LAB:
Protocol & Packet Analysis
with **wireshark**

… with some focus on IPv6

Sebastian Büttrich, NSRC

Last edit: March 2015

ICTP Workshop on

Scientific Applications for the Internet of Things (IoT)

Protocol & Packet Analysis

Motivation: why are we *sniffing packets*?

- Network analysis
- Problem identification
- Security auditing
- Statistics

Some terms

You will frequently find words like ...

- Dumping = capturing packets from interfaces and saving to file (called a dump)
- Sniffing = looking/listening for packets to dump

Packet dumping & sniffing – some tools

- **Tcpdump**, using libpcap (Linux)
- WinDump, using WinPcap (Windows)
- GUI interface: **wireshark** (Linux, Win, Mac)
- Wireshark without GUI: **TShark**
- Specifically wireless: **Kismet**

Where to capture packets

In order to capture relevant data,

we need to dump packets on network interfaces

that can see the traffic of interest -

e.g.

on a switch, a gateway, a router,

on wireless interfaces in monitor mode

What to capture

Your choice of interface and the software capabilities determine what you will be able to see:

- Don't expect to see wireless traffic on an ethernet (cable) interface
- Don't expect to see Bluetooth or 802.15.4 on an 802.11 interface, even though it is the same frequency

Some typical use cases

(from personal experience … yours will be different)

- Finding Rogue Access Points (ARP, spoofing)
- Identifying sources of broadcast storms

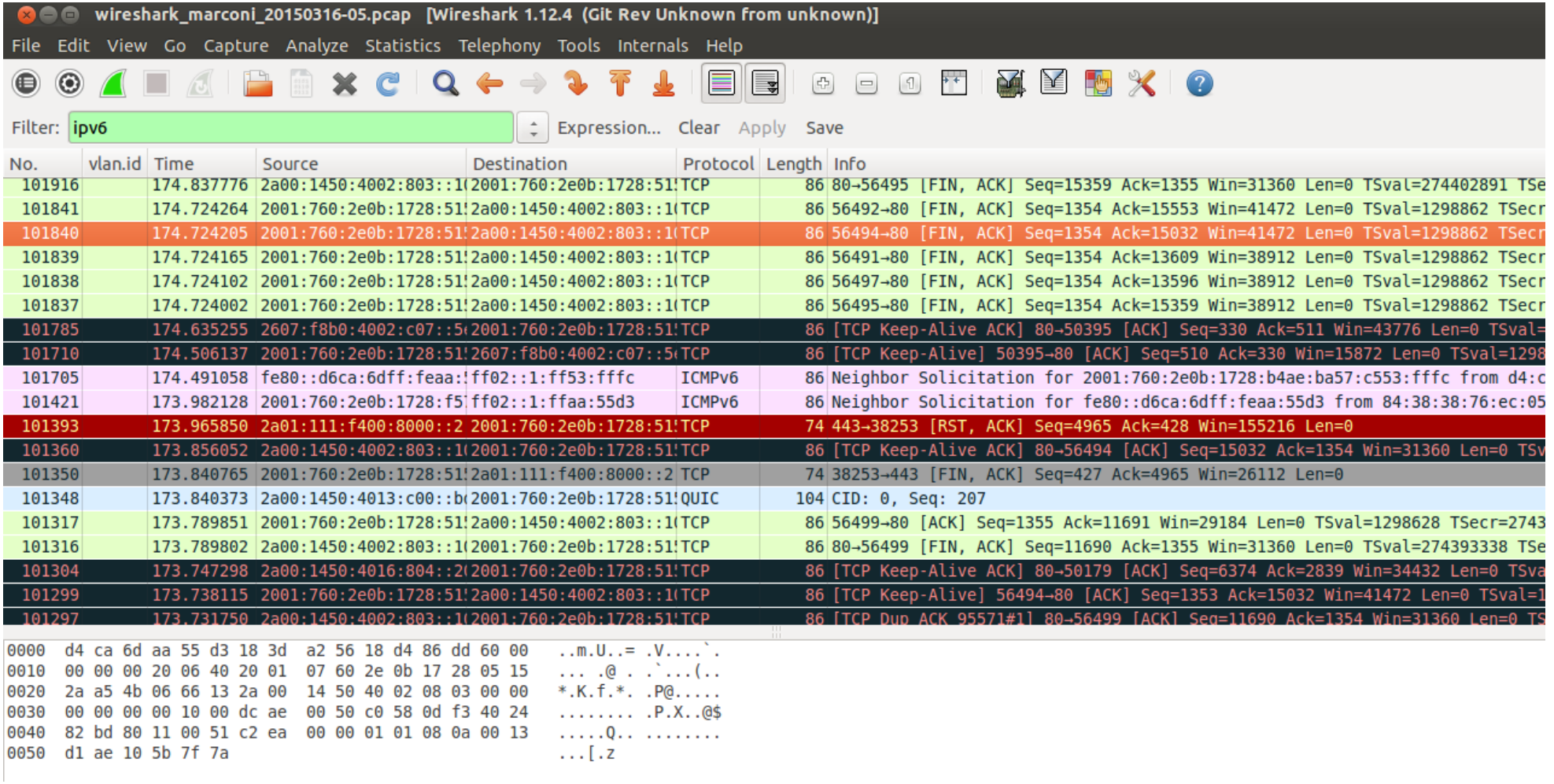
  (due to malware, virus)
- Troubleshooting 802.1x authentication problems, radius, WPA2
- Identifying network device failure
- Identifying impact of unwanted usage

  (e.g. Dropbox LAN Sync, Bittorrent, video streams)

wireshark

”Wireshark is a **free and open-source packet analyzer.** It is used for **network troubleshooting, analysis, software and communications protocol development, and education**. Originally named Ethereal, the project was renamed Wireshark in May 2006 due to trademark issues.” [wikipedia]

wireshark


wireshark_marconi_20150316-05.pcap [Wireshark 1.12.4 (Git Rev Unknown from unknown)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: ipv6
Expression... Clear Apply Save
No. vlan.id Time Source Destination Protocol Length Info
101916 174.837776 2a00:1450:4002:803::1(2001:760:2e0b:1728:51! TCP 86 80→56495 [FIN, ACK] Seq=15359 Ack=1355 Win=31360 Len=0 TSval=274402891 TSe
101841 174.724264 2001:760:2e0b:1728:51! 2a00:1450:4002:803::1(TCP 86 56492→80 [FIN, ACK] Seq=1354 Ack=15553 Win=41472 Len=0 TSval=1298862 TSecr
101840 174.724205 2001:760:2e0b:1728:51! 2a00:1450:4002:803::1(TCP 86 56494→80 [FIN, ACK] Seq=1354 Ack=15032 Win=41472 Len=0 TSval=1298862 TSecr
101839 174.724165 2001:760:2e0b:1728:51! 2a00:1450:4002:803::1(TCP 86 56491→80 [FIN, ACK] Seq=1354 Ack=13609 Win=38912 Len=0 TSval=1298862 TSecr
101838 174.724102 2001:760:2e0b:1728:51! 2a00:1450:4002:803::1(TCP 86 56497→80 [FIN, ACK] Seq=1354 Ack=13596 Win=38912 Len=0 TSval=1298862 TSecr
101837 174.724002 2001:760:2e0b:1728:51! 2a00:1450:4002:803::1(TCP 86 56495→80 [FIN, ACK] Seq=1354 Ack=15359 Win=38912 Len=0 TSval=1298862 TSecr
101785 174.635255 2607:f8b0:4002:c07::5(2001:760:2e0b:1728:51! TCP 86 [TCP Keep-Alive ACK] 80→50395 [ACK] Seq=330 Ack=511 Win=43776 Len=0 TSval=
101710 174.506137 2001:760:2e0b:1728:51! 2607:f8b0:4002:c07::5(TCP 86 [TCP Keep-Alive] 50395→80 [ACK] Seq=510 Ack=330 Win=15872 Len=0 TSval=1298
101705 174.491058 fe80::d6ca:6dff:feaa:! ff02::1:ff53:fffc ICMPv6 86 Neighbor Solicitation for 2001:760:2e0b:1728:b4ae:ba57:c553:fffc from d4:c
101421 173.982128 2001:760:2e0b:1728:f5! ff02::1:ffaa:55d3 ICMPv6 86 Neighbor Solicitation for fe80::d6ca:6dff:feaa:55d3 from 84:38:38:76:ec:05
101393 173.965850 2a01:111:f400:8000::2 2001:760:2e0b:1728:51! TCP 74 443→38253 [RST, ACK] Seq=4965 Ack=428 Win=155216 Len=0
101360 173.856052 2a00:1450:4002:803::1(2001:760:2e0b:1728:51! TCP 86 [TCP Keep-Alive ACK] 80→56494 [ACK] Seq=15032 Ack=1354 Win=31360 Len=0 TSv
101350 173.840765 2001:760:2e0b:1728:51! 2a01:111:f400:8000::2 TCP 74 38253→443 [FIN, ACK] Seq=427 Ack=4965 Win=26112 Len=0
101348 173.840373 2a00:1450:4013:c00::b(2001:760:2e0b:1728:51! QUIC 104 CID: 0, Seq: 207
101317 173.789851 2001:760:2e0b:1728:51! 2a00:1450:4002:803::1(TCP 86 56499→80 [ACK] Seq=1355 Ack=11691 Win=29184 Len=0 TSval=1298628 TSecr=2743
101316 173.789802 2a00:1450:4002:803::1(2001:760:2e0b:1728:51! TCP 86 80→56499 [FIN, ACK] Seq=11690 Ack=1355 Win=31360 Len=0 TSval=274393338 TSe
101304 173.747298 2a00:1450:4016:804::2(2001:760:2e0b:1728:51! TCP 86 [TCP Keep-Alive ACK] 80→50179 [ACK] Seq=6374 Ack=2839 Win=34432 Len=0 TSva
101299 173.738115 2001:760:2e0b:1728:51! 2a00:1450:4002:803::1(TCP 86 [TCP Keep-Alive] 56494→80 [ACK] Seq=1353 Ack=15032 Win=41472 Len=0 TSval=1
101297 173.731750 2a00:1450:4002:803::1(2001:760:2e0b:1728:51! TCP 86 [TCP Dup ACK 95571#1] 80→56499 [ACK] Seq=11690 Ack=1354 Win=31360 Len=0 TS
0000 d4 ca 6d aa 55 d3 18 3d a2 56 18 d4 86 dd 60 00 ..m.U..= .V....`.
0010 00 00 00 20 06 40 20 01 07 60 2e 0b 17 28 05 15 ... .@ . .`...(..
0020 2a a5 4b 06 66 13 2a 00 14 50 40 02 08 03 00 00 *.K.f.*. .P@.....
0030 00 00 00 00 10 00 dc ae 00 50 c0 58 0d f3 40 24 .... .P.X..@$
0040 82 bd 80 11 00 51 c2 ea 00 00 01 01 08 0a 00 13 .....Q.. ....
0050 d1 ae 10 5b 7f 7a ...[.z

Remote dumping & importing

Often, we would like to use wireshark GUI on dumps collected in remote places without GUI, e.g. via tcpdump:

```
$ tcpdump -i <interface> -s 65535 -w <some-file>
```

The output file can then be imported by wireshark.

wireshark and ...

- IPv6 – yes (filter: ipv6)
- 6Lowpan – yes (filter: 6lowpan)
- 802.15.4 – yes (filter: wpan)
- Bluetooth - ...

Bluetooth and wireshark

You can capture Bluetooth traffic to or from your machine on Linux in Wireshark with libpcap 0.9.6 and later, if the kernel includes the BlueZ Bluetooth stack;

starting with the 2.4.6 kernel, the BlueZ stack was incorporated into the mainline kernel.

Note that Debian and Debian-derived derivatives call the libpcap package "libpcap-0.8"; this does ***NOT*** mean that all such systems use libpcap 0.8.

Debian and its derivatives continue to use the name "libpcap-0.8",

even though newer versions' libpcap packages use newer versions of libpcap;

for example, Wheezy's libpcap-0.8 package uses libpcap 1.3.0.

If it's supported, and if you have sufficient privileges to capture,

there will be interfaces named bluetoothN for various values of N starting with 0.

To passively capture Bluetooth traffic between other machines, you can use the Ubertooth USB device. There is currently no libpcap support for Ubertooth, so you can't capture with Wireshark.

However, there is a plugin for Kismet - look for "Kismet" on the "Getting Started" Ubertooth page - and it produces capture files that can be dissected with a Wireshark plugin.

wireshark LAB:

- Installation
- Start
- Basic functionality
- Some exercises
- Your own use cases?

wireshark - installation

- **https://www.wireshark.org/download.html**
- **Linux: via repositories or build**

  on Ubuntu: `$sudo apt-get install wireshark`

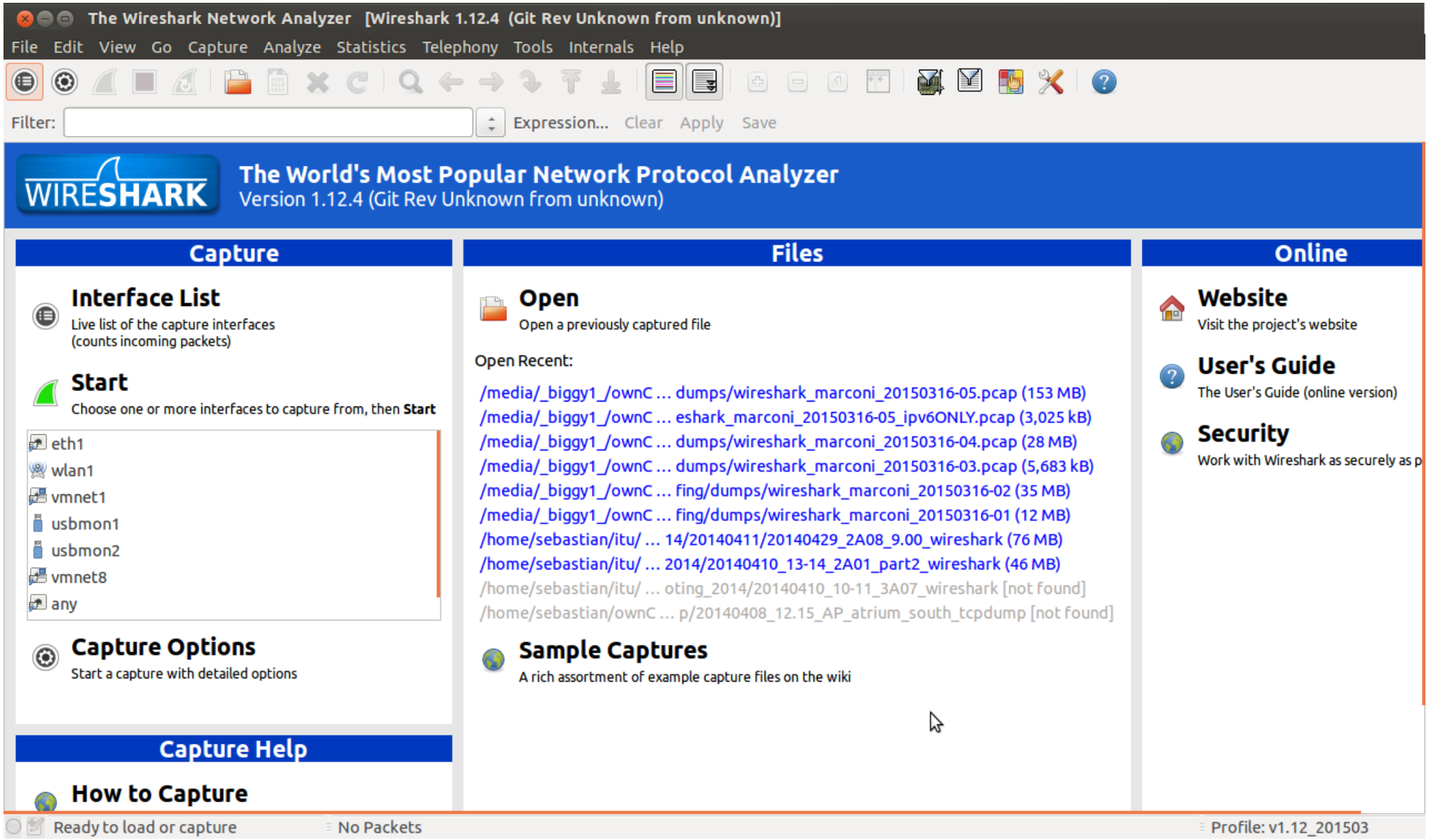
- **Windows: binary**
- **Mac: binary**

(note: needs X11, which is no longer part of OSX -

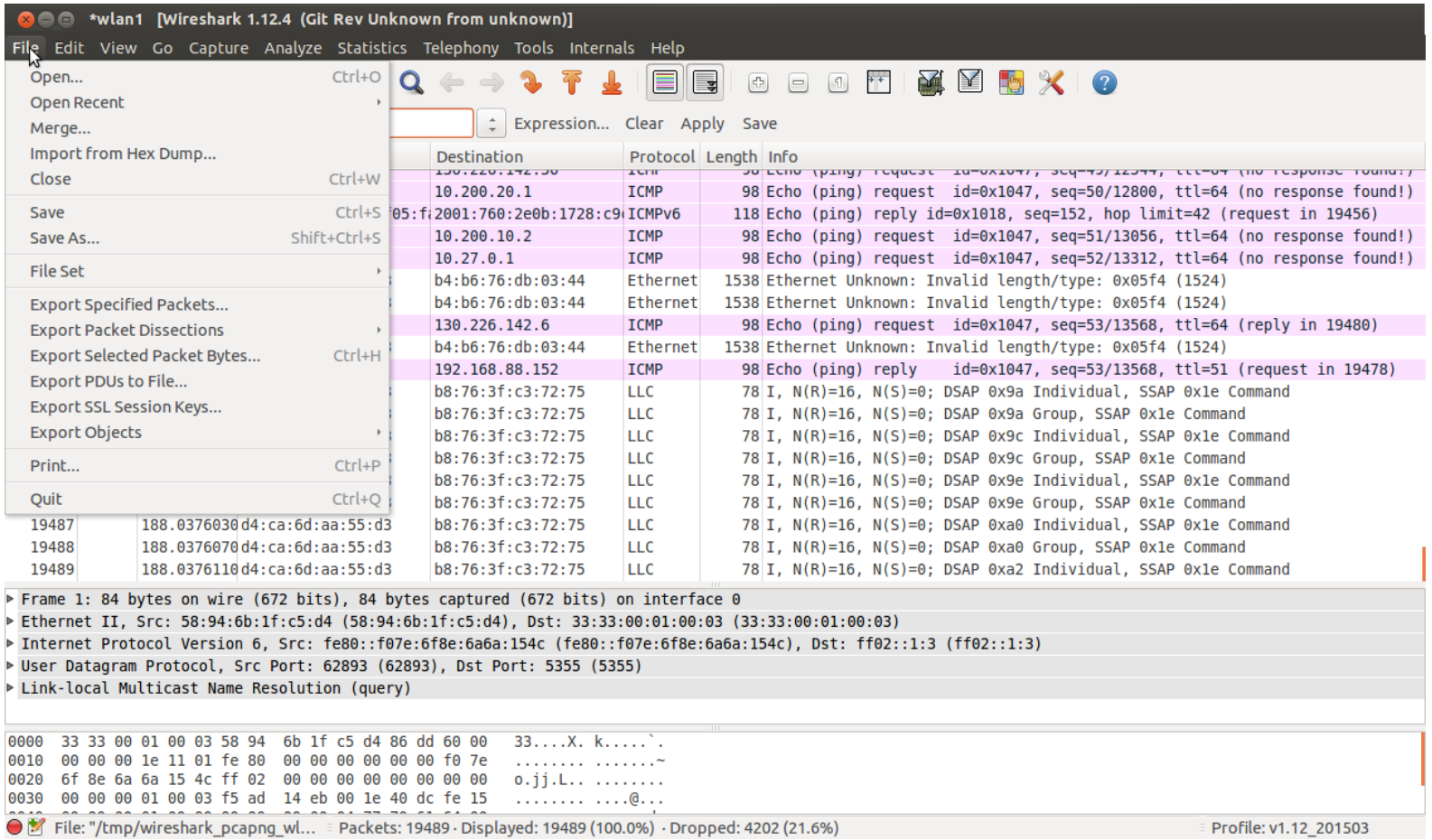
installation process will point you at XQuartz,

http://xquartz.macosforge.org)

wireshark - start

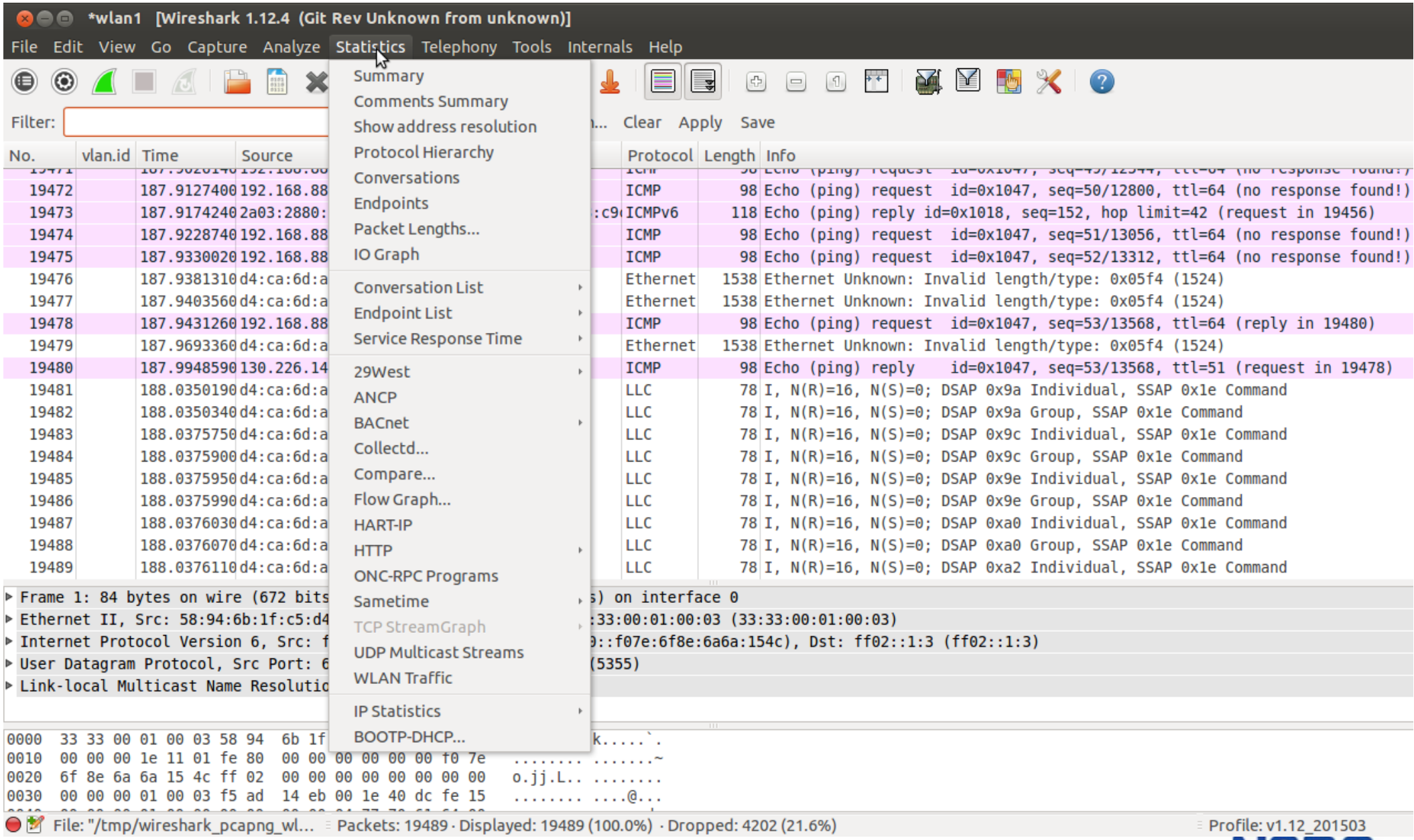


wireshark – file menu

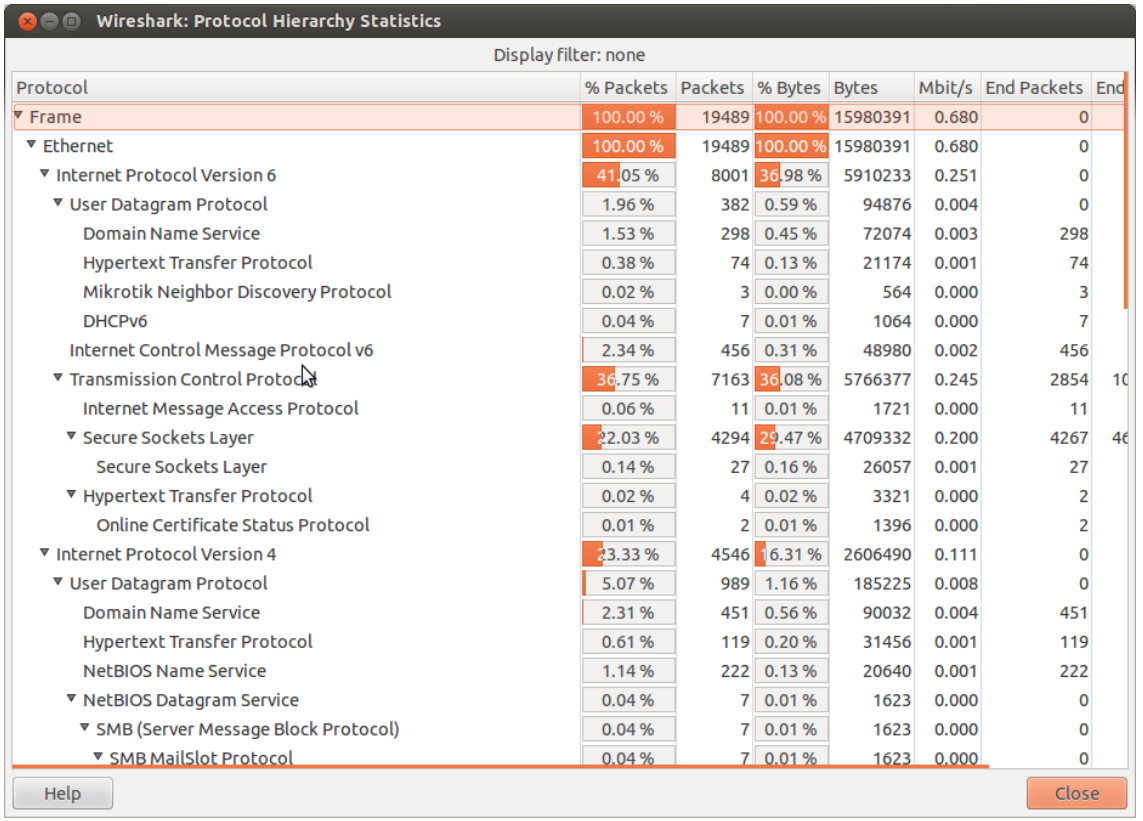


- **Exercise 2: save your capture to file, and then open it again**
- **Exercise 3: find a 6Lowpan capture on https://wiki.wireshark.org and open it**

wireshark – statistics



wireshark – statistics

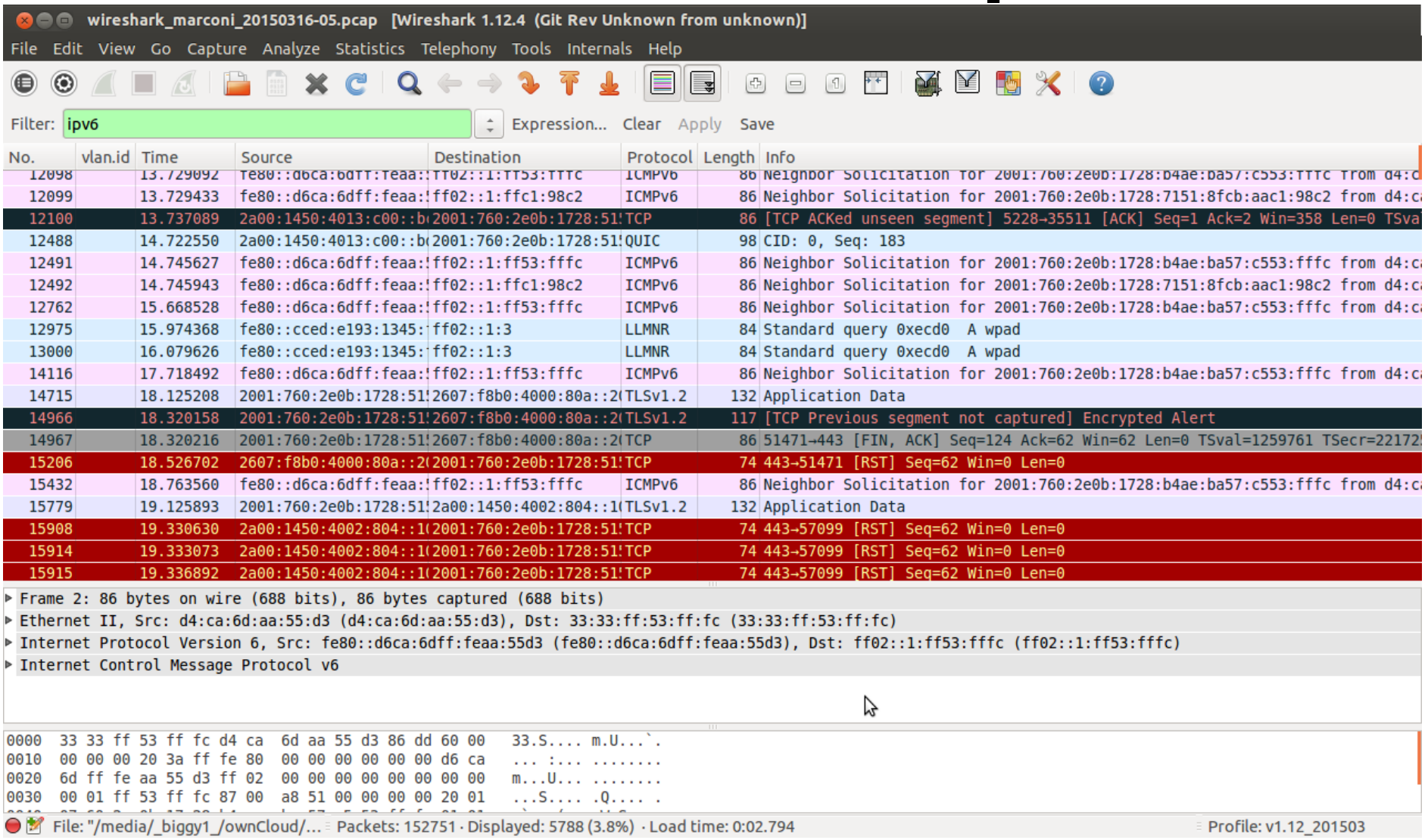


- **Exercise 4: do a new capture on your interface,**

go to menu > statistics > protocol hierarchy – what is the ratio between IPv4 and IPv6 on your interface?

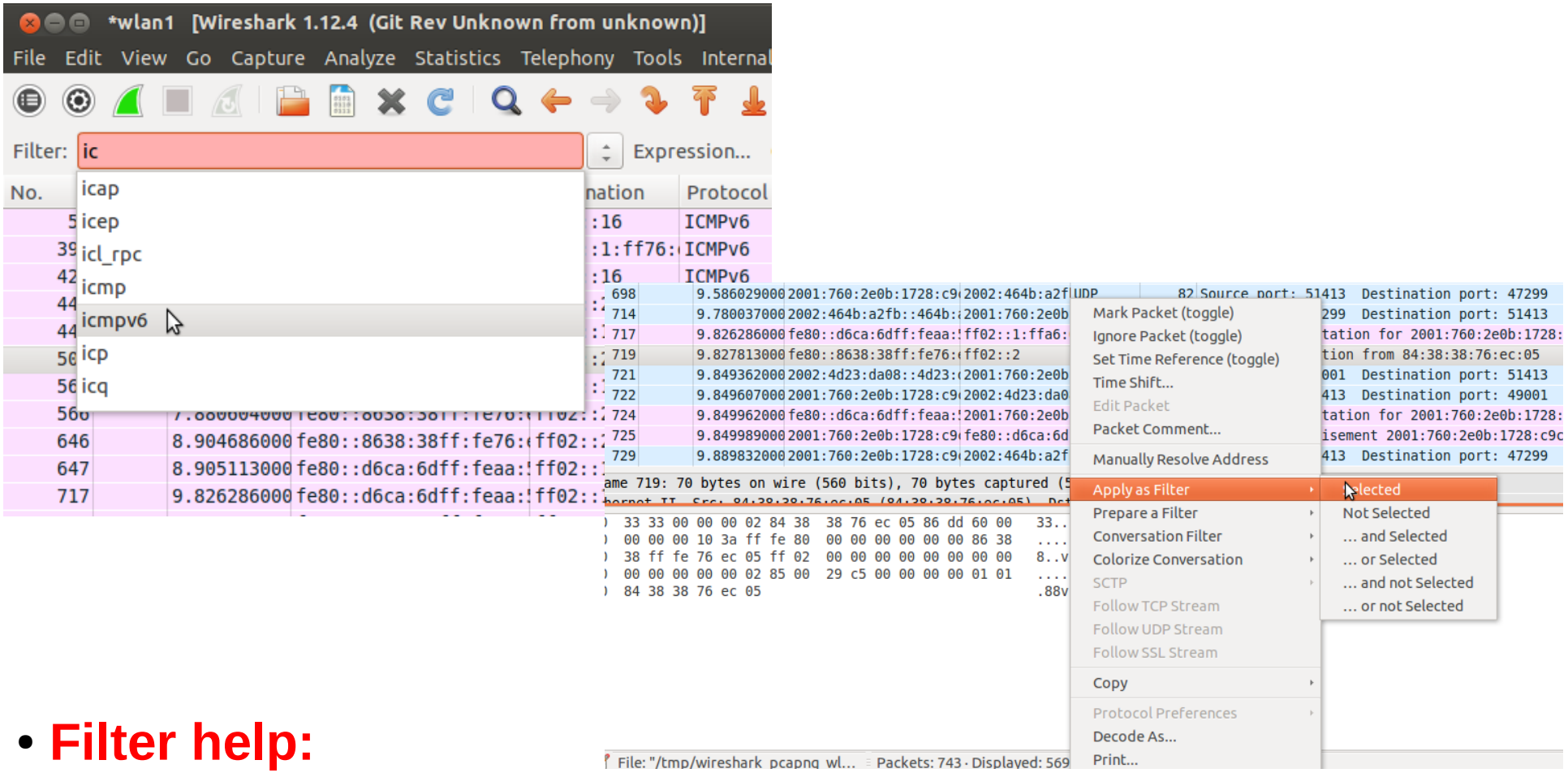
…. % IPv4 <==> …. % IPv6

wireshark – filters & expressions



- **Exercise 5: use Filter to show all IPv6 traffic**

wireshark – filters & expressions



- **Filter help:**

  there is autocompletion in the filter field,

  right-clicking a packet gives you context filters

wireshark – filters & expressions

- **Exercise 6:**

leave wireshark for a moment and find facebooks ipv6 address – if they have one? …..

- **ping6 or produce some other traffic to that address, capture and then find it in your wireshark capture**

- **Can you find any broadcast traffic in IPv6 packets? …..**

- **Move over to IPv4 – can you see broadcast traffic there? Look at your protocols – what protocols create broadcast traffic?**

wireshark – filters & expressions

- **Exercise 7:**

start a new dump and filter for ICMPv6 – what kinds of packets do you see?

……

……

- **Try to find a Router Advertisement – and look into it.**

What prefix is being advertised?

……

Who sent this Advertisement?

……

wireshark – additonal fun

Additonal exercises:

- **Explore statistics possibilities -**

what are your most active protocols, IPs, streams?

- **Do a tcpdump on a (remote) command line and analyze it with wireshark**

- **Consider if and how wireshark would be useful in**
  - **our sensor deployment?**
  - **your own work?**

Questions?

Thanks! **sebastian@nsrc.org**