

## Samples of transmission opportunities (txops) on Orange 4G commercial network

These txops files allow to emulate some behaviors of one base station deployed in Orange 4G network, in France. Each line of the txops file represents a time (in ms) where the base station can transmit (in uplink or downlink) an MTU-sized packet.

### Measurements of txops on Orange 4G network:

#### Measurements conditions:

The measurements were performed on Orange 4G network in January 2022, in France, and each capture lasts approximatively 10 minutes. We choosed different network conditions based on categorization inspired by the ARCEP one on their results on measurements campaigns on the quality of mobile services on 2021 ([Qualité des services mobiles | Arcep](#)). We go beyond and use a finer categorization of cellular network conditions under 5 static configurations based on the average downlink throughput. We also perform measurement inside a car traveling at the steady speed of 110km/h on a highway (RN12 between Guingamp and Lannion, in France) to capture txops in mobility conditions.

The table below summarizes the measurements conditions.

Conditions	Average downlink throughput (Mbps)	Location	Hour
File 1	220	Orange Lannion	2pm
File 2	160	Orange Lannion	11am
File 3	120	Lannion (Brélévenez)	9am
File 4	80	Lannion (Brélévenez)	1pm
File 5	40	Plemeur Bodou	3pm
File 6	45	Guingamp – Lannion	10am

### Usage:

The Mahimahi Linkshell tool can be installed from [Mahimahi \(mit.edu\)](#). It works on Ubuntu 14.04 and later and can be used with command line. It takes as main arguments the uplink txops file and the downlink txops file. Some optional arguments can be specified such as the type of up/downlink queue.

```

Usage: mm-link UPLINK-TRACE DOWNLINK-TRACE [OPTION]... [COMMAND]

Options = --once
          --uplink-log=FILENAME --downlink-log=FILENAME
          --meter-uplink --meter-uplink-delay
          --meter-downlink --meter-downlink-delay
          --meter-all
          --uplink-queue=QUEUE_TYPE --downlink-queue=QUEUE_TYPE
          --uplink-queue-args=QUEUE_ARGS --downlink-queue-args=QUEUE_ARGS

QUEUE_TYPE = infinite | droptail | drophead | codel | pie
QUEUE_ARGS = "NAME=NUMBER[, NAME2=NUMBER2, ...]"
              (with NAME = bytes | packets | target | interval | qdelay_ref | max_burst)
              target, interval, qdelay_ref, max_burst are in milli-second

(base) iteq@iteq:~/Datasets/orange_traces$
(base) iteq@iteq:~/Datasets/orange_traces$ mm-link orange_file1_01_2022_up.pps orange_file1_01_2022_down.pps
[link] (base) iteq@iteq:~/Datasets/orange_traces$

```

### Experiment protocol:

Based on the saturator tool ([GitHub - keithw/multisend](https://github.com/keithw/multisend)), the experiment to generate the txops files, consists in capturing network traces samples by saturating link radio of a User Equipment (UE) and the base station. Our testbed is constituted of 02 UE (Smartphone Xiaomi MI M2007J3SY 5G), a Linux Laptop (Ubuntu 20.04) and a Linux Server (Ubuntu 18.04). A UE, USB-tethered to the laptop is used for saturation in both directions. The other UE, Wi-Fi-tethered to the laptop, is used for sending and receiving Acks. This latter is not saturated.

During traces captures, each time the UE sends data to the server, it inserts its timestamp in the packets and when it receives the Ack associated to the packets it can compute the RTT. This RTT value is used to adjust the client congestion window according to the following rules:

$$\begin{aligned}
 & \text{if } (rtt < RTT_{lower} \ \&\& \ congestion \ window < window_{upper}) \text{ then } congestion \ window ++ \\
 & \text{if } (rtt > RTT_{upper} \ \&\& \ congestion \ window > window_{lower}) \text{ then } congestion \ window -- = 20
 \end{aligned}$$

The same process is applied on the server-side, so that it can modify its congestion window accordingly and hence saturate the link radio of the base station.

The description of the functioning of our measurement tool can be summarized in the figure below:

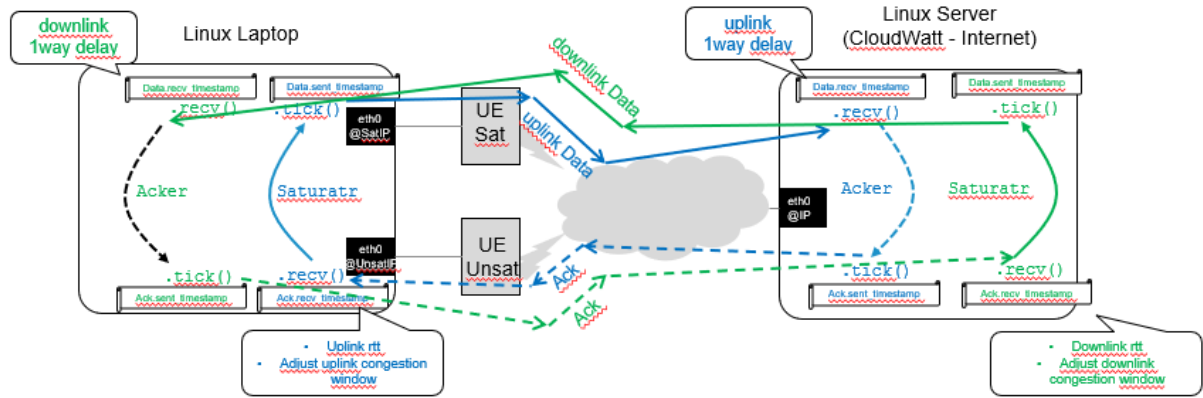


Figure 1: Measurement tool: saturator

After capturing traces, we can generate a Mahimahi txops file, that will be used by Mahimahi tool ([GitHub - ravinet/mahimahi: Web performance measurement toolkit](https://github.com/ravinet/mahimahi)), Link Shell, to emulate time varying cellular conditions, by scheduling packet transmissions at an interface at a given network condition, based on the transmission opportunities file that were generated at this condition.

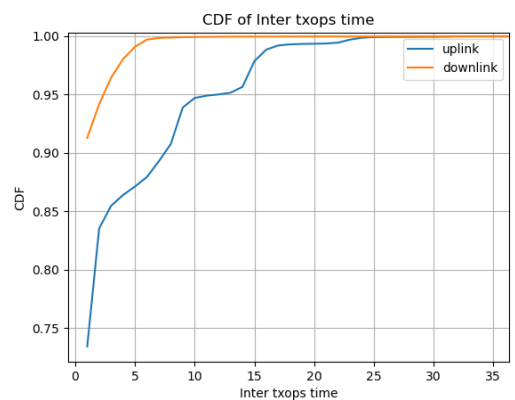
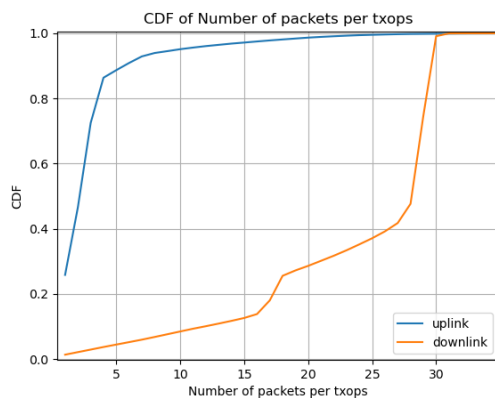
### Characterization of the txops files:

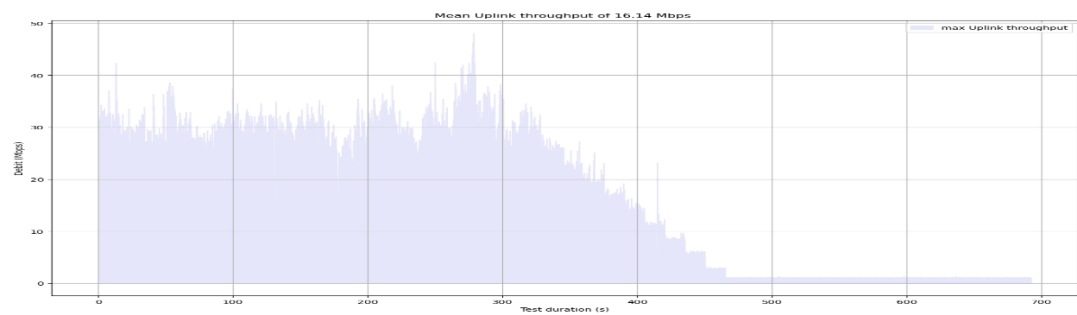
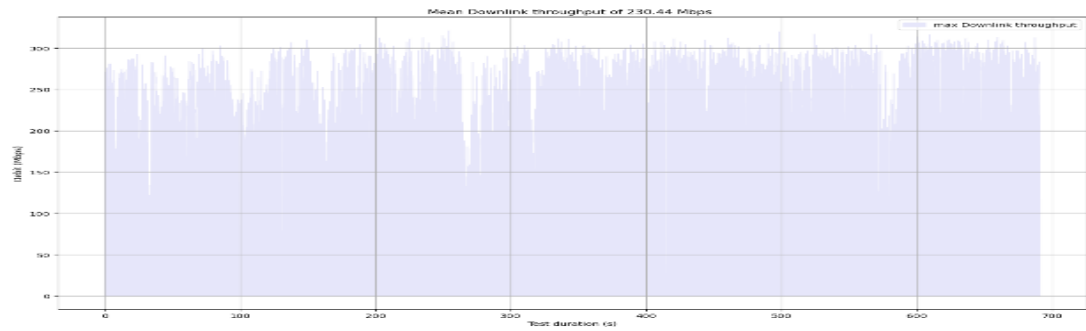
The following pictures present the Cumulative Distribution Function (CDF) of the number of packets that are sent per txops and the CDF of the time between txops, each for uplink and downlink flows. We also add the mean uplink and downlink throughput that were measured during each capture and the txops file name associated.

File 1:

Downlink throughput= ~219Mbps - Uplink throughput= ~27Mbps

Files name: orange\_file\_1\_01\_2022\_XX.pps (XX in {down; up})

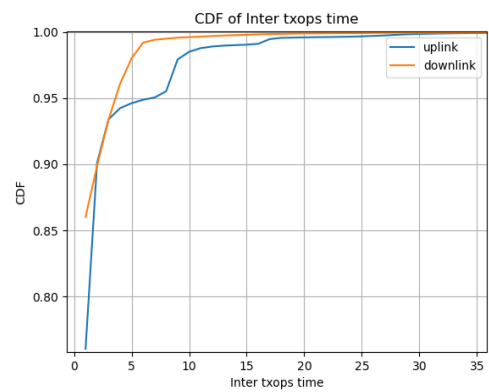
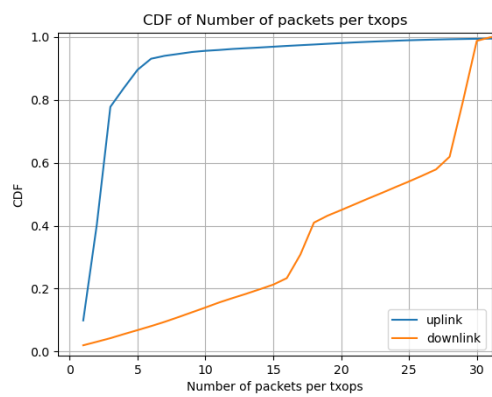


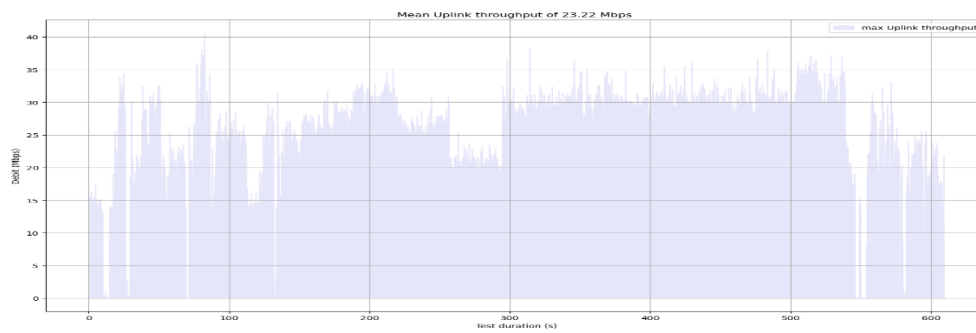
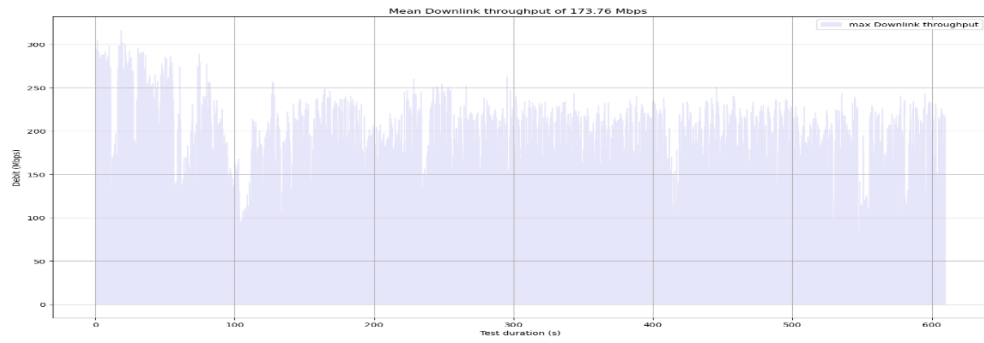


File 2:

Downlink throughput= ~154Mbps - Uplink throughput= ~16Mbps

Files name: *orange\_file\_2\_01\_2022\_XX.pps* (XX in {down; up})

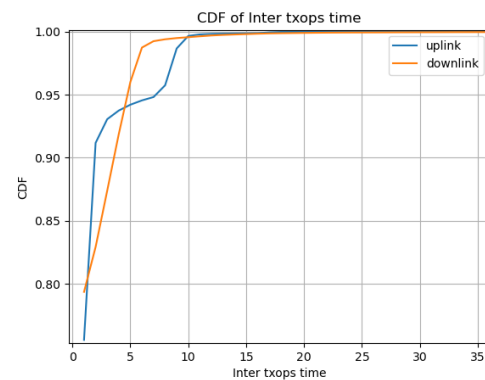
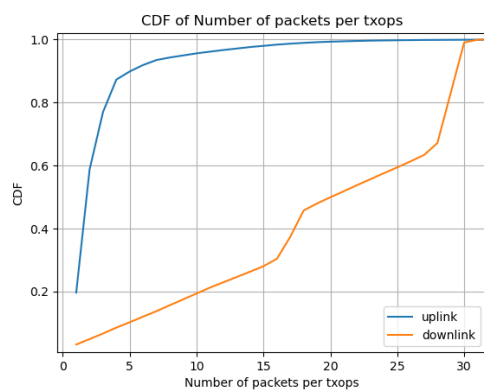


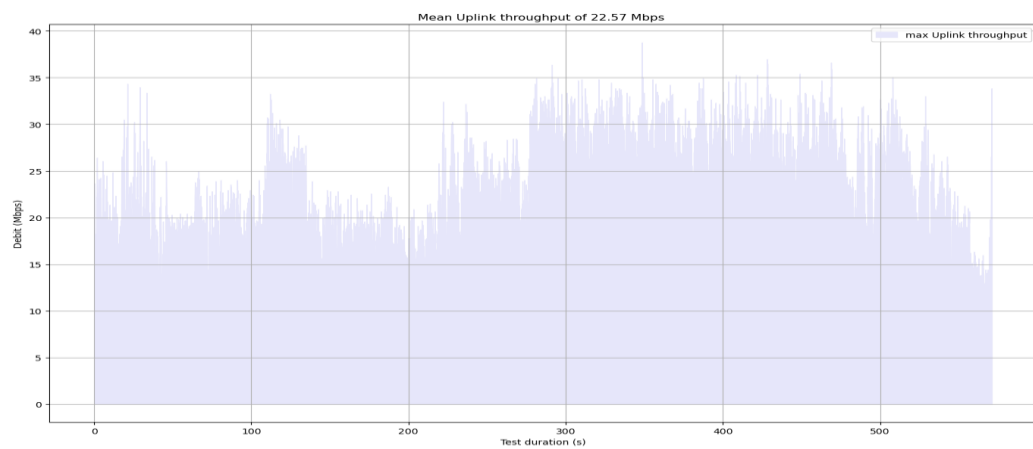
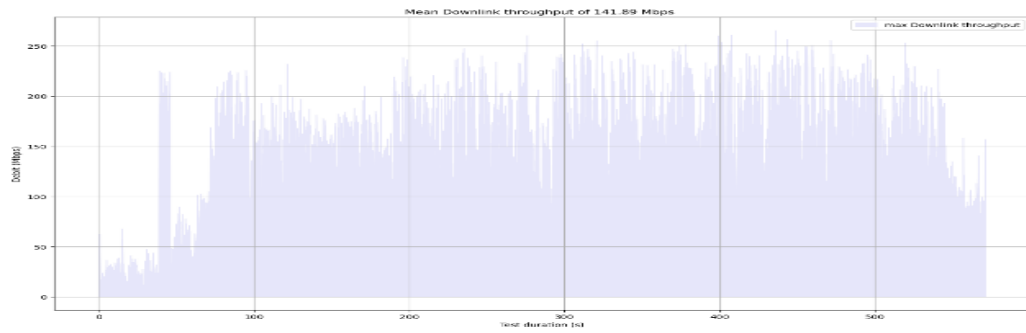


File 3:

**Downlink throughput= ~122Mbps - Uplink throughput= ~11Mbps**

**Files name: *orange\_file\_3\_01\_2022\_XX.pps* (XX in {down; up})**

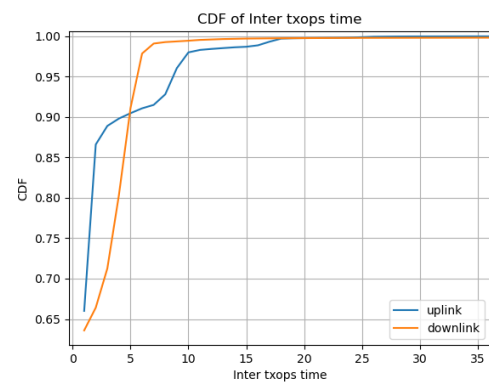
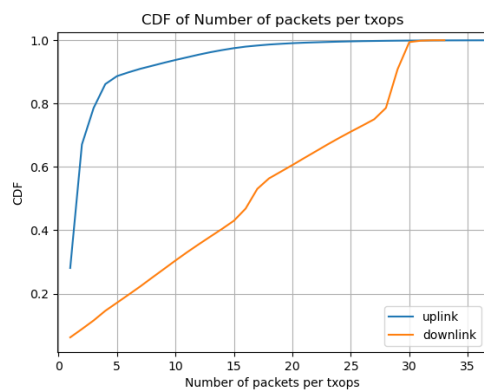


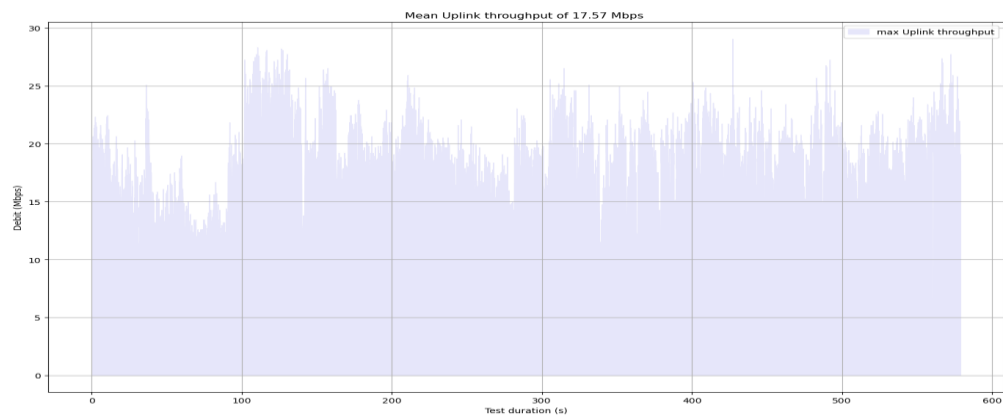
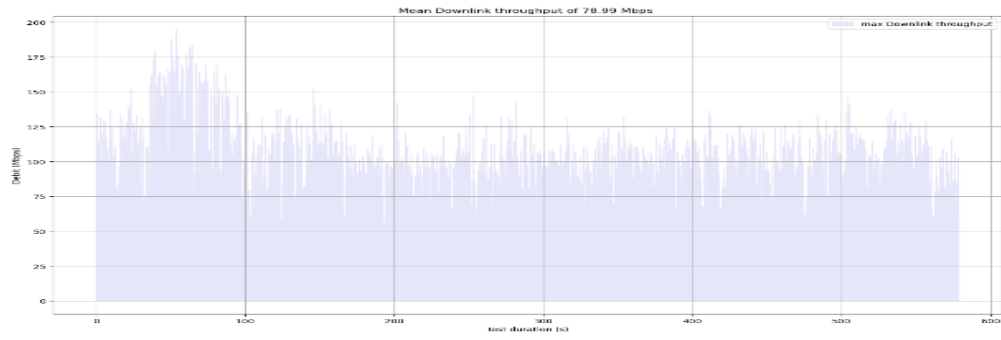


File 4:

Downlink throughput= ~84Mbps - Uplink throughput= ~17Mbps

Files name: *orange\_file\_4\_01\_2022\_XX.pps* (XX in {down; up})

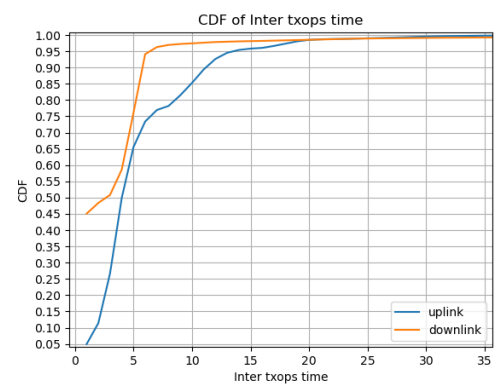
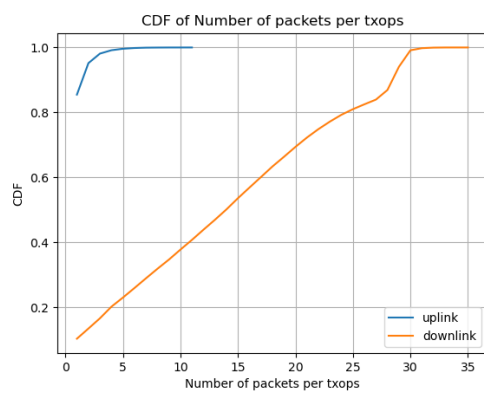


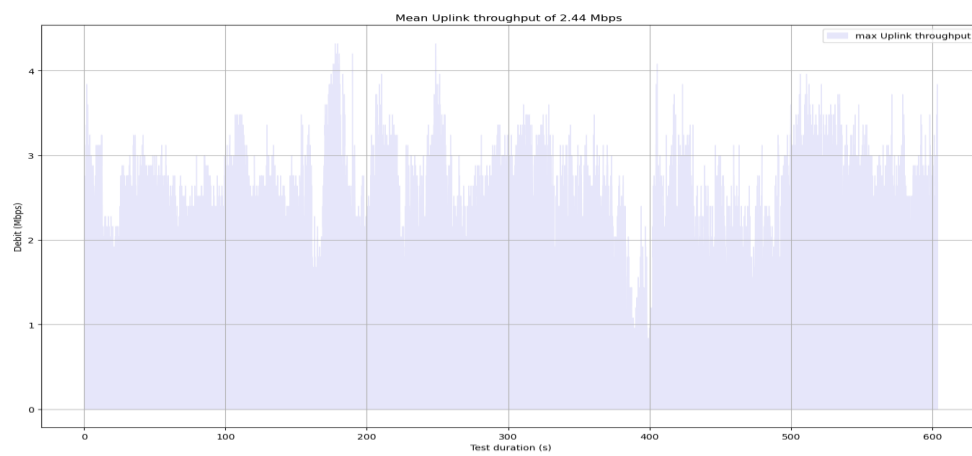
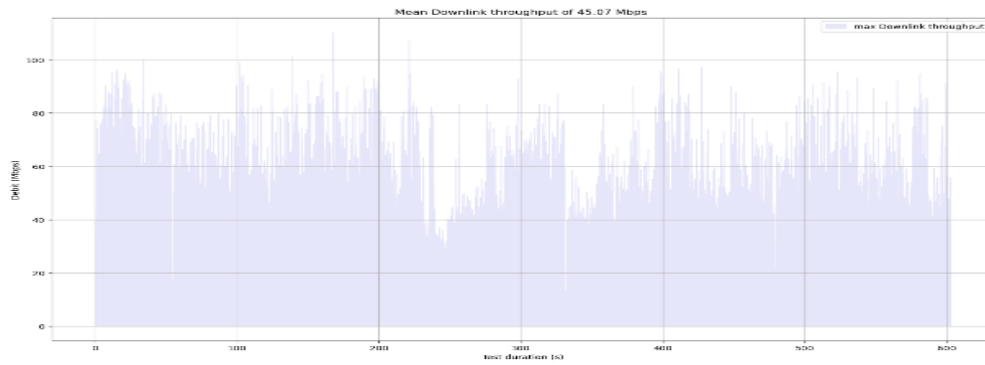


File 5:

Downlink throughput= ~47Mbps - Uplink throughput= ~2Mbps

Files name: *orange\_file\_5\_01\_2022\_XX.pps* (XX in {down; up})





File 6:

**Downlink throughput= ~45Mbps - Uplink throughput= ~6Mbps**

**Files name: orange\_file\_6\_01\_2022\_XX.pps (XX in {down; up})**

