Wireless internet - WiFi traffic classification.

Luca Ferraro (10748116), Fabio Losavio (10567493)
https://github.com/LucaFerraro/Wireless_internet_project

A.Y. 2019/2020

Contents

List of Figures

Listings

1 Introduction

The aim of this project is using the protocol analyzer *Wireshark* for classifying traffic at *MAC* layer.
To reach this goal, we used *Wireshark* in *monitor mode*: in this mode, the *WiFi* module of our PC intercepts all the traffic in range that is transiting via *WiFi*. This is a limitation is the sense that at *MAC* layer it's possible to check whether a frame is transporting data, but it's not possible to distinguish the application that has generated the data carried in the payload of the frame. Therefore, our analysis only provides a quantitative estimation of the traffic of data transported over *WiFi*. The reason why we have used this solution even if we were aware of this fact is that working in *monitor mode* is the only solution we had to sniff traffic that was not meant only for the computer that was performing the capure.

1.1 Our analysis.

In this project, what we have done is looking at the *data* and *QoS data* frames captured by our computers and try to understand what those packets are meant to in terms of type of application that has generated them. As specified, this can only be a deduction: we can't be sure of the application that has generated a packet at *MAC* layer.
To do this we have scanned many sample captures and for each one of them:

1. Found all the *MAC* addresses that have generated/received at least one *data/QoS data* frame.
2. Associated to all the *MAC* addresses we have revealed:
   - The number of transmitted/received bytes.
   - The number of transmitted/received packets.
   - The average uplink/downlink rate.
3. Found the vendor associated with the *MAC* address.

In order to analyze the captured traffic and try to understand the type of application a user (so a *MAC*) was using, we have plotted:

- A pair of histograms (one for the bytes, one for the number of packets) in which there is a bar for each *MAC* address that have generated/received at least a certain amount of packets (better description of this later on).
- A graph for the cumulative uplink/downlink traffic (considering then all the bytes generated/received by all the revealed *MACs*).
- A graph for the discrete traffic in uplink/downlink, that is the total amount of bytes exchanged by all the revealed *MACs* in a fixed time interval (customizable in the program).

2 Project description

2.1 Program setup

To launch and use the program, first some pre-requisites have to be meet:

- install pyshark library in order to analyze the capture file
- install numpy library to perform some numerical analysis
- install matplotlib library to visualize plots

The program can run in 3 different modes selected by adding an attribute to the standard python call, using the command `python Traffic_analyzer.py -file "PATH TO A FILE"` the program will scan the file provided after the *-file* parameter. In this mode the program will open the selected capture (in a .pcap/.pcapng format) and start scanning all the packets in order to obtain some information.

The second mode is the live capturing mode in which, using the command `sudo python Traffic_analyzer.py -live "INTERFACE" "DURATION"` the program will first start a capture on the given interface,that has to be enabled to work in monitor mode, for a given amount of time provided with the "DURATION" attribute, will save this capture and will work on it. With this mode administrator privileges are required to allow tshark to start the capture in monitor mode.

The last mode is a default mode, launched using the command `python Traffic_analyzer.py`, which will perform the analysis on a default capture inserted in the code. This mode is mostly used as a debug tool but can also be useful to understand how the output will look like.

2.2 Program structure

To obtain information from a capture, the code will run a for loop on all the packets in the capture file and will analyze only those which can be useful for our purpose. To select the useful the following filter is used:

```
1     (int(packet.wlan.fc_type) == 2) and
2     ((int(packet.wlan.fc_subtype) >= 0 and
3         int(packet.wlan.fc_subtype) <= 3)) or
4     (int(packet.wlan.fc_subtype) >= 8 and
5     int(packet.wlan.fc_subtype) <= 11)
```

Listing 1: Packet filter

it will select all the **data** frames (wlan.fc_type == 2) and among those it will select the ones actually containing data or QoS data excluding null packets or ACKs. From those packets it will extract the destination address and will add it to a dictionary.