Networking Competition

Analysis of a Wireshark Trace

Nicola DEAN e Marco FASANELLA

SUMMARY

- Sniffing
- How to extract data
- Create a table
- Creative Task 1
- Creative Task 2

SNIFFING

```
1  pkts = sniff(offline=CaptureFileName ,prn=AnalyzePacket,store=0)
```

We called the sniffing function with the following parameters:

- **filename = ””**

- **prn = ””** a personalized function that will be called for each sniffed packet

- **store = 0** to avoid the script from storing all the 800’000 packets on the ram.

HOW TO EXTRACT DATA

```
def AnalyzePacket(packet):
    global i
    DataRow = []
    global TableData

    for pacchetti in packet[IP]:
        try:

            DataRow.append(pacchetti.payload.sport)
            DataRow.append(pacchetti.len)

            TableData.append(DataRow)
            pass
        except:
            pass
```

This function will be used to:

- Analyze the packets flow
- Extract all the required data, in order to be stored in a multidimensional list

CREATE A TABLE

```
1  Table = pd.DataFrame.from_records( TableData, columns=TableColumnsName)
```

Then, the list was converted into a Panda dataframe…

```
1  IpTraffic = Table.groupby("ip_addr")['amount_of_traffic'].sum()
```

And then it has been modified to analyze the information using some of the most useful functions of panda as “Groupby” or “idxMax”.

CREATIVE TASKS

Finally, we spent hours thinking about an original way to solve the assigned task:

- A live graph of the downloading speed of our network
- A live graph showing whenever there is a new connection or one is closed

CREATIVE TASK 1

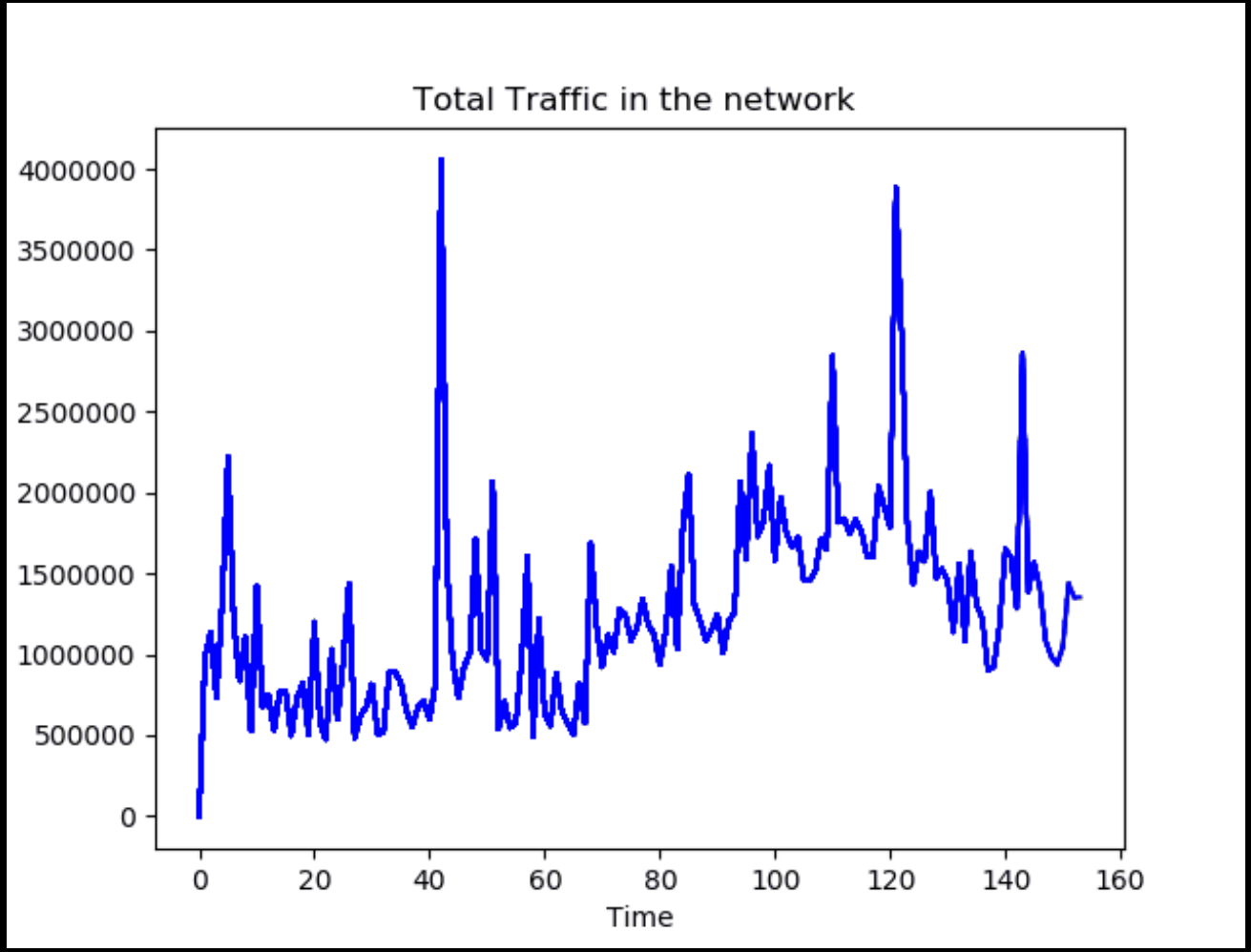
```
for pacchetti in packet[IP]:
        if  packet_time > start_time + intervallo:
            start_time = packet_time
            traffic_per_minute.append(traffic_ammount)
            traffic_ammount = 0
            plt.plot(traffic_per_minute,color="blue")
            plt.pause(0.01)
        else:
            traffic_ammount = traffic_ammount + pacchetti.len
```

For the first task the time field in packet class was extracted from the specific timestamp of the Wireshark sniffing session.

With the following code, a simple way to sum the length of the traffic is now possible.

With a specific time range the connection speed can be calculated.

CREATIVE TASK 1



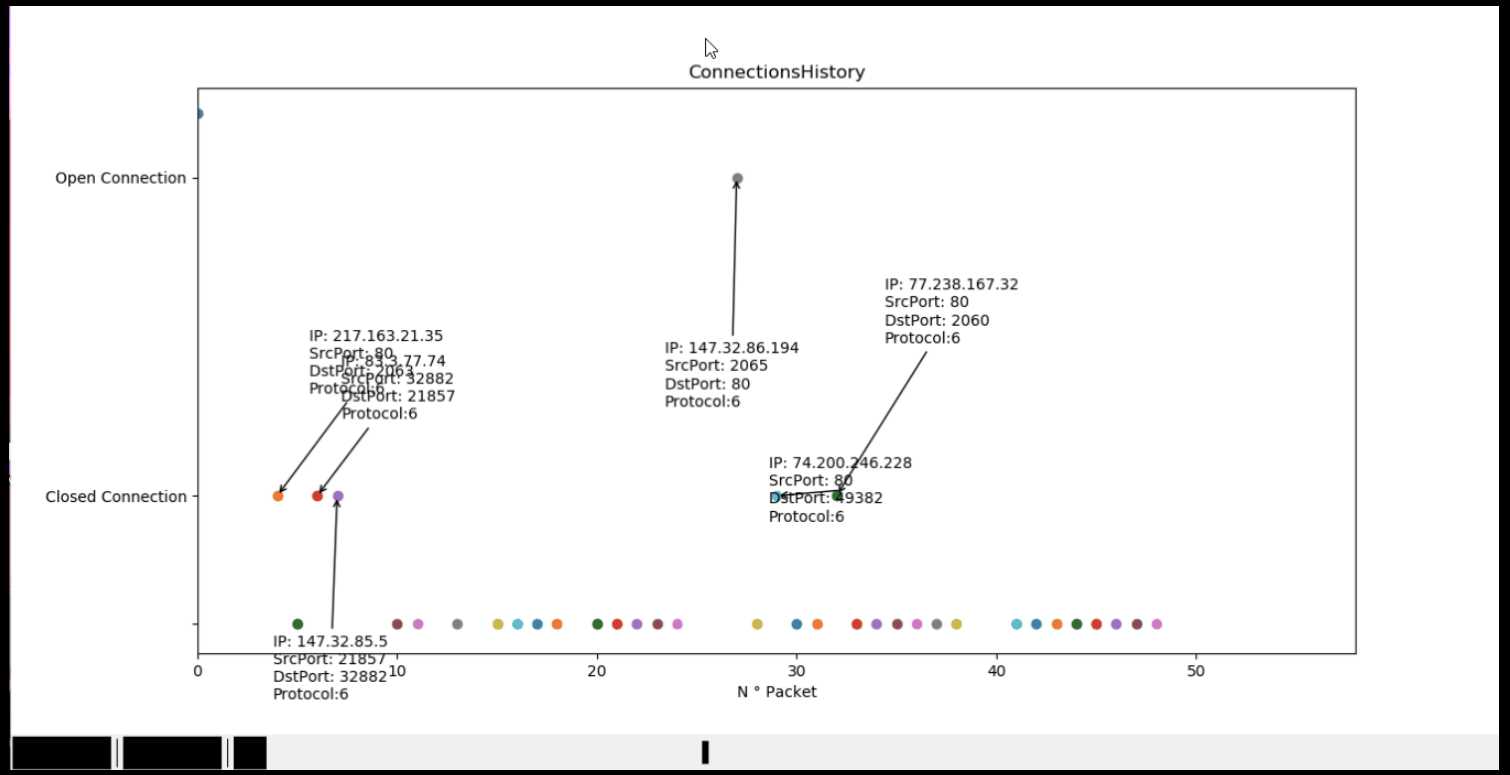
CREATIVE TASK 2

```
1  FIN = 0x01
2  SYN = 0x02
3
4  Flag = pacchetti.payload.flags
5
6  if(Flag & SYN):
7      ....
8  elif(Flag & FIN):
9      ....
```

For the second task a bit to bit "&"(and) is used to extract flags data from "flags" field and understand when a specific flag is on or off.
Using this information, a live scatter graph can now be drawn with a specific height for each different event:

SYN=0 and FIN=0	(Nothing)	-> Zero Level
SYN=0 and FIN=1	(connection closed)	-> Mid Level
SYN=1 and FIN=0	(connection open)	-> Max Level

CREATIVE TASK 2



THANK YOU!