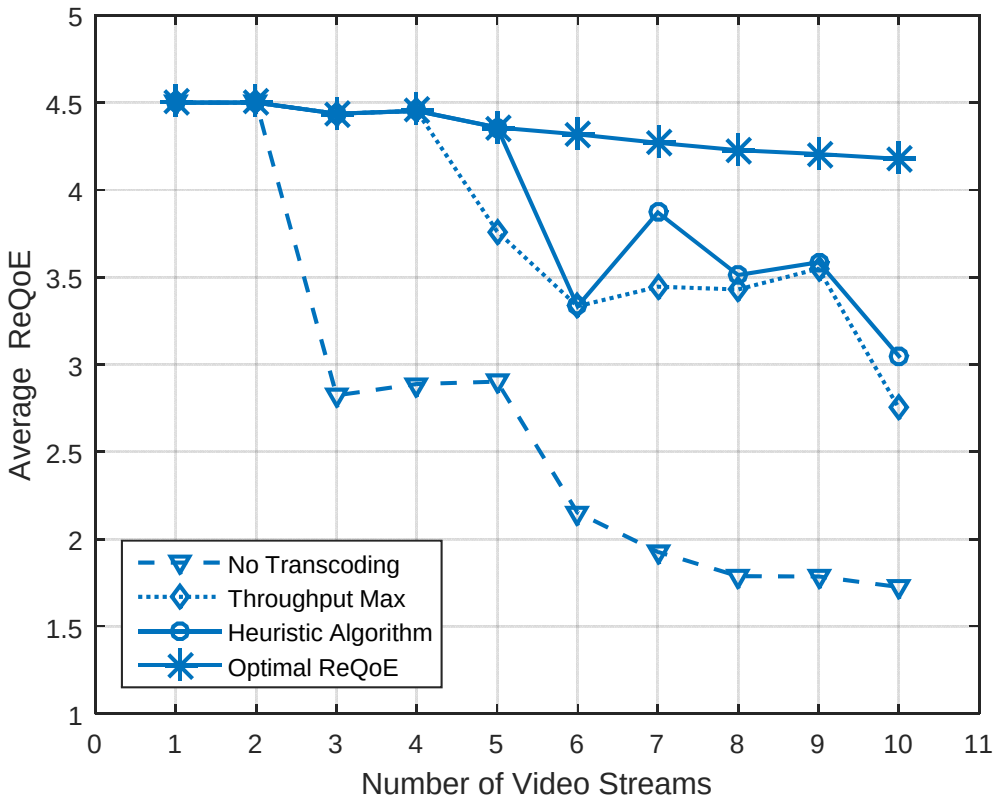

Figure 25: Average ReQoE of AP users requesting 720p content and connecting at various data rates

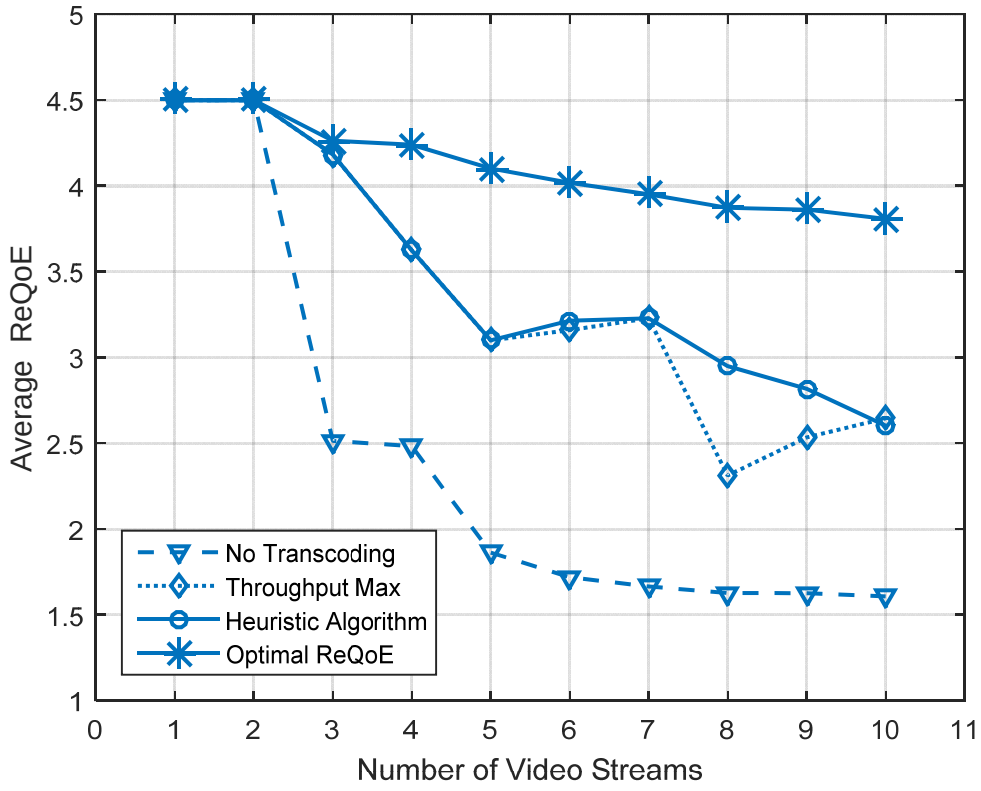

Figure 26: Average ReQoE of AP users requesting 1080p content and connecting at various data rates

Figure 27 and Figure 28 show the same data as Figure 24, but separate it out into one plot for the 1080p scenarios and one plot for the 720p scenarios in order to clearly see the differences between the different algorithms. Although the heuristic algorithm always meets or exceeds the performance of the throughput maximization, the optimal ReQoE successfully identifies values that will provide a higher average ReQoE. This becomes even more apparent as the access network becomes congested and more video streams are requested.

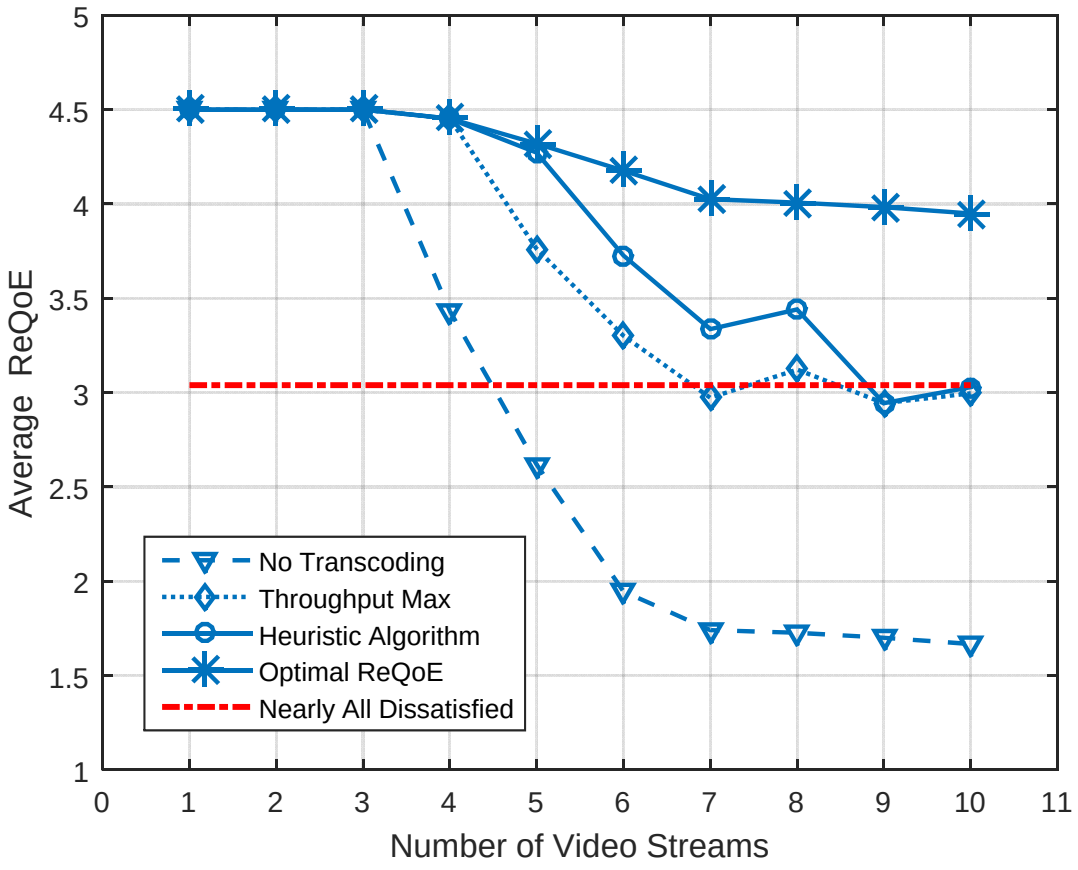

Figure 27: Average ReQoE of AP users requesting 1080p content, and connecting at various data rates

The average ReQoE of the heuristic algorithm drops quickly because it makes an attempt at being fair to all users, by transcoding all users once, before transcoding a single user twice. While the exhaustive search is capable of discovering combinations of data rates, the heuristic algorithm cannot, which results in the optimal ReQoE which is a higher average ReQoE. An example of such a case, is when a few users' low SNRs result

in very low data rates, then in order to achieve the highest average ReQoE for the AP, several users must be transcoded down three times, while other users are not transcoded at all.

Next, Figure 28 and Figure 29 show additional results using a different set of data rates. In Figure 28, it is not until the tenth user that the heuristic algorithm out-performs the results of the throughput maximization; and the model with no transcoding is perfectly successful until the sixth user joins. This occurs because the data requirements for the 720p resolution are lower than 1080p and because the model starts off with high data rate users and each step adds a user at a lower data rate.

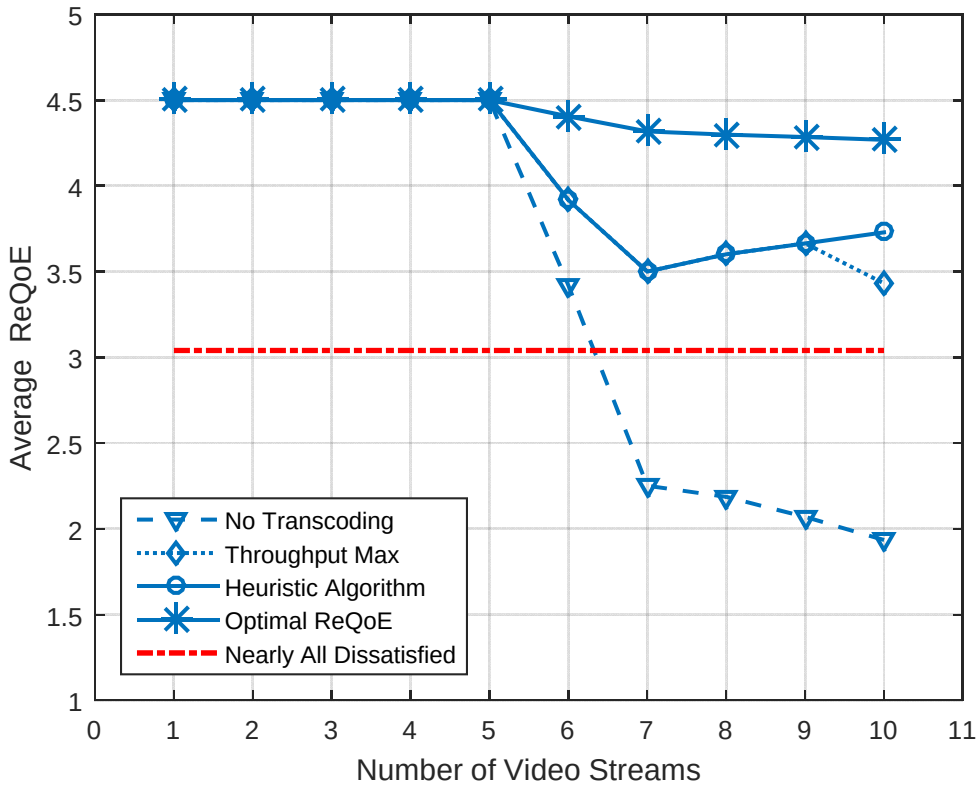

Figure 28: Average ReQoE of AP users requesting 720p content, and connecting at various data rates

The next several figures show the same set of algorithms, but with users joining at different data rates, starting with the initial users connecting with low data rates, and each new users joins at a higher data rate. In Figure 29, it is interesting to see the heuristic algorithm in action. As soon as the second user joins, it dramatically out-performs the throughput maximization and is nearly as good as the optimal ReQoE. With the third user on the network, the performance of the heuristic algorithm is hindered by its attempt

to be fair to all users, and results in the same average ReQoE as the throughput maximization. This happens several times again, for the 5, 6, and 7 user scenarios. In the four user scenario, the heuristic algorithm again significantly out-performs the throughput maximization, but now it is also much lower than the optimal ReQoE. And when the eighth user joins, the heuristic algorithm again out-performs the throughput maximization, but the average is still too low, with nearly all users being dissatisfied. This scenario was modeled because it represents a very challenging case in which there are always several users with very poor SNRs and therefore very low data rates attempting to stream data. The Optimal ReQoE performs well and even improves a bit from the three user case to the four; again, this is because the exhaustive search method used to find the optimal ReQoE will completely drop one or more users in order to give other users the highest ReQoE, when doing so results in the highest average ReQoE.

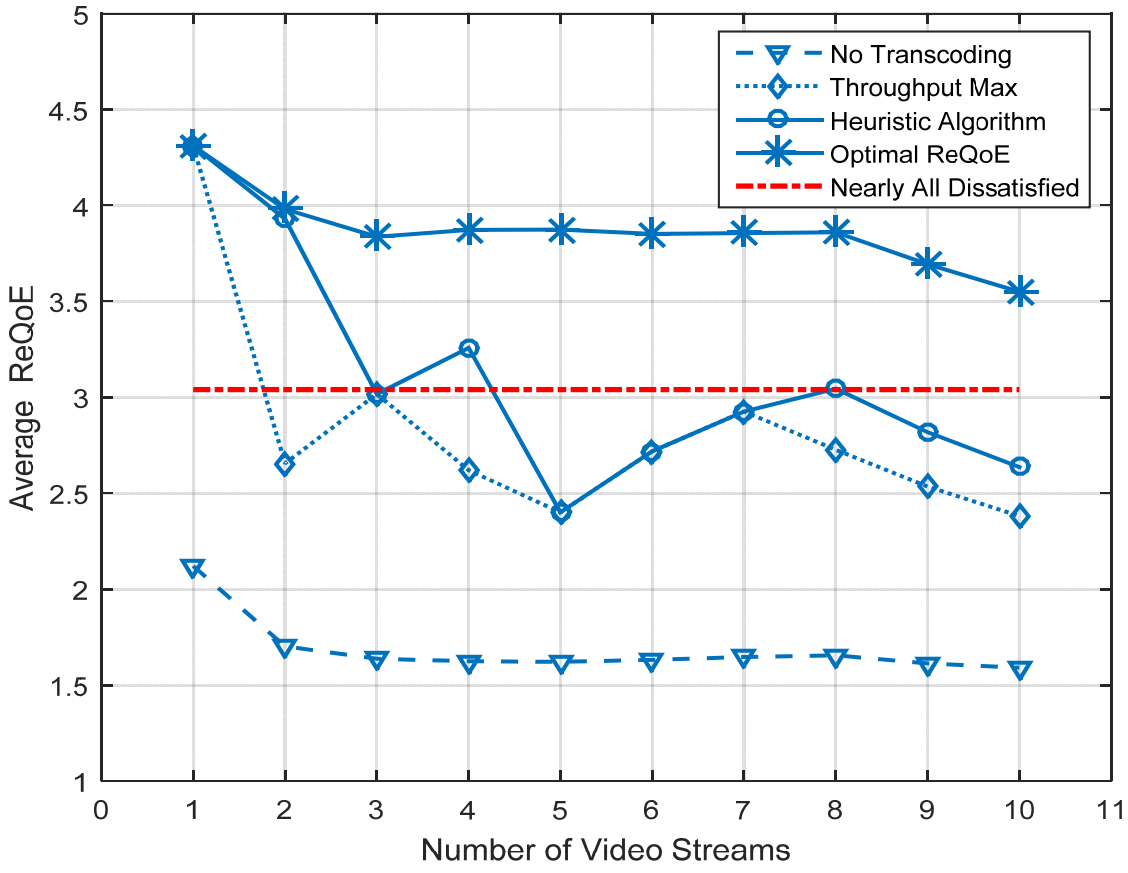

Figure 29: Average ReQoE of AP with new users requesting 1080p content, (first user connects at the lowest data rate, each additional user has a higher data)

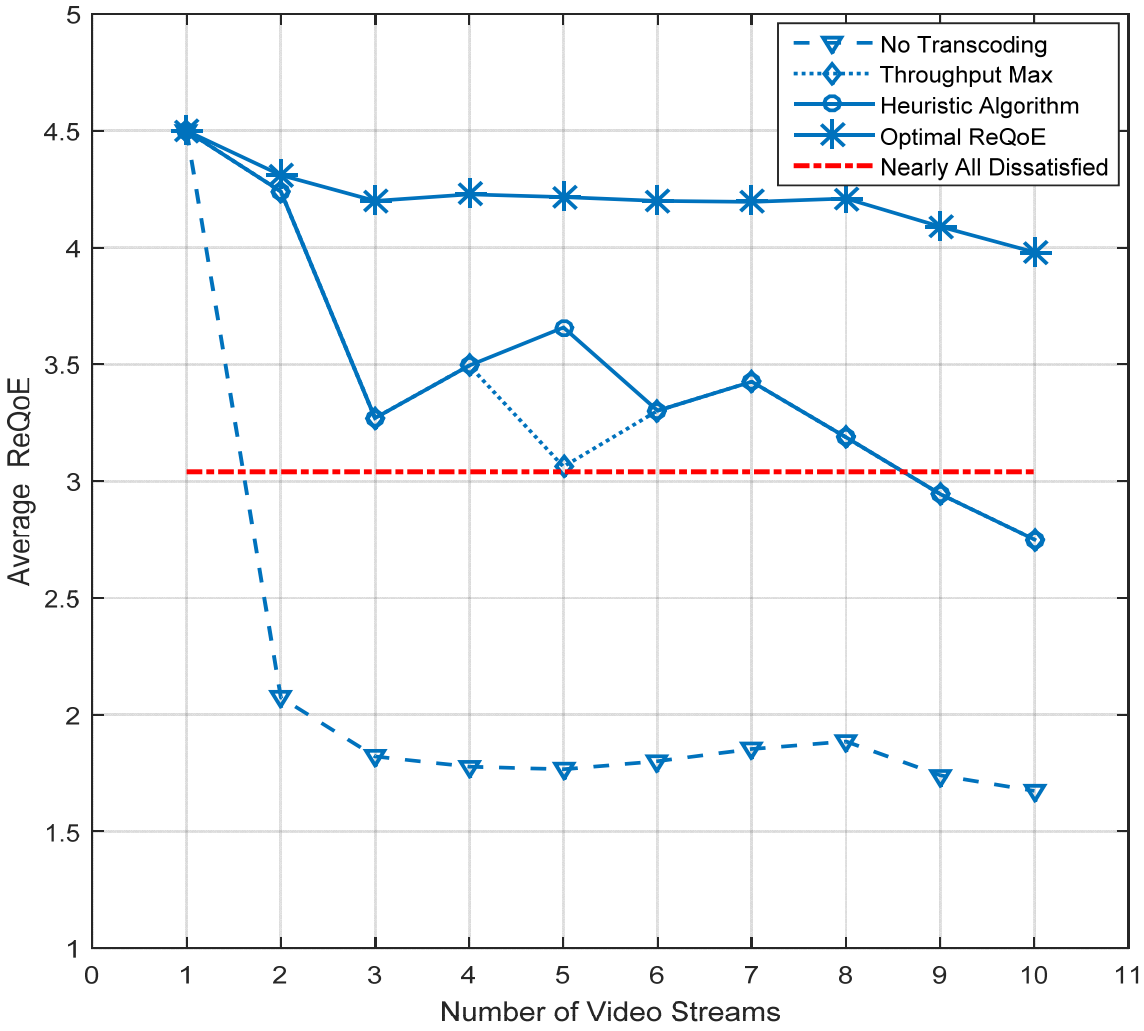

Figure 30: Average ReQoE of AP with new users requesting 720p content, (first user connects at the lowest data rate, each additional user has a higher data)

Figure 31 and Figure 32, below, show the average ReQoE of all AP users who are requesting 1080p and 720p content respectively, with the initial user having the highest possible data rate and each new user joins with a lower data rate. Once all data rates

have been used, the remaining users joining the network continue to join at the lowest possible data rate. In both cases, the Optimal ReQoE significantly out performs either of the approximation algorithms, after more than four of five users are simultaneously attempting to stream content. In Figure 31, both approximation algorithms and the no transcoding case provide the optimal ReQoE to the first three users to stream data through the AP. The throughput maximization algorithm continues to provide the optimal ReQoE for the first four users, then the heuristic algorithm begins to outperform it and produces nearly the same result as the optimal ReQoE for the first five users. In the 720p case, shown in Figure 32, the performance of both algorithms is the same. Additionally, it can be observed that both of the approximation algorithms and the no transcoding case, all provide the optimal ReQoE to the first five users of the AP until the sixth user requests content.

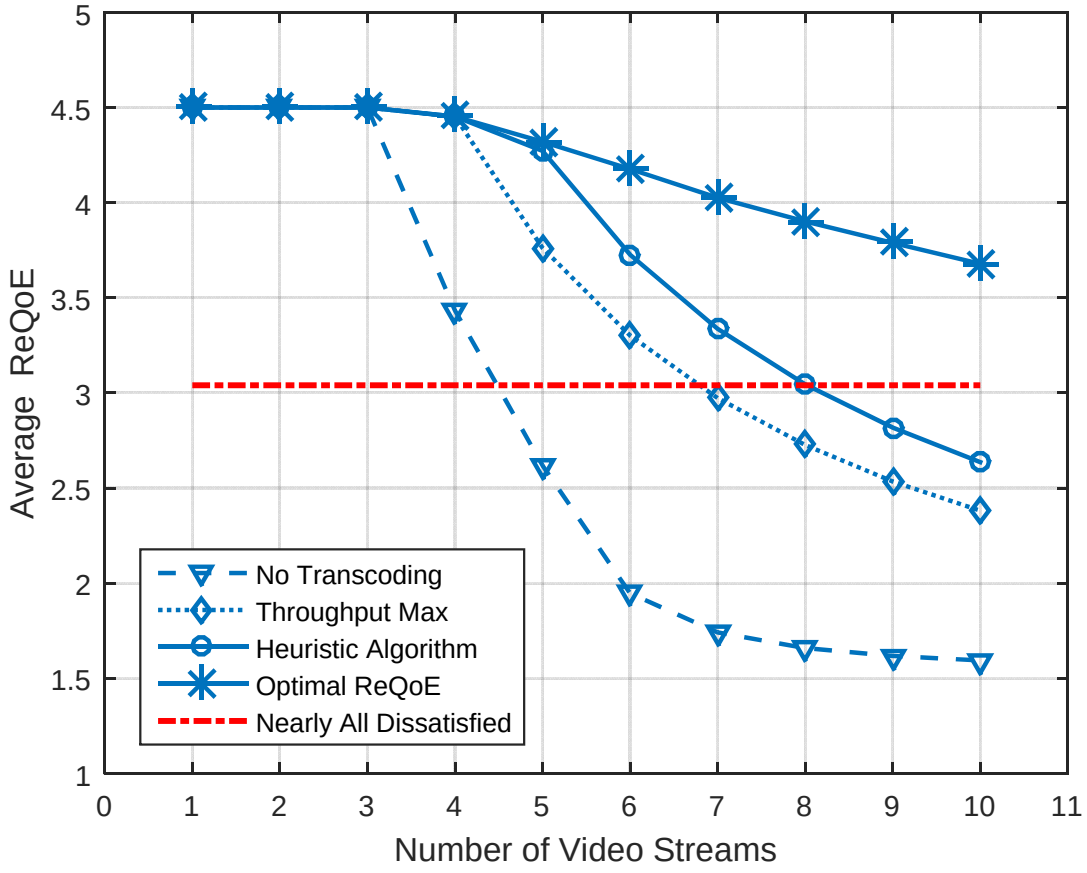

Figure 31: Average ReQoE of AP with new users requesting 1080p content, (first user connects at the highest data rate, each additional user has a lower data rate)

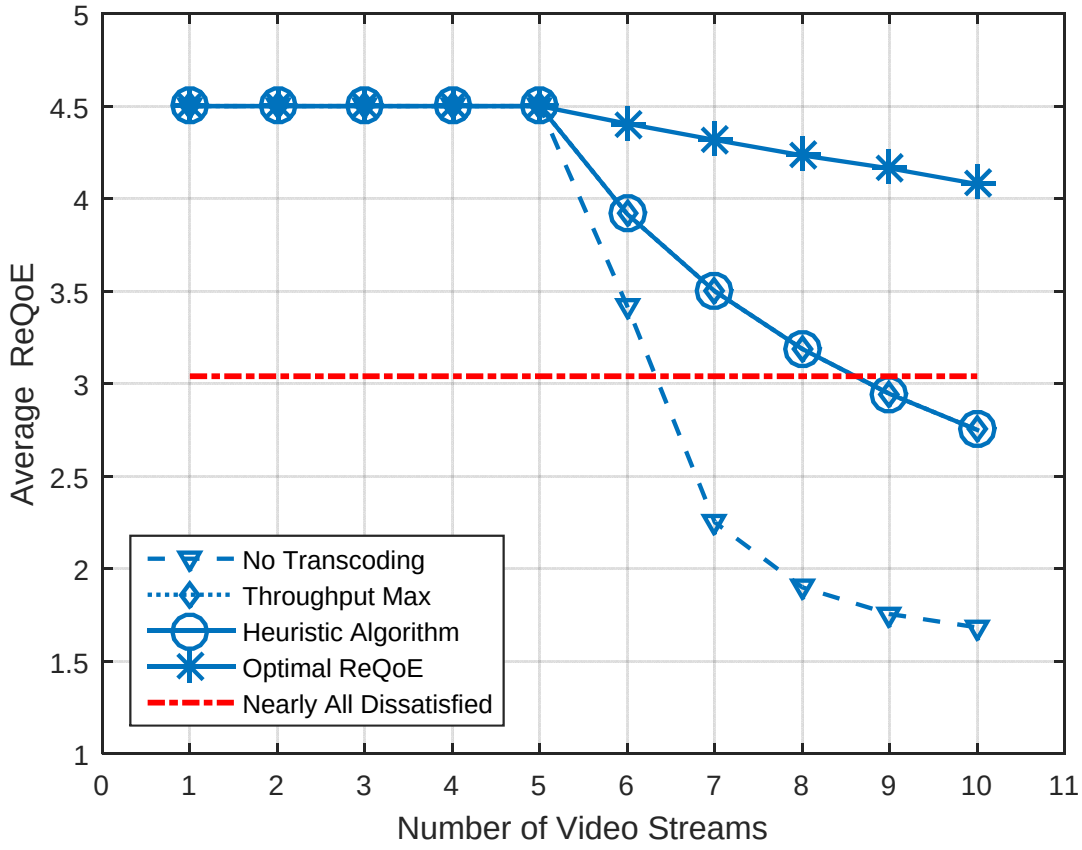

Figure 32: Average ReQoE of AP with new users requesting 720p content, (first user connects at the highest data rate, each additional user has a lower data rate)

The results shown here demonstrate the potential value of enabling access networks with computational, storage, and caching capabilities. Although the approximation algorithms are far from ideal, they are significantly better than no transcoding at all; and in most cases the heuristic algorithm performs much better than the throughput maximization. Because the heuristic algorithm is able to provide the optimal ReQoE in

the special case of all users connecting at the same data rate, it is believed that with further refinement, it may provide significantly better results while still not coming close to the computational power necessary to perform the exhaustive search required to find the optimal ReQoE in all cases.

4.7.5 Impact of Transcoding on Individual Users

The following set of three graphs, Figure 33, Figure 34, and Figure 35, are the results of a single simulation. They show additional details for the output of the heuristic algorithm with users connecting at the same data rate and requesting either 720p or 1080p streams. The plot lines on Figure 33 show the average ReQoE of the AP, the details of which are shown in Figure 34 and Figure 35. These bar graphs show each user's ReQoE and the effect on existing users as additional users join the access network. Because all users are initially requesting the same rate, it can be observed which users are transcoded or dropped first.

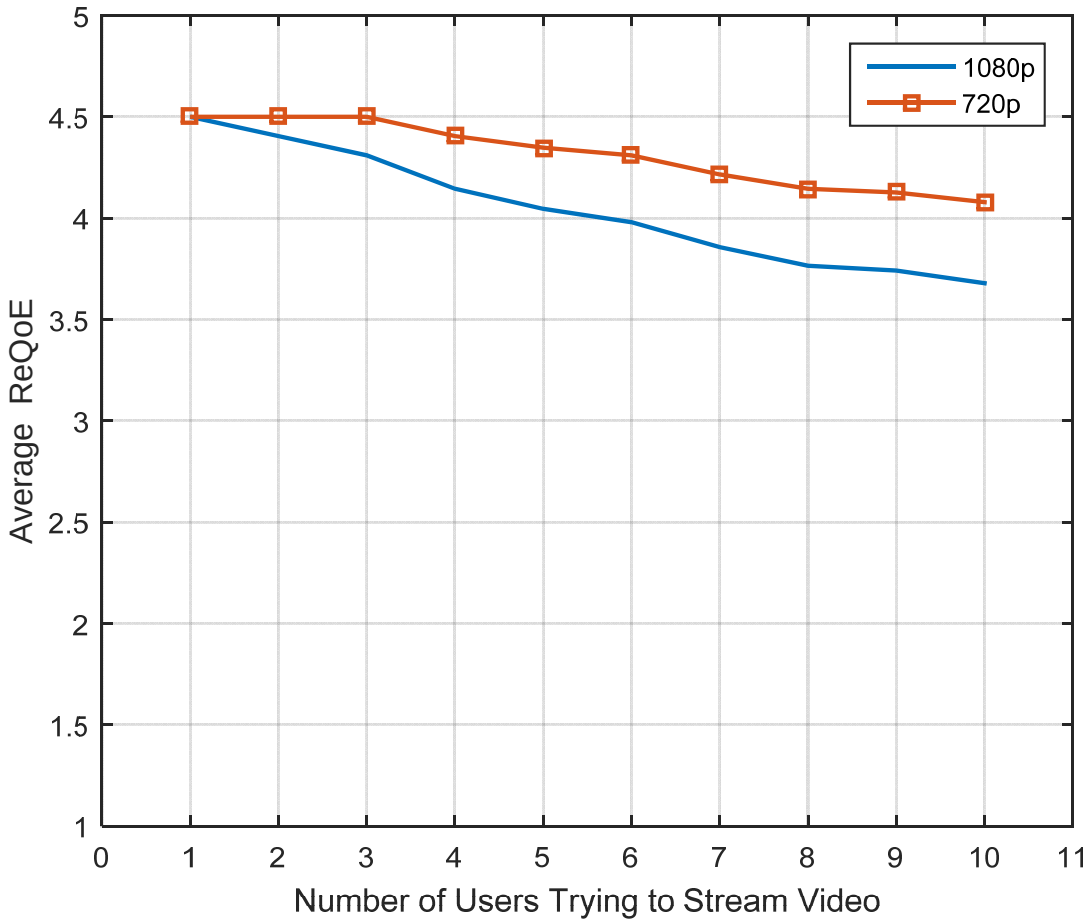

Figure 33: Average ReQoE of AP calculated using the Heuristic Algorithm with all users connecting at the same data rate

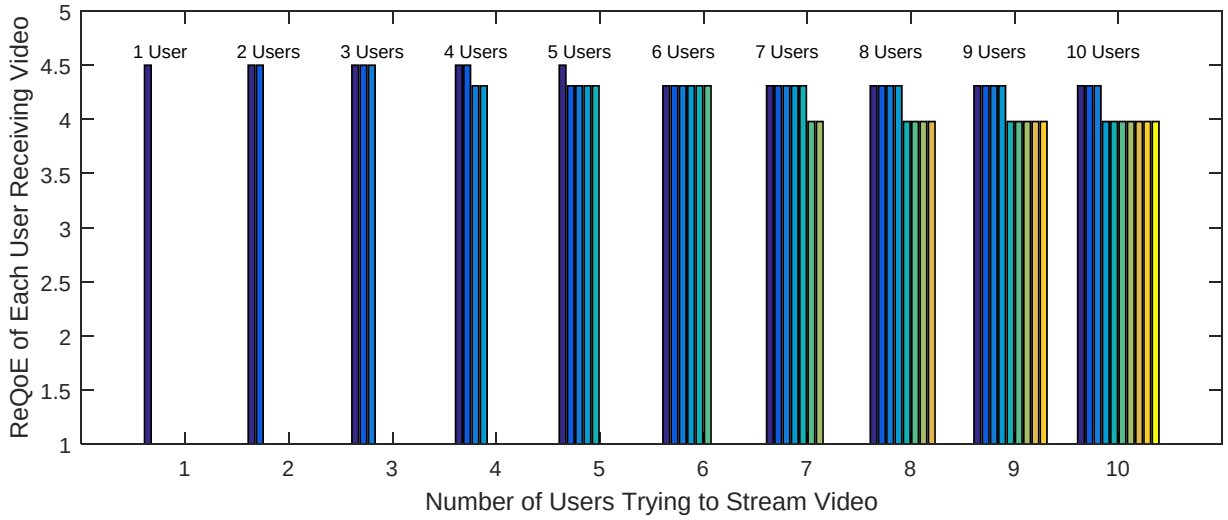

Figure 34: ReQoE of each user requesting 720p content and connecting at the same data rate

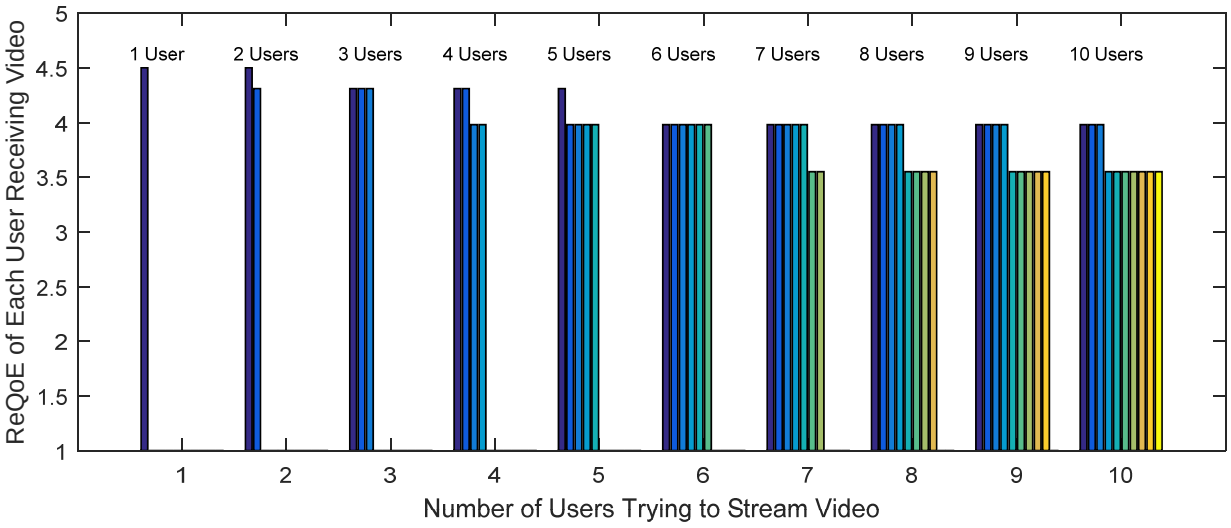

Figure 35: ReQoE of each user requesting 1080p content and connecting at the same data rate

The following set of three graphs, Figure 36, Figure 37, and Figure 38, are again the result of a single simulation. They show additional details for the output of the heuristic algorithm with users connecting at various data rates and requesting either 720p or 1080p streams. The plot lines on Figure 36 show the average ReQoE of the AP, the details of which are shown in Figure 37 and Figure 38. These bar graphs show each user's ReQoE and the effect on existing users as additional users join the access network. Because all users are initially requesting the same rate, it can be observed which users are transcoded or dropped first.

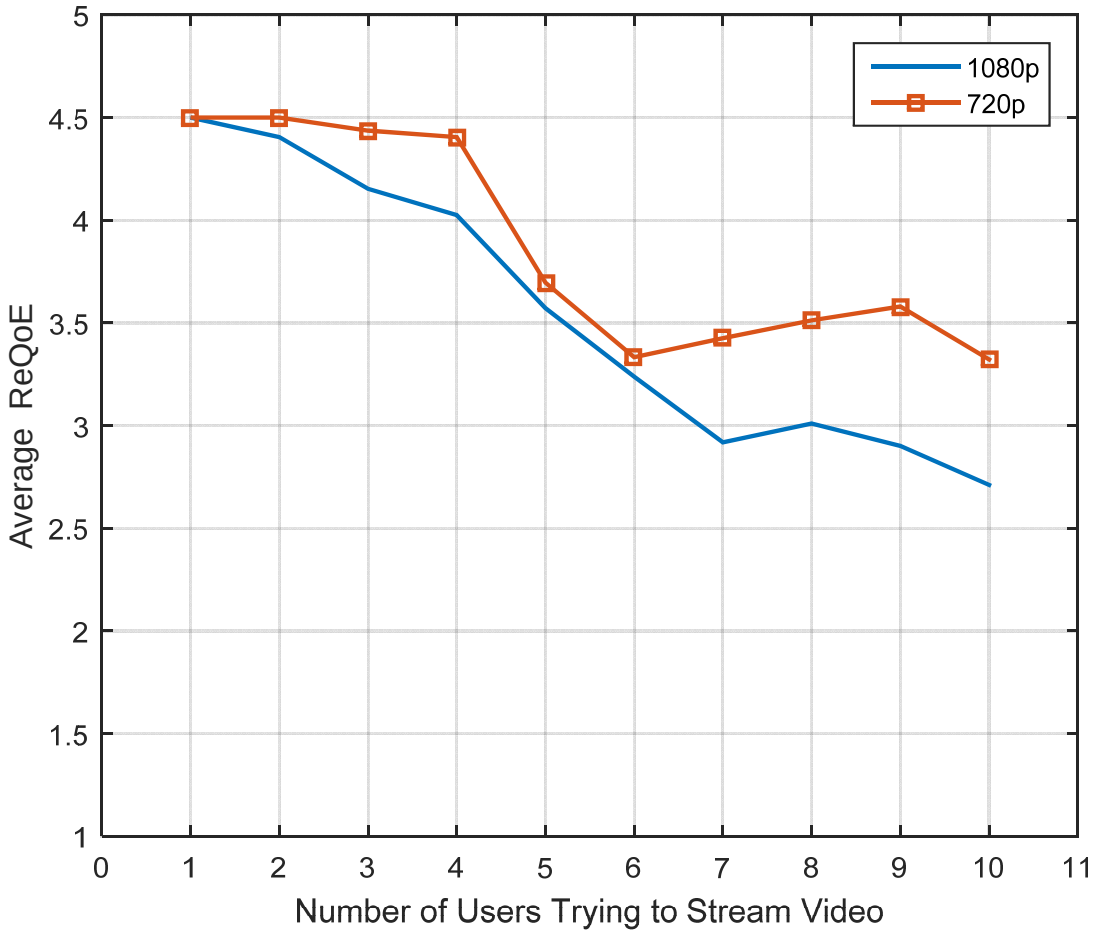

Figure 36: Average ReQoE calculated using the Heuristic Algorithm with users connecting at various data rates

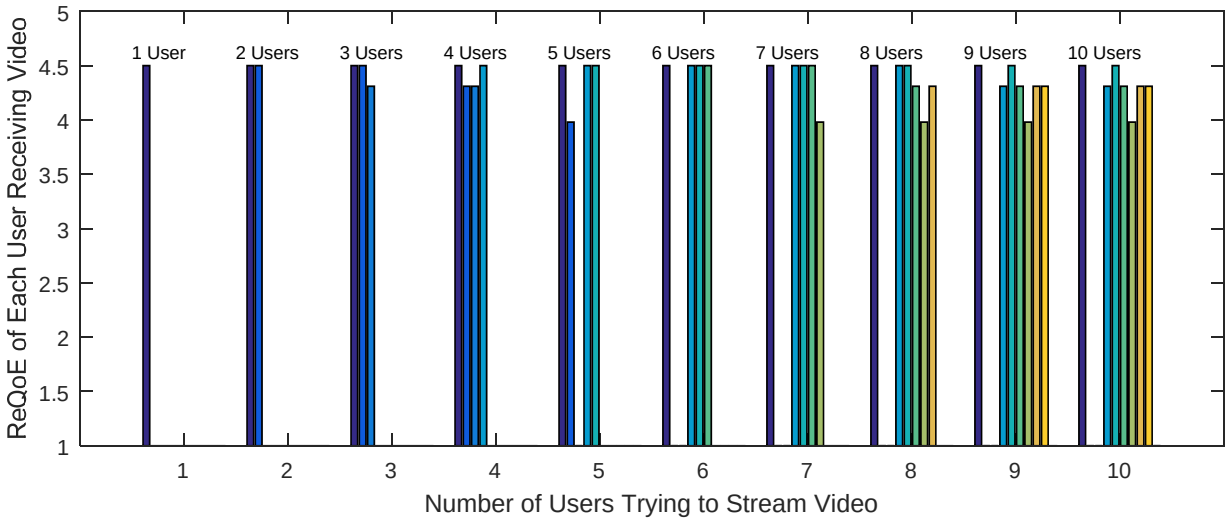

Figure 37: ReQoE of each user requesting 720p content and connecting at various data rates

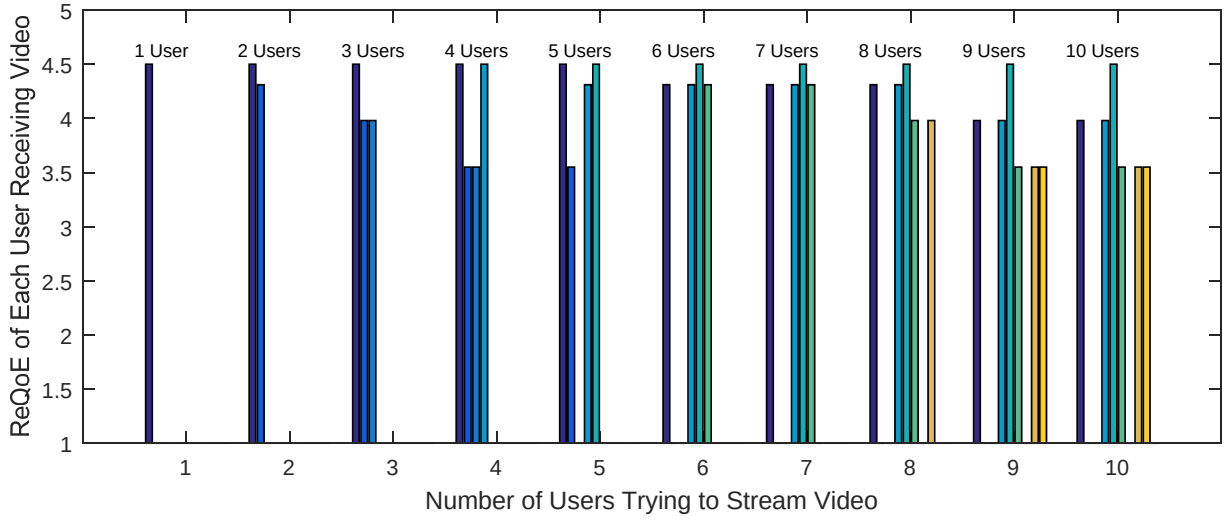

Figure 38: ReQoE of each user requesting 1080p content and connecting at various data rates

4.7.6 Impact of additional Transcoders

The following four figures, Figure 39 through Figure 42, are three axis graphs that provide further insights into the performance of the heuristic algorithm, and the impact CloudEdge and transcoding may have on an access network. Each graph shows the impact of limited transcoders, the number of transcoders used when they are used, and the resulting average ReQoE. First, Figure 39 shows all users requesting 1080p with only three transcoders available, this quickly becomes a very challenging, and resource constrained scenario. When the fifth user joins, the CloudEdge is no longer able to maintain a high level of satisfaction, and attempting to utilize more than one transcoder results in a lower average ReQoE than simply dropping users and maintaining the ReQoE of the initial users.

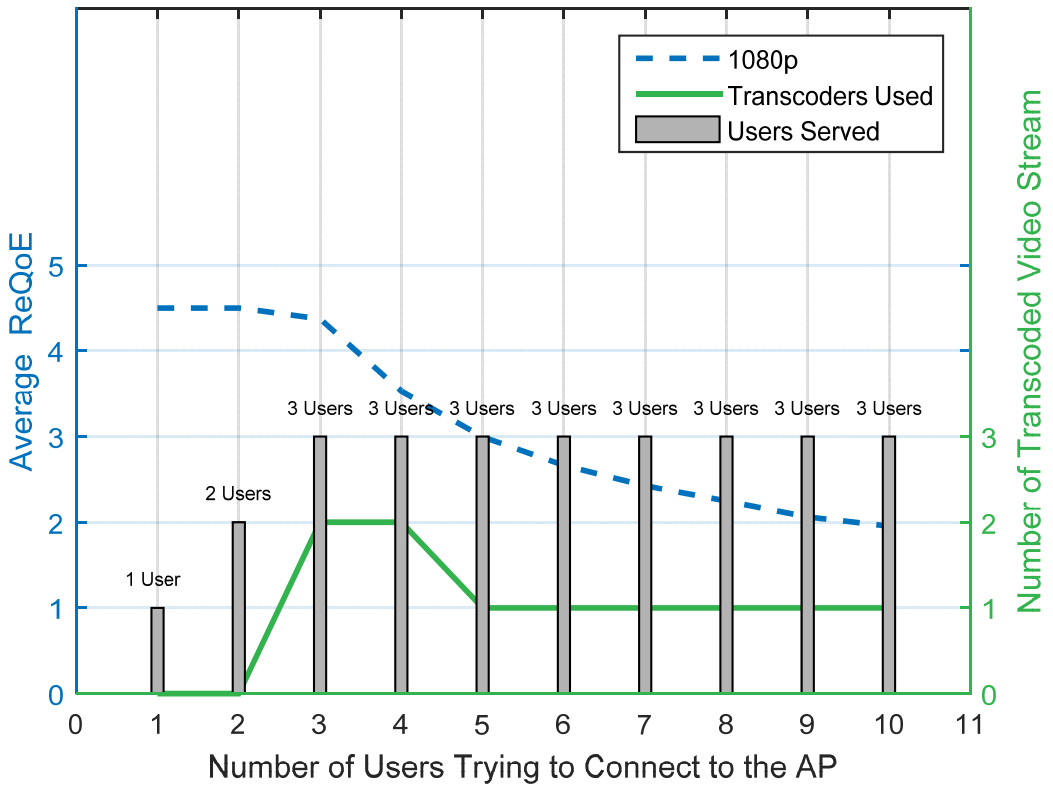

Figure 39: Average ReQoE, Number of Users, and Number of Transcoders all users connecting at the same data rate, limited to 3 transcoders

Figure 40 is the same model as Figure 39, however, with five transcoders available instead of only three. This makes a remarkable impact on the average ReQoE in all instances with four or more users, each user from the fourth onward requires transcoding. From the addition of the sixth user onward, the average ReQoE drops quickly, because a maximum of five users are able to be served and all users that are not

able to be transcoded are dropped, due to the lack of transcoders and the limited bandwidth.

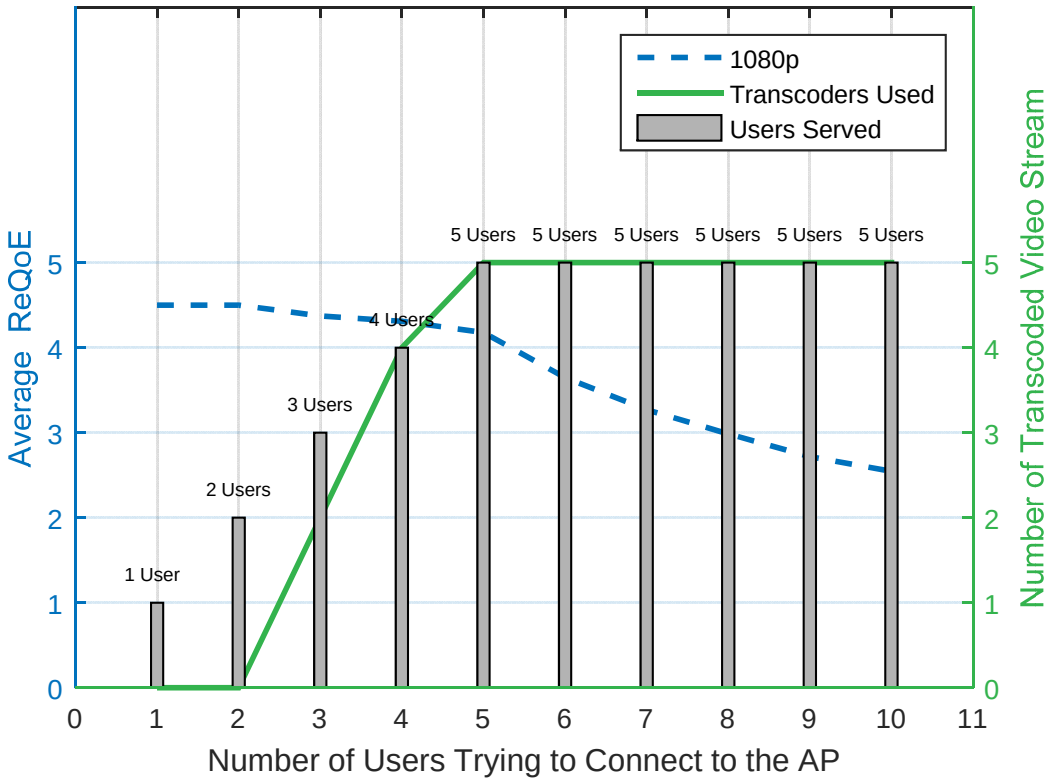

Figure 40: Average ReQoE, Number of Users, and Number of Transcoders all users connecting at the same data rate, limited to 5 transcoders

Figure 41 and Figure 42 utilize the same simulation model as Figure 39 and Figure 40 with the difference being that users are joining at various data rates as opposed to all joining at the same data rate. In each case, the impact of transcoding and the benefit of having transcoders available directly impacts the number of users that are able to be

served by the AP. The simulation in Figure 42, with five transcoders, supports seven users before the average ReQoE drops to the level at which most users are dissatisfied, whereas in the simulation with three transcoders, in Figure 41, only supports four users before the average ReQoE is too low.

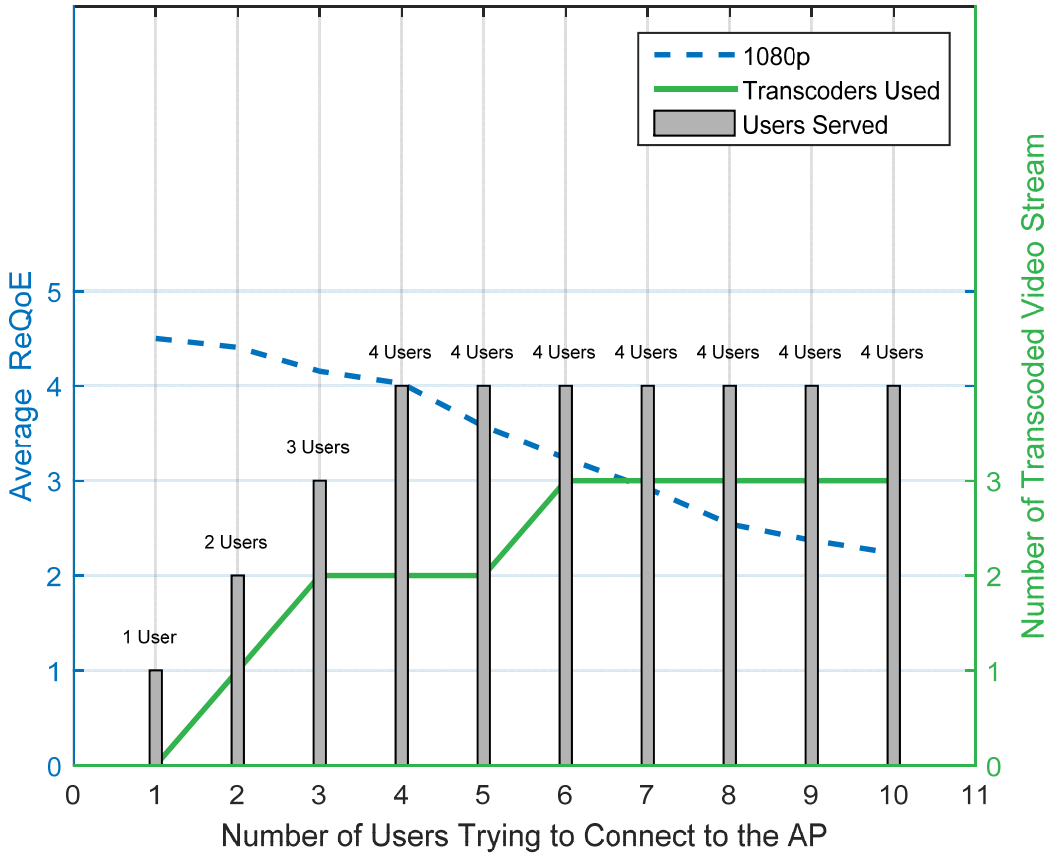

Figure 41: Average ReQoE, Number of Users, and Number of Transcoders users connecting at various data rates, limited to 3 transcoders

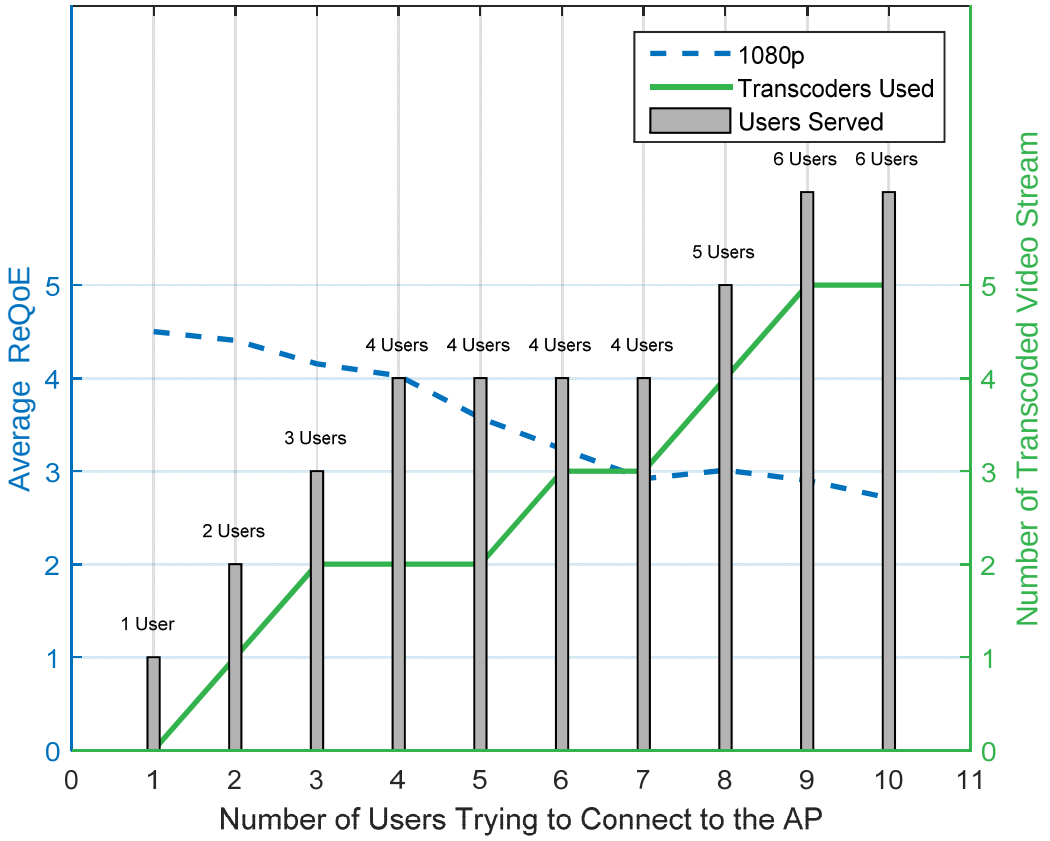

Figure 42: Average ReQoE, Number of Users, and Number of Transcoders of users connecting at various data rates, limited to 5 transcoders

4.7.7 Jain's Fairness Index

The review of the Jain's Fairness results begins with the model of users connecting at various data rates. The fairness index begins at 100; and for the single user case, all models have ideal fairness. Once additional users join the network, fairness calculations can begin and the plots in Figure 43 are the result of using the Jain's Fairness Equation [48] on the

received data rate of each user. In this scenario at first, the throughput maximization and heuristic algorithm, have the same fairness until the fourth stream begins. When the sixth stream begins, both of the approximation algorithms, and the Optimal ReQoE have the same fairness index. For the remainder of the cases, because the exhaustive search used to calculate the Optimal ReQoE routinely supports one or more users at the highest data rate while simultaneously dropping one or more users with poor SNRs, it is the most unfair of the optimization techniques.

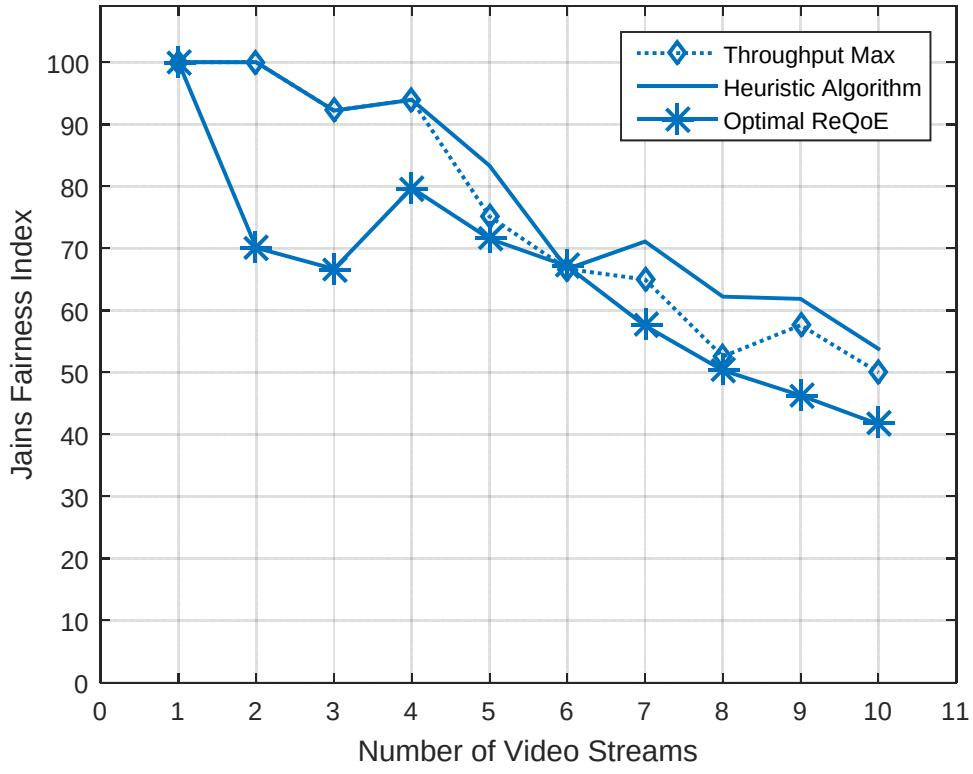

Figure 43: Jain's Fairness Index of a 720p scenario,
(with users connecting at various data rates)

Figure 44 shows the Jain's Fairness results for a model with all users connecting at the same AP data rate. The results are based on the final received data rate of each user, which tells an interesting story. Recall that the plot lines in Figure 21 and Figure 23 appeared to be the same, due to the resulting average ReQoE of the optimal ReQoE matching that of the heuristic algorithm, for cases in which all users connect at the same data rate. In the Jain's Fairness calculations, however, that is not the case. The optimal ReQoE scores lower on the Jain's Fairness Index than the heuristic algorithm; this is because the goal of the exhaustive search is different. The goal of the exhaustive search is to find the highest average ReQoE, and while the average ReQoE that it finds may be equal to that of the heuristic model, it may be found through a different combination of user resolutions, resulting in a lower Jain's Fairness Index. Because once the exhaustive search is complete if multiple ReQoE combinations exist that produce the same highest average ReQoE, the output of the algorithm is the first combination of ReQoEs discovered that results in the highest average ReQoE, even if a fairer combination exists.

The heuristic algorithm produces a high Jain's Fairness Index score, but the throughput maximization is often higher, because the heuristic algorithm was engineered to reallocate unutilized bandwidth in order to attempt to improve the AP average ReQoE of other users after determining if a user needs to be transcoded. This results in one or

more users having a higher resolution then the rest, and is therefore less fair. On the other hand, the throughput maximization, in the case where all users have the same date rate, spreads the AP utilization equally among all users. When the tenth user joins the throughput maximization fails completely, because the available AP throughput divided by ten users results in each user receiving a bandwidth allocation that is too low to support even the lowest resolution.

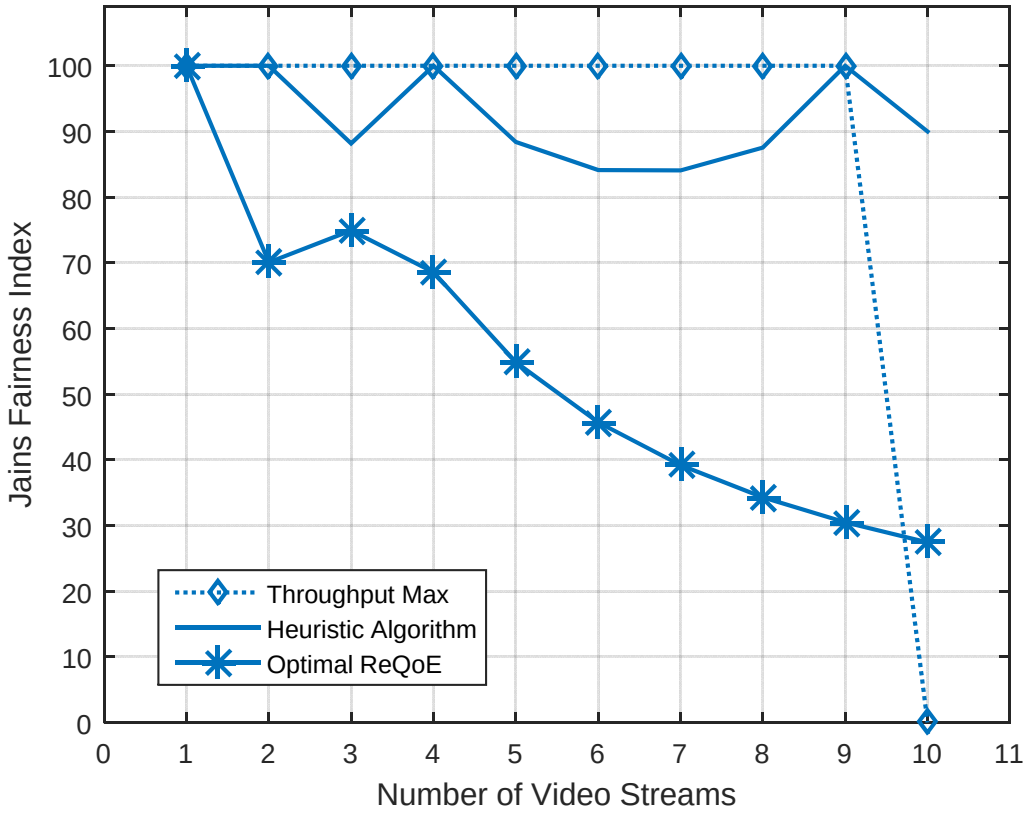

Figure 44: Jain's Fairness Index of a 720p scenario,
(with all users connecting at the same data rate)

Chapter 5 Conclusions

This dissertation evaluated the network QoS metrics and their interdependencies relating bandwidth to packet loss and latency through queueing theory to observe the impact on QoE. It proposed the concepts of Desired Quality of Experience (DeQoE) and Realized Quality of Experience (ReQoE). These quantities can be used to compare the performance of heterogeneous scenarios (e.g., different content, different devices, different providers, and different networks). The DeQoE also allows evaluation of the performance of a whole system based upon how well it met the users' expectations, instead of generically observing the QoS metrics. It provides a versatile scale for evaluating and better understanding the network resources required by a single user and, in some cases, it may prevent dedicating more network resources to a single user than necessary to ensure their DeQoE is met. By calculating the QoS requirements for each user/data stream, it is possible to optimize the AP resources to potentially support additional satisfied users.

Next, this dissertation proposed CloudEdge, a computation-capable and programmable wireless access network architecture. This architecture exploits in-

network processing and storage capabilities enabled through a local Cloud server and employs SDN to seamlessly integrate CloudEdge into the network. To address the multi-resource management challenges (available bandwidth and computation power) that arise in integrating in-network services for video content delivery, two approximation algorithms were proposed and evaluated for optimizing the average ReQoE of an AP. As a point of comparison, the optimal average ReQoE was calculated using an exhaustive search of the variables. Then the optimal ReQoE and the results of the two approximation algorithms were evaluated using Jain's Fairness Index. The simulation results clearly demonstrate the benefits of the proposed architecture and algorithms which significantly improved the average ReQoE as opposed to simply allowing users of oversubscribed wireless access networks to suffer from packet loss.

In the scenarios modeled with users connecting at different data rates, the resulting ReQoE from the heuristic model did perform as well or better than the throughput maximization, but it was often far lower than the Optimal ReQoE found through the exhaustive search. However, in the scenarios modeled with all users having the same date rate, the results of the heuristic model were equal to the Optimal ReQoE. This should prove to be a valuable insight for future work dedicated to refining the performance of the heuristic algorithm proposed here, in cases with users connected at different date rates.

Based on the results of the Jain's Fairness analysis, which showed the Optimal ReQoE to be the most unfair, and the fact that the exhaustive search is exceptionally computationally expensive and too slow to be implemented in a system that may need to make real-time adjustments, there is value in formulating and refining the heuristic algorithm proposed in this dissertation.

In the end, the greatest customer satisfaction, bandwidth savings, and cost savings can be achieved by determining the minimum QoS required to provide the maximum possible QoE. Due to unforeseeable network issues in real world scenarios, service providers will always be required to exceed the minimum QoS requirements by a small factor to ensure users receive a high QoE. Also, as shown here, having a better understanding of the impact the QoS has on a user's QoE may allow for more satisfied users on a single network.

Future work on CloudEdge, should expand on the multi-resource multiple constraint network management simulation to include users connecting to a single access point and requesting different resolutions of content. Another concept that should be further explored in terms of the DeQoE protocol is the concept of Hysteresis [28]; in short, that a user's current perceived QoE will affect how they qualitatively rate their experience in the near future. Finally, the heuristic algorithm here should be tested in a prototype

CloudEdge network as a proof of concept to validate the algorithm, and gain deeper insights in to the benefits of integrating in-network processing, caching, and delivery in a SDN-based wireless access network.

Bibliography

[1] ITU-T E.800: “Definitions of terms related to quality of service” September 2008.

[2] Definition of Quality of Experience (QoE), ITU Technical Document 109rev2 (PLEN/12) Geneva, 16-25 January 2007.

[3] Cisco, “Cisco Visual Network Index: Global Mobile Traffic Forecast Update 2014–2019,” Feb 2015.

[4] N. McKeown et al., ”OpenFlow: Enabling Innovation in Campus Networks,” ACM Comp. Commun. Rev., Apr. 2008.

[5] R. Curtis et al., “DevoFlow: Scaling Flow Management for High-Performance Networks,” Proc. ACM SIGCOMM’11, 2011.

[6] M. Banikazemi et al., “Meridian: An SDN Platform for Cloud Network Services,” IEEE Commun. Mag. February 2013.

[7] S. Jain et al., “B4: Experience with a Globally-Deployed Software Defined WAN,” SIGCOMM’13, Hong Kong, Aug. 2013.

[8] Aditya Gudipati, Daniel Perry, Erran Li and Sachin Katti, “SoftRAN: Software Defined Radio Access Networks,” ACM HotSDN, Aug. 2013.

[9] P. Dely, A. Kassler, and N. Bayer, “Openflow for wireless mesh networks,” 20th International Conference on Computer Communications and Networks (ICCCN), Aug. 2011.

[10] G. Bhanage, I. Seskar, R. Mahindra, and D. Raychaudhuri, “Virtual basestation: Architecture for an open shared wimax framework” ACM SIGCOMM VISA Workshop, 2010.

[11] Georgios Gardikis et al., “Cross-Layer Monitoring in IPTV Networks,” IEEE Communications Magazine July 2012.

[12] UmaMaheswari C. Devi, Ritesh Kalle, and Shivkumar Kalyanaraman, “Multi-Tiered, Burstiness-Aware Bandwidth Estimation and Scheduling for VBR Video Flows,” IEEE Transactions on Network and Service Management, Vol. 10, No. 1, March 2013.

[13] ITU-T G.107 : “The E-model: a computational model for use in transmission planning,” December 2011.

[14] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, “Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet,” SIGMOBILE Mob. Comput. Commun. Rev., 16(3):2–13, 2012.

[15] Cisco, “Cisco Fog Computing Solutions: Unleash the Power of the Internet of Things,” May 2015.

[16] J. Kurose and K. Ross "Computer Networking A Top-Down Approach Featuring the Internet," Addison Wesley, 2001.

[17] T. Hossfeld, et. al., "Initial Delay Vs. Interruptions: Between The Devil and the Deep Blue Sea" Quality of Multimedia Experience (QoMEX), 2012 Fourth International Workshop on, Yarra Valley, VIC, 2012, pp. 1-6.

[18] M. Mathis, J. Semke, J. Mahdavi, and T. Ott "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," ACM SIGCOMM Computer Communication Review, Volume 27 Issue 3, July 1997.

[19] ITU-T P.800: "Methods for subjective determination of transmission quality," August 1996.

[20] Baltoglou, G., Karapistoli, E., and Chatzimisios, P., "IPTV QoS and QoE Measurements in Wired and Wireless Networks," Global Communications Conference (GLOBECOM), pages 1757 – 1762, IEEE, 3-7 Dec. 2012.

[21] "Software-Defined Networking: The New Norm for Networks," Open Network Foundation, April 13, 2012.

[22] "Network Functions Virtualisation (NFV); Architectural Framework," ETSI GS NFV 002 V1.1.1, October 2013.

[23] "The NIST Definition of Cloud Computing" NIST Special Publication 800-145, September 2011.

[24] Markus Fiedler, Tobias Hossfeld, Phuoc Tran-Gia, "A Generic Quantitative Relationship Between Quality of Experience and Quality of Service Network," IEEE Network, Vol. 24, No. 2., pp. 36-41, Mar. 2010.

[25] Sofiene Jelassi et al., "Quality of Experience of VoIP Service: A Survey of Assessment Approaches and Open Issues," IEEE Communications Surveys & Tutorials, Vol. 14, No. 2, Second Quarter 2012.

[26] ETSI 3GPP 3GPP TS 26.247; Transparent end-to-end packet-switched streaming service (PSS); Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH).

[27] Mohammed Alreshoodi and John Woods, Survey on QoE\QoS Correlation Models For multimedia Services, International Journal of Distributed and Parallel Systems (IJDPS) Vol.4, No.3, May 2013.

[28] Markus Fiedler and Tobias Hossfeld, "Quality of Experience-Related Differential Equations and Provisioning-Delivery Hysteresis," IEICE ITC-SS21, March 2010.

[29] Toon De Pessemier et al., "Quantifying the Influence of Rebuffering Interruptions on the User's Quality of Experience during Mobile Video Watching," Broadcasting, IEEE Transactions on Broadcasting, Vol. 59, No. 1, March 2013.

[30] Leonard Kleinrock, “Queueing Systems Volume I: Theory,” Wiley-Interscience, January 2, 1975.

[31] Mischa Schwartz, “Telecommunication Networks: Protocols, Modeling and Analysis,” Prentice Hall, January 11, 1987.

[32] Leonard Kleinrock, “Queueing Systems Volume II: Computer Systems,” Wiley-Interscience, April 22, 1976.

[33] All Clipart used in this figure was downloaded from https://openclipart.org it is royalty-free and has been released into the Public Domain.

[34] V. Jacobson et al., “Networking Named Content,” CoNEXT ’09, New York, NY, 2009.

[35] Pursuit Website, http://www.fp7-pursuit.eu/PursuitWeb/

[36] Scalable and Adaptive Internet Solutions (SAIL), http://www.sail-project.eu/

[37] H. Liu, X. De Foy, D. Zhang, “A Multi-Level DHT Routing Framework with Aggregation,” ACM SIGCOMM ICN’12, August 2012.

[38] J. Sherry et al., “Making middleboxes someone else’s problem: Network processing as a cloud service. In Proc. of ACM SIGCOMM, 2012.

[39] M. Satyanarayanan et al., “The case for VM-based cloudlets in mobile computing,” Pervasive Comput, 8(4):14–23, 2009.

[40] V. Sekar et al., “Design and implementation of a consolidated middlebox architecture,” NSDI, 2012.

[41] Y. Chen, B. Liu, Y. Chen, A. Li, X. Yang, and J. Bi, “Packetcloud:an open platform for elastic in-network services,” Eighth ACM international workshop on mobility in the evolving internet architecture, 2013.

[42] F. Bronzino et al., “In-Network Compute Extensions for Rate-Adaptive Content Delivery in Mobile Networks,” IEEE ICNP’14, Raleigh, NC, 2014.

[43] ISO/IEC 23009-1:2014, “Information technology -- Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats,” May 2014.

[44] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, “Confused, timid, and unstable: picking a video streaming rate is hard,” ACM conference on Internet measurement, 2012.

[45] J. Chen, R. Mahindra, M. A. Khojastepour, S. Rangarajan, and M. Chiang, “A scheduling framework for adaptive video delivery over cellular networks,” 19th annual international conference on Mobile computing & networking, 2013.

[46] Kumar, “Mobile Broadcasting with WiMAX Principles: Technology and Applications,” Focal Press 2008.

[47] Youtube.com, “Recommended upload encoding settings: Recommended bitrates, codecs, and resolutions, and more.” https://support.google.com/youtube/answer/1722171

[48] R. Jain, W. Hawe, D. Chiu, “A Quantitative measure of fairness and discrimination for resource allocation in Shared Computer Systems,” DEC-TR-301, Sept 26, 1984.

Appendix A: MATLAB Code

A.1 MOS and R-Factor Functions

A.1.1 MOS to R-Factor

The equation used here was defined in [13]

```
RVAL=@(M) (20/3)*((8-sqrt(226)*cos([(1/3)*(atan2(15*sqrt(-903522+1113960*M-
202500*(M^2)), 18566-6750*M))]+(pi/3))));

figure;
fplot(RVAL, [1, 4.5], 'blue');
xlim([1 5]);
ylim([0 100]);
hold on;
ylabel('R-Factor');
xlabel('MOS');
title('MOS to R-Factor Function');
hold off;
```

A.1.2 R-Factor to MOS

The equation used here was defined in [13]

```
MOS=@(R) 1-(7/1000)*R+(7/6250)*R^2-(7/1000000)*R^3;

figure;
fplot(MOS, [0 , 100], 'blue');
xlim([1 100]);
ylim([0 5]);
hold on;
xlabel('R-Factor');
ylabel('MOS');
title('R-Factor to MOS Function');
hold off;
```

A.2 QoS to QoE Mapping Functions

A.2.1 Packet Loss and Packet Reordering Merged Graph

Both equations used here were defined in [24].

```
PLQOE=@(QOE) 3.010*exp(-4.473*(QOE/100))+(1.065+0.425);
PRRQOE=@(QOE) 2.482*exp(-10.453*(QOE/100))+1.141;

figure;
fplot(PLQOE, [0,50]); %Packet Loss Function
xlabel('% Packet Loss and Reordered Ratio');
ylabel('QoE MOS');
hold on;
fplot(PRRQOE, [0,50],'Color',':red'); %Reordered Ratio Function
xlim([0 50]);
ylim([0 5.5]);
set(findall(gca, 'Type', 'Line'),'LineWidth',1.5);
legend('Packet Loss','Reordered Ratio');
hold off;
```

A.2.2 Buffered Packet Loss and Packet Reordering Merged Graph

Both equations used here are based on the equations defined in [24].

```
buff = 20; %in Megabytes
rate = 200; %fill rate in MBps
rPL = buff/rate;
rJ = buff/rate;

% Calculate the time period of buffer (how long it will last)
rGBuff = (.5*rJ+.5*rPL)*100;

PLQOE=@(QOE) 3.010*exp(-4.473*(QOE-(rPL*100))/100)+1.065; %(-rPL) shifts
start of function to the right rPL units
PRRQOE=@(QOE) 2.482*exp(-10.453*(QOE-(rJ*100))/100)+1.141; %(-rJ) shifts
start of function to the right rJ units

GBMaxPL = (PLQOE(rGBuff));
GBMaxPRR = (PRRQOE(rGBuff));

figure;
hold on;
box on;
plot([rGBuff 0], [GBMaxPL GBMaxPL],':blue'); %line from 0 to 'rGBuff' on x
axis at GBMax on y axis
plot([rGBuff 0], [GBMaxPRR GBMaxPRR],':red'); %line from 0 to 'rGBuff' on x
axis at GBMax on y axis
fplot(PLQOE, [rGBuff,50],'Color','blue'); %Packet Loss Function plotted from
rGBuff to 1
xlabel('% Packet Loss and Reordered Ratio');
ylabel('QoE MOS');
fplot(PRRQOE, [rGBuff,50],'Color','--red'); %Reordered Ratio plotted from
rGBuff to 1
```

```
xlim([0 50]);
ylim([0 5.5]);
set(findall(gca, 'Type', 'Line'),'LineWidth',1.5);
legend('Buffer Period','Buffer Period','Packet Loss','Reordered Ratio');
hold off;
```

A.2.3 Original Packet Loss and Modified Packet Loss Graph

The equation labeled PLQOE2 below was defined in [24], the equation labeled PLQOE is a modification of that function.

```
PLQOE=@(QOE) 3.010*exp(-4.473*(QOE/100))+(1.065+0.425);
PLQOE2=@(QOE) 3.010*exp(-4.473*(QOE/100))+(1.065);

figure;

fplot(PLQOE, [0,50]); %Packet Loss Function
xlabel('% Packet Loss');
ylabel('QoE');
hold on;
fplot(PLQOE2, [0,50],':');
xlim([0 50]);
ylim([1 5]);
set(findall(gca, 'Type', 'Line'),'LineWidth',1.5);

rline = refline([0 3.04]);
rline.Color = 'r';
rline.LineStyle = '-.';
rline.LineWidth = 2;

legend('Modified Packet Loss Equation','Original Packet Loss
Equation','Nearly All Users Dissatisfied');
hold off;
```

A.2.4 Latency

Equation based off data set generated from ITU E-Model [13]

```
LQOE=@(L) (-1.1101e-10*(L)^4+1.7505e-07*(L)^3-9.8251e-
05*(L)^2+0.019332*(L)+3.1809);

%plot(Output(:,2),Output(:,7),'--d','Color',[.2 .7 .3],'LineWidth',1.5);
figure;
plot([150 0], [4.41 4.41],'blue'); %line from 0 to 150 on x axis at 4.41 on y
axis
```

```
hold on;

fplot(LQOE, [150, 500], 'blue');
hold on;
xlabel('Latency in ms');
ylabel('QoE/MOS Score');
%xlabel('Wait Time in Seconds'); %original Equation Labels
%ylabel('Probability QoE is Acceptable'); %original Equation Labels
xlim([0 500]);
ylim([3 5]);
title('Impact of Latency/Transmission Delay on QoE');
legend('Latency');
hold off;
```

A.2.5 Startup Delay

Equation based off [29] but modified to inverse and scale from 0 to 100%

```
DQOE=@(QOE) 100*(1-((exp(-2.4437+.0626*(QOE))) / (1+exp(-
2.4437+.0626*(QOE)))));

figure;
fplot(DQOE, [0,100]);
xlabel('Delay in seconds');
ylabel('Percent Probability that Startup Delay is Acceptable');
%title('Impact of Delay on QoE');
set(findall(gca, 'Type', 'Line'),'LineWidth',1.5);
legend('Startup Delay');
```

A.3 Optimization Models

A.3.1 Transcoding vs No Transcoding Adding Video Streams

The equation `=3.010*exp(-4.473*(User(k,5)/100))+(1.065+0.425)` used to calculate the impact of packet loss on QoE in the following MATLAB m file is a modification of the equation presented in [24].

```
%Required Input:
n = 10; %Number of Users
To = 1.1; %TCP/IP Overhead Percentage Multiplier

% User will be the primary Data Matrix organized such that:
    % User(1,1) is the users Number,
    % User(1,2) is the users APRate (access point rate, static)
```

```
    % User(1,3) is the users RDRate (Desired Data rate, static)
    % User(1,4) is the users DRate (Requested Data Rate, may change)
    % User(1,5) is the users DRate (Actual Data Rate, may change)
    % User(1,6) is the calculated alpha for User 1
    % User(1,7) is 1 if Tc is needed and 0 if it is not
    % User(1,8) is the actual Video Tx Rate that is supported
    % User(1,9) is the MOS of the Video Stream
UData = 9;
User = zeros(n+1, UData);

%Fixed Values
SRate360p = 1.128; % Service Rate for 360p
SRate480p = 2.628; % Service Rate for 480p
SRate720p = 5.384; % Service Rate for 720p
SRate1080p = 8.384; % Service Rate for 1080p
DDRate360p = round(SRate360p*To,4); %Rounded Desired Data Rate for 360p
DDRate480p = round(SRate480p*To,4); %Rounded Desired Data Rate for 480p
DDRate720p = round(SRate720p*To,4); %Rounded Desired Data Rate for 720p
DDRate1080p = round(SRate1080p*To,4); %Rounded Desired Data Rate for 1080p
DRates = 4; %There are Four Potential Data Rates
DDRate = [DDRate360p; DDRate480p; DDRate720p; DDRate1080p];
%VRate = {'360p'; '480p'; '720p'; '1080p'};
VRate = {'720p'; '1080p'};
VRates = numel(DDRate);
MOS = [3.55; 3.98; 4.31; 4.5];

%Required Input: Maximum Rate for each user based on SNR Table
%(6,9,12,18,24,36,48,54) in Mbps
APRate = [6;9;12;18;24;36;48;54];

%APRates for 10 Users
APRates(1:n,1) = 12; %Set all Users to same AP Rate ( = same SNR)

figure;
hold on;
vr = VRates; %Number of Video Rates
while vr > 2
    m = n; %Total Number of Users
    while m > 0
        User = zeros(m+1, UData); %Reset Users Array for each run
        n1 = m;%Current Number of Users for this loop
        while n1 > 0
            User(n1,1)=n1;% Set user Number
            User(n1,2)=APRates(n1);% Set all users to have an APRate
            User(n1,3)=DDRate(vr);% Set all users to have an DDRate
            User(n1,4)=User(n1,3);% Set users initial Requested to DDRate
            User(n1,5)=User(n1,3);% Set users initial Date Rate to DDRate
            n1 = n1-1;
        end

        % This Loop makes the LP iterative in order to find the highest
        % Video Data Rate that can be transmitted while ensuring any excess
```

```
        % bandwidth can be used by other users
        k = m*4;
        while k > 1
            fn = m;
            f = zeros(1,fn+1);
            f(1,fn+1) = -1; %0 or -1 results appear the same
            A = zeros(fn+1,fn+1);
            A(1,fn+1) = 1;
            b = ones(1,fn+1);
            lb = zeros((fn+1),1);

            while fn > 0
                f(1,fn) = -User(fn,2);
                A(fn+1,fn) = User(fn,2);
                A(1,fn) = 1;
                b(1,fn+1) = User(fn,4);
                fn = fn - 1;
            end

            [X,fmax] = linprog(f,A,b,[],[],lb);
            X; %Test Output disabled
            abs(fmax); %Test Output disabled

            wn = m;
            while wn > 0 %Store the Calculated Value for Actual Data Rate in
the User Array
                User(wn,5) = round(User(wn,2)*X(wn,1),4);
                wn = wn - 1;
            end

%Start Function for changing Users Requested Data Rate if needed
            Reqmax = User(:,4); %build matrix of column 4 values
            Rqmax = max(Reqmax); %Find Max Value in column 4
            Reqmin = User(:,4); %build matrix of column 4 values
            Rqmin = min(Reqmin(Reqmin~=0)); %Find non-Zero Minimum of Column
4
            APRatemin = User(:,2); %build matrix of column 2 values
            APmin = min(APRatemin(APRatemin~=0)); %Find non-Zero Minimum of
Column 2

            kk = m+1;
            while kk > 1
                kk = kk - 1;
                if User(kk,4) == Rqmax && User(kk,4) == Rqmin
                    if User(kk,5) < User(kk,4) && User(kk,5) > 0
                        if User(kk,5) < User(kk,4) && User(kk,5) < DDRate(1)
                            User(kk,4) = 0;
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(2)
                            User(kk,4) = DDRate(1);
                            break
```

```
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(3)
                            User(kk,4) = DDRate(2);
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(4)
                            User(kk,4) = DDRate(3);
                            break
                        end
                    end
                elseif User(kk,4) == Rqmax
                    if User(kk,5) < User(kk,4) && User(kk,5) > 0
                        if User(kk,5) < User(kk,4) && User(kk,5) < DDRate(1)
                            User(kk,4) = 0;
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(2)
                            User(kk,4) = DDRate(1);
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(3)
                            User(kk,4) = DDRate(2);
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(4)
                            User(kk,4) = DDRate(3);
                            break
                        end
                    end
                end
            end %End Function for changing Users Requested Data Rate if
needed

            n2 = m;
            while n2 > 0 % Store Final Alpha in User Array
                User(n2,6) = X(n2);
                n2 = n2-1;
            end

            k = k - 1;
        end

        n3 = m;
        while n3 > 0 % Set Transcode Value to 1 if needed
            if round(User(n3,3),4) <= round(User(n3,5),4)
                User(n3,7) = 0;
            elseif round(User(n3,5),4) < DDRate(1)
                User(n3,7) = 0;
            else
                User(n3,7) = 1;
            end
            n3 = n3-1;
```

```
        end

        n4 = m;
        while n4 > 0 %Find Tx Video Rate to Calculate Average ReQoE for the
Access Point
            if User(n4,5) > User(n4,4)
                if User(n4,5) < DDRate(1)
                    User(n4,8) = 0;
                elseif User(n4,5) < DDRate(2)
                    User(n4,8) = DDRate(1);
                elseif User(n4,5) < DDRate(3)
                    User(n4,8) = DDRate(2);
                elseif User(n4,5) < DDRate(4)
                    User(n4,8) = DDRate(3);
                elseif  User(n4,5) > DDRate(4) && User(n4,3) == DDRate(4)
                    User(n4,8) = DDRate(4);
                elseif  User(n4,5) > DDRate(3) && User(n4,3) == DDRate(3)
                    User(n4,8) = DDRate(3);
                elseif  User(n4,5) > DDRate(2) && User(n4,3) == DDRate(2)
                    User(n4,8) = DDRate(2);
                elseif  User(n4,5) > DDRate(1) && User(n4,3) == DDRate(1)
                    User(n4,8) = DDRate(1);
                end
            %Set Values to next closest, if Tx rate is to low to Tx
            elseif  User(n4,5) < User(n4,4) && User(n4,5) > DDRate(3)
                User(n4,8) = DDRate(3);
            elseif  User(n4,5) < User(n4,4) && User(n4,5) > DDRate(2)
                User(n4,8) = DDRate(2);
            elseif User(n4,5) < User(n4,4) && User(n4,5) > DDRate(1)
                User(n4,8) = DDRate(1);
            elseif User(n4,5) < DDRate(1)
                User(n4,8) = 0;
            else
                User(n4,8) = User(n4,4);
            end
            n4 = n4 - 1;
        end

        %Determine MOS of each Stream
        apMOS=0;
        n5 = m;
        while n5 > 0
            if User(n5,8) > 0
                %Sets MOS to highest value, if expected data rate = received
data rate
                if User(n5,8) == User(n5,3)
                    User(n5,9) = MOS(4);
                %Loswers MOS by 1 step
                elseif User(n5,3) == DDRate(2) && User(n5,8) == DDRate(1)
                    User(n5,9) = MOS(3);
                elseif User(n5,3) == DDRate(3) && User(n5,8) == DDRate(2)
                    User(n5,9) = MOS(3);
```

```
                elseif User(n5,3) == DDRate(4) && User(n5,8) == DDRate(3)
                    User(n5,9) = MOS(3);
                %Lowers MOS by 2 steps
                elseif User(n5,3) == DDRate(3) && User(n5,8) == DDRate(1)
                    User(n5,9) = MOS(2);
                elseif User(n5,3) == DDRate(4) && User(n5,8) == DDRate(2)
                    User(n5,9) = MOS(2);
                %Lowers MOS by 3 steps (e.g., 1080p -> 360p)
                elseif User(n5,3) == DDRate(4) && User(n5,8) == DDRate(1)
                    User(n5,9) = MOS(1);
                end
            else %MOS is a 1-5 scale, so setting to 1 is the lowest value
                User(n5,9) = 1;
            end
            apMOS = apMOS + User(n5,9);
            ILPUsers(m,n5,vr) = User(n5,9); %Store Each Users MOS value
            n5 = n5 - 1;
        end

        % Output will be the Results Data Matrix organized such that:
            % Output(1,1) is the Number of Users,
            % Output(1,2) is the Average QoE of the Access Point
            % Output(1,3) is the total number of Transcoded Users
            % Output(1,4) is the Number of Users being Served (maybe <
connected)

        Output(m,1,vr) = m;
        Output(m,2,vr) = apMOS/m;
        Output(m,3,vr) = sum(User(:,7));
        Output(m,4,vr) = sum(User(:,5)>0);

        m = m - 1;
    end

    if vr == 4
        ILP = plot(Output(:,1,vr),Output(:,2,vr),'LineWidth',1.5);
    else
        ILP = plot(Output(:,1,vr),Output(:,2,vr),'-d','LineWidth',1.5);
    end

vr = vr - 1;
end

%
%
%     Break, to run with No SDN, No Optimization, and No-Transcoding
%     This is done by treating all Data request above the limit of the
%     access point as Packet Loss
%
%
```

```
set(gca,'ColorOrderIndex',1) %Reset Plot Color Order
vr = VRates; %Number of Video Rates

while vr > 2
    m = n; %Total Number of Users
    while m > 0
        User = zeros(m, UData); %Reset Users Array for each run
        n1 = m; %Current Number of Users for this loop
        while n1 > 0
            User(n1,1) = n1;% Set user Number
            User(n1,2) = APRates(n1);% Set all users to have an APRate
            User(n1,3) = DDRate(vr);% Set all users to have an DDRate
            User(n1,4) = (User(n1,3))/(User(n1,2)); %alpha = Desired data /
AP Rate
            User(n1,5) = 0;% Set initial Packet Loss Rate
            User(n1,6) = 0;% Set initial MOS per User
            n1 = n1 - 1;
        end

        sumA = sum(User(:,4));
        MaxRate = (sum(User(:,2)/m)/m);  %May not always work... check

        if sumA > 1
            k = m; %Current Number of Users
            while k > 0 %Calculate % Packet Loss per User
                User(k,5) = abs((1 - (MaxRate / User(k,3)))*100);  % ==
Percent Packet Loss
                User(k,6) = 3.010*exp(-4.473*(User(k,5)/100))+(1.065+0.425);
                k = k - 1;
            end
        elseif sumA <= 1
            User(:,6) = MOS(4);
        end

        n2 = m;
        apMOS = 0;
        while n2 > 0 % Average MOS of the AP
            apMOS = apMOS + (User(n2,6));
            NLPUsers(m,n2,vr) = User(n2,9); %Store Each Users MOS value
            n2 = n2-1;
        end

        % Output will be the Results Data Matrix organized such that:
            % Output(1,1) is the Number of Users,
            % Output(1,2) is the Average QoE of the Access Point
            % Output(1,3) is the total number of Transcoded Users
            % Output(1,4) is the Number of Users being Served (maybe <
connected)
        Output(m,1,vr) = m;
        Output(m,2,vr) = (apMOS / m); %For Mean MOS
        m = m - 1;
    end
```

```
    if vr == 4
        NLP = plot(Output(:,1,vr),Output(:,2,vr),'--','LineWidth',1.5);
    else
        NLP = plot(Output(:,1,vr),Output(:,2,vr),'--d','LineWidth',1.5);
    end

    vr = vr - 1;
end

rline = refline([0 3.04]);
rline.Color = 'r';
rline.LineStyle = '-.';
rline.LineWidth = 2;


flprate = flipud(VRate);
legend('1080p Transcoding', '720p Transcoding', '1080p No Transcoding', '720p
No Transcoding','Nearly All Users Dissatisfied');

xlim([0 n+1]);
ylim([1 5]);
xlabel('Number of Video Streams');
ylabel('Average ReQoE');
set(gca, 'box', 'on');
grid on;
hold off;
```

A.3.2 Impact of Adding Transcoders with Heuristic algorithm

```
%Required Input:
n = 12; %Number of Users
To = 1.1; %TCP/IP Overhead Percentage Multiplier

% User will be the primary Data Matrix organized such that:
    % User(1,1) is the users Number,
    % User(1,2) is the users APRate (access point rate, static)
    % User(1,3) is the users RDRate (Desired Data rate, static)
    % User(1,4) is the users DRate (Requested Data Rate, may change)
    % User(1,5) is the users DRate (Actual Data Rate, may change)
    % User(1,6) is the calculated alpha for User 1
    % User(1,7) is 1 if Tc is needed and 0 if it is not
    % User(1,8) is the actual Video Tx Rate that is supported
    % User(1,9) is the MOS of the Video Stream
UData = 9;
User = zeros(n+1, UData);
```

```
%Fixed Values
SRate360p = 1.128; % Service Rate for 360p
SRate480p = 2.628; % Service Rate for 480p
SRate720p = 5.384; % Service Rate for 720p
SRate1080p = 8.384; % Service Rate for 1080p
DDRate360p = round(SRate360p*To,4); %Rounded Desired Data Rate for 360p
DDRate480p = round(SRate480p*To,4); %Rounded Desired Data Rate for 480p
DDRate720p = round(SRate720p*To,4); %Rounded Desired Data Rate for 720p
DDRate1080p = round(SRate1080p*To,4); %Rounded Desired Data Rate for 1080p
DRates = 4; %There are Four Potential Data Rates
DDRate = [DDRate360p; DDRate480p; DDRate720p; DDRate1080p];
%VRate = {'360p'; '480p'; '720p'; '1080p'};
VRate = {'720p'; '1080p'};
VRates = numel(DDRate);
MOS = [3.55; 3.98; 4.31; 4.5];

%Required Input: Maximum Rate for each user based on SNR Table
%(6,9,12,18,24,36,48,54) in Mbps
APRate = [6;9;12;18;24;36;48;54];

%APRates for 10 Users
APRates(1:n,1) = 48; %Set all Users to same AP Rate ( = same SNR)

figure;
hold on;
vr = VRates; %Number of Video Rates

while vr > 2
    m = n; %Total Number of Users
    MaxTc = n+1;
    while MaxTc > 0
        User = zeros(m+1, UData); %Reset Users Array for each run
        n1 = m;%Current Number of Users for this loop
        while n1 > 0
            User(n1,1)=n1;% Set user Number
            User(n1,2)=APRates(n1);% Set all users to have an APRate
            User(n1,3)=DDRate(vr);% Set all users to have an DDRate
            User(n1,4)=User(n1,3);% Set users initial Requested to DDRate
            User(n1,5)=User(n1,3);% Set users initial Date Rate to DDRate
            n1 = n1-1;
        end

        % This Loop makes the LP iterative in order to find the highest
        % Video Data Rate that can be transmitted while ensuring any excess
        % bandwidth can be used by other users
        k = m*4;
        while k > 1
            %fprintf('Starting LP Function %d\n');  %Test Output disabled
            fn = m;
            f = zeros(1,fn+1);
            f(1,fn+1) = -1;
            A = zeros(fn+1,fn+1);
```

```
            A(1,fn+1) = 1;
            b = ones(1,fn+1);
            lb = zeros((fn+1),1);

            while fn > 0
                f(1,fn) = -User(fn,2);
                A(fn+1,fn) = User(fn,2);
                A(1,fn) = 1;
                b(1,fn+1) = User(fn,4);
                fn = fn - 1;
            end

            [X,fmax] = linprog(f,A,b,[],[],lb);
            X; %Test Output disabled
            abs(fmax); %Test Output disabled

            wn = m;
            while wn > 0 %Store the Calculated Value for Actual Data Rate in
the User Array
                User(wn,5) = round(User(wn,2)*X(wn,1),4);
                wn = wn - 1;
            end

%Start Function for changing Users Requested Data Rate if needed
            Reqmax = User(:,4); %build matrix of column 4 values
            Rqmax = max(Reqmax); %Find Max Value in column 4
            Reqmin = User(:,4); %build matrix of column 4 values
            Rqmin = min(Reqmin(Reqmin~=0)); %Find non-Zero Minimum of Column
4
            APRatemin = User(:,2); %build matrix of column 2 values
            APmin = min(APRatemin(APRatemin~=0)); %Find non-Zero Minimum of
Column 2

            kk = m+1;
            while kk > 1
                kk = kk - 1;
                if User(kk,4) == Rqmax && User(kk,4) == Rqmin
                    if User(kk,5) < User(kk,4) && User(kk,5) > 0
                        if User(kk,5) < User(kk,4) && User(kk,5) < DDRate(1)
                            User(kk,4) = 0;
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(2)
                            User(kk,4) = DDRate(1);
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(3)
                            User(kk,4) = DDRate(2);
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(4)
                            User(kk,4) = DDRate(3);
```

```
                        break
                    end
                end
            elseif User(kk,4) == Rqmax
                if User(kk,5) < User(kk,4) && User(kk,5) > 0
                    if User(kk,5) < User(kk,4) && User(kk,5) < DDRate(1)
                        User(kk,4) = 0;
                        break
                    elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(2)
                        User(kk,4) = DDRate(1);
                        break
                    elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(3)
                        User(kk,4) = DDRate(2);
                        break
                    elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(4)
                        User(kk,4) = DDRate(3);
                        break
                    end
                end
            end
        end %End Function for changing Users Requested Data Rate if
needed

        n2 = m;
        while n2 > 0 % Store Final Alpha in User Array
            User(n2,6) = X(n2);
            n2 = n2-1;
        end

        k = k - 1;
    end

    n3 = m;
    TC = 0;
    while n3 > 0 % Set Transcode Value to 1 if needed
        if round(User(n3,3),4) <= round(User(n3,5),4)
            User(n3,7) = 0;
        elseif round(User(n3,5),4) < DDRate(1)
            User(n3,7) = 0;
        elseif TC < MaxTc-1
            User(n3,7) = 1;
            TC = TC + 1;
        elseif TC == MaxTc-1
            User(n3,4) = 0;
            User(n3,5) = 0;
        end
        n3 = n3-1;
    end
```

```
        n4 = m;
        while n4 > 0 %Find Tx Video Rate to Calculate Average ReQoE for the
Access Point
            if User(n4,5) > User(n4,4)
                if User(n4,5) < DDRate(1)
                    User(n4,8) = 0;
                elseif User(n4,5) < DDRate(2)
                    User(n4,8) = DDRate(1);
                elseif User(n4,5) < DDRate(3)
                    User(n4,8) = DDRate(2);
                elseif User(n4,5) < DDRate(4)
                    User(n4,8) = DDRate(3);
                elseif  User(n4,5) > DDRate(4) && User(n4,3) == DDRate(4)
                    User(n4,8) = DDRate(4);
                elseif  User(n4,5) > DDRate(3) && User(n4,3) == DDRate(3)
                    User(n4,8) = DDRate(3);
                elseif  User(n4,5) > DDRate(2) && User(n4,3) == DDRate(2)
                    User(n4,8) = DDRate(2);
                elseif  User(n4,5) > DDRate(1) && User(n4,3) == DDRate(1)
                    User(n4,8) = DDRate(1);
                end
            %Set Values to next lower, if Tx rate is to low to Tx
            elseif  User(n4,5) < User(n4,4) && User(n4,5) > DDRate(3)
                User(n4,8) = DDRate(3);
            elseif  User(n4,5) < User(n4,4) && User(n4,5) > DDRate(2)
                User(n4,8) = DDRate(2);
            elseif User(n4,5) < User(n4,4) && User(n4,5) > DDRate(1)
                User(n4,8) = DDRate(1);
            elseif User(n4,5) < DDRate(1)
                User(n4,8) = 0;
            else
                User(n4,8) = User(n4,4);
            end
            n4 = n4 - 1;
        end

        %Insert MOS work here
        %Determine MOS of each Stream
        apMOS=0;
        n5 = m;
        while n5 > 0
            if User(n5,8) > 0
                %Sets MOS to highest value, if expected data rate = received
data rate
                if User(n5,8) == User(n5,3)
                    User(n5,9) = MOS(4);
                %Loswers MOS by 1 step
                elseif User(n5,3) == DDRate(2) && User(n5,8) == DDRate(1)
                    User(n5,9) = MOS(3);
                elseif User(n5,3) == DDRate(3) && User(n5,8) == DDRate(2)
                    User(n5,9) = MOS(3);
                elseif User(n5,3) == DDRate(4) && User(n5,8) == DDRate(3)
                    User(n5,9) = MOS(3);
```

```
                %Lowers MOS by 2 steps
                elseif User(n5,3) == DDRate(3) && User(n5,8) == DDRate(1)
                    User(n5,9) = MOS(2);
                elseif User(n5,3) == DDRate(4) && User(n5,8) == DDRate(2)
                    User(n5,9) = MOS(2);
                %Lowers MOS by 3 steps (e.g., 1080p -> 360p)
                elseif User(n5,3) == DDRate(4) && User(n5,8) == DDRate(1)
                    User(n5,9) = MOS(1);
                end
            else %MOS is a 1-5 scale, so setting to 1 is the lowest value
                User(n5,9) = 1;
            end
            apMOS = apMOS + User(n5,9);
%             ILPUsers(m,n5,vr) = User(n5,9); %Store Each Users MOS value
            n5 = n5 - 1;
        end

        % Output will be the Results Data Matrix organized such that:
            % Output(1,1) is the Number of Users,
            % Output(1,2) is the Average QoE of the Access Point
            % Output(1,3) is the total number of Transcoded Users
            % Output(1,4) is the Number of Users being Served (maybe <
connected)

        Output(MaxTc,1,vr) = MaxTc-1;
        Output(MaxTc,2,vr) = apMOS/m;
        Output(MaxTc,3,vr) = sum(User(:,7));
        Output(MaxTc,4,vr) = sum(User(:,5)>0);

        MaxTc = MaxTc - 1;
    end

    if vr == 4
        plot(Output(:,1,vr),Output(:,2,vr),'LineWidth',1.5);
    else
        plot(Output(:,1,vr),Output(:,2,vr),'-s','LineWidth',1.5);
    end

    vr = vr - 1;
end

rline = refline([0 3.04]);
rline.Color = 'r';
rline.LineStyle = '-.';
rline.LineWidth = 2;

legend('1080p Transcoding', '720p Transcoding','Nearly All Users
Dissatisfied','Location','southeast');

xlim([0 n+1]);
ylim([1 5]);
```

```
xlabel('Maximum Streams able to be Transcoded');
set(gca,'Xtick',linspace(0,n,n+1));
ylabel('Average ReQoE');
grid on;
set(gca, 'box', 'on');
hold off;
```

A.3.3 No Transcoding vs. Throughput vs. Heuristic at 720p & 1080p Adding Video Streams

The equation `=3.010*exp(-4.473*(User(k,5)/100))+(1.065+0.425)` used to calculate the impact of packet loss on QoE in the following MATLAB m file is a modification of the equation presented in [24].

```
%Required Input:
n = 10; %Number of Users
To = 1.1; %TCP/IP Overhead Percentage Multiplier

% User will be the primary Data Matrix organized such that:
    % User(1,1) is the users Number,
    % User(1,2) is the users APRate (access point rate, static)
    % User(1,3) is the users RDRate (Desired Data rate, static)
    % User(1,4) is the users DRate (Requested Data Rate, may change)
    % User(1,5) is the users DRate (Actual Data Rate, may change)
    % User(1,6) is the calculated alpha for User 1
    % User(1,7) is 1 if Tc is needed and 0 if it is not
    % User(1,8) is the actual Video Tx Rate that is supported
    % User(1,9) is the MOS of the Video Stream
UData = 9;
User = zeros(n+1, UData);

%Fixed Values
SRate360p = 1.128; % Service Rate for 360p
SRate480p = 2.628; % Service Rate for 480p
SRate720p = 5.384; % Service Rate for 720p
SRate1080p = 8.384; % Service Rate for 1080p
DDRate360p = round(SRate360p*To,4); %Rounded Desired Data Rate for 360p
DDRate480p = round(SRate480p*To,4); %Rounded Desired Data Rate for 480p
DDRate720p = round(SRate720p*To,4); %Rounded Desired Data Rate for 720p
DDRate1080p = round(SRate1080p*To,4); %Rounded Desired Data Rate for 1080p
DRates = 4; %There are Four Potential Data Rates
DDRate = [DDRate360p; DDRate480p; DDRate720p; DDRate1080p];
%VRate = {'360p'; '480p'; '720p'; '1080p'};
VRate = {'720p'; '1080p'};
VRates = numel(DDRate);
MOS = [3.55; 3.98; 4.31; 4.5];
```

```
%Required Input: Maximum Rate for each user based on SNR Table
%(6,9,12,18,24,36,48,54) in Mbps
APRate = [6;9;12;18;24;36;48;54];

%APRates for 10 Users
APRates(1:n,1) = 12; %Set all Users to same AP Rate ( = same SNR)

figure;
hold on;
vr = VRates; %Number of Video Rates

%
%
%     Run Scenario with No SDN, No Optimization, and No-Transcoding
%     This is done by treating all Data request above the limit of the
%     access point as Packet Loss
%
%

set(gca,'ColorOrderIndex',1) %Reset Plot Color Order
vr = VRates; %Number of Video Rates

while vr > 2
    m = n; %Total Number of Users
    while m > 0
        User = zeros(m, UData); %Reset Users Array for each run
        n1 = m; %Current Number of Users for this loop
        while n1 > 0
            User(n1,1) = n1;% Set user Number
            User(n1,2) = APRates(n1);% Set all users to have an APRate
            User(n1,3) = DDRate(vr);% Set all users to have an DDRate
            User(n1,4) = (User(n1,3))/(User(n1,2)); %alpha = Desired data /
AP Rate
            User(n1,5) = 0;% Set initial Packet Loss Rate
            User(n1,6) = 0;% Set initial MOS per User
            n1 = n1 - 1;
        end

        sumA = sum(User(:,4));
        MaxRate = (sum(User(:,2)/m)/m);  %May not always work... check

        if sumA > 1
            k = m; %Current Number of Users
            while k > 0 %Calculate % Packet Loss per User
                User(k,5) = abs((1 - (MaxRate / User(k,3)))*100);  % ==
Percent Packet Loss
                User(k,6) = 3.010*exp(-4.473*(User(k,5)/100))+(1.065+0.425);
                k = k - 1;
            end
        elseif sumA <= 1
            User(:,6) = MOS(4);
```

```
        end

        n2 = m;
        apMOS = 0;
        while n2 > 0 % Average MOS of the AP
            apMOS = apMOS + (User(n2,6));
            NLPUsers(m,n2,vr) = User(n2,9); %Store Each Users MOS value
            n2 = n2-1;
        end

        % Output will be the Results Data Matrix organized such that:
            % Output(1,1) is the Number of Users,
            % Output(1,2) is the Average QoE of the Access Point
            % Output(1,3) is the total number of Transcoded Users
            % Output(1,4) is the Number of Users being Served (maybe <
connected)
        Output(m,1,vr) = m;
        Output(m,2,vr) = (apMOS / m); %For Mean MOS
        m = m - 1;
    end

    if vr == 4
        NLP = plot(Output(:,1,vr),Output(:,2,vr),'--','LineWidth',1.5);
    else
        NLP = plot(Output(:,1,vr),Output(:,2,vr),'--o','LineWidth',1.5);
    end

    vr = vr - 1;
end

%
%
%     Break, to run Transcoding with Throughput Optimization
%
%

set(gca,'ColorOrderIndex',1) %Reset Plot Color Order
vr = VRates; %Number of Video Rates
while vr > 2
    m = n; %Total Number of Users
    while m > 0
        User = zeros(m+1, UData); %Reset Users Array for each run
        n1 = m;%Current Number of Users for this loop
        while n1 > 0
            User(n1,1)=n1;% Set user Number
            User(n1,2)=APRates(n1);% Set all users to have an APRate
            User(n1,3)=DDRate(vr);% Set all users to have an DDRate
            User(n1,4)=User(n1,3);% Set users initial Requested to DDRate
            User(n1,5)=User(n1,3);% Set users initial Date Rate to DDRate
            n1 = n1-1;
        end
```

```
            fn = m;  %Initialize LP Parameter Matrices
            f = zeros(1,fn+1);
            f(1,fn+1) = -1;
            A = zeros(fn+1,fn+1);
            A(1,fn+1) = 1;
            b = ones(1,fn+1);
            lb = zeros((fn+1),1);

            while fn > 0  %Set LP Parameters
                f(1,fn) = -User(fn,2);
                A(fn+1,fn) = User(fn,2);
                A(1,fn) = 1;
                b(1,fn+1) = User(fn,4);
                fn = fn - 1;
            end

            [X,fmax] = linprog(f,A,b,[],[],lb);
            X; %Test Output disabled
            abs(fmax); %Test Output disabled

            wn = m;
            while wn > 0 %Store the Calculated Value for Actual Data Rate in
the User Array
                User(wn,5) = round(User(wn,2)*X(wn,1),4);
                wn = wn - 1;
            end

           n2 = m;
           while n2 > 0 % Store Final Alpha in User Array
               User(n2,6) = X(n2);
               n2 = n2-1;
           end

        n3 = m;
        while n3 > 0 % Set Transcode Value to 1 if needed
            if round(User(n3,3),4) <= round(User(n3,5),4)
                User(n3,7) = 0;
            elseif round(User(n3,5),4) < DDRate(1)
                User(n3,7) = 0;
            else
                User(n3,7) = 1;
            end
            n3 = n3-1;
        end

        n4 = m;
        while n4 > 0 %Find Tx Video Rate to Calculate Average ReQoE for the
Access Point
            if User(n4,5) > User(n4,4)
                if User(n4,5) < DDRate(1)
```

```
                    User(n4,8) = 0;
                elseif User(n4,5) < DDRate(2)
                    User(n4,8) = DDRate(1);
                elseif User(n4,5) < DDRate(3)
                    User(n4,8) = DDRate(2);
                elseif User(n4,5) < DDRate(4)
                    User(n4,8) = DDRate(3);
                elseif  User(n4,5) > DDRate(4) && User(n4,3) == DDRate(4)
                    User(n4,8) = DDRate(4);
                elseif  User(n4,5) > DDRate(3) && User(n4,3) == DDRate(3)
                    User(n4,8) = DDRate(3);
                elseif  User(n4,5) > DDRate(2) && User(n4,3) == DDRate(2)
                    User(n4,8) = DDRate(2);
                elseif  User(n4,5) > DDRate(1) && User(n4,3) == DDRate(1)
                    User(n4,8) = DDRate(1);
                end
            %Set Values to next lower, if Tx rate is to low to Tx
            elseif  User(n4,5) < User(n4,4) && User(n4,5) > DDRate(3)
                User(n4,8) = DDRate(3);
            elseif  User(n4,5) < User(n4,4) && User(n4,5) > DDRate(2)
                User(n4,8) = DDRate(2);
            elseif User(n4,5) < User(n4,4) && User(n4,5) > DDRate(1)
                User(n4,8) = DDRate(1);
            elseif User(n4,5) < DDRate(1)
                User(n4,8) = 0;
            else
                User(n4,8) = User(n4,4);
            end
            n4 = n4 - 1;
        end

        %Insert MOS work here to determine MOS of each Stream
        apMOS=0;
        n5 = m;
        while n5 > 0
            if User(n5,8) > 0
                %Sets MOS to highest value, if expected data rate = received
data rate
                if User(n5,8) == User(n5,3)
                    User(n5,9) = MOS(4);
                %Loswers MOS by 1 step
                elseif User(n5,3) == DDRate(2) && User(n5,8) == DDRate(1)
                    User(n5,9) = MOS(3);
                elseif User(n5,3) == DDRate(3) && User(n5,8) == DDRate(2)
                    User(n5,9) = MOS(3);
                elseif User(n5,3) == DDRate(4) && User(n5,8) == DDRate(3)
                    User(n5,9) = MOS(3);
                %Lowers MOS by 2 steps
                elseif User(n5,3) == DDRate(3) && User(n5,8) == DDRate(1)
                    User(n5,9) = MOS(2);
                elseif User(n5,3) == DDRate(4) && User(n5,8) == DDRate(2)
                    User(n5,9) = MOS(2);
                %Lowers MOS by 3 steps (e.g., 1080p -> 360p)
```

```
                elseif User(n5,3) == DDRate(4) && User(n5,8) == DDRate(1)
                    User(n5,9) = MOS(1);
                end
            else %MOS is a 1-5 scale, so setting to 1 is the lowest value
                User(n5,9) = 1;
            end
            apMOS = apMOS + User(n5,9);
            BLPUsers(m,n5,vr) = User(n5,9); %Store Each Users MOS value
            n5 = n5 - 1;
        end

        % Output will be the Results Data Matrix organized such that:
            % Output(1,1) is the Number of Users,
            % Output(1,2) is the Average QoE of the Access Point
            % Output(1,3) is the total number of Transcoded Users
            % Output(1,4) is the Number of Users being Served (maybe <
connected)

        Output(m,1,vr) = m;
        Output(m,2,vr) = apMOS/m;
        Output(m,3,vr) = sum(User(:,7));
        Output(m,4,vr) = sum(User(:,5)>0);

        m = m - 1;
    end

    %BLP = plot(Output(:,1,vr),Output(:,2,vr),':','LineWidth',1.5);

    if vr == 4
        BLP = plot(Output(:,1,vr),Output(:,2,vr),':','LineWidth',1.5);
    else
        BLP = plot(Output(:,1,vr),Output(:,2,vr),':x','LineWidth',1.5);
        end


vr = vr - 1;
end

%
%
%BREAK to run Transcoding with Heuristic algorithm
%
%
set(gca,'ColorOrderIndex',1) %Reset Plot Color Order
vr = VRates; %Number of Video Rates
while vr > 2
    m = n; %Total Number of Users
    while m > 0
        User = zeros(m+1, UData); %Reset Users Array for each run
        n1 = m;%Current Number of Users for this loop
        while n1 > 0
```

```
            User(n1,1)=n1;% Set user Number
            User(n1,2)=APRates(n1);% Set all users to have an APRate
            User(n1,3)=DDRate(vr);% Set all users to have an DDRate
            User(n1,4)=User(n1,3);% Set users initial Requested to DDRate
            User(n1,5)=User(n1,3);% Set users initial Date Rate to DDRate
            n1 = n1-1;
        end

        % This Loop makes the LP iterative in order to find the highest
        % Video Data Rate that can be transmitted while ensuring any excess
        % bandwidth can be used by other users
        k = m*4;
        while k > 1
            fn = m;
            f = zeros(1,fn+1);
            f(1,fn+1) = -1; %0 or -1 results appear the same
            A = zeros(fn+1,fn+1);
            A(1,fn+1) = 1;
            b = ones(1,fn+1);
            lb = zeros((fn+1),1);

            while fn > 0
                f(1,fn) = -User(fn,2);
                A(fn+1,fn) = User(fn,2);
                A(1,fn) = 1;
                b(1,fn+1) = User(fn,4);
                fn = fn - 1;
            end

            [X,fmax] = linprog(f,A,b,[],[],lb);
            X; %Test Output disabled
            abs(fmax); %Test Output disabled

            wn = m;
            while wn > 0 %Store the Calculated Value for Actual Data Rate in
the User Array
                User(wn,5) = round(User(wn,2)*X(wn,1),4);
                wn = wn - 1;
            end

%Start Function for changing Users Requested Data Rate if needed
            Reqmax = User(:,4); %build matrix of column 4 values
            Rqmax = max(Reqmax); %Find Max Value in column 4
            Reqmin = User(:,4); %build matrix of column 4 values
            Rqmin = min(Reqmin(Reqmin~=0)); %Find non-Zero Minimum of Column
4
            APRatemin = User(:,2); %build matrix of column 2 values
            APmin = min(APRatemin(APRatemin~=0)); %Find non-Zero Minimum of
Column 2

            kk = m+1;
            while kk > 1
```

```
                kk = kk - 1;
                if User(kk,4) == Rqmax && User(kk,4) == Rqmin
                    if User(kk,5) < User(kk,4) && User(kk,5) > 0
                        if User(kk,5) < User(kk,4) && User(kk,5) < DDRate(1)
                            User(kk,4) = 0;
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(2)
                            User(kk,4) = DDRate(1);
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(3)
                            User(kk,4) = DDRate(2);
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(4)
                            User(kk,4) = DDRate(3);
                            break
                        end
                    end
                elseif User(kk,4) == Rqmax
                    if User(kk,5) < User(kk,4) && User(kk,5) > 0
                        if User(kk,5) < User(kk,4) && User(kk,5) < DDRate(1)
                            User(kk,4) = 0;
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(2)
                            User(kk,4) = DDRate(1);
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(3)
                            User(kk,4) = DDRate(2);
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(4)
                            User(kk,4) = DDRate(3);
                            break
                        end
                    end
                end
            end %End Function for changing Users Requested Data Rate if
needed

            n2 = m;
            while n2 > 0 % Store Final Alpha in User Array
                User(n2,6) = X(n2);
                n2 = n2-1;
            end

            k = k - 1;
        end
```

```
        n3 = m;
        while n3 > 0 % Set Transcode Value to 1 if needed
            if round(User(n3,3),4) <= round(User(n3,5),4)
                User(n3,7) = 0;
            elseif round(User(n3,5),4) < DDRate(1)
                User(n3,7) = 0;
            else
                User(n3,7) = 1;
            end
            n3 = n3-1;
        end

        n4 = m;
        while n4 > 0 %Find Tx Video Rate to Calculate Average ReQoE for the
Access Point
            if User(n4,5) > User(n4,4)
                if User(n4,5) < DDRate(1)
                    User(n4,8) = 0;
                elseif User(n4,5) < DDRate(2)
                    User(n4,8) = DDRate(1);
                elseif User(n4,5) < DDRate(3)
                    User(n4,8) = DDRate(2);
                elseif User(n4,5) < DDRate(4)
                    User(n4,8) = DDRate(3);
                elseif  User(n4,5) > DDRate(4) && User(n4,3) == DDRate(4)
                    User(n4,8) = DDRate(4);
                elseif  User(n4,5) > DDRate(3) && User(n4,3) == DDRate(3)
                    User(n4,8) = DDRate(3);
                elseif  User(n4,5) > DDRate(2) && User(n4,3) == DDRate(2)
                    User(n4,8) = DDRate(2);
                elseif  User(n4,5) > DDRate(1) && User(n4,3) == DDRate(1)
                    User(n4,8) = DDRate(1);
                end
            %Set Values to next lower, if Tx rate is to low to Tx
            elseif  User(n4,5) < User(n4,4) && User(n4,5) > DDRate(3)
                User(n4,8) = DDRate(3);
            elseif  User(n4,5) < User(n4,4) && User(n4,5) > DDRate(2)
                User(n4,8) = DDRate(2);
            elseif User(n4,5) < User(n4,4) && User(n4,5) > DDRate(1)
                User(n4,8) = DDRate(1);
            elseif User(n4,5) < DDRate(1)
                User(n4,8) = 0;
            else
                User(n4,8) = User(n4,4);
            end
            n4 = n4 - 1;
        end

        %Insert MOS work here
        %Determine MOS of each Stream
        apMOS=0;
        n5 = m;
```

```
        while n5 > 0
            if User(n5,8) > 0
                %Sets MOS to highest value, if expected data rate = received
data rate
                if User(n5,8) == User(n5,3)
                    User(n5,9) = MOS(4);
                %Loswers MOS by 1 step
                elseif User(n5,3) == DDRate(2) && User(n5,8) == DDRate(1)
                    User(n5,9) = MOS(3);
                elseif User(n5,3) == DDRate(3) && User(n5,8) == DDRate(2)
                    User(n5,9) = MOS(3);
                elseif User(n5,3) == DDRate(4) && User(n5,8) == DDRate(3)
                    User(n5,9) = MOS(3);
                %Lowers MOS by 2 steps
                elseif User(n5,3) == DDRate(3) && User(n5,8) == DDRate(1)
                    User(n5,9) = MOS(2);
                elseif User(n5,3) == DDRate(4) && User(n5,8) == DDRate(2)
                    User(n5,9) = MOS(2);
                %Lowers MOS by 3 steps (e.g., 1080p -> 360p)
                elseif User(n5,3) == DDRate(4) && User(n5,8) == DDRate(1)
                    User(n5,9) = MOS(1);
                end
            else %MOS is a 1-5 scale, so setting to 1 is the lowest value
                User(n5,9) = 1;
            end
            apMOS = apMOS + User(n5,9);
            ILPUsers(m,n5,vr) = User(n5,9); %Store Each Users MOS value
            n5 = n5 - 1;
        end

        % Output will be the Results Data Matrix organized such that:
            % Output(1,1) is the Number of Users,
            % Output(1,2) is the Average QoE of the Access Point
            % Output(1,3) is the total number of Transcoded Users
            % Output(1,4) is the Number of Users being Served (maybe <
connected)

        Output(m,1,vr) = m;
        Output(m,2,vr) = apMOS/m;
        Output(m,3,vr) = sum(User(:,7));
        Output(m,4,vr) = sum(User(:,5)>0);

        m = m - 1;
    end

    if vr == 4
        ILP = plot(Output(:,1,vr),Output(:,2,vr),'LineWidth',1.5);
    else
        ILP = plot(Output(:,1,vr),Output(:,2,vr),'-v','LineWidth',1.5);
    end

vr = vr - 1;
```

```
end

rline = refline([0 3.04]);
rline.Color = 'r';
rline.LineStyle = '-.';
rline.LineWidth = 2;

flprate = flipud(VRate);
%legend(flprate);
legend('1080p No Transcoding', '720p No Transcoding', '1080p Throughput Opt',
'720p Throughput Opt', '1080p Heuristic Opt', '720p Heuristic Opt',  'Nearly
All Dissatisfied');

xlim([0 n+1]);
ylim([1 5]);
set(gca,'XTick',0:1:(n+1));
xlabel('Number of Video Streams');
ylabel('Average ReQoE');
set(gca, 'box', 'on');
grid on;
hold off;
```

A.4 Additional Results using the Heuristic Algorithm

A.4.1 Bar Graphs

```
%Required Input:
n = 10; %Number of Users
To = 1.1; %TCP/IP Overhead Percentage Multiplier

% User will be the primary Data Matrix organized such that:
    % User(1,1) is the users Number,
    % User(1,2) is the users APRate (access point rate, static)
    % User(1,3) is the users RDRate (Desired Data rate, static)
    % User(1,4) is the users DRate (Requested Data Rate, may change)
    % User(1,5) is the users DRate (Actual Data Rate, may change)
    % User(1,6) is the calculated alpha for User 1
    % User(1,7) is 1 if Tc is needed and 0 if it is not
    % User(1,8) is the actual Video Tx Rate that is supported
    % User(1,9) is the MOS of the Video Stream
UData = 9;
User = zeros(n+1, UData);

%Fixed Values
SRate360p = 1.128; % Service Rate for 360p
```

```
SRate480p = 2.628; % Service Rate for 480p
SRate720p = 5.384; % Service Rate for 720p
SRate1080p = 8.384; % Service Rate for 1080p
DDRate360p = round(SRate360p*To,4); %Rounded Desired Data Rate for 360p
DDRate480p = round(SRate480p*To,4); %Rounded Desired Data Rate for 480p
DDRate720p = round(SRate720p*To,4); %Rounded Desired Data Rate for 720p
DDRate1080p = round(SRate1080p*To,4); %Rounded Desired Data Rate for 1080p
DDRate = [DDRate360p; DDRate480p; DDRate720p; DDRate1080p];
VRate = {'360p'; '480p'; '720p'; '1080p'};
VRates = numel(DDRate);
MOS = [3.55; 3.98; 4.31; 4.5];

%Required Input: Maximum Rate for each user based on SNR Table
%(6,9,12,18,24,36,48,54) in Mbps
%Set all Users to same AP Rate ( = same SNR)
APRates(1:n,1) = 18;

figure;
hold on;
vr = VRates; %Number of Video Rates
while vr > 2
    m = n; %Total Number of Users
    while m > 0
        User = zeros(m+1, UData); %Reset Users Array for each run
        n1 = m;%Current Number of Users for this loop
        while n1 > 0
            User(n1,1)=n1;% Set user Number
            User(n1,2)=APRates(n1);% Set all users to have an APRate
            User(n1,3)=DDRate(vr);% Set all users to have an DDRate
            User(n1,4)=User(n1,3);% Set users initial Requested to DDRate
            User(n1,5)=User(n1,3);% Set users initial Date Rate to DDRate
            n1 = n1-1;
        end

        % This Loop makes the LP iterative in order to find the highest
        % Video Data Rate that can be transmitted while ensuring any excess
        % bandwidth can be used by other users
        k = m*4;
        kk = m;
        while k > 1
            fn = m;
            f = zeros(1,fn+1);
            f(1,fn+1) = 0;
            A = zeros(fn+1,fn+1);
            A(1,fn+1) = 1;
            b = ones(1,fn+1);
            lb = zeros((fn+1),1);

            while fn > 0
                f(1,fn) = -User(fn,2);
                A(fn+1,fn) = User(fn,2);
                A(1,fn) = 1;
```

```
            b(1,fn+1) = User(fn,4);
            fn = fn - 1;
        end

        [X,fmax] = linprog(f,A,b,[],[],lb);
        abs(fmax); %Test Output disabled

        wn = m;
        while wn > 0 %Store the Calculated Value for Actual Data Rate in
the User Array
            User(wn,5) = round(User(wn,2)*X(wn,1),4);
            wn = wn - 1;
        end

        Reqmax = User(:,4); %build matrix of column 4 values
        Rqmax = max(Reqmax); %Find Max Value in column 4
        Reqmin = User(:,4); %build matrix of column 4 values
        Rqmin = min(Reqmin(Reqmin~=0)); %Find non-Zero Minimum of Column
4
        APRatemin = User(:,2); %build matrix of column 2 values
        APmin = min(APRatemin(APRatemin~=0)); %Find non-Zero Minimum of
Column 2


        kk = m+1;
        while kk > 1
            kk = kk - 1;
            if User(kk,4) == Rqmax && User(kk,4) == Rqmin
                if User(kk,5) < User(kk,4) && User(kk,5) > 0
                    if User(kk,5) < User(kk,4) && User(kk,5) < DDRate(1)
                        User(kk,4) = 0;
                        break
                    elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(2)
                        User(kk,4) = DDRate(1);
                        break
                    elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(3)
                        User(kk,4) = DDRate(2);
                        break
                    elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(4)
                        User(kk,4) = DDRate(3);
                        break
                    end
                end
            elseif User(kk,4) == Rqmax
                if User(kk,5) < User(kk,4) && User(kk,5) > 0
                    if User(kk,5) < User(kk,4) && User(kk,5) < DDRate(1)
                        User(kk,4) = 0;
                        break
```

```
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(2)
                            User(kk,4) = DDRate(1);
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(3)
                            User(kk,4) = DDRate(2);
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(4)
                            User(kk,4) = DDRate(3);
                            break
                        end
                    end
                end
            end

            n2 = m;
            while n2 > 0 % Store Final Alpha in User Array
                User(n2,6) = X(n2);
                n2 = n2-1;
            end

            k = k - 1;
        end

        n3 = m;
        while n3 > 0 % Set Transcode Value to 1 if needed
            if round(User(n3,3),4) <= round(User(n3,5),4)
                User(n3,7) = 0;
            elseif round(User(n3,5),4) < DDRate(1)
                User(n3,7) = 0;
            else
                User(n3,7) = 1;
            end
            n3 = n3-1;
        end

        n4 = m;
        while n4 > 0 %Find Tx Video Rate to Calculate Average ReQoE for the
Access Point
            if User(n4,5) > User(n4,4)
                if User(n4,5) < DDRate(1)
                    User(n4,8) = 0;
                elseif User(n4,5) < DDRate(2)
                    User(n4,8) = DDRate(1);
                elseif User(n4,5) < DDRate(3)
                    User(n4,8) = DDRate(2);
                elseif User(n4,5) < DDRate(4)
                    User(n4,8) = DDRate(3);
                elseif  User(n4,5) > DDRate(4) && User(n4,3) == DDRate(4)
                    User(n4,8) = DDRate(4);
```

```
                elseif  User(n4,5) > DDRate(3) && User(n4,3) == DDRate(3)
                    User(n4,8) = DDRate(3);
                elseif  User(n4,5) > DDRate(2) && User(n4,3) == DDRate(2)
                    User(n4,8) = DDRate(2);
                elseif  User(n4,5) > DDRate(1) && User(n4,3) == DDRate(1)
                    User(n4,8) = DDRate(1);
                end
            %Set Values to next cloest, if Tx rate is to low to Tx
            elseif  User(n4,5) < User(n4,4) && User(n4,5) > DDRate(3)
                User(n4,8) = DDRate(3);
            elseif  User(n4,5) < User(n4,4) && User(n4,5) > DDRate(2)
                User(n4,8) = DDRate(2);
            elseif User(n4,5) < User(n4,4) && User(n4,5) > DDRate(1)
                User(n4,8) = DDRate(1);
            elseif User(n4,5) < DDRate(1)
                User(n4,8) = 0;
            else
                User(n4,8) = User(n4,4);
            end
            n4 = n4 - 1;
        end

        %Insert MOS work here
        %Determine MOS of each Stream
        apMOS=0;
        n5 = m;
        while n5 > 0
            if User(n5,8) > 0
                %Sets MOS to highest value, if expected data rate = received
data rate
                if User(n5,8) == User(n5,3)
                    User(n5,9) = MOS(4);
                %Loswers MOS by 1 step
                elseif User(n5,3) == DDRate(2) && User(n5,8) == DDRate(1)
                    User(n5,9) = MOS(3);
                elseif User(n5,3) == DDRate(3) && User(n5,8) == DDRate(2)
                    User(n5,9) = MOS(3);
                elseif User(n5,3) == DDRate(4) && User(n5,8) == DDRate(3)
                    User(n5,9) = MOS(3);
                %Lowers MOS by 2 steps
                elseif User(n5,3) == DDRate(3) && User(n5,8) == DDRate(1)
                    User(n5,9) = MOS(2);
                elseif User(n5,3) == DDRate(4) && User(n5,8) == DDRate(2)
                    User(n5,9) = MOS(2);
                %Lowers MOS by 3 steps (e.g., 1080p -> 360p)
                elseif User(n5,3) == DDRate(4) && User(n5,8) == DDRate(1)
                    User(n5,9) = MOS(1);
                end
            else %MOS is a 1-5 scale, so setting to 1 is the lowest value
                User(n5,9) = 1;
            end
            apMOS = apMOS + User(n5,9);
            Users(m,n5,vr) = User(n5,9); %Store Each Users MOS value
```

```
            n5 = n5 - 1;
        end

        % Output will be the Results Data Matrix organized such that:
            % Output(1,1) is the Number of Users,
            % Output(1,2) is the Average QoE of the Access Point
            % Output(1,3) is the total number of Transcoded Users
            % Output(1,4) is the Number of Users being Served (maybe <
connected)

        Output(m,1,vr) = m;
        Output(m,2,vr) = apMOS/m;
        Output(m,3,vr) = sum(User(:,7));
        Output(m,4,vr) = sum(User(:,5)>0);

User
        m = m - 1;
    end

    plot(Output(:,1,vr),Output(:,2,vr),'LineWidth',1.5);

vr = vr - 1;
end

flprate = flipud(VRate);
legend(flprate);

xlim([0 n+1]);
ylim([1 5]);
xlabel('Number of Users Trying to Stream Video');
ylabel('Average ReQoE');
title({['Throughput Optimized Access Point w/ & w/o Transcoding'], ['Each
User Joins with a different SNR, a -- Line Means No Transcoding']});
grid on;
box on;
hold off;

vr1 = VRates;
while vr1 > 2
    figure;
    bar(Users(:,:,vr1));
    xlim([0 n+1]);
    ylim([1 5]);
    xlabel('Number of Users Trying to Stream Video');
    ylabel('ReQoE of Each User Receiving Video');
    title({['Numbers of Users Requesting Service vs MOS of Users Receiving
Service'], ['With Each User Requesting a Video Rate of = ', VRate{vr1}]});
    %# Add a text string above each bin
    i = n;
    while i > 0
```

```
        if i > 1;
            text(i-.5, 4.75, [num2str(nnz(Users(i,:,vr1))), ' Users'],
'VerticalAlignment', 'top', 'FontSize', 8)
        else
            text(i-.5, 4.75, [num2str(nnz(Users(i,:,vr1))), ' User'],
'VerticalAlignment', 'top', 'FontSize', 8)
        end
        i = i - 1;
    end
    vr1 = vr1 - 1;
end
```

A.4.2 Three Output Graphs

```
%Required Input:
n = 10; %Number of Users
To = 1.1; %TCP/IP Overhead Percentage Multiplier

% User will be the primary Data Matrix organized such that:
    % User(1,1) is the users Number,
    % User(1,2) is the users APRate (access point rate, static)
    % User(1,3) is the users RDRate (Desired Data rate, static)
    % User(1,4) is the users DRate (Requested Data Rate, may change)
    % User(1,5) is the users DRate (Actual Data Rate, may change)
    % User(1,6) is the calculated alpha for User 1
    % User(1,7) is 1 if Tc is needed and 0 if it is not
    % User(1,8) is the actual Video Tx Rate that is supported
    % User(1,9) is the MOS of the Video Stream
UData = 9;
User = zeros(n+1, UData);

%Fixed Values
SRate360p = 1.128; % Service Rate for 360p
SRate480p = 2.628; % Service Rate for 480p
SRate720p = 5.384; % Service Rate for 720p
SRate1080p = 8.384; % Service Rate for 1080p
DDRate360p = round(SRate360p*To,4); %Rounded Desired Data Rate for 360p
DDRate480p = round(SRate480p*To,4); %Rounded Desired Data Rate for 480p
DDRate720p = round(SRate720p*To,4); %Rounded Desired Data Rate for 720p
DDRate1080p = round(SRate1080p*To,4); %Rounded Desired Data Rate for 1080p
DDRate = [DDRate360p; DDRate480p; DDRate720p; DDRate1080p];
VRate = {'360p'; '480p'; '720p'; '1080p'};
VRates = numel(DDRate);
MOS = [3.55; 3.98; 4.31; 4.5];

%Required Input: Maximum Rate for each user based on SNR Table
%(6,9,12,18,24,36,48,54) in Mbps
%Set all Users to same AP Rate ( = same SNR)
```

```
APRates(1:n,1) = 24;

vr = 4; %Requested Resolution
MaxTc = 3; %Number of Available Transcoders

    m = n; %Total Number of Users
    while m > 0
        User = zeros(m+1, UData); %Reset Users Array for each run
        n1 = m;%Current Number of Users for this loop
        while n1 > 0
            User(n1,1)=n1;% Set user Number
            User(n1,2)=APRates(n1);% Set all users to have an APRate
            User(n1,3)=DDRate(vr);% Set all users to have an DDRate
            User(n1,4)=User(n1,3);% Set users initial Requested to DDRate
            User(n1,5)=User(n1,3);% Set users initial Date Rate to DDRate
            n1 = n1-1;
        end

        % This Loop makes the LP iterative in order to find the highest
        % Video Data Rate that can be transmitted while ensuring any excess
        % bandwidth can be used by other users
        k = m*4;
        kk = m;
        while k > 1
            fn = m;
            f = zeros(1,fn+1);
            f(1,fn+1) = -1; %0 or -1 results appear the same
            A = zeros(fn+1,fn+1);
            A(1,fn+1) = 1;
            b = ones(1,fn+1);
            lb = zeros((fn+1),1);

            while fn > 0
                f(1,fn) = -User(fn,2);
                A(fn+1,fn) = User(fn,2);
                A(1,fn) = 1;
                b(1,fn+1) = User(fn,4);
                fn = fn - 1;
            end

            [X,fmax] = linprog(f,A,b,[],[],lb);
            X %Test Output disabled
            abs(fmax); %Test Output disabled

            wn = m;
            while wn > 0 %Store the Calculated Value for Actual Data Rate in
the User Array
                User(wn,5) = round(User(wn,2)*X(wn,1),4);
                wn = wn - 1;
            end
```

```
            Reqmax = User(:,4); %build matrix of column 4 values
            Rqmax = max(Reqmax); %Find Max Value in column 4
            Reqmin = User(:,4); %build matrix of column 4 values
            Rqmin = min(Reqmin(Reqmin~=0)); %Find non-Zero Minimum of Column
4
            APRatemin = User(:,2); %build matrix of column 2 values
            APmin = min(APRatemin(APRatemin~=0)); %Find non-Zero Minimum of
Column 2


            kk = m+1;
            while kk > 1
                kk = kk - 1;
                if User(kk,4) == Rqmax && User(kk,4) == Rqmin
                    if User(kk,5) < User(kk,4) && User(kk,5) > 0
                        if User(kk,5) < User(kk,4) && User(kk,5) < DDRate(1)
                            User(kk,4) = 0;
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(2)
                            User(kk,4) = DDRate(1);
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(3)
                            User(kk,4) = DDRate(2);
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(4)
                            User(kk,4) = DDRate(3);
                            break
                        end
                    end
                elseif User(kk,4) == Rqmax
                    if User(kk,5) < User(kk,4) && User(kk,5) > 0
                        if User(kk,5) < User(kk,4) && User(kk,5) < DDRate(1)
                            User(kk,4) = 0;
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(2)
                            User(kk,4) = DDRate(1);
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(3)
                            User(kk,4) = DDRate(2);
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(4)
                            User(kk,4) = DDRate(3);
                            break
                        end
                    end
                end
```

```
            end

            n2 = m;
            while n2 > 0 % Store Final Alpha in User Array
                User(n2,6) = X(n2);
                n2 = n2-1;
            end



        n3 = m;
        TC = 0;
        while n3 > 0 % Set Transcode Value to 1 if needed
            if round(User(n3,3),4) <= round(User(n3,5),4)
                User(n3,7) = 0;
            elseif round(User(n3,5),4) < DDRate(1)
                User(n3,7) = 0;
            elseif TC < MaxTc
                User(n3,7) = 1;
                TC = TC + 1;
            elseif TC == MaxTc
                User(n3,4) = 0;
                User(n3,5) = 0;

            end

            n3 = n3-1;
        end



            k = k - 1;
        end



        n4 = m;
        while n4 > 0 %Find Tx Video Rate to Calculate Average ReQoE for the
Access Point
            if User(n4,5) > User(n4,4)
                if User(n4,5) < DDRate(1)
                    User(n4,8) = 0;
                elseif User(n4,5) < DDRate(2)
                    User(n4,8) = DDRate(1);
                elseif User(n4,5) < DDRate(3)
                    User(n4,8) = DDRate(2);
                elseif User(n4,5) < DDRate(4)
                    User(n4,8) = DDRate(3);
                elseif  User(n4,5) > DDRate(4) && User(n4,3) == DDRate(4)
                    User(n4,8) = DDRate(4);
```

```
                elseif  User(n4,5) > DDRate(3) && User(n4,3) == DDRate(3)
                    User(n4,8) = DDRate(3);
                elseif  User(n4,5) > DDRate(2) && User(n4,3) == DDRate(2)
                    User(n4,8) = DDRate(2);
                elseif  User(n4,5) > DDRate(1) && User(n4,3) == DDRate(1)
                    User(n4,8) = DDRate(1);
                end
            %Set Values to next cloest, if Tx rate is to low to Tx
            elseif  User(n4,5) < User(n4,4) && User(n4,5) > DDRate(3)
                User(n4,8) = DDRate(3);
            elseif  User(n4,5) < User(n4,4) && User(n4,5) > DDRate(2)
                User(n4,8) = DDRate(2);
            elseif User(n4,5) < User(n4,4) && User(n4,5) > DDRate(1)
                User(n4,8) = DDRate(1);
            elseif User(n4,5) < DDRate(1)
                User(n4,8) = 0;
            else
                User(n4,8) = User(n4,4);
            end
            n4 = n4 - 1;
        end

        %Insert MOS work here
        %Determine MOS of each Stream
        apMOS=0;
        n5 = m;
        while n5 > 0
            if User(n5,8) > 0
                %Sets MOS to highest value, if expected data rate = received
data rate
                if User(n5,8) == User(n5,3)
                    User(n5,9) = MOS(4);
                %Loswers MOS by 1 step
                elseif User(n5,3) == DDRate(2) && User(n5,8) == DDRate(1)
                    User(n5,9) = MOS(3);
                elseif User(n5,3) == DDRate(3) && User(n5,8) == DDRate(2)
                    User(n5,9) = MOS(3);
                elseif User(n5,3) == DDRate(4) && User(n5,8) == DDRate(3)
                    User(n5,9) = MOS(3);
                %Lowers MOS by 2 steps
                elseif User(n5,3) == DDRate(3) && User(n5,8) == DDRate(1)
                    User(n5,9) = MOS(2);
                elseif User(n5,3) == DDRate(4) && User(n5,8) == DDRate(2)
                    User(n5,9) = MOS(2);
                %Lowers MOS by 3 steps (e.g., 1080p -> 360p)
                elseif User(n5,3) == DDRate(4) && User(n5,8) == DDRate(1)
                    User(n5,9) = MOS(1);
                end
            else %MOS is a 1-5 scale, so setting to 1 is the lowest value
                User(n5,9) = 1;
            end
            apMOS = apMOS + User(n5,9);
            Users(m,n5,vr) = User(n5,9); %Store Each Users MOS value
```

```
            n5 = n5 - 1;
        end

        % Output will be the Results Data Matrix organized such that:
            % Output(1,1) is the Number of Users,
            % Output(1,2) is the Average QoE of the Access Point
            % Output(1,3) is the total number of Transcoded Users
            % Output(1,4) is the Number of Users being Served (maybe <
connected)

        Output(m,1,vr) = m;
        Output(m,2,vr) = apMOS/m;
        Output(m,3,vr) = sum(User(:,7));
        Output(m,4,vr) = sum(User(:,8)>0);
        User
        m = m - 1;
    end

figure;
hold on;
[ax,h1,h2] =
plotyy(Output(:,1,vr),Output(:,2,vr),Output(:,1,vr),Output(:,4,vr),'plot','ba
r');

h2.FaceColor = [.7 .7 .7];  % grey
h2.BarWidth = .15;
h1.Color = [0, 0.4470, 0.7410]; %Blue
h1.LineWidth = 2;

set(gca,'YTick',0:1:(n+1));

set(ax(1),'XLim',[0 n+1])
set(ax(2),'XLim',[0 n+1])
set(ax(2),'YLim',[0 8],'YColor','.2 .7 .3') %Green
set(ax(1),'YLim',[0 8],'YColor','0 0.4470 0.7410')  %Blue
set(ax(1),'YTick',[0:1:5]) %Show Tick's for only MOS values

grid on;
xlabel('Number of Users Trying to Connect to the AP');
ylabel(ax(1),'Average ReQoE');
ylabel(ax(2),'Number of Transcoded Video Stream');

plot(Output(:,1,vr),Output(:,3,vr),'color',[.2 .7 .3],'linewidth',2);

legend([(VRate(vr)), 'Transcoders Used', 'Users Served']);
title(['Transcoder Impact on eQoE, with a limit of ' num2str(MaxTc)]);

i = n;
while i > 0
    if i > 1;
```

```
        text(i-.4, Output(i,4,vr)+.5, [num2str(Output(i,4,vr)), ' Users'],
'VerticalAlignment', 'top', 'FontSize', 7)
    else
        text(i-.4, Output(i,4,vr)+.5, [num2str(Output(i,4,vr)), ' User'],
'VerticalAlignment', 'top', 'FontSize', 7)
    end
    i = i - 1;
end
```

A.5 Jain's Fairness Results

```
%Required Input:
n = 10; %Number of Users
To = 1.1; %TCP/IP Overhead Percentage Multiplier

% User will be the primary Data Matrix organized such that:
    % User(1,1) is the users Number,
    % User(1,2) is the users APRate (access point rate, static)
    % User(1,3) is the users RDRate (Desired Data rate, static)
    % User(1,4) is the users DRate (Requested Data Rate, may change)
    % User(1,5) is the users DRate (Actual Data Rate, may change)
    % User(1,6) is the calculated alpha for User 1
    % User(1,7) is 1 if Tc is needed and 0 if it is not
    % User(1,8) is the actual Video Tx Rate that is supported
    % User(1,9) is the MOS of the Video Stream
UData = 9;
User = zeros(n+1, UData);

%Fixed Values
SRate360p = 1.128; % Service Rate for 360p
SRate480p = 2.628; % Service Rate for 480p
SRate720p = 5.384; % Service Rate for 720p
SRate1080p = 8.384; % Service Rate for 1080p
DDRate360p = round(SRate360p*To,4); %Rounded Desired Data Rate for 360p
DDRate480p = round(SRate480p*To,4); %Rounded Desired Data Rate for 480p
DDRate720p = round(SRate720p*To,4); %Rounded Desired Data Rate for 720p
DDRate1080p = round(SRate1080p*To,4); %Rounded Desired Data Rate for 1080p
DRates = 4; %There are Four Potential Data Rates
DDRate = [DDRate360p; DDRate480p; DDRate720p; DDRate1080p];
%VRate = {'360p'; '480p'; '720p'; '1080p'};
VRate = {'720p'; '1080p'};
VRates = numel(DDRate);
MOS = [3.55; 3.98; 4.31; 4.5];

%Required Input: Maximum Rate for each user based on SNR Table
%(6,9,12,18,24,36,48,54) in Mbps
APRate = [6;9;12;18;24;36;48;54];
```

```
%%% APRates for 10 Users
APRates = [18; 36; 9; 48; 12; 18; 24; 18; 48; 24];

figure;
hold on;
%
%     Break, to run Throughput Optimization
%
%
set(gca,'ColorOrderIndex',1) %Reset Plot Color Order
vr = 3; %Number of Video Rates
while vr > 2
    m = n; %Total Number of Users
    while m > 0
        User = zeros(m+1, UData); %Reset Users Array for each run
        n1 = m;%Current Number of Users for this loop
        while n1 > 0
            User(n1,1)=n1;% Set user Number
            User(n1,2)=APRates(n1);% Set all users to have an APRate
            User(n1,3)=DDRate(vr);% Set all users to have an DDRate
            User(n1,4)=User(n1,3);% Set users initial Requested to DDRate
            User(n1,5)=User(n1,3);% Set users initial Date Rate to DDRate
            n1 = n1-1;
        end

            fn = m;  %Initialize LP Parameter Matrices
            f = zeros(1,fn+1);
            f(1,fn+1) = -1;
            A = zeros(fn+1,fn+1);
            A(1,fn+1) = 1;
            b = ones(1,fn+1);
            lb = zeros((fn+1),1);

            while fn > 0  %Set LP Parameters
                f(1,fn) = -User(fn,2);
                A(fn+1,fn) = User(fn,2);
                A(1,fn) = 1;
                b(1,fn+1) = User(fn,4);
                fn = fn - 1;
            end

            [X,fmax] = linprog(f,A,b,[],[],lb);
            X; %Test Output disabled
            abs(fmax); %Test Output disabled

            wn = m;
            while wn > 0 %Store the Calculated Value for Actual Data Rate in
the User Array
                User(wn,5) = round(User(wn,2)*X(wn,1),4);
                wn = wn - 1;
            end
```

```
            n2 = m;
            while n2 > 0 % Store Final Alpha in User Array
                User(n2,6) = X(n2);
                n2 = n2-1;
            end

        n3 = m;
        while n3 > 0 % Set Transcode Value to 1 if needed
            if round(User(n3,3),4) <= round(User(n3,5),4)
                User(n3,7) = 0;
            elseif round(User(n3,5),4) < DDRate(1)
                User(n3,7) = 0;
            else
                User(n3,7) = 1;
            end
            n3 = n3-1;
        end

        n4 = m;
        while n4 > 0 %Find Tx Video Rate to Calculate Average eQoE for the
Access Point
            if User(n4,5) > User(n4,4)
                if User(n4,5) < DDRate(1)
                    User(n4,8) = 0;
                elseif User(n4,5) < DDRate(2)
                    User(n4,8) = DDRate(1);
                elseif User(n4,5) < DDRate(3)
                    User(n4,8) = DDRate(2);
                elseif User(n4,5) < DDRate(4)
                    User(n4,8) = DDRate(3);
                elseif  User(n4,5) > DDRate(4) && User(n4,3) == DDRate(4)
                    User(n4,8) = DDRate(4);
                elseif  User(n4,5) > DDRate(3) && User(n4,3) == DDRate(3)
                    User(n4,8) = DDRate(3);
                elseif  User(n4,5) > DDRate(2) && User(n4,3) == DDRate(2)
                    User(n4,8) = DDRate(2);
                elseif  User(n4,5) > DDRate(1) && User(n4,3) == DDRate(1)
                    User(n4,8) = DDRate(1);
                end
            %Set Values to next cloest, if Tx rate is to low to Tx
            elseif  User(n4,5) < User(n4,4) && User(n4,5) > DDRate(3)
                User(n4,8) = DDRate(3);
            elseif  User(n4,5) < User(n4,4) && User(n4,5) > DDRate(2)
                User(n4,8) = DDRate(2);
            elseif User(n4,5) < User(n4,4) && User(n4,5) > DDRate(1)
                User(n4,8) = DDRate(1);
            elseif User(n4,5) < DDRate(1)
                User(n4,8) = 0;
            else
                User(n4,8) = User(n4,4);
            end
            n4 = n4 - 1;
```

```
        end

        %Start Jain's Fairness Calculations Here

        j=m;
        jnum = 0;
        jdem = 0;
        while j > 0
            BLPUsers(m,j,vr) = User(j,8); %Store Each Users DataRate value
            jnum = jnum + User(j,8);
            jdem = jdem + (User(j,8)^2);
            j=j-1;
        end
        jnumSQ = jnum^2;
        if jdem > 0 %Fix for divide by 0 error, that is possible
            jains = (jnumSQ/(m*jdem))*100;
        else jains = 0;
        end
        %End Jain's Fairness Calculations

            % Output will be the Results Data Matrix organized such that:
            % Output(1,1) is the Number of Users,
            % Output(1,2) is the Jains Fairness Index
            % Output(1,3) is the total number of Transcoded Users
            % Output(1,4) is the Number of Users being Served (maybe <
connected)

        Output(m,1,vr) = m;
        Output(m,2,vr) = jains;
        Output(m,3,vr) = sum(User(:,7));
        Output(m,4,vr) = sum(User(:,5)>0);

        m = m - 1;
    end

    BLP = plot(Output(:,1,vr),Output(:,2,vr),':d','LineWidth',1.5);

vr = vr - 1;
end

% %
% %BREAK to run Transcoding with Heuristic Algorithm
% %
% %
% %
set(gca,'ColorOrderIndex',1) %Reset Plot Color Order
vr = 3; %Number of Video Rates
while vr > 2
    m = n; %Total Number of Users
    while m > 0
        User = zeros(m+1, UData); %Reset Users Array for each run
```

```
        n1 = m;%Current Number of Users for this loop
        while n1 > 0
            User(n1,1)=n1;% Set user Number
            User(n1,2)=APRates(n1);% Set all users to have an APRate
            User(n1,3)=DDRate(vr);% Set all users to have an DDRate
            User(n1,4)=User(n1,3);% Set users initial Requested to DDRate
            User(n1,5)=User(n1,3);% Set users initial Date Rate to DDRate
            n1 = n1-1;
        end

        % This Loop makes the LP iterative in order to find the highest
        % Video Data Rate that can be transmitted while ensuring any excess
        % bandwidth can be used by other users
        k = m*4;
        while k > 1
            %fprintf('Starting LP Function %d\n');  %Test Output disabled
            fn = m;
            f = zeros(1,fn+1);
            f(1,fn+1) = -1; %0 or -1 results appear the same
            A = zeros(fn+1,fn+1);
            A(1,fn+1) = 1;
            b = ones(1,fn+1);
            lb = zeros((fn+1),1);

            while fn > 0
                f(1,fn) = -User(fn,2);
                A(fn+1,fn) = User(fn,2);
                A(1,fn) = 1;
                b(1,fn+1) = User(fn,4);
                fn = fn - 1;
            end

            [X,fmax] = linprog(f,A,b,[],[],lb);
            X; %Test Output disabled
            abs(fmax); %Test Output disabled

            wn = m;
            while wn > 0 %Store the Calculated Value for Actual Data Rate in
the User Array
                User(wn,5) = round(User(wn,2)*X(wn,1),4);
                wn = wn - 1;
            end

%Start Function for changing Users Requested Data Rate if needed
            Reqmax = User(:,4); %build matrix of column 4 values
            Rqmax = max(Reqmax); %Find Max Value in column 4
            Reqmin = User(:,4); %build matrix of column 4 values
            Rqmin = min(Reqmin(Reqmin~=0)); %Find non-Zero Minimum of Column
4
            APRatemin = User(:,2); %build matrix of column 2 values
            APmin = min(APRatemin(APRatemin~=0)); %Find non-Zero Minimum of
Column 2
```

```
            kk = m+1;
            while kk > 1
                kk = kk - 1;
                if User(kk,4) == Rqmax && User(kk,4) == Rqmin
                    if User(kk,5) < User(kk,4) && User(kk,5) > 0
                        if User(kk,5) < User(kk,4) && User(kk,5) < DDRate(1)
                            User(kk,4) = 0;
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(2)
                            User(kk,4) = DDRate(1);
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(3)
                            User(kk,4) = DDRate(2);
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(4)
                            User(kk,4) = DDRate(3);
                            break
                        end
                    end
                elseif User(kk,4) == Rqmax
                    if User(kk,5) < User(kk,4) && User(kk,5) > 0
                        if User(kk,5) < User(kk,4) && User(kk,5) < DDRate(1)
                            User(kk,4) = 0;
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(2)
                            User(kk,4) = DDRate(1);
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(3)
                            User(kk,4) = DDRate(2);
                            break
                        elseif User(kk,5) < User(kk,4) && User(kk,5) <
DDRate(4)
                            User(kk,4) = DDRate(3);
                            break
                        end
                    end
                end
            end %End Function for changing Users Requested Data Rate if
needed

            n2 = m;
            while n2 > 0 % Store Final Alpha in User Array
                User(n2,6) = X(n2);
                n2 = n2-1;
            end
```

```
           k = k - 1;
        end

        n3 = m;
        while n3 > 0 % Set Transcode Value to 1 if needed
            if round(User(n3,3),4) <= round(User(n3,5),4)
                User(n3,7) = 0;
            elseif round(User(n3,5),4) < DDRate(1)
                User(n3,7) = 0;
            else
                User(n3,7) = 1;
            end
            n3 = n3-1;
        end

        n4 = m;
        while n4 > 0 %Find Tx Video Rate to Calculate Average eQoE for the
Access Point
            if User(n4,5) > User(n4,4)
                if User(n4,5) < DDRate(1)
                    User(n4,8) = 0;
                elseif User(n4,5) < DDRate(2)
                    User(n4,8) = DDRate(1);
                elseif User(n4,5) < DDRate(3)
                    User(n4,8) = DDRate(2);
                elseif User(n4,5) < DDRate(4)
                    User(n4,8) = DDRate(3);
                elseif  User(n4,5) > DDRate(4) && User(n4,3) == DDRate(4)
                    User(n4,8) = DDRate(4);
                elseif  User(n4,5) > DDRate(3) && User(n4,3) == DDRate(3)
                    User(n4,8) = DDRate(3);
                elseif  User(n4,5) > DDRate(2) && User(n4,3) == DDRate(2)
                    User(n4,8) = DDRate(2);
                elseif  User(n4,5) > DDRate(1) && User(n4,3) == DDRate(1)
                    User(n4,8) = DDRate(1);
                end
            %Set Values to next cloest, if Tx rate is to low to Tx
            elseif  User(n4,5) < User(n4,4) && User(n4,5) > DDRate(3)
                User(n4,8) = DDRate(3);
            elseif  User(n4,5) < User(n4,4) && User(n4,5) > DDRate(2)
                User(n4,8) = DDRate(2);
            elseif User(n4,5) < User(n4,4) && User(n4,5) > DDRate(1)
                User(n4,8) = DDRate(1);
            elseif User(n4,5) < DDRate(1)
                User(n4,8) = 0;
            else
                User(n4,8) = User(n4,4);
            end
            n4 = n4 - 1;
        end

        %Start Jain's Fairness Calculations Here
```

```
        j=m;
        jnum =0;
        jdem = 0;
        while j > 0
            ILPUsers(m,j,vr) = User(j,8); %Store Each Users DataRate value
            jnum = jnum + User(j,8);
            jdem = jdem + (User(j,8)^2);
            j=j-1;
        end
        jnumSQ = jnum^2;
        if jdem > 0 %Fix divide by 0 error that is possible
            jains = (jnumSQ/(m*jdem))*100;
        else jains = 0;
        end
        %End Jain's Fairness Calculations

        % Output will be the Results Data Matrix organized such that:
            % Output(1,1) is the Number of Users,
            % Output(1,2) is the Jains Fairness Index
            % Output(1,3) is the total number of Transcoded Users
            % Output(1,4) is the Number of Users being Served (maybe <
connected)

        Output(m,1,vr) = m;
        Output(m,2,vr) = jains;
        Output(m,3,vr) = sum(User(:,7));
        Output(m,4,vr) = sum(User(:,5)>0);

        m = m - 1;
    end

    ILP = plot(Output(:,1,vr),Output(:,2,vr),'LineWidth',1.5);
vr = vr - 1;
end
%
% %
% %Exhaustive Results
% %
% %
% %
%
set(gca,'ColorOrderIndex',1) %Reset Plot Color Order
vr = 3; %Number of Video Rates
while vr > 2
m1 = n; %Total Number of Users
while m1 > 0
    %start new work Exhaustive Search [5 levels, 1080p ... 0]
    %Must be done each time the number of users changes!!!

    %Find number of scenarios
    SN = 0;
```

```
    nm = m1;
    while nm > 3
        SN = SN +((vr+1)^nm)-1;
        nm = nm-1;
    end
    SN = SN + ((vr+1)^3)+3^3+1;
    %end Find Number of Scenarios

    srch = zeros(m1+1, 9, SN);
    SN = 1;
        %Rates are our 4 Video Rates as well as 0 for no transmission
    TxRates = [0; DDRate360p; DDRate480p; DDRate720p; DDRate1080p];
    RateNum = 1:(vr+1); %Matrix of Values [1 2 3 ... (vr+1)]

    %Initialize Fixed Values for all Scenarios
    n1 = m1+1;%Current Number of Users for this loop + 1 line for results
    while n1 > 1 %Reserve 1st line in matrix for results to be calculated
        srch(n1,1,:)=n1-1;% Set user Number (-1 b/c all users shifted down 1
row)
        srch(n1,2,:)=APRates(n1-1);% Set all users to have an APRate
        srch(n1,3,:)=DDRate(vr);% Set all users DRate
        %srch(n1,4,SN)=srch(n2,3,SN);% Sets users TxRates
        n1 = n1 - 1;
    end

    %Initialization of Fixed Values complete
    %Set up Master counter Array, like odometer, has value for each user
    UTxRateNum = ones(1,m1);
    %UTxRateNum = UTxRateNum * RateNum(vr+1); %sets starting rate at the
requested rate and not higher
    UTxRateNum = UTxRateNum * (vr+1); %sets starting rate at the requested
rate and not higher
    % EXAMPLE for 3 users requesting 1080p UTxRateNum starts at = [5 5 5] and
deciments to [0 0 0]

    krs = m1;
    while krs > 0
        K1 = m1;
        while K1 > 0
            n3 = m1+1;%Current Number of Users for this loop + 1 line for
results
            while n3 > 1 %Reserve 1st line in matrix for results to be
calculated
                srch(n3,4,SN) = TxRates(UTxRateNum(n3-1));% Sets TxRates for
User n3 based on Counter Position
                n3 = n3 - 1;
            end

            %
            %
            n2 = m1+1;%Current Number of Users for this loop + 1 line for
results
```

```
            while n2 > 1
                %Find Single Users eQoE based on Desired vs Transmitted Rate
Table
                if srch(n2,4,SN) > 0
                    %Sets MOS to highest value, if expected data rate =
received data rate
                    if srch(n2,4,SN) == srch(n2,3,SN)
                        srch(n2,5,SN) = MOS(4);
                    %Loswers MOS by 1 step
                    elseif srch(n2,3,SN) == DDRate(2) && srch(n2,4,SN) ==
DDRate(1)
                        srch(n2,5,SN) = MOS(3);
                    elseif srch(n2,3,SN) == DDRate(3) && srch(n2,4,SN) ==
DDRate(2)
                        srch(n2,5,SN) = MOS(3);
                    elseif srch(n2,3,SN) == DDRate(4) && srch(n2,4,SN) ==
DDRate(3)
                        srch(n2,5,SN) = MOS(3);
                    %Lowers MOS by 2 steps
                    elseif srch(n2,3,SN) == DDRate(3) && srch(n2,4,SN) ==
DDRate(1)
                        srch(n2,5,SN) = MOS(2);
                    elseif srch(n2,3,SN) == DDRate(4) && srch(n2,4,SN) ==
DDRate(2)
                        srch(n2,5,SN) = MOS(2);
                    %Lowers MOS by 3 steps (e.g., 1080p -> 360p)
                    elseif srch(n2,3,SN) == DDRate(4) && srch(n2,4,SN) ==
DDRate(1)
                        srch(n2,5,SN) = MOS(1);
                    end
                else %MOS is a 1-5 scale, so setting to 1 is the lowest value
                    srch(n2,5,SN) = 1;
                end
                %END Single User QoE Code Block

                % Find Single Users Alpha based on their AP & Tx Rates
                srch(n2,6,SN) = srch(n2,4,SN) / srch(n2,2,SN);
                n2 = n2 - 1;
            end

            %Determine results of this scenario
            %(n2 from this point forward should be == 1)
            srch(n2,7,SN) = sum(srch(:,6,SN));% Finds the sum of Alpha for
all users [May need to round for accuracy]
            if srch(n2,7,SN) > 1 % Determine if Alpha is low enough to be a
valid case
                srch(n2,8,SN) = 0; %Set QoE to 0 since this scenario is not
feasible
            else
                srch(n2,8,SN) = (sum(srch(:,5,SN))/m1);% Finds the mean QoE
of the AP
            end
```

```
                %
                %

            %Counter Iteration Code below
            SN = SN+1; % Last Last Line before starting new scenario
            if sum(UTxRateNum) == m1 %if all positions have decremented then
kounter is done
                break
            elseif UTxRateNum(K1) > 1
                UTxRateNum(K1) = UTxRateNum(K1)-1; %%Same as K1 = K1-1?
                break
            elseif UTxRateNum(K1) == 1 && sum(UTxRateNum) > m1
                UTxRateNum(K1) = vr+1; %Reset counter position to 5 (# of
data rates)
                K1 = K1-1; %Shift to next counter position to the left
            else
                break
            end
        end
        if sum(UTxRateNum) <= m1 %if all positions have decremented then
kounter is done
            break
        end
    end

    MaxeQoE = nonzeros(srch(:,8,:)); %build matrix of column 8 values (Mean
QoE's)
    MxeQoE(m1) = max(MaxeQoE); %Find Max Value for QoE, and store in matrix
based on # of users

[a,b,c]=ind2sub(size(srch), find(abs(srch==(MaxeQoE(3)))));
%srch(a(1),b(1),c(1)); Gives position of optimal eQoE
%a = user number (1 is blank always)
%b = Tx rate of user a
%c = matrix containing all data for the optimal scenario
%srch(:,:,c(1)); %Gives Optimal eQoE matrix of values

User = zeros(m1, UData);

%Start Jain's Fairness Calculations Here %%Updated 12.11.15
        n5 = m1+1;
        while n5 > 1;
            if srch(n5,4,c(1)) == DDRate(4)
                User(n5-1,9) = DDRate(4);
            elseif srch(n5,4,c(1)) == DDRate(3)
                User(n5-1,9) = DDRate(3);
            elseif srch(n5,4,c(1)) == DDRate(2)
                User(n5-1,9) = DDRate(2);
            elseif srch(n5,4,c(1)) == DDRate(1)
                User(n5-1,9) = DDRate(1);
            elseif srch(n5,4,c(1)) == 0
                User(n5-1,9) = 0;
```

```
            end
            n5=n5-1;
        end

    j=m1;
    jnum=0;
    jdem=0;
    while j > 0
        ESUsers(m1,j,vr) = User(j,9); %Store Each Users DataRate value
        jnum = jnum + User(j,9);
        jdem = jdem + (User(j,9)^2);
        j=j-1;
    end
    jnumSQ = jnum^2;
    if jdem > 0 %Fix divide by 0 error that is possible
        jains = (jnumSQ/(m1*jdem))*100;
    else jains = 0;
    end

%End Jain's Fairness Calculations
    JainF(m1)=jains;
    m1 = m1-1;

end


%results 'markersize',20
% if vr == 4
%     plot(MxeQoE,'-*','markersize',10,'LineWidth',1.5);
% else
%     plot(MxeQoE,'-.*','markersize',10,'LineWidth',1.5);
% end
plot(JainF,'-*','markersize',10,'LineWidth',1.5);
vr = vr - 1;
end

flprate = flipud(VRate);
%legend(flprate);
%legend('1080p No Transcoding', '720p No Transcoding', '1080p Single LP
Optimization', '720p Single LP Optimization', '1080p Heuristic Algorithm',
'720p Heuristic Algorithm', 'Exhaustive 1080p','Exhaustive 720p','Nearly All
Users Dissatisfied');
legend('Throughput Opt', 'Heuristic Algorithm', 'Exhaustive Search');
xlim([0 n+1]);
ylim([0 109]);
set(gca,'XTick',0:1:(n+1));
xlabel('Number of Video Streams');
ylabel('Jains Fairness Index');
%title({['eQoE of a Wireless Access Point with Users Connecting at 12 Mbps']
['Solid Line is Transcoded and Optimized'] ['Dashed Line No Transcoding and
No Optimization']});
```

```
%title({['QoE of a Wireless Access Point with Iterative Linear
Optimization'], ['"... Line" = Basic Linear Optimization & "-- Line" = No
Transcoding']});
set(gca, 'box', 'on');
grid on;
hold off;
```