POLITECNICO DI TORINO

Department of Electronics and Telecommunication (DET)

Master Degree Program in Engineering

Communication and Computer Networks

Master Degree Thesis

Internet Bandwidth Measurements On FTTH Access Network

Supervisors
Prof. Marco Mellia
Prof. Maurizio M. Munafò

Candidate
Seyed Sina Vagheh Dashti

July 2019

Dedication

To my beloved Mom and Dad...

Acknowledgements

I express the deep sense of gratitude to Professor Marco Mellia, for giving me the opportunity of performing my thesis under his supervision. I am very much thankful, because of his excellent guidance and help throughout the thesis. Also, without his persistent support and favor, this dissertation could not be started at the company.

I submit my heartiest gratitude to other respected supervisor of this thesis, Professor Maurizio M. Munafò for his kind support and exemplary advices to successfully complete my thesis. His companionship and assistance led this survey move forward.

I would like to thank my manager at the company, Andrea Fregosi that accepted me to carry out my thesis in Fastweb. In addition, I was really fortunate that I had the great help and support of Andrea Sannino, my supervisor in the corporation. Plus, a thank to all the colleagues that they helped me during the time I was working in Fastweb.

Finally, I must thank to my close friends. Without them, it was almost impossible to tolerate tough moments, knotty circumstances and homesickness during these years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem statement

Nowadays, Internet speed measurement in ultra-broadband consumer network, with capacity up to 1000 Mbps is very challenging. Internet service providers try to improve the quality of service that they offer to their customers. Fastweb, one of the main ISPs in Italy wants to increase the quality of its services too. Fastweb has two core networks located at Milan and Rome which are constructed by 36 and 24 PoP respectively. Fastweb offers to its client by the help of FTTH based on GPON architecture, a brand new technology in access networks, up to 1 Gbps for downstream, and 200 Mbps for upstream, and this study is concentrated on the speed greater than 100 Mbps.

The web offers many public speed test services. A very popular one is speedtest.net presented by Ookla and the goal of this study is about understanding of this service, for the Internet speed measurement. An ISP company like Fastweb by fulfilling Ookla requirements can use its services and provide a custom interface for its client for Internet bandwidth measurements. The results of Ookla speed test method include ping, jitter, download and upload value. Fastweb clients sometimes complaining about the quality of the service provided by the company based on this speed test results.

The backbone of the thesis is based on the data analysis. Therefor, it has been tried to find the features that might influence on the speed test results. Fastweb had a large collection of its clients speed measurement for a period about eight months. This data set is collected at server side of the speed test interface. The data set that is used for this study has 23 fields and 3 other fields extracted from them, so in total we have 26 columns in our data set. The total amount of available data was 341155 samples, which are the measurements that are conducted towards the Milan and Rome core networks from a specific location.

Since it is not possible to get a uniform service for every customer, the company defined a threshold for the quality of the service. Accordingly, for the speed greater than 100 Mbps up to 1 Gbps, if a client can utilize its service with at least 50% of the promised quality, that client is at good condition.

Based on this threshold and available data set, the first part of the study is devoted to investigation of this data. So, we wanted to verify how much the results are reliable. Beside, the correlation between physical infrastructure, distance, device capability, etc that might affect the proficiency of this speed test method has examined. Moreover, some statistics based on these samples are provided for the company to give a general view about the behavior of the Fastweb core networks. After these steps, we reached to a basic knowledge about the factors that affect the speed test outcomes more, and they have been focused further in detail.

Other part of the thesis needed significant repetition with different context. A great number of speed test in the provided test bed of Milan core network at the test plant have performed. The goal of these tests was having equal number of samples for different situations in the lab environment. So, the tests have done from the test plant in Milan towards the Milan and Rome core networks. Here we tried to reduce the influences of some factors like noise, interference and other factors that could reduce the device capability for performing the test.

Then, another data analyses have done based on the main factors that affect the speed test results and they have been found in the first part of the analyses. At this stage we reached to an idea of the appropriate choices for different contexts for performing speed test based on Ookla method.

At the end, by the help of packet traces and Wireshark, we tried to understand for example why a specific browser gives better result with respect to the other ones. Here we tried to find a correlation between the performance of each browser and number of persistent TCP connections, TCP window size and encryption.

1.2 Objective and layout of the thesis

In chapter two, we start by, a general and short description of FTTH technology based on GPON architecture for access network. Also, a brief explanation about the architecture of the core network of Fastweb is presented. Moreover, the procedure that GPON uses for downstream and upstream and the elements of the GPON architecture in short is explained.

In chapter three, the Ookla speed test system is going to be explained. First of all the two available Ookla speed test interfaces are illustrated. Plus, we explained how the test results components are prepared by Ookla system. Then, the network and server requirements of the Ookla are described.

Methodologies of the thesis are described in chapter four. At this part, first an example of a sample of the data set is presented, then the structure and the field of the data set is explained. Also, we needed to perform some repetitive tests. Based on this, Selenium which is a tool for test automation with browsers is described. Moreover, the interesting contexts for more investigation and the procedure of the tests are described. At the end of this section, The tools and software for extracting information and analyses are described in this section too.

The results are presented at chapter five. this chapter can be divided to three parts. First, we started by the analyses of first data set and we reached to the factors that affect more the speed test results, like type of browser and operating system. Then, the results of our automated tests at the test bed is illustrated with the similar structure that have been used before. Finally, the packet traces part and its results are shown.

The conclusion, is presented at chapter six. Based on our data set and analyses we present the best choices for performing the speed test at different contexts that have been investigated.

Chapter 2

Background

The goal of this part is to explain a little bit about the architecture, in particular the one that the Fastweb is using, plus, a short description of the GPON access network. Also, it should be mentioned that, the FTTH technology is going to be used for our purpose, considering the fact that there are different types of technologies that a user can use like FO, ADSL, VDSL and etc.

We are almost trying to measure things that are happening away from the part of network that we control. First of all, we assume that the user at home is capable to use Fiber Optic technology. So, the user can be connected directly to an ONT/ONU or CPE device.

2.1 FTTH Access Network

The increasing demand for extra speed and bandwidth led to move to new technologies. Telecommunication operators have to upgrade their access networks that are clearly becoming the bottleneck in terms of bandwidth. Therefore most of them are replacing their old copper network technology, with optical fiber networks. The optical fibers are reaching to the doorstep of the client for faster speed, and Fiber To The Home appears the best choice [1].

2.2 PON

Passive optical network (PON) based FTTH access network is a point-to-multipoint and these networks are capable to provide all communication services including voice, data and video from one network platform[1].

PONs have a tree topology in order to maximize their coverage with minimum network splits.Because PON has no amplifires or regenerator, therefor decreasing optical power loss is crucial. There are three types of PON: Ethernet PON (EPON), Broadband PON (BPON) and Gigabit PON (GPON). For all of these models, two wavelengths are being used, one for downstream and one for upstream data traffic. There are two main multiplexing schemes using in the architecture, WDM and TDM and the last one is deployed for our purpose. In Time Division Multiplexed PONs

(TDM-PONs) the total bandwidth by time-sharing is available per user and it is limited, in particular if the connection is going to be used for CBR applications. Wavelength Division Multiplexing (WDM-PONs) for increasing the throughput can be utilised but this method is costly [2].
Some standards for TDM-PON are available at table 2.1 [1].

Parameter	BPON	EPON	GPON	XGPON	10G-EPON
Standard	ITU-T G.983	IEEE 802.3ah	ITU-T G.984	ITU-T G.987	IEEE 802.3av
Downstream data Rate	622 Mbps	1.25 Gbps	2.5 Gbps	10 Gbps	10 Gbps
Upstream data rate	155 Mbps	1.25 Gbps	1.25 Gbps	2.5 Gbps	10 Gbps/Symmetric , 1 Gbps/Asymmetric

Table 2.1: TDM PON standards

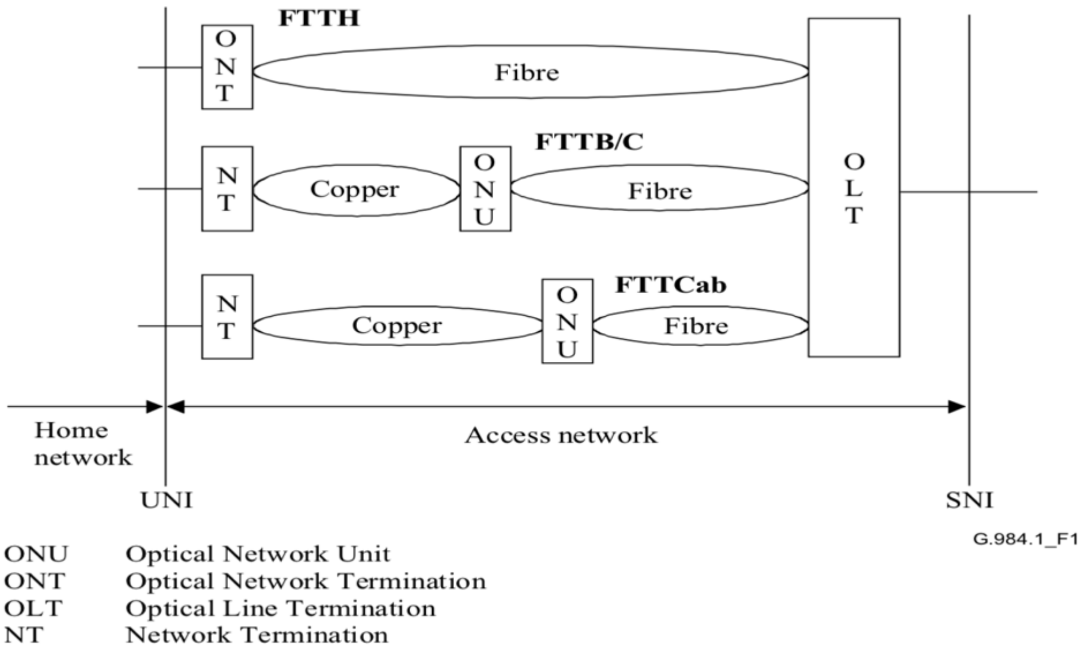


Figure 2.1: Architecture of the optical access network

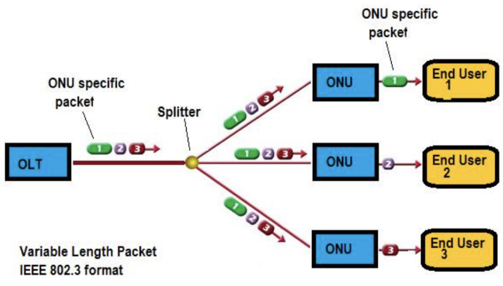
In the figure 2.1, UNI means User Network Interface and SNI means Service Node Interface. The above figure, illustrates, the architecture ranging from FTTH to Fibre to the Building/Curb (FTTB/C) and Fibre to the Cabinet (FTTCab).

The differences between FTTH, FTTB/C and FTTCab network options are due mainly due to the different services supported. Three services that FTTH provide are [3]:

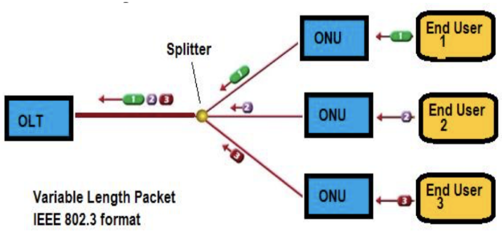
- Asymmetric broadband services (e.g., digital broadcast services, file download, etc.).
- Symmetric broadband services (e.g., content broadcast, online-game, etc.).
- POTS and ISDN

Managing Upstream/Downstream Traffic in PON

In PON the process of sending data downstream from the OLT to multiple ONUs is completely different from transmitting data upstream from multiple ONUs to the OLT. As illustrated in figure 2.2a, Each packet carries a header that uniquely recognizes it as data aimed for ONU-1, ONU-2 or ONU-3. Plus, some packets might be intended for all of the ONUs (broadcast packets) or a specific group of ONUs (multicast packets). At the splitter the traffic is divided into three distinct signals, each taking all of the ONU-specific packets. When the data reaches the ONU it admits the packets that are intended for it and discards the packets that are destined for other ONUs [4].



(a) downstream traffic flow in PON



(b) Upstram traffic flow in PON

Figure 2.2: Traffic directionality in PON

Figure 2.2b illustrates how upstream traffic is managed using TDM, in which transmission time slots are devoted to the ONUs. The time slots are synchronized in order to upstream packets from the ONUs do not conflict with each other once the data are joined into the common fiber [4].

2.3 GPON

The GPON architecture can be either point-to-point or point-to-multipoint. It can be described by an OLT, ONU/ONT, splitters and Optical Distribution Network (ODN) which interconnects these equipment [3].

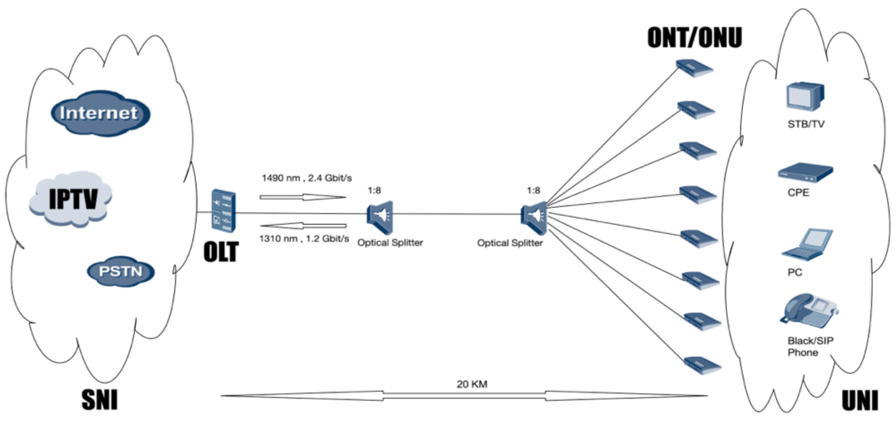


Figure 2.3: GPON access network architecture general view

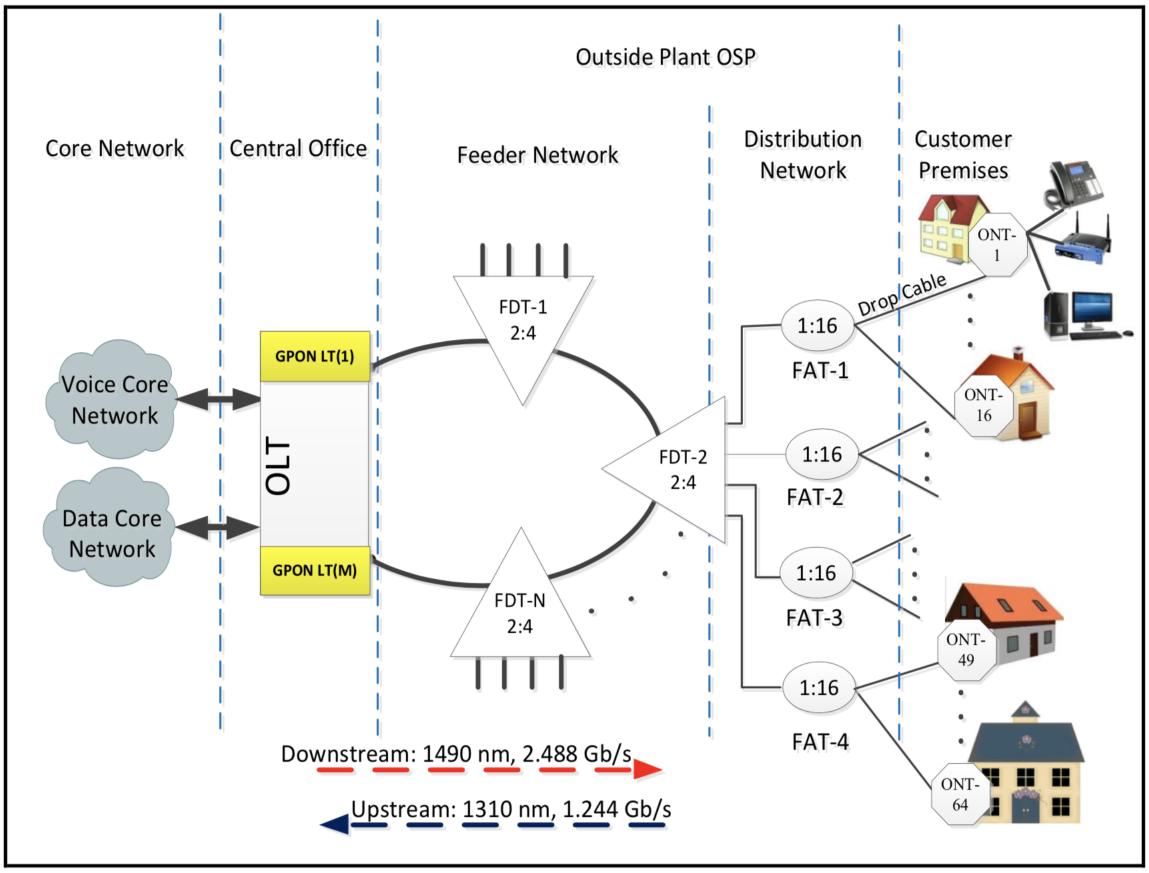


Figure 2.4: GPON FTTH access network architecture

A typical length between OLT and ONU is about 20 KMs, and The GPON intend to transmit at speed greater than 1.2 Gbit/s. For our purpose, the GPON aims to transmit 2.4 Gbit/s downstream and 1.2 Gbit/s upstream. It should be mentioned that, different wavelengths, 1310 nm and 1490 nm for upstream and downstream is going to be use respectively. Based on current technology, 1:64 split ratio could be realistic. Splitting can be centralized or cascade. Although, the second method is more attractive for operators, the first one is more efficient. Now, in short, the key parts of the GPON architecture are going to be explained. Also, the architecture that Fastweb uses is shown at figure 2.5 [1].

Optical Line Terminal OLT

OLT located at Local Exchange and it is the dominant element of the network. Actually, this is the engine that operate FTTH system. Traffic scheduling, buffer control and bandwidth allocation are the main operation of OLT [1].

Optical Splitters

The goal of this device is splitting the power of the signal. Inside each splitter, the fibers enter, and they might be splitted to a specific number of links which leave the splitter. Typically, two or three levels of fibers corresponding to two or more levels of splitter exist. This allow us to share the fibers between the users [1].

Optical Network Terminal ONT
This device is placed at client location. Without any active element through the link, ONTs are connected to OLT with optical fiber. In GPON the transceiver in the ONT is the physical connection between the customer premises and the central office OLT [1].

Optical Network Unit ONU
The user-side interface of Optical Access Network (OAN) is provided by ONU (directly or remotely) [3]. Here the optical signal is terminated before being further distributed to all the subscribers attached to this ONU via other media, such as copper wire, etc[5].

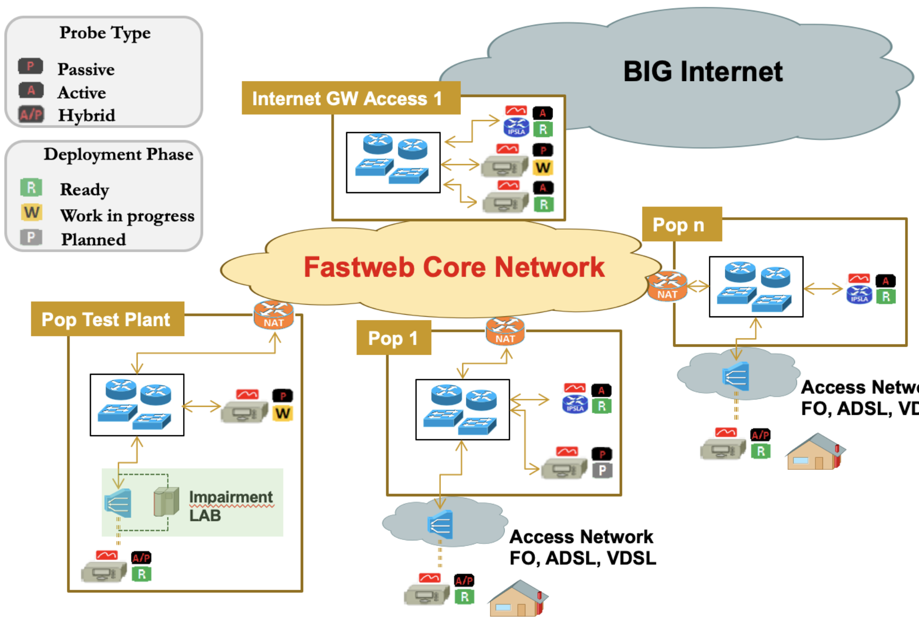


Figure 2.5: Fastweb architecture proposal

Backbone network
If we consider some PoP make a core network for a specific ISP in one region, connection of all core networks at different regions together, in total makes the backbone network of that ISP.

Internet access gate way
This node is the gate between the core network and the big Internet, from this node you can go to different ISPs or other Internet nodes through the IXPs, direct peering towards other ISPs, Data Centers, OTT provider and etc.

PoP
An Internet Point of Presence contains servers, routers, frame relays or ATM switches, multiplexers and other network interface equipment. With more PoP we could get better coverage.

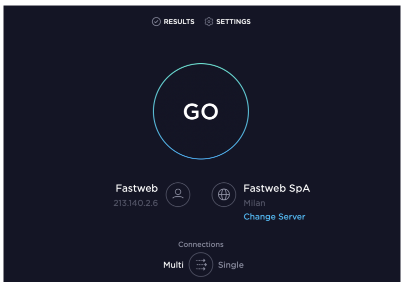
Access network
It is connecting the end user to core network. Some types of the access networks are: ADSL, VDSL, Wireless LANs, Fiber Optic. The one which is going to be use for our purpose is GPON.

Chapter 3

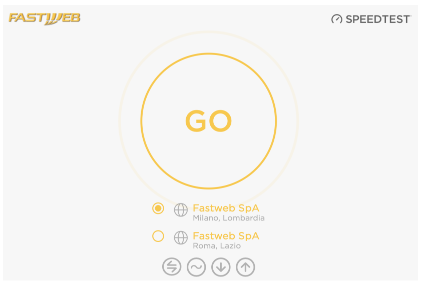
Ookla Speedtest

3.1 Ookla Speedtest Overview

Ookla Speedtest is a powerful and authentic testing method based on HTML5, Flash-free and supports both mobile and desktop browsers. Testing using open standards, including HTML5, JavaScript and WebSockets and it utilizes TCP. Ookla Speed test measures download speed, upload speed, ping (latency), and jitter.
The test front-end is managed by Ookla, either the usual one that all the people can use https://www.speedtest.net or at the sub domain of your choice (e.g. https://fastweb.speedtestcustom.com) [6]. During the thesis the first and second options will be mentioned by the name “public” and “private” respectively. Below their interfaces are shown. From now on the word platform is being used instead of interface.



(a) Public Platform



(b) Private Platform

Figure 3.1: Speedtest Platforms

3.2 Test Component

3.2.1 Ping/Jitter

At the beginning of the test, clients sends some packets to the server. At the time of receiving this information, the server replies back. By measuring the time it takes for the host to response to a request from the user's client, one test has been fulfilled. At the end, after some repetition, the smallest value regulate the final result, and the round-trip time is computed in ms (milliseconds) [7]. One should consider, Only the private platform gives the jitter value.

3.2.2 Download

1. Numerous connections with the server over port 8080/8181 set-up by client and it asks from the server to send a primary block of data.
2. The real-time velocity of the transfers is measured by client, then it modifies the chunk and buffer dimension based on this computation to boost usage of the network link.
3. During the time that blocks are collected by the client, more chunks are requested.
4. Throughout the first fraction of the experiment, the client will establish additional links to the host if it discovers extra threads are needed to more accurately measure the download speed.
5. The experiment finishes when the configured amount of time has been reached. [7].

3.2.3 Upload

There are a few differences with respect to computing download value, therefor just these differences are mentioned. At first step, client sends data, but without requesting a block of packets. At the third step, the chunks are received by server not client and also there is no request from client. [7].

3.3 One Measurement Sample

A measurement sample at front-end of the public platform, gives the test component that described before. The private platform also gives the jitter. At the back-end because Fastweb has got a perimum account in Ookla can get more information about each test that will be explained more in the following sections.

3.4 Speedtest Server Requirements

For installing Ookla speed test on a server there are some requirements that must be fulfilled.

3.4.1 Network Requirements

- Network Capacity
  - 1 Gbps Upstream and Downstream Capacity
  - Rare exceptions are made for underdeveloped and underserved regions.
- DNS
  - We require public DNS resolvable hostnames, IP addresses are not valid hostnames.
    - Example HTTP Legacy URL: http://sp1.domain.com/speedtest/upload.php - 201.12.28.12
    - Example OoklaServer URL: sp1.domain.com:8080-201.12.28.11
- Ports
  - TCP/UDP inbound/outbound port 8080 (OoklaServer)
  - TCP/UDP inbound/outbound port 5060 (OoklaServer)
  - TCP inbound/outbound port 80 (HTTP Legacy)
  - Optional TCP outbound port 80/443 to Ookla (optinal) speed test web site for updates and LE provisioning if enabled.
  - All ports are required to be open for any public internet IP as users will connect directly [8] .

3.4.2 Server Requirements

The following is required to become a Speedtest Server sponsor [8] :

	CPU	Memory	Network	Disk
Minimum	Quad Core	4GB	1Gbps	1GB
Recommended	Quad Core	8GB	2Gbps	1GB
1Gbps Testing	Dual Socket Quad Core	8GB	10Gbps	1GB

Table 3.1: Ookla Server Requirements

3.4.3 Supported Operating Systems for OoklaServer

- Server operating system [8] :
  - Windows Server (2008, or 2012 with IIS 6, 7, 7.5, or 8)
  - Linux (2.6.18 kernel or newer)
  - Mac OS X (built on 10.8, previous versions may function but are not fully supported)
  - FreeBSD (kernel 7.3 or newer)

3.4.4 Supported Web Servers

- Supported Web Servers / Server Side Code for HTTP Legacy Fallback [8] :
  - Web server software with PHP, ASP, ASP.NET or JSP support.
  - Examples: Apache, IIS, nginx, and lighttpd.
  - Make sure the Maximum POST request size is raised to 10 MB.
  - Keep Alive Enabled and Compression Disabled.
  - Administrator or root access might be needed if default server settings are inadequate.

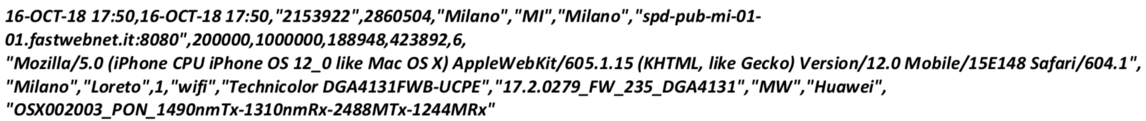
Chapter 4

Methodology

4.1 Dataset

4.1.1 Dataset overview

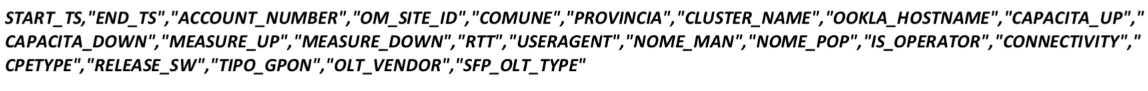
There are different fields in raw dataset that the most important ones are going to be explained.The raw dataset provides a means to gain superior knowledge about various wire and wireless services including Wi-Fi and cellular networks with the accuracy and unparalleled volume of Ookla's authoritative Speedtest.net [9, 10]. We changed the field name based on our interest. Also, there are some fields extracted from raw data for better data analysis like hour, OS and browser. Below is an example record:



```
16-OCT-18 17:50,16-OCT-18 17:50,"2153922",2860504,"Milano","MI","Milano","spd-pub-mi-01-
01.fastwebnet.it:8080",200000,1000000,188948,423892,6,
"Mozilla/5.0 (iPhone CPU iPhone OS 12_0 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.0 Mobile/15E148 Safari/604.1",
"Milano","Loreto",1,"wifi","Technicolor DGA4131FWB-UCPE","17.2.0279_FW_235_DGA4131","MW","Huawei",
"OSX002003_PON_1490nmTx-1310nmRx-2488MTx-1244MRx"
```

Figure 4.1: A sample record

the general header example in which the default names have changed:



```
START_TS,"END_TS","ACCOUNT_NUMBER","OM_SITE_ID","COMUNE","PROVINCIA","CLUSTER_NAME","OOKLA_HOSTNAME","CAPACITA_UP","
CAPACITA_DOWN","MEASURE_UP","MEASURE_DOWN","RTT","USERAGENT","NOME_MAN","NOME_POP","IS_OPERATOR","CONNECTIVITY","
CPETYPE","RELEASE_SW","TIPO_GPON","OLT_VENDOR","SFP_OLT_TYPE"
```

Figure 4.2: Header of first set of data

4.1.2 Dataset fields

In the following page, the main variables and those which are not clear are explained. It should be mentioned that, this data set is available at back-end, and someone who is just able to access the front-end information, cannot access to them. Normally, at front-end we see ping, download, upload and server location.

Field Name	Description	Data Type
START_TS END_TS	YYYY-MM-DD HH:MM:SS (defaults to Pacific Time but Ookla can accommodate different timezones)	datetime
OOKLA_HOSTNAME	name of the server tested	text
CONNECTIVITY	WIRE/WIFI	text
USERAGENT	Identifies OS and browser	text
IS_OPERATOR	Test has performed by Fastweb operator (1), Test has performed by ordinary user (0)	number
OM_SITE_ID	Client ID	number
PROVINCIA	Italy province	text
CLUSTER_NAME	Core network location	text
CAPACITA_UP	Upload maximum capacity	number
MEASURE_UP	Specific sample upload value	number
RELEASE_SW	CPE version	text
CPETYPE	CPE type(name)	text
RELEASE_SW	The CPE version	text
TIPO_GPON	Related GPON physical infrastructure used for the test	text
OLT_VENDOR	Manufacturer of OLT	text
SFP_OLT_TYPE	Type of SFP port of OLT devices	text
HOUR	The hour part of START_TS END_TS field	number
OS	Operating system name extracted from user agent	text
BROWSER	Browser name extracted from user agent	text

Table 4.1: Data set fields

OOKLA_HOSTNAME: There are different Ookla host name including two Fastweb core servers, and others not related to Fastweb, with much fewer samples in this field. An example of the sample of data in this field is "spd-pub-mi-01-01.fastwebnet.it:8080". We are just interested to the Ookla host name related to the core network of Fastweb in Milan and Rome.

USERAGENT: The User-Agent header field consists of a characteristic string, used by servers and allows the network protocol peers to describe the scope of the application type, operating system, software manufacture or software version of the requesting program user agent [11, 12].

IS_OPERATOR:When a Fastweb operator at the installation phase of connection at client home perform a test with his device to check the quality of the line, this value is set to 1, otherwise is 0 (The total in following figures means all specific samples including IS_OPERATOR = 1/0).

OM_SITE_ID: This is the Fastweb's client account number.

PROVINCIA: Each province is related to a specific core network.So, the provinces of Milan network is different from Rome.

CLUSTER_NAME: There are two cluster in the data set. One is related to the Milan core network and the other to Rome core network.

CAPACITA_UP: It is equal to 200 Mbps. This is the maximm speed that users at home can reach for their upload. This value in the lab is 1000 Mbps and the 200 Mbps limitation does not exist there.

MEASURE_UP: The specific upload value at each test.

RELEASE_SW: This field shows us the exact version of CPE.

OLT_VENDOR: There are just two company that their CPU has been used in the data set, Nokia and Huawei.

SFP_OLT_TYPE: Small Form-factor Pluggable,this is a small transceiver that attach in a network switch and connects to fibre channel and Gigabit Ethernet (GbE) optical fiber cables at the other end. SFP ports permit Gigabit switches to connect to a broad collection of fiber and Ethernet cables so as to increase switching performance throughout the network. The SFP works in single and multimode, it also allows switches to connect to various speeds (1 Gbps, 10 Gbps) . Modern Gigabit switch is usually designed with two or more SFP ports, allowing them to turn into a part of a ring or star-based network topology [13, 14].

CPETYPE: Five types of CPE are available in our data set. ASKEY01-UCPE, ASKEY02-UCPE, DN8245F-UCPE, Technicolor DGA4131FWB-UCPE andTechnicolor FGA2130FWB-UCPE_1.1. ASKEY01 and ASKEY02 samples, are aggregated to ASKEY, and the last two model, to Technicolor. The reason for sample aggregation is first of all, they are so similar, then having enough samples for comparing different CPE.

HOUR: Simply extracted from START_TS END_TS field, by help of HOUR formula in excel. This information is extracted, because we want to analyze data set based on different hours of the day. Obviously, at different hours, we have different amount of traffic in the network.

OS: This field gives us the operating system of a specific sample, based on the user agent information. First of all, the first part of user agent string is filtered and then by help of a simple python script, three different operating system is extracted from that. An example of user agent string:

"Mozilla/5.0 (iPhone CPU iPhone OS 12_0 like Mac OS X")

It should be mentioned that the Mozilla/5.0 is almost included at the beginning of all user agent information and it does not show the correct browser nor related to the type of operating system. Although, there are some user agent parser-which is used for parsing and extracting information about the browser- for this field we preferred to use a custom script (please see: A.2) for our interest which seems naive, but for this case, it is faster, plus, it gives the three different types of operating systems that we are going to check, including Windows, MAC OS and Linux.

BROWSER: Here, the data is not filtered, and a “Python” user agent parser (please see: A.3 [15]) which gives the browser information which is used. Moreover, the diversity of browsers is numerous, on the other hand, some browsers are used very rarely such as “QQ browser”, “Vivaldi”, etc. Also, some browsers, did not use in a specific OS, like IE on MAC OS. It is clear that, we should pay attention to the reliable amount of browser samples, not only based on their total amount, but also by considering the OS. As a result, we concentrate more on Chrome, Firefox, Safari and Edge (IE).

4.2 Tools

4.2.1 Selenium

There are many interests to test automation. Most are link with reiteration of the tests and test execution pace . A great portion of profitable and open source means are feasible. The one used for our purpose is “Selenium” which is open source. Selenium is built by different software tools and any of them has a unique character [16].

We wanted to perform one hundred tests for different situations for three major browsers at each operating system including Windows and MAC OS,but, manual testing is tedious. Different settings are:

- Fastweb core network location (Milan, Rome)
- Operating system (Windows, MAC OS)
- Type of connection (Wire, Wireless)
- Ookla speed test platforms (Public, Private)
- Browser (Edge, Chrome, Firefox, Safari)

What was needed was a straightforward mechanism to create rapid automation and in the same time capable, so these tests can be extended at least with little changes in other browsers [17]. The changes can be different methods for getting a web element, or for example ignoring elements like banners that comes in front of a desired web element in public platform of Ookla speed test interface. Also, it should be mentioned that, sometime it is not possible to use the same “XPATH” or “CSS selector” for different browsers. Basic procedure was similar for all test cases:

1. opening browser
2. waiting for the speed test page to be loaded
3. selecting the appropriate server
4. starting the test
5. waiting for the test to be finished
6. extracting and saving the test result
7. closing the browser

This procedure performed with using Selenium web driver and Python script. An example of python script for the test automation can be find at A.4 without considering iteration.

It must be considered that, the time of each test, is highly dependent on the speed of the connection, in the sense that, the test cannot be started exactly after opening the url. Therefore, for each particular browser and platform, we used a distinct sleep time to prevent script from crashing. Choosing correct timing for each situation is important for choosing the server, waiting during the test procedure and getting the result. As an example, it has been needed for public platform, to control these times in a good manner, because, we did not want to wait a lot of time for each test, besides that, as a consequence of removing irrelevant web elements, we should wait enough until the page could be loaded.
In total, for each location, operating system, connection and platform we performed 100 tests, just with exception of executing the test with Safari and Edge on Windows and MAC OS respectively. For example, one of our test condition was: (Rome,MAC, Wire, Private, Safari). So, at the end, after implementing all circumstances, in the lab environment, we had 4800 results. The architecture of test plant can be found at A.1.

4.2.2 Other tools

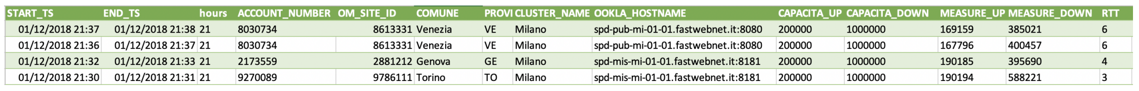
For the purpose of analyses and repeated task and tests some application are used. The first part of analyses has been done by the help of Matlab. For the test automation, as mentioned, Python helped us to use Selenium. Also, the last part of our analyses are performed by Python and in particular, with its modules like, Scipy, Numpy, Panda, Matplotlib and Seaborn. Moreover, Wireshark, the famous software for packet tracing is utilised.

Chapter 5

Results

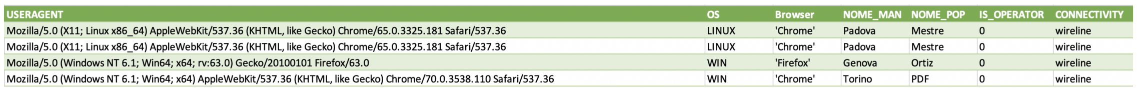
5.1 Data analysis of first data set

At this part we are going to describe and visualise the first set of data provided by Fastweb based on Ookla speed test measurements, from April to December 2018. The goal of this process is probably finding the main reasons that affect the results. The structure of data set is explained in previous section. Now, we analyse these data sets based on different field and their relation. Moreover, some statistics about the performance of network, according to this type of measurement and our available data will be shown. The data sets are divided in four main sets based on Fastweb core network location and type of connection. Also, it should be mentioned that, the main focus of analysis is on the Milan core network and wire connection but other situations investigated too. At this stage, also some results of Linux operating system probably is illustrated and considered, but when we start our automation speed test, just the Windows and MAC OS are checked. The number of available samples present at appendix.



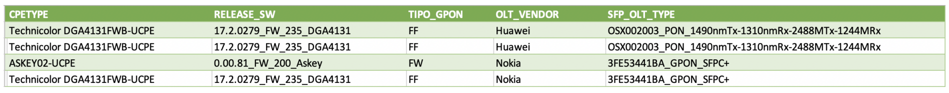
START_TS	END_TS	hours	ACCOUNT_NUMBER	OM_SITE_ID	COMUNE	PROVI	CLUSTER_NAME	OOKLA_HOSTNAME	CAPACITA_UP	CAPACITA_DOWN	MEASURE_UP	MEASURE_DOWN	RTT
01/12/2018 21:37	01/12/2018 21:38	21	8030734	8613331	Venezia	VE	Milano	spd-pub-mi-01-01.fastwebnet.it:8080	200000	1000000	169159	385021	6
01/12/2018 21:36	01/12/2018 21:37	21	8030734	8613331	Venezia	VE	Milano	spd-pub-mi-01-01.fastwebnet.it:8080	200000	1000000	167796	400457	6
01/12/2018 21:32	01/12/2018 21:33	21	2173559	2881212	Genova	GE	Milano	spd-mis-mi-01-01.fastwebnet.it:8181	200000	1000000	190185	395690	4
01/12/2018 21:30	01/12/2018 21:31	21	9270089	9786111	Torino	TO	Milano	spd-mis-mi-01-01.fastwebnet.it:8181	200000	1000000	190194	588221	3

Figure 5.1: Data set sample



USERAGENT	OS	Browser	NOME_MAN	NOME_POP	IS_OPERATOR	CONNECTIVITY
Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36	LINUX	'Chrome'	Padova	Mestre	0	wireline
Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36	LINUX	'Chrome'	Padova	Mestre	0	wireline
Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:63.0) Gecko/20100101 Firefox/63.0	WIN	'Firefox'	Genova	Ortiz	0	wireline
Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.110 Safari/537.36	WIN	'Chrome'	Torino	PDF	0	wireline

Figure 5.2: Data set sample (continue 1)



CPETYPE	RELEASE_SW	TIPO_GPON	OLT_VENDOR	SFP_OLT_TYPE
Technicolor DGA4131FWB-UCPE	17.2.0279_FW_235_DGA4131	FF	Huawei	OSX002003_PON_1490nmTx-1310nmRx-2488MTx-1244MRx
Technicolor DGA4131FWB-UCPE	17.2.0279_FW_235_DGA4131	FF	Huawei	OSX002003_PON_1490nmTx-1310nmRx-2488MTx-1244MRx
ASKEY02-UCPE	0.00.81_FW_200_Askey	FW	Nokia	3FE53441BA_GPON_SFPC+
Technicolor DGA4131FWB-UCPE	17.2.0279_FW_235_DGA4131	FF	Nokia	3FE53441BA_GPON_SFPC+

Figure 5.3: Data set sample (continue 2)

5.1.1 A general view of core networks

Here just by considering the impact of Milan and Rome core networks and type of connection, some statistics about them are illustrated. It should be considered that, the number available samples of Milan networks are greater than Rome.

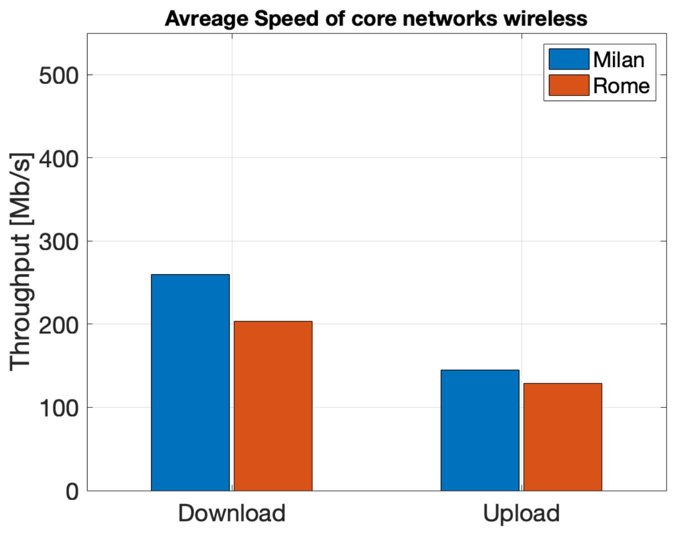


Figure 5.4: Average of total samples of wifi connection

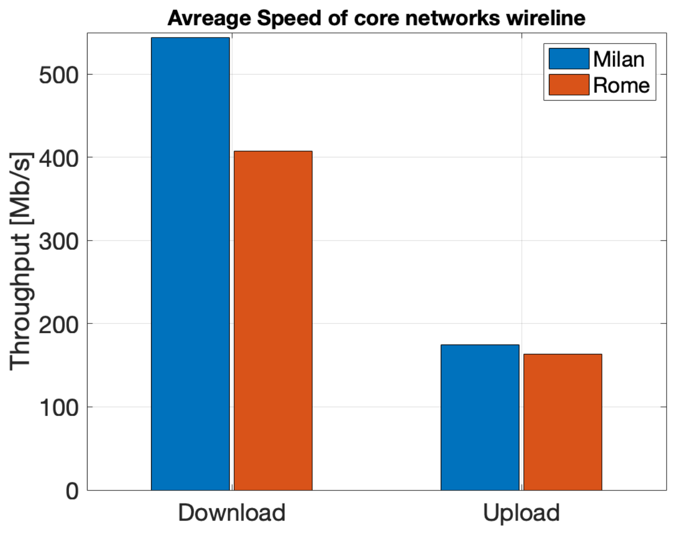


Figure 5.5: Average of total samples of wire connection

From now on, a great majority of our analyses and figure representation will be based on CDF. Other type of plots will be used like histogram and box plot where they can show us valuable information.

Cumulative Distribution Function (CDF)

This is a metric to represent the probability associated with random quantities. the CDF is defined for discrete random variables as:

$$F_X(x) = \mathrm{P}(X \leq x) = \textstyle\sum_{x_i \leq x} \mathrm{P}(X = x_i) = \sum_{x_i \leq x} p(x_i)$$

F(x) accumulates all of the probability less than or equal to x. The CDF of a continuous random variable X describes as the integral of its probability density function f_X. So, just the summation must be replaced with integral:[18]

$$F_X(x) = \textstyle\int_{-\infty}^{x} f_X(t)\,dt$$

Histogram

This function m_i calculates the number of observations that fall into each of the separate level, recognized as bins. Considering k and n the total number of bins and observations respectively, the histogram m_i has the following conditions: [19]

$$n = \textstyle\sum_{i=1}^{k} m_i$$

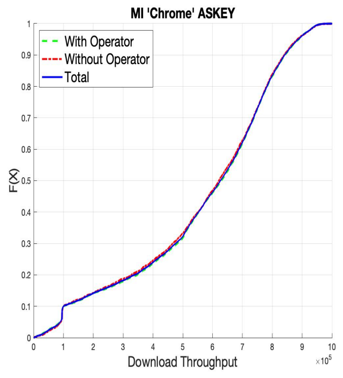
CDF vs Histogram

Histogram shows the frequency of each value in the data and it is very useful for finding the distribution of the data. Also, it seems that, Histogram is easier to be explained, but CDF provides more benefits (at least for our case). One of this advantages is, the main properties like minimum, maximum, median, quantiles and percentiles can be extracted from CDF graph, which is not be read from histogram. Moreover, comparing the differences of two or more distributions could be difficult with histogram. On the other hand, CDF is an applicable option. Other factors that, in particular for our analyses affect the result and visualization, are the number of samples and size of bins. Different sample size can mislead us for comparing various situations. Besides that, based on the bin size, it is possible to get different distribution [20, 21]. With this background, the CDF plots are used most of the time for our purpose.

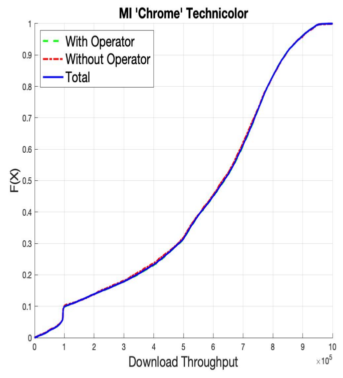
5.2 Impact of CPE and distance

The goal of this part is to show, when we have enough samples ,there is not a big difference related to geographical area, different CPE and also if the test is accomplished by a user or an operator. Different operating systems did not consider here. The influences of operating system and browser will be discussed later. To make this section short, just the figures for Milan and Turin for Chrome are shown. Download and upload are shown at the scope of Kbps and Rtt in ms. We have similar behavior for this kind of comparison for Milan Wifi and Rome Wireline/Wifi samples too. Also, we filtered our data hourly, which means we separated the samples from midnight to midday and vice versa, which does not show impressive effects. For this reason, the amount of graphs are not equal to available ones in this assessment. So, it is illustrated by download figures from two cities, and different CPEs are ASKEY, Technicolr and DNF from left to right. Then, similar structure for upload has been used.

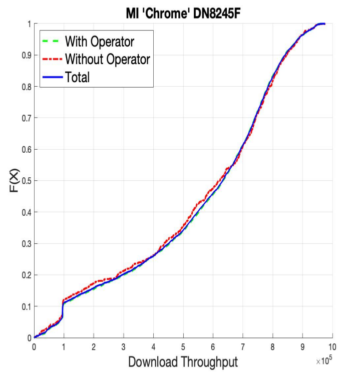
CPE comparison Milan



(a) ASKEY

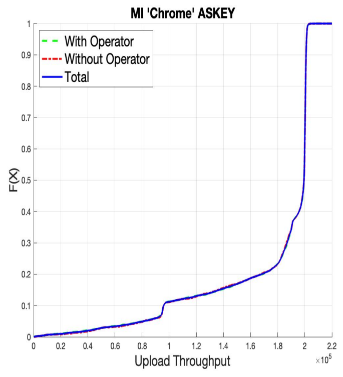


(b) Technicolor

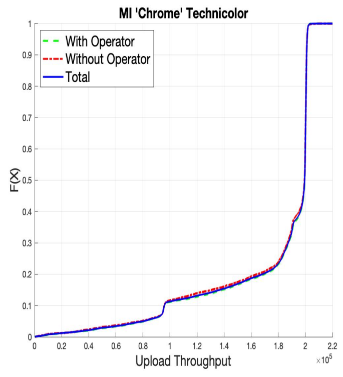


(c) DNF

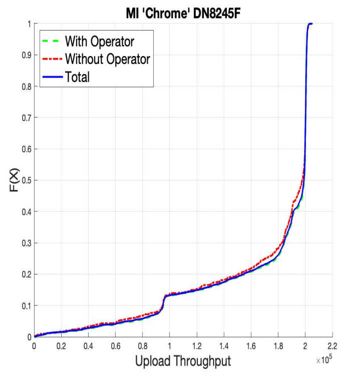
Figure 5.6: CDF Comparison of Download throughput for three CPE - Milan



(a) ASKEY

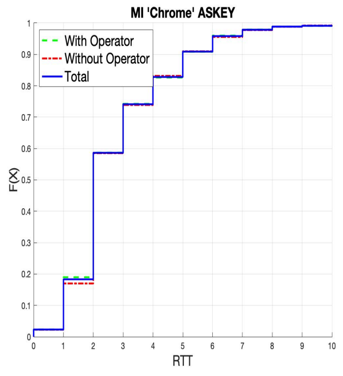


(b) Technicolor

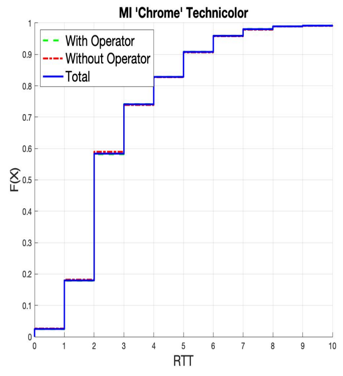


(c) DNF

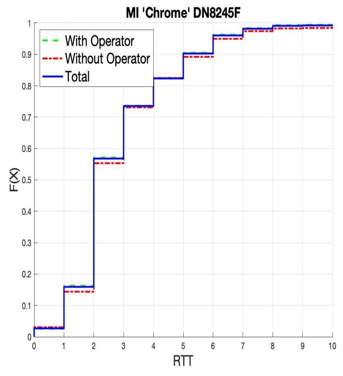
Figure 5.7: CDF Comparison of Upload throughput for three CPE - Milan



(a) ASKEY



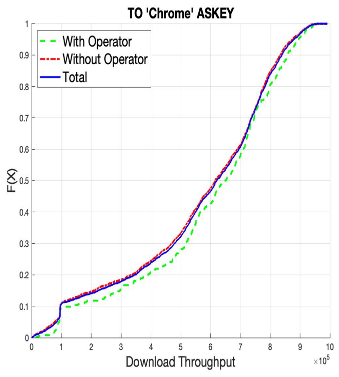
(b) Technicolor



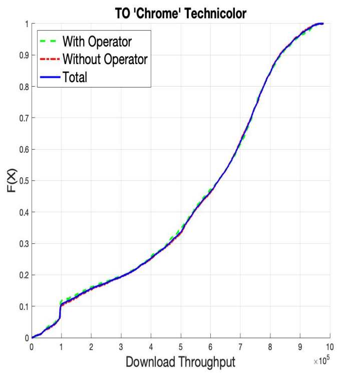
(c) DNF

Figure 5.8: CDF Comparison of RTT for three CPE - Milan

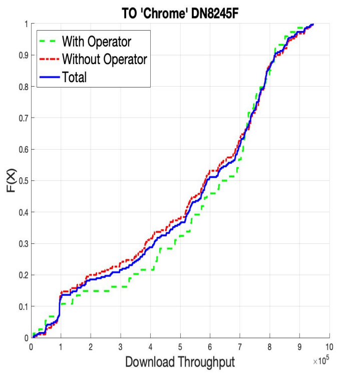
CPE comparison Turin



(a) ASKEY

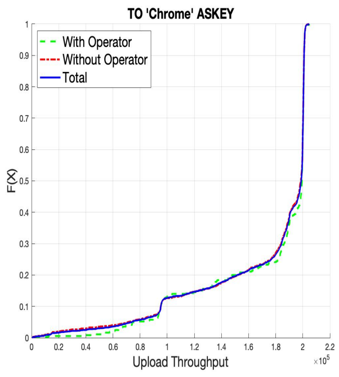


(b) Technicolor

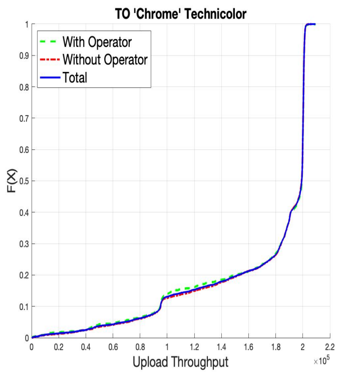


(c) DNF

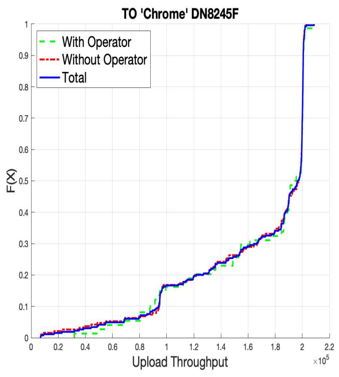
Figure 5.9: CDF Comparison of Download throughput for three CPE - Turin



(a) ASKEY

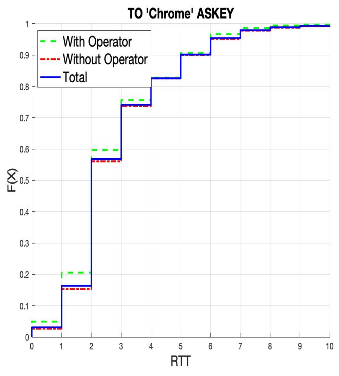


(b) Technicolor

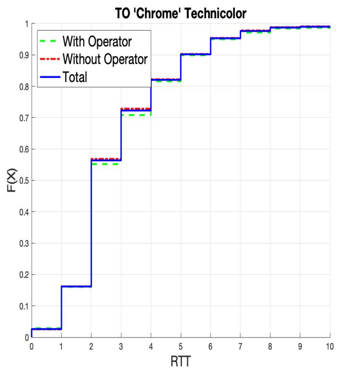


(c) DNF

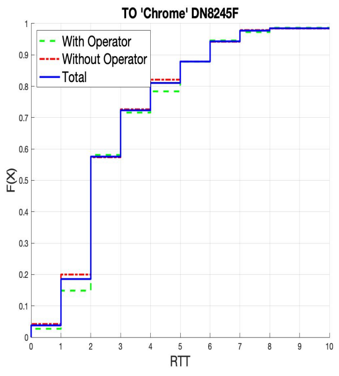
Figure 5.10: CDF Comparison of Upload throughput for three CPE - Turin



(a) ASKEY



(b) Technicolor

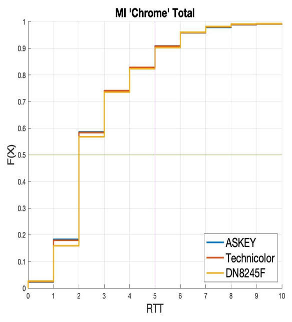


(c) DNF

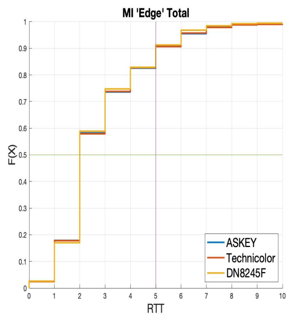
Figure 5.11: CDF Comparison of RTT for three CPE - Turin

5.2.1 Comparing Rtt based on different CPE and distance

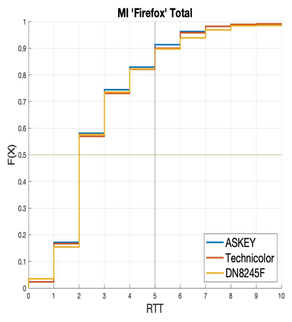
Here the Rtt figures for three CPE and cities are being shown.Also, this time samples are separated based on three major browsers Chrome, Edge and Firefox, left to right. In this way, it can be seen better how they behave similarly. These figures like previous section, are extracted from wire connection and Milan core network.



(a) Chrome

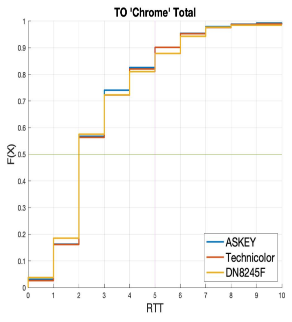


(b) Edge

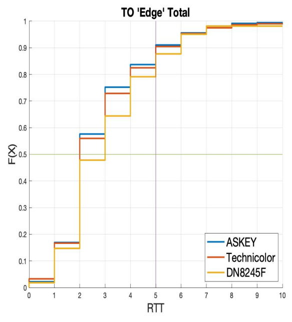


(c) Firefox

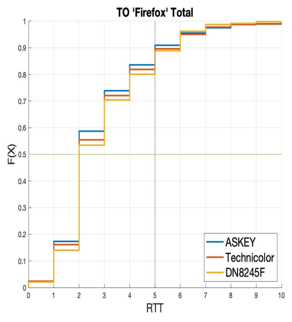
Figure 5.12: CDF Comparison of Rtt - Milan



(a) Chrome

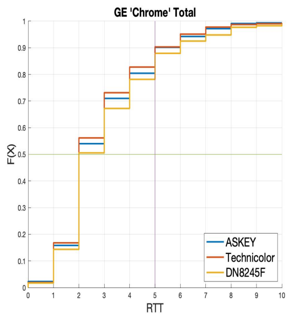


(b) Edge

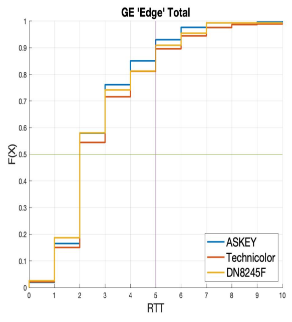


(c) Firefox

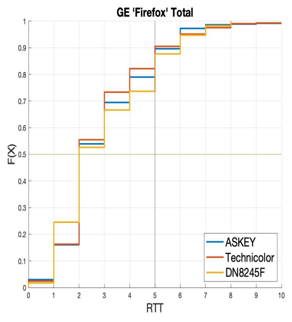
Figure 5.13: CDF Comparison of Rtt - Turin



(a) Chrome



(b) Edge



(c) Firefox

Figure 5.14: CDF Comparison of Rtt - Genova

At appendix (see A.5), a short statistical output of the available samples, for three browsers and Just for Milan and Turin is available. This type of results are available in general for 5 browsers, and all the provinces in the data set.

5.2.2 Comparing download/upload based on Rtt and time

At this part the relation between Download/Upload throughput versus Rtt and the time is shown, in which the time is divided in two section, from 00:00 A.M. to 12:00 P.M. and 12:00 P.M. to 00:00 A.M which will be indicated as A.M. and P.M. respectively. At the left side, the scatter plots show how the samples distributed during the time. At the right we see the CDF figures which show us the changes are so similar. It should be mentioned that, the differences in CDF plots for this figures are so close, and because the subplots are small these differences can not be seen, but in scatter plots, the differences are a little bit visible. To make this section as short as possible, the figures just for Milan and Chrome and the ASKEY CPE are shown (for similar figures please see A.6). Also we can see that in the same location, with equivalent OS and operating system, during the different hours of a day, there is not big changes in the CDF of download and upload. The scatter plots shows us even in the time that the traffic is high we can reach to maximum speed. So, we should focus on other factors.

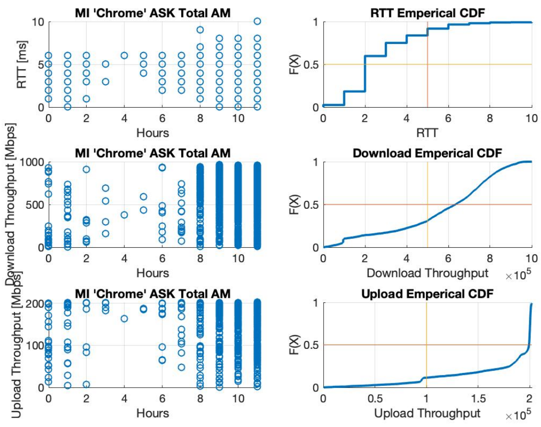


Figure 5.15: ASKEY download/upload comparison based on Rtt and time - A.M.

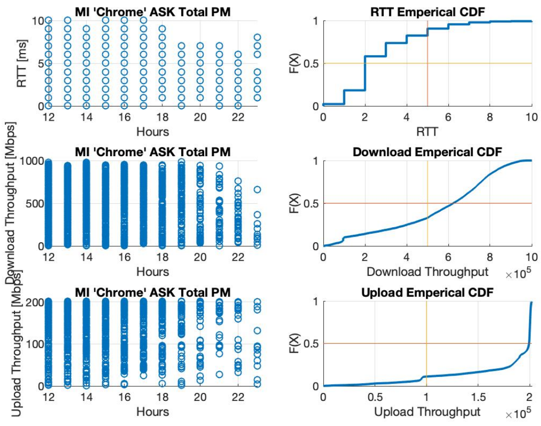


Figure 5.16: ASKEY download/upload comparison based on Rtt and time - P.M.

5.3 Impact of operating system and browser

Finally, we reached to the most important factors, that affected the results, including download, upload and Rtt, but the first two ones show this changes better. As mentioned at section 5.2, the results where enough number of samples are available are going to be shown. We have very small amount of samples in some cases, specially for Linux operating system. Some browsers, either do not have sufficient number of samples too, as an example Safari in Linux, or there is no sample for Edge or IE in MAC OS or Linux and similarly Safari in Windows. So, we tried to choose browser that most of the times are available in all operating systems. It should be mentioned that, in the data set, we cannot understand from a specific sample, those specific client network card, allow him/her to reach the speed more than 100 Mbps or not. Moreover, there are some samples, in which the download or upload value has passed the maximum available speed. So, we filtered this data out. Although, we can understand, the OS type of each sample, obviously it is not clear the capability of that machine, specially the network card. Moreover, the browser information at this point does not give us a view that, a specific client uses the last available version of browser. Considering these factors and also the point that, in all the cases the number of samples are not equal, we just investigated our data to prove that the main reason for poor or excellent performance is related to OS and browser. The plots are comparing the behavior of different browsers but at the same OS. One main reason, except the fact of lack of samples was, the difference in system configuration. With this background, after visualisation of some results to illustrate the impact of OS and browser, at the section 5.4.2 we just focus on main OS and browser which are more interesting for the company to be investigated. Below the table of available samples is listed, and the complete tables can be find at section A.4. The table 5.3 is presented after some filtering.

Core Network	**Number of Samples**
Milan	127850

Table 5.1: Milan wire connection total samples

Operating System	**Number of Samples**
Windows	114490
MAC OS	5890
Linux	5839

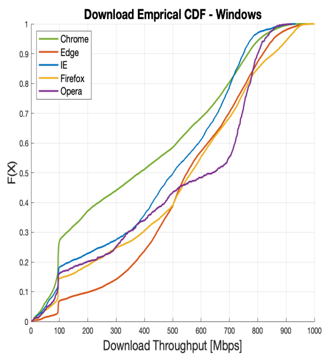
Table 5.2: Milan wire connection OS samples

	Edge	**Safari**	**Firefox**	**Chrome**	**IE**	**Opera**	**Total**
Windows	21890	0	14649	31687	5308	902	74436
MAC OS	0	3266	702	1500	0	52	5520
Linux	0	19	5003	292	0	45	5359
Total	21890	3285	20354	33479	5308	999	85315

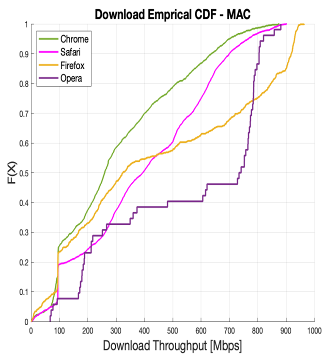
Table 5.3: Milan wire connection browser samples

5.3.1 Comparing download/upload/Rtt based on OS and browser

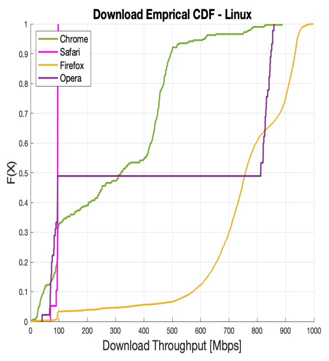
All the figures are plotted when we have enough number of samples for major browsers. Also, the throughput threshold is plotted, in which company considers that, if the users measure a value greater than or equal 500 Mbps (Download), 100 Mbps (Upload) and less than or equal 5 ms (Rtt), they are at good condition. Little amount of samples in Linux, for download and upload gives those vertical and horizontal line. These figures are related to Milan wire connection.



(a) Windows

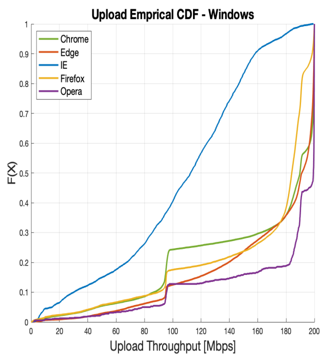


(b) MAC OS

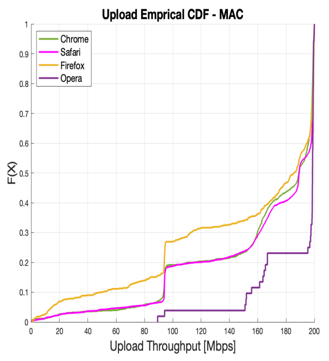


(c) Linux

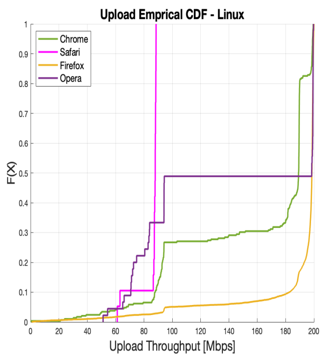
Figure 5.17: Windows, Mac and Linux download emprical CDF



(a) Windows

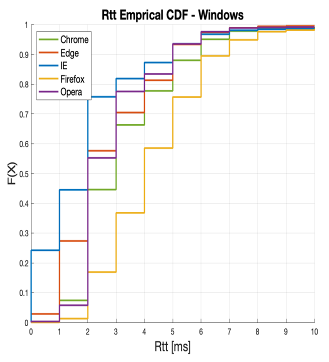


(b) MAC OS

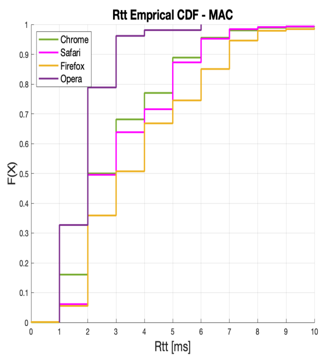


(c) Linux

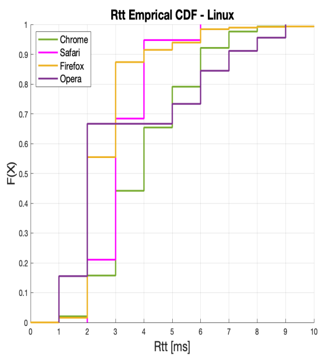
Figure 5.18: Windows, Mac and Linux upload emprical CDF



(a) Windows



(b) MAC OS



(c) Linux

Figure 5.19: Windows, Mac and Linux Rtt emprical CDF

In the previous figures, it can be seen that, for download and upload, when the speed is less than 100 Mbps, the distribution are so close. For better visualization, the sample where download and upload are less and greater than 100 Mbps are separated here. Left to right less and greater than 100 Mbps.

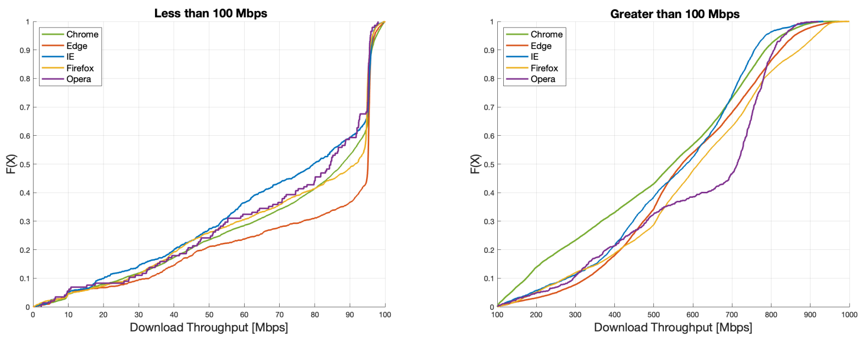


Figure 5.20: Milan CDF plots for less/greater than 100 Mbps - Windows - Download

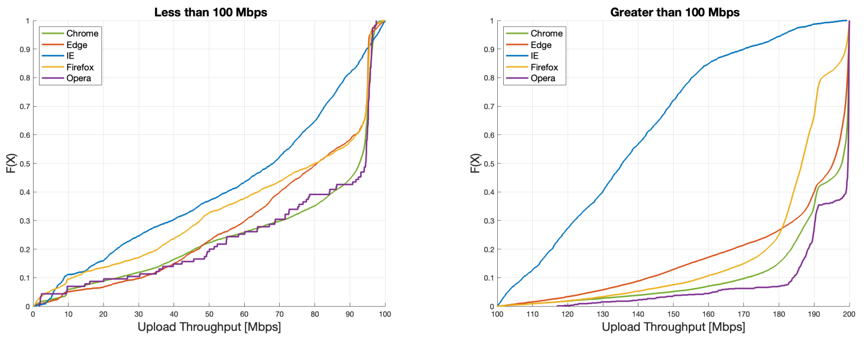


Figure 5.21: Milan CDF plots for less/greater than 100 Mbps - Windows - Upload

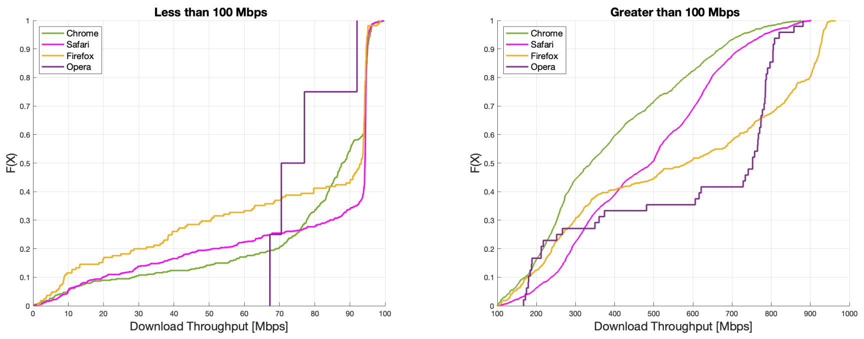


Figure 5.22: Milan CDF plots for less/greater than 100 Mbps - MAC - Download

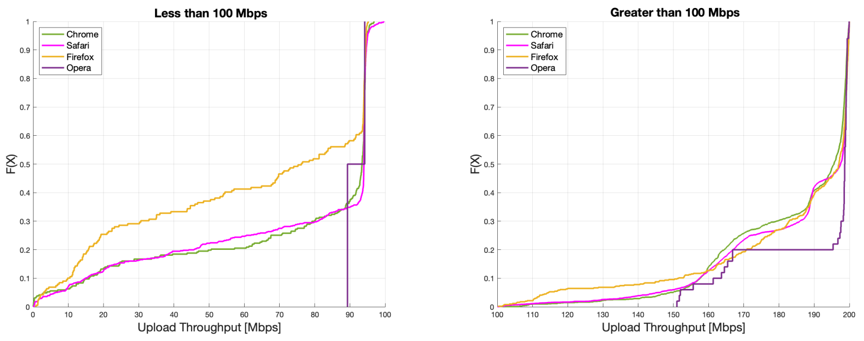
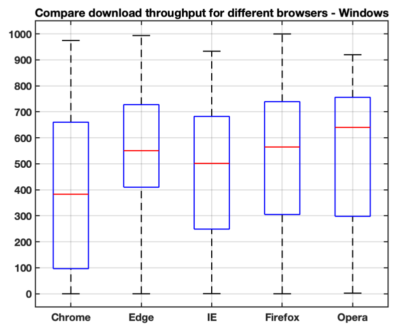
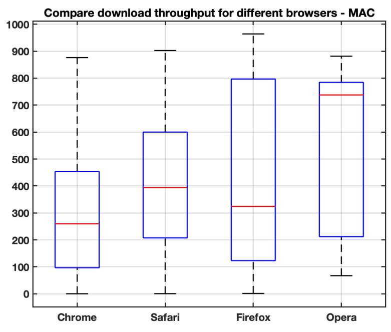


Figure 5.23: Milan CDF plots for less/greater than 100 Mbps - MAC - Upload

If we just look at the graphs of previous page, without considering the influence of number of samples, based on our data set and analyses, it can be stated that, for measures with speed less than 100 Mbps, plots distributed in the same fashion, both for download and upload. On the other hand, for speed greater than 100 Mbps, the differences are more visible. Moreover, the trends of browser are so different for speed greater than 100 Mbps, and the diversity of contrasting styles is further conspicuous. In all respects, these figures lead us to main aspect. After understanding the main circumstances of changes in our results, the fair comparison of browsers in Windows and MAC OS, which are the favorite OS of clients, will be carried out.

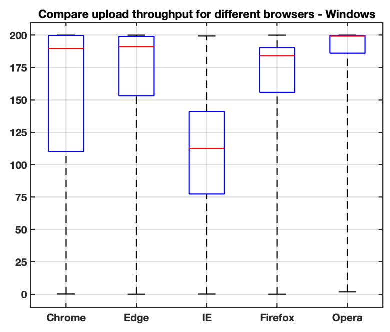


(a) Windows

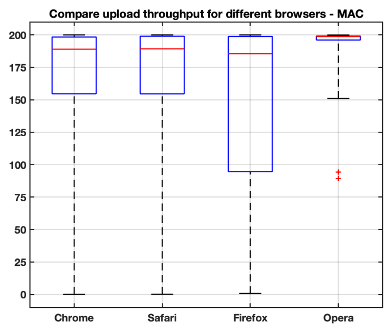


(b) MAC OS

Figure 5.24: Browser throughput comparison - Milan wireline - Download



(a) Windows



(b) MAC OS

Figure 5.25: Browser throughput comparison - Milan wireline - Upload

As previously stated, all the clients upload speed is limited to 200 Mbps, and because of this reason this similarity in their trend is exist. Opera has not got enough number of samples, so it can mislead us to a false decision.

5.4 Test automation

Unequal number of samples, plus the ambiguous device and browsers capability in the data set, can cause to inequitable comparison. As a result, we decided to repeat measurements with an admissible range, for main conditions. These tests took a lot of time, because we had one 1 Gbps GPON connection at the lab. So, it was not possible to perform the tests in parallel. Each test took about 1 minute. All the tests have been performed from 10 A.M. to 5 P.M. . The lab environment is isolated and the samples for wifi have gathered with lowest interference and noise, moreover, the devices were so close to CPE. Technicolor CPE, model FGA2130FWB is used for our tests, because in our first set of data this CPE had the most number of samples.

5.4.1 Main goal

Now, we have equal samples, we can compare the main factors that affect the Ookla speed test measurements at application layer. Not only a great difference in network quality can be seen, but also, different trend in CDF plots of important browsers. These changes, helped us to focus in detail on main browsers at specific situation. Here, we do not have any limitation for upload like before, so they can be compared better too. Although, we know our device capability at each test situation, but the goal of our comparison is based on browser not operating system. The main reason is each operating system uses different TCP congestion control, which is the main protocol Ookla uses for the test.

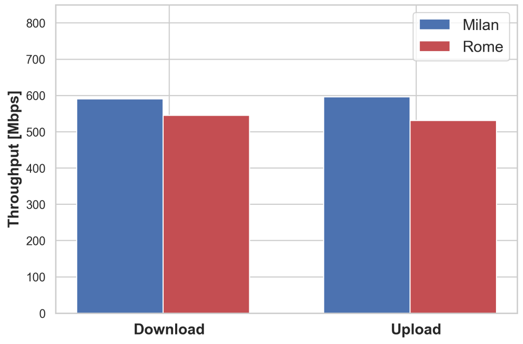


Figure 5.26: Average of test samples of wifi connection

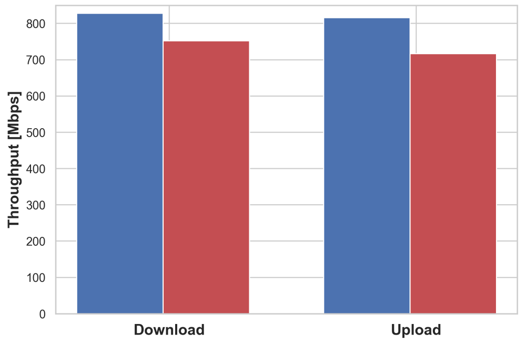


Figure 5.27: Average of test samples of wire connection

5.4.2 Results assessment

Based on our tests and outputs, some evaluation can be conducted. Firs of all, The trends of graphs show us the clear differences of using different browser at each condition. Now, the vertical lines are because of stability of the results. The upload figures do not have similar behavior like before.

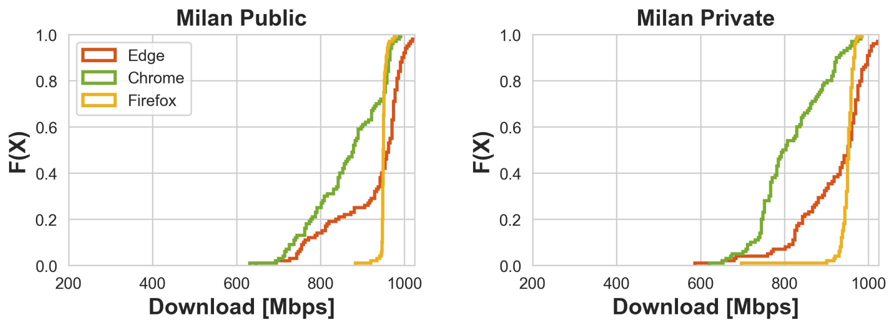


Figure 5.28: Test samples CDF plots of Wire Connection - Milan - Windows

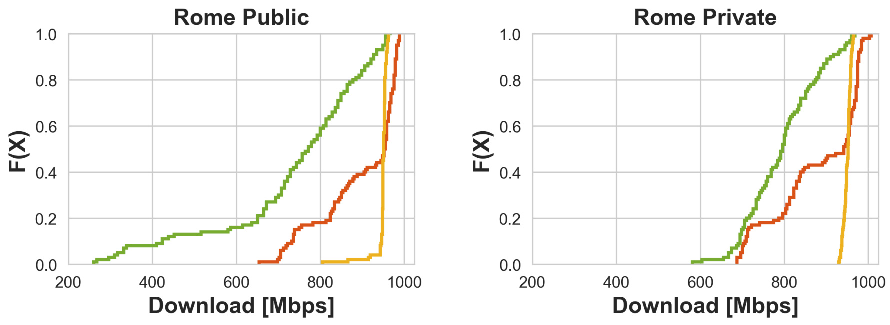


Figure 5.29: Test samples CDF plots of Wire Connection - Rome - Windows

As illustrated at section 5.3.1 and figure 5.20 for speed greater than 100 Mbps, even there can be seen the impact of browser in performance, but it did not show us that, Firefox is the best choice for download in Windows for wire connection. But here after tests in the lab, the results tell us that clearly Firefox worked perfectly.

The perfect behavior of Firefox does not exist for upload in Windows,which can be seen at figures 5.36 and 5.36. Also, when Firefox used at MAC OS, for both download and upload, this great difference at different operating system (MAC OS) for Firefox is clearly visible at figures 5.30, 5.31, 5.38 and 5.39 . On the other hand, even though, Edge was a good competitor at download for Firefox, But for upload in Windows (figures 5.36 and 5.37), it had the best performance.

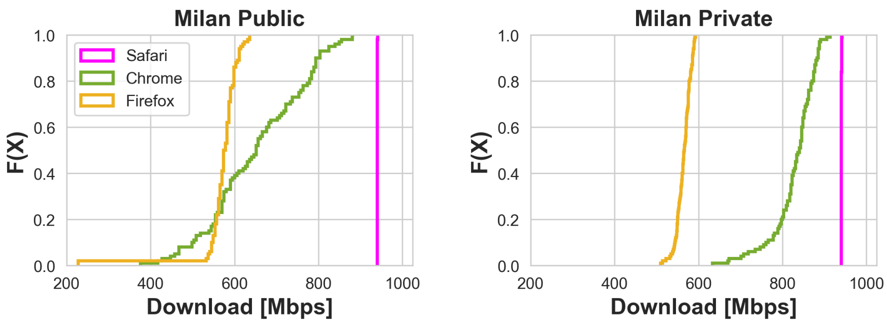


Figure 5.30: Test samples CDF plots of Wire Connection - Milan - MAC OS

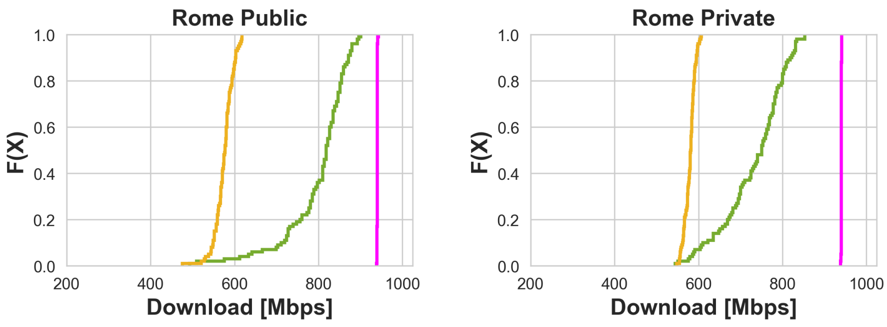


Figure 5.31: Test samples CDF plots of Wire Connection - Rome - MAC OS

One of the big differences with respect to figure 5.22 is the efficiency of Safari. The charming constancy of this browser for wire connection and the fantastic average speed for download, which was about 940 Mbps, without doubt tell us this is the best option for MAC OS user at least for downstream and wire connection.

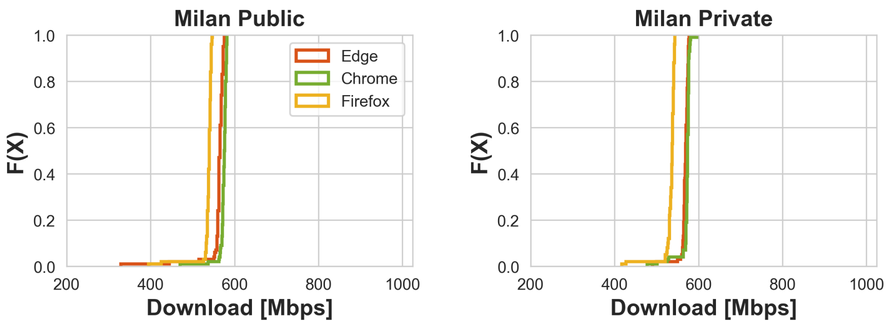


Figure 5.32: Test samples CDF plots of Wifi Connection - Milan - Windows

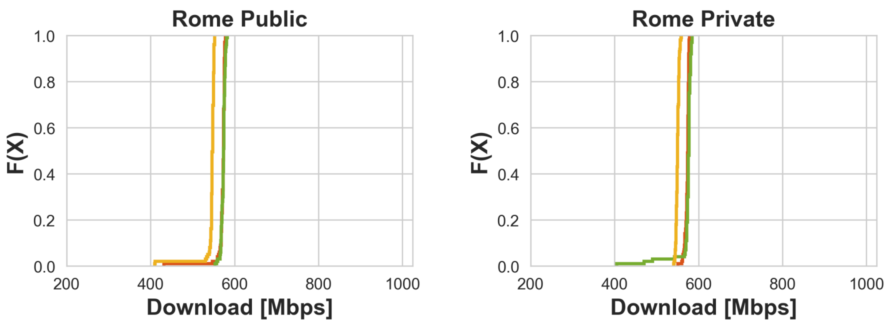


Figure 5.33: Test samples CDF plots of Wifi Connection - Rome - Windows

Unfortunately, this behavior could not be seen in previous data set, even the mean of Safari was better than Chrome and Firefox, but it did not reach to the half of its capability, also most of our client were using Chrome unluckily.

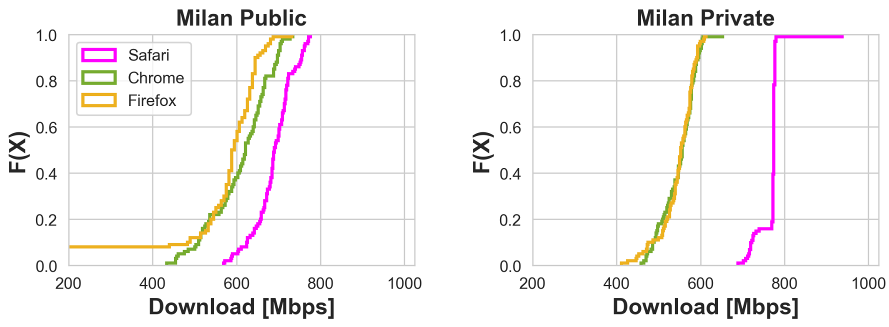


Figure 5.34: Test samples CDF plots of Wifi Connection - Milan - MAC OS

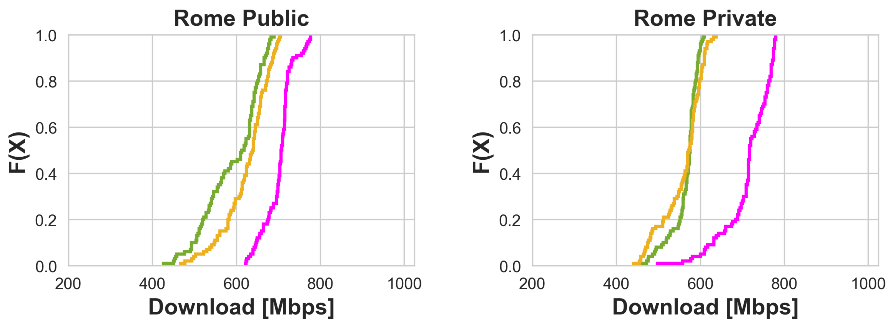


Figure 5.35: Test samples CDF plots of Wifi Connection - Rome - MAC OS

Other factor that should be mentioned is, the variation of results for Chrome in most of the cases which does not seem a good feature.Also, it was not a superior browser for a majority of conditions, except for upload for the test which performed towards Rome server in MAC OS and it is illustrated at figure 5.39 for Private platform with wire connection. Even in the public platform of Ookla, Safari and Chrome are so close in performance, so, it is possible to get similar result if we repeated our test for private interface.

For wifi, when the Windows used as the operating system, the browsers do not show us that a specific browser is a head and shoulder above the other one and also they cannot reach to speed greater than 600 Mbps. Although, we used the best available device, but unfortunately our network card capability or maybe the lap top did not allow us that the tests gives us the results in a way that we can compare the browser truly. Their similar trend can be seen at figures 5.32 and 5.33.

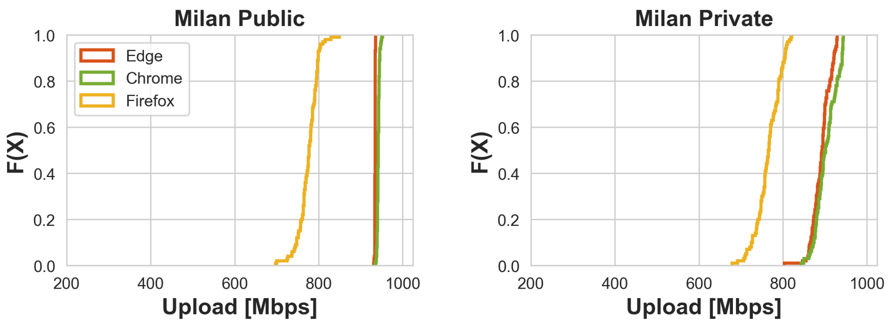


Figure 5.36: Test samples CDF plots of Wire Connection - Milan - Windows

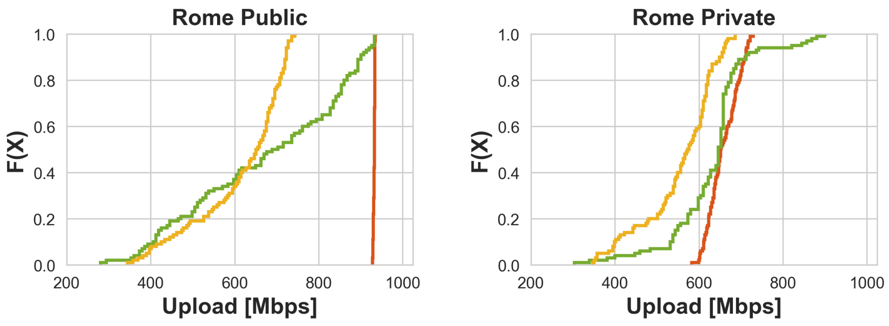


Figure 5.37: Test samples CDF plots of Wire Connection - Rome - Windows

But, the powerful MACbook Pro did not have this problem and again Safari was the best browser for upload for wireless connection, despite the fact that, it did not offer stability once more.

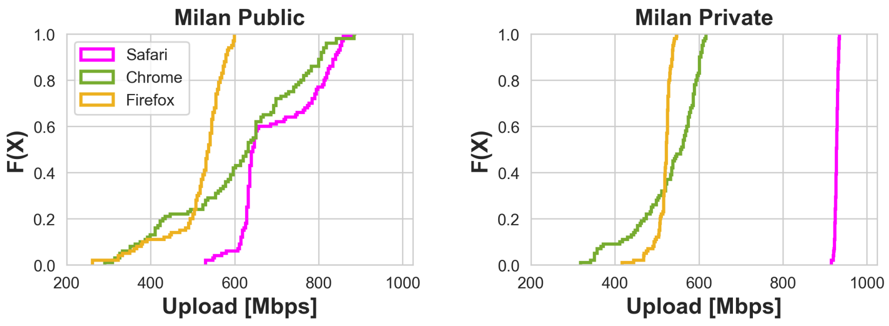


Figure 5.38: Test samples CDF plots of Wire Connection - Milan - MAC OS

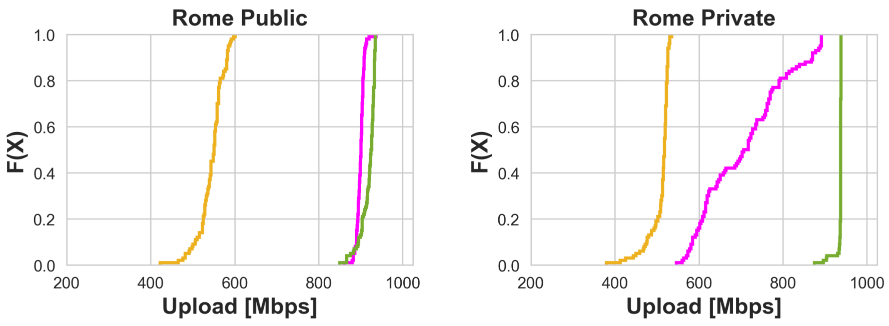


Figure 5.39: Test samples CDF plots of Wire Connection - Rome - MAC OS

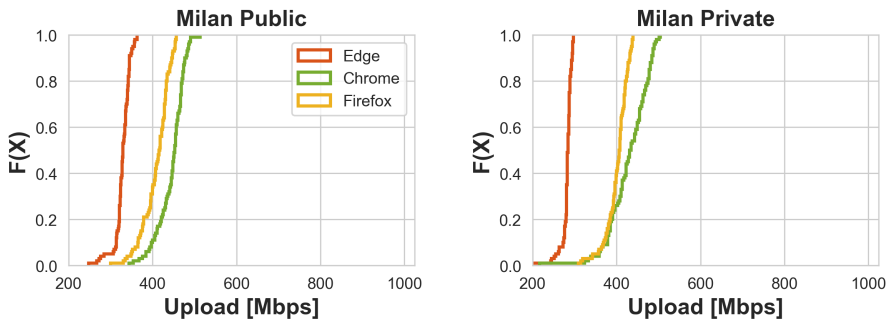


Figure 5.40: Test samples CDF plots of Wifi Connection - Milan - Windows

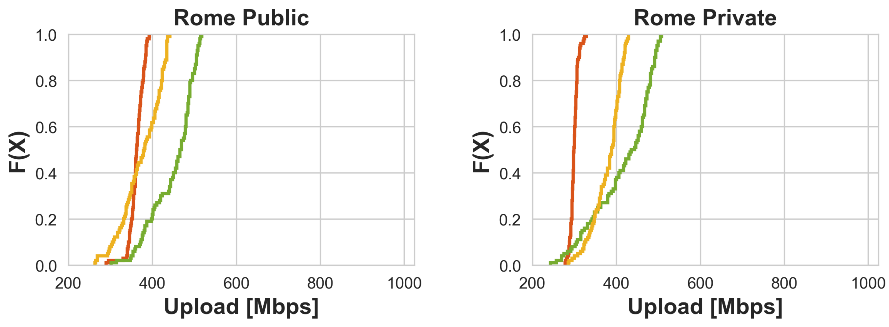


Figure 5.41: Test samples CDF plots of Wifi Connection - Rome - Windows

At wireless connection for upload, Edge did not give us the best performance, and approved again that, better performance in download does not guarantee finer efficiency in upload. Moreover, the same condition exists for safari. As can be seen in figures 5.42 and 5.43, now Chrome is a solemn rival. In short, device capability has more effects for wireless connection and in particular for upload.

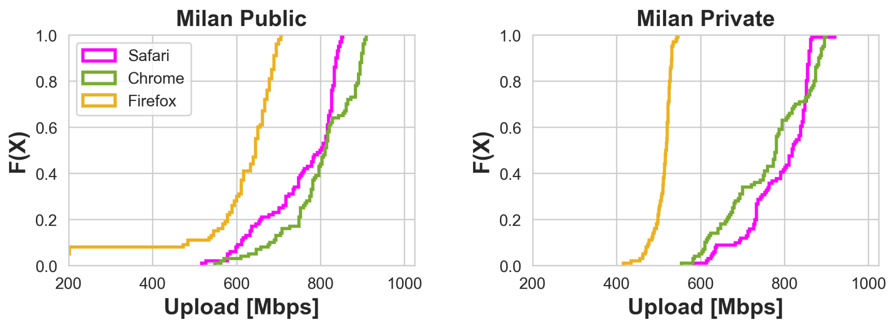


Figure 5.42: Test samples CDF plots of Wifi Connection - Milan - MAC OS

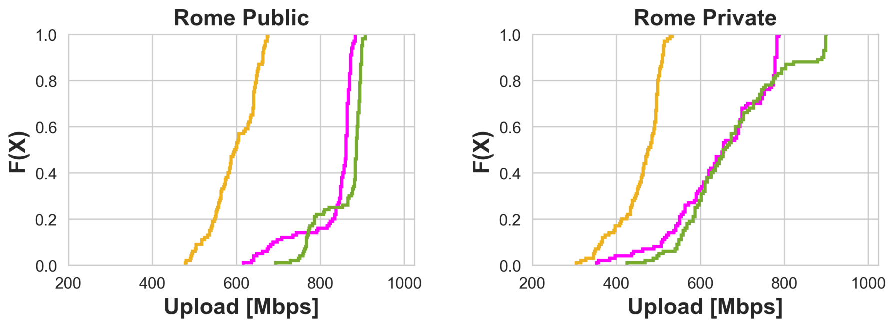


Figure 5.43: Test samples CDF plots of Wifi Connection - Rome - MAC OS

It has been told that, we are not interested to compare browsers in different operating system, but it is better to point that, for example, there is a substantial change in workmanship of Firefox in different operating system. We also tried to, at least find the reason for these behaviors.

5.5 Packet tracing

As we saw in previous section, different browsers give us different performance, even at same operating system. Some effort have been done to find the reasons of these differences of behavior. At this stage, we did not repeat the test for packet tracing a lot of time, but we tried to investigate the Wireshark traces in most of the cases, when they are at the same condition with the test results. Obviously, sometimes for upload or download the results of this part are not close to the average behavior of previous part, but because of our time limitation we could not repeat the tests a lot. For each browser at each scenario, we have a Wireshark trace. Statistics about this tests are available at A.8 . We focus on main connection between the IP address of our device and the Fastweb core networks in Milan and Rome for Public and Private platform with port numbers 8080 and 8181 respectively. Clearly, client uses random port numbers for TCP connections. For each test, the total TCP streams of Wireshark has been collected in an Excel file and the main ones which are related to speed test is also extracted. The figures that are provided based on I/O graph capability of this software. Because, it is not possible to use this option of Wireshark for different traces, first the numbers of each I/O graph is extracted, then with the help of Python we put all the figures in one graph to show their differences better.

Below an example of TCP streams just during the test for the case of Safari in MAC OS and wifi connection for private platform is presented. For the sake of simplicity and finer visualization some fields are removed. A means client and B server.

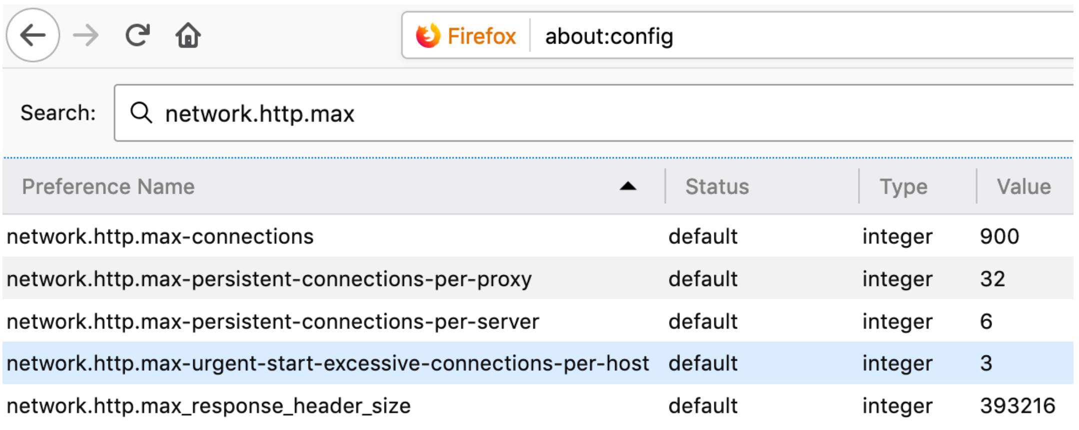
Port A	Port B	Packets A $\rightarrow$ B	Packets B $\rightarrow$ A	Real Start	Duration
56869	8181	43	27	0.638233	0.502936
56870	8181	17470	117777	3.096478	15.125856
56871	8181	30510	213840	312.596	15.097437
56872	8181	18098	123105	3.127148	15.102171
56873	8181	33107	238542	3.128159	15.198077
56874	8181	14707	108487	3.731.025	14.498118
56875	8181	9269	63318	3.733791	14.495344
56876	8181	47859	21622	18.954927	15.110433
56877	8181	90103	37707	18.955862	15.108433
56878	8181	177584	70996	18.955948	15.105613
56879	8181	169137	71311	18.95658	15.097218
56880	8181	150997	66736	19.424928	1.464038
56881	8181	155683	67455	19.432776	14.630041

Table 5.4: An example of TCP conversation just during the test

5.6 Analysis of traces

At this part some factors that might affect each specific browser output are investigated. For sure, it cannot be all the factors, and also the differences are presented based on our result and system and browser configuration. Moreover, the way that Ookla implement for each browser could be different. Plus, each operating system uses different type of TCP.

It should be mentioned that, we did not change the default configuration of the Firefox. Below you can see the Firefox network preferences.



Preference Name	Status	Type	Value
network.http.max-connections	default	integer	900
network.http.max-persistent-connections-per-proxy	default	integer	32
network.http.max-persistent-connections-per-server	default	integer	6
network.http.max-urgent-start-excessive-connections-per-host	default	integer	3
network.http.max_response_header_size	default	integer	393216

Figure 5.44: Firefox network configuration

5.6.1 Number of TCP connection

Unfortunately, there is no reference or standard about the number of TCP connections that a browser can open, even there are suggestions in HTTP standard. In short Safari and Chrome never open more than 6 connections for download or upload.

Edge in Windows for download in both Ookla platforms in our tests used 6 connections, also for upload in Public platform, but in Private platform it establishes variable amount of connections up to 32.

Firefox followed the default configuration, in the sense that, it did not open more than 32 connections for download and 6 for upload. Firefox in most of the cases, used 6 connections for upload and some times less than this number, but for download this number varied a lot. Statistics about number of connections are available at A.8.

This numbers that we talked about are persistent connections, of course there are some TCP connections even between desired client and servers, during the speed test, but they did not use for test, they used for encryption with SSL(TLS) protocol.

5.6.2 Connection management in HTTP/1.x

Some factors that clearly affect the the performance of web site and web applications are opening and preserving the connection. Some models are: short-lived connections, persistent connections, and HTTP pipelining.

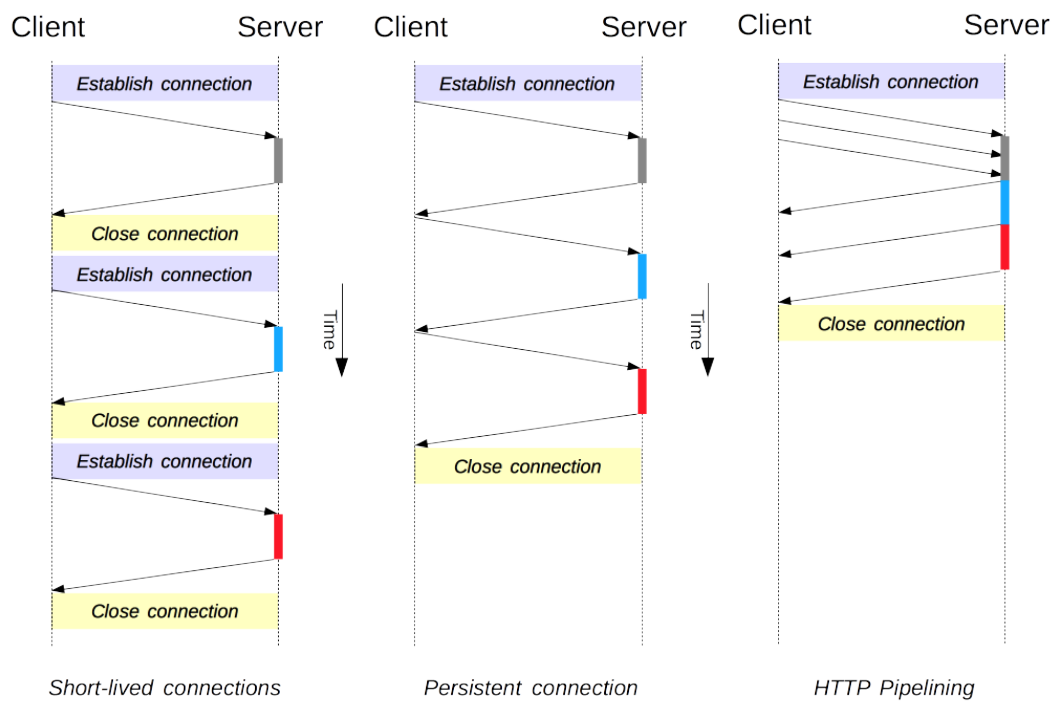


Figure 5.45: Three models of HTTP connection management

Short live connections deal with very old systems, those that not support the persistent connection. Also, modern browsers do not use the pipelining. So, we should focus on persistent connection. HTTP/1.1 defaults to the use of persistent connections, allowing multiple requests and responses to be carried over a single connection. The close connection option is used to signal that a connection will not persist after the current request/response. HTTP implementations should support persistent connections [22].

Persistent connection

This type of connection continue opening for a time interval, and it allows multiple requests and responses to be carried over a single connection, also it reduces the need for new TCP handshakes, and improves TCP's performance. It will be closed after some time. In order to remain persistent, all messages on a connection need to have a self-defined message length. [23, 22]
There are some advantages of using persistent connections, some of them mentioned bellow:

- Decreased delay in successive requests (no handshaking)
- Reduced CPU usage and round-trips because of less new connections and TLS handshakes
- Reduced network congestion (fewer TCP connections) [24]

Now we try to show the impact of these factors in different browsers. Based on the section 5.6.1 and the statistics on tables A.7 and A.8, Firefox gets better results for download when it opens at maximum 6 connections in Windows. The impact of number of connections can be see better when we compare the results of Public platform with Private in which it opens more connections for download. Also, Firefox opens more connections in download in MAC OS, which leads to low performance comparing to Safari and Chrome. This does not mean that if we reduce the number of connections, we should get better result. As it can be seen in the tables, when we have less than 6 connections, we get very low performance, except one sample for Safari.
But, why Edge in upload behaves better in windows, specially in Private platform, even it opens more connections? First, it should be said that, Edge always opened more than 6 connections, just in the Private platform. This is the only browser that passed the maximum number of 6 connections in both operating systems. Below the TCP traces of Edge just for upload connections, in Public and Private platform is presented.

Packets B → A	Rel Start	Duration
9210	21.337704	30.060393
10175	21.338004	30.073931
8725	21.801276	29.615399
8681	21.827214	29.580738

(a) Public platform

Packets B → A	Rel Start	Duration
5367	19.067237	15.195865
4679	19.522135	14.742005
4749	19.525022	14.740587
4074	19.528648	14.737213
4682	19.531082	14.734526
4348	19.534162	14.731697
4611	19.538219	14.727641
4334	19.541683	14.727881
4203	19.544839	14.718769
4468	19.548053	14.717556
4973	19.553692	14.715079
4785	19.557055	14.708806
4127	20.108986	14.156624
4048	20.112063	14.153797
4081	20.115281	14.150328
3831	20.122659	14.143202
4376	20.126634	14.138976
4118	20.129248	14.137781
4470	20.132735	14.135378
4168	20.137143	14.130248
4233	20.146211	14.120829
4555	20.149751	14.118412
4328	20.153803	14.113587
4294	20.157865	14.109164
4522	20.160987	14.107784
3726	20.164666	14.102361
3494	20.669135	13.596887
4353	20.671896	13.597668
4021	20.674568	13.591455
4140	20.677583	13.591981
3741	20.680332	13.589453
3755	20.682678	13.584351

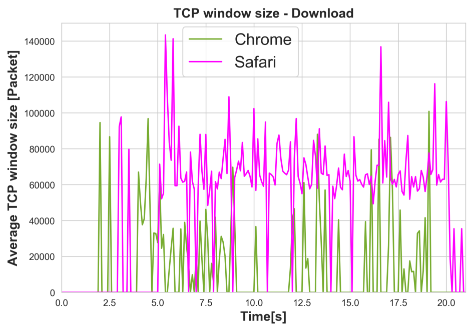
(b) Private platform

Table 5.5: Edge upload TCP streams

Edge transfers 36791 and 137654 in Public and Private platform respectively. So, it seems that, if a browser can send more packets even with more number of connections, for upload gives better result. So, the condition of sending more packets is so important and in the MSC OS we can see this impact better.

TCP window size

Without doubt, the TCP window size is one of the factor that affect the results. Here we show at the same operating system which browser performed better.



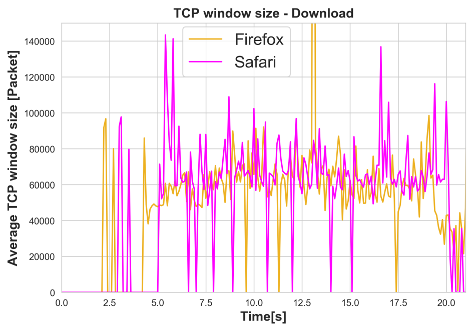


Figure 5.46: TCP window size

At the left side the impact of window size is clear. Safari has bigger window size during download and can receive more packets. Both browsers have 6 tcp connections.

On the other hand, there is not a big difference in the window size. The reason that Safari with respect to Firefox gives better performance is the number of connections. Firefox opened 13 connections for download.This would create more bursty traffic, congesting the buffers more quickly. Similarly, having N connections implies each connection gets 1/N of the available capacity. In such case, then each TCP connection works on lower rates, and could thus affect the congestion control algorithm too.

Packets B $\rightarrow$ A	Rel Start	Duration
96512	4.344419	15.229628
130996	4.362664	15.212181
29517	4.379444	15.253200
68867	4.400586	15.173459
11989	5.672146	14.674728
10399	6.238856	14.106792
7201	7.511156	13.373546
8034	8.929358	12.002968
6297	10.01870	10.084478
7141	11.29015	9.7195701
5661	12.90705	7.7909330
4463	14.13112	6.7362730
8	16.01667	1.9057600
3	19.01121	0.6216900
5	19.78192	0.2645649
7	20.18589	0.2846630
8	20.58140	0.3029770
10	20.90712	0.0893279

Table 5.6: Firefox Download TCP streams in MAC OS

Encryption

As mentioned at section 5.6.1, one of the factors that can affect the throughput is encryption.As in the table 5.7 the statistics of TCP streams of firefox for download is presented, there are more than 6 connections. But, there are 6 connections which are used for encryption. These connections are not used download speed measurements, but they use CPU. Moreover, this kind of encryption does not exists in our results for Windows. The situation is also worse when we look at the upload streams of Firefox. Not only there are these types of connections, but also they opened during the test not after that.

Packets B → A	Rel Start	Duration
12211	20.959748	15.202395
13213	20.966751	15.195386
12100	20.966894	15.195246
14927	20.967017	15.196248
10	20.993703	0.0384619
10	21.021604	0.0931570
9	21.112575	0.0839600
6	21.195499	0.2454979
10	21.241916	0.2945339
10	21.535741	0.4599068
8	21.858288	0.8735439
5941	22.289671	13.874892
5158	22.289853	13.874858
4	22.732179	0.7925381
4	23.779619	0.2887191
10	24.474475	1.2382121
2	25.915151	0.0036379
5	28.267235	0.5024319
7	29.528252	1.5490051

Table 5.7: Firefox Upload TCP streams in MAC OS

Clearly, in the above figure we can see that, 7 connections for encryption opened after 4 connections in the beginning, and other 6 after 2 last connections for upload. Also, our statistics at A.8 show that, Firefox when opens 6 connections in Windows performed better.

In short, admissible number of connections, which are persistent, and those that have bigger TCP window size get better result. At the end, we could suggest to use Safari for speed test with Ookla speed test system both for download and upload in MAC OS, and Firefox and Edge for download and upload respectively in Windows.

Chapter 6

Conclusion

Conclusion

This study as its subject tells, is more practical than theoretical. Although, internet bandwidth measurement is not a new topic, but the type of measurement and the application that used is a specific type of measurements that has got its own advantages and disadvantages. The Ookla speed test system is used and investigated, because a lot of users, not only Fastweb customers, but also other Internet users utilizing this platform to examine their speed. For people who are not expert, it could be the first choice to measure the quality of the service that they are using, they can start complaining based on these results about the level of their facilities. The purpose of this study was to find the factors that might affect the result that the user can experience.

Without doubt, ISPs are trying to improve their excellence of offering services to customers, and they benefit from new technologies, and in particular for our case, GPON. Obviously, the objective of the thesis is not to improve this architecture, but it seemed necessary to remark at least the features of this brand new technology at the beginning.

Similarly, the main platform which is investigated, needed to be explained more. Ookla speed test system measures three important factors of Internet speed, round trip time, download and upload. After talking about the network architecture, the procedure that this system uses for computing these elements has explained. Also, the requirements of each method of calculation, should be considered and prepared, so these demands mentioned too.

The result of each speed measurement, could have its own structure. At chapter four, this pattern is clarified. Plus, the tools and software that this study has used for analysing. Clearly, to get more reliable measurements, we need to repeat the task. These iteration could be frustrating if a correct way will not be considered. Moreover, an accurate way could be extended easier. With this background, Selenium used for doing repetitive tests. This tool helped us a lot, in the sense that, our measurements performed at the application layer.

The analyses has started withe hypothesis that, the CPEs are capable enough for quality assurance. Fortunately, this idea by the help of first data set proved. There were some assumptions about the distance and the influence of round trip time on the results. In an unusual way, it has been illustrated that, for our case, this important factor, did not affect our results as we predicted. Our analyses showed that we should focus more on operating system and the type of browsers. Two main operating systems and three major browsers at each operating systems chose. The tests this time performed with equal number of iteration, to prevent driving outcome based on sloppy samples. In short, it could be stated that, this does not seems correct to expect similar result with different browser at different operating system. Moreover, it is not very accurate to compare different browser at different operating system with each other. As it shown, a specific browser can be the best choice for a specific operating system, but at the same time the worst option for the other. Clearly, different operating systems, have their own configurations, and in particular for out case, the TCP behave differently at distinct platform. It is illustrated that, even we use the perfect browser at each operating system, the type of connection can affect the results enormously. So, even we were in the lab and almost in a ideal condition, but we could not reach to performance of wire condition. Based on this, we can state that, if we do compare the results of different browsers at a particular operating system, the type of connection can reduce the efficiency between 10 to 50 percents.

To conclude, it is suggested based on this study, if the Ookla speed test is the tools for examining the speed, the users should employ the correct browser at each operating system, and it is better to consider the downstream or upstream circumstances. As a result, it seems preferable to use Firefox for testing download both in wire and wireless connection in Windows, but for upload Edge and Chrome are finer choices in Windows for wire and wifi connection respectively. On the other hand, Safari in the MAC OS is the shining option in all the situation.

Appendix A

Appendix

A.1 Architecture of Test Plant-GPON

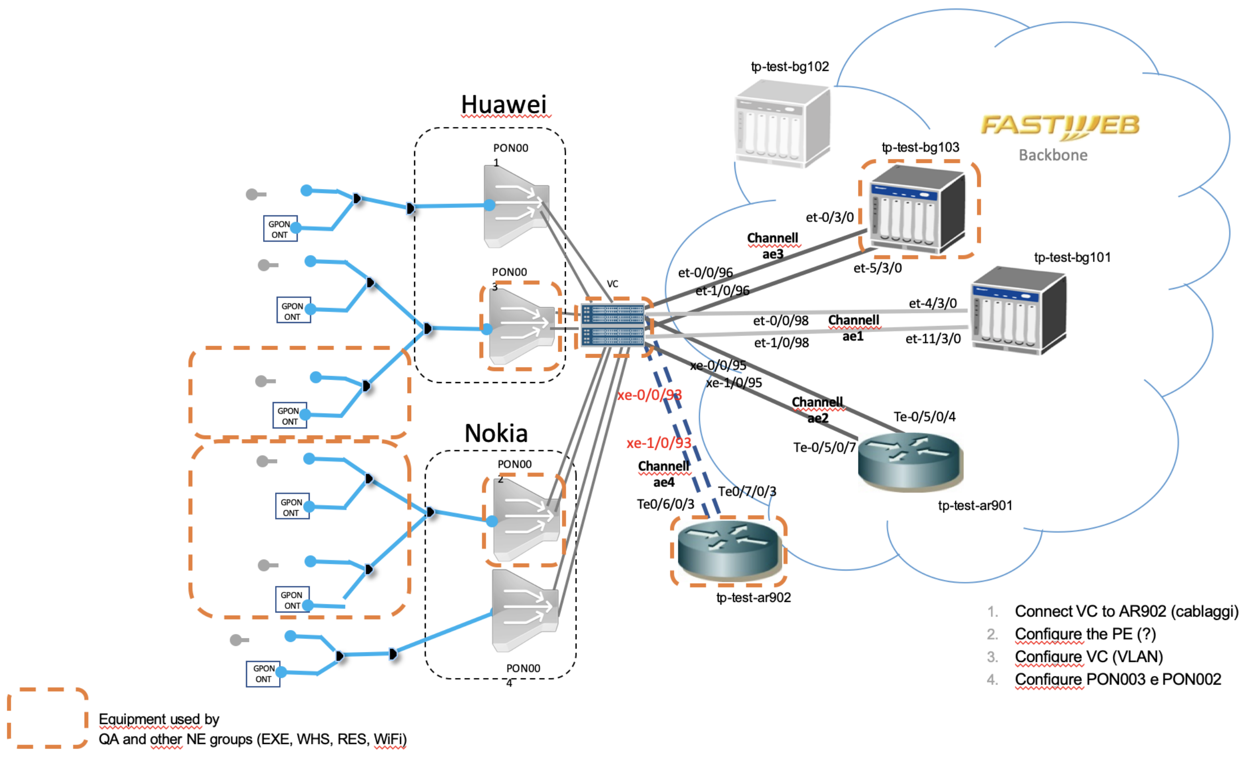
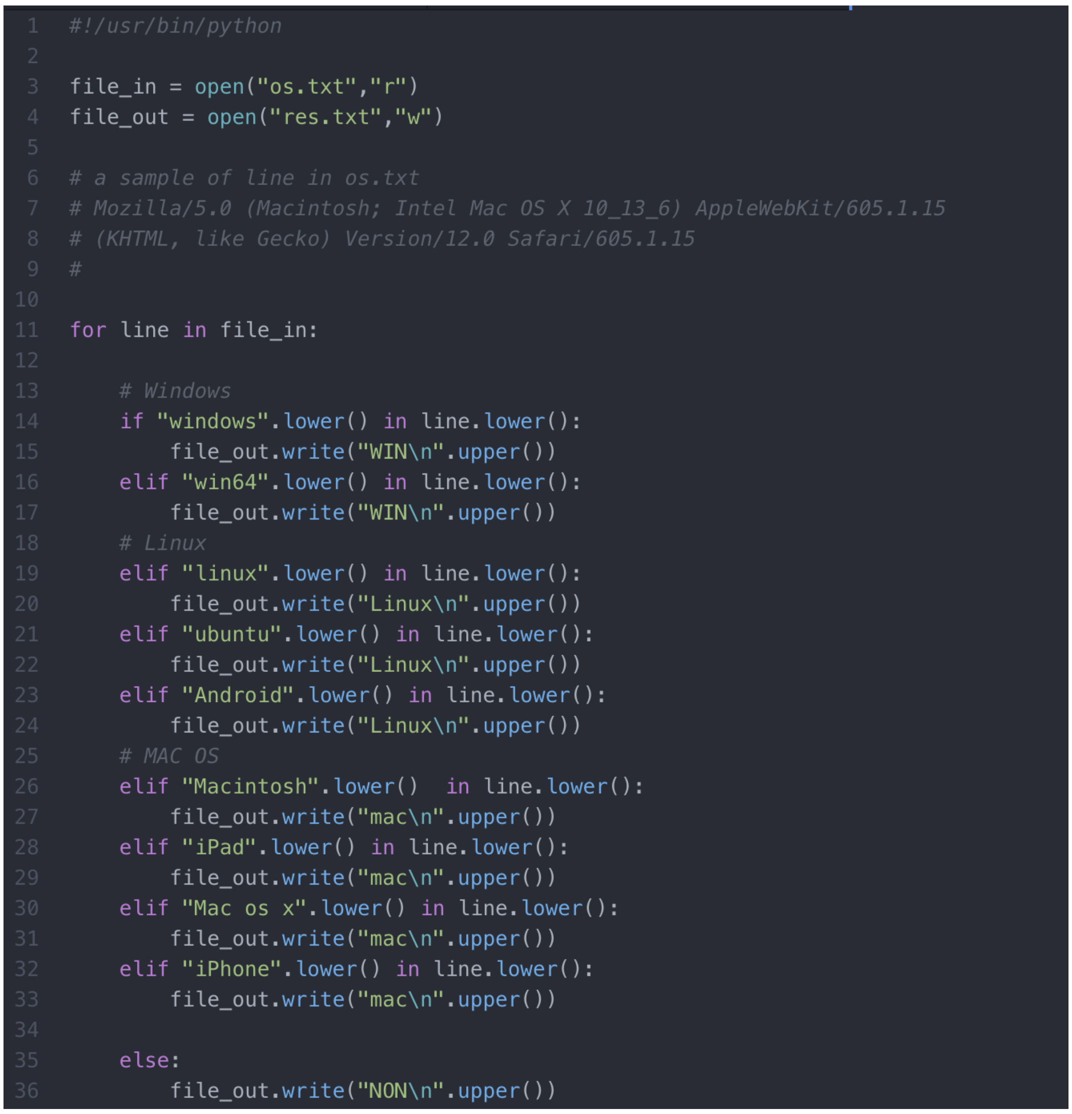


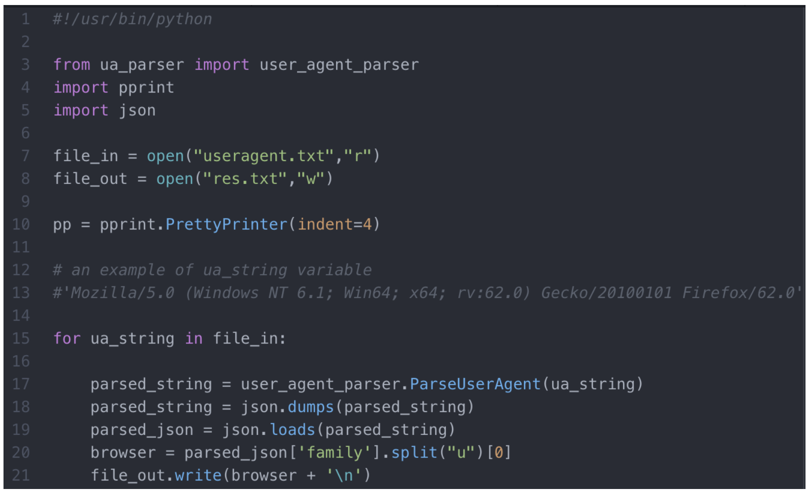
Figure A.1: Fastweb test plant architecture

A.2 Example of user agent parser



```python
#!/usr/bin/python

file_in = open("os.txt","r")
file_out = open("res.txt","w")

# a sample of line in os.txt
# Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/605.1.15
# (KHTML, like Gecko) Version/12.0 Safari/605.1.15
#

for line in file_in:

    # Windows
    if "windows".lower() in line.lower():
        file_out.write("WIN\n".upper())
    elif "win64".lower() in line.lower():
        file_out.write("WIN\n".upper())
    # Linux
    elif "linux".lower() in line.lower():
        file_out.write("Linux\n".upper())
    elif "ubuntu".lower() in line.lower():
        file_out.write("Linux\n".upper())
    elif "Android".lower() in line.lower():
        file_out.write("Linux\n".upper())
    # MAC OS
    elif "Macintosh".lower()  in line.lower():
        file_out.write("mac\n".upper())
    elif "iPad".lower() in line.lower():
        file_out.write("mac\n".upper())
    elif "Mac os x".lower() in line.lower():
        file_out.write("mac\n".upper())
    elif "iPhone".lower() in line.lower():
        file_out.write("mac\n".upper())

    else:
        file_out.write("NON\n".upper())
```

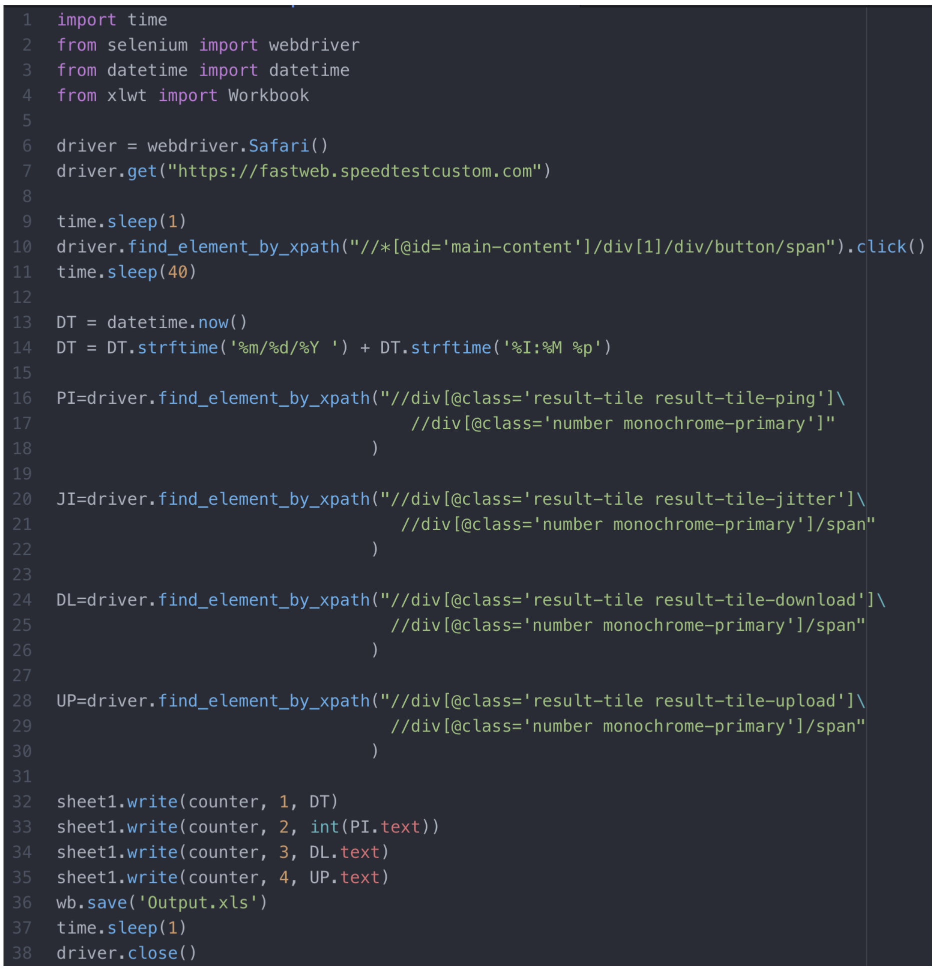
Figure A.2: Python script for extracting OS type



```python
#!/usr/bin/python

from ua_parser import user_agent_parser
import pprint
import json

file_in = open("useragent.txt","r")
file_out = open("res.txt","w")

pp = pprint.PrettyPrinter(indent=4)

# an example of ua_string variable
#'Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:62.0) Gecko/20100101 Firefox/62.0'

for ua_string in file_in:

    parsed_string = user_agent_parser.ParseUserAgent(ua_string)
    parsed_string = json.dumps(parsed_string)
    parsed_json = json.loads(parsed_string)
    browser = parsed_json['family'].split("u")[0]
    file_out.write(browser + '\n')
```

Figure A.3: Python script for extracting browser name

A.3 Example of test automation



```python
import time
from selenium import webdriver
from datetime import datetime
from xlwt import Workbook

driver = webdriver.Safari()
driver.get("https://fastweb.speedtestcustom.com")

time.sleep(1)
driver.find_element_by_xpath("//*[@id='main-content']/div[1]/div/button/span").click()
time.sleep(40)

DT = datetime.now()
DT = DT.strftime('%m/%d/%Y ') + DT.strftime('%I:%M %p')

PI=driver.find_element_by_xpath("//div[@class='result-tile result-tile-ping']\
                                    //div[@class='number monochrome-primary']"
                                )

JI=driver.find_element_by_xpath("//div[@class='result-tile result-tile-jitter']\
                                   //div[@class='number monochrome-primary']/span"
                                )

DL=driver.find_element_by_xpath("//div[@class='result-tile result-tile-download']\
                                  //div[@class='number monochrome-primary']/span"
                                )

UP=driver.find_element_by_xpath("//div[@class='result-tile result-tile-upload']\
                                  //div[@class='number monochrome-primary']/span"
                                )

sheet1.write(counter, 1, DT)
sheet1.write(counter, 2, int(PI.text))
sheet1.write(counter, 3, DL.text)
sheet1.write(counter, 4, UP.text)
wb.save('Output.xls')
time.sleep(1)
driver.close()
```

Figure A.4: Python script for running test on Safari without iteration

A.4 Tables of the existing samples

Core Network	Number of Samples
Milan	128165

(a) Milan wifi connection total samples

Operating System	Number of Samples
Windows	101451
MAC OS	14326
Linux	12369

(b) Milan wifi connection OS samples

	Edge	Safari	Firefox	Chrome	IE	Opera	Total
Windows	32028	0	15854	47923	4661	919	101385
MAC OS	0	5228	1202	2634	0	65	9129
Linux	0	27	5128	1005	0	5	6165
Total	32028	5255	22184	51562	4661	989	116679

Table A.2: Milan wifi connection browser samples

Core Network	Number of Samples
Rome	40242

(a) Rome wire connection total samples

Operating System	Number of Samples
Windows	34214
MAC OS	2915
Linux	3087

(b) Rome wire connection OS samples

	Edge	Safari	Firefox	Chrome	IE	Opera	Total
Windows	5491	0	11996	15190	1194	312	34183
MAC OS	0	1332	667	899	0	1	2899
Linux	0	59	2883	74	0	23	3039
Total	5491	1391	15546	16163	1194	336	40121

Table A.4: Rome wire connection browser samples

Core Network	Number of Samples
Rome	37495

(a) Rome wifi connection total samples

Operating System	Number of Samples
Windows	27087
MAC OS	5068
Linux	5339

(b) Rome wifi connection OS samples

	Edge	Safari	Firefox	Chrome	IE	Opera	Total
Windows	5250	0	10221	10545	898	146	27060
MAC OS	0	1800	572	1010	0	0	3382
Linux	0	47	2052	542	0	6	2647
Total	5250	1847	12845	12097	898	152	33089

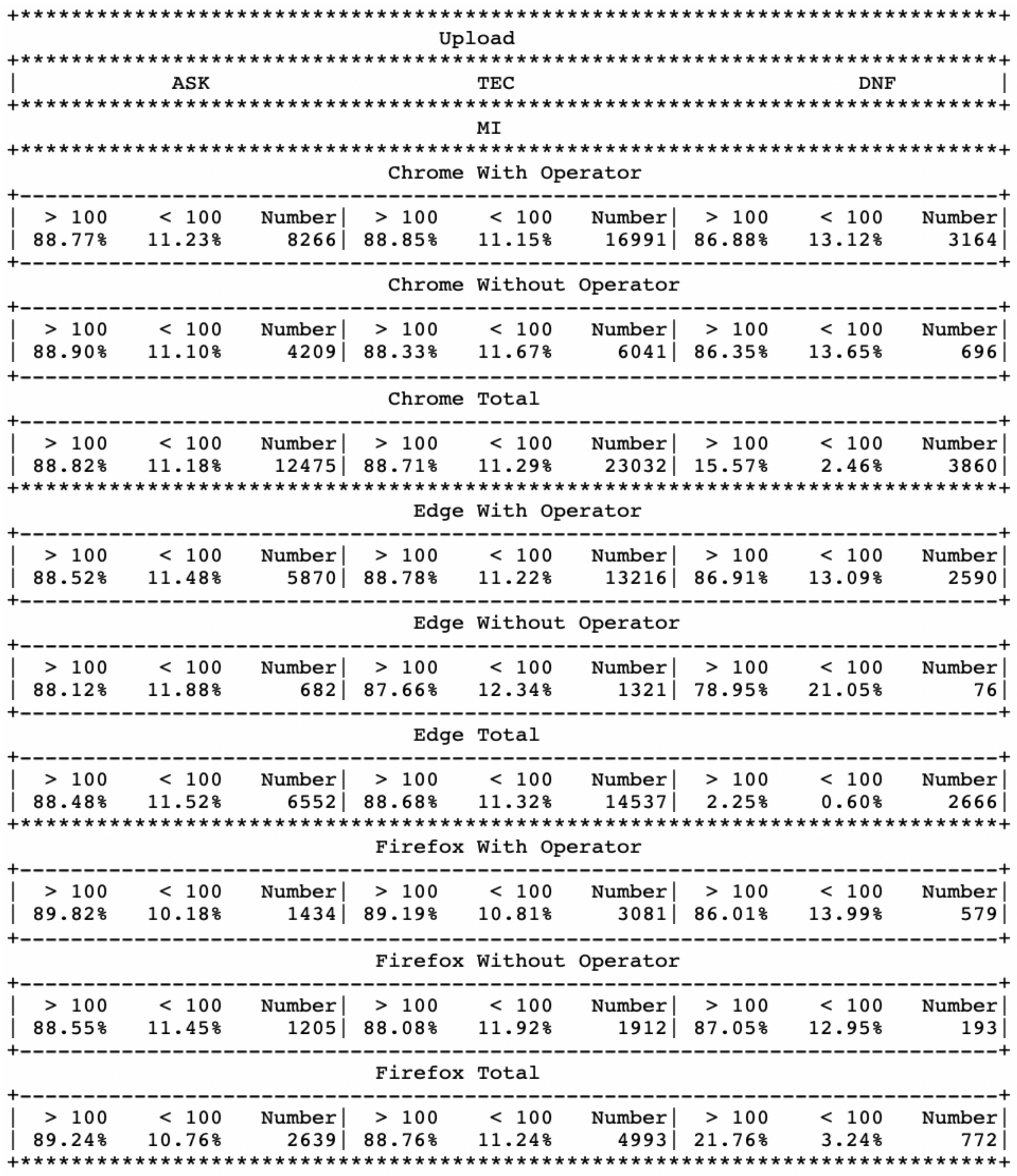
Table A.6: Rome wifi connection browser samples

A.5 Statistics related to CPE, Rtt and distance

This type of results are available in general for 5 browsers, and all the province in the data set. Name of CPE is written in short as ASK, TEC and DNF, Also, MI and TO indicate Milan and Turin respectively. Number indicates the available samples for each case. Company considers that, if the users measure a value greater than or equal 500 Mbps (Download), 100 Mbps (Upload) and less than or equal 5 ms (Rtt), they are at good condition.

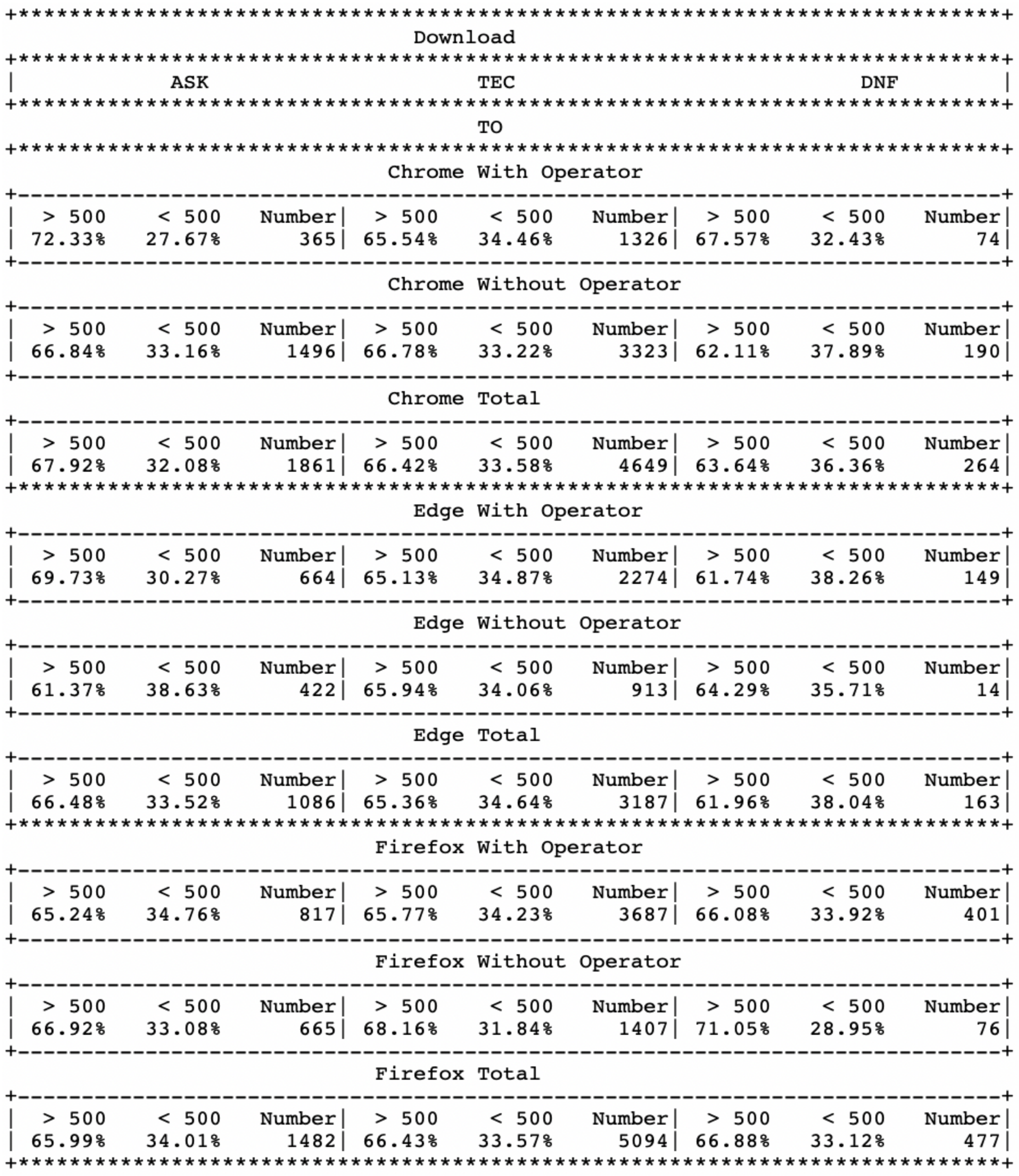
```
+**************************************************************************+
                                 Download
+**************************************************************************+
|           ASK                     TEC                          DNF       |
+**************************************************************************+
                                     MI
+**************************************************************************+
                           Chrome With Operator
+--------------------------------------------------------------------------+
| > 500    < 500   Number| > 500    < 500   Number| > 500    < 500   Number|
| 68.29%   31.71%    8266| 68.62%   31.38%   16991| 66.02%   33.98%    3164|
+--------------------------------------------------------------------------+
                          Chrome Without Operator
+--------------------------------------------------------------------------+
| > 500    < 500   Number| > 500    < 500   Number| > 500    < 500   Number|
| 66.83%   33.17%    4209| 68.15%   31.85%    6041| 64.66%   35.34%     696|
+--------------------------------------------------------------------------+
                               Chrome Total
+--------------------------------------------------------------------------+
| > 500    < 500   Number| > 500    < 500   Number| > 500    < 500   Number|
| 67.80%   32.20%   12475| 68.50%   31.50%   23032| 65.78%   34.22%    3860|
+**************************************************************************+
                            Edge With Operator
+--------------------------------------------------------------------------+
| > 500    < 500   Number| > 500    < 500   Number| > 500    < 500   Number|
| 67.04%   32.96%    5870| 67.76%   32.24%   13216| 65.10%   34.90%    2590|
+--------------------------------------------------------------------------+
                           Edge Without Operator
+--------------------------------------------------------------------------+
| > 500    < 500   Number| > 500    < 500   Number| > 500    < 500   Number|
| 67.89%   32.11%     682| 66.77%   33.23%    1321| 60.53%   39.47%      76|
+--------------------------------------------------------------------------+
                                Edge Total
+--------------------------------------------------------------------------+
| > 500    < 500   Number| > 500    < 500   Number| > 500    < 500   Number|
| 67.12%   32.88%    6552| 67.67%   32.33%   14537| 64.97%   35.03%    2666|
+**************************************************************************+
                          Firefox With Operator
+--------------------------------------------------------------------------+
| > 500    < 500   Number| > 500    < 500   Number| > 500    < 500   Number|
| 68.48%   31.52%    1434| 67.84%   32.16%    3081| 65.63%   34.37%     579|
+--------------------------------------------------------------------------+
                          Firefox Without Operator
+--------------------------------------------------------------------------+
| > 500    < 500   Number| > 500    < 500   Number| > 500    < 500   Number|
| 69.96%   30.04%    1205| 66.42%   33.58%    1912| 65.80%   34.20%     193|
+--------------------------------------------------------------------------+
                              Firefox Total
+--------------------------------------------------------------------------+
| > 500    < 500   Number| > 500    < 500   Number| > 500    < 500   Number|
| 69.15%   30.85%    2639| 67.29%   32.71%    4993| 65.67%   34.33%     772|
+**************************************************************************+
```

Figure A.5: Milan - Download



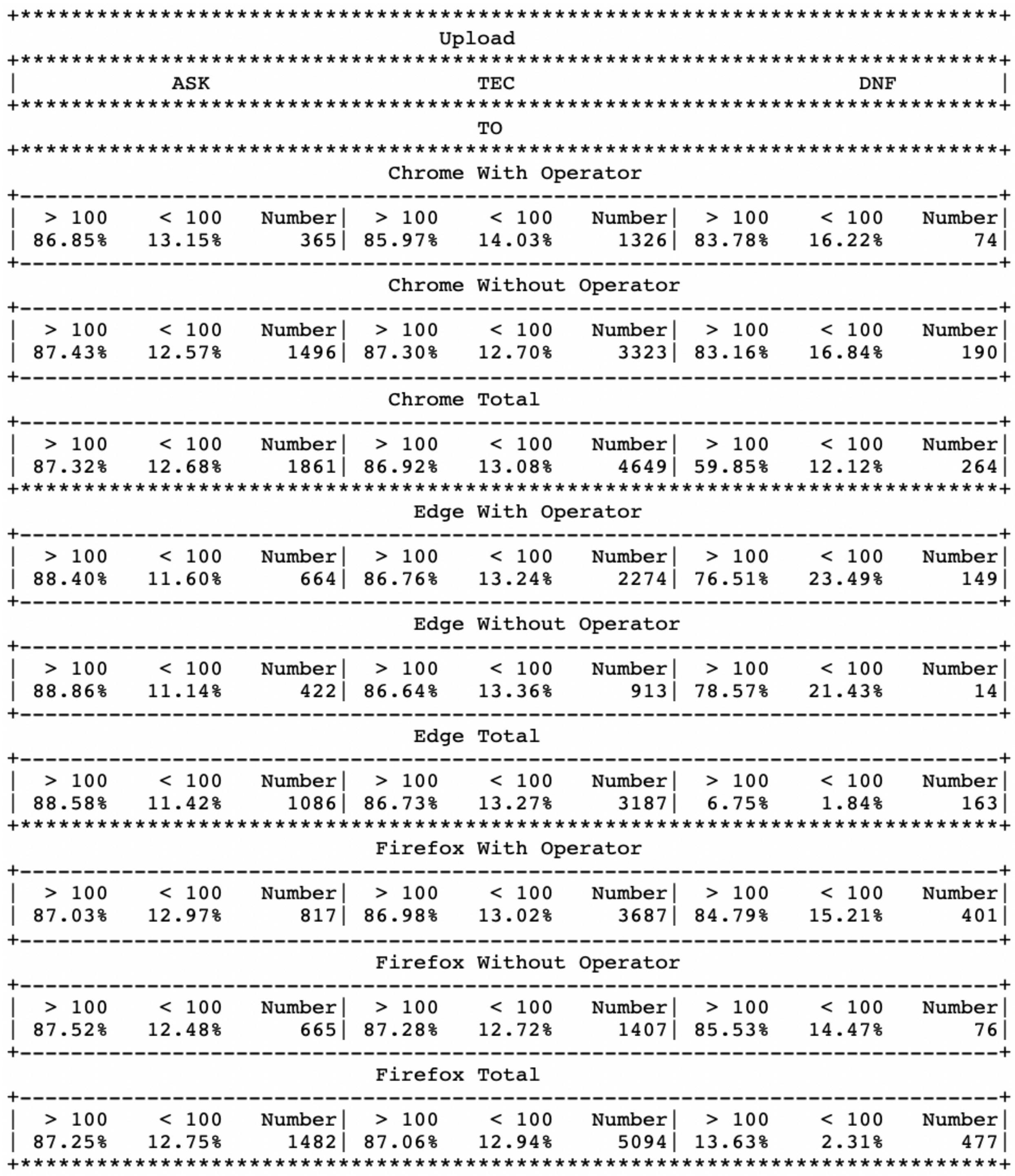
```
+**************************************************************************+
                                   Upload
+**************************************************************************+
|            ASK                     TEC                         DNF        |
+**************************************************************************+
                                     MI
+**************************************************************************+
                           Chrome With Operator
+--------------------------------------------------------------------------+
|  > 100    < 100    Number|  > 100    < 100    Number|  > 100    < 100    Number|
| 88.77%   11.23%      8266| 88.85%   11.15%     16991| 86.88%   13.12%      3164|
+--------------------------------------------------------------------------+
                           Chrome Without Operator
+--------------------------------------------------------------------------+
|  > 100    < 100    Number|  > 100    < 100    Number|  > 100    < 100    Number|
| 88.90%   11.10%      4209| 88.33%   11.67%      6041| 86.35%   13.65%       696|
+--------------------------------------------------------------------------+
                           Chrome Total
+--------------------------------------------------------------------------+
|  > 100    < 100    Number|  > 100    < 100    Number|  > 100    < 100    Number|
| 88.82%   11.18%     12475| 88.71%   11.29%     23032| 15.57%    2.46%      3860|
+**************************************************************************+
                             Edge With Operator
+--------------------------------------------------------------------------+
|  > 100    < 100    Number|  > 100    < 100    Number|  > 100    < 100    Number|
| 88.52%   11.48%      5870| 88.78%   11.22%     13216| 86.91%   13.09%      2590|
+--------------------------------------------------------------------------+
                             Edge Without Operator
+--------------------------------------------------------------------------+
|  > 100    < 100    Number|  > 100    < 100    Number|  > 100    < 100    Number|
| 88.12%   11.88%       682| 87.66%   12.34%      1321| 78.95%   21.05%        76|
+--------------------------------------------------------------------------+
                             Edge Total
+--------------------------------------------------------------------------+
|  > 100    < 100    Number|  > 100    < 100    Number|  > 100    < 100    Number|
| 88.48%   11.52%      6552| 88.68%   11.32%     14537|  2.25%    0.60%      2666|
+**************************************************************************+
                          Firefox With Operator
+--------------------------------------------------------------------------+
|  > 100    < 100    Number|  > 100    < 100    Number|  > 100    < 100    Number|
| 89.82%   10.18%      1434| 89.19%   10.81%      3081| 86.01%   13.99%       579|
+--------------------------------------------------------------------------+
                          Firefox Without Operator
+--------------------------------------------------------------------------+
|  > 100    < 100    Number|  > 100    < 100    Number|  > 100    < 100    Number|
| 88.55%   11.45%      1205| 88.08%   11.92%      1912| 87.05%   12.95%       193|
+--------------------------------------------------------------------------+
                          Firefox Total
+--------------------------------------------------------------------------+
|  > 100    < 100    Number|  > 100    < 100    Number|  > 100    < 100    Number|
| 89.24%   10.76%      2639| 88.76%   11.24%      4993| 21.76%    3.24%       772|
+**************************************************************************+
```

Figure A.6: Milan - Upload



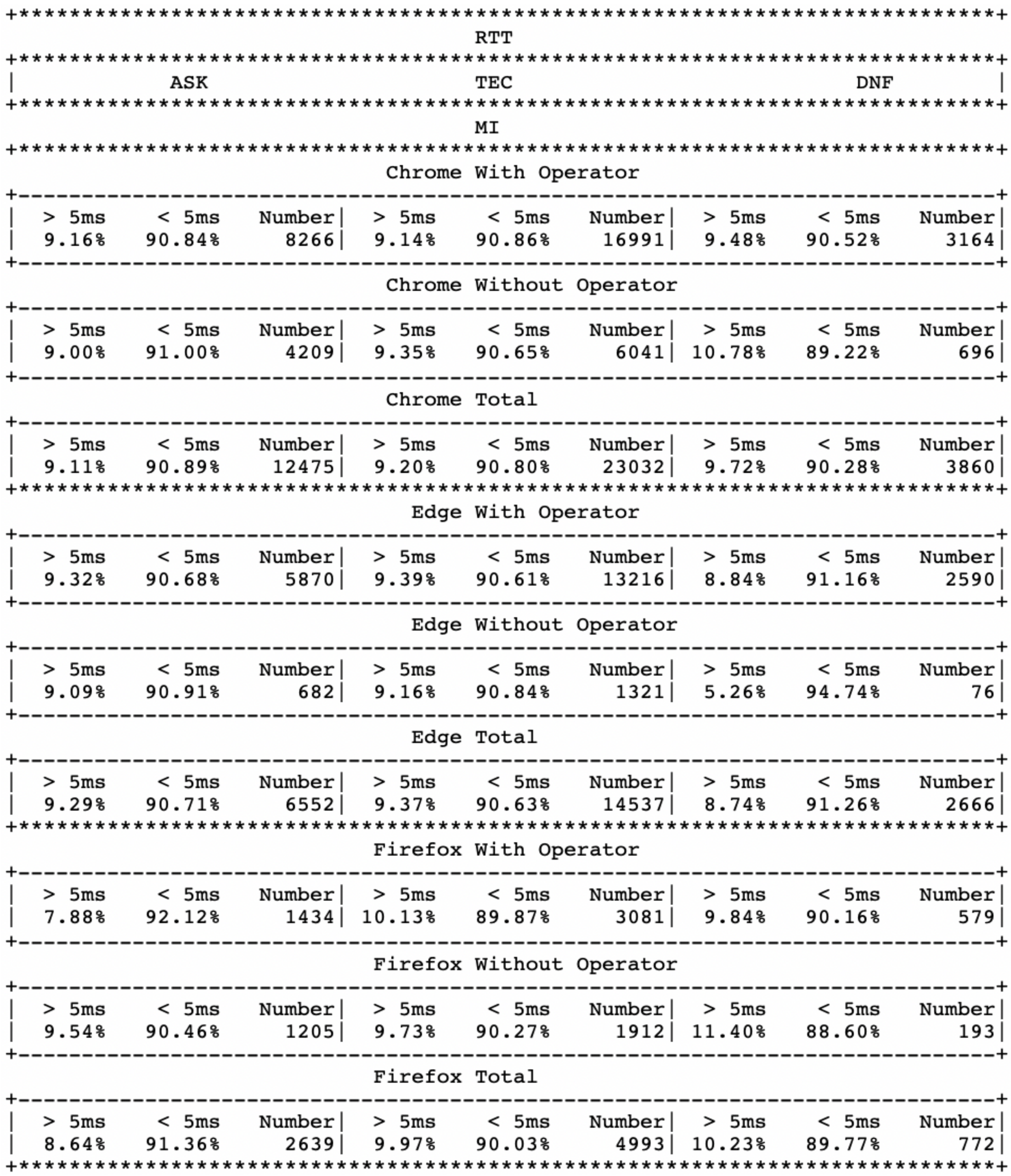
```
+****************************************************************************+
                                 Download
+****************************************************************************+
|           ASK                     TEC                          DNF          |
+****************************************************************************+
                                     TO
+****************************************************************************+
                            Chrome With Operator
+----------------------------------------------------------------------------+
|  > 500    < 500   Number|  > 500    < 500   Number|  > 500    < 500   Number|
| 72.33%   27.67%      365| 65.54%   34.46%     1326| 67.57%   32.43%       74|
+----------------------------------------------------------------------------+
                            Chrome Without Operator
+----------------------------------------------------------------------------+
|  > 500    < 500   Number|  > 500    < 500   Number|  > 500    < 500   Number|
| 66.84%   33.16%     1496| 66.78%   33.22%     3323| 62.11%   37.89%      190|
+----------------------------------------------------------------------------+
                            Chrome Total
+----------------------------------------------------------------------------+
|  > 500    < 500   Number|  > 500    < 500   Number|  > 500    < 500   Number|
| 67.92%   32.08%     1861| 66.42%   33.58%     4649| 63.64%   36.36%      264|
+****************************************************************************+
                              Edge With Operator
+----------------------------------------------------------------------------+
|  > 500    < 500   Number|  > 500    < 500   Number|  > 500    < 500   Number|
| 69.73%   30.27%      664| 65.13%   34.87%     2274| 61.74%   38.26%      149|
+----------------------------------------------------------------------------+
                              Edge Without Operator
+----------------------------------------------------------------------------+
|  > 500    < 500   Number|  > 500    < 500   Number|  > 500    < 500   Number|
| 61.37%   38.63%      422| 65.94%   34.06%      913| 64.29%   35.71%       14|
+----------------------------------------------------------------------------+
                              Edge Total
+----------------------------------------------------------------------------+
|  > 500    < 500   Number|  > 500    < 500   Number|  > 500    < 500   Number|
| 66.48%   33.52%     1086| 65.36%   34.64%     3187| 61.96%   38.04%      163|
+****************************************************************************+
                           Firefox With Operator
+----------------------------------------------------------------------------+
|  > 500    < 500   Number|  > 500    < 500   Number|  > 500    < 500   Number|
| 65.24%   34.76%      817| 65.77%   34.23%     3687| 66.08%   33.92%      401|
+----------------------------------------------------------------------------+
                           Firefox Without Operator
+----------------------------------------------------------------------------+
|  > 500    < 500   Number|  > 500    < 500   Number|  > 500    < 500   Number|
| 66.92%   33.08%      665| 68.16%   31.84%     1407| 71.05%   28.95%       76|
+----------------------------------------------------------------------------+
                           Firefox Total
+----------------------------------------------------------------------------+
|  > 500    < 500   Number|  > 500    < 500   Number|  > 500    < 500   Number|
| 65.99%   34.01%     1482| 66.43%   33.57%     5094| 66.88%   33.12%      477|
+****************************************************************************+
```

Figure A.7: Turin - Download



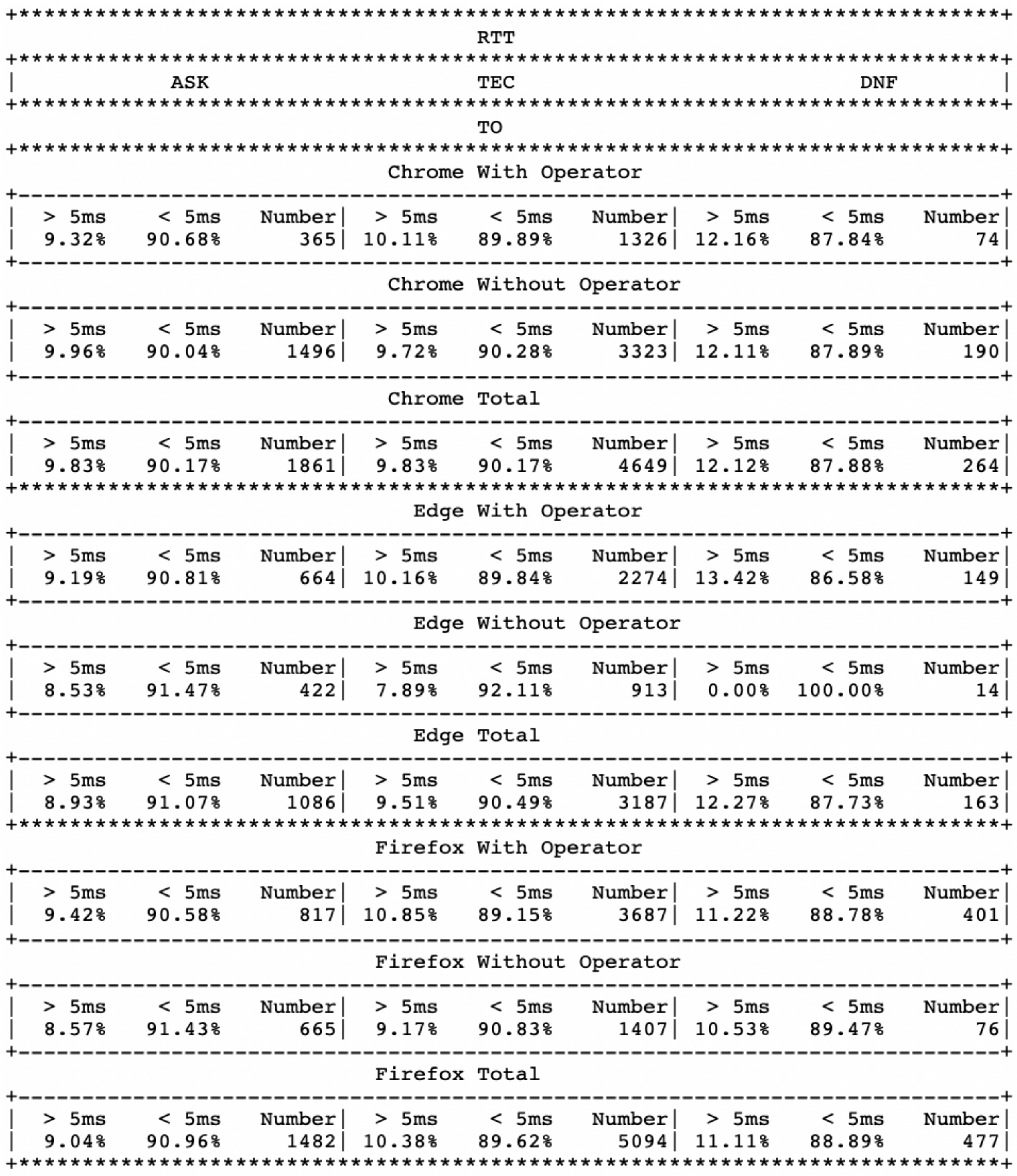
```
+***************************************************************************+
                                  Upload
+***************************************************************************+
|            ASK                      TEC                         DNF       |
+***************************************************************************+
                                    TO
+***************************************************************************+
                            Chrome With Operator
+---------------------------------------------------------------------------+
|  > 100    < 100    Number|  > 100    < 100    Number|  > 100    < 100    Number|
| 86.85%   13.15%       365| 85.97%   14.03%      1326| 83.78%   16.22%        74|
+---------------------------------------------------------------------------+
                          Chrome Without Operator
+---------------------------------------------------------------------------+
|  > 100    < 100    Number|  > 100    < 100    Number|  > 100    < 100    Number|
| 87.43%   12.57%      1496| 87.30%   12.70%      3323| 83.16%   16.84%       190|
+---------------------------------------------------------------------------+
                               Chrome Total
+---------------------------------------------------------------------------+
|  > 100    < 100    Number|  > 100    < 100    Number|  > 100    < 100    Number|
| 87.32%   12.68%      1861| 86.92%   13.08%      4649| 59.85%   12.12%       264|
+***************************************************************************+
                             Edge With Operator
+---------------------------------------------------------------------------+
|  > 100    < 100    Number|  > 100    < 100    Number|  > 100    < 100    Number|
| 88.40%   11.60%       664| 86.76%   13.24%      2274| 76.51%   23.49%       149|
+---------------------------------------------------------------------------+
                           Edge Without Operator
+---------------------------------------------------------------------------+
|  > 100    < 100    Number|  > 100    < 100    Number|  > 100    < 100    Number|
| 88.86%   11.14%       422| 86.64%   13.36%       913| 78.57%   21.43%        14|
+---------------------------------------------------------------------------+
                                Edge Total
+---------------------------------------------------------------------------+
|  > 100    < 100    Number|  > 100    < 100    Number|  > 100    < 100    Number|
| 88.58%   11.42%      1086| 86.73%   13.27%      3187|  6.75%    1.84%       163|
+***************************************************************************+
                           Firefox With Operator
+---------------------------------------------------------------------------+
|  > 100    < 100    Number|  > 100    < 100    Number|  > 100    < 100    Number|
| 87.03%   12.97%       817| 86.98%   13.02%      3687| 84.79%   15.21%       401|
+---------------------------------------------------------------------------+
                          Firefox Without Operator
+---------------------------------------------------------------------------+
|  > 100    < 100    Number|  > 100    < 100    Number|  > 100    < 100    Number|
| 87.52%   12.48%       665| 87.28%   12.72%      1407| 85.53%   14.47%        76|
+---------------------------------------------------------------------------+
                               Firefox Total
+---------------------------------------------------------------------------+
|  > 100    < 100    Number|  > 100    < 100    Number|  > 100    < 100    Number|
| 87.25%   12.75%      1482| 87.06%   12.94%      5094| 13.63%    2.31%       477|
+***************************************************************************+
```

Figure A.8: Turin - Upload



```
+*****************************************************************************+
                                     RTT
+*****************************************************************************+
|           ASK                        TEC                         DNF         |
+*****************************************************************************+
                                     MI
+*****************************************************************************+
                              Chrome With Operator
+-----------------------------------------------------------------------------+
|  > 5ms    < 5ms   Number|  > 5ms    < 5ms   Number|  > 5ms    < 5ms   Number|
|  9.16%   90.84%     8266|  9.14%   90.86%    16991|  9.48%   90.52%     3164|
+-----------------------------------------------------------------------------+
                              Chrome Without Operator
+-----------------------------------------------------------------------------+
|  > 5ms    < 5ms   Number|  > 5ms    < 5ms   Number|  > 5ms    < 5ms   Number|
|  9.00%   91.00%     4209|  9.35%   90.65%     6041| 10.78%   89.22%      696|
+-----------------------------------------------------------------------------+
                              Chrome Total
+-----------------------------------------------------------------------------+
|  > 5ms    < 5ms   Number|  > 5ms    < 5ms   Number|  > 5ms    < 5ms   Number|
|  9.11%   90.89%    12475|  9.20%   90.80%    23032|  9.72%   90.28%     3860|
+*****************************************************************************+
                                Edge With Operator
+-----------------------------------------------------------------------------+
|  > 5ms    < 5ms   Number|  > 5ms    < 5ms   Number|  > 5ms    < 5ms   Number|
|  9.32%   90.68%     5870|  9.39%   90.61%    13216|  8.84%   91.16%     2590|
+-----------------------------------------------------------------------------+
                                Edge Without Operator
+-----------------------------------------------------------------------------+
|  > 5ms    < 5ms   Number|  > 5ms    < 5ms   Number|  > 5ms    < 5ms   Number|
|  9.09%   90.91%      682|  9.16%   90.84%     1321|  5.26%   94.74%       76|
+-----------------------------------------------------------------------------+
                                Edge Total
+-----------------------------------------------------------------------------+
|  > 5ms    < 5ms   Number|  > 5ms    < 5ms   Number|  > 5ms    < 5ms   Number|
|  9.29%   90.71%     6552|  9.37%   90.63%    14537|  8.74%   91.26%     2666|
+*****************************************************************************+
                             Firefox With Operator
+-----------------------------------------------------------------------------+
|  > 5ms    < 5ms   Number|  > 5ms    < 5ms   Number|  > 5ms    < 5ms   Number|
|  7.88%   92.12%     1434| 10.13%   89.87%     3081|  9.84%   90.16%      579|
+-----------------------------------------------------------------------------+
                             Firefox Without Operator
+-----------------------------------------------------------------------------+
|  > 5ms    < 5ms   Number|  > 5ms    < 5ms   Number|  > 5ms    < 5ms   Number|
|  9.54%   90.46%     1205|  9.73%   90.27%     1912| 11.40%   88.60%      193|
+-----------------------------------------------------------------------------+
                             Firefox Total
+-----------------------------------------------------------------------------+
|  > 5ms    < 5ms   Number|  > 5ms    < 5ms   Number|  > 5ms    < 5ms   Number|
|  8.64%   91.36%     2639|  9.97%   90.03%     4993| 10.23%   89.77%      772|
+*****************************************************************************+
```

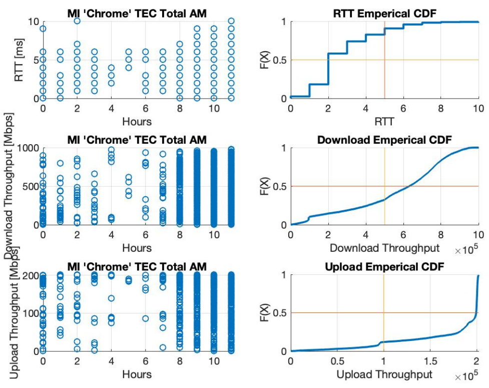
Figure A.9: Milan - Rtt



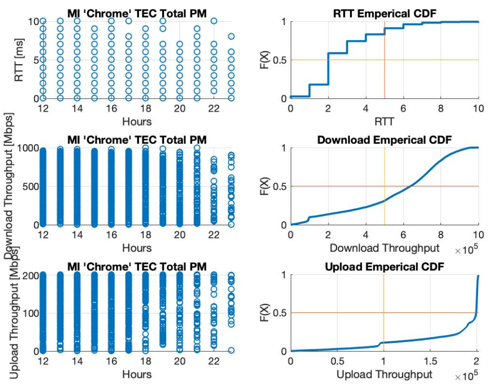
```
+*******************************************************************************+
                                      RTT
+*******************************************************************************+
|            ASK                        TEC                          DNF         |
+*******************************************************************************+
                                      TO
+*******************************************************************************+
                              Chrome With Operator
+-------------------------------------------------------------------------------+
|  > 5ms    < 5ms    Number|  > 5ms    < 5ms    Number|  > 5ms    < 5ms    Number|
|  9.32%   90.68%       365| 10.11%   89.89%      1326| 12.16%   87.84%        74|
+-------------------------------------------------------------------------------+
                              Chrome Without Operator
+-------------------------------------------------------------------------------+
|  > 5ms    < 5ms    Number|  > 5ms    < 5ms    Number|  > 5ms    < 5ms    Number|
|  9.96%   90.04%      1496|  9.72%   90.28%      3323| 12.11%   87.89%       190|
+-------------------------------------------------------------------------------+
                              Chrome Total
+-------------------------------------------------------------------------------+
|  > 5ms    < 5ms    Number|  > 5ms    < 5ms    Number|  > 5ms    < 5ms    Number|
|  9.83%   90.17%      1861|  9.83%   90.17%      4649| 12.12%   87.88%       264|
+*******************************************************************************+
                                Edge With Operator
+-------------------------------------------------------------------------------+
|  > 5ms    < 5ms    Number|  > 5ms    < 5ms    Number|  > 5ms    < 5ms    Number|
|  9.19%   90.81%       664| 10.16%   89.84%      2274| 13.42%   86.58%       149|
+-------------------------------------------------------------------------------+
                                Edge Without Operator
+-------------------------------------------------------------------------------+
|  > 5ms    < 5ms    Number|  > 5ms    < 5ms    Number|  > 5ms    < 5ms    Number|
|  8.53%   91.47%       422|  7.89%   92.11%       913|  0.00%  100.00%        14|
+-------------------------------------------------------------------------------+
                                Edge Total
+-------------------------------------------------------------------------------+
|  > 5ms    < 5ms    Number|  > 5ms    < 5ms    Number|  > 5ms    < 5ms    Number|
|  8.93%   91.07%      1086|  9.51%   90.49%      3187| 12.27%   87.73%       163|
+*******************************************************************************+
                               Firefox With Operator
+-------------------------------------------------------------------------------+
|  > 5ms    < 5ms    Number|  > 5ms    < 5ms    Number|  > 5ms    < 5ms    Number|
|  9.42%   90.58%       817| 10.85%   89.15%      3687| 11.22%   88.78%       401|
+-------------------------------------------------------------------------------+
                               Firefox Without Operator
+-------------------------------------------------------------------------------+
|  > 5ms    < 5ms    Number|  > 5ms    < 5ms    Number|  > 5ms    < 5ms    Number|
|  8.57%   91.43%       665|  9.17%   90.83%      1407| 10.53%   89.47%        76|
+-------------------------------------------------------------------------------+
                               Firefox Total
+-------------------------------------------------------------------------------+
|  > 5ms    < 5ms    Number|  > 5ms    < 5ms    Number|  > 5ms    < 5ms    Number|
|  9.04%   90.96%      1482| 10.38%   89.62%      5094| 11.11%   88.89%       477|
+*******************************************************************************+
```

Figure A.10: Turin - Rtt

A.6 Technicolor and DNF figures

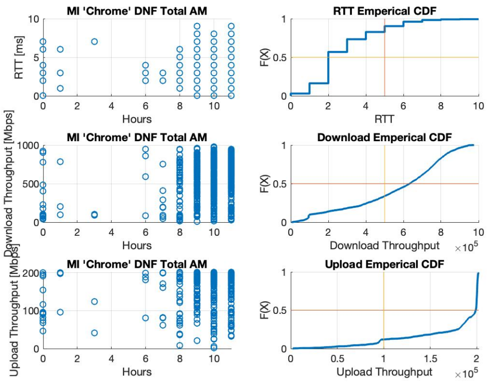


(a) A.M.

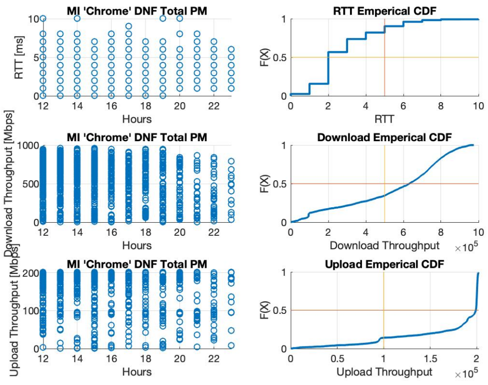


(b) P.M.

Figure A.11: Technicolor download/upload comparison based on Rtt and time



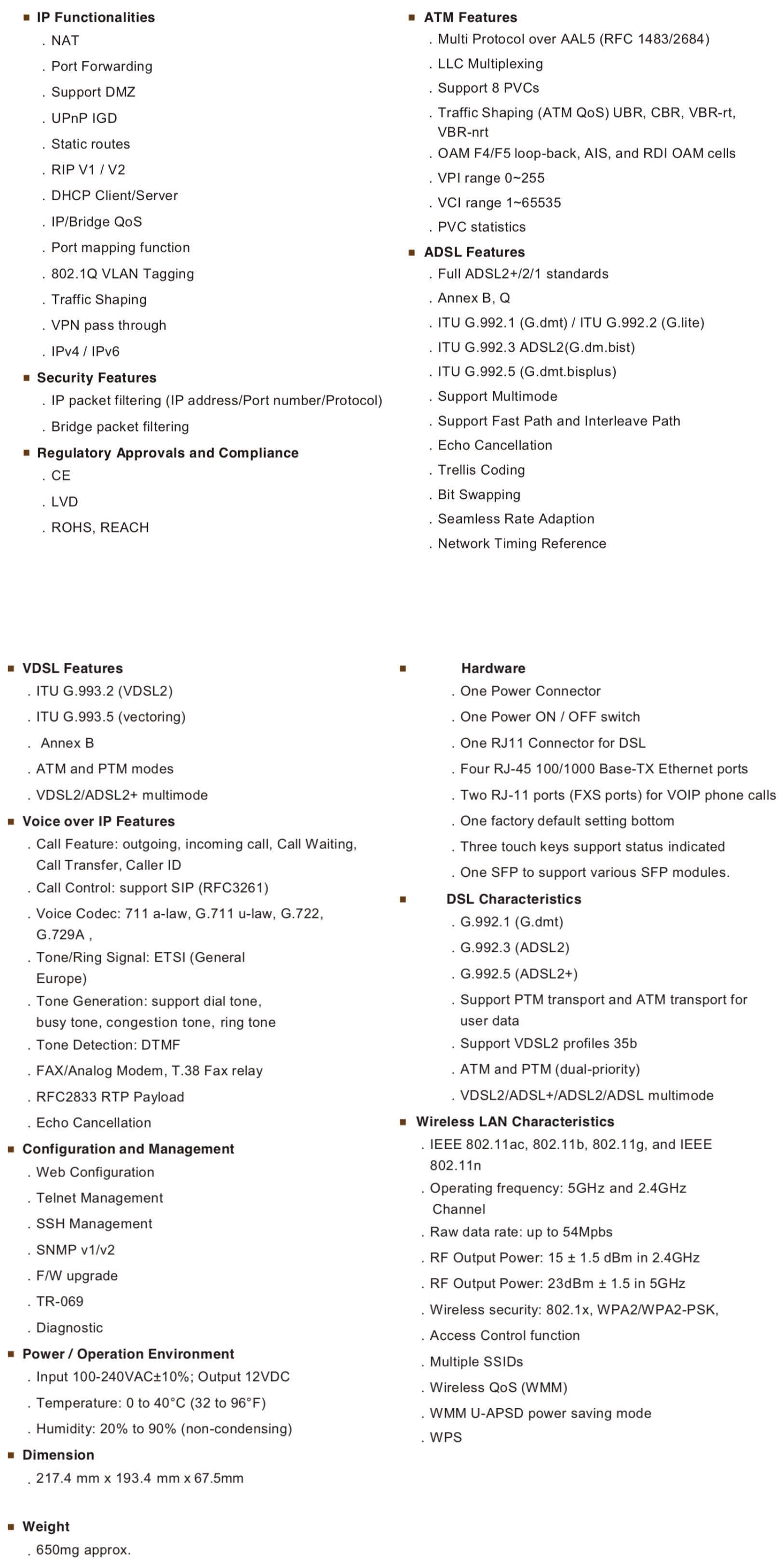
(a) A.M.



(b) P.M.

Figure A.12: DNF download/upload comparison based on Rtt and time

A.7 Askey residential gateway datasheet



- **IP Functionalities**
  - . NAT
  - . Port Forwarding
  - . Support DMZ
  - . UPnP IGD
  - . Static routes
  - . RIP V1 / V2
  - . DHCP Client/Server
  - . IP/Bridge QoS
  - . Port mapping function
  - . 802.1Q VLAN Tagging
  - . Traffic Shaping
  - . VPN pass through
  - . IPv4 / IPv6
- **Security Features**
  - . IP packet filtering (IP address/Port number/Protocol)
  - . Bridge packet filtering
- **Regulatory Approvals and Compliance**
  - . CE
  - . LVD
  - . ROHS, REACH
- **ATM Features**
  - . Multi Protocol over AAL5 (RFC 1483/2684)
  - . LLC Multiplexing
  - . Support 8 PVCs
  - . Traffic Shaping (ATM QoS) UBR, CBR, VBR-rt, VBR-nrt
  - . OAM F4/F5 loop-back, AIS, and RDI OAM cells
  - . VPI range 0~255
  - . VCI range 1~65535
  - . PVC statistics
- **ADSL Features**
  - . Full ADSL2+/2/1 standards
  - . Annex B, Q
  - . ITU G.992.1 (G.dmt) / ITU G.992.2 (G.lite)
  - . ITU G.992.3 ADSL2(G.dm.bist)
  - . ITU G.992.5 (G.dmt.bisplus)
  - . Support Multimode
  - . Support Fast Path and Interleave Path
  - . Echo Cancellation
  - . Trellis Coding
  - . Bit Swapping
  - . Seamless Rate Adaption
  - . Network Timing Reference

- **VDSL Features**
  - . ITU G.993.2 (VDSL2)
  - . ITU G.993.5 (vectoring)
  - . Annex B
  - . ATM and PTM modes
  - . VDSL2/ADSL2+ multimode
- **Voice over IP Features**
  - . Call Feature: outgoing, incoming call, Call Waiting, Call Transfer, Caller ID
  - . Call Control: support SIP (RFC3261)
  - . Voice Codec: 711 a-law, G.711 u-law, G.722, G.729A ,
  - . Tone/Ring Signal: ETSI (General Europe)
  - . Tone Generation: support dial tone, busy tone, congestion tone, ring tone
  - . Tone Detection: DTMF
  - . FAX/Analog Modem, T.38 Fax relay
  - . RFC2833 RTP Payload
  - . Echo Cancellation
- **Configuration and Management**
  - . Web Configuration
  - . Telnet Management
  - . SSH Management
  - . SNMP v1/v2
  - . F/W upgrade
  - . TR-069
  - . Diagnostic
- **Power / Operation Environment**
  - . Input 100-240VAC±10%; Output 12VDC
  - . Temperature: 0 to 40°C (32 to 96°F)
  - . Humidity: 20% to 90% (non-condensing)
- **Dimension**
  - . 217.4 mm x 193.4 mm x 67.5mm
- **Weight**
  - . 650mg approx.
- **Hardware**
  - . One Power Connector
  - . One Power ON / OFF switch
  - . One RJ11 Connector for DSL
  - . Four RJ-45 100/1000 Base-TX Ethernet ports
  - . Two RJ-11 ports (FXS ports) for VOIP phone calls
  - . One factory default setting bottom
  - . Three touch keys support status indicated
  - . One SFP to support various SFP modules.
- **DSL Characteristics**
  - . G.992.1 (G.dmt)
  - . G.992.3 (ADSL2)
  - . G.992.5 (ADSL2+)
  - . Support PTM transport and ATM transport for user data
  - . Support VDSL2 profiles 35b
  - . ATM and PTM (dual-priority)
  - . VDSL2/ADSL+/ADSL2/ADSL multimode
- **Wireless LAN Characteristics**
  - . IEEE 802.11ac, 802.11b, 802.11g, and IEEE 802.11n
  - . Operating frequency: 5GHz and 2.4GHz Channel
  - . Raw data rate: up to 54Mpbs
  - . RF Output Power: 15 ± 1.5 dBm in 2.4GHz
  - . RF Output Power: 23dBm ± 1.5 in 5GHz
  - . Wireless security: 802.1x, WPA2/WPA2-PSK,
  - . Access Control function
  - . Multiple SSIDs
  - . Wireless QoS (WMM)
  - . WMM U-APSD power saving mode
  - . WPS

Figure A.13: Askey datasheet

A.8 Wireshark statistics

Download and upload speeds are in the scale of Mbps. Total_TCP refers to the all TCP connection from beginning of the tracing until the end. TCP_DL and TCP_UP are the number of TCP connections dedicated to download and upload measurements respectively.

OS	Connection	Browser	Platform	Download	Upload	Total_TCP	TCP_DL	TCP_UP
MAC	Wire	Safri	Public	941.22	868.74	117	6	6
MAC	Wire	Safri	Private	943.8	900.1	47	6	5
MAC	Wire	Chrome	Public	347.82	807.41	177	6	6
MAC	Wire	Chrome	Private	294.4	797.4	21	6	6
MAC	Wire	Firefox	Public	404.08	383.38	202	13	6
MAC	Wire	Firefox	Private	490.1	432.0	46	15	6
MAC	Wifi	Safri	Public	705.2	835.38	62	6	6
MAC	Wifi	Safri	Private	705.7	853.1	32	6	6
MAC	Wifi	Chrome	Public	657.45	832.86	170	6	6
MAC	Wifi	Chrome	Private	678.3	854.5	16	6	6
MAC	Wifi	Firefox	Public	408.82	333.97	178	13	6
MAC	Wifi	Firefox	Private	408.2	337.5	57	14	6
Windows	Wire	Edge	Public	375.45	146.92	121	6	4
Windows	Wire	Edge	Private	323.3	486.6	141	6	21
Windows	Wire	Chrome	Public	353.89	68.48	190	6	6
Windows	Wire	Chrome	Private	344.2	784.9	89	6	6
Windows	Wire	Firefox	Public	407.64	64.98	211	6	6
Windows	Wire	Firefox	Private	302.4	336.1	100	32	6
Windows	Wifi	Edge	Public	410.26	53.98	175	6	4
Windows	Wifi	Edge	Private	415.8	151.5	45	6	27
Windows	Wifi	Chrome	Public	263.41	48.18	202	6	6
Windows	Wifi	Chrome	Private	320.1	244.6	52	6	5
Windows	Wifi	Firefox	Public	399.03	44.72	144	6	6
Windows	Wifi	Firefox	Private	311.7	135.0	54	32	6

Table A.7: Milan TCP Streams

OS	Connection	Browser	Platform	Download	Upload	Total_TCP	TCP_DL	TCP_UP
MAC	Wire	Safri	Public	943.66	150.53	50	6	4
MAC	Wire	Safri	Private	940.0	915.2	14	6	6
MAC	Wire	Chrome	Public	395.08	122.02	104	6	6
MAC	Wire	Chrome	Private	435.3	888.8	41	6	6
MAC	Wire	Firefox	Public	552.08	120.64	217	6	6
MAC	Wire	Firefox	Private	460.8	409.2	108	12	6
MAC	Wifi	Safri	Public	572.95	814.2	50	6	6
MAC	Wifi	Safri	Private	623.3	840.2	19	6	6
MAC	Wifi	Chrome	Public	602.13	843.23	84	6	6
MAC	Wifi	Chrome	Private	668.6	799.6	34	6	6
MAC	Wifi	Firefox	Public	391.32	326.29	121	13	6
MAC	Wifi	Firefox	Private	388.2	331.7	51	14	6
Windows	Wire	Edge	Public	207.87	66.73	130	6	6
Windows	Wire	Edge	Private	222.3	443.4	56	6	32
Windows	Wire	Chrome	Public	191.67	58.08	158	6	6
Windows	Wire	Chrome	Private	292.5	714.3	72	6	6
Windows	Wire	Firefox	Public	234.7	51.39	171	6	6
Windows	Wire	Firefox	Private	199.9	227.1	114	30	6
Windows	Wifi	Edge	Public	413.64	54.41	81	6	6
Windows	Wifi	Edge	Private	431.2	186.4	50	6	25
Windows	Wifi	Chrome	Public	323.56	50.58	142	6	6
Windows	Wifi	Chrome	Private	349.4	227.8	68	6	6
Windows	Wifi	Firefox	Public	363.11	31.93	123	6	2
Windows	Wifi	Firefox	Private	353.7	141.9	52	32	6

Table A.8: Rome TCP Streams

Bibliography

[1] M. M. Al-Quzwini, "Design and implementation of a fiber to the home ftth access network based on gpon," *International Journal of Computer Applications*, vol. 92, no. 6, 2014.

[2] D. Gutierrez, K. S. Kim, S. Rotolo, F.-T. An, and L. G. Kazovsky, "Ftth standards, deployments and research issues," in *8th Joint Conference on Information Sciences, Salt Lake City, Utah*, pp. 21–26, 2005.

[3] ITU-T, "Gigabit-capable passive optical networks (gpon): General characteristics," 2008.

[4] A. ANILKUMAR, "Simulative performance evaluation of gpon and wdmpon," *International Journal of Latest Research in Science and Technology*, vol. 2, pp. 58–61, 2013.

[5] C. F. Lam, *Passive optical networks: principles and practice*. Elsevier, 2011.

[6] https://support.ookla.com/hc/en-us/articles/115003369327-Speedtest-Custom-Overview.

[7] https://support.ookla.com/hc/en-us/articles/115000234391-How-Does-Speedtest-%20Custom-Work-.

[8] https://support.ookla.com/hc/en-us/articles/234578628-Speedtest-Server-.

[9] https://support.ookla.com/hc/en-us/articles/234578448-What-fields-are-included-.

[10] https://support.ookla.com/hc/en-us/articles/234578408-What-fields-are-included-.

[11] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent.

[12] https://tools.ietf.org/html/rfc2616#section-14.43.

[13] https://blog.router-switch.com/2017/11/what-are-sfp-ports-used-for/.

[14] https://www.pcmag.com/encyclopedia/term/64582/sfp.

[15] https://github.com/ua-parser/uap-python.

[16] https://www.seleniumhq.org/docs/01_introducing_selenium.jsp#introducing-selenium.

[17] Z. H. Stanislav Stresnjak, "Usage of robot framework in automation of functional test," *ICSEA 2011 : The Sixth International Conference on Software Engineering Advances.*

[18] https://en.wikipedia.org/wikiCumulative_distribution_function#cite_note-KunIlPark-1.

[19] https://en.wikipedia.org/wiki/Histogram.

[20] https://www.andata.at/en/software-blog-reader/why-we-love-the-cdf-and-do-not-like-histograms-that-much.html.

[21] https://iandzy.com/histograms-cumulative-distribution/.

[22] https://tools.ietf.org/html/rfc7230#section-6.3.

[23] https://developer.mozilla.org/en-US/docs/Web/HTTP/Connection_management_in_HTTP_1.x.

[24] https://en.wikipedia.org/wiki/HTTP_persistent_connection#Advantages.