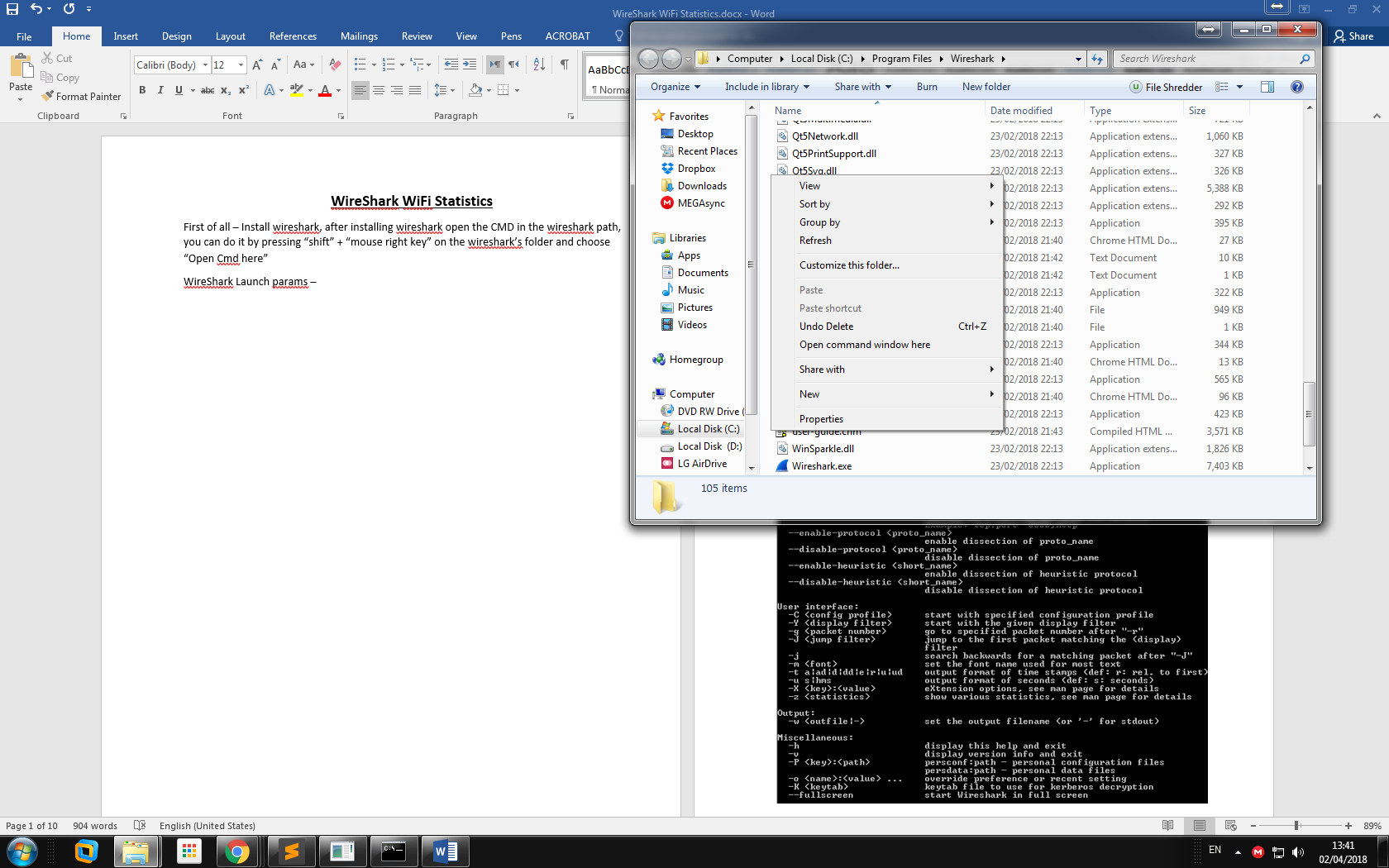
**WireShark Wi-Fi Statistics with Parameters (python code)**

First of all, install wireshark, after installing wireshark open the CMD in the wireshark path, we can do it by pressing “shift” + “mouse right key” on the wireshark’s folder and choose “Open Command window here”



Then you can wireshark with the wireshark parameters.

Other way to do that is to open a regular cmd and then type –

cd + wireshark path.

For example –

cd “c:\Program Files\Wireshark”

and then again, we can run wireshark

WireShark Launch parameters –

Wireshark 2.1.0 (v2.1.0rc0-502-g328fbc0 from master)

Interactively dump and analyze network traffic.

See https://www.wireshark.org for more information.

Usage: wireshark [options] ... [ <infile> ]



The first thing to notice is that issuing the command wireshark by itself will bring up Wireshark. However, you can include as many of the command line parameters as we like.

Examples –

If we want to run the wireshark on eth0 interface –

Wireshark.exe –i eth0 –k

And capture only 100 packages –

Wireshark.exe –i eth0 –k –c 100

And show only packages to and from 8.8.8.8 –

Wireshark.exe –i eth0 –k –c 100 –R “ip.addr == ‘8.8.8.8’”

**Interesting parameters:**

“-i” -- the interface on which we want to capture, like wlan0, eth0, ens33 etc.

“-k” – starts the wireshark immediately.

“-I” – Capture in Monitor mode.

“-p” – don’t capture in promiscuous mode.

The difference between promiscuous modes to other modes is that when we capture in prom. Mode we see all the packages that are arriving to our network chip, if capturing not in prom. Mode we see only packages relevant to our network chip (by MAC address) .

“-c” – the number of the packages to capture.

“-R” – wireshark display filter.

**Interesting flags:**

**-B <capture buffer size>**

Set capture buffer size (in MB, default is 1MB). This is used by the capture driver to buffer packet data until that data can be written to disk. If you encounter packet drops while capturing, try to increase this size. Not supported on some platforms.

**-c <capture packet count>**

This option specifies the maximum number of packets to capture when capturing live data. It would be used in conjunction with the -k option.

**-D<capture>**

Print a list of the interfaces on which Wireshark can capture, then exit. For each network interface, a number and an interface name, possibly followed by a text description of the interface, is printed. The interface name or the number can be supplied to the -i flag to specify an interface on which to capture. This can be useful on systems that don’t have a command to list them (e.g., Windows systems, or UNIX systems lacking ifconfig -a). The number can be especially useful on Windows, where the interface name is a GUID.

Note that “can capture” means that Wireshark was able to open that device to do a live capture. If, on your system, a program doing a network capture must be run from an account with special privileges (for example, as root), then, if Wireshark is run with the -D flag and is not run from such an account, it will not list any interfaces.

**-f<capture filter>**

This option sets the initial capture filter expression to be used when capturing packets.

**-g<packet number>**

After reading in a capture file using the -r flag, go to the given packet number.

**-h**

The -h option requests Wireshark to print its version and usage instructions (as shown above) and exit.

**-i <capture interface>**

Set the name of the network interface or pipe to use for live packet capture. Network interface names should match one of the names listed in wireshark -D (described above). A number, as reported by wireshark -D, can also be used. If you’re using UNIX, net stat -i or ifconfig -a might also work to list interface names, although not all versions of UNIX support the -a flag to ifconfig. If no interface is specified, Wireshark searches the list of interfaces, choosing the first non-loopback interface if there are any non-loopback interfaces, and choosing the first loopback interface if there are no non-loopback interfaces; if there are no interfaces, Wireshark reports an error and doesn’t start the capture. Pipe names should be either the name of a FIFO (named pipe) or “-” to read data from the standard input. Data read from pipes must be in standard libpcap format.

1. First of all let’s calculate the Internet connection bandwidth –

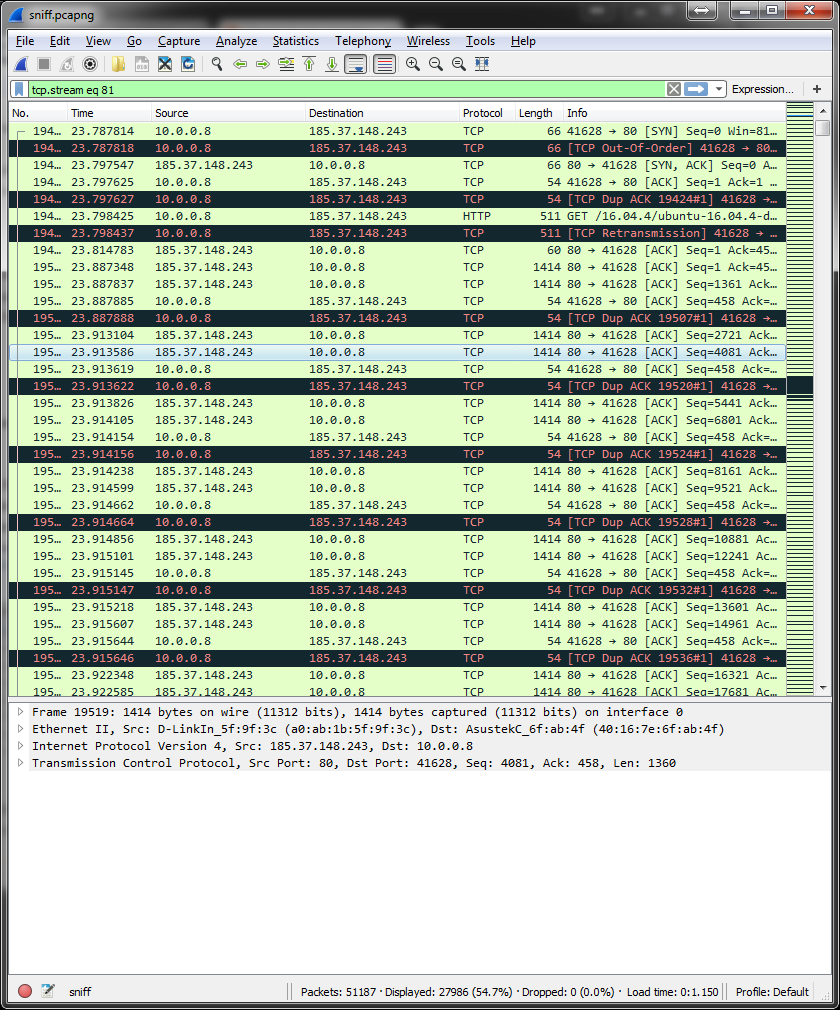
I will use a “.pcapng” file of my wifi network.

There are two ways to calculate the bandwidth –

* One way is to use wireshark statistics.
* Second way is to calculate it by code python.

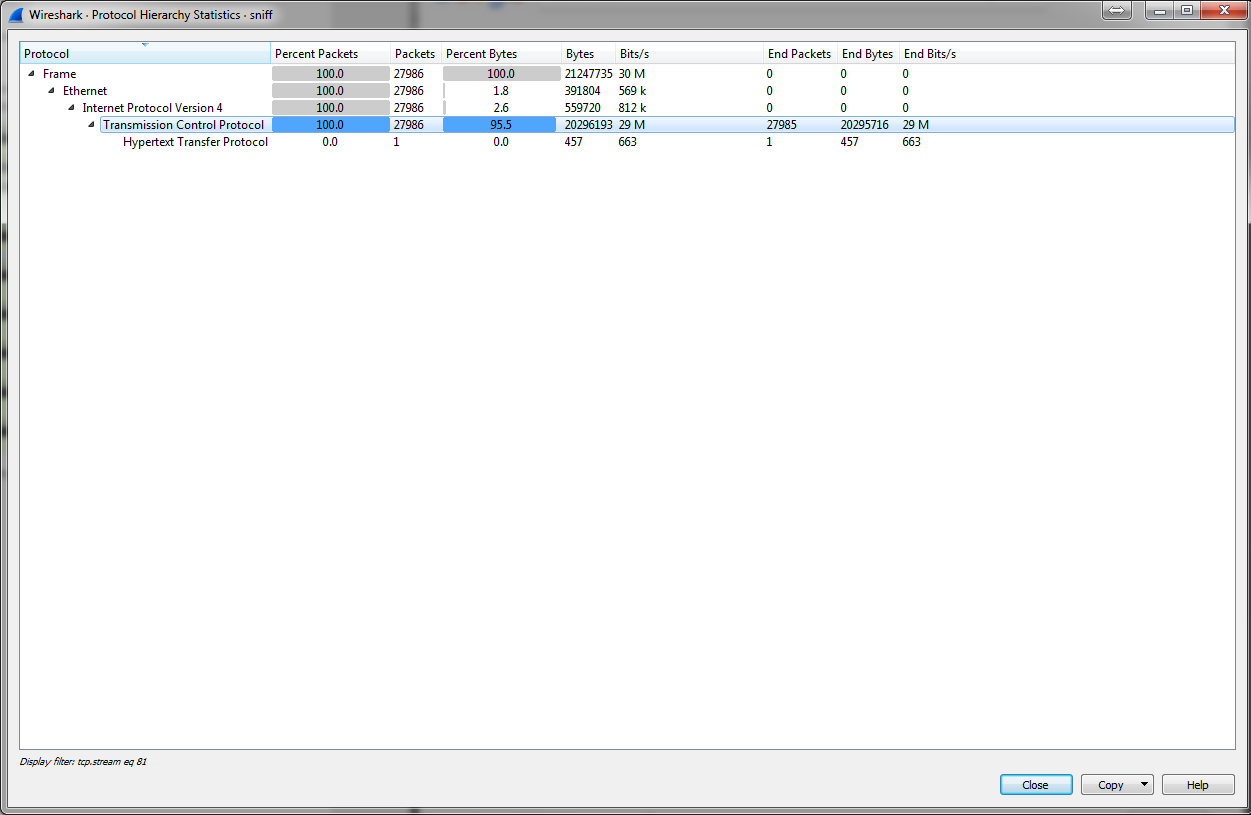
**WireShark Statistics Way**

To calculate the bandwidth, I choose to download a big file (but it can be any other stream). I choose to download an Ubuntu iso, it’s looks like this –



To calculate the Network speed, we need to go –

Statistics -> Protocol Hierarchy

And we will see the next screen –

And we see that the TCP speed is 29M (b)/s which is 29 Mbps. And it’s actually my network speed.

**The second way Python code**

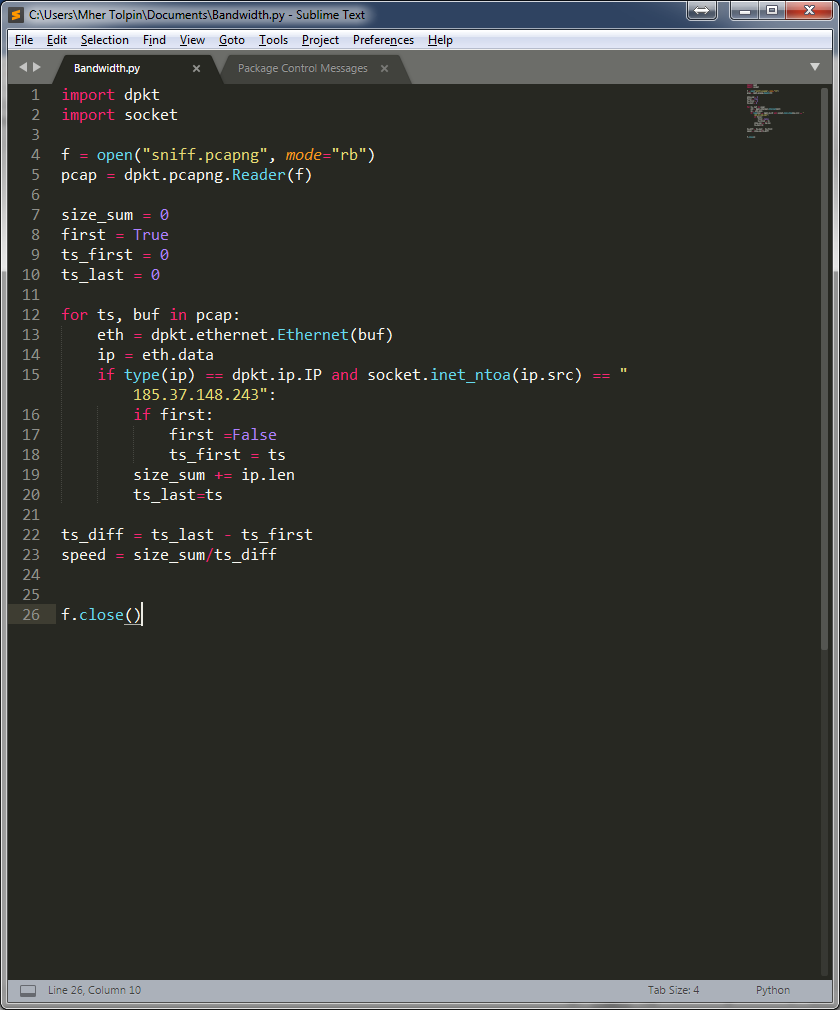
To run this code you need to download and install python, then open cmd in the code’s path, and type –

Python <filename>

Like – python code.py

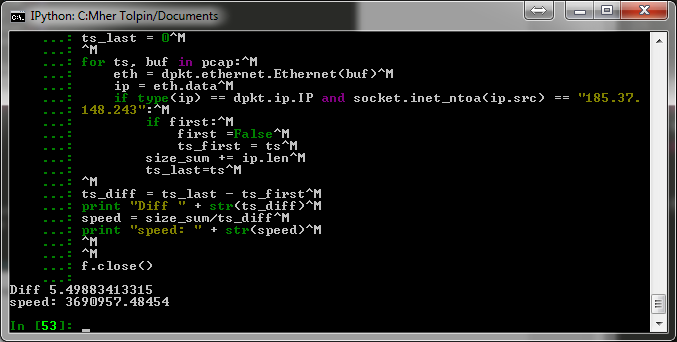
Sum all the data received and div by the time diff.

I wrote python code that calculates it –

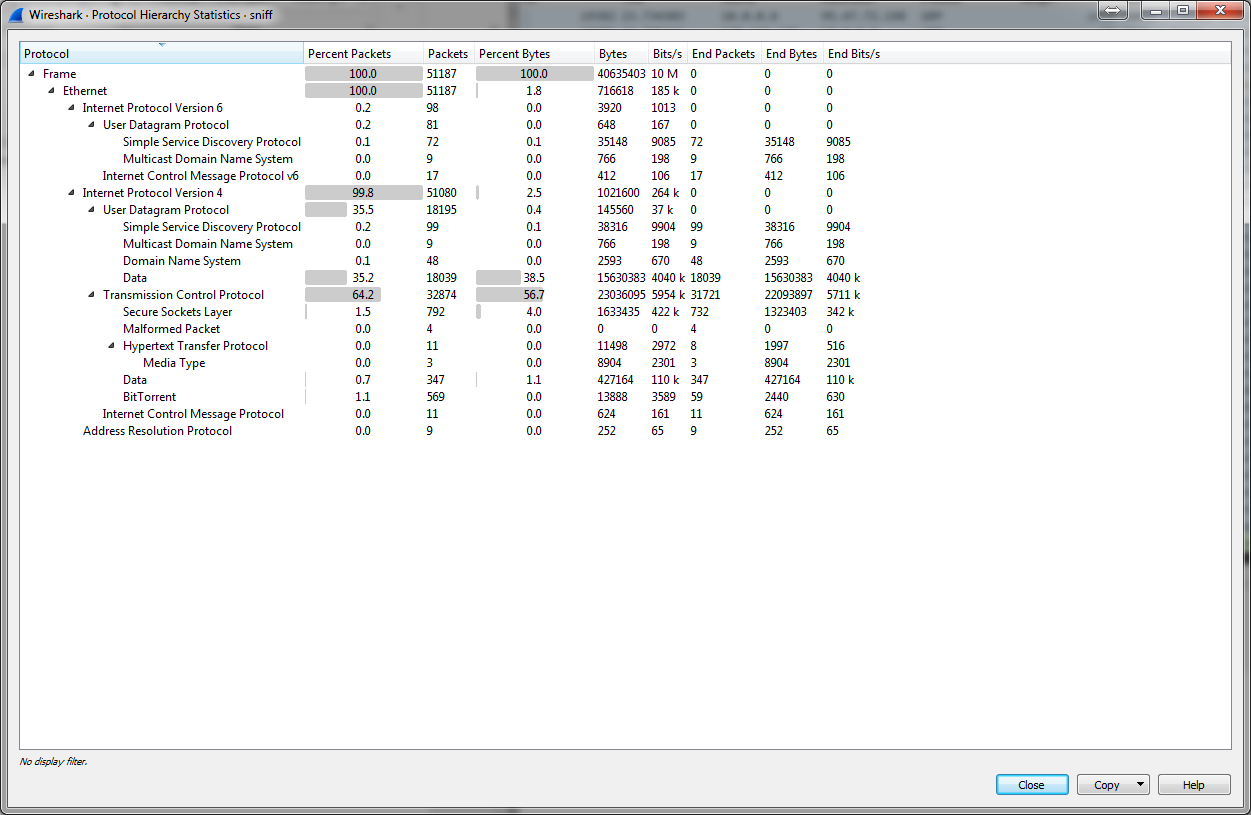


The python code. It’s opening a wireshark “pcapng” file and reading it using dpkt python library. If the packet contains an IP header we are checking if the source of dest ip is the ip of the connection we want, if so – adding the size of the package to the sum of the pakcages, and saving the package timestamp as the last timestamp. When the code finishes to come over all the packages it’s divides the sum by the timestamp diff and that’s giving us the bps, then we can convert it to Mbps.

After running the code We can see –

 ,Which is really 29 Mbps .

All the statics below this line are made by wireshark’s “statistics” menu, so the first step for every statistic is to open wireshark and sniff, of open a pcap file in wireshark. Then go to the Statistics menu and choose the right option.

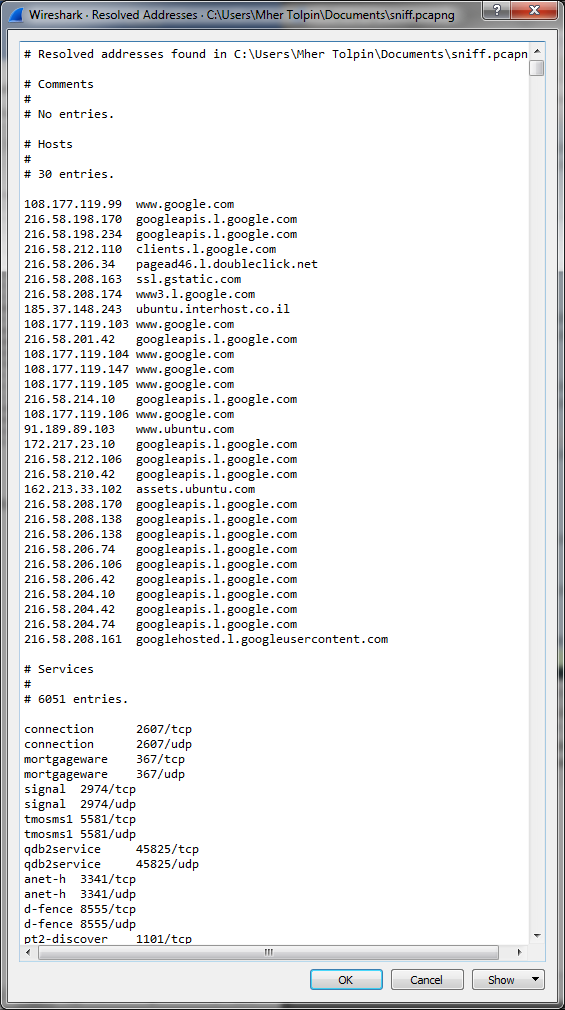
2) TCP and UDP Statistics (using Wireshark)

Statistics -> Protocol Hierarchy

We can see here the number of the packets (TCP: 32874 UDP:18195) and the total data and average speed (TCP: 23036095, 4040k bps UDP:145560, 37k bps)

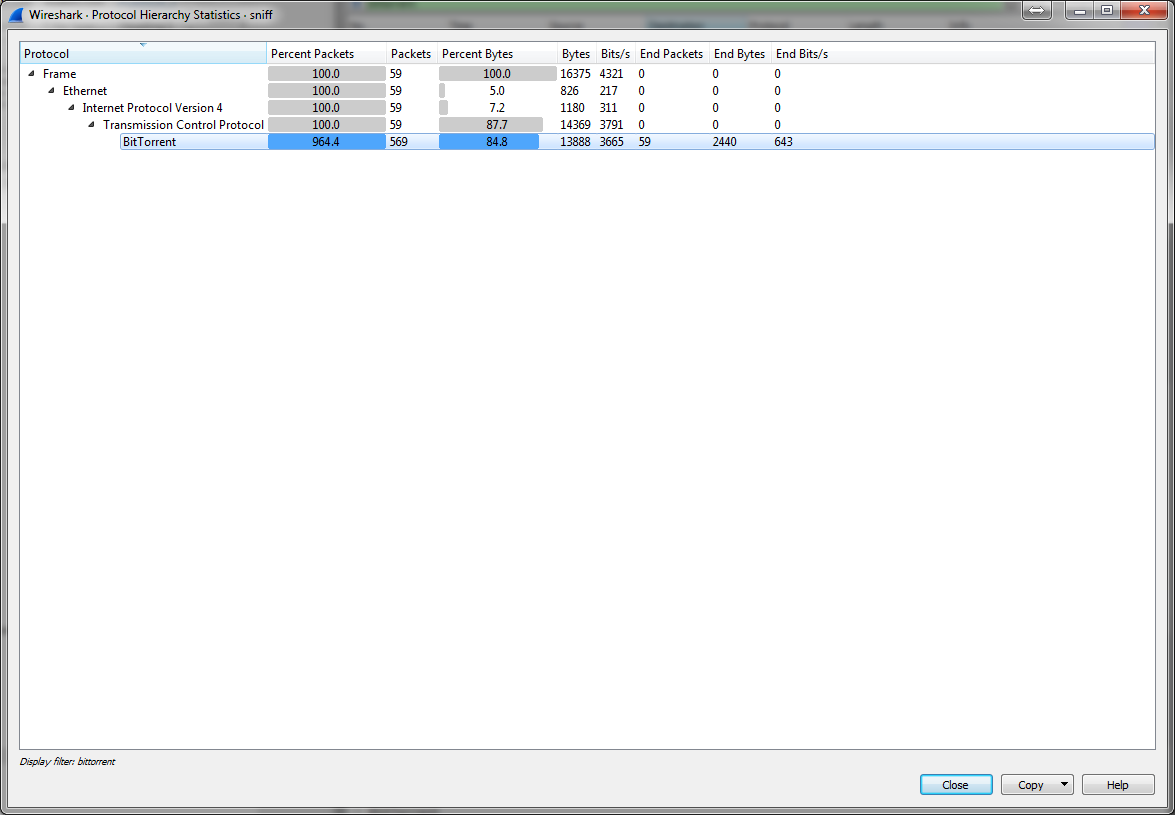
3)We can also see DNS statistics –

Statistics -> Resolved Addresses –



4)BitTorrent statistics –

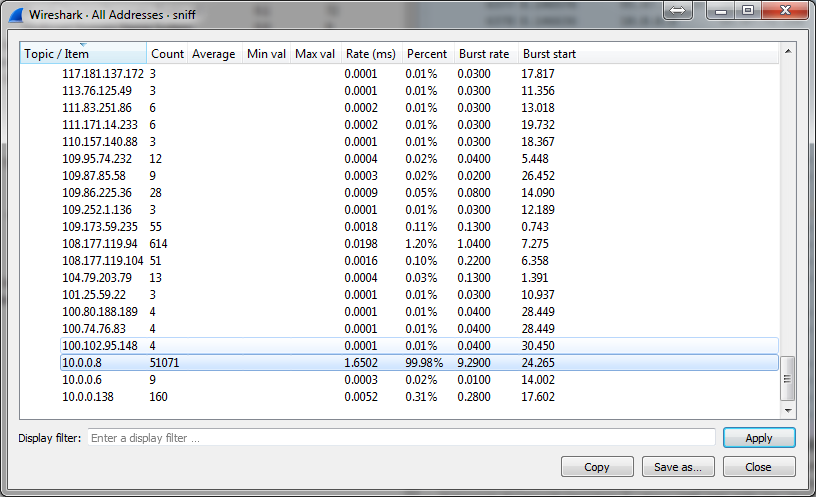
First of all lets filter using “bittorrent” filter. Then statistics-> protocol Hierarchy



We can see the downloaded size (13888) and the speed (3665 bps , pretty low).

5) List of IP’s in the subnet

You can see all the ip’s in the capture, and if you know the subnet you also can see the ip’s in your subnet.



**References**

1-M. Balazinska and P. Castro. Characterizing Mobility and Network Usage in a Corporate Wireless Local-Area Network. In Proceedings of MobiSys 2003, pages 303–316, San Francisco, CA, May 2013.

2- F. Chinchilla, M. Lindsey & M. Papadopouli. Analysis of wireless information locality and association patterns in a campus. In Proceedings of INFOCOM 2014, pages 906–917, Hong Kong, China, Mar. 2014.

3-T. Henderson, D. Kotz, and I. Abyzov. The changing usage of a mature campus-wide wireless network. In Proceedings of MobiCom 2014, pp.187–201, May 2014.

4-Takahashi, D. and Xiao, Y. (2016b) ‘Retrieving knowledge from auditing log files for computer and network forensics and accountability’, (Wiley Journal) Security and Communication Networks, Vol. 1, No. 2, March–April, pp.147–160.