QoE Aware Real-time Multimedia Streaming in Software-Defined Networks

by

Ibtihal Ellawindy

A thesis submitted to the
School of Graduate and Postdoctoral Studies in partial
fulfillment of the requirements for the degree of

Master of Computer Science

Faculty of Business and IT

University of Ontario Institute of Technology

Oshawa, Ontario, Canada

August 2019

THESIS EXAMINATION INFORMATION

Submitted by: **Ibtihal Ellawindy**

Master of Computer Science

Thesis title: QoE Aware Real-Time Multimedia Streaming in Software-Defined Networks

An oral defense of this thesis took place on August 14th, 2019 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee	Faisal Qureshi
Research Supervisor	Shahram Shah Heydari
Examining Committee Member	Khalil El-Khatib
Thesis Examiner	Bill Kapralos

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

ABSTRACT

The exponential increase in bandwidth-sensitive multimedia traffic on the net has given rise to new challenges and services. There is a need to have quality management measures to serve the high needs of efficient transmission and delivery in time-constrained environments over IP networks. Quality of Experience (QoE) is one of the major techniques introduced to achieve the goals of application efficiency and user satisfaction from an end-user perspective. By utilizing crowdsourcing techniques, QoE becomes more cost-efficient and easier to measure. In this paper, I propose a framework that takes real time QoE feedback and forward it to SD-WAN controllers in order to enhance streaming routes based on real-time user quality perceptions. We analyze how QoE can be affected by different streaming protocols and which streaming protocols perform better when dynamic quality changes are introduced. Real-time feedback is compared to predefined dynamic changes to identify if participants able to capture all degradation events or whether not all degradations event combinations are noticeable to the participants. This QoE timestamped feedback will be fed to a SD-WAN controller, in-order to allow end users interaction and the possibility to point out issues in the current service path and to enable the network controllers to take corrective action by rerouting the streamed traffic. Our aim is to demonstrate that real-time QoE feedback can enhance cloud-based services and can adjust services quality based on real-time active participants' interaction.

Keywords: QoE; QoS; SDN; SD-WAN; Crowdsourcing

AUTHOR'S DECLARATION

I hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize the University Of Ontario Institute Of Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize University of Ontario Institute of Technology to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

The research work in this thesis that was performed in compliance with the regulations of UOIT's Research Ethics Board/Animal Care Committee under **REB Certificate number 14780.**

Ibtihal Ellawindy

YOUR NAME

STATEMENT OF CONTRIBUTIONS

I hereby certify that I am the sole author of this thesis. I have used standard referencing practices to acknowledge ideas, research techniques, or other materials that belong to others. Part of the work described in Chapter 4, 5 and 6 has been published as:

I. Ellawindy and S. Heydari. "QoE-Aware Real-Time Multimedia Streaming in SD-WANs." 2019 IEEE NetSoft Conference and Workshops (NetSoft). IEEE, 2019.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor Shahram Heydari for the continuous support of my research and studies, for his patience, motivation and extensive knowledge through the learning process of this master thesis. Furthermore, I would like to thank the rest of my thesis committee: Dr. Faisal Qureshi, Dr. Khalil El-Khatib and Dr. Bill Kapralos. Also, I would like to thank all the participants in my survey, who have willingly shared their precious time during the process. I would like to thank my husband, who has supported me throughout entire process, both by keeping me harmonious and helping me putting pieces together, I wouldn't make it without him. This work was supported by grants from NSERC and Ericsson Canada. We thank Subjectify.us and MSU Graphics & Media Lab for helping us to conduct subjective and objective measurements with MSU Video Quality Measurement Tool.

TABLES OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND SYMBOLS

QoE	Quality of Experience
QoS	Quality of Service
SDN	Software-Defined Networking
SD-WAN	Software-Defined Wide-Area Network
RTP	Real-Time Transport Protocol
RTCP	Real-time Control Protocol
RTSP	Real-Time Streaming Protocol
SCTP	Stream Control Transmission Protocol
SRTP	Secure Real-Time Transport Protocol
SLA	Service Level Agreement
MOS	Mean Opinion Score
IETF	Internet Engineering Task Force
IQMF	In-network QoE Measurement Framework
HAS	HTTP Adaptive Streaming
DASH	Dynamic Adaptive Streaming over HTTP
SQMF	SDN QoE Monitoring Framework
RaaS	Routing as a Service
QoP	Quality of Perception
IF	Influence Factors
HIF	Human Influence Factors
CIF	Content Influence Factors
SIF	System Influence Factors
PSNR	Peak-Signal-to-Noise-Ratio
SSIM	Structural Similarity Index
VQM	Video Quality Measurement
LC	Local Contrast

PESQ	Perceptual Evaluation of Speech Quality
PSQA	Pseudo Subjective Quality Assessment
TUQ	Testing User-perceived QoS
SSQ	Surveying Subjective QoE
MMQ	Modelling Media Quality
ONOS	Open Network Operating System
NBI	North Bound Interface
MTurk	Amazon Mechanical Turk
VM	Virtual Machine
OVS	Open vSwitch
UI	User Interface
HPC	Human Paired Comparison

Chapter 1

1 Introduction

1.1 Background

Over the past few years, real-time multimedia content streaming over the Internet has gained momentum in several industries such as communication, entertainment, interactive-gaming and music industries. The main bulk of Internet traffic nowadays is multimedia content, on-demand video and live video streaming.

Real-time multimedia traffic requires high bandwidth, which should be allocated dynamically according to traffic priority. With the emergence of high functioning mobile devices, network service providers are in a continuous effort to support a wider range of applications and quality of service (QoS) requirements with highly utilizing network capacities [1]. Most of the real-time streaming content originates from applications based on Real-time Transport Protocol [2]. With mobility, delays and losses in multimedia streaming become more common and achieving the QoS [3] becomes more challenging.

A multimedia service can be described by the content it supplies, the transmission means used to supply this content and services to enable content exchange between different parties. Multimedia service designs are considered the end user's requirements, and include: the cost of the service, ease of accessibility to the content, content quality and multimedia desirability [4]. Furthermore, Internet usage has shifted towards content-centric rather than host-centric. End-user expectations are constantly elevating, and the multimedia content providers are becoming more aware of the importance of service quality. We must consider network conditions and QoS parameters such as delay, jitter, bandwidth, and packet loss on multimedia streaming quality. QoS is mostly concerned with network

conditions and the related service level agreement parameters. On the other hand, Quality of Experience (QoE) is a more subjective and user-centric assessment technique that is concerned with end-user perception of the service and as such, QoE-based assessments are quickly becoming the guidelines for managing the end-user's quality expectations.

In terms of transport services, Real-time Transport Protocol (RTP), along with Real-Time Control Protocol (RTCP), or Real-Time Streaming Protocol (RTSP), provide a reliable foundation for real-time services [5]. The emergence of Software-Defined Networking (SDN) has promised better control and management of the end-to-end service quality in the networks [6]. Leveraging SDN advantages of dynamic programmability, centrally controlled, cost-effectiveness, and greater adaptability to networking environment changes, makes the Software-Defined Wide-Area Network (SD-WAN) a suitable architecture to control QoE for multimedia streaming applications and services. With real time user feedback during live video streaming, it will provide a better QoE by trying to enhance streaming routes using SD-WAN controllers.

1.2 Related Work

Several prior works have examined the possibility of managing QoS and QoE using the advantages of SDN architectures. For instance in [7, 8], authors focused on how QoE could be managed efficiently over cloud services and the challenges facing QoE management in cloud applications, especially the quality of multimedia streaming. The goal of QoE management is to provide high quality services to end users on the cloud while taking into consideration relaying costs behind such quality.

Network and service performance indicators for multimedia applications have been discussed in [4]. The most important performance indicators include the following:

- One-way end-to-end delay (including network, propagation, and equipment) for video or audio should be within 100 to 150 ms.
- Mean-opinion-score (MOS) – a QoE rating system on a scale from 1 to 5 level - for audio should be within 4.0 and 5.0. MOS level for video should be within 3.5 and 5.0. (more about MOS in Chapter 2)
- End-to-end delay jitter (packet delay variance) must be short, normally less than 250 ms.
- Synchronization of intermedia and intramedia should be maintained using suitable algorithms. To maintain intermedia synchronization, differential delay between audio and video transmission should be within −20 ms to +40 ms.

In order to tackle the requirements of multimedia over IP, multimedia services should have the ability to classify traffic, prioritize different applications and make the necessary reservations accordingly. The Internet Engineering Task Force (IETF) has developed an Integrated Service framework that consists of real-time and best effort services. RTP, along with RTCP, and RTSP, to provide a reliable foundation for real-time services.

Different parameters that should be taken into consideration while designing a QoE framework for multimedia services are described in [9] as follows:

- Video quality at the provider source.
- How the content is delivered over the network and QoS SLA.
- End user perception, expectations and ambiance.

More recently, crowdsourcing techniques have been considered to collect users' QoE. In [10], authors designed a crowdsourcing framework that overcomes some of the disadvantages of the MOS technique, namely: 1) difficulty and inconsistency for participants to map their rating due to 5-point scaling, 2) Rating scale heterogeneity, and 3) lack of cheat detection mechanism. By introducing the ability to have QoE measured in a real-life environment using crowdsourcing rather than a controlled environment in a laboratory, the new method provides comparable result consistency similar to the MOS methodology. Another approach, the OneClick framework [11], captures multiple users' perception in a simple one-click procedure where experiments are held to gather the users' feedback and then collected data are processed to calculate the accumulative QoE of all users. Programmable QoE-SDN APP discussed in [12], aims to improve QoE for customers using video services by minimizing stalling events occurrences focusing on HTTP Adaptive Streaming (HAS) applications by utilizing forecast and rating estimations provided by mobile network operators.

Some research has focused on the specific use of SDN controllers and the importance of the selection of SDN controllers in designing network models. Recently, research conducted by [13], focuses on using intent-based programming using the Open Network Operating System (ONOS) controller [14] to allow more dynamic monitoring and rerouting services by using intents. Intent Framework [15], enables applications to provide network requests in the form of a policy and not as a mechanism. Intents provide a high level abstraction where programmers only focus on the task that should be accomplished, rather than how these tasks will be translated into a low-level rules and how these rules can be installed into the network devices, by only expressing required "intentions" via high-level

policies. The research aims to enhance Intent Framework to compile more than one intent at the same time and to re-optimize paths based on flow statistics.

Leveraging SDN in routing, a recently published paper [16] on how routing services can be customized for applications. It proposed a new open framework called Routing as a Service (RaaS) by reusing virtualized network functions. Upon selecting appropriate functions, the authors build customized routing services on the routing paths for different applications.

Using QoS over SDN in [17], the authors designed an approach to introduce QoS into IP multicasting using SDN in order to have a proper flexible control management of the network environment. OpenFlow protocol was adopted to allow a controller to monitor IP multicasting statistics for each flow to provide end-to-end QoS. They implemented a learning algorithm to allocate intelligently the required network resources without dropping low priority packets that have performance impacts. It thus demonstrated that SDN could be used for network quality management. Next sections will discuss different contributions and QoE Models.

1.2.1 In-network QoE Models

Farshad, et al [18], they proposed an In-network QoE Measurement Framework (IQMF), where user feedback is not considered as an input parameter, and the streams are being monitored within the network. This method is based on a QoE architecture and depends strongly on user participation. Two QoE metrics are adopted by IQMF for measuring experience concerned with 1) quality of video, and 2) switching impact over HAS streams. Through an API, Figure 1, IQMF offers the above measurements for QoE as a service. This service could be provided to a content distributor or a network provider.

By leveraging SDN, it allows the control plane interaction with IQMF framework which enabled it to analyse and measure participant's QoE with more flexibility. SDN enabled IQMF to do dynamic traffic management and ability to deploy more measurements agents to make it more scalable. IQMF interacts with OpenFlow controller that keeps the forwarding behavior of the network to ensure that all necessary information about flow duplications is provided to allow better monitoring of QoE.

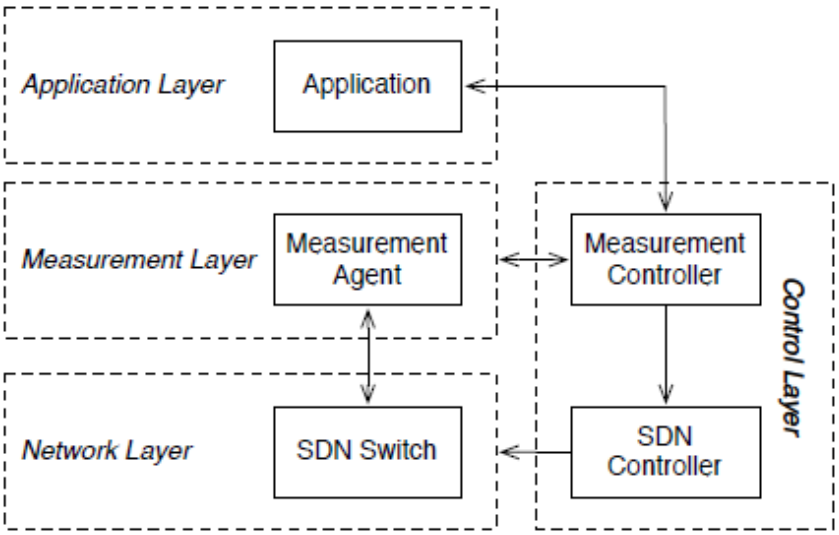


Figure 1: In-network QoE Measurement Framework [18].

The QoE measurement framework works by filtering through HTTP packets. It then identifies HTTP GET requests followed by examination of the GET requests for the identification of the manifest files available, such as: Media Presentation Description file (MPD). The MPD parser extracts information from the MPD file - different representations that includes references to different resolutions, quality multitude as well playback codecs. The measurement engine then merges parsed information with supplementary details from the HTTP packet filter in order to monitor the behavior of the user whilst playback.

Another model described in [19], aims to enhance capabilities of Dynamic Adaptive Streaming over HTTP (DASH) – a standard for multimedia streaming that changes content quality presented automatically in accordance with network conditions - to take into consideration the user's perceived QoE. It integrates the user's ongoing perception while the content represented dynamically changes. Such enhancements will provide more efficient QoE measurements and increases positive feedback of users. This model allows automated estimated MOS measurements from the below QoS parameters, Table 1.

There are three main metrics used in this model:

1. Buffer underflow/overflow: to prevent freezing images and losing packets, buffer threshold has been identified. TCP has been used as well for reliable transmission.
2. Switching quality frequencies and amplitude: the frequency of quality switches of the represented content is one of factors affecting QoE.
3. QoS media parameters: which are the parameters concerned with the content of media, Table 1.

Table 1. Affected QoS video parameters on QoE

Parameter	Description	Symbol
Video Bitrate	Indicates the video bit rate Scale: actual bits per second (bps) Range: 0 to infinity	BR
Video Frame rate	Indicates the video frame rate Scale: actual frames per second (fps) Range: 0 to 60 fps	FR
Video Quantization Parameter	Video quantization parameter Scale: average value of AVC QP Range: 0 to 51	QP

It was found during experiments that it takes seconds to measure presentation intervals that are affected by media parameters. Where, “the representation quality switch rate, required a recursive approach where the eMOS is calculated based on previous eMOS variations in order to take into account the entity of the quality switch in addition to the rate” [19]. This model has huge potential is enhancing DASH logic of adaptation capabilities in selecting the best level of a video quality by integrating the capabilities of QoE monitoring.

The OpenE2EQoS model discussed in [17] aims to introduce QoS into IP multicasting using SDN, to have a proper flexible control management of the network environment. In this approach, OpenFlow protocol was adopted to allow the controller to monitor IP multicasting statistics for each flow to provide end-to-end QoS. The system makes use of the additive-decrease/multiplicative-increase (AIMD) algorithm to enhance adaptive learning of preserved bandwidth for efficient link utilization over time to improve QoS. N-dimensional statistically algorithm is used as well in that approach, by redirecting low priority traffic packets from overly crowded links rather than rerouting multimedia packets.

Kaleidoscope [20] is a real-time content delivery architecture based on Software Defined Infrastructure. It emphasizes network virtualization, SDN-based broadcasting and provisioning of resources to achieve better performance and efficient resources. The main feature of Kaleidoscope that it dynamically changes configuration and resource allocation during runtime based on perceived demands. The system architecture has three main tiers: Producer, Cloud and User tiers. In Producer tier, producers upload and register their content. In Cloud tier, these contents are processed by the servers and NFV modules and

then distributed among end users using multicasting techniques. In User tier, by using a unicast technique the content is then distributed among the end users. Leveraging SDNs, allowed dynamic configuration for unicasting and broadcasting and open interface between resources.

In [21], a method to predict QoE by using machine learning algorithms leveraging SDN is proposed. It is an architecture that uses the previously collected measured MOS from end-users during different network changing parameters along with objective measures and fed to machine learning algorithms to predict MOS values. SDN QoE Monitoring Framework (SQMF) [22], a monitoring application that aims to preserve QoE for both video and VoIP applications in real-time regardless of unexpected network issues, by continuously monitoring network parameters and using QoE estimation models. In [23], a new QoE-Aware management architecture over SDN, which was able to predict MOS by mapping the QoS different parameters in QoE, and it was designed to function as self-determining solution to edit underlying network resources infrastructure with the ability to avoid QoE degradation, optimize resources use and improves QoS performance.

1.2.2 Crowdsourcing QoE Models

OneClick Framework, [11], captures a user's perception in a simple one click procedure. Whenever a user is not satisfied with the quality of the content viewed, they can click on a button that informs the system of their dissatisfaction. In contrast to the MOS technique, the user doesn't have to decide between different grading scales and what best suits their perception. OneClick is a real-time framework which means the clickable button is available along the whole viewing experience. The user can record their dissatisfaction

multiple times along the process where each click is time captured. This framework is based on PESQ and VQM, where both techniques are objective measurements.

The key advantages of OneClick Framework:

1. Initiative: Participants are not required to decide about the perceived quality, they just record their dissatisfaction through the one click button
2. Lightweight: Framework doesn't require any specific deployments and it is not expensive to roll out.
3. Efficient: The user can record their dissatisfaction several times along the test, to understand participant perception well enough.
4. Time-aware: Participant can record their dissatisfaction several times along the process where each click is time captured, thus providing an indication of how perception changes over time.
5. Independent: OneClick can be used in conjunction with several applications and not limited to a specific one.

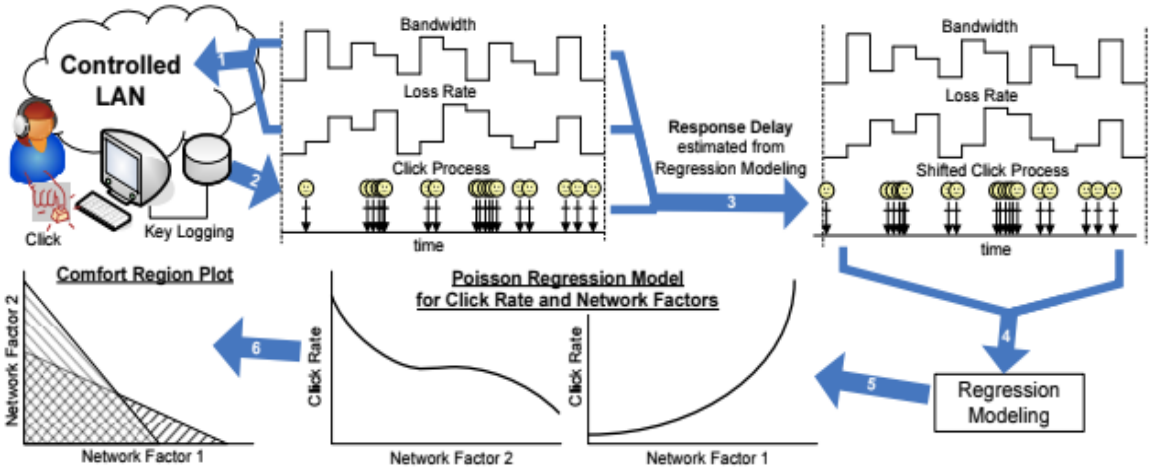


Figure 2: The flow of a complete OneClick assessment procedure [9].

OneClick has two main steps: 1) experiments are held to gather user's perception feedback on different network conditions, and 2) collected data are then processed to identify the QoE of users. Figure 2, shows the whole OneClick process assessment technique as following: 1) preparing test materials (optional), 2) asking subjects to do experiments, 3) inferring average response delays, 4) modeling the relationship between network factors and click rates, 5) predicting the click rate given each network factor, and 6) summarizing an application's QoE over various network conditions by comfort region.

The main goals of the crowdsourcing framework discussed in [10] is to overcome some of the disadvantages of MOS technique by utilizing paired comparison technique for two stimuli. As well the ability to have QoE measured in a real-life environment using crowdsourcing rather than a controlled environment in a laboratory. Four case studies were conducted using audio and video content to evaluate the effectiveness of the proposed framework.

The key features of using this framework for QoE evaluation is:

1. It could be generalized for different types of multimedia content with no adjustments.
2. Pair comparison rating technique for simplicity and less difficulty on users rather than MOS technique
3. Results from compared judgements can be evaluated by probability models
4. Cheat identification supported which provides trusted quality framework measures by utilizing the transitivity property, which validates the judgments consistency and remove untrusted inputs.

5. Budget efficient for reliable QoE measurements and less participant effort in comparison to MOS

6. Using reward and punishment, were users are given appropriate incentive to provide honest feedback thus trusted quality measures.

This framework overall is a very promising evaluation technique to measure QoE. However the authors also mentioned what has been done is not a QoE evaluation but rather a quality of perception (QoP) [10]. “QoP reflects a user’s detectability of a change in quality or the acceptability of a quality level”.

1.3 Research Gap

Major research focused on quality of video streaming, video quality, underlying network quality and QoS, metrics for conducting QoE, and how to acquire participant. Though extensive research in QoE feedback analysis and measurement, acquiring and gathering feedback and work in real-time multimedia streaming, it was found that there was no research published to allow any real-time QoE feedback during multimedia streaming sessions that instantly enhances the real-time stream quality during a live session. QoE is usually measured and assessed after completion of streaming session/or paired comparisons and results are fed back to frameworks to enhance user QoE. However, no methods provided insights on how QoE feedback can be gathered in real-time, and how it should be communicated to the network controllers to allow dynamic network changes to enhance QoE and streaming quality while participants are observing the session.

1.4 Contributions

This research aims to enhance the quality of streamed content in real-time based on current viewer feedback. We propose a framework based on a combination of real-time QoE measurement application and QoS quality parameters which can accommodate a variety of streaming protocols. This framework emphasizes the dependability between QoE and QoS and how the overall user QoE perspective can be affected. We study how dynamic changes in the network will affect the performance of different streaming protocols, and how each streaming protocol adjusts to the network changes and consequently the perceived QoE of the streaming content. Main protocols used in our model are RTP, RTP over TCP, SCTP, and UDP, since they are major streaming protocols currently used.

The proposed model is based on real-time QoE feedback of quality degradation during live video streaming over a cloud-based SD-WAN environment. A QoE-rating application is deployed on the users' end to return user feedback during real-time streaming, where participants click on a button when they feel the quality has been degraded. Accordingly, the end user application (which can be deployed as a plugin on web browsers or multimedia players) will send the feedback to the SDN-WAN controller to inform the controller about potential problems in the video streaming. This QoE feedback enables the SD-WAN controller to detect problems in the service path and to take corrective action by making changes to the virtual topology of the content delivery network, reassigning the users, or rerouting the traffic.

In March 2019, Google announced the plan to have cloud streaming services for gaming (Google Stadia). Microsoft project xCloud and Sony PlayStation now are following the same steps. Gaming requires a very low latency which is very critical for as

positive user experience and must support real-time interaction. The currently available streaming infrastructure is sufficient for one-way video or a live steaming, where you can wait for loading and buffering as you go. However, gaming requires to move packets only when a user performs specific actions hence the importance of updating our current technology. Google demo for Stadia showed they will rely on their vast data centers around the work to ensure quality streaming. There is no mechanism that allows users provide their feedback interactively. By applying our model for real-time feedback, it would enable gamers to provide real-time inputs and possible network and quality optimization can be implemented on-spot.

The main feature of our proposed framework is the ability to allow the end users to identify changes in their perceived QoE of streamed videos with dynamically changing network conditions in real-time by providing instant feedback to the SDN controller to point issues in the service. The questions we try to answer are the following: Can QoE feedback be applied in real-time? Will the participants be able to identify quality changes along with network changes? Does video content affect participants' quality-rating decisions despite the noticeable quality degradation? When comparing objective analysis against human perceived quality, would the results be consistent? Which protocols are best for unstable / changing networks and which are the worst? Is there consistency in protocol performances when video content changes, or when different events and scenarios occur?

A paper proposing the above model was accepted in the IEEE Conference on Network Softwarization (IEEENetSoft19) under IEEE PVE-SDN 2019 workshop and is to be published in the conference proceedings and IEEE Explore digital library.

1.5 Thesis Outline

The rest of this thesis is organized as follows: In Chapter 2, I present QoE basics and overview of the SD-WAN network and QoE measurement models. In Chapter 3, I present current multimedia transport protocols. In Chapter 4, I present my proposed QoE-Aware real-time models. In Chapter 5, implementation is detailed. In Chapter 6, I will discuss the proposed model performance results and analysis. And the thesis is finally concluded with Chapter 7.

Chapter 2

2 QoE Basics and Models

2.1 QoE Definition

"Quality of Experience (QoE) is the degree of delight or annoyance of the user of an application or service. It results from the fulfillment of his or her expectations with respect to the utility and / or enjoyment of the application or service in the light of the user's personality and current state" [24].

QoE is not a predefined criterion that can be generally used with all types of applications. QoE is context oriented and defined by the domain where the application belongs. Each application should have a set of identified QoE parameters that are important for its success. Figure 3 shows how QoE should be present in the application as well as service provider eco system [24]. QoE could be applied in a variety of application domains, such as multimedia streaming, telecommunications, social media, medical and educations applications.

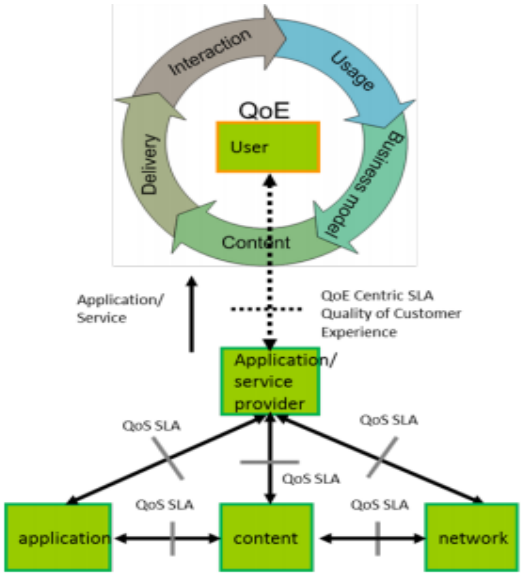


Figure 3: QoE introduction in service provider/ application eco system[24].

QoE and QoS are tightly coupled. Many QoS parameters can affect the QoE perceived by the users. QoS and network conditions should be considered during the evaluation of QoE of a specific application. Application performance should be planned for QoE models based on user experience, not just on QoS parameters. This is introduced by having SLAs (Service level agreement) in which QoE requirements are defined by the service.

2.2 QoE Influence Factors

There are several factors that could affect QoE. Some of these factors can be identified, measured and impacts are known in advance. However, other factors cannot be predicted or quantified, it may be hard to forecast their impacts, and they are not totally dependent on certain situations and factors such as environment, content, demography, etc. Influence factors of QoE as mentioned in the Qualinet White paper [24]is:

The "Influence Factor (IF) is any characteristic of a user, system, service, application, or context whose actual state or setting may have influence on the Quality of Experience for the user".

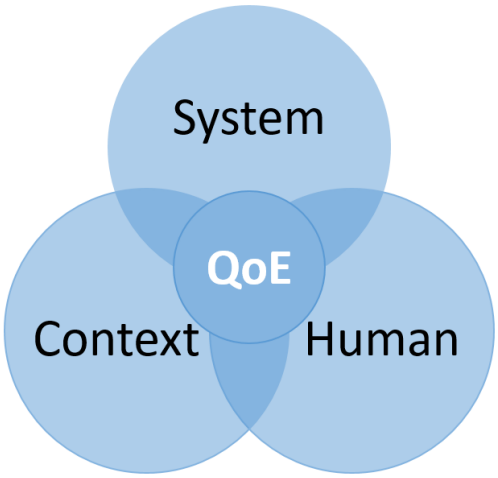


Figure 4: QoE Influence Factors.

There are three main factors that influence QoE, as shown in Figure 4. These influence factors are human, system, and context, which could also be overlapping. We will discuss each factor in the following.

2.2.1 Human Influence Factors (HIF)

This group includes factors based on user human characteristics, such as: background, emotional state, physical and as well mental constitutions. HIF are very challenging and could be considered as extremely complex factors. HIF are intangible and subjective to states and backgrounds. HIF cannot be used independently to determine QoE due to its dynamic factors [24, 25]. There are two main HIF characteristics:

1. Low-level processing such as: gender, attention, age, moods, visual sharpness, auditory alertness, motivation, etc.
2. High-level processing, such as: judgments, previous knowledge, behavior and personal perception, education and social background.

2.2.2 System Influence Factors (SIF)

Used to describe the technical quality characteristics of an application or a service. SIF are related to the technical factors that affect the development of a specific application or a service. SIF are classified within the four factors below:

1. Content related: type of the content (video, audio or date) and quality of the content.
2. Media related: factors related to media configuration such as: frame rate, encoding, sampling rate, etc.

3. Network related: such as jitter, delay, bandwidth and loss during data transmission, it is highly related to QoS parameters of a network

4. Device related: end devices connected in the same communication link over network.

2.2.3 Context Influence Factors (CIF)

Used to describe and determine factors affected by the surrounding environment of a user. CIF are divided into contexts below [25]:

1. Physical, such as indoor or outdoor location, personal space or workplace.

2. Social: personal interactions within surroundings of an individual while the observation experience is conducted

3. Economic, such as: value and the make of an application/service

4. Task: could be defined in terms of the nature of the experience, whether the user is in a multitasking situation or faced any kind of interruptions and the nature of the task itself.

5. Technical and information context: defining the relationship between interested system and other relevant services or systems: devices, applications, networks, etc.

2.3 QoE Measurements

In order to measure QoE, there are three main methodologies [9]:

- No-Reference Model: No knowledge of streaming source file. QoE is predicted by real-time monitoring of QoS parameters,

- Reduced Reference Model: knowledge of the streaming source file is limited and QoE is predicted by combining this knowledge with the real-time measurements.
- Full-Reference Model: full access to the Referenced video combined with real-time measurements.

There are two main approaches for QoE assessments: subjective and objective methods. Subjective techniques are based on user interaction and feedback. The most common subjective approach is Mean Opinion Score (MOS) [10, 26-28]. MOS is based on a rating system on a scale from 1 to 5. Where 1 stands for 'Bad', 2 stands for 'Poor', 3 stands for 'Fair', 4 stands for 'Good' and 5 stands for 'Excellent'. 3.5 is the minimum acceptable threshold for a video MOS [9]. MOS scaling still may give room for inaccurate representation of user's perception [10], due to the non-similarity of the scales interpretation by the participants.

While subjective approaches are considered more accurate in reflecting the user perception, they are expensive to roll out because such QoE assessments require a large scale of participants in order to obtain reliable results. They are also time consuming, as traditional QoE experiments are conducted in a controlled lab-environment, making it difficult to collect sufficient results from experiments in a limited time-frame [10]. To overcome these constraints, QoE crowdsourcing techniques have been proposed, where obtaining subjective results becomes relatively cheaper and more efficient than in a traditional lab-controlled environment, because it takes advantage of employing a diverse group of online participants. Crowdsourcing allows subjective measures for paired comparisons (a method that compares sets in pairs to judge or rate the preferred one, or

determine if it has the desired result) as well as MOS-based rating comparisons, with the flexibility to choose participants' demographics too.

Despite their scalability, crowdsourcing experiments lack supervision since it is not a controlled environment, which makes some results not fully trustable. Researchers should be able to identify trusted and untrusted participants to make sure accuracy of obtained results. This can be achieved by designing the crowdsourcing campaigns based on certain best practices. These best practices are majorly concerned with technical implementation aspects of the experiment, campaign and test design, and a thorough statistical analysis of results [29]. Campaigns should be simple enough for participants to understand how the experiment is designed and what is required to complete it.

As opposed to subjective assessment techniques, objective QoE assessment techniques are mostly based on network analysis and technical comparisons that aim to produce a quantitative assessment. Quantitative QoE assessment is tightly related to the QoS of an application or service. Peak-Signal-to-Noise-Ratio (PSNR) is considered an objective approach for measuring quality, as it assesses how much similarity exists between two different video images. It is widely used in video streaming assessment, where the higher PSNR value the higher similarity between the original and received video images. One drawback of the PSNR method is, it does not take into consideration how human perception works, it's only based on image similarity.

Structural Similarity Index (SSIM) is another measurement approach based on perceived estimation of video structural distortion. SSIM is based on the comparison of three measures, luminance (l) , contrast (c), and structure (s). The overall index is a multiplicative combination of the three comparative measures as shown in (1), where μ_x,

μ_y, σ_x, σ_y, and σ_{xy} are the local means, standard deviations, and cross-covariance for images x, y.

$$\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = [l(\mathbf{x},\mathbf{y})]^{\alpha}[c(\mathbf{x},\mathbf{y})]^{\beta}[s(\mathbf{x},\mathbf{y})]^{\gamma},$$

$$l(\mathbf{x},\mathbf{y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2\mu_y^2 + C_1},$$

$$c(\mathbf{x},\mathbf{y}) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2\sigma_y^2 + C_2},$$

$$s(\mathbf{x},\mathbf{y}) = \frac{2\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}。 \quad (1)$$

SSIM addresses the shortcoming of PSNR mentioned above by combining the contrast, luminance and structure similarity factors. SSIM compares the correlation between perceived video images and original video images, hence considered as a Full-Reference model. The higher the ratio, the higher the signal similarity.

Another objective approach is Video Quality Measurement (VQM) [30], a metric to measure the perception of video quality that closely resembles human perception. The VQM metric is designated to be a general- purpose quality model for a range of video systems with different resolutions, frame rates, coding techniques and bit rates. VQM takes into consideration noise, blurring, and block and color distortions. VQM provides an output value of zero if no impairment is perceived and the output increases with a rising level of impairment.

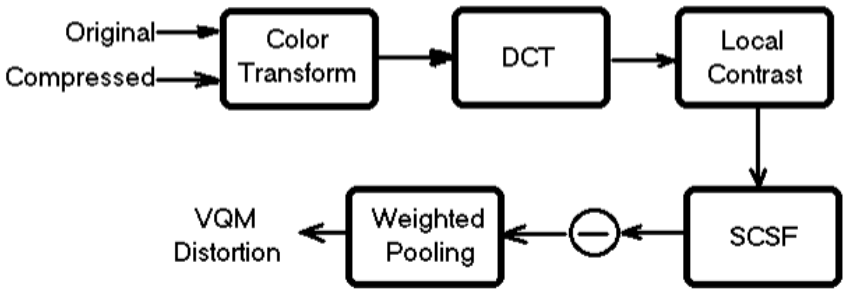


Figure 5: VQM process overview.

Figure 5 gives an overview of the VQM process [30]:

1. Color transform: both MPEG and H.263 use the YUV color space, so it can use the raw data directly.

2. DCT transform: this step separates incoming images into different spatial frequency components.

3. Converts each DCT coefficients to local contrast (LC) using following equation (2):

$$LC(i,j) = DCT(i,j) \times Power(DC/1024, 0.65) / DC \qquad (2)$$

DC is the DC component of each block. For 8-bit image, 1024 is mean DCT value. 0.65 is the best parameter for fitting psychophysics data. After this step, most values lie between [-1, 1].

4. Converts LC to just-noticeable differences (jnds)

5. Weighted pooling of mean and maximum distortion using below formula:

$$Mean\_Dis\ t = 1000 \times mean(mean(abs(diff))). \qquad (3)$$

$$Max\_dist = 1000 \times maximum(maximum(abs(diff))) \qquad (4)$$

$$VQM = (Mean\_dist + 0.005 \times Max\_dist). \qquad (5)$$

Maximum distortion weight parameter 0.005 is chosen based on several primitive psychophysics experiments. Parameter 1000 is the standardization ratio.

Perceptual Evaluation of Speech Quality (PESQ) is another objective measurement used widely in telecommunications and speech codecs. It is used to assess the quality of end-to-end speech. If used in correlation to various network conditions, it can be a good model to predict subjective quality [11].

There is a hybrid approach that combines both objective and subjective measures. The advantage of this approach is that it could be implemented in real-time and takes user QoE into consideration. One of these hybrid approaches is Pseudo Subjective Quality Assessment (PSQA). The advantage of this approach is it could be implemented in real-time as well as take user QoE [27].

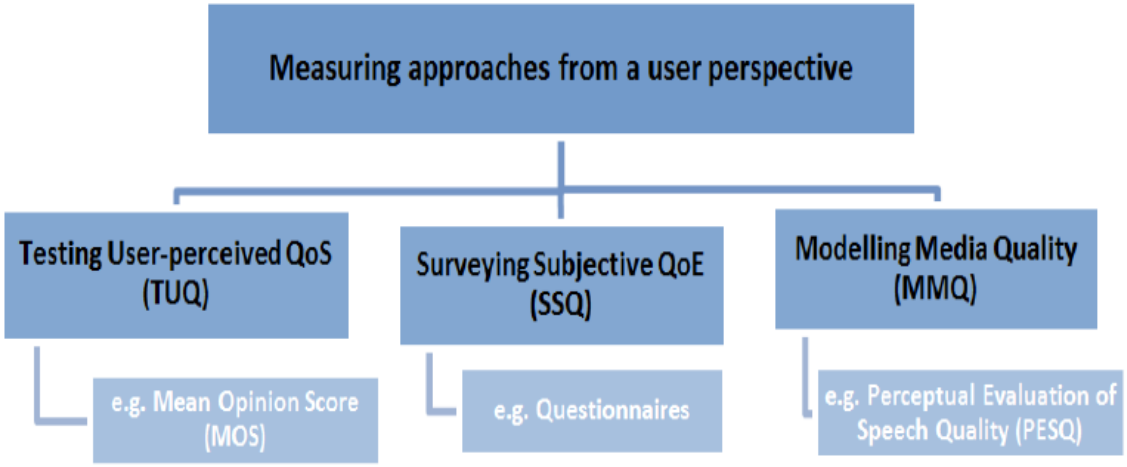


Figure 6: User perspective quality measuring approaches [28].

Figure 6, shows the diagram on how quality is measured based on perspective of the user based on Testing User-perceived QoS (TUQ), Surveying Subjective QoE (SSQ), or Modelling Media Quality (MMQ) [28].

2.4 QoE and Crowdsourcing

Crowdsourcing is the activity to outsource a specific task/job to a crowd [31]. A more formal definition [32]:

"Simply defined, crowdsourcing represents the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call. This can take the form of peer-production (when the job is performed collaboratively) but is also often undertaken by sole individuals. The crucial prerequisite is the use of the open call format and the large network of potential laborers."

Crowdsourcing consists of two words: crowd, which indicates the people who are willing to participate in specific initiative; and sourcing, which involves procurement practices in order to find, assess, and engage different suppliers providing services, products or goods. In Crowdsourcing, tasks are submitted in a form of an open invitation to a wide variety of anonymous crowd workers. These tasks don't require a long-term contract and can be completed in minutes or even hours. Most of the tasks are considered repetitive e.g. video or image comparison, and usually grouped into campaigns. Such campaigns can be submitted via mediator platforms that maintain crowd and effectively manage created campaigns.

Crowdsourcing platforms are the mediators that offer web services [33]. There are three types of crowdsourcing platforms: aggregators, specialized, and crowd providers as shown in Figure 7. Aggregator platforms are considered a high-level platform since they don't own their crowd, however, they hire a crowd from other channels such as crowd providers or specialized platforms. They are mostly focused on specific tasks with a set of

predefined mechanisms to ensure quality. One of the main advantages of these types of platforms is the abstraction from issues related to hiring workers or controlling quality. However, among the disadvantages is the workers available might be limited in diversity as they could be already pre-filtered. Examples for aggregator platforms are CrowdFlower and Crowdsource. Specialized crowdsourcing platforms focus on a specific type of workers or tasks subsets. Microtask is a specialized crowdsourcing platform.

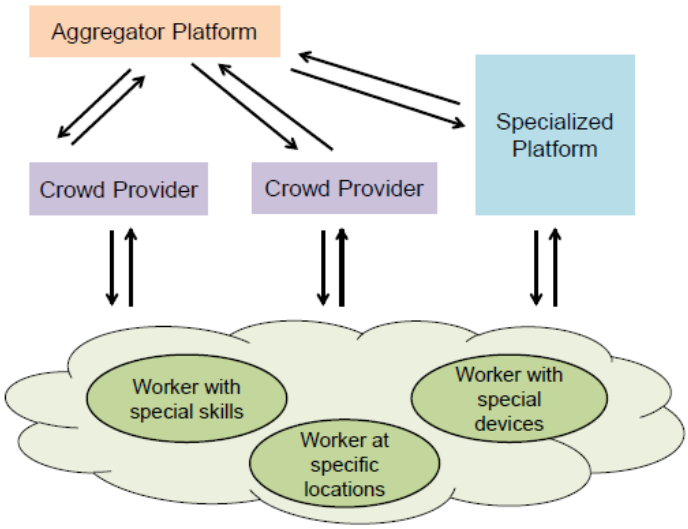


Figure 7: Types of crowdsourcing platforms and interactions [29].

The last type of platforms are called crowd providers, such as: Amazon Mechanical Turk (MTurk) [34] and Microworker [35]. MTurk is widely used by research and commercial communities. Crowd provider platforms are commercial platforms that usually maintain their own crowd and are based on self-service. The crowd accesses the services via a web interface or API interactions. The crowd is paid through the platform upon the employer confirming the successful completion of a task. These types of platforms are flexible and provide a wide range of tasks and campaigns that fits the employers' needs where no restrictions are applied on the type of task. Due to the diversity of the crowd

adopted by this platforms, there is a limited mechanism for quality assurance and it's the employer's responsibility to integrate more advanced mechanisms to ensure quality [29, 33]. There are certain characteristics for any crowdsourcing activity [36]:

1. The crowd is defined clearly (demography, sex, age, race, etc.).
2. A clear goal is set for the task required.
3. Crowd receives compensation.
4. Identification of crowdsource platform.
5. Crowdsource compensation.
6. It's an activity where crowd participates online through the internet.

QoE assessments require a large number of participants in order to obtain reliable results. This makes it more expensive to roll out. The assessments are also time consuming, as traditional QoE experiments are conducted in a controlled lab-environment, making it difficult to collect enough results from experiments in a limited timeframe [8]. To overcome these constraints, QoE crowdsourcing techniques have been proposed, where obtaining subjective results becomes relatively cheaper and more efficient than traditional ways, because it takes advantage of employing a diverse group of online participants. Crowdsourcing allows subjective measures based on both video-pair comparisons and MOS-based rating comparisons, with the flexibility to choose participants' demographics too.

Despite their scalability, crowdsourcing experiments lack supervision, making some results not fully trustable. Researchers should be able to identify trusted and untrusted participants.

This can be achieved by designing the crowdsourcing campaigns based on certain best practices. These best practices are majorly concerned with technical implementation aspects of the experiment, campaign and test design and thorough statistical analysis of results [16]. Campaigns should be simple enough for participants to understand how the experiment is designed and what is required to complete it. To ensure rating reliability for video streamed QoE crowd testing, task design should enforce cheating prevention, filtering the crowd by performing short and simple reliable tests before hiring a crowd for the QoE. After the QoE task is completed, a verifications technique "golden question" should be implemented to ensure that the hired crowd successfully conducted the QoE task, e.g.: content questions "what animals did you see from below list?", golden questions such as "were there any stops in video during streaming session? (No, Yes)", meanwhile there were no stop events in the video streaming session. Table 2 shows the differences between QoE testing in the lab and crowdsourcing. There are some crowdsourcing platforms that are specialized in conducting crowdsourced testing QoE evaluation.

Subjectify.us [37] is a web platform that specialized in conducting crowdsourcing subjective comparisons for videos, sound processing methods, and images, were paid workers can rate the higher QoE in a paired comparison of two stimuli. The platform offers reliability measures as well and provides results with different computational methods, converting the pairwise comparison data to scores in detailed reports.

Table 2. Differences in QoE studies in the laboratory and crowdsourcing [33].

Differences	Crowdsourcing	Laboratory
Conceptual		
Test duration	5–15 min	30–60 min
Outlier detection	Common statistical methods	Common statistical methods
	often not applicable	
Training	Can not be ensured	Training with feedback
Technical		
Environment and equipment	Real-life environment	Standardised and artificial
Stimuli	Mostly limited to	Multi-sensory
	web-supported audio and video	
Design	Limited by internet access,	No limitations
	web-browsers and devices	
Motivational and subjects		
Demography	Global and diverse	Local and limited
Incentives	Mostly financial	Financial and altruistic
Cost	Cheap	Expensive

Chapter 3

3 Multimedia Transport Protocols

3.1 Real Time Transport Protocol

One of the primary multimedia transport protocols is Real Time Transport Protocol (RTP). RTP was developed by the Audio-Video Transport Working Group of the Internet Engineering Task Force (IETF) and first published in 1996 as RFC 1889, superseded by RFC 3550 in 2003. RTP is used heavily in communication industry, especially TV services, video teleconference services, as well as telephony.

3.1.1 Protocol Functionality

RTP is used primarily in transporting streaming media, video or audio, over IP networks. It provides a means for jitter compensation and measures to detect out of sequence data. Through IP multicasting, RTP can transport data to multiple destinations.

Most real-time multimedia streaming applications can tolerate packet loss, although time is essential for data delivery. Accordingly, RTP is commonly used with UDP rather than TCP, as UDP favors time over reliability.

As RTP is responsible for transferring data, it provides timestamps, packet sequencing as well as payload format of the encoded data format. However, no information is provided regarding quality specifications and means of synchronizations, accordingly RTP Control Protocol (RTCP) is associated with RTP to complement it. We will be discussing RTCP in detail in the next section.

Signaling protocols are used as well with RTP to initialize the communication session between peers, such as Session Initiation Protocol (SIP), H.323, or Jingle (XMPP). Each

RTP session consists of an IP address with two ports, one for RTP and other for RTCP taking into consideration that video and audio stream each has separate session. It's recommended by the specification that RTP port should have even number and the RTCP port should have the next higher odd number for each associated RTCP port.

3.1.2 Profiles and Payload Formats

RTP is designed to have a multimedia range format and allows new formats without the need to revise the RTP standard. It was designed based on Application Level Format (ALF) architecture principal, where the application's specific information is not implemented in the RTP header, but it's provided in the profiles and payload format of RTP. RTP defines for each application class a profile and one or more associated payload formats. The profile provides the codecs needed to encode the payload data and their mapping to the payload format codes in the field payload Type (PT) of the RTP header. There are multiple payload format specifications for each profile, each format defines an encoded data transportation. Payload formats for audio include QCELP, MP3, G.711, G.723, G.726, G.729, GSM, and DTMF, and for video H.261, H.263, H.264, and MPEG-1/MPEG-2.

3.1.3 RTP Packet Structure

RTP packet header consists of 12 bytes followed by optional header extension. RTP header format is shown in Figure 8.

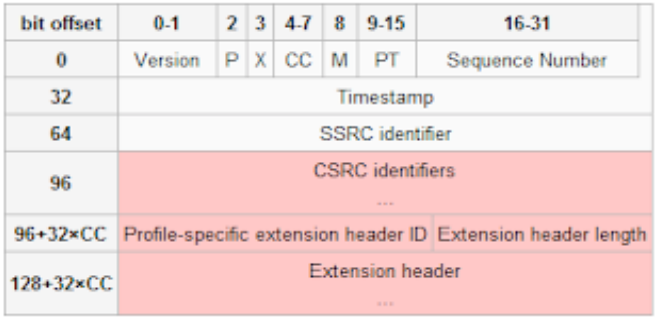


Figure 8: RTP Packet Header Format

In the below section we will discuss each header field [2]:

- Version: (2 bits) indicates protocol version.
- P (Padding): (1 bit) indicates the extra padding bytes at the end of the RTP packet.
- X (Extension): (1 bit) indicates the availability of extension header between payload data and standard header. Extensions are application specific.
- CC (CSRC count): (4 bits) Includes CSRC identifiers numbers that are after the fixed header.
- M (Marker): (1 bit) Defined by a profile and used in the application level. If marked, then current data are relevant to the application.
- PT (Payload type): (7 bits) Indicates the format of the payload and determines its interpretation by the application.
- Sequence number: (16 bits) the sequence number is incremented by one for each RTP data packet sent and is to be used by the receiver to detect packet loss and to restore packet sequence. The RTP has no measures for packet loss; however, the application is left to take appropriate action accordingly. For example, video applications may play

the last known frame in place of the missing frame. In RFC 3550 specification, the sequence number initial value should be random to make it difficult for attacks. RTP doesn't guarantee delivery, however the presence of sequence numbers allows the possibility to detect packets that are missing.

- Timestamp: (32 bits) Used to enable the receiver to play back the received samples at appropriate intervals. When several media streams are present, the timestamps are independent in each stream, and may not be relied upon for media synchronization. The granularity of the timing is application specific. For example, an audio application that samples data once every 125 μs (8 kHz, a common sample rate in digital telephony) would use that value as its clock resolution. The clock granularity is one of the details that is specified in the RTP profile for an application.

- SSRC: (32 bits) Synchronization source identifier uniquely identifies the source of a stream. The synchronization sources within the same RTP session will be unique.

- CSRC: (32 bits each) Contributing source IDs enumerate contributing sources to a stream which has been generated from multiple sources.

- Header extension: (optional) First 32-bit word contains a profile-specific identifier (16 bits) and a length specifier (16 bits) that indicates the length of the extension (EHL = extension header length) in 32-bit units, excluding the 32 bits of the extension header.

3.2 RTP Control Protocol

The RTP Control Protocol (RTCP) is complementary control protocol used to work in conjunction with RTP. RTCP packet structure and main functionalities are stated in RFC 3550 along with RTP specification. RTCP is designed to perform along with RTP, it's

responsible for packaging, delivery and quality control of multimedia data but not for any related transport functionality, which is mainly RTP functionality. The RTCP main functionality is providing feedback information on media distribution QoS, by regularly sending statistical information such as packet loss, packet latency, round-trip delay times, and octet transmission and packet counts to users in an active streaming multimedia session.

3.2.1 Protocol Functionality

There are four basic functions provided by RTCP [2]:

1. The primary function is to provide feedback on the quality of the data distribution. This is an integral part of the RTP's role as a transport protocol and is related to the flow and congestion control functions of other transport protocols. The feedback may be directly useful for control of adaptive encodings, but experiments with IP multicasting have shown that it is also critical to get feedback from the receivers to diagnose faults in the distribution. Sending reception feedback reports to all participants allows one who is observing problems to evaluate whether those problems are local or global. With a distribution mechanism like IP multicast, it is also possible for an entity such as a network service provider who is not otherwise involved in the session to receive the feedback information and act as a third-party monitor to diagnose network problems.

2. RTCP carries a persistent transport-level identifier for an RTP source called the canonical name or CNAME, Since the SSRC identifier may change if a conflict is discovered or a program is restarted, receivers require the CNAME to keep track of

each participant. Receivers may also require the CNAME to associate multiple data streams from a given participant in a set of related RTP sessions.

3. The first two functions require that all participants send RTCP packets, therefore the rate must be controlled for RTP to scale up to a large number of participants. By having each participant send its control packets to all the others, each can independently observe the number of participants. This number is used to calculate the rate at which the packets are sent.

4. An optional function is to convey minimal session control information, for example, participant identification to be displayed in the user interface. This is most likely to be useful in "loosely controlled" sessions where participants enter and leave without membership control or parameter negotiation. RTCP serves as a convenient channel to reach all the participants, but it is not necessarily expected to support all the control communication requirements of an application.

3.2.2 RTCP Packet Structure

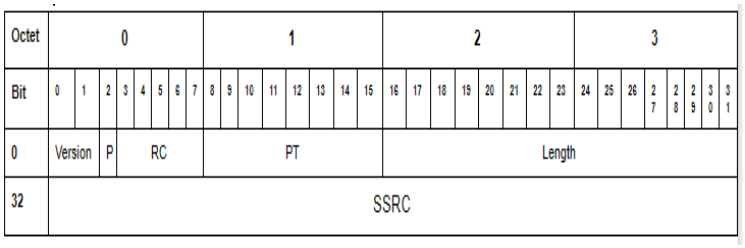


Figure 9: RTCP Packet Header

Figure 9 shows RTCP Packet Header, below will discuss each field [2]:

- Version: (2 bits) Identifies the version of RTP, which is the same in RTCP packets as in RTP data packets.

- P (Padding): (1 bits) Used to indicate if there are extra padding bytes at the end of the RTP packet. A padding might be used to fill up a block of certain size, for example as required by an encryption algorithm. The last byte of the padding contains the number of padding bytes that were added.
- RC (Reception report count): (5 bits) Number of reception report blocks contained in this packet. A value of zero is valid.
- PT (Packet type): (8 bits) Contains a constant to identify RTCP packet type [X].
- Length: (16 bits) Indicates the length of this RTCP packet.
- SSRC: (32 bits) Synchronization source identifier uniquely identifies the source of a stream.

3.2.3 RTCP Message Types

There are several message types in RTCP: sender report, receiver report, source description, goodbye, and application specific messages.

- Sender report (SR): A regular report sent during a conference by an active sender to report transmitting and receiving statistics for RTP packets. Absolute timestamp is included in the sender's report, to allow RTP messages synchronization.
- Receiver report (RR): A report sent to the non-sender RTP packets participants. It's mainly gives information about QoS for both senders and receivers.
- Source description (SDES): CNAME item is sent through this message to the participants in active session. It also provides information about source's owner/controller such as name, email address, address and phone number.

- Goodbye (BYE): To shut down a stream, the source sends a BYE message. It's announcement that the source is ending the conference.
- Application-specific message (APP): The application-specific message provides a mechanism to design application-specific extensions to the RTCP protocol [2].

3.3 Real -Time Streaming Protocol

Real-Time Streaming Protocol (RTSP) is an application layer protocol, used mainly to control end-to-end communication of real-time streaming media. It acts as "network remote control" for multimedia servers [38]. It provides a "VCR-style" remote control property for streamed media either video or audio, such as play, pause, reverse, fast forward and positioning. Streamed media could be video on demand or live audio streaming. RTSP is designed to work hand in hand with RTP and RTCP protocols. RTSP like RTCP is not responsible for multimedia data transmission, however RTP is the protocol responsible for multimedia data transmission. RTSP is designed to provide ways to choose delivery channels whether it's TCP, UDP or multicast UDP as well as delivery mechanisms based on RTP [38].

3.3.1 Protocol Functionality

RTSP establishes and controls streams of continuous audio and video media between the media servers and the clients. A media server provides playback or recording services for the media streams while a client requests continuous media data from the media server [5]. RTSP is the "network remote control" between the client and the server. It implements the following operations [5]:

1. Media Retrieval from the server: The client requests a session setup to the server to send the required data.

2. Invitation of a media server to a conference: The media server can be invited to the conference to play back media or to record a presentation.

3. Adding media to an existing presentation: The server or the client can notify each other about any additional media becoming available.

Like HTTP, each presentation and media stream is identified by an RTSP URL. A presentation description file is created to include all presentation and media properties such as encoding, destination addresses, RTSP URLs, language, port and other required parameters. This file can be obtained using HTTP or email or by the client. However, RTSP still differs from HTTP on several levels. More specifically, the RTSP server must maintain "session states" to be able to relate RTSP requests with a stream unlike HTTP which is stateless. HTTP is asymmetric protocol, where only the client can issue requests and server responds, whereas in RTSP, both the server and the client can issue requests.

3.3.2 RTSP Methods

RTSP requests Methods are discussed briefly below [38].

- OPTIONS: Accepted options by either client or server for the other party.

- DESCRIBE: Used to retrieve presentation or media description from a server.

- SETUP: Used in two different cases, namely, creating an RTSP session and changing the transport parameters of media streams that are already set up. SETUP can be used in all three states, Initiation (Init), Ready, and Play, to change the transport parameters.

- PLAY: The PLAY method tells the server to start sending data via the mechanism specified in SETUP and which part of the media should be played out. PLAY requests are valid when the session is in Ready or Play state.

- PLAY_NOTIFY: is issued by a server to inform a client about an asynchronous event for a session in Play state. The Session header MUST be presented in a PLAY_NOTIFY request and indicates the scope of the request. Sending of PLAY_NOTIFY requests requires a persistent connection between server and client; otherwise, there is no way for the server to send this request method to the client [6].

- PAUSE: halting the delivery of the stream temporarily without releasing server resources.

- TEARDOWN: Requesting the server to end delivery of the specified stream and release the resources associated with it.

- GET_PARAMETER: Retrieves the value of a parameter of a presentation or a stream specified in the URI.

- SET_PARAMETER: Sets the value of a parameter for a presentation or stream specified by the URI.

- REDIRECT: is issued by the server to inform the client that connection is terminated, and the client is required to connect to another server location. The mandatory location header indicates the new URL the client should connect to.

3.4 Stream Control Transmission Protocol

Stream Control Transmission Protocol (SCTP) is a reliable transport protocol running over IP-networks. SCTP protocol is equivalent to UDP and TCP. SCTP is similar TCP in terms of reliable transportation. SCTP ensure that all data is transported through network with no error as well in the correct sequence. SCTP as well is a connection oriented protocol, it requires connection establishment between end points before transmission occurs [39].

SCTP is designed to provide the following services:

1. Acknowledged error-free non-duplicated transfer of user data,
2. Data fragmentation to conform to discovered path MTU size,
3. Sequenced delivery of user messages within multiple streams, with an option for order-of-arrival delivery of individual user messages,
4. Optional bundling of multiple user messages into a single SCTP packet
5. Network-level fault tolerance through supporting of multi-homing at either or both ends of an association.

SCTP design includes congestion avoidance behavior and resistance to flooding and masquerade attacks that improves security [40].

3.4.1 Protocol Functionality

SCTP is message-oriented, it transports data as sequences of messages unlike TCP which transports bytes. A sender in SCTP sends each message in one operation, where it's passed to the receiver exactly the same as well as in one operation, which is like how UDP

functions. SCTP allows multi-streaming. It allows partitioning data into multiple streams each with independent delivery sequence, accordingly the loss of one stream will only affect transmission of that stream and not the others.

Multi-streaming is accomplished by separating data transmission and delivery. Each payload DATA "chunk" in the protocol uses two sets of sequence numbers. More specifically, a Transmission Sequence Number that is responsible for message transmission and message loss detection, and the Stream ID/Stream Sequence Number pair, which is used to determine delivery sequence of received data [40]. SCTP supports multi-homing, where one SCTP endpoint can support more than one IP address. This feature is very beneficial as it has higher potential for keeping session survived if network failure occurs. SCTP provides redundant paths to increase reliability. There are several types of multi-homing: Asymmetric multi homing, Local multi homing - Remote single homing, and Local single homing - remote multi homing.

3.4.2 SCTP Packet Structure

SCTP consists of two sections: (1) First 12 bytes consists of the common SCTP header. (2) The rest of bytes are occupied by data chunks. In Figure 10, shows SCTP packet structure, below we will describe header fields briefly:

- Source Port Number:(16 bits) SCTP sender's port number, used by the receiver along the source IP address, the SCTP destination port, and possibly the destination IP address to identify to which this packet belongs.

- Destination Port Number: (16 bits) SCTP port number to which the packet is to be delivered. The receiver uses this port number to de-multiplex the SCTP packet to the correct receiving endpoint/application [40].

Bits	0-7	8-15	16-23	24-31
0	Source Port		Destination Port	
32	Verification Tag			
64	Checksum			
96	Chunk 1 type	Chunk 1 flags	Chunk 1 length	
128	Chunk 1 data			
...	...			
...	Chunk N type		Chunk N flags	Chunk N length
...	Chunk 1 data			

Figure 10: SCTP Packet Structure.

- Verification Tag: (32 bits) Used by the receiver to validate the SCTP packet sender.
- Checksum: (32 bits) Includes the checksum value of the SCTP packet.
- Chunk Type: (8 bits) contains Chunk Value field type information. It takes a value from 0 to 254.
- Chunk Flags: (8 bits) depends on the Chunk type as given by the Chunk Type field. They are set to 0 unless otherwise specified on transmission.
- Chunk Length: (16 bits) identify the chunk size in bytes, including the Chunk Type, Chunk Flags, Chunk Length, and Chunk Value fields.
- Chunk Value: (variable length) contains the actual information to be transferred in the chunk. The usage and format of this field is dependent on the Chunk Type [40].

3.5 Secure Real Time Transport Protocol

Secure Real Time Transport Protocol (SRTP) states an extended profile of RTP, it is a protocol used to ensure message encryption, authentication, and confidentiality to apply traffic protection for RTP and RTCP streams. SRTP provides security for RTP applications whether it's unicast or multicast [41]. SRTP is designed to work along with RTP and RTCP protocols to provide secure transmission and delivery. SRTP is used to provide security for RTP and Secure RTCP (SRTCP) is used to provide security for RTCP.

3.5.1 Protocol Functionality

SRTP main security goals are to ensure: 1) RTP and RTCP payload confidentiality, and 2) RTP and RTCP packets integrity and protection from replayed packets. Some of the above security services are optional and others are mandatory, and these services are as well independent from each other. However, SRTP integrity protection is mandatory and must be included. One of the SRTP framework functionality is the ability to be upgradable which allows new cryptographic transforms. The SRTP framework has low bandwidth and computational cost. It preserves compression efficiency of the RTP header. By using pre-defined transforms, SRTP provides small sized code and memory for keying information and replay lists. SRTP is independent from underlying transport, network, and physical [41]. As mentioned earlier, SRTP is used to enhance security and used for key management. "A single "master key" can provide keying material for confidentiality and integrity protection, both for the SRTP stream and the corresponding SRTCP stream. This is achieved with a key derivation function, providing "session keys" for the respective security primitive, securely derived from the master key. In addition, the key derivation can be configured to periodically refresh the session keys, which limits the amount of

cipher text produced by a fixed key, available for an adversary to Cryptanalyze. "Salting keys" are used to protect against pre-computation and time-memory trade-off attacks" [41].

Multimedia Internet Keying (MIKEY) could be used with SRTP for key exchange, which could make an adequate security combination to provide protection for multimedia applications. SRTP applications must convert RTP packets to SRTP packets before transmission within the network and vice versa. Figure 11, shows this process [42].

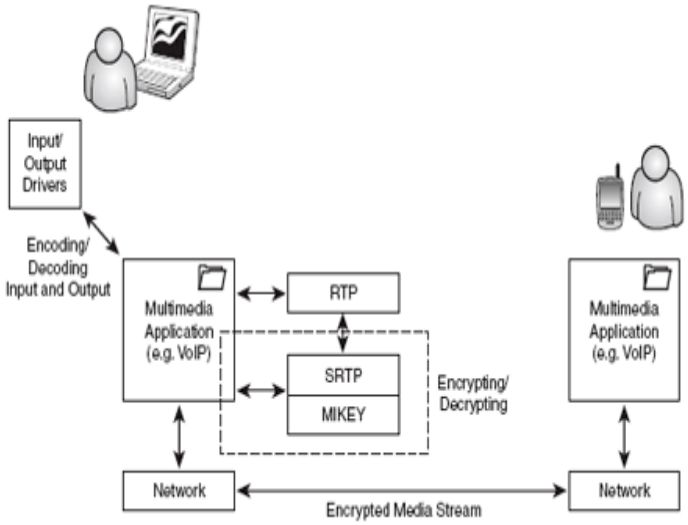


Figure 11: SRTP Encoding/Decoding [42].

Message authentication and integrity is supported by SRTP as mentioned before. SHA-1 is common algorithm used for message authentication. It uses 160-bit key length. By computing RTP message hash, RTP headers and encrypted payload, message authentication code is calculated and placed in the Authentication tag header as discussed in the section below [42].

3.5.2 SRTP Packet Structure

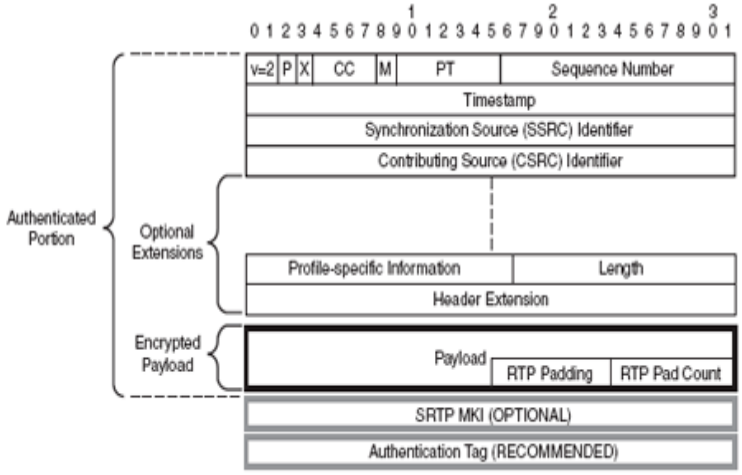


Figure 12: SRTP Packet Format.

SRTP packet format is shown in Figure 12. There are two portions one is for authentication and the other is for encryption. The encrypted portion is used for the encrypted RTP Payload including RTP Padding and Pad Count if existed. No padding is needed for pre-defined encryption transforms accordingly, SRTP and RTP payload sizes matches. The SRTP authenticated portion includes the RTP header followed by the encrypted portion of the SRTP packet. Encryption is applied before authentication on the sender side if both authentication and encryption are included, and vice versa on the receiver side. Field description for MKI and Authentication Tag:

- MKI (Master Key Identifier): For key management purposes. It provides master key derived from packet authentication/encryption session. MKI doesn't provide SRTP cryptographic context and is only used for re-keying and master key identification by key management [41].

- Authentication Tag: message authentication data is carried in this field. It authenticates both RTP header and payload as well and authenticates sequence number to provide replay protection.

Chapter 4

4 Proposed Design

4.1 Network Models in SDN

The difference between SDN and other traditional networks is the separation of network planes. SDN separates control, forwarding, and data planes to allow an agile, flexible configuration of the network and more programmability providing an open programmable interface. The software-defined and centralized nature of SDN allows more efficient traffic engineering to meet dynamic service requirements. SDN controllers have a global view of the network, making it an ideal physical substrate for cloud-based content networks environment. Controllers are entities responsible for monitoring and manage network flows, whether to forward, drop or buffer flows. Controllers provide statistical information about network, and identify network devices and capabilities, they are part of the control plan.

Forwarding plane includes any network devices: access points, routers, switches, etc. The OpenFlow protocol [43] is an open-source, standard protocol that governs communication between the SDN controller and network switches. OpenFlow allows full control over packet flows where the controller specifies the routing paths as well as how packets should be processed. One of the most important features in SDN networks is the fact that it can adapt to a service or a user requirement after setting up the whole network environment. Utilizing SDN provides the opportunity to do network optimization in specific domains. Implementing QoE applications over SDN allows us to enhance video quality and the ability to adjust video quality based on user feedback, not just by relying on QoS SLA.

There are a wide range of SDN controllers varying between a vendor specific controller and open source controllers that can support all vendors. Some popular controllers are NOX [44], Open Network Operating System (ONOS) [14], POX [45] and OpenDaylight [46]. In our design we chose ONOS as our main controller.

The reason for choosing ONOS as our controller as it offers flexibility to deploy and create new services by using easy to use programmatic API interfaces. It is a high performance platform that is scalable and allows for abstraction. It also supports real-time network control and configuration without the need to re-run control protocols of switching and routing inside the network. With ONOS, there is no need to adjust data plane systems in order to create new applications. The ONOS core is built on the ability to interact with services application via North Bound Interface (NBI) APIs. In addition ONOS has an intent-based programming (Intent Framework).

Intent Framework enables applications to provide network requests in a form of a policy not as a mechanism. Intents provide a high level abstraction where programmers only focus on the task that should be accomplished, rather than how these tasks will be translated into a low-level rules and how these rules can be installed into the network devices, by only expressing required “intentions” via high-level policies. Through the use of NBI APIs, ONOS can submit these intents and manage to translate them, through a compilation process See Figure 13 from high-level to low-level rules to achieve the programmer desired goals. Intent Framework provides the ability to recompile intents in case of network failures and dynamic network changes.

By using Rest API, we designed our QoE-Aware rating feedback application to communicate with ONOS via NBI. This design took advantage of utilizing ONOS

application abstraction and Intent Framework. QoE-Aware rating application provides feedback to ONOS in a form of an HTTP request and ONOS translate these intents to re-route traffic to an alternative shortest path to enhance QoE using ONOS Intent Reactive Forwarding application (IFWD) [47].

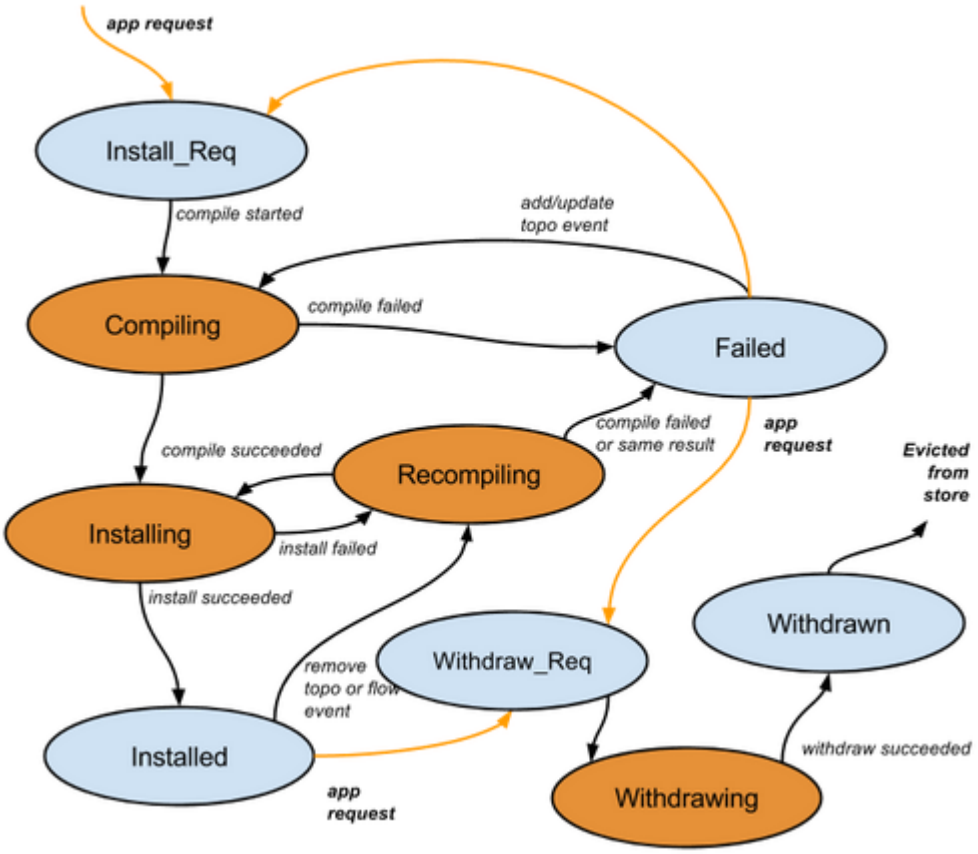


Figure 13: Intent compilation process [15].

A real-time cloud-based content delivery network model is shown in Figure 14. The network model is based on three tiers. Tier 1 contains the multimedia content providers that produce content for distribution over the network. Content can be a live stream or broadcast, music, video on demand, etc. Content is made to be accessible and processed by the various servers and forwarding modules in the SD-WAN in tier 2. Tier 2 consists of a collection of interconnected network switches and network devices, controlled by one or

more centralized controller(s) through a southbound protocol, such as Openflow. Tier 3 contains the wide range of end users that are accessing the desired content and provide constant feedback of QoE perceived video quality through the SD-WAN.

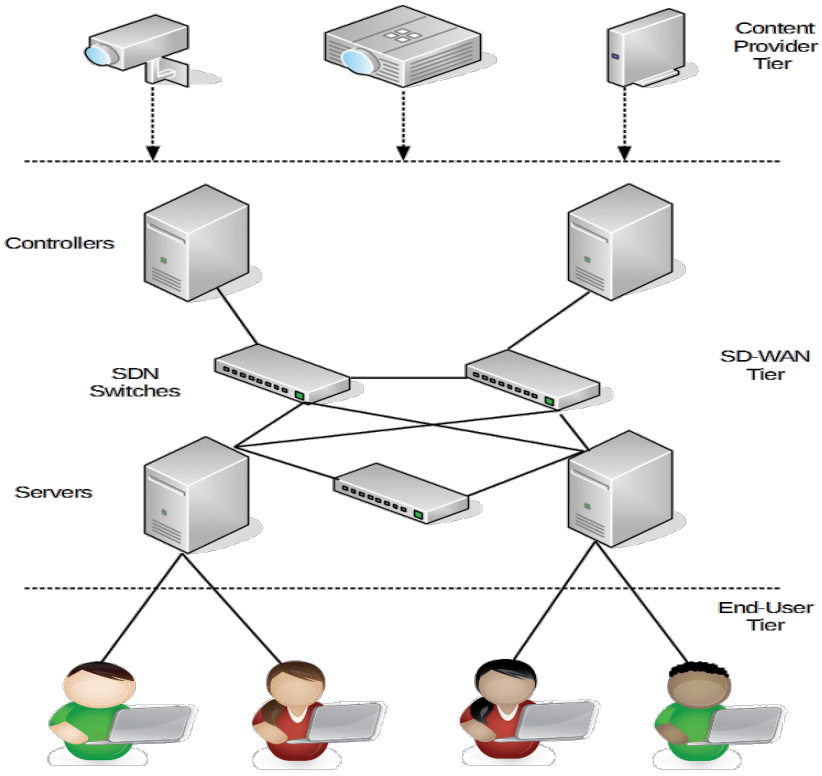


Figure 14: Real-time QoE content-based network model.

4.2 QoE-Aware Real-time Rerouting Model Design

The proposed model is based on real-time QoE crowdsourcing feedback of quality degradation during live video streaming over a cloud-based SD-WAN environment. Figure 15 highlights details of our model, in which we have a streaming server that streams multimedia content over the SD-WAN environment to multiple clients. The SDN environment provides a centralized view of the network allowing more control and flexibility to adapt to dynamic unexpected changes. Unexpected dynamic network changes and events could be imposed during video streaming experience. Several streaming

protocols can be adapted in our model including RTP, SCTP, TCP and UDP. SD-WAN processes the video stream and delivers it to the participant end users. A QoE-rating feedback application is deployed on the users' end to return user's feedback during a real-time streaming, where participants click on a button when they feel the quality has been degraded.

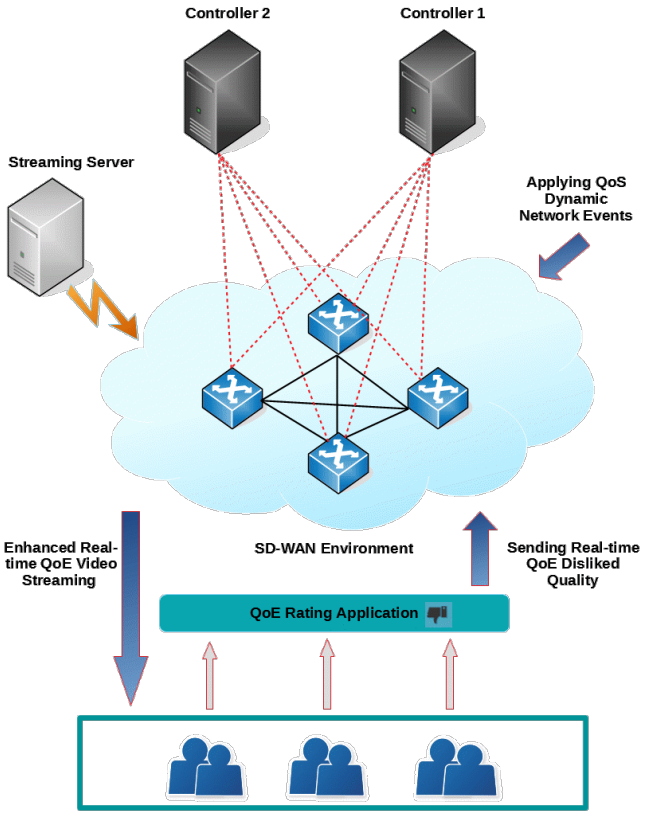


Figure 15: Real-Time QoE crowdsourcing feedback based on SD-WAN environment

The QoE-rating feedback application is designed to capture real-time user feedback via a click button and in return the application reacts by sending REST API requests to the SDN controller to inform the controller about potential problems in the current video stream and a possible request of a corrective action such as traffic rerouting. The QoE rating application can be deployed as a plugin on web browsers, multimedia players, or as a desktop application. In contrast to the MOS technique, participants do not have to decide

between different grading scales and what best suits their perception; instead, they only alert the SD-WAN controller to quality degradation, therefore allowing the users to provide more decisive feedback. The timing between feedbacks could be an indication of quality; i.e. more frequent feedback ("dislike" clicks) indicate a lower quality than less frequent feedback. As such, all clicks are timestamped before transmission to the controller.

Providing QoE feedback enables the SD-WAN controller to detect problems in the service path and to take corrective action by making changes to the virtual topology of the content delivery network, reassigning the users, or rerouting the traffic. Ideally, a resource optimization algorithm such as the method described in [48] can be executed in real time to respond to QoE degradation. However, our focus here is mainly on design and performance analysis of the framework for collecting and analyzing real-time QoE feedback.

4.3 QoE Crowdsourcing Paired Comparison Campaign Model design

Paired comparisons can be seen as an alternative for the MOS based method, due to its simplicity where the need to decide between five different ratings is eliminated, making it easier for intuitive judgment. A paired comparison makes it easier for end users to express their opinions, decision making and have an easier experience interaction when multiple factors are applied. We will discuss in the following part how we adapted our QoE-Aware SDN model discussed in Section 4.2 to utilize QoE paired comparison using the crowdsourcing method as an intuitive QoE feedback.

Using crowdsourcing reduces costs of subjects hired as there is no need to have end users physically present and will reduce experiment time and handling making it more efficient. In a crowdsourcing experiment, end users choose to participate on their own time

which eliminates time constraints during the experiment. Crowdsourcing allows for a wide subject demographic making it more scalable. Figure 16 illustrate the crowdsourcing pair comparison model design.

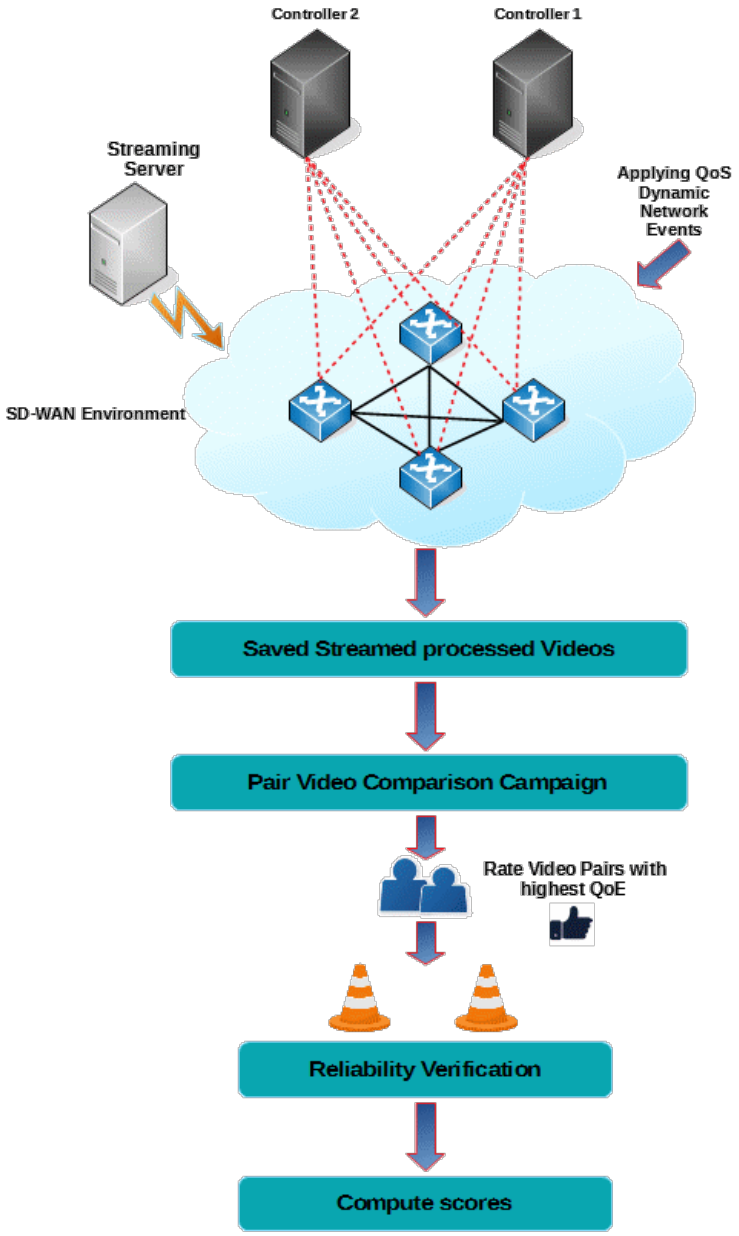


Figure 16: QoE crowdsourcing pair video comparison model design.

As mentioned earlier, any QoE network model requires a tier where the content is provided. A multimedia streaming service is used to enable multimedia content streaming

over the SD-WAN environment. The content is processed and transmitted via SD-WAN environment. SDN environment provides a centralized view of the network allowing more control and flexibility to adapt to changes. Unexpected dynamic network changes and events could be imposed during any video streaming experience. In our model, we consider network factors (loss, delay, jitter, etc.) that may affect the perceived quality at the user's end. We consider different streaming protocols which can behave differently with imposed dynamic changes and factors. Our model is designed to be flexible and allow different streaming protocols such as RTP, SCTP, TCP, UDP, etc.

By simulating these dynamic factors, the video stream is saved to be compared against different changing factors and protocols to measure QoE via end user's participation. These saved videos are used to construct a paired comparison stimulus in any chosen crowdsourcing platform in terms of a campaign to be advertised to end users asking for participation. These campaigns include instruction of what is required to participate, consent and terms of any designed stimuli. It includes all processed saved videos and paired comparison stimuli are constructed accordingly.

Participants are then hired to rate videos in a paired comparison stimulus with different streaming protocols to provide feedback regarding which video has the higher QoE. In any crowdsourcing model design, there must be a method to identify reliable trustworthy participants since experiments are not controlled and lack monitoring that traditional controlled lab environments provide. To verify reliable participant, trap questions must be introduced. Participants are presented with reliability verification paired comparison stimuli "golden question", where a stimulus constructed between the original video with no applied processing as a reference and an event degradation processed video. If

participant didn't rate original video with higher QoE, then participant's results are excluded, and those participants are considered unreliable. Any campaign is conducted over a period of time depending on the number of participants required to conduct the study. For example, a campaign is concluded if a certain number is met or a specific date was chosen as an end date. Rating scores can be computed at any point during the campaign to show trends and results. Different methods can be used to compute these scores such as Crowd Bradley-Terry Model [49].

Chapter 5

5 Implementation Details and Simulation Models

In this chapter, we will discuss implementation steps and the development of simulation models. We have conducted three different experiments to assess QoE. The first two experiments are designed on subjective QoE: crowdsourcing human paired comparison (HPC) and QoE-aware real-time rerouting. The last experiment is based on objective QoE and how perceived QoE can be calculated. The VLC player was chosen as default player due to its wide support for different streaming protocols and video codecs.

5.1 Dynamic QoS Network changes

Implementation of dynamic network changes is based on the minievent [50] module. Minievent is built on mininet as a framework to introduce event generation. Time changing events are applied to the mininet network links during streaming session. These events are defined an external minievent.json file. A set of changing events are known as scenarios. This module can provide information of UDP and TCP traffic and allow loss, delay and bandwidth modification. An example of minievent.json file in Figure 17. We applied modifications to the main script minievent.json to accommodate experiments and the built-in vlc streaming commands.

5.2 Human-based Participant Experiments

Two experiments were conducted using human participants, one was based on real-time feedback during video streaming in a controlled lab environment including 20 under grad and graduate student's participants, and the other is based on paired video comparison using QoE crowdsourcing technique with 135 anonymous unique participants. These

experiments were approved by Ontario Tech University Research Ethics Board (number 14780).

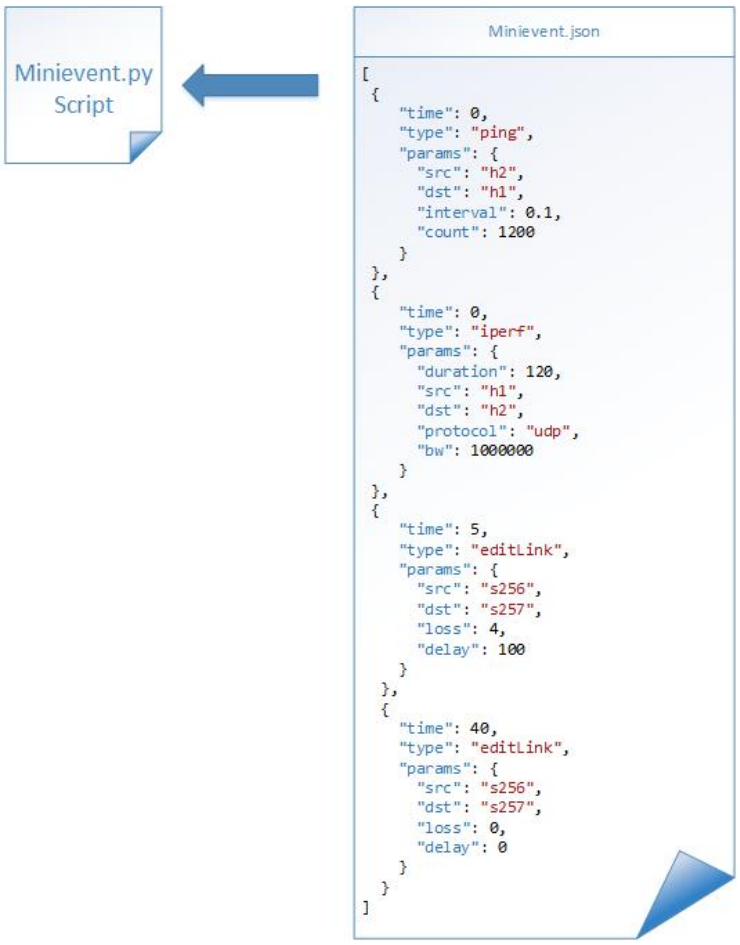


Figure 17: Example of minievent.json file and how loss, delay, bandwidth, ping and iPerf can be fed to minivent.py script over time.

5.2.1 QoE-Aware Real-Time Rerouting Experiment

The main objective of this experiment is to demonstrate that QoE feedback can be captured in real-time during live video streaming session. Feedback can be sent on-spot to the SDN controller to alert controller of issues in the steaming service. We demonstrate how traffic reroute can be done instantly based on this feedback. Results will be compared against already known time degradation events generated and to determine whether all event delay and loss event changes were detected by participant.

5.2.1.1 Configuration

For this experiment, a VM was built on VMware Workstation 15 player, with Ubuntu 8.04.02 LTS running as Operating System (OS). With Mininet 2.2.2 version, Python 2.7.15rc1 version and Open vSwitch (OVS) 2.9.2 version. Mininet uses ONOS 2.1.0 as a remote controller. ONOS was setup using Docker CE 18.09.6 version. For video steaming, VLC player version 3.0.4 was installed and VLC client lua plug-in enabled to integrate feedback QoE rating application. To allow logs generations and graph plotting, we installed the following python libraries: moreutils, NumPy, and matplotlib.

5.2.1.2 Implementation Setup

Based on proposed design discussed in Chapter 4, I created a SDN-WAN environment using mininet and ONOS remote controller, which provides an emulation of a software-defined virtual network similar to a real networking environment, running kernel, switching and application code in one single virtual machine using a simple in line code. Network topology consists of one remote controller (172.17.0.2), three hosts: one acts as a server (h1: 10.0.0.1), the second acts as a client (h2: 10.0.0.2) and the third un-namespaced server (h3) which can communicate with ONOS, and 10 OVS switch devices with 22 links and 50 flows. The streaming session is always running between same the client (10.0.0.2) and server (10.0.0.1). RTP was used as the default streaming protocol. Figure 18 illustrates the network topology build captured from ONOS Web User Interface (UI); the route marked green denotes the current streaming path:

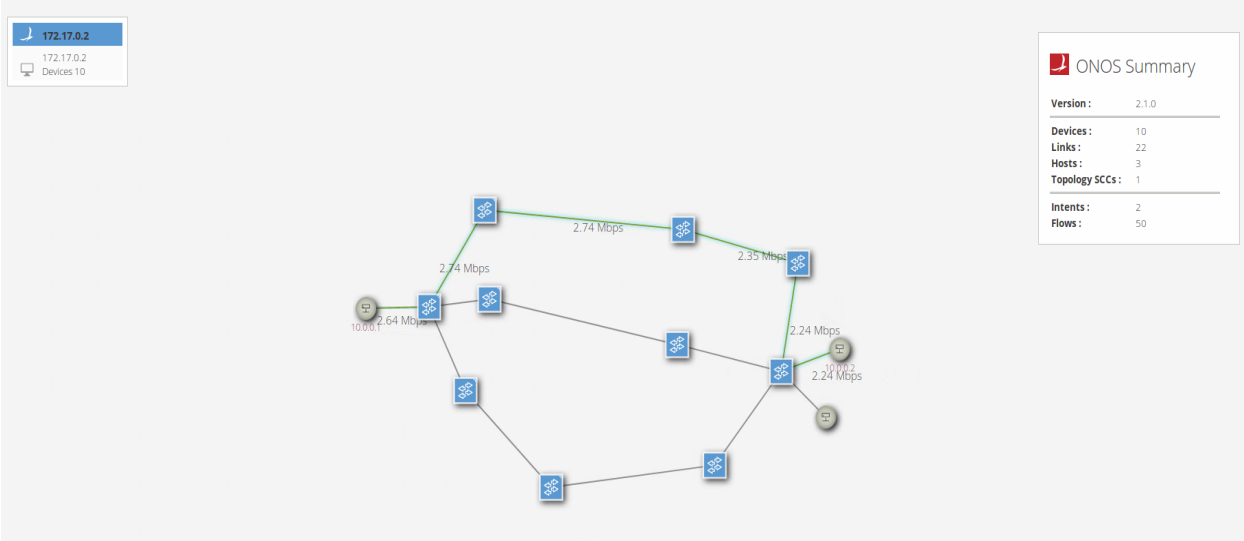


Figure 18: SDN network topology.

In Figure 19, shows switch devices established with mac addresses, device ID, Mininet name and number of ports.

	FRIENDLY NAME ▲	DEVICE ID	NAME	PORTS
✓	of:0000000000000001	of:0000000000000001	s1	5
✓	of:0000000000000002	of:0000000000000002	s2	6
✓	of:0000000000000100	of:0000000000000100	s256	3
✓	of:0000000000000101	of:0000000000000101	s257	3
✓	of:0000000000000200	of:0000000000000200	s512	3
✓	of:0000000000000201	of:0000000000000201	s513	3
✓	of:0000000000000202	of:0000000000000202	s514	3
✓	of:0000000000000300	of:0000000000000300	s768	3
✓	of:0000000000000301	of:0000000000000301	s769	3
✓	of:0000000000000302	of:0000000000000302	s770	3

Figure 19: Switch devices details.

Figure 20 illustrated Mininet network topology build, starting with adding ONOS remote controller, followed by establishing and discovering network connections, starting VLC video stream and finally activating required ONOS applications and their dependencies; IFWD application [47] to enable installation of host2host intent automatically when packet-in arrives, OpenFlow app and Proxy ARP app.

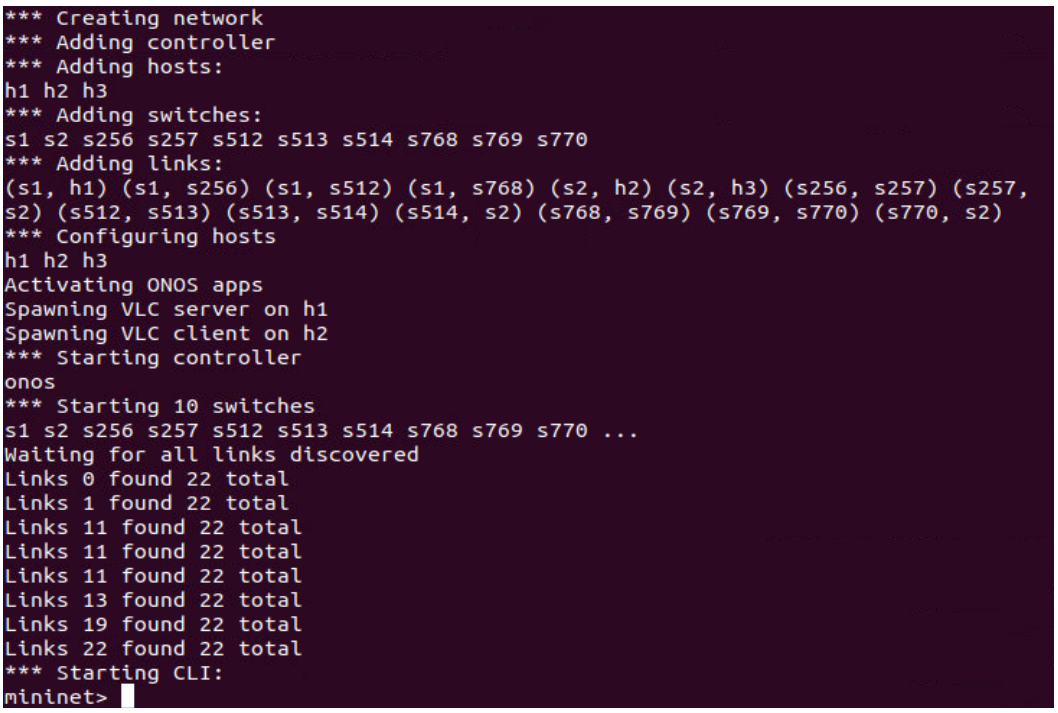


Figure 20: Starting SDN environment, ONOS, ONOS apps and initiate video streaming via VLC player.

To initiate the experiment, the feedback "Dislike" button must be activated via VLC menu. Button can be activated by clicking View -> SDN (activate). Through Mininet. CLI, dynamic network events scenarios can be initiated by calling a scenario you wish to apply for the current streaming session by running the following command: mininet> events loss-delay.json as show in Figure 21. At this point the experiment setup is ready.

During video streaming experiences, we applied network degradation parameters that affected the quality of the network in order to indicate if participants will be able to detect that change with their user experience interaction. Is the feedback timely relevant with imposed degradation change events? And whether these changing events are noticeable to the participants.

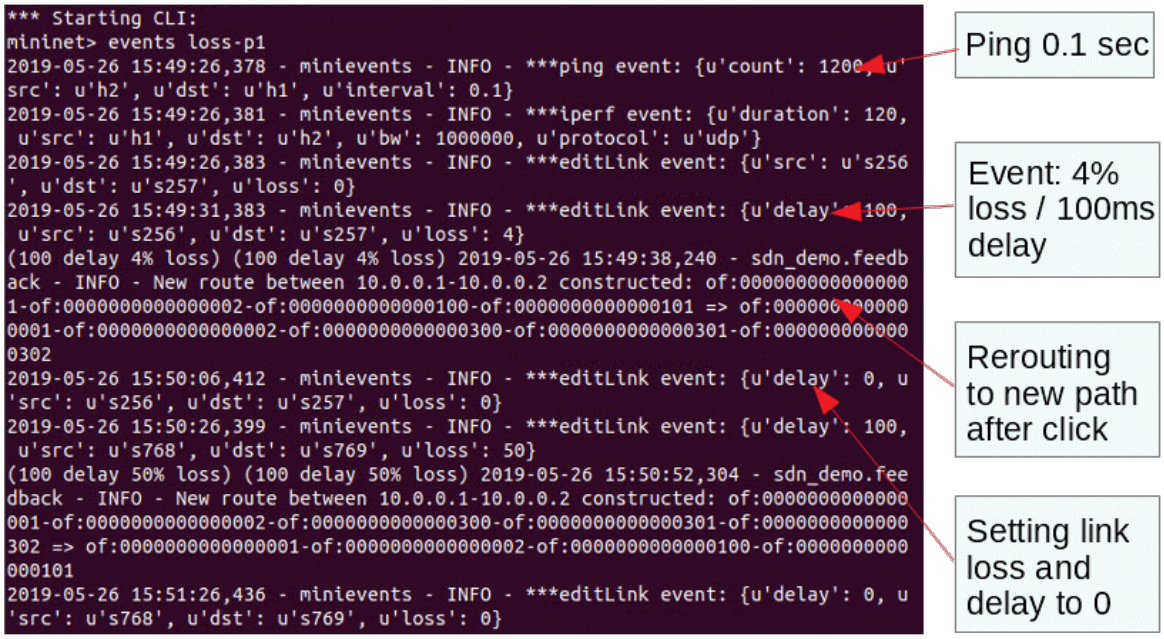


Figure 21: Running events command, RTT ping, edit delay and loss on link, and rerouting confirmation from one path to another after 'Dislike' click.

The experiment works as follows:

1. User clicks ‘Dislike’ button when quality degrades as shown in Figure 22.
2. HTTP request received by server on 10.0.0.3 requesting for rerouting.
3. OSD message displays on VLC player to notify user that a new route is being searched
4. Server (10.0.0.3) communicate with controller and call main logic in controller.py
5. Resolve client and server IPs into MACs using REST.
6. Retrieve current intent information (the intent was created using IFWD application reactively).
7. Query current traffic path.
8. Query some alternative paths between same two points.

9. For all devices on the current path count how many times this device belongs to an alternative path (weighted random of how many times this device meets other paths)

10. From devices on the current path remove unavoidable devices (shared by all known paths) and choose a new Obstacle Constraint randomly.

11. Alter intent by adding obstacle to one of devices on current path (so ONOS is forced to rebuild a new path)

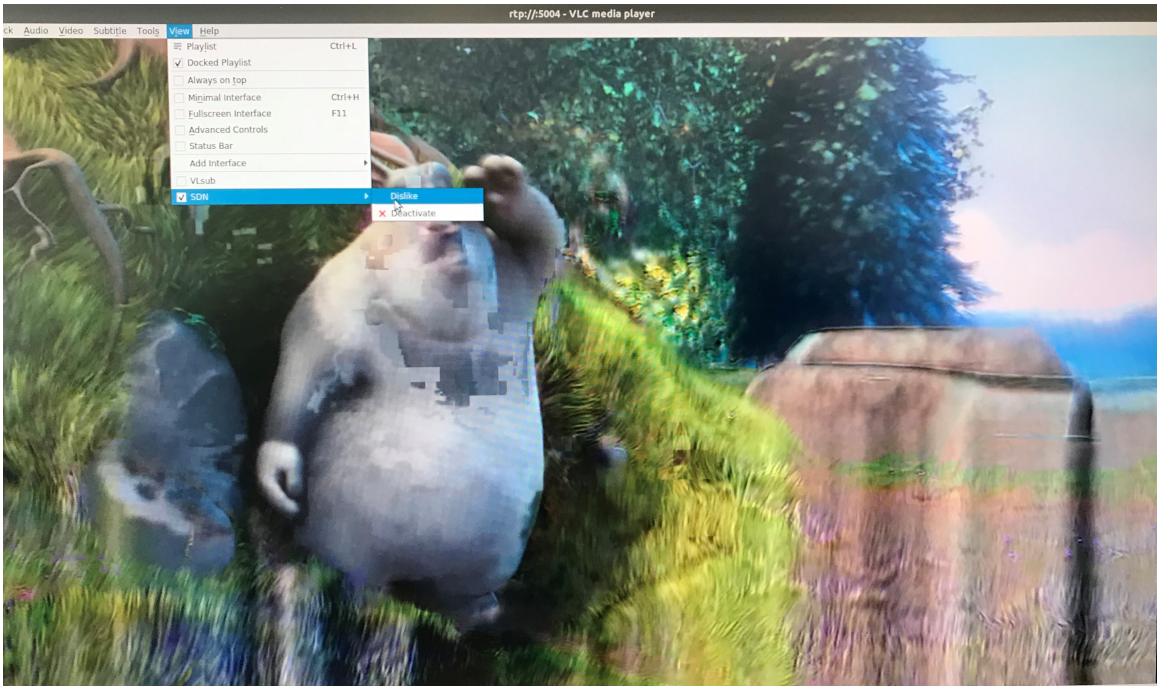


Figure 22: VLC embedded dislike button.

12. POST modified intent with new obstacle.

13. Wait until this modified intent installed and query new path (ONOS computes the shortest path in terms of hops. So, if there are 3 paths of length 1, 1, and 2 and the obstacle was placed on first path, ONOS will always choose path 2 not 3, unless a link is down on path 2).

14. OSD message displays on VLC player search is complete

15. New streaming path established and QoE is enhanced.

Figure 23 shows current traffic path before requesting traffic reroute in green and Figure 24 shows traffic path in green after rerouting captured from ONOS web UI.

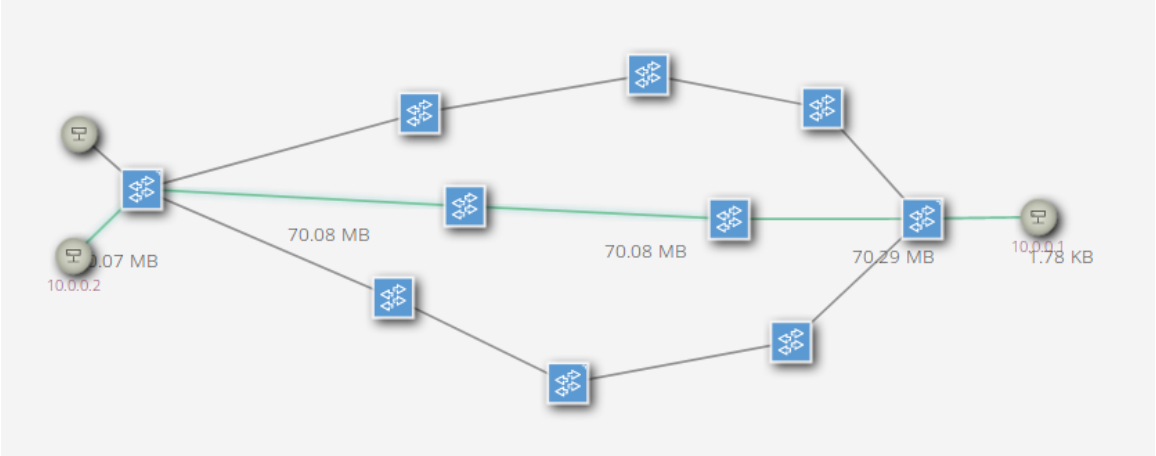


Figure 23: Current streaming traffic in green before submitting QoE feedback for rerouting request.

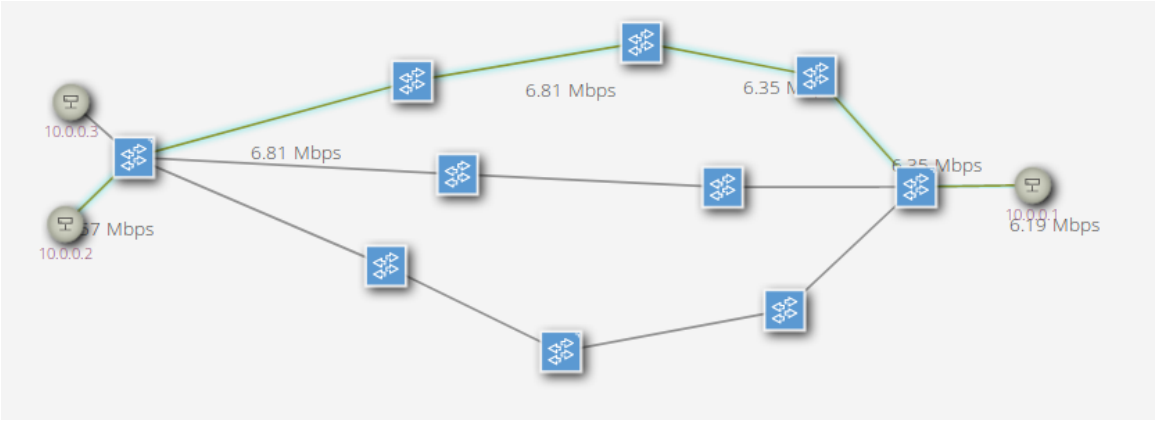


Figure 24: Current streaming traffic in green after submitting QoE feedback for rerouting request.

5.2.1.3 QoE Simulation Scenarios

In our experiment, we had three scenarios in one three-minute streaming session. Dislike click is deactivated 10 sec between clicks to ensure that path rerouting is

successfully implemented. Table 3 below shows each scenario time period, and events introduced for delay and loss:

Table 3. QoE-Aware real-time rerouting experiment simulation scenarios.

	Scenario 1 (5s-60s)				Scenario 2 (65s-150s)						Scenario 3 (160s-180s)			
Time	**5s**	**40s**	**45s**	**60s**	**65s**	**90s**	**110s**	**130s**	**131s**	**150s**	**160s**	**161s**	**170s**	**180s**
Delay	50ms	0ms	100ms	0ms	100ms	1ms	2ms	0ms	100ms	0ms	100ms	2ms	0ms	0ms
Loss	4%	0%	50%	0%	0%	1%	1%	0%	2%	0%	3%	1%	0%	0%

In Scenario 1, I introduced a delay and loss on current path. User is expected to express dislike of current streaming quality and rerouting will occur after some time with enhanced QoE after first click. I introduced another event changes on the newly rerouted streaming path, user is expected to press dislike and ONOS controller would look for another path and the streaming QoE is enhanced accordingly. Scenario 1 was the first 60 secs of the video streaming session and it was designed to demonstrate how clicks can help enhance participants QoE. In Scenario 2, only delay event was applied on the current stream path, then gradually adjusted the delay with a small loss%, then increased both the delay and loss%, all these events applied on same and current streaming path. In this scenario, I was trying to determine if users was able to capture first event change or not, for 2nd event change would little latency tolerated by the user? Would the user click to reroute traffic before 3rd event change or the user would tolerate the little degraded quality before making final decision that Quality must be enhanced and ask for rerouting after 3rd change events occured. This scenario took 85 sec. In Scenario 3, delay and loss were introduced on two different streaming paths, one with bad quality and the other with a slightly enhanced quality, User is expected to keep rerouting between streaming paths until ONOS finds the better QoE. Last scenario elapsed for 20 sec and requires at least two dislike clicks to enhance quality.

5.2.2 Crowdsourcing Human Paired Comparison (HPC)

5.2.2.1 Configuration

For the HPC experiment, I built VM on AWS platform. The machine was running on Ubuntu 6.04 LTS operating system. With Mininet 2.2.2 version, Python 2.7.12+ version and Open vSwitch 1.6 version. Default Mininet controller was used. I have installed VLC player 2.2.4 version for video streaming.

5.2.2.2 Implementation Setup

HPC experiments was set as a QoE crowdsourcing campaign of a paired video comparison on a set of processed videos using subjectfy.us web service [37]. Using Mininet simple topology, I created a network topology that consists of one remote controller (127.0.0.1) on port number 6653, two hosts: one acts as a server (h1: 10.0.0.1), and the second acts as a client (h2: 10.0.0.2) and 1 OVS switch device with 2 links. The streaming session is always running between same client (10.0.0.2) and server (10.0.0.1). By using minievent module, I applied changes to adjust the delay and loss over a period of time during real-time video streaming, in order to measure the impact of changes in network conditions on QoE. During video streaming experiences, I applied network degradation parameters which affect the quality of the network to indicate if the participating end user will be able to detect such changes within the network during video streaming and whether these changes affect their QoE. Whether the feedback was submitted within changing events time window. And how streaming protocols will be able to adapt to these changing events, and if all or some of these changing events will be noticeable to participants, then these feedbacks are compared to the previously calculated QoE by objective VQM analysis measure.

Our event scenarios were chosen to check how QoE feedback is affected based on these criteria: 1) Fixing delay and changing loss over time 2) Switching from high delay to a substantial lower delay with fixing loss 3) Gradually decreasing delay over time and then gradually increasing it while applying loss.

5.2.2.3 QoE Simulation Scenarios

In this experiment, the link delay was chosen in the range [0-80ms] and loss was [0 or 1%] where 0 indicates resetting loss/delay to default configuration during specific event scenario. The link bandwidth was fixed at 50 Mbs. We created scenario files of 40 secs in length with timed changing events where delay and loss were changed during the video streaming session and feed to the mini-event events changing script shown in Appendix A. A VLC server was used as the video streaming application. For every scenario, we streamed video with different streaming protocols, saved all output videos and ran a rating paired comparison.

Table 4. HPC video specification.

	Content	Frame Width	Frame Height	Bitrate	Frame rate
Video 1	Beach waves	1920	1080	17388 kbps	50 frames/sec
Video 2	Ski views	1920	1080	5141 kbps	30 frame/sec
Video 3	Animation	1920	1080	1249 kbps	60 frame/sec
Video 4	Wild animals	640	360	1554 kbps	30 frame/sec

We created comparable video sets based on four selected HD videos in Table 4. Each video is 40 sec in time. We created five random scenario files with changing delay and loss events every 10 or 5 sec shown in Table 5, and applied RTP, Legacy UDP and RTP over TCP as streaming protocols in different scenarios. 135 unique anonymous users participated in this experiment, each was asked 10 questions and two verification questions. With total of 2430 questions. To verify trusted reliable participants, a golden question as

video paired comparison between unprocessed original HD video and processed video was asked. If participants choose processed video, then their feedback results were considered as untrustworthy.

Table 5. Scenarios of events applied in streamed videos in Human Paired Comparison Experiment

Time	**0s**	**10s**	**20s**	**30s**	**35s**
Scenario 1					
Delay	10ms	10ms	50ms	0ms	25ms
Loss	1%	0%	0%	1%	1%
Scenario 2					
Delay	70ms	10ms	5ms	5ms	25ms
Loss	0%	1%	1%	0%	0%
Scenario 3					
Delay	75ms	0ms	10ms	35ms	5ms
Loss	1%	0%	1%	0%	1%
Scenario 4					
Delay	35ms	80ms	5ms	25ms	5ms
Loss	0%	0%	1%	1%	0%
Scenario 5					
Delay	35ms	45ms	45ms	25ms	65ms
Loss	1%	0%	1%	0%	0%

Using MSU-VQM objective analysis tool [51], we computed comparative results against the original videos for PSNR, VQM and SSIM.

5.3 VQM Experiment

For this experiment, I used same VM configuration and mininet topology used in HPC experiment. Link delay was chosen randomly in the range [0-50ms] and loss was [0 or 1%] where 0 indicates resetting loss/delay to default configuration during specific event scenario. The link bandwidth was fixed at 50 Mbs. Ten random scenario files of 40 sec in length were created with a timed changing events every 0.5 sec during the video streaming session as shown in Table 6. These changing events scenarios were applied on Video 1, Video 2 and Video 3 with specification mentioned in Table 4, and applied RTP, Legacy UDP, SCTP, RTP over UDP and RTP over TCP as streaming protocols with a result of 150 processed videos. Using MSU-VQM objective analysis tool, I computed comparative results against the original videos for PSNR, VQM and SSIM.

Table 6. VQM Experiment applied events scenarios.

Time	**20s**	**25s**	**30s**	**35s**	**40s**	**45s**	**50s**
Scenario 1							
Delay	15ms	45ms	25ms	5ms	30ms	40ms	50ms
Loss	1%	1%	0%	1%	1%	1%	1%
Scenario 2							
Delay	20ms	5ms	40ms	20ms	40ms	30ms	35ms
Loss	0%	0%	0%	1%	0%	0%	0%
Scenario 3							
Delay	30ms	10ms	10ms	45ms	10ms	15ms	15ms
Loss	0%	0%	1%	0%	0%	0%	1%
Scenario 4							
Delay	45ms	5ms	50ms	40ms	0ms	35ms	50ms
Loss	0%	0%	1%	1%	1%	0%	0%
Scenario 5							
Delay	35ms	35ms	50ms	35ms	50ms	15ms	15ms
Loss	1%	0%	0%	0%	1%	1%	0%
Scenario 6							
Delay	15ms	25ms	15ms	25ms	35ms	15ms	15ms
Loss	0%	0%	0%	0%	0%	1%	1%
Scenario 7							
Delay	50ms	50ms	40ms	35ms	50ms	20ms	25ms
Loss	0%	1%	0%	1%	1%	0%	0%
Scenario 8							
Delay	0ms	15ms	50ms	5ms	30ms	45ms	15ms
Loss	0%	1%	1%	0%	0%	1%	0%
Scenario 9							
Delay	0ms	5ms	30ms	50ms	25ms	15ms	40ms
Loss	1%	0%	1%	0%	0%	0%	1%
Scenario 10							
Delay	0ms	0ms	50ms	5ms	50ms	45ms	25ms
Loss	0%	0%	1%	0%	0%	0%	1%

Chapter 6

6 Results and Analysis

6.1 QoE-Aware Real-Time Rerouting Results and Analysis

As mentioned earlier, this experiment is based on a controlled lab-environment. We had 20 under graduate and graduate student participants between the ages of 19-32 years old. All signed consent forms based on RFB. Each participant watched a 3 minutes live streaming video and was asked to provide QoE feedback in terms of a 'dislike' click when quality degrades based on timely changed scenarios of event. Participants can provide feedback clicks any time during the viewing session. Between each click, there is a 10 sec enforced waiting time to ensure controller's rerouting is complete.

```
mininet> events loss-delay
2019-07-06 22:17:18,742 - minievents - INFO - ***ping event: {u'count': 1200, u
'src': u'h2', u'dst': u'h1', u'interval': 0.1}
2019-07-06 22:17:18,743 - minievents - INFO - ***iperf event: {u'duration': 200
, u'src': u'h1', u'dst': u'h2', u'bw': 1000000, u'protocol': u'udp'}
2019-07-06 22:17:18,745 - minievents - INFO - ***editLink event: {u'src': u's25
6', u'dst': u's257', u'loss': 0}
2019-07-06 22:17:23,745 - minievents - INFO - ***editLink event: {u'delay': 50,
 u'src': u's256', u'dst': u's257', u'loss': 4}
(50 delay 4% loss) (50 delay 4% loss) 2019-07-06 22:17:30,293 - sdn_demo.feedba
ck - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:000000000000000
1-of:0000000000000002-of:0000000000000100-of:0000000000000101 => of:00000000000
00001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000
000302
2019-07-06 22:17:58,773 - minievents - INFO - ***editLink event: {u'delay': 0,
u'src': u's256', u'dst': u's257', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-07-06 22:18:03,746 - minievents - INFO - ***editLink event: {u'delay': 100
, u'src': u's768', u'dst': u's769', u'loss': 50}
(100 delay 50% loss) (100 delay 50% loss) 2019-07-06 22:18:10,982 - sdn_demo.fe
edback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:00000000000
00001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000
000302 => of:0000000000000001-of:0000000000000002-of:0000000000000100-of:000000
0000000101
2019-07-06 22:18:18,756 - minievents - INFO - ***editLink event: {u'delay': 0,
u'src': u's768', u'dst': u's769', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-07-06 22:18:23,747 - minievents - INFO - ***editLink event: {u'delay': 100
, u'src': u's256', u'dst': u's257'}
(100 delay) (100 delay) 2019-07-06 22:18:48,767 - minievents - INFO - ***editLi
nk event: {u'delay': 1, u'src': u's256', u'dst': u's257', u'loss': 1}
(1 delay 1% loss) (1 delay 1% loss) 2019-07-06 22:18:55,505 - sdn_demo.feedback
 - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-
of:0000000000000002-of:0000000000000100-of:0000000000000101 => of:0000000000000
001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:000000000000
0302
2019-07-06 22:19:08,760 - minievents - INFO - ***editLink event: {u'delay': 2,
u'src': u's256', u'dst': u's257', u'loss': 1}
(2 delay 1% loss) (2 delay 1% loss) 2019-07-06 22:19:28,761 - minievents - INFO
 - ***editLink event: {u'delay': 0, u'src': u's256', u'dst': u's257', u'bw': 10
00, u'loss': 0}
```

```
2019-07-06 22:19:29,742 - minievents - INFO - ***editLink event: {u'delay': 100
, u'src': u's768', u'dst': u's769', u'loss': 2}
(100 delay 2% loss) (100 delay 2% loss) 2019-07-06 22:19:37,057 - sdn_demo.feed
back - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000
001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:000000000000
0302 => of:0000000000000001-of:0000000000000002-of:0000000000000100-of:00000000
00000101
2019-07-06 22:19:48,760 - minievents - INFO - ***editLink event: {u'delay': 0,
u'src': u's768', u'dst': u's769', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss)
(1000.00Mbit 0 delay 0% loss)
2019-07-06 22:19:58,751 - minievents - INFO - ***editLink event: {u'delay': 100
, u'src': u's256', u'dst': u's257', u'loss': 3}
(100 delay 3% loss) (100 delay 3% loss) 2019-07-06 22:19:59,742 - minievents -
INFO - ***editLink event: {u'delay': 2, u'src': u's768', u'dst': u's769', u'los
s': 1}
(2 delay 1% loss) (2 delay 1% loss) 2019-07-06 22:20:06,133 - sdn_demo.feedback
 - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-
of:0000000000000002-of:0000000000000100-of:0000000000000101 => of:0000000000000
001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:000000000000
0302
2019-07-06 22:20:08,750 - minievents - INFO - ***editLink event: {u'delay': 0,
u'src': u's256', u'dst': u's257', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-07-06 22:20:13,664 - sdn_demo.feedback - INFO - New route between 10.0.0.1
-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000
300-of:0000000000000301-of:0000000000000302 => of:0000000000000001-of:000000000
0000002-of:0000000000000100-of:0000000000000101
2019-07-06 22:20:18,751 - minievents - INFO - ***editLink event: {u'delay': 0,
u'src': u's768', u'dst': u's769', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss)
(1000.00Mbit 0 delay 0% loss)
mininet> exit
```

Figure 25: Applying minievents changes, capturing timestamped QoE feedback button clicks by participant 1 in QoE-aware experiment and rerouting physical path information before and after during real-time streaming session.

Above figure shows the results of applying the three scenarios of changing events during video streaming, QoE feedback decisions by Participant 1 in our experiment and the controller's rerouting decision after each QoE feedback click by Participant 1. More results samples can be found in Appendix B.

By plotting resulted data, Figure 26.a ,b ,c ,d ,e show the results for QoE-aware rerouting experiment for five participants, where red dots is ping time (right axis), blue line is iperf %loss (left axis), green bars are vlc error and the gray background - minievents scenarios active (>0 links with loss! = 0 or bw < 512 or delay>0).

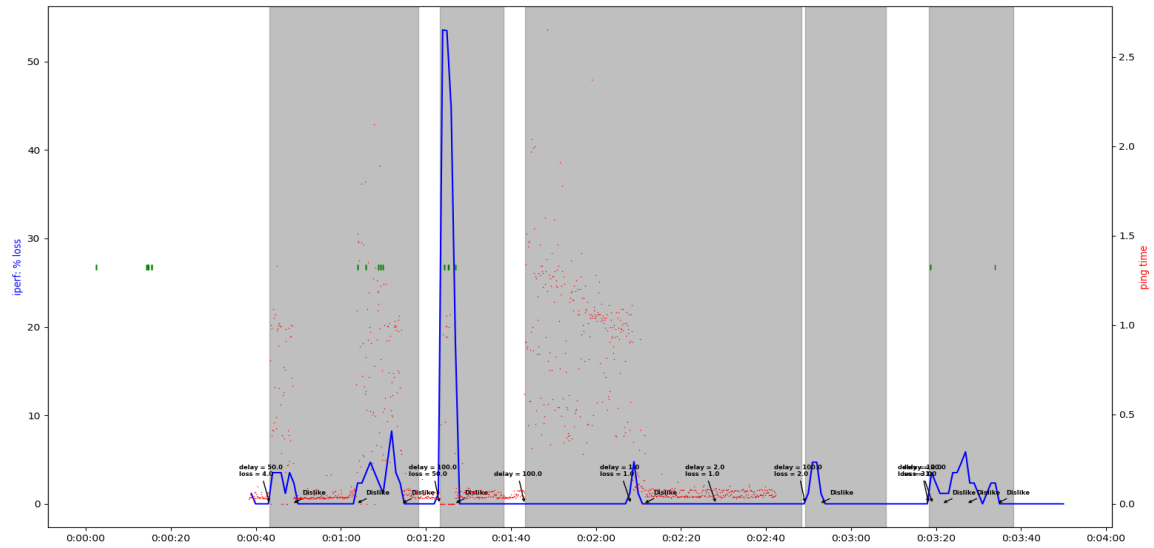


Figure 26.a: Plotted QoE Rerouting Results decisions for participant 1

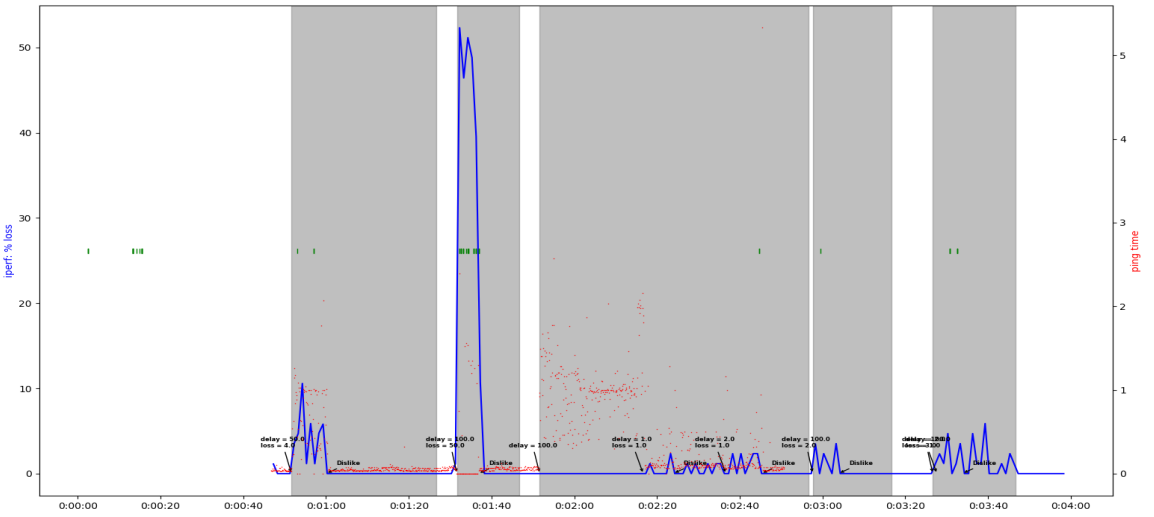


Figure 26.b: Plotted QoE Rerouting Results decisions for participant 2

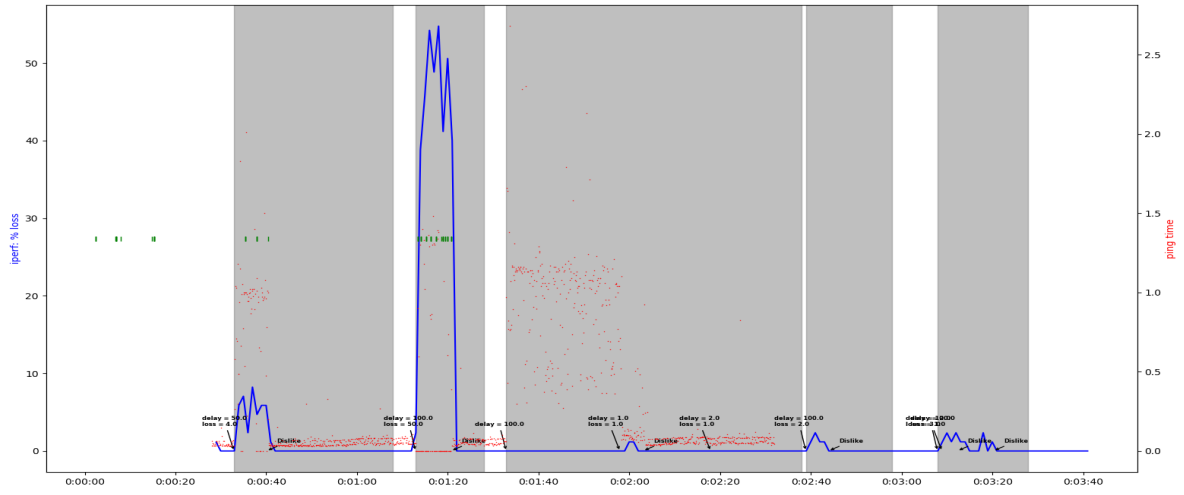


Figure 26.c: Plotted QoE Rerouting Results decisions for participant 3

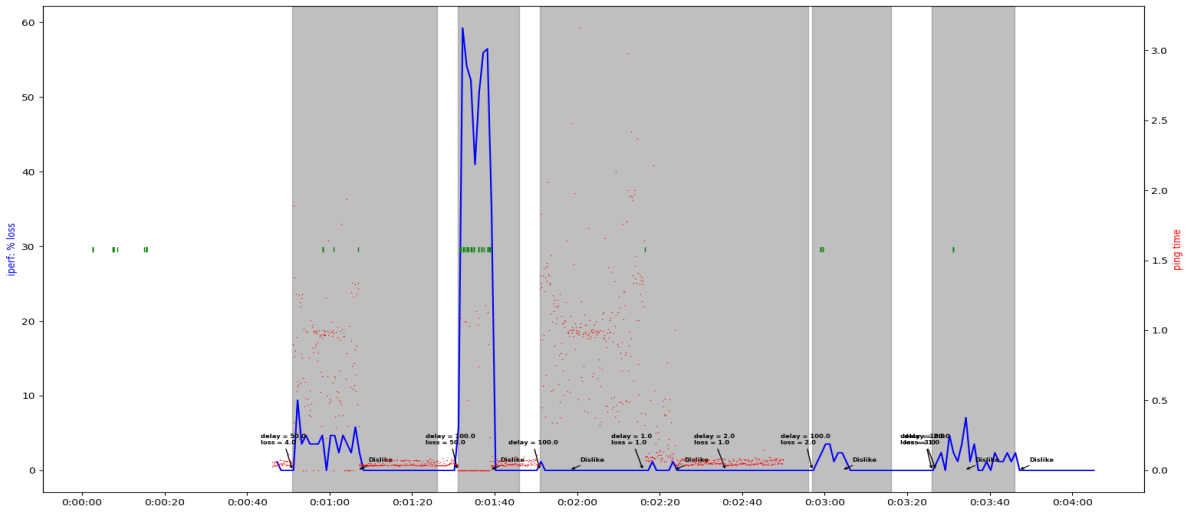


Figure 26.d: Plotted QoE Rerouting Results decisions for participant 4

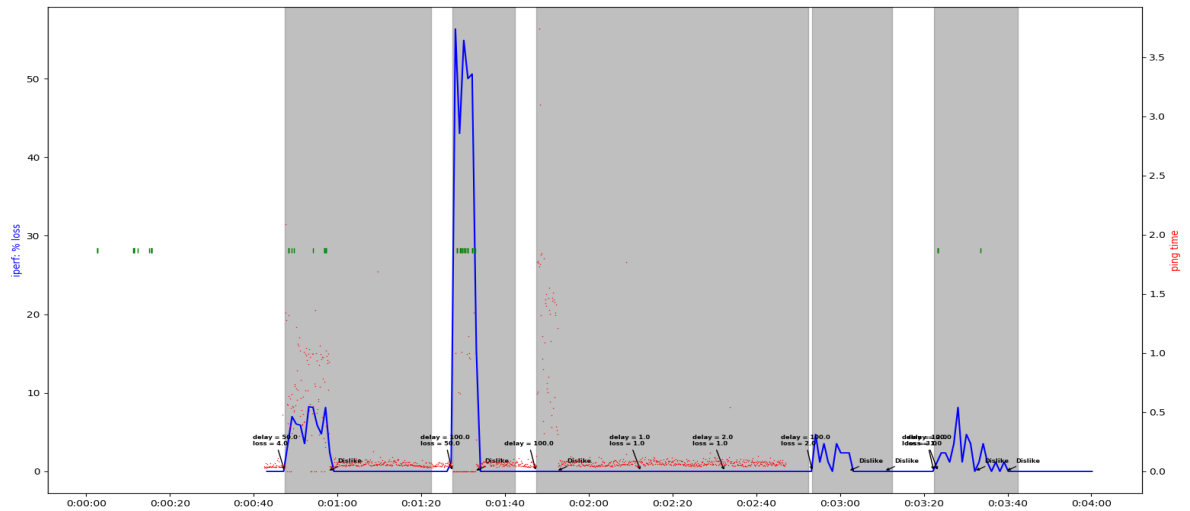


Figure 26.e: Plotted QoE Rerouting Results decisions for participants 5

By analysing the above results, streaming QoE enhanced after every participant feedback click, where it was noted the blue line indicates no loss on the streaming link after clicks provided, mean streaming was enhanced and no more delay. It was noticed that when applying a typical delay without loss, users don't sense any quality degradation and they never hit the feedback button, due to the buffering capabilities of the VLC player. With gradually applying small loss% with minimum delay, the QoE was affected immediately and accordingly the feedback for reroute was received. By applying degrading changes on two paths out of three, it was found that ONOS kept search for the alternative routes with every click until the QoE is acceptable for the user.

With all participants it was found that ONOS managed to reroute traffic based on user's feedback and the quality of the stream was enhanced after controller's enhancement action. Among all participants, ONOS reaction and response was consistent and timely performed to do rerouting decision. It took ONOS between 10-15 ms to construct a new route (intent) as a new streaming path and between 15-20 ms to reroute traffic. It was proved that user's interactive feedback can be taken into consideration during streaming session and this feedback can be fed to the SDN controller to alert of an existing issue and that a corrective action is required. It was proven that rerouting decision can be made on spot and quality can be enhanced based on external user feedback.

6.2 Crowdsourcing Human Paired Comparison Results and Analysis

For HPC experiment, there were 2430 paired comparison questions with a total of 259 participants. Only 243 participants were successful and 16 failed. Ranks were computed based on Crowd Bradley-Terry model [49]. Figure 27.a, b, c, and d show participant's

ratings scores for Video 1, Video 2, Video 3 and Video 4 for each protocol and scenario of events respectively.

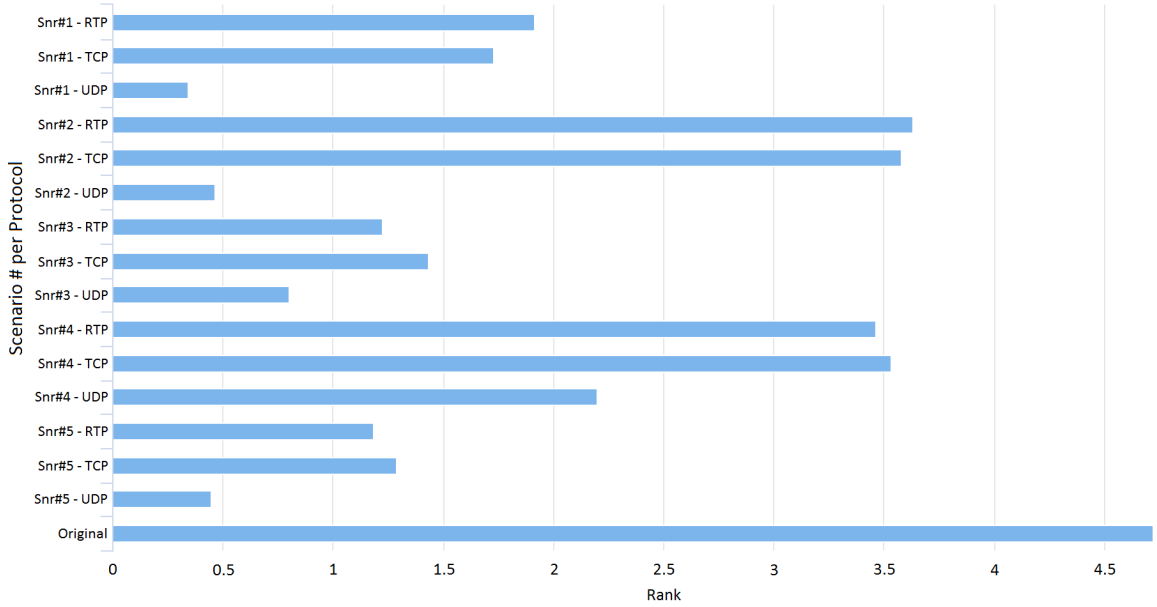


Figure 27.a: HPC QoE Rating results for Video1 using Crowd Bradley-Terry Model.

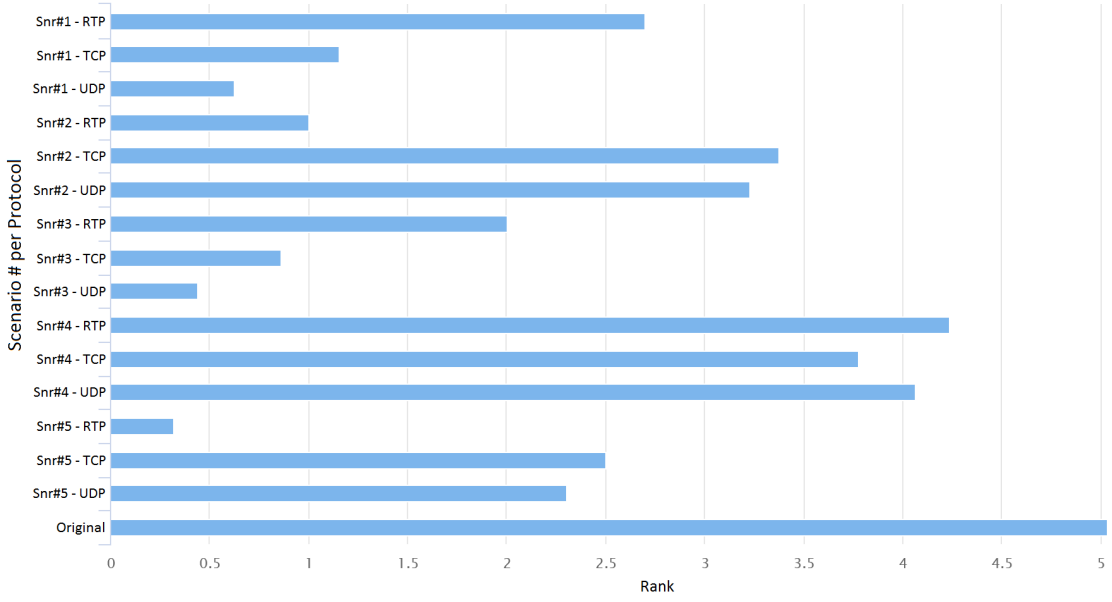


Figure 27.b: HPC QoE Rating results for Video 2 using Crowd Bradley-Terry Model.

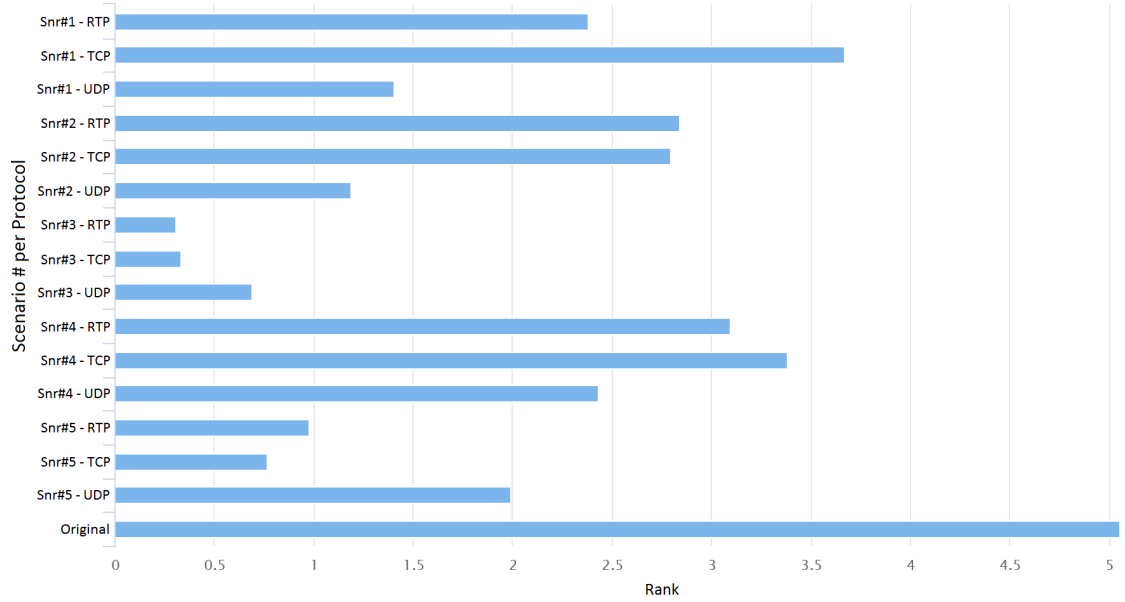


Figure 27.c: HPC QoE Rating results for Video 3 using Crowd Bradley-Terry Model.

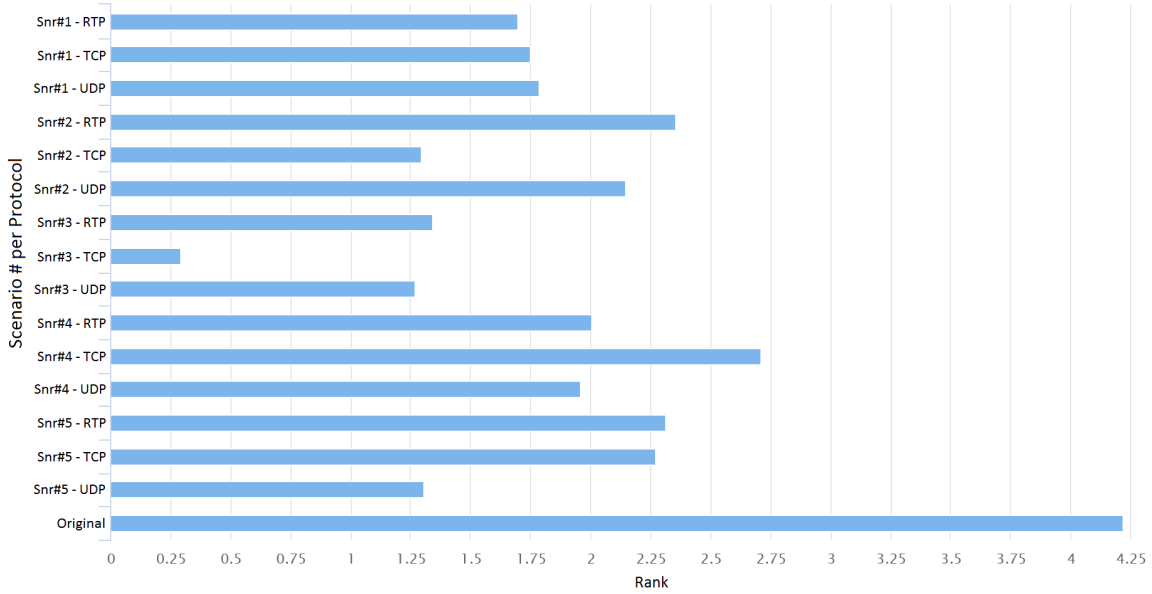


Figure 27.d: HPC QoE Rating results for Video 4 using Crowd Bradley-Terry Model.

By using MSU-VQM, we have computed VQM objective values of QoE perceived for processed videos used in HPC experiment against the original video. Table 7 shows VQM rounded values for each resulted video.

Table 7. VQM analysis output for HPC experiment

Scenario	1	2	3	4	5
Video 1					
RTP	3.45	3.15	5.29	3.18	3.86
TCP	3.51	3.15	3.45	3.15	4.32
UDP	7.28	8.52	5.35	3.68	4.28
Video 2					
RTP	4.31	4.15	4.39	4.10	3.94
TCP	4.33	4.08	4.36	4.18	4.37
UDP	4.10	4.10	4.11	4.09	4.3
Video 3					
RTP	2.15	1.82	2.46	2.39	2.37
TCP	2.71	1.63	3.05	1.84	2.74
UDP	2.49	2.33	2.20	2.26	2.47
Video 4					
RTP	2.53	2.42	2.45	2.49	2.64
TCP	2.55	2.09	2.9	2.96	2.57
UDP	2.01	2.6	2.09	2.18	2.61

By comparing subjective and objective results, we found that HPC and VQM results are consistent. Where mostly the highly ranked protocol in HPC had the best VQM lowest value across the three protocols per video. On average 3-4 out of 5 rankings are similar to VQM calculated values. Table 8 shows the highest ranked protocol in HPC against best VQM value for each Video across different scenarios.

Table 8. HPC/VQM Results analysis by highlighting protocols.

	Video 1		Video 2		Video 3		Video 4	
	VQM	**HPC**	**VQM**	**HPC**	**VQM**	**HPC**	**VQM**	**HPC**
Scenario 1	RTP/TCP	RTP	UDP	RTP	RTP	TCP	UDP	UDP
Scenario 2	RTP/TCP	RTP	TCP	TCP	TCP/ RTP	RTP	TCP/ RTP	RTP
Scenario 3	TCP	TCP	UDP	RTP	UDP	UDP	UDP/RTP	RTP/UDP
Scenario 4	TCP/RTP	TCP/RTP	RTP	RTP	TCP	TCP	UDP	TCP
Scenario 5	RTP	TCP/RTP	RTP	TCP	RTP	UDP	TCP	TCP/RTP

For video1, scenario1 showed only 0.07 difference values between RTP and TCP in VQM, however in HPC the ranking difference was high 1.91 / 0.34 respectively. In scenario 2, although both VQM and HPC protocols selection matched, it was noted that UDP VQM value was very high compared to other protocol results, and for RTP and TCP

ranking was only 0.06 difference 3.63/3.57 and shown nearly same VQM values. TCP and RTP values and ranks were consistent in both VQM and HPC in scenario 3 and 4. In scenario 5, VQM best value was RTP, where TCP had the best rating score with a very close rating to RTP with only 0.1 difference. It was noticed that RTP and TCP has highest VQM values across all scenarios, with a very close rating difference to each. UDP had the lowest rating among all scenarios in video 1. For Video 2, scenario 1 shown that RTP is significantly better than UDP with rating 2.69/0.6 respectively in HPC, however VQM values between the two protocols was 0.21 variance. Although VQM and HPC best score/ rating protocol is same in scenario 2, the 3 protocols values in VQM analysis showing a very small variance of 0.01, on the other hand there is a huge gap in ranking between TCP and RTP where UDP was closer in ranking to TCP. The ranking gap between RTP and the other two protocols with very high in HPC for scenario 3, however this gap was not present in VQM values among the 3 protocols. For Video3, RTP had best VQM value and the TCP has the best ranking in HPC for scenario 1. TCP was having the best VQM values followed by RTP, it should be noted that overall ranking for the three protocols in HPC was very low for scenario 3. UDP is ranked lowest among 3 scenarios out of 5 in video 3, and in VQM it had high values in only 2 scenarios. For Video 4, TCP rankings in HPC are consistent with VQM values. Figure 28 shows VQM values vs. HPC rating scores for each video.

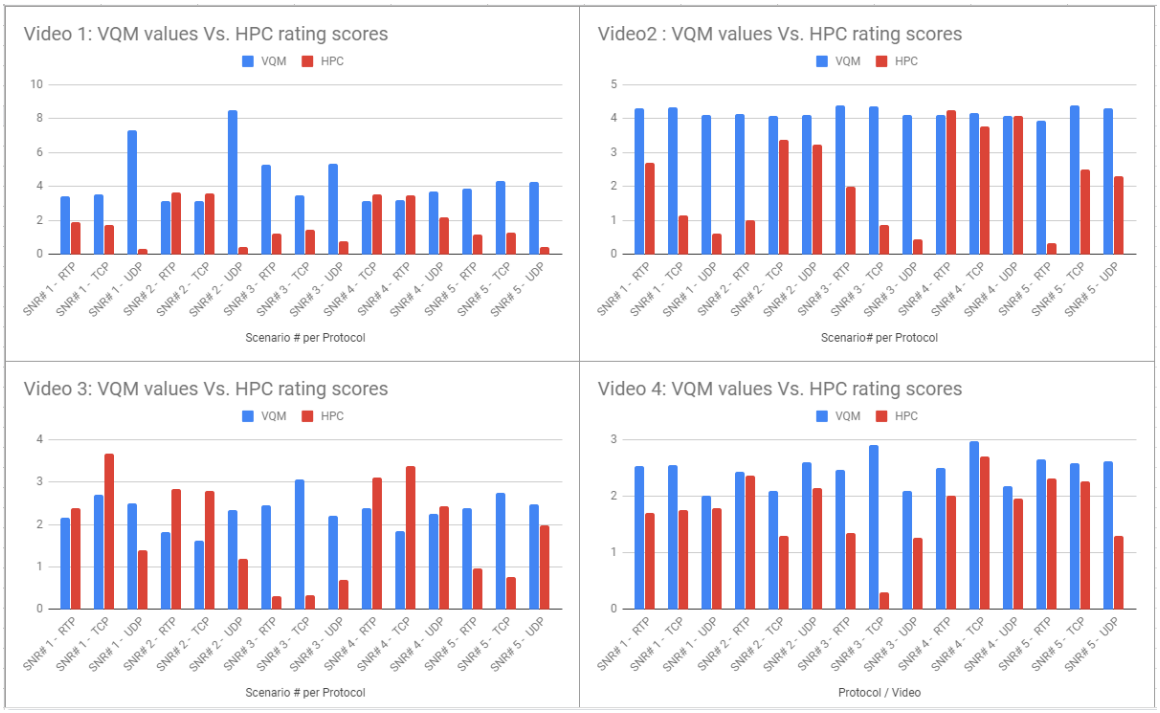


Figure 28: VQM values vs. HPC rating scores for each video.

Among all rated videos with different changing event scenarios in HPC, RTP and TCP were alternating with highest users score ratings. Both showed consistency where they had mostly the higher rating probability and UDP usually came last. Same applies to VQM resulted values, RTP and TCP had close VQM values. User perceptions ratings was consistent to what was produced by VQM values, only minor differences were noticed.

6.3 VQM QoE Results and Analysis

As mentioned earlier, we have run 10 different scenarios with 5 different protocols and we have ran objective measures on the resulted videos to perform analysis on which protocols preforms better for an objective point. Below figures are Video1 and 3 results of running VQM and SSIM values against 150 videos, using 3 different videos where Legacy UPD represented as "Red", RTP as "Orange", SCTP as "Green", RTP over TCP as "Purple" and RTP over UDP "Blue". Results for video 2 can be found in Appendix B.

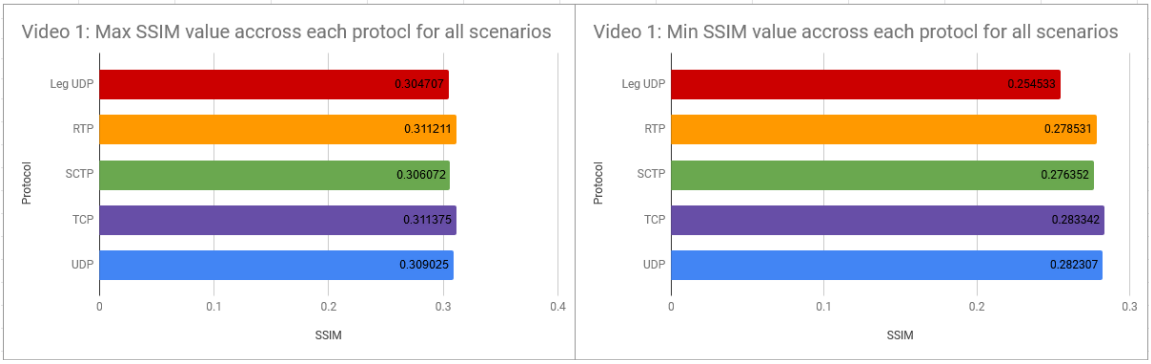


Figure 29.a: Video1 SSIM minimum and maximum values for each protocol across all 10 scenarios.

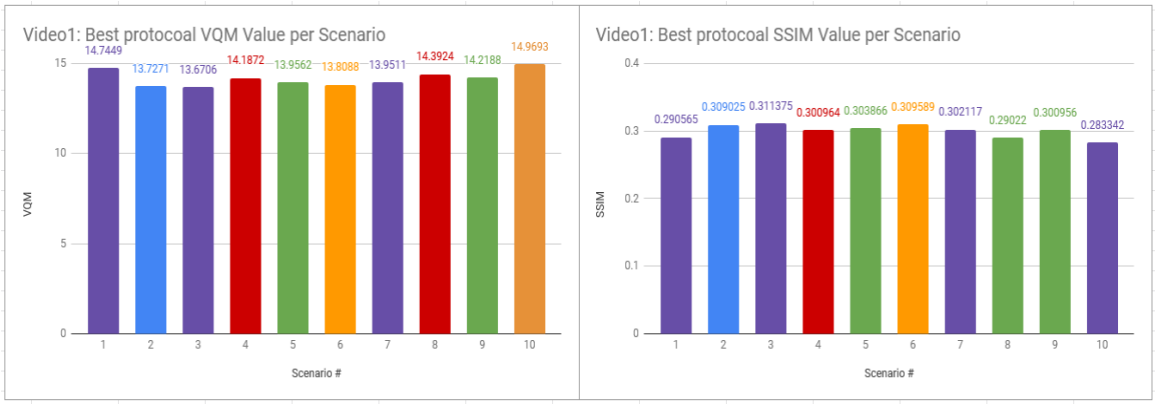


Figure 29.b: Video1 best protocol VQM value (lowest) and SSIM value (highest) for each scenario.

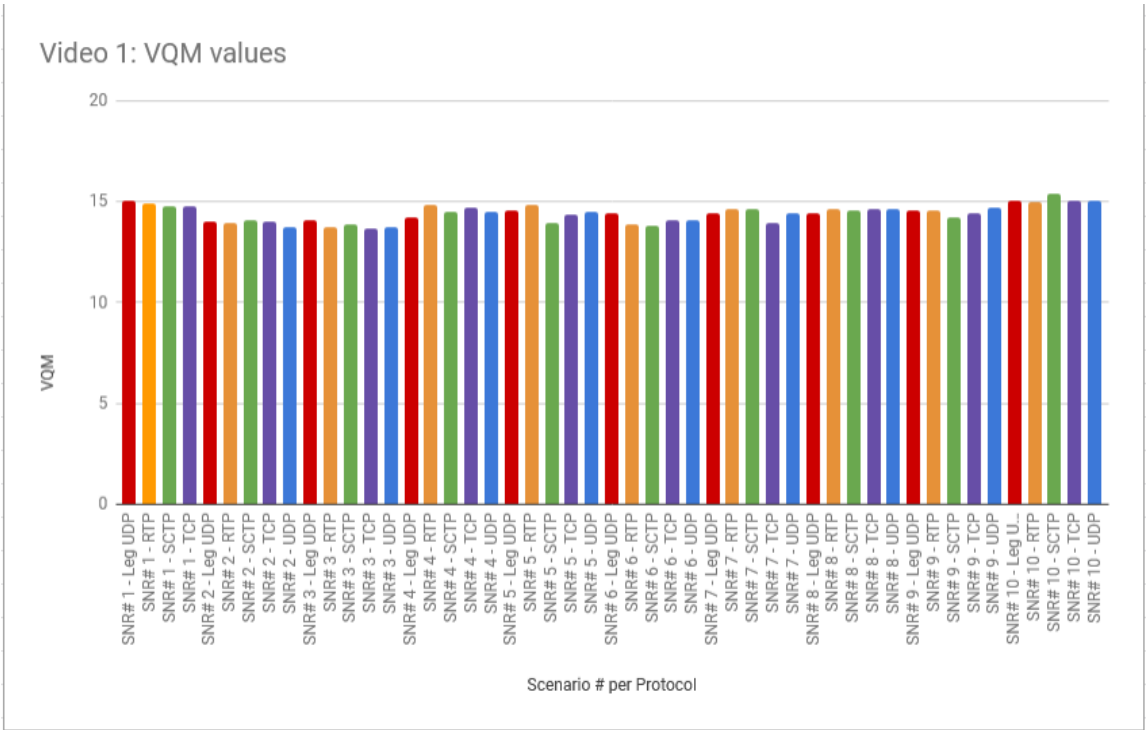


Figure 29.c: Video1 calculated VQM values.

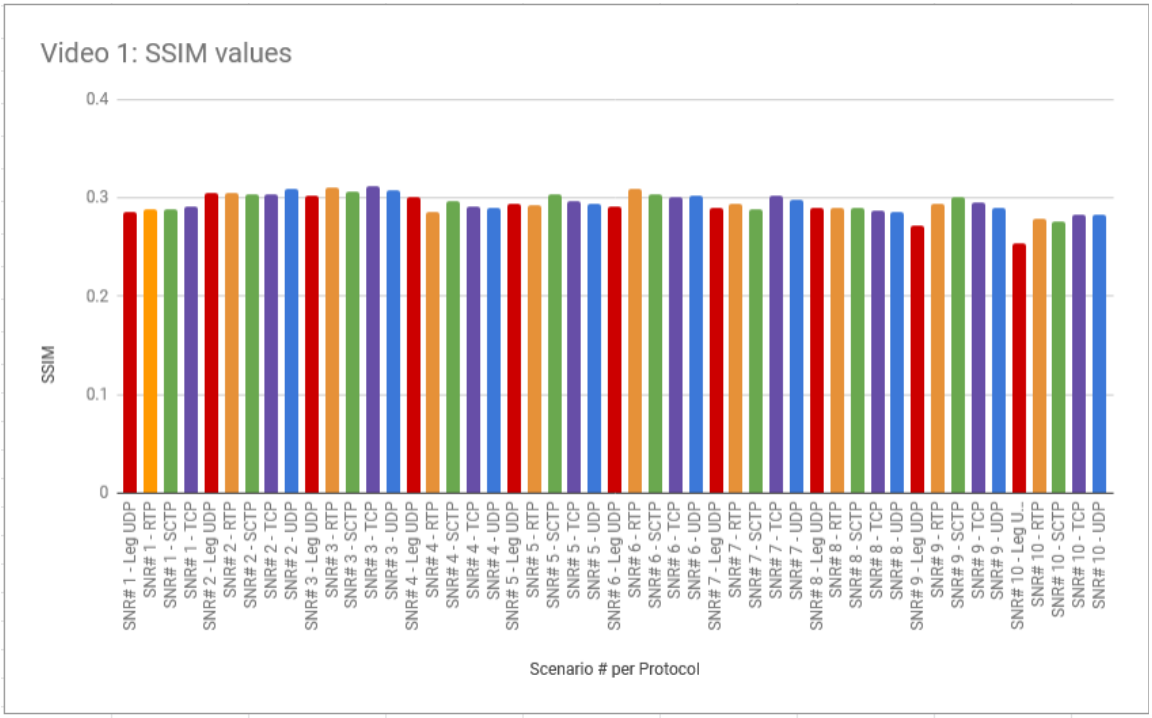


Figure 29.d: Video1 calculated SSIM values.

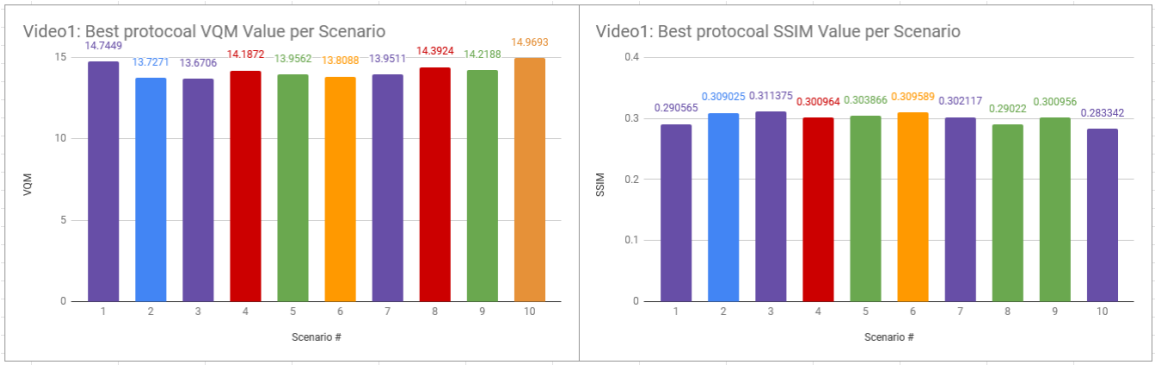


Figure 29.e: Video3 SSIM minimum and maximum values for each protocol across all 10 scenarios.

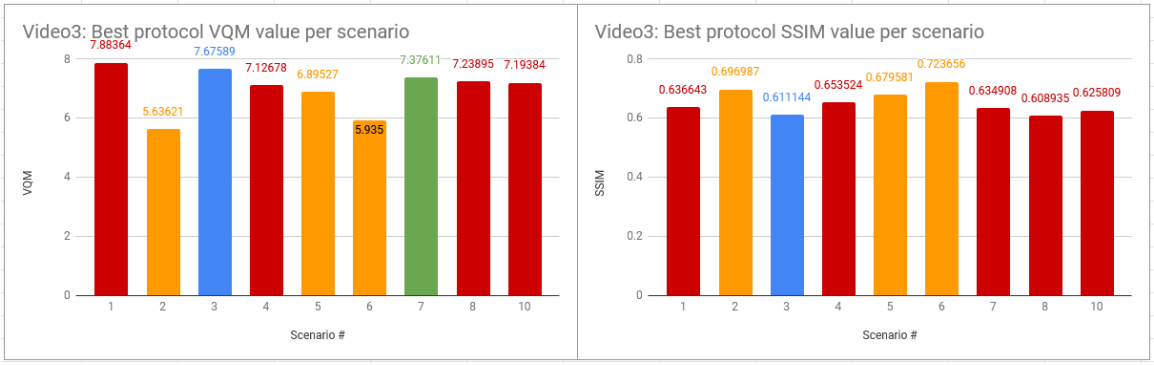


Figure 29.f: Video3 best protocol VQM value (lowest) and SSIM value (highest) for each scenario.

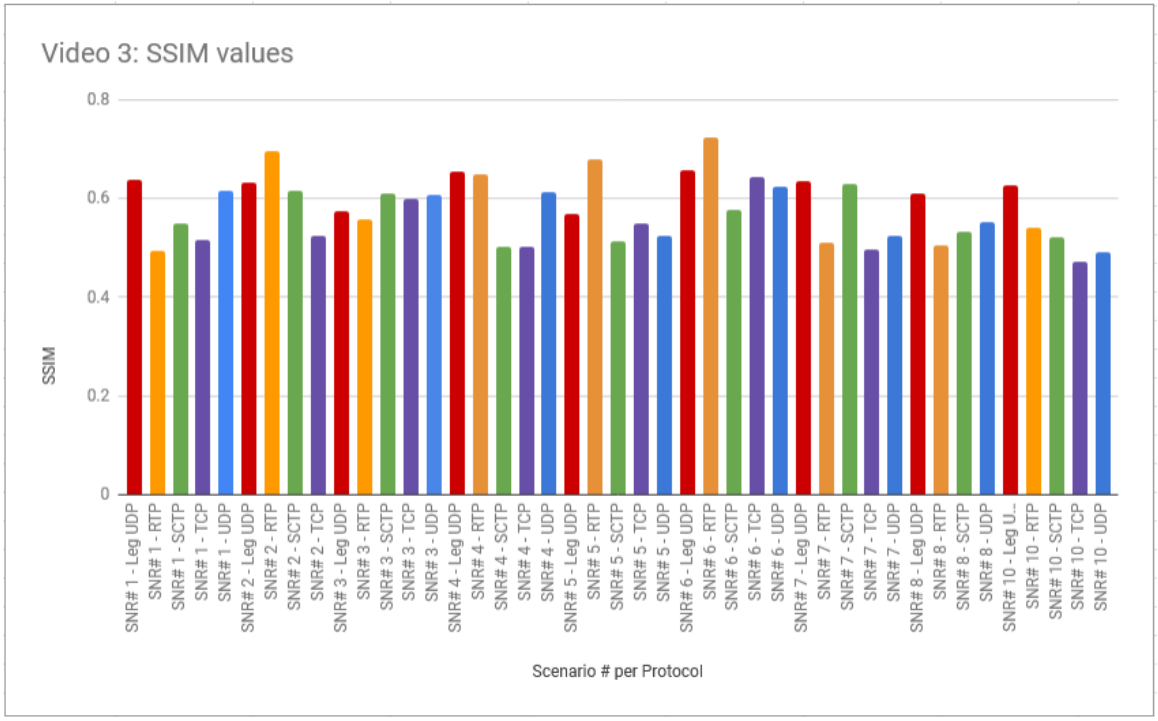


Figure 29.g: Video3 calculated SSIM values.

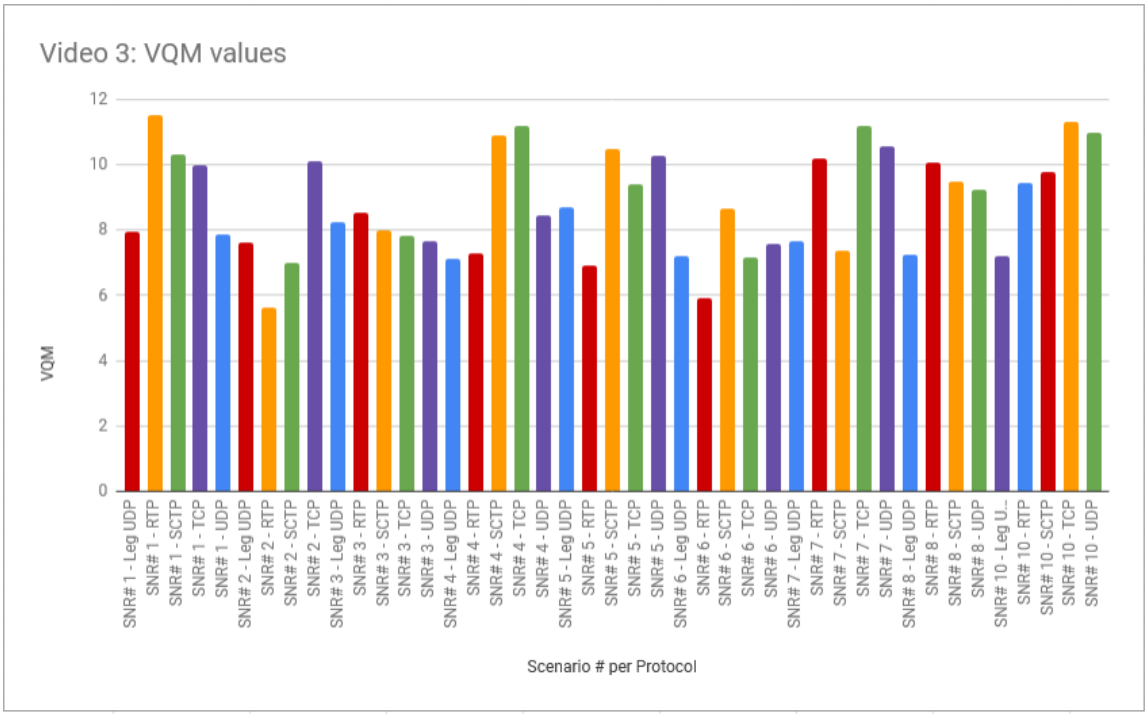


Figure 29.h: Video3 calculated VQM values.

It was found that best performed protocol VQM calculated values were consistent with SSIM values for all Videos with very minor discrepancies between the two measures,

where protocols measured values a tiny fraction difference. It was notices that Video two measured values for both VQM and SSIM were too close to each other making no specific protocol to perform better than the others. For Video 1, RTP over TCP performed better with different scenarios than other protocols. It was found that RTP and Leg UDP performed best among all protocols with different scenarios for Video 2 and 3. Though, Leg UDP performed worst with previous HPC experiments and was rarely rated best among participants. By analysing objective measured results, it was shown that RTP and RTP over TCP are showing better performance to accommodate unexpected changes in the service path. The choice of these two types of protocols were consistent to protocols highly rated in HPC experiment, however it should be noted UDP was not one of best rated protocols and human participants sometimes have different perspective than what measured objectively.

Chapter 7

7 Conclusions

I proposed a real-time QoE-aware crowdsourcing model. The proposed model is based on a combination of QoE measurement application and QoS quality parameters that accommodate a variety of different streaming protocols. It emphasizes the dependability between QoE and QoS and how the overall user's QoE perspective can be affected. We have analyzed how dynamic changes of events can affect the performance of different streaming protocols, and accordingly perceived quality. We have compared objective and subjective results and found that both results are consistent, where RTP and RTP over TCP are more suitable to adjust for dynamic changes over UDP. We have proved that real-time feedback can be captured and sent to SD-WAN controller to alert of possible issues. By applying QoE feedback, we showed that controller can reroute traffic during streaming session. Our aim is to prove that real-time QoE feedback can enhance cloud-based services and can adjust services quality based on real-time active participants' interaction.

It was found that user interactive interaction can be captured during a live streaming session and their feedback can be fed on-spot to allow corrective actions during a current streaming session. After testing several protocols with multiple network dynamic changes, RTP was consistent to accommodate unexpected dynamic changes by objective and subjective measures and performed best with different network SLA.

For future work, we will apply the QoE-aware crowdsourcing rerouting model on a crowdsourcing platform such as MTurk to test scalability and user interaction in a wider demography with an open environment. Where participants provide real-time feedback, these feedbacks will be compared to predefined dynamic changes to identify if participants

will be able to capture all degradation events or whether not all degradations event combinations are noticeable to the participants. We will feed in these timestamped feedbacks to SD-WAN controller in-order to detect problems in the service path and to take corrective action by making changes to the virtual topology of the content delivery network, reassigning the users, or rerouting the traffic.

By using Artificial intelligence (AI), it is possible to learn feedback patterns received by external user participation and an AI algorithm can decide if rerouting is required or not, where we have a threshold for number of clicks. This AI algorithm over time can determine this threshold dynamically. We will look into algorithms that can find optimal route and allow SND environment optimization, taking into consideration QoS SLA and minimum QoE requirements. Where these algorithms take decisions based on incoming QoE feedback along with underlying changing network condition to ensure that the SDN controller will always find the best path and hence the best QoE. Intent-based programming is evolving currently and some new research was immerged to leverage the Intent Framework of SDN controller to optimize re-routing instead of traditional MPLS.

8 Appendices

Appendix A.

1.1. QoE Feedback Rating Application

Feedback.py

```
import os
import time
import sys
import threading
import random
import BaseHTTPServer
from logging import getLogger

IFWD_APPID = "org.onosproject.ifwd"
log = getLogger(__name__)


""" Determenistically concatenate two hostids """
def ifwd_intent_key(host1, host2):
    if host1 < host2:
        return host1 + host2
    else:
        return host2 + host1

def hash_path(path):
    from sdn_demo.namesgenerator import get_random_name
    r = random.Random()
    h = "-".join(sorted(path))
    r.seed(h)
    return get_random_name(r)

class FeedbackHandler(BaseHTTPServer.BaseHTTPRequestHandler):
    def respond(self, code, msg):
        if code == 200:
            self.send_response(code)
            self.end_headers()
            self.wfile.write(msg)
        else:
            self.send_error(code, msg)

    def do_POST(self):
        onosnb = self.server.onosnb
        vlc_ip = self.server.vlc_ip
        user_ip = self.client_address[0]
```

```
        if self.path == "/reroute":
            try:
                vlc_mac, user_mac = onosnb.hostid_by_ip(vlc_ip, user_ip)
                if user_mac is None:
                    return self.respond(403, "Unrecognized IP")
                key = ifwd_intent_key(vlc_mac, user_mac)
                before, after = onosnb.reroute_intent(IFWD_APPID, key)
                log.info("New route between %s-%s constructed: %s => %s",
                         vlc_ip, user_ip,
                         "-".join(sorted(before)),
                         "-".join(sorted(after)))
            except Exception as e:
                self.respond(500, "ONOS REST Error")
                raise

            msg = "%s => %s" % (hash_path(before), hash_path(after))
            self.respond(200, msg)
        else:
            self.respond(404, "Try /reroute")

    def log_message(self, format, *args):
        log = self.server.log
        log.write("%s - - [%s] %s\n" %
                  (self.client_address[0],
                   self.log_date_time_string(),
                   format%args))
        log.flush()

class HttpServer(BaseHTTPServer.HTTPServer):
    def __init__(self, bind, vlc_ip, onosnb, log):
        self.vlc_ip = vlc_ip
        self.onosnb = onosnb
        self.log = log
        BaseHTTPServer.HTTPServer.__init__(self, bind, FeedbackHandler)

class HttpServerThread(object):
    def __init__(self, config, net, roles, onosnb):
        ip = net.getNodeByName(roles['world']).IP()
        port = config.getint('demo', 'feedback_port')
        vlc_ip = net.getNodeByName(roles['server']).IP()
        self.logfile = os.path.join(config.get('demo', 'workdir'), "http.log")
        self.log = open(self.logfile, 'w')
        self.httpd = HttpServer((ip, port), vlc_ip, onosnb, self.log)
        self.thread = None

    def start(self):
        self.thread = threading.Thread(target=self.httpd.serve_forever)
        self.thread.start()

    def stop(self):
        self.httpd.shutdown()
        self.httpd.server_close()
        self.log.close()


if __name__ == '__main__':
    import sys; sys.path.append('..')
    from sdn_demo.controller import OnosClient
    c = OnosClient("http://172.17.0.2:8181/onos/v1", "onos", "rocks")
    s = HttpServer(('0.0.0.0', 5000), '10.0.0.1', c)
```

Sdn_button.lua: VLC Feedback button embedding using .lua extension

```
local mtitle = "[sdn_menu]"
local osdchan

function log(m)
    vlc.msg.info(mtitle.." "..m)
end

function alert(m)
    vlc.osd.message(m, osdchan, "top", 3000000)
end

function descriptor()
    return {
        title = "SDN reroute button",
        version = "0.1",
        author = "",
        url = 'http://www.example.org/',
        shortdesc = "SDN";
        description = "Reroutes RTP stream over backup links",
        capabilities = {"menu"}
    }
end

function activate()
    osdchan = vlc.osd.channel_register()
    log("Welcome! OSD: "..osdchan)
end

function deactivate()
    vlc.osd.channel_clear(osdchan)
    osdchan = nil
    log("Bye bye!")
end
function menu()
    return {
        "Dislike",
    }
end
function trigger_menu(id)
    if id == 1 then
        request_rerouting()
    end
    collectgarbage()
end
function meta_changed()
    return false
end

function request_rerouting()
    log("Re-routing requested")

    local request = ""
    local host = "10.0.0.3"
    local path = "/reroute"
    local header = {
      "POST "..path.." HTTP/1.0",
```

```
      "Host: "..host,
      "User-Agent: VLC",
      "Content-Length: "..string.len(request),
      "",
      ""
    }
    request = table.concat(header, "\r\n")..request

    alert("Searching for solution...")
    local status, response = http_req(host, 5000, request)
    alert("Done. Details: "..response)

    log("Response: "..status.." "..response)
end

-----------------------------------------------------------------------------
-----------------------------------
-- This code is taken from built-in VLSub extension: https://git.io/fjWnV
-----------------------------------------------------------------------------
-----------------------------------
function http_req(host, port, request)
    local fd = vlc.net.connect_tcp(host, port)
    if not fd then return false end
    local pollfds = [52]

    pollfds[fd] = vlc.net.POLLIN
    vlc.net.send(fd, request)
    vlc.net.poll(pollfds)

    local chunk = vlc.net.recv(fd, 2048)
    local response = ""
    local headerStr, header, body
    local contentLength, status
    local pct = 0

    while chunk do
        response = response..chunk
        if not header then
            headerStr, body = response:match("(.-\r?\n)\r?\n(.*)")
            if headerStr then
                response = body
                header = parse_header(headerStr)
                contentLength = tonumber(header["Content-Length"])
                status = tonumber(header["statuscode"])
            end
        end

        if contentLength then
            bodyLenght = #response
            pct = bodyLenght / contentLength * 100
            -- setMessage(openSub.actionLabel..": "..progressBarContent(pct))
            if bodyLenght >= contentLength then
                break
            end
        end

        vlc.net.poll(pollfds)
        chunk = vlc.net.recv(fd, 1024)
```

```
    end

    vlc.net.close(fd)

    if status == 301
    and header["Location"] then
        local host, path = parse_url(trim(header["Location"]))
        request = request
            :gsub("^([^%s]+ )([^%s]+)", "%1"..path)
            :gsub("(Host: )([^\n]*)", "%1"..host)

        return http_req(host, port, request)
    end

    return status, response
end

function parse_url(url)
    local url_parsed = vlc.strings.url_parse(url)
    return  url_parsed["host"],
            url_parsed["path"],
            url_parsed["option"]
end

function trim(str)
    if not str then return "" end
    return string.gsub(str, "^[\r\n%s]*(.-)[\r\n%s]*$", "%1")
end

function parse_header(data)
    local header = [52]

    for name, s, val in string.gmatch(
        data,
        "([^%s:]+)(:?)%s([^\n]+)\r?\n")
    do
        if s == "" then
            header['statuscode'] = tonumber(string.sub(val, 1 , 3))
        else
            header[name] = val
        end
    end
    return header
end
```

1.2. QoE-Aware Rerouting Application

Controller.py

```
import time
import itertools
import random
import collections
from copy import deepcopy

import requests
from requests.utils import quote

"""
  A wrapper for ONOS REST API.
  Docs:
    Jagger: http://172.17.0.2:8181/onos/v1/docs/
    Wiki: https://wiki.onosproject.org/display/ONOS/Appendix+B%3A+REST+API
"""
class OnosClient(object):
    def __init__(self, baseurl, user, password, timeout=1.0):
        self.baseurl = baseurl
        self.auth = (user, password)
        self.timeout = timeout

    def get(self, url, check=True, **kwargs):
        kwargs.setdefault('auth', self.auth)
        kwargs.setdefault('timeout', self.timeout)
        resp = requests.get(self.baseurl + url, **kwargs)
        if check:
            resp.raise_for_status()
        return resp.json()

    def post(self, url, json=None, **kwargs):
        kwargs.setdefault('auth', self.auth)
        kwargs.setdefault('timeout', self.timeout)
        resp = requests.post(self.baseurl + url, json=json, **kwargs)
        resp.raise_for_status()
        return resp

    def delete(self, url, json=None, **kwargs):
        kwargs.setdefault('auth', self.auth)
        kwargs.setdefault('timeout', self.timeout)
        resp = requests.delete(self.baseurl + url, json=json, **kwargs)
        resp.raise_for_status()
        return resp.text

    """ Activate/deactivate required applications if needed """
    def ensure_apps(self, ifwd_bundle=None):
        self.post("/applications/org.onosproject.openflow/active", timeout=10)
        self.post("/applications/org.onosproject.proxyarp/active", timeout=10)
        self.delete("/applications/org.onosproject.fwd/active", timeout=10)
        if ifwd_bundle is None:
            return

        ifwd = self.get("/applications/org.onosproject.ifwd", check=False)
        if ifwd.has_key('id'):
            return
        with open(ifwd_bundle, 'rb') as f:
            self.post("/applications", data=f.read(), timeout=10)

    def ifwd_active(self, flag):
        if flag:
            self.post("/applications/org.onosproject.ifwd/active", timeout=5)
```

```
        else:
            self.delete("/applications/org.onosproject.ifwd/active", timeout=5)

    """ Find a "MAC/Vlan" pair by IPs """
    def hostid_by_ip(self, ip1, ip2):
        mac1 = None
        mac2 = None
        for host in self.get('/hosts')['hosts']:
            if ip1 in host['ipAddresses']:
                mac1 = host['id']
            if ip2 in host['ipAddresses']:
                mac2 = host['id']
        return mac1, mac2

    """ Query a set of devices that steers an intent traffic at the moment """
    def devices_touched_by_intent(self, app, quoted_intent_key):
        url = '/intents/installables/%s/%s' % (app, quoted_intent_key)
        paths = self.get(url)['installables']
        devices = set()
        for p in paths:
            typename = p['type']
            links = p['resources']
            if typename != 'FlowRuleIntent':
                raise RuntimeError("Unsupported installable type: %s" % typename)
            devices_along_path(links, devices)
        return devices

    """ Query a flat list of unique disjoint pathsets between 2 points """
    def disjoint_pathsets(self, p1, p2):
        url = '/paths/%s/%s/disjoint' % (quote(p1, safe=''), quote(p2, safe=''))
        disjoints = self.get(url)['paths']
        tuplify = lambda d: (d['primary']['links'], d['backup']['links'])
        flatpaths = itertools.chain(*map(tuplify, disjoints))
        return map(devices_along_path, flatpaths)

    """" Reroute intent over random alternative path """
    def reroute_intent(self, app, key):
        key = quote(key, safe='')

        intent = self.get('/intents/%s/%s' % (app, key))
        if intent['type'] != 'HostToHostIntent':
            raise RuntimeError("Only Host2Host intent supported")

        for cons in intent['constraints']:
            if cons['type'] == 'ObstacleConstraint':
                obstacles = cons['obstacles']
                break
        else:
            obstacles = []

        curpath = self.devices_touched_by_intent(app, key)
        disjoint = self.disjoint_pathsets(intent['one'], intent['two'])
        choices = compute_obstacle_choices(curpath, disjoint, obstacles)

        if len(choices) > 0:
            obstacle = random.choice(choices)
            new_intent = toggle_obstacle(intent, obstacle)
            resp = self.post('/intents', json=new_intent)
            loc = resp.headers.get('location')
            newid = int(loc.rsplit('/', 1)[-1])
```

```
            for i in range(10):
                intent = self.get('/intents/%s/%s' % (app, key), check=False)
                curid = int(intent.get('id', '0'), 16)
                if newid == curid:
                    break
                time.sleep(0.5)

            newpath = self.devices_touched_by_intent(app, key)
            return curpath, newpath
        else:
            return curpath, curpath

""" Compute possible choices for a new obstacle """
def compute_obstacle_choices(curpath, allpaths, exclude):
    uniqpath = unique_paths([curpath] + list(allpaths))
    noderank = collections.Counter(itertools.chain(*uniqpath))

    delete_marker = list(curpath)[0] * 0
    choices = [delete_marker] if len(exclude) else []

    for device in curpath:
        count = noderank[device]
        if count == len(uniqpath):
            # The device cannot be avoided
            continue
        if device in exclude:
            # Don't repeat old choice
            continue
        # Give more chances to disjoint paths
        choices += [device] * count
    return choices

""" Compute a unique set of pathsets """
def unique_paths(paths):
    uniq = {}
    keyfn = lambda ps: "-".join(map(str, sorted(ps)))
    for path in paths:
        uniq.setdefault(keyfn(path), path)
    return uniq.values()

""" Return a set of network devices along path """
def devices_along_path(path, inplace=None):
    d = set() if inplace is None else inplace
    for link in path:
        d.add(link['src'].get('device'))
        d.add(link['dst'].get('device'))
    d.discard(None)
    return d
```

```
    """ Replaces old intent obstacles with new one """
    def toggle_obstacle(intent, obstacle):
        intent = deepcopy(intent)
        intent.pop('id', None)
        intent.pop('state', None)

        constraints = intent['constraints']
        if obstacle*0 != obstacle:
            for cons in constraints:
                if cons['type'] == 'ObstacleConstraint':
                    cons['obstacles'] = [obstacle]
                    break
            else:
                intent['constraints'].append({
                    'type': 'ObstacleConstraint',
                    'obstacles': [obstacle]
                })
        else:
            intent['constraints'][:] = [c for c in constraints if c['type'] != 'ObstacleConstraint']
        return intent

    # For debugging purposes
    if __name__ == '__main__':
        c = OnosClient("http://172.17.0.2:8181/onos/v1", "onos", "rocks")
```

Topo.py: topology creation and build

```
from mininet.topo import Topo

class MultipathTopo(Topo):
    """Topology with two hosts and multiple disjoint paths between them"""

    def build(self, *hops):
        assert(all([x >= 0 for x in hops]))

        s1 = self.addSwitch('s1')
        h1 = self.addHost('h1', mac='00:00:00:00:00:11')
        self.addLink(s1, h1)

        s2 = self.addSwitch('s2')
        h2 = self.addHost('h2', mac='00:00:00:00:00:22')
        self.addLink(s2, h2)

        h3 = self.addHost('h3')
        self.addLink(s2, h3)

        for p, nhops in enumerate(hops):
            if nhops == 0:
                self.addLink(s1, s2)
                continue

            dpid = (p + 1) * 0x100
            sadj = self.addSwitch("s%d" % dpid)
            self.addLink(s1, sadj)

            for h in xrange(1, nhops):
                dpid += 1
                s = self.addSwitch("s%d" % dpid)
                self.addLink(sadj, s)
                sadj = s

            self.addLink(sadj, s2)
```

minievent.json (delay-loss.json)

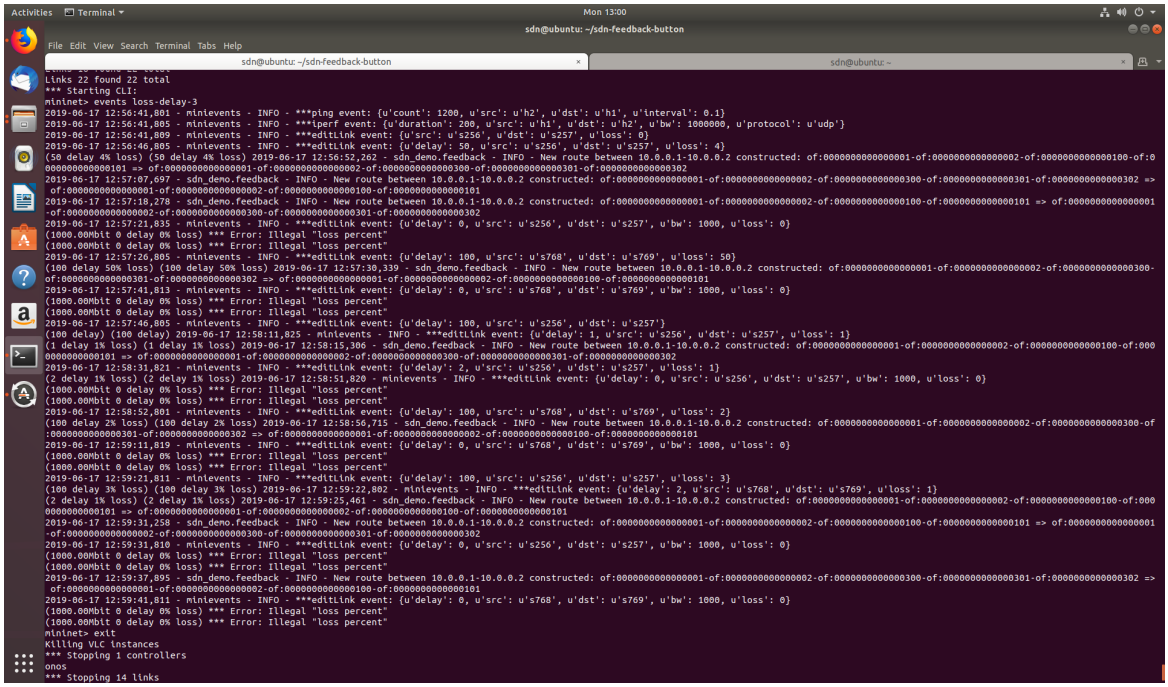

```
{
    "time": 0,
    "type": "ping",
    "params": {
      "src": "h2",
      "dst": "h1",
      "interval": 0.1,
      "count": 1200
    }
 },
 {
    "time": 0,
    "type": "iperf",
    "params": {
      "duration": 200,
      "src": "h1",
      "dst": "h2",
      "protocol": "udp",
      "bw": 1000000
    }
 },
 {
    "time": 5,
    "type": "editLink",
    "params": {
      "src": "s256",
      "dst": "s257",
      "loss": 4,
      "delay": 50
    }
  },
  {
    "time": 40,
    "type": "editLink",
    "params": {
      "src": "s256",
      "dst": "s257",
      "loss": 0,
      "delay": 0,
      "bw": 1000
    }
  },
  {
    "time": 45,
    "type": "editLink",
    "params": {
      "src": "s512",
      "dst": "s513",
      "loss": 50,
      "delay": 100
    }
  },
  {
    "time": 60,
    "type": "editLink",
    "params": {
```

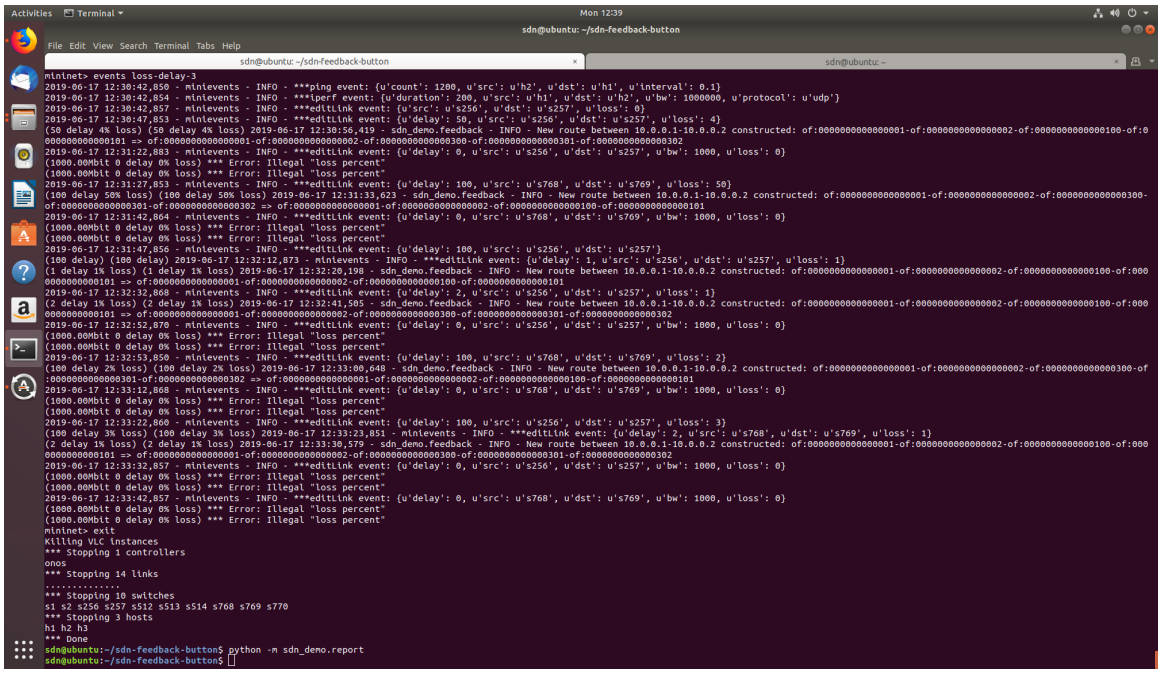
```
      "src": "s512",
      "dst": "s513",
      "loss": 0,
      "delay": 0,
      "bw": 1000
    }
  },
{
    "time": 65,
    "type": "editLink",
    "params": {
      "src": "s256",
      "dst": "s257",
      "delay": 100
    }
  },
  {
    "time": 90,
    "type": "editLink",
    "params": {
      "src": "s256",
      "dst": "s257",
      "loss": 1,
      "delay": 1
    }
  },
  {
    "time": 110,
    "type": "editLink",
    "params": {
      "src": "s256",
      "dst": "s257",
      "loss": 1,
      "delay": 2
    }
  },
  {
    "time": 130,
    "type": "editLink",
    "params": {
      "src": "s256",
      "dst": "s257",
      "loss": 0,
      "delay": 0,
      "bw": 1000
    }
  },
  {
    "time": 131,
    "type": "editLink",
    "params": {
      "src": "s512",
      "dst": "s513",
      "loss": 2,
      "delay": 100
    }
  },
  {
```

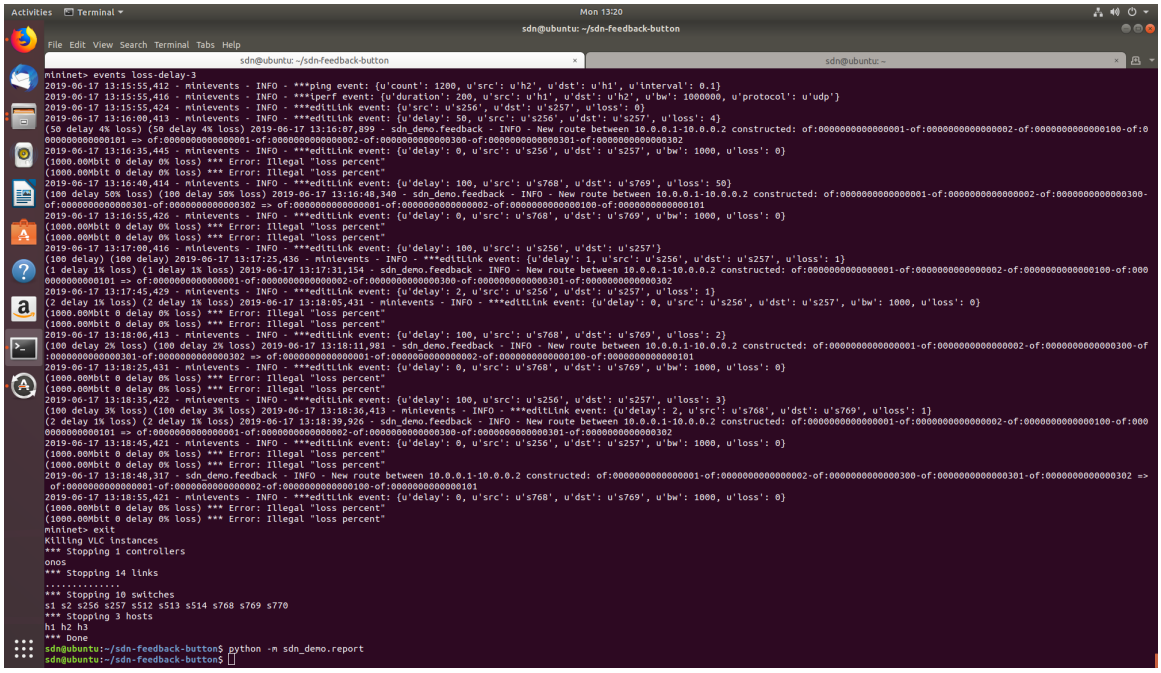
```
    "time": 150,
    "type": "editLink",
    "params": {
      "src": "s512",
      "dst": "s513",
      "loss": 0,
      "delay": 0,
      "bw": 1000
    }
  },
  {
    "time": 160,
    "type": "editLink",
    "params": {
      "src": "s256",
      "dst": "s257",
      "loss": 3,
      "delay": 100
    }
  },
  {
    "time": 161,
    "type": "editLink",
    "params": {
      "src": "s512",
      "dst": "s513",
      "loss": 1,
      "delay": 2
    }
  },
  {
    "time": 170,
    "type": "editLink",
    "params": {
      "src": "s256",
      "dst": "s257",
      "loss": 0,
      "delay": 0,
      "bw": 1000
    }
  },
  {
    "time": 180,
    "type": "editLink",
    "params": {
      "src": "s512",
      "dst": "s513",
      "loss": 0,
      "delay": 0,
      "bw": 1000
    }
  }
]
```

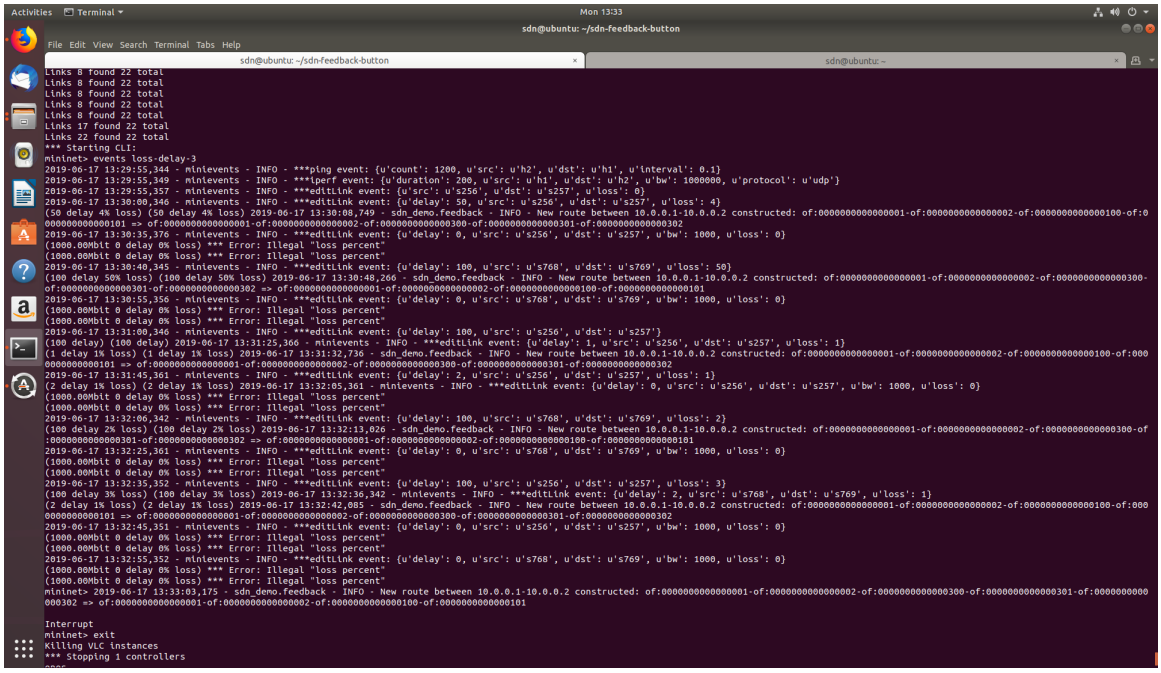
Appendix B.

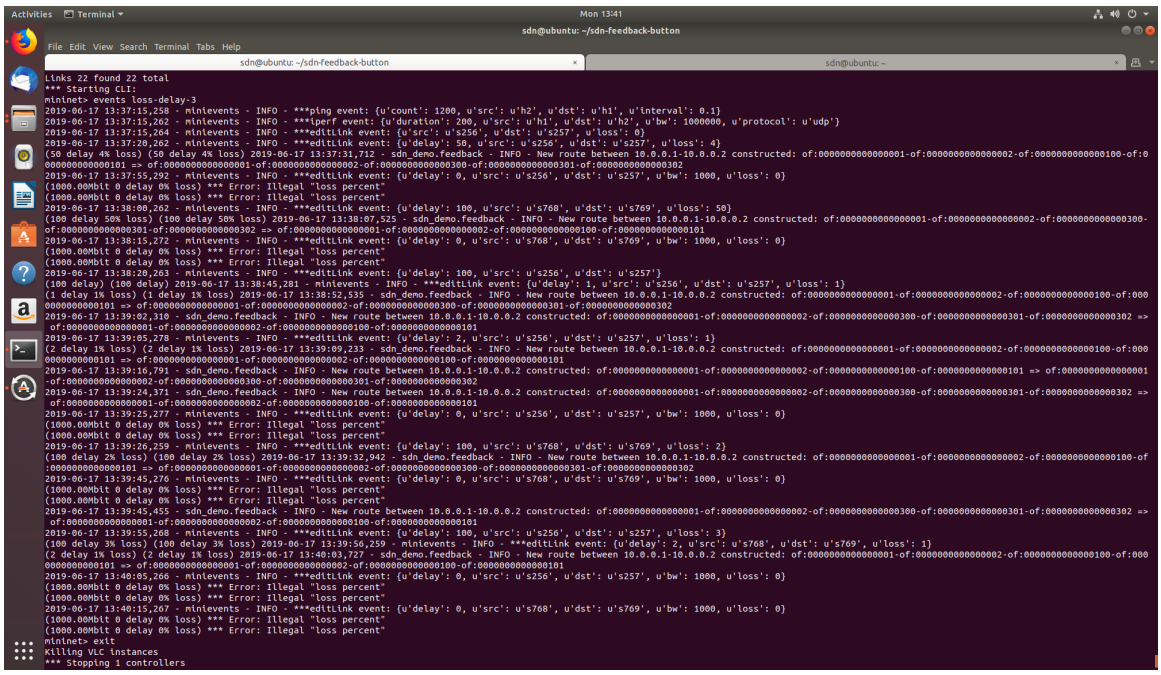
B1. Sample results of applying dynamic event changes, user QoE feedback and rerouting discussed in section 6.1 for in lab user participation.

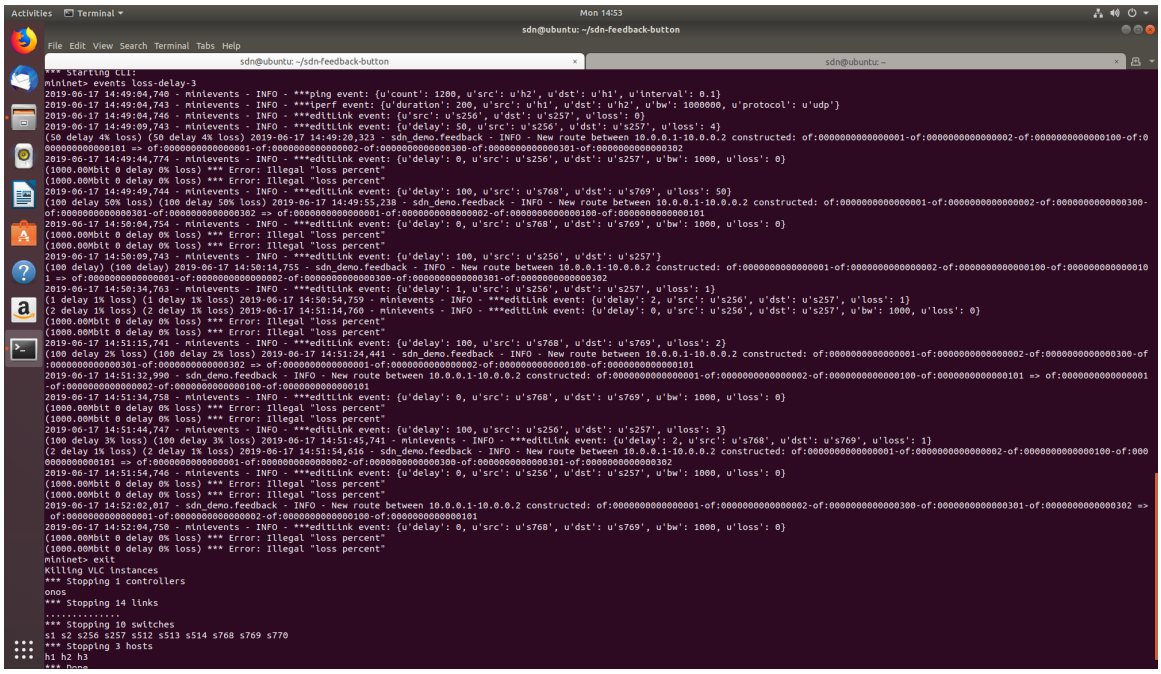




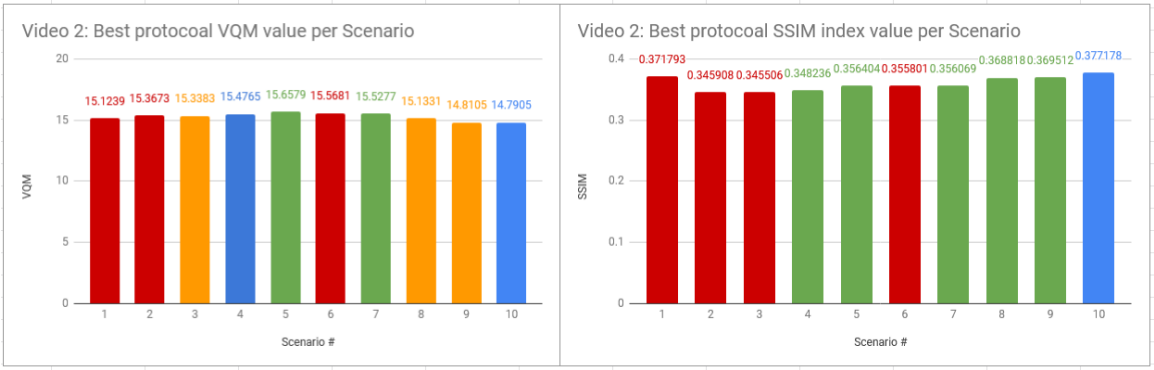

Activities
Terminal
Mon 13:20
sdn@ubuntu: ~/sdn-feedback-button
File Edit View Search Terminal Tabs Help
sdn@ubuntu: ~/sdn-feedback-button
sdn@ubuntu: ~
mininet> events loss-delay-3
2019-06-17 13:15:55,412 - minievents - INFO - ***ping event: {u'count': 1200, u'src': u'h2', u'dst': u'h1', u'interval': 0.1}
2019-06-17 13:15:55,416 - minievents - INFO - ***iperf event: {u'duration': 200, u'src': u'h1', u'dst': u'h2', u'bw': 1000000, u'protocol': u'udp'}
2019-06-17 13:15:55,424 - minievents - INFO - ***editLink event: {u'src': u's256', u'dst': u's257', u'loss': 0}
2019-06-17 13:16:00,413 - minievents - INFO - ***editLink event: {u'delay': 50, u'src': u's256', u'dst': u's257', u'loss': 4}
(50 delay 4% loss) (50 delay 4% loss) 2019-06-17 13:16:07,899 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101 => of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302
2019-06-17 13:16:35,445 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's256', u'dst': u's257', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-06-17 13:16:40,414 - minievents - INFO - ***editLink event: {u'delay': 100, u'src': u's768', u'dst': u's769', u'loss': 50}
(100 delay 50% loss) (100 delay 50% loss) 2019-06-17 13:16:48,340 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302 => of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101
2019-06-17 13:16:55,426 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's768', u'dst': u's769', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-06-17 13:17:00,416 - minievents - INFO - ***editLink event: {u'delay': 100, u'src': u's256', u'dst': u's257'}
(100 delay) (100 delay) 2019-06-17 13:17:25,436 - minievents - INFO - ***editLink event: {u'delay': 1, u'src': u's256', u'dst': u's257', u'loss': 1}
(1 delay 1% loss) (1 delay 1% loss) 2019-06-17 13:17:31,154 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101 => of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302
2019-06-17 13:17:45,429 - minievents - INFO - ***editLink event: {u'delay': 2, u'src': u's256', u'dst': u's257', u'loss': 1}
(2 delay 1% loss) (2 delay 1% loss) 2019-06-17 13:18:05,431 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's256', u'dst': u's257', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-06-17 13:18:06,413 - minievents - INFO - ***editLink event: {u'delay': 100, u'src': u's768', u'dst': u's769', u'loss': 2}
(100 delay 2% loss) (100 delay 2% loss) 2019-06-17 13:18:11,981 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302 => of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101
2019-06-17 13:18:25,431 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's768', u'dst': u's769', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-06-17 13:18:35,422 - minievents - INFO - ***editLink event: {u'delay': 100, u'src': u's256', u'dst': u's257', u'loss': 3}
(100 delay 3% loss) (100 delay 3% loss) 2019-06-17 13:18:36,413 - minievents - INFO - ***editLink event: {u'delay': 2, u'src': u's768', u'dst': u's769', u'loss': 1}
(2 delay 1% loss) (2 delay 1% loss) 2019-06-17 13:18:39,926 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101 => of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302
2019-06-17 13:18:45,421 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's256', u'dst': u's257', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-06-17 13:18:48,317 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302 => of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101
2019-06-17 13:18:55,421 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's768', u'dst': u's769', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
mininet> exit
Killing VLC instances
*** Stopping 1 controllers
onos
*** Stopping 14 links
....
*** Stopping 10 switches
s1 s2 s256 s257 s512 s513 s514 s768 s769 s770
*** Stopping 3 hosts
h1 h2 h3
*** Done
sdn@ubuntu:~/sdn-feedback-button$ python -m sdn_demo.report
sdn@ubuntu:~/sdn-feedback-button$

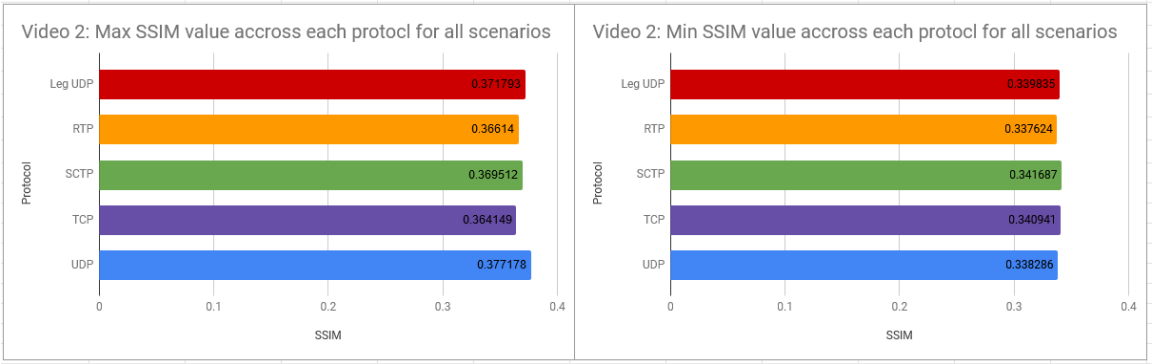

Activities
Terminal
Mon 13:33
sdn@ubuntu: ~/sdn-feedback-button
File Edit View Search Terminal Tabs Help
sdn@ubuntu: ~/sdn-feedback-button
sdn@ubuntu: ~
Links 8 found 22 total
Links 8 found 22 total
Links 8 found 22 total
Links 8 found 22 total
Links 8 found 22 total
Links 17 found 22 total
Links 22 found 22 total
*** Starting CLI:
mininet> events loss-delay-3
2019-06-17 13:29:55,344 - minievents - INFO - ***ping event: {u'count': 1200, u'src': u'h2', u'dst': u'h1', u'interval': 0.1}
2019-06-17 13:29:55,349 - minievents - INFO - ***iperf event: {u'duration': 200, u'src': u'h1', u'dst': u'h2', u'bw': 1000000, u'protocol': u'udp'}
2019-06-17 13:29:55,357 - minievents - INFO - ***editLink event: {u'src': u's256', u'dst': u's257', u'loss': 0}
2019-06-17 13:30:00,346 - minievents - INFO - ***editLink event: {u'delay': 50, u'src': u's256', u'dst': u's257', u'loss': 4}
(50 delay 4% loss) (50 delay 4% loss) 2019-06-17 13:30:08,749 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101 => of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302
2019-06-17 13:30:35,376 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's256', u'dst': u's257', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-06-17 13:30:40,345 - minievents - INFO - ***editLink event: {u'delay': 100, u'src': u's768', u'dst': u's769', u'loss': 50}
(100 delay 50% loss) (100 delay 50% loss) 2019-06-17 13:30:48,266 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302 => of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101
2019-06-17 13:30:55,356 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's768', u'dst': u's769', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-06-17 13:31:00,346 - minievents - INFO - ***editLink event: {u'delay': 100, u'src': u's256', u'dst': u's257'}
(100 delay) (100 delay) 2019-06-17 13:31:25,366 - minievents - INFO - ***editLink event: {u'delay': 1, u'src': u's256', u'dst': u's257', u'loss': 1}
(1 delay 1% loss) (1 delay 1% loss) 2019-06-17 13:31:32,736 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101 => of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302
2019-06-17 13:31:45,361 - minievents - INFO - ***editLink event: {u'delay': 2, u'src': u's256', u'dst': u's257', u'loss': 1}
(2 delay 1% loss) (2 delay 1% loss) 2019-06-17 13:32:05,361 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's256', u'dst': u's257', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-06-17 13:32:06,342 - minievents - INFO - ***editLink event: {u'delay': 100, u'src': u's768', u'dst': u's769', u'loss': 2}
(100 delay 2% loss) (100 delay 2% loss) 2019-06-17 13:32:13,026 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302 => of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101
2019-06-17 13:32:25,361 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's768', u'dst': u's769', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-06-17 13:32:35,352 - minievents - INFO - ***editLink event: {u'delay': 100, u'src': u's256', u'dst': u's257', u'loss': 3}
(100 delay 3% loss) (100 delay 3% loss) 2019-06-17 13:32:36,342 - minievents - INFO - ***editLink event: {u'delay': 2, u'src': u's768', u'dst': u's769', u'loss': 1}
(2 delay 1% loss) (2 delay 1% loss) 2019-06-17 13:32:42,885 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101 => of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302
2019-06-17 13:32:45,351 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's256', u'dst': u's257', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-06-17 13:32:55,352 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's768', u'dst': u's769', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
mininet> 2019-06-17 13:33:03,175 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302 => of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101
Interrupt
mininet> exit
Killing VLC instances
*** Stopping 1 controllers

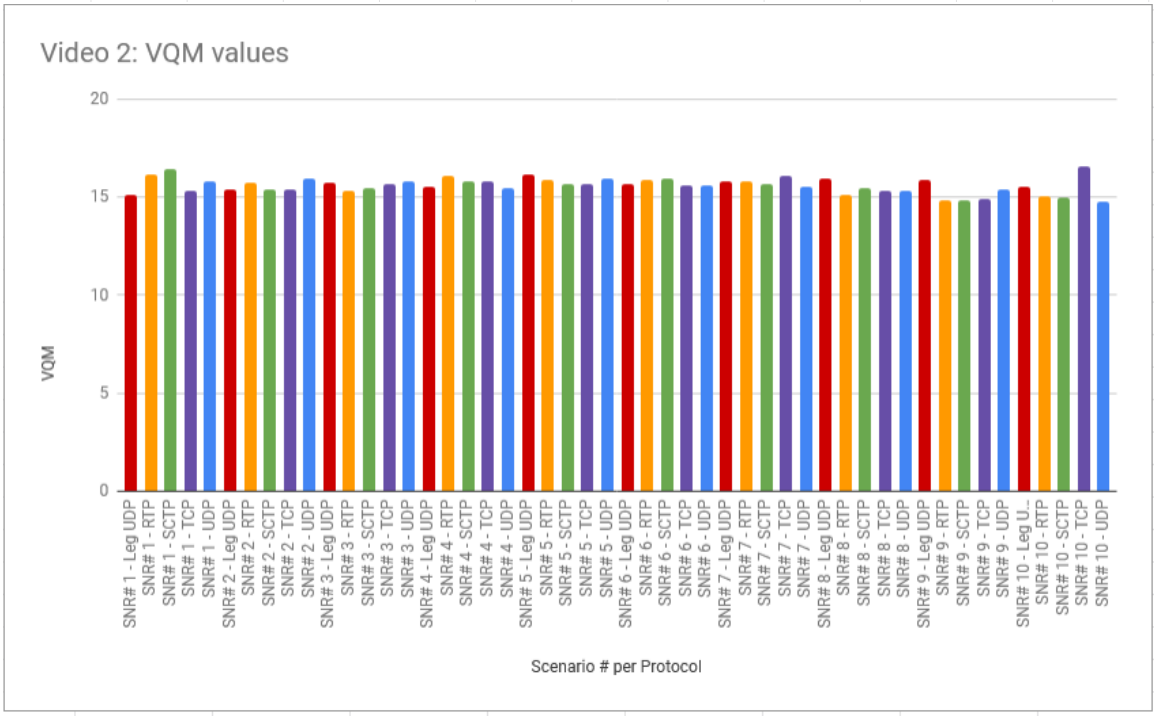

Activities
Terminal
Mon 13:41
sdn@ubuntu: ~/sdn-feedback-button
File Edit View Search Terminal Tabs Help
sdn@ubuntu: ~/sdn-feedback-button
sdn@ubuntu: ~
Links 22 found 22 total
*** Starting CLI:
mininet> events loss-delay-3
2019-06-17 13:37:15,258 - minievents - INFO - ***ping event: {u'count': 1200, u'src': u'h2', u'dst': u'h1', u'interval': 0.1}
2019-06-17 13:37:15,262 - minievents - INFO - ***iperf event: {u'duration': 200, u'src': u'h1', u'dst': u'h2', u'bw': 1000000, u'protocol': u'udp'}
2019-06-17 13:37:15,264 - minievents - INFO - ***editLink event: {u'src': u's256', u'dst': u's257', u'loss': 0}
2019-06-17 13:37:20,262 - minievents - INFO - ***editLink event: {u'delay': 50, u'src': u's256', u'dst': u's257', u'loss': 4}
(50 delay 4% loss) (50 delay 4% loss) 2019-06-17 13:37:31,712 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0
000000000000101 => of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302
2019-06-17 13:37:55,292 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's256', u'dst': u's257', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-06-17 13:38:00,262 - minievents - INFO - ***editLink event: {u'delay': 100, u'src': u's768', u'dst': u's769', u'loss': 50}
(100 delay 50% loss) (100 delay 50% loss) 2019-06-17 13:38:07,525 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000300-
of:0000000000000301-of:0000000000000302 => of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101
2019-06-17 13:38:15,272 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's768', u'dst': u's769', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-06-17 13:38:20,263 - minievents - INFO - ***editLink event: {u'delay': 100, u'src': u's256', u'dst': u's257'}
(100 delay) (100 delay) 2019-06-17 13:38:45,281 - minievents - INFO - ***editLink event: {u'delay': 1, u'src': u's256', u'dst': u's257', u'loss': 1}
(1 delay 1% loss) (1 delay 1% loss) 2019-06-17 13:38:52,535 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000100-of:000
0000000000101 => of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302
2019-06-17 13:39:02,310 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302 =>
of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101
2019-06-17 13:39:05,278 - minievents - INFO - ***editLink event: {u'delay': 2, u'src': u's256', u'dst': u's257', u'loss': 1}
(2 delay 1% loss) (2 delay 1% loss) 2019-06-17 13:39:09,233 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000100-of:000
0000000000101 => of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101
2019-06-17 13:39:16,791 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101 => of:0000000000000001
-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302
2019-06-17 13:39:24,371 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302 =>
of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101
2019-06-17 13:39:25,277 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's256', u'dst': u's257', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-06-17 13:39:26,259 - minievents - INFO - ***editLink event: {u'delay': 100, u'src': u's768', u'dst': u's769', u'loss': 2}
(100 delay 2% loss) (100 delay 2% loss) 2019-06-17 13:39:32,942 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000100-of
:0000000000000101 => of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302
2019-06-17 13:39:45,276 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's768', u'dst': u's769', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-06-17 13:39:45,455 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302 =>
of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101
2019-06-17 13:39:55,268 - minievents - INFO - ***editLink event: {u'delay': 100, u'src': u's256', u'dst': u's257', u'loss': 3}
(100 delay 3% loss) (100 delay 3% loss) 2019-06-17 13:39:56,259 - minievents - INFO - ***editLink event: {u'delay': 2, u'src': u's768', u'dst': u's769', u'loss': 1}
(2 delay 1% loss) (2 delay 1% loss) 2019-06-17 13:40:03,727 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000100-of:000
0000000000101 => of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101
2019-06-17 13:40:05,266 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's256', u'dst': u's257', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-06-17 13:40:15,267 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's768', u'dst': u's769', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
mininet> exit
Killing VLC instances
*** Stopping 1 controllers

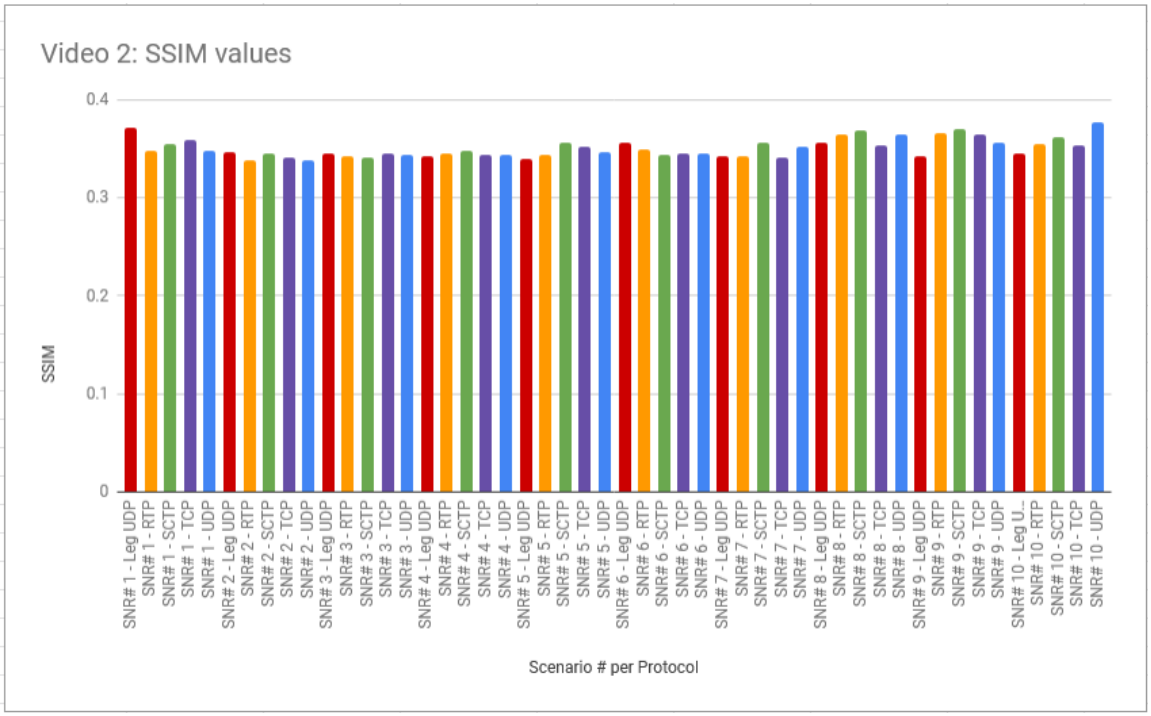

Activities
Terminal
Mon 14:53
sdn@ubuntu: ~/sdn-feedback-button
File Edit View Search Terminal Tabs Help
sdn@ubuntu: ~/sdn-feedback-button
sdn@ubuntu: ~
*** Starting CLI:
mininet> events loss-delay-3
2019-06-17 14:49:04,740 - minievents - INFO - ***ping event: {u'count': 1200, u'src': u'h2', u'dst': u'h1', u'interval': 0.1}
2019-06-17 14:49:04,743 - minievents - INFO - ***iperf event: {u'duration': 200, u'src': u'h1', u'dst': u'h2', u'bw': 1000000, u'protocol': u'udp'}
2019-06-17 14:49:04,746 - minievents - INFO - ***editLink event: {u'src': u's256', u'dst': u's257', u'loss': 0}
2019-06-17 14:49:09,743 - minievents - INFO - ***editLink event: {u'delay': 50, u'src': u's256', u'dst': u's257', u'loss': 4}
(50 delay 4% loss) (50 delay 4% loss) 2019-06-17 14:49:20,323 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0
000000000000101 => of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302
2019-06-17 14:49:44,774 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's256', u'dst': u's257', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-06-17 14:49:49,744 - minievents - INFO - ***editLink event: {u'delay': 100, u'src': u's768', u'dst': u's769', u'loss': 50}
(100 delay 50% loss) (100 delay 50% loss) 2019-06-17 14:49:55,238 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000300-
of:0000000000000301-of:0000000000000302 => of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101
2019-06-17 14:50:04,754 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's768', u'dst': u's769', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-06-17 14:50:09,743 - minievents - INFO - ***editLink event: {u'delay': 100, u'src': u's256', u'dst': u's257'}
(100 delay) (100 delay) 2019-06-17 14:50:14,755 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000100-of:000000000000010
1 => of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302
2019-06-17 14:50:34,763 - minievents - INFO - ***editLink event: {u'delay': 1, u'src': u's256', u'dst': u's257', u'loss': 1}
(1 delay 1% loss) (1 delay 1% loss) 2019-06-17 14:50:54,759 - minievents - INFO - ***editLink event: {u'delay': 2, u'src': u's256', u'dst': u's257', u'loss': 1}
(2 delay 1% loss) (2 delay 1% loss) 2019-06-17 14:51:14,760 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's256', u'dst': u's257', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-06-17 14:51:15,741 - minievents - INFO - ***editLink event: {u'delay': 100, u'src': u's768', u'dst': u's769', u'loss': 2}
(100 delay 2% loss) (100 delay 2% loss) 2019-06-17 14:51:24,441 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000300-of
:0000000000000301-of:0000000000000302 => of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101
2019-06-17 14:51:32,990 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101 => of:0000000000000001
-of:0000000000000002-of:0000000000000100-of:0000000000000101
2019-06-17 14:51:34,758 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's768', u'dst': u's769', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-06-17 14:51:44,747 - minievents - INFO - ***editLink event: {u'delay': 100, u'src': u's256', u'dst': u's257', u'loss': 3}
(100 delay 3% loss) (100 delay 3% loss) 2019-06-17 14:51:45,741 - minievents - INFO - ***editLink event: {u'delay': 2, u'src': u's768', u'dst': u's769', u'loss': 1}
(2 delay 1% loss) (2 delay 1% loss) 2019-06-17 14:51:54,616 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000100-of:000
0000000000101 => of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302
2019-06-17 14:51:54,746 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's256', u'dst': u's257', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
2019-06-17 14:52:02,017 - sdn_demo.feedback - INFO - New route between 10.0.0.1-10.0.0.2 constructed: of:0000000000000001-of:0000000000000002-of:0000000000000300-of:0000000000000301-of:0000000000000302 =>
of:0000000000000001-of:0000000000000002-of:0000000000000100-of:0000000000000101
2019-06-17 14:52:04,750 - minievents - INFO - ***editLink event: {u'delay': 0, u'src': u's768', u'dst': u's769', u'bw': 1000, u'loss': 0}
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
(1000.00Mbit 0 delay 0% loss) *** Error: Illegal "loss percent"
mininet> exit
Killing VLC instances
*** Stopping 1 controllers
onos
*** Stopping 14 links
....
*** Stopping 10 switches
s1 s2 s256 s257 s512 s513 s514 s768 s769 s770
*** Stopping 3 hosts
h1 h2 h3

B2. Video 2 VQM Analysis experiment results.








Video 2: SSIM values
SSIM
0.4
0.3
0.2
0.1
0
SNR# 1 - Leg UDP
SNR# 1 - RTP
SNR# 1 - SCTP
SNR# 1 - TCP
SNR# 1 - UDP
SNR# 2 - Leg UDP
SNR# 2 - RTP
SNR# 2 - SCTP
SNR# 2 - TCP
SNR# 2 - UDP
SNR# 3 - Leg UDP
SNR# 3 - RTP
SNR# 3 - SCTP
SNR# 3 - TCP
SNR# 3 - UDP
SNR# 4 - Leg UDP
SNR# 4 - RTP
SNR# 4 - SCTP
SNR# 4 - TCP
SNR# 4 - UDP
SNR# 5 - Leg UDP
SNR# 5 - RTP
SNR# 5 - SCTP
SNR# 5 - TCP
SNR# 5 - UDP
SNR# 6 - Leg UDP
SNR# 6 - RTP
SNR# 6 - SCTP
SNR# 6 - TCP
SNR# 6 - UDP
SNR# 7 - Leg UDP
SNR# 7 - RTP
SNR# 7 - SCTP
SNR# 7 - TCP
SNR# 7 - UDP
SNR# 8 - Leg UDP
SNR# 8 - RTP
SNR# 8 - SCTP
SNR# 8 - TCP
SNR# 8 - UDP
SNR# 9 - Leg UDP
SNR# 9 - RTP
SNR# 9 - SCTP
SNR# 9 - TCP
SNR# 9 - UDP
SNR# 10 - Leg U...
SNR# 10 - RTP
SNR# 10 - SCTP
SNR# 10 - TCP
SNR# 10 - UDP
Scenario # per Protocol

9 REFERENCES

[1] A. Dutta, J. Chennikara, W. Chen, O. Altintas, and H. Schulzrinne, "Multicasting streaming media to mobile users," *IEEE Communications Magazine,* vol. 41, pp. 81-89, 2003.

[2] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," 2070-1721, 2003.

[3] M. Karakus and A. Durresi, "Quality of service (QoS) in software defined networking (SDN): A survey," *Journal of Network and Computer Applications,* vol. 80, pp. 200-218, 2017.

[4] R. R. Roy, *Handbook on Session Initiation Protocol: Networked Multimedia Communications for IP Telephony*: Crc Press, 2016.

[5] C. Liu, "Multimedia over ip: Rsvp, rtp, rtcp, rtsp," *Handbook of emerging communications technologies: the next decade,* pp. 29-46, 1997.

[6] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine,* vol. 51, pp. 114-119, 2013.

[7] T. Hobfeld, R. Schatz, M. Varela, and C. Timmerer, "Challenges of QoE management for cloud applications," *IEEE Communications Magazine,* vol. 50, 2012.

[8] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan*, et al.*, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Communications Magazine,* vol. 51, pp. 36-43, 2013.

[9] F. Kuipers, R. Kooij, D. De Vleeschauwer, and K. Brunnström, "Techniques for measuring quality of experience," in *International Conference on Wired/Wireless Internet Communications*, 2010, pp. 216-227.

[10] C.-C. Wu, K.-T. Chen, Y.-C. Chang, and C.-L. Lei, "Crowdsourcing multimedia QoE evaluation: A trusted framework," *IEEE transactions on multimedia,* vol. 15, pp. 1121-1137, 2013.

[11] K.-T. Chen, C.-C. Tu, and W.-C. Xiao, "Oneclick: A framework for measuring network quality of experience," in *INFOCOM 2009, IEEE*, 2009, pp. 702-710.

[12] E. Liotou, K. Samdanis, E. Pateromichelakis, N. Passas, and L. Merakos, "Qoe-sdn app: A rate-guided qoe-aware sdn-app for http adaptive video streaming," *IEEE Journal on Selected Areas in Communications,* vol. 36, pp. 598-615, 2018.

[13] D. Sanvito, D. Moro, M. Gullì, I. Filippini, A. Capone, and A. Campanella, "ONOS Intent Monitor and Reroute service: enabling plug&play routing logic," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, 2018, pp. 272-276.

[14] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide*, et al.*, "ONOS: towards an open, distributed SDN OS," in *Proceedings of the third workshop on Hot topics in software defined networking*, 2014, pp. 1-6.

[15] *ONOS Intent Framework*. Available: https://wiki.onosproject.org/display/ONOS/Intent+Framework

[16] C. Bu, X. Wang, H. Cheng, M. Huang, and K. Li, "Routing as a service (RaaS): An open framework for customizing routing services," *Journal of Network and Computer Applications,* vol. 125, pp. 130-145, 2019.

[17] T.-N. Lin, Y.-M. Hsu, S.-Y. Kao, and P.-W. Chi, "OpenE2EQoS: Meter-based method for end-to-end QoS of multimedia services over SDN," in *Personal, Indoor, and Mobile Radio Communications (PIMRC), 2016 IEEE 27th Annual International Symposium on*, 2016, pp. 1-6.

[18] A. Farshad, P. Georgopoulos, M. Broadbent, M. Mu, and N. Race, "Leveraging SDN to provide an in-network QoE measurement framework," in *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*, 2015, pp. 239-244.

[19] C. Alberti, D. Renzi, C. Timmerer, C. Mueller, S. Lederer, S. Battista*, et al.*, "Automated QoE evaluation of dynamic adaptive streaming over HTTP," in *Quality of Multimedia Experience (QoMEX), 2013 Fifth International Workshop on*, 2013, pp. 58-63.

[20] Q. Zhang, S. Q. Zhang, J. Lin, H. Bannazadeh, and A. Leon-Garcia, "Kaleidoscope: Real-time content delivery in software defined infrastructures," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 686-692.

[21] A. B. Letaifa, "Real Time ML-Based QoE Adaptive Approach in SDN Context for HTTP Video Services," *Wireless Personal Communications,* vol. 103, pp. 2633-2656, 2018.

[22] M.-E. Xezonaki, E. Liotou, N. Passas, and L. Merakos, "An SDN QoE Monitoring Framework for VoIP and Video Applications," in *2018 IEEE 19th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, 2018, pp. 1-6.

[23] F. Volpato, M. P. Da Silva, A. L. Gonçalves, and M. A. R. Dantas, "An autonomic QoE-aware management architecture for software-defined networking," in *2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, 2017, pp. 220-225.

[24] P. Le Callet, S. Möller, and A. Perkis, "Qualinet white paper on definitions of quality of experience," *European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003),* vol. 3, 2012.

[25] U. Reiter, K. Brunnström, K. De Moor, M.-C. Larabi, M. Pereira, A. Pinheiro*, et al.*, "Factors influencing quality of experience," in *Quality of experience*, ed: Springer, 2014, pp. 55-72.

[26] R. Serral-Gracià, E. Cerqueira, M. Curado, M. Yannuzzi, E. Monteiro, and X. Masip-Bruin, "An overview of quality of experience measurement challenges for video applications in IP networks," in *International Conference on Wired/Wireless Internet Communications*, 2010, pp. 252-263.

[27] K. Piamrat, C. Viho, J.-M. Bonnin, and A. Ksentini, "Quality of experience measurements for video streaming over wireless networks," in *Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference on*, 2009, pp. 1184-1189.

[28] M. Alreshoodi and J. Woods, "Survey on QoE\QoS correlation models for multimedia services," *arXiv preprint arXiv:1306.0221,* 2013.

[29] T. Hossfeld, C. Keimel, M. Hirth, B. Gardlo, J. Habigt, K. Diepold, *et al.*, "Best practices for QoE crowdtesting: QoE assessment with crowdsourcing," *IEEE Transactions on Multimedia,* vol. 16, pp. 541-558, 2014.
[30] F. Xiao, "DCT-based video quality evaluation," *Final Project for EE392J,* vol. 769, 2000.
[31] J. Howe, "The rise of crowdsourcing," *Wired magazine,* vol. 14, pp. 1-4, 2006.
[32] J. Howe, "Crowdsourcing: A definition," 2006.
[33] T. Hoßfeld and C. Keimel, "Crowdsourcing in QoE evaluation," in *Quality of Experience*, ed: Springer, 2014, pp. 315-327.
[34] *Amazon Mechanical Turk (2013)*. Available: http://mturk.com
[35] (2013). *Microworker.* Available: http://microworkers.com
[36] E. Estellés-Arolas and F. González-Ladrón-De-Guevara, "Towards an integrated crowdsourcing definition," *Journal of Information science,* vol. 38, pp. 189-200, 2012.
[37] *Subjectify.us*. Available: http://www.subjectify.us
[38] H. Schulzrinne, A. Rao, R. Lanphier, M. Westerlund, and M. Stiemerling, "Real-time streaming protocol version 2.0," 2070-1721, 2016.
[39] L. Ong and J. Yoakum, "An introduction to the stream control transmission protocol (SCTP)," 2070-1721, 2002.
[40] R. Stewart, "Stream control transmission protocol," 2070-1721, 2007.
[41] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman, "The secure real-time transport protocol (SRTP)," 2070-1721, 2004.
[42] P. Thermos and A. Takanen, *Securing VoIP networks: threats, vulnerabilities, and countermeasures*: Pearson Education, 2007.
[43] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, *et al.*, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review,* vol. 38, pp. 69-74, 2008.
[44] *NOX controller*. Available: https://github.com/noxrepo/nox
[45] *POX Controller*. Available: https://github.com/noxrepo/pox
[46] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, 2014, pp. 1-6.
[47] *ifwd ONOS Package*. Available: https://github.com/opennetworkinglab/onos-app-samples/tree/master/ifwd
[48] A. A. Haghighi, S. Shahbazpanahi, and S. S. Heydari, "Stochastic QoE-Aware Optimization in Cloud-Based Content Delivery Networks," *IEEE Access,* vol. 6, pp. 32662-32672, 2018.
[49] X. Chen, P. N. Bennett, K. Collins-Thompson, and E. Horvitz, "Pairwise ranking aggregation in a crowdsourced setting," in *Proceedings of the sixth ACM international conference on Web search and data mining*, 2013, pp. 193-202.
[50] C. Giraldo, "Minievents: A mininet Framework to define events in mininet networks," 2015.
[51] D. Vatolin, A. Moskvin, O. Petrov, and N. Trunichkin, "Msu video quality measurement tool," ed, 2009.

[52] A. A. Haghighi, S. S. Heydari, and S. Shahbazpanahi, "Dynamic QoS-Aware Resource Assignment in Cloud-Based Content-Delivery Networks," *IEEE Access,* vol. 6, pp. 2298-2309, 2018.