Investigate how network conditions/impairments affect end-to-end video streaming quality

Alistair Drew

10567887

Contents

[Abstract (150) 2](#_Toc73368006)

[Introduction and background (700) 2](#_Toc73368007)

[SDN 3](#_Toc73368008)

[Dash 3](#_Toc73368009)

[Video Streaming Quality 3](#_Toc73368010)

[Steps Taken 4](#_Toc73368011)

[Report Structure 4](#_Toc73368012)

[Testbed and development (600) 4](#_Toc73368013)

[Setup – general structure of the testbed and configuration 4](#_Toc73368014)

[Code developed. 7](#_Toc73368015)

[Experiments, Results and Analysis (600) 8](#_Toc73368016)

[Discussions, network management via REST API (700) 8](#_Toc73368017)

[Conclusions (150) 8](#_Toc73368018)

[References 8](#_Toc73368019)

[Appendix 9](#_Toc73368020)

## Abstract (150)

* Adaptive video streaming over SDN and NFV
* Effects of packet loss and delay are different form UDP and are displayed as buffering events
* Explain briefly the investigation and the measures of how it is getting carried out
* Explain Network QoS, Application QoS and User QoE.

Extremely popular HTTP adaptive video streaming (HAS) services over the internet such as Over the Top solutions (OTT) like YouTube, Netflix and Twitch must maintain the Quality of Experience of their end-to-end video streaming to remain popular. Due to this Video Service Providers (VSP’s) need to investigate how network conditions, the basest level of the operation effect the Quality of Experience for the user. This report is an investigation into the effect network conditions have on the end-to-end video streaming quality. Software Defined Networks (SDN) allow for which are designed to increase the flexibility of the network by making it easy to manage and adapt how traffic is handled. and Network Function Virtualization (NFV) that allows abstraction of network functions from dedicated hardware to standard hardware allowing for flexible network topology creation

are used to run a battery of experiments on a test bed.

## Introduction and background (700)

* Background information and literature on SDN, DASH
* Video streaming quality
  + Objective and Subjective measures of video quality
  + How its measured in this context
* Tasks to be carried out
  + Steps the investigation needs to carry out
    - Research into the background of the investigation topic
    - Explanation of the necessary technologies and their roles in the investigation
    - Creation of the testbed
    - Designing the experiments with research and analysing the results
    - Critically evaluation the process of the investigation.
* Structure of the report
  + Intro to the components of the investigation and the investigation itself
  + The testbed and its development
  + How the experiments where designed, the results and how they tie into the investigation. Analyse the patterns and trends.
  + Critical evaluation on the investigation and explaining and justifying the network management via REST API
  + A summarization of the investigation findings in the conclusion.

### SDN

Proprietary hardware was king in the networking world until the arrival of software-defined networking (SDN), an approach which can both simplify and allow granular control of networks adaptively to adjust for network conditions. This facilitates network configuration and management (Bonfim, Dias and Fernandes, 2019). This makes it especially well-suited for testing the effects network (QoS) parameters have on application (QoS) parameters and in turn the users Quality of Experience (QoE) as shown in Figure 1. (radar chart ref)

SDN is done through ‘programming’ the network via a centralised controller located in the control plane, the SDN controller makes the decisions on how network traffic is controlled. The data plane which contains the network nodes like switches and hosts are responsible for forwarding traffic according to a set of rules created by the SDN controller. The SDN controller has direct control of the data plain through the southbound protocol OpenFlow.

The purpose of using SDN is to allow easier network management and greater flexibility of traffic flows while also abstracting the implementation of the control plane to non-proprietary software such as open API’s like RYU.

Explain how the SDN controller and OpenFlow switch communicate using flow tables and headers ecs. Then tie it into the investigation 🡪 How changing network QoS effects application QoS and in turn user QoE 🡪 How that benefits the investigation.

### Dash

Dynamic Adaptive Streaming over HTTP (DASH) is a MPEG standard defining a multimedia format and description (DASH reference). DASH comprises of media files divided into modular chunks between 2 to 10 seconds called segments. Each segment is encoded into multiple versions of bit-rate quality levels and depending on the frame rate, bit rate and resolution the quality level can change. The details of the different quality levels are then stored in an XML manifest called a Media Presentation Description file (MPD). Within an MPD file the segments are arranged in adaption sets that contain a set of representations such as bitrate or resolutions. These representations can hold interchangeable versions of media content such as different resolutions or bitrates. By having multiple representation sets in a single MPD file it enables the client to adapt the media stream to the current requirements for network bandwidth (Adaptive HTTP video streaming) by interchanging between these representation sets of differing qualities.

### Video Streaming Quality

HAS quality is defined as … it can be measured in two ways. Objectively using MOS metrics and Subjectively using…

Video streaming quality is measured in two ways, through subjective methods such as Double and Single stimulus testing with users. The subjective video quality is degree of delight or annoyance of the user evaluating the video, from ranging excellent to very annoying.

Or through objective methods such as Full, Reduced and zero reference models. Both are tabulated into a mean opinion score (MOS), the difference being subjective is based of human opinion and objective is based off criteria and metrics that can be measured objectively and automatically via computer program.

Initial delay, buffering duration and buffering frequency.

Application Layer: Rest API 🡪 DASH Client (DASHjs) and DASH server (Apache2) are based

### Steps Taken

The investigation can be divided into a series of steps to be taken sequentially, the first of these steps is to research into the background of HAS and how network conditions effect the QoE of end-to-end video streaming. The second step is to use the research to choose suitable technologies to build the test bed. The third step is to create the test bed itself. The fourth step is to design experiments that measure the correct metrics (packet loss, delay, bw, initial delay, average buffering length, buffering frequency). The fifth step is to carry out the experiments, then analyse and critically evaluate the results. The sixth step is to summarize the key points and conclude.

### Report Structure

The structure of the report is split into five parts, section one is the introduction to content of the report and the background information as well as pertinent literature. Section two details and justifies the design of the test bed and its development. Section three explains and justifies the experiments carried out on the test bed as well as analyses the results from the experiments. Section four is about critical evaluation of the report so far as well as the network management via REST API. Finally, section five is the conclusion that is an executive summery of the report.

## Testbed and development (600)

* General overview of the Structure of the testbed
* Setup of testbed section by section
  + Explain the automated script file setup in sections
  + Give the justification of the configuration step by step and why it was designed the way it was, and any problems in its development
* Code developed.
  + Python, JavaScript or shell scripts created.

### Setup – general structure of the testbed and configuration

The testbed’s general structure is based on the SDN architecture, Mininet was installed to create the network topology adaptively via a python file.

Mininet Creates the SDN remote controller, an OpenvSwitch and four hosts. The SDN remote controller is connected to a REST API called ODL. Within the testbed DASHjs is installed to act as the DASH client while Apache2 acts as the DASH server, both also interact with the REST API ODL and the controller.

Key structure

REST API

ODL

Application Layer

DASH Server

DASH Client

Control Layer

SDN Controller

Data Layer

OpenvSwitch

Hosts

The SDN controller Mininet created in the control layer is ODL installed on a separate VM of an ubuntu 18.4 server image.

Mininet also creates an OpenvSwitch server and four hosts in the Data layer.

Apache 2 is the DASH server

DASHjs is the DASH client

Explain general structure of the testbed - Use the Diagram

Explain installation of Packages - what each section of packages do Then why they are installed

Explain how Mininet was installed and why it was installed (to act as the NFV) tying it to the general structure.

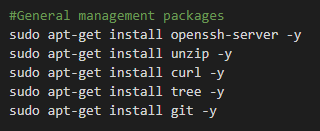
Explain how Microsoft Edge was installed and why the browser was installed (lower cpu resources)

Explain how the video and DASHjs is pulled from the uni server and Why DASHjs was installed (To act as the DASH Client)

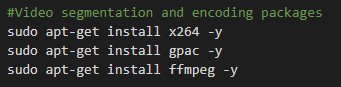
Explain how video is clipped, encoded and segmented and concatenated. Then why it was processed like this with reference to how DASH works?

Explain how the python file was created, how it works and why it was created.

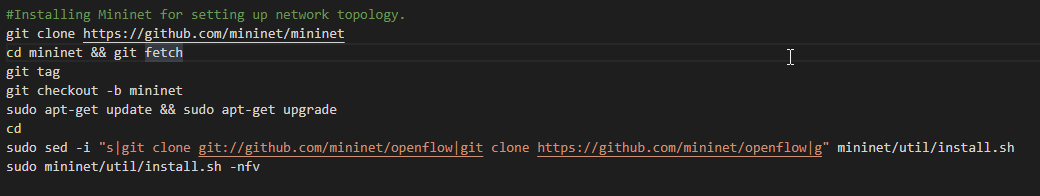
Packages for testbed setup



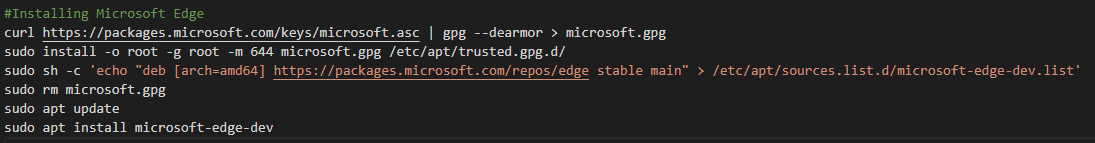




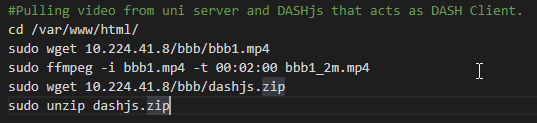
Installing for adaptively creating a network



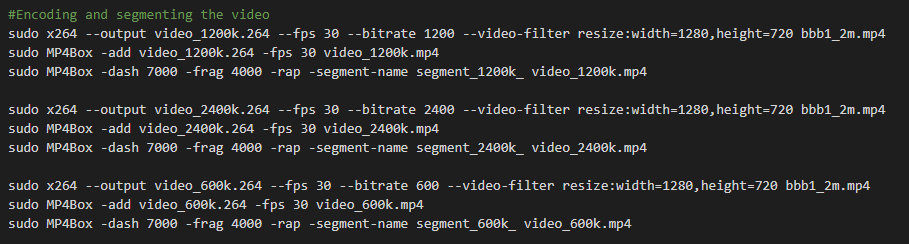
Installing browser that runs on less CPU resources.



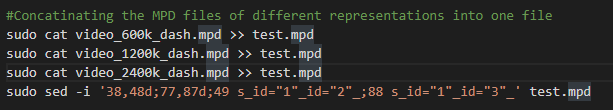
Pulling DASHjs from university server to act as the DASH Client and the video



Segmenting and Encoding the video



Concatenating the MPD file



debug.js

Python Code for network setup via Mininet.

### Code developed.

Additional Code for taking more accurate metrics.

#Putting code into Dash (main.js) to view the average buffer length in experiments

sudo sed -i '759 s~^~var totalBuffer = 0;\n var prevBuffer = 0;\n var bufferCounter = 0;\n var totalBitrate =0; \n var bitrateCounter=0;\n~;778 s~^~\nif (bufferLevel != prevBuffer) {\ntotalBuffer = totalBuffer + bufferLevel;\nbufferCounter++;\n }\nprevBuffer = bufferLevel;\n totalBitrate = Bitrate + Bitrate;\n bitrateCounter++;\n var averageBuffer = totalBuffer/bufferCounter;\n var averageBitrate = totalBitrate/bitrateCounter;\n  console.log("Average Bufferlength: " + averageBuffer.toString() + "Average Bitrate: " + averageBitrate.toString());~' dashjs/app/main.js

# Putting code into Dash (desh.all.debug.js) to display the initial delay in expermiments

sudo sed -i '25543 s#^#\nconsole.time("answer time");\n#;30920 s#^#\n console.timeLog("answer time");\nconsole.timeEnd("answer time");\n#' /var/www/html/dashjs/dist/dash.all.debug.js

## Experiments, Results and Analysis (600)

Results table culminating the core results of my experiments.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Experiment/Network Conditions | Starting Bitrate | Ending Bitrate | Tested Average Initial Delay (s) | Tested Average Buffer frequency | Tested Average Buffer Length (s) | Subjective MOS |
| Experiment 1.1 --> 1Mbps - 0ms - 0% | 600 | 600 | 0.201 | 0 | 14.775 | 2 |
| Experiment 1.3 --> 1Mbps - 0ms - 8% | 600 | 600 | 4.6 | 11 | 3.95 | 1 |
| Experiment 3.1 --> 10Mbps - 0ms - 0% | 2400 | 2400 | 0.236 | 0 | 25.125 | 4 |
| Experiment 3.3 --> 10Mbps - 0ms - 8% | 600 | 600 | 0.171 | 9.75 | 4.525 | 1 |

## Discussions, network management via REST API (700)

## Conclusions (150)

## References

<https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8930519&tag=1>

Barakabitze, A., Barman, N., Ahmad, A., Zadtootaghaj, S., Sun, L., Martini, M. and Atzori, L., 2020. QoE Management of Multimedia Streaming Services in Future Networks: A Tutorial and Survey. *IEEE COMMUNICATIONS SURVEYS & TUTORIALS*, 22(1).

**In-text:**(Barakabitze et al., 2020)

https://dl.acm.org/doi/pdf/10.1145/3172866

Bonfim, M., Dias, K. and Fernandes, S., 2019. Integrated NFV/SDN Architectures. *ACM Computing Surveys*, 51(6), pp.1-39.

**In-text:**(Bonfim, Dias and Fernandes, 2019)

<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6994333>

Kreutz, D., M. V Ramos, F., Verı´ssimo, P., Rothenberg, C., Azodolmolky, S. and Uhlig, S., 2015. https://ieeexplore.ieee.org/abstract/document/6994333. *Proceedings of the IEEE*, 103(1).

**In-text:**(Kreutz et al., 2015)

SDN Intro

<https://www.researchgate.net/publication/221293512_Measuring_the_Quality_of_Experience_of_HTTP_Video_Streaming>

Mok, R., Chan, E. and Chang, R., 2011. Measuring the Quality of Experience of HTTP Video Streaming. In: *12th IFIP/IEEE International Symposium on Integrated Network Management*. [online] Dublin, Ireland: IEEE. Available at: <https://www.researchgate.net/publication/221293512\_Measuring\_the\_Quality\_of\_Experience\_of\_HTTP\_Video\_Streaming> [Accessed 18 May 2021].

**In-text:**(Mok, Chan and Chang, 2011)

Radar chart

<https://www.mdpi.com/2076-3417/10/21/7691/htm>

Gohar, A. and Lee, S., 2020. Multipath Dynamic Adaptive Streaming over HTTP Using Scalable Video Coding in Software Defined Networking. *Applied Sciences*, 10(21), p.7691.

**In-text:**(Gohar and Lee, 2020)

Dash

<https://ieeexplore.ieee.org/abstract/document/5461979>

Chan, A., Zeng, K., Mohapatra, P., Lee, S. and Banerjee, S., 2010. Metrics for Evaluating Video Streaming Quality in Lossy IEEE 802.11 Wireless Networks. In: *2010 Proceedings IEEE INFOCOM*. [online] San Diego, CA, USA: IEEE. Available at: <https://ieeexplore.ieee.org/abstract/document/5461979> [Accessed 20 May 2021].

**In-text:**(Chan et al., 2010)

## Appendix

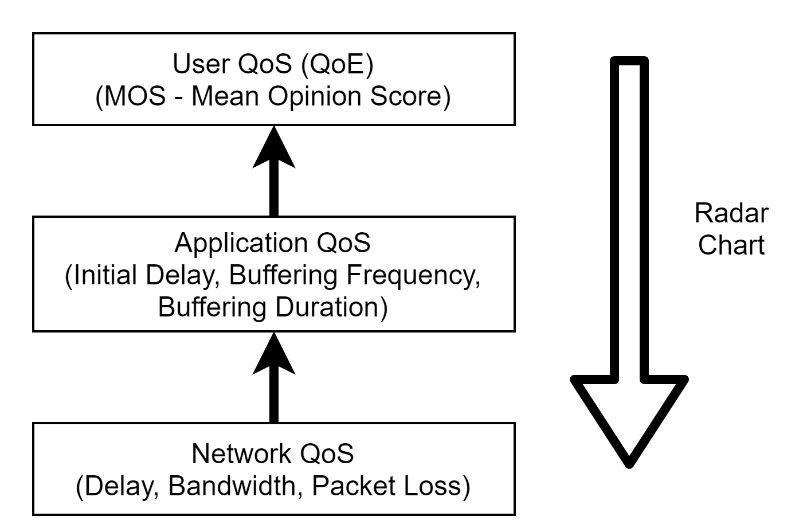


Figure - Interaction of different QoS layers

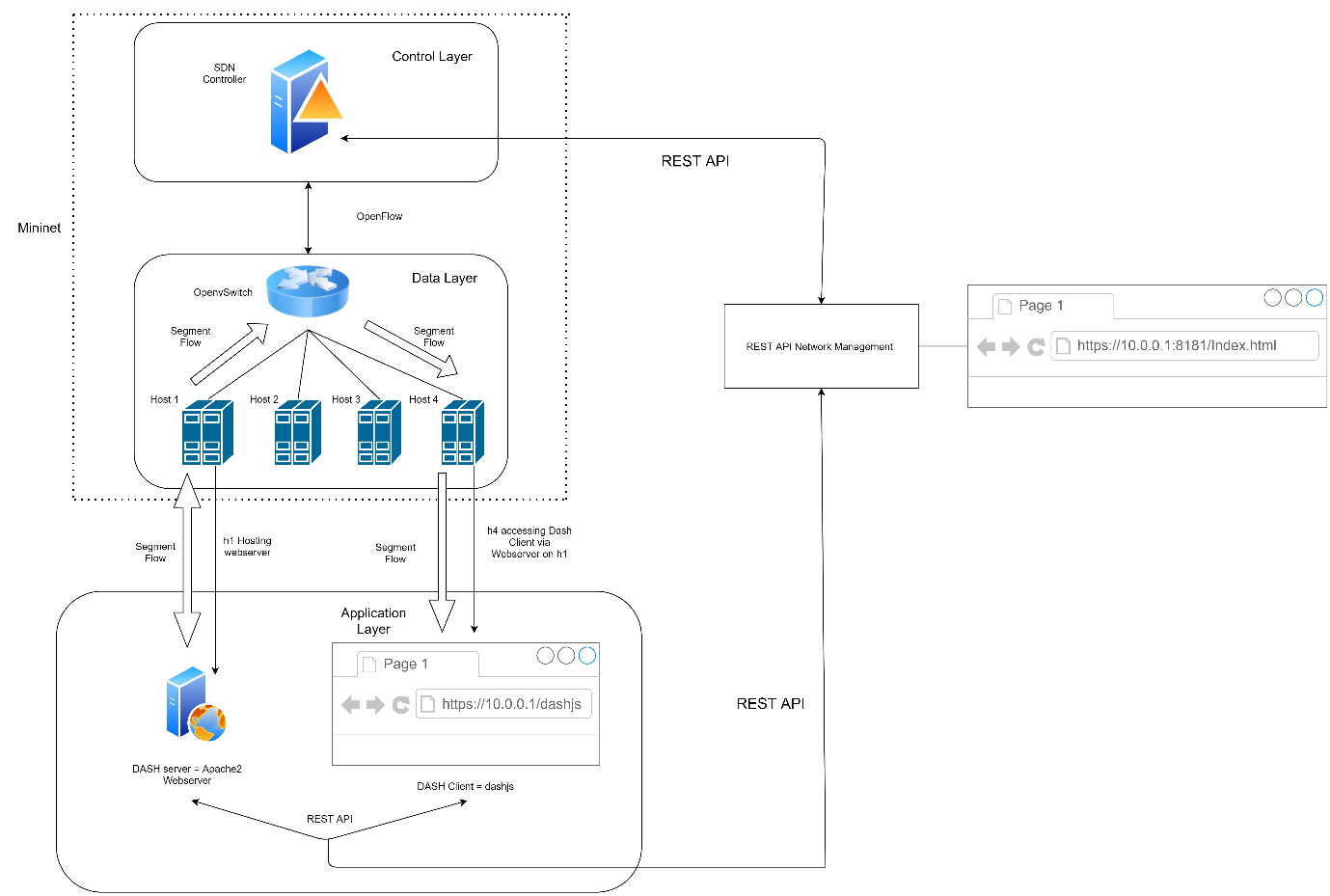


Figure - Overall Testbed Architecture

#Creating the Python file for the network topology

cd

echo "

#!/usr/bin/python

import subprocess

import sys

import os

from mininet.net import Mininet

from mininet.node import Controller

from mininet.cli import CLI

from mininet.link import TCLink

from mininet.log import setLogLevel, info

from mininet.node import OVSKernelSwitch, RemoteController

def myNetwork():

    net = Mininet( topo=None,

                   build=False)

    info( '\*\*\* Adding controller\n' )

    net.addController(name='c0',controller=RemoteController,ip='192.168.10.245, port=6653)

    bw=input("Input bw: ")

    dl=input("Input dl: ")

    ls=input("Input ls: ")

    info( '\*\*\* Add single switch\n')

    s1 = net.addSwitch('s1')

    info( '\*\*\* Add hosts\n')

    h1 = net.addHost('h1')

    h2 = net.addHost('h2')

    h3 = net.addHost('h3')

    h4 = net.addHost('h4')

    info( '\*\*\* Add links with QoS parameters\n')

    net.addLink(h1, s1, cls=TCLink, bw=bw, delay=dl, loss=ls)

    net.addLink(h2, s1, cls=TCLink, bw=bw, delay=dl, loss=ls)

    net.addLink(h3, s1, cls=TCLink, bw=bw, delay=dl, loss=ls)

    net.addLink(h4, s1, cls=TCLink, bw=bw, delay=dl, loss=ls)

    info( '\*\*\* Starting network\n')

    net.start()

    CLI(net)

    net.stop()

if \_\_name\_\_ == '\_\_main\_\_':

    setLogLevel( 'info' )

    myNetwork()

" > network.py

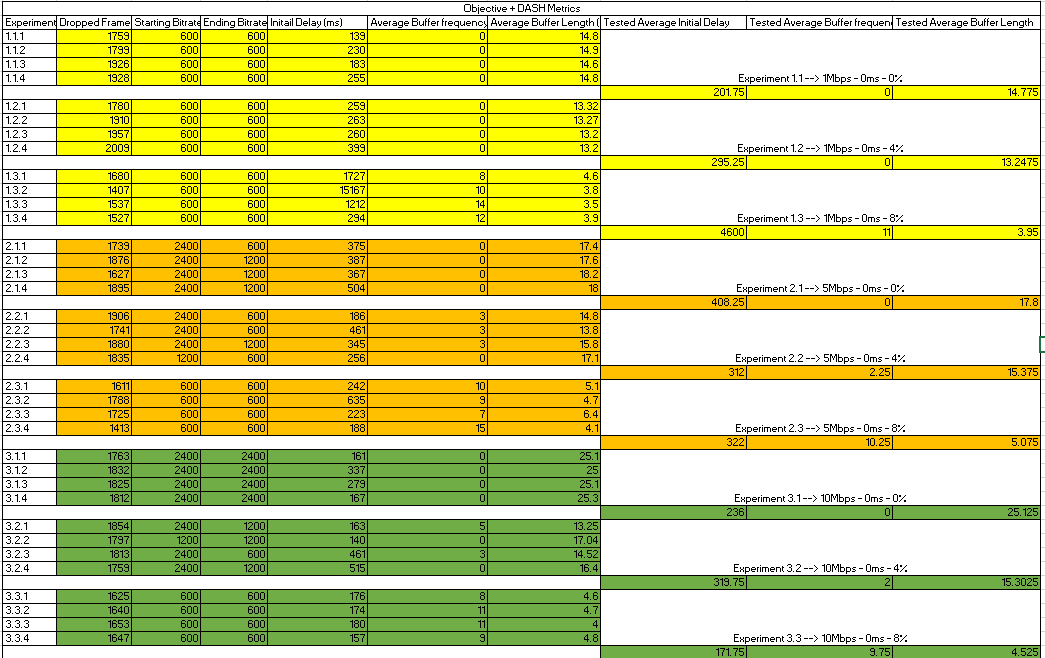


Figure - Full table of objective tests

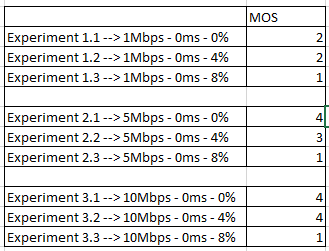


Figure - Full table of subjective tests

https://unilearning.uow.edu.au/report/4bi1.html

https://www.encoding.com/mpeg-dash/