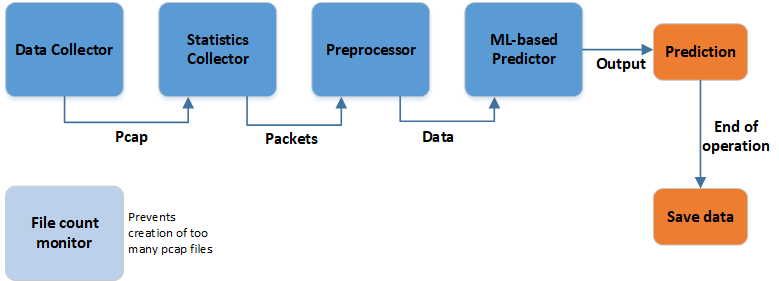
# Overview

The pipeline consists of 4 different modules which run in parallel to predict video events in real-time. Each module gets activated when it receives the input from the previous module, therefore the framework acts as a pipeline. These modules are:

1. Data Collector: Monitors the network and creates a pcap file each t seconds (t=1 in our case)
2. Statistics Collector: Extracts useful statistics from the raw pcap data
3. Preprocessor: Prepares the extracted information to be passed to the ML-model
4. ML- Based Predictor: In this module we have the previously trained machine learning model which makes the predictions on the newly arriving data at each second. The output is printed both on the command-line in real-time and also into a csv file which is saved once the execution is finished. The ML model that is used inside this module can be changed when necessary.

Other than the modules, there is also a file count monitor which deletes the older pcap files created in the previous seconds that are no longer relevant for the operation.



For more detailed information about the pipeline, testbed and the model training/testing you can read our short paper on “Machine Learning based KPI Monitoring of Video Streaming Traffic for QoE Estimation, available on the repository.

# Requirements

## OS & Software Requirements

* Linux Environment (tried on Ubuntu 18 & 20)
* Tcpdump (make sure to give tcpdump the necessary permissions to capture packets and write pcaps)
* Python3
* (optional) Android device with QTS app & root
* (optional) tc tool for emulating bad network conditions

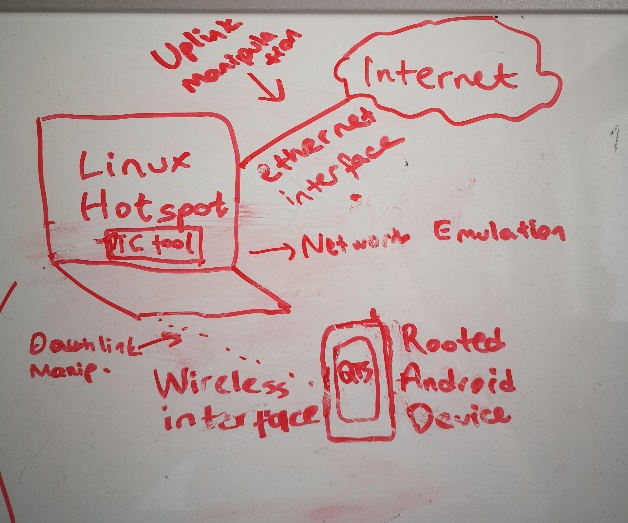
## Python Libraries

* Dpkt
* OS
* Netifaces
* Numpy
* Pandas
* Scikit-learn

Note: Depending on your system (and Python versions) there might be other dependencies which requires extra installations. Feel free to ask me if you run into further problems with installation and setting up.

## Operational Requirements

If you want to stream videos using the mobile device, you will have to create a bridge network where you set your computer to serve as a “WiFi Access Point” to be able to capture the traffic on the mobile devices and also to emulate bad network conditions. This is how I set up my testbed:



* **Fig: Example Testbed**
* For more information on setting up the AP see: <https://askubuntu.com/questions/180733/how-to-setup-an-access-point-mode-wi-fi-hotspot>
* For the network emulation you can use the script that I have written which switches between different scenarios (set of rules) using tc-tool to limit bandwidth and introduce packet loss & delay to trigger buffering events/resolution changes during streaming.
* Additionally, some extra files are necessary for running the pipeline. These are:

Config.py: Defines some parameters for the pipeline. The most important ones are the network interface to be used (NETWORK\_INTERFACE) and time granularity (T\_OBSERVATION) parameters.

# Usage

1. Start the traffic emulation script (make sure to navigate to the directory of the code)

**$ sudo python3 emulation.py**

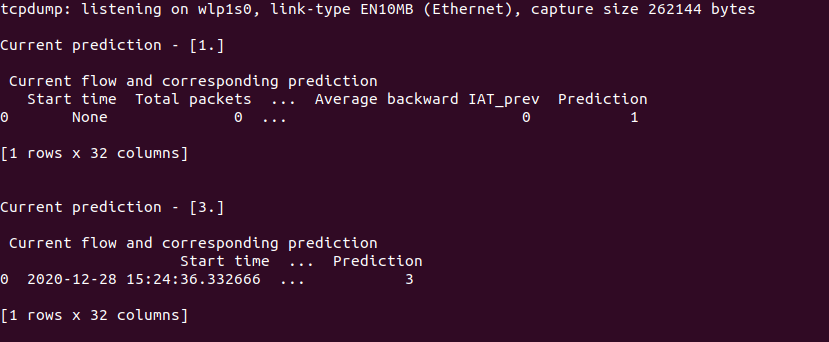
The rules will start to apply automatically until the program is terminated (ctrl+c)

You can change the parameters (e.g duration, delay etc) in the script easily if you want to.

1. Start the video streaming on the mobile device: For establishing the ground truth you can use the QTS App (see QTS App Manual) and start the videos automatically. When the video finishes, QTS App saves a csv file automatically where you can observe all the video events and their timestamps and so you can compare the predictions of the ML model with the correct results using the timestamps.
2. Start the pipeline in a separate terminal window:

**$ sudo python3 youtube\_framework.py**

If the pipeline has started correctly, you should start seeing predictions printed on the screen each second.



As it can be seen on the screenshot, Prediction of the current second is given as 0,1,2 or 3. The meaning of these numbers are:

0: Buffering

1: Playing

2: Quality Downgrade

3: Quality Upgrade

The pipeline will continue to run until terminated (ctrl+c).

4. Once the pipeline is terminated, the results (statistics and predictions) will be also available as a csv file which will be saved to the same directory. Using this csv file and the other csv file generated by the QTS App (saved into the memory of the mobile device, see the QTS App manual) can be compared to see whether the predictions were correct.