

Module 3

Quality of Service QoS/Part 3

Prepared by Dr. Manal Mizher

Lecture 7

جامعة الجميع الذكية

تسعى لتكون الكيان الأول الذي يتبنى مبدأ المجتمع التعليمي
التعاوني ويقدم العلم مقابل العمل لتنمية وبناء المجتمعات.

Network programming

Course ID: CYB615



جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY

1. START WITH NETWORK ANALYSIS

(Port Scanner using Sockets, Network Traffic)

2. Wireshark

(Definition, installation, Display Filters, PCAP File (.pcap))

3. Scapy

(Definition, installation)

4. Npcap Npcap

(Definition, installation, Reading and writing .pcap, Packets[sending, receiving, sniffing], Packets with [ICMP, QoS, ToS, CoS])

What is Network Traffic?

Network Traffic or Network Data:

is the amount of data(packets) moving across a network at any given point of time.

$$\text{Traffic volume} = \text{Traffic Intensity} * \text{Time}$$

No.	Time	Source	Destination	Protocol	Length	Epoch Time	Info
1	0.000000	192.168.0.5	192.168.0.1	DNS	74	1499489239.102306000	Standard query 0x6d41 A www.goog...
2	0.001157	192.168.0.1	192.168.0.5	DNS	90	1499489239.103463000	Standard query response 0x6d41 A...

What is Network Traffic?

These are all the information which is sent over along with data packets on a network. It includes several information like:

Time: Time at which packets are captured.

Source: Source from which the packet originated.

Destination: The Sink where packets reach finally.

Protocol: Type of Internet protocol(or set of rules) the packet followed during its journey like: TCP, UDP, SMTP, ARP etc.

Info: The information that packet contains. Generally, the information is encrypted for obvious reasons. But in some cases its unencrypted also, but that is rare.



■ What is a Port Scanner? What is it used for?

A port scan is a common technique hackers use to discover open doors or weak points in a network. A port scan attack helps cyber criminals find open ports and figure out whether they are receiving or sending data. It can also reveal whether active security devices like firewalls are being used by an organization.

- The fundamental step to enter a system is to scan the system for opening ports.
- It is always advised to close unnecessary ports of your system(server) to avoid any mishappening.
- For a **web server port 80** is open by default for HTTP request and response.

Port Scanner using Sockets

```
import socket

t_host = str(input("Enter the host to be scanned: ")) # Target Host, www.example.com
t_ip = socket.gethostbyname(t_host) # Resolve t_host to IPv4 address

print(t_ip) # Print the IP address

while 1:
    t_port = int(input("Enter the port: ")) # Enter the port to be scanned

    try:
        sock = socket.socket()
        res = sock.connect((t_ip, t_port))
        print("Port {}: Open".format(t_port))
        sock.close()
    except:
        print("Port {}: Closed".format(t_port))

print("Port Scanning complete")
```

Module 3

Captured Network Traffic using Wireshark

No.	Time	Source	Destination	Protocol	Length	Epoch Time	Info
1	0.000000000	172.16.172.22	172.16.173.255	UDP	305	1502398036.004486480	54915 → 54915 Len=263
2	0.075215081	172.16.172.22	239.255.255.250	SSDP	179	1502398036.079701561	M-SEARCH * HTTP/1.1
3	0.102547747	172.16.172.46	255.255.255.255	UDP	139	1502398036.107034227	54547 → 10505 Len=97
4	0.254055840	HewlettP_0e:e1:0e	Broadcast	ARP	60	1502398036.258542320	Who has 172.16.173.253? Te...
5	0.713768978	HewlettP_02:28:f9	Broadcast	ARP	60	1502398036.718255458	Who has 172.16.173.170? Te...
6	0.856452626	fe80::cca3:c363:774...	ff02::1:2	DHCPv6	155	1502398036.860939106	Solicit XID: 0x1ae545 CID:...
7	1.000982946	172.16.172.22	172.16.173.255	UDP	305	1502398037.005469426	54915 → 54915 Len=263
8	1.147574073	fe80::15d3:5371:eed...	ff02::c	SSDP	208	1502398037.152060553	M-SEARCH * HTTP/1.1
9	1.198315233	172.16.173.219	239.255.255.250	SSDP	319	1502398037.202801713	NOTIFY * HTTP/1.1
10	1.329985538	HewlettP_02:28:f9	Broadcast	ARP	60	1502398037.334472018	Who has 172.16.173.170? Te...
11	2.000733124	172.16.172.22	172.16.173.255	UDP	305	1502398038.005219604	54915 → 54915 Len=263
12	2.105650027	172.16.172.46	255.255.255.255	UDP	139	1502398038.110136507	54549 → 10505 Len=97
13	2.329588700	HewlettP_02:28:f9	Broadcast	ARP	60	1502398038.334075180	Who has 172.16.173.170? Te...
14	2.542134373	172.16.172.11	239.255.255.250	SSDP	179	1502398038.546620853	M-SEARCH * HTTP/1.1
15	3.000418629	172.16.172.22	172.16.173.255	UDP	305	1502398039.004905109	54915 → 54915 Len=263
16	3.075410707	172.16.172.22	239.255.255.250	SSDP	179	1502398039.079897187	M-SEARCH * HTTP/1.1
17	3.207700365	172.16.172.60	172.16.173.255	BROWS...	243	1502398039.212186845	Host Announcement DESKTOP-...
18	3.835443041	fe80::2bbb:eaaa:813...	ff02::fb	MDNS	203	1502398039.839929521	Standard query 0x0000 PTR ...
19	3.950286276	172.16.172.22	224.0.0.251	MDNS	106	1502398039.954772756	Standard query 0x0000 PTR ...
20	3.951062841	BestItWo_00:14:dd	Broadcast	ARP	60	1502398039.955549321	Who has 224.0.0.251? Tell ...
21	4.000221327	172.16.172.22	172.16.173.255	UDP	305	1502398040.004707807	54915 → 54915 Len=263
22	4.111433514	172.16.172.46	255.255.255.255	UDP	139	1502398040.115919994	54551 → 10505 Len=97
23	4.151186893	fe80::15d3:5371:eed...	ff02::c	SSDP	208	1502398040.155673373	M-SEARCH * HTTP/1.1
24	4.340437216	fe80::d5b4:d6bd:41b...	ff02::16	ICMPv6	90	1502398040.344923696	Multicast Listener Report ...
25	4.386055797	fe80::d5b4:d6bd:41b...	ff02::16	ICMPv6	90	1502398040.390542277	Multicast Listener Report ...
26	4.386386844	172.16.172.112	224.0.0.252	IGMPv2	60	1502398040.390873324	Membership Report group 22...





- Wireshark is a network packet analyzer. A network packet analyzer presents captured packet data in as much detail as possible.
 - Measuring device for examining what's happening inside a network cable, just like an electrician uses a voltmeter for examining what's happening inside an electric cable (but at a higher level, of course).
 - Wireshark is available for free, is open source, and is one of the best packet analyzers available today.

<https://www.wireshark.org/>

https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html

Some reasons for using Wireshark:

- 1) Network administrators use it to troubleshoot network problems.
- 2) Network security engineers use it to examine security problems.
- 3) Quality assurance (QA) engineers use it to verify network applications.
- 4) Developers use it to debug protocol implementations.
- 5) People use it to learn network protocol internals.

Features provided by Wireshark:

- 1) Available for UNIX and Windows.
- 2) Capture live packet data from a network interface.
- 3) Open files containing packet data captured with tcpdump/WinDump, Wireshark, and many other packet capture programs.
- 4) Import packets from text files containing hex dumps of packet data.
- 5) Display packets with very detailed protocol information.
- 6) Save packet data captured.

Features:

- 7) Export some or all packets in a number of capture file formats.
- 8) Filter packets on many criteria.
- 9) Search for packets on many criteria.
- 10) Colorize packet display based on filters.
- 11) Create various statistics.
- 12) ...and a lot more!

Some things Wireshark does not provide :

- 1) Wireshark isn't an intrusion detection system. It will not warn you when someone does strange things on your network that he/she isn't allowed to do. However, if strange things happen, Wireshark might help you figure out what is really going on.
- 2) Wireshark will not manipulate things on the network, it will only "measure" things from it. Wireshark doesn't send packets on the network or do other active things (except domain name resolution, but that can be disabled).

The Wireshark Display Filter

- Wireshark's display filter a bar located right above the column display section. This is where you type expressions to filter the frames, IP packets, or TCP segments that Wireshark displays from a pcap.
- Wireshark's display filter uses Boolean expressions, so you can specify values and chain them together. The following expressions are commonly used:

Equals: `==` or ***eq***

And: `&&` or ***and***

Or: `//` (double pipe) or ***or***

The Wireshark Display Filter

- **Examples of these filter expressions follow:**

```
ip.dst eq 192.168.10.195 and ip.src == 192.168.10.1
```

```
http.request && ip.dst == 192.168.10.195
```

```
http.request || http.response
```

```
dns.qry.name contains microsoft or dns.qry.name contains windows
```

- **What is pcap file? How to open pcap files?**

PCAP file extension - Captured packets

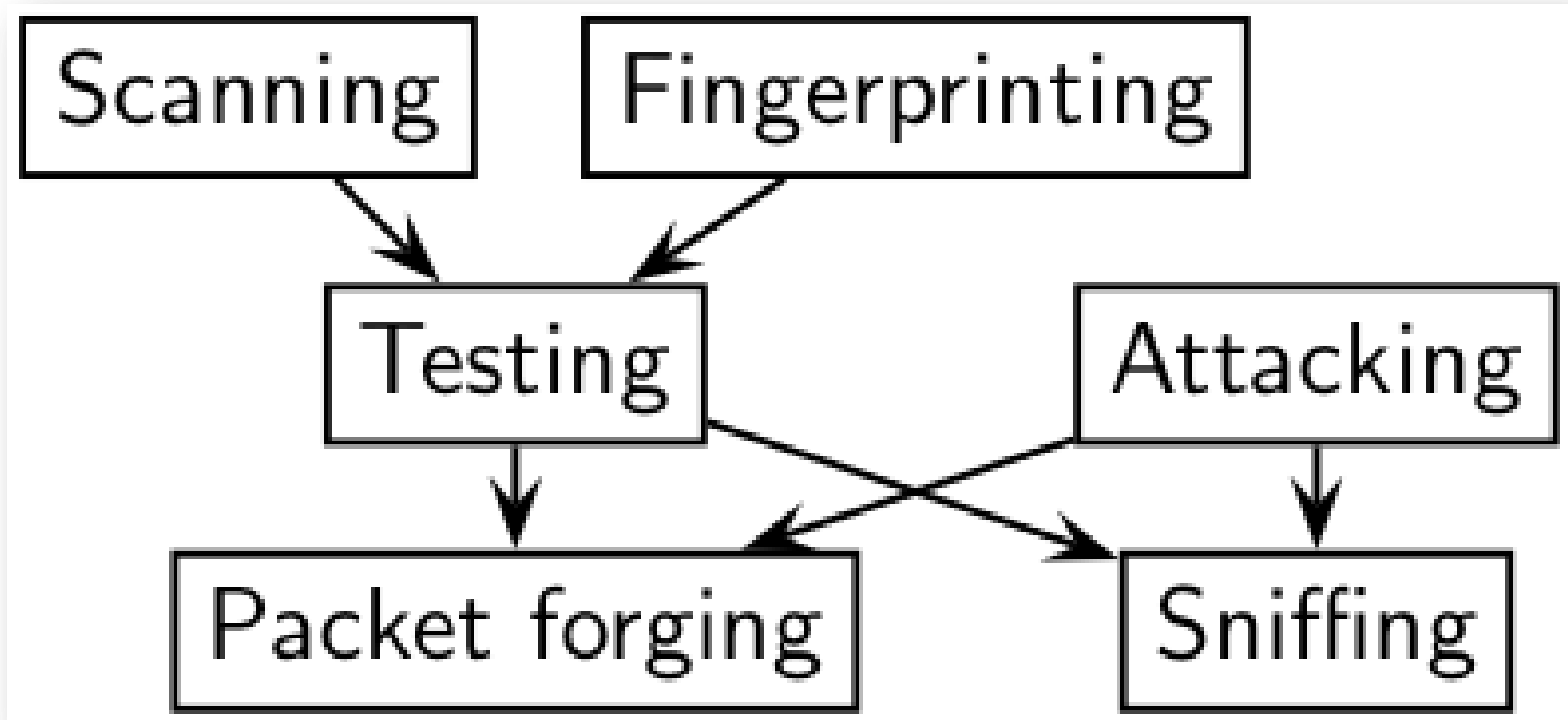
- The pcap file extension is associated with Wireshark.
- Wireshark pcap files contain captured data sniffed from network packets. PCAP data file format is de facto standard file format used by other network analyzers.
- Packet capture data can be checked and viewed by applications, which use libpcap or **Npcap** libraries.
- The pcap files are used to save the captured packets for later use. We can read packets from a pcap file and write them to a pcap file using **Scapy**.

- Scapy is a Python program that enables the user to send, sniff and dissect and forge network packets. This capability allows construction of tools that can probe, scan or attack networks.
- Scapy is a powerful interactive packet manipulation program:
 - Forge, decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies, and much more.
 - Scanning, tracerouting, probing, unit tests, attacks or network discovery.
- Scapy also performs very well on a lot of other specific tasks that most other tools can't handle, like sending invalid frames, injecting your own 802.11 frames, combining techniques (VLAN hopping+ARP cache poisoning, VOIP decoding on WEP encrypted channel, ...), etc.

<https://scapy.readthedocs.io/en/latest/index.html>

- Scapy is a library made in Python, with its own command line interpreter (CLI), which allows to create, modify, send and capture network packets.
- It can be used interactively through the command line interface or as a library by importing it into Python programs. It can also run on Linux, Mac OS X and Windows systems.

<https://santandercto.com/en/guide-using-scapy-with-python/>



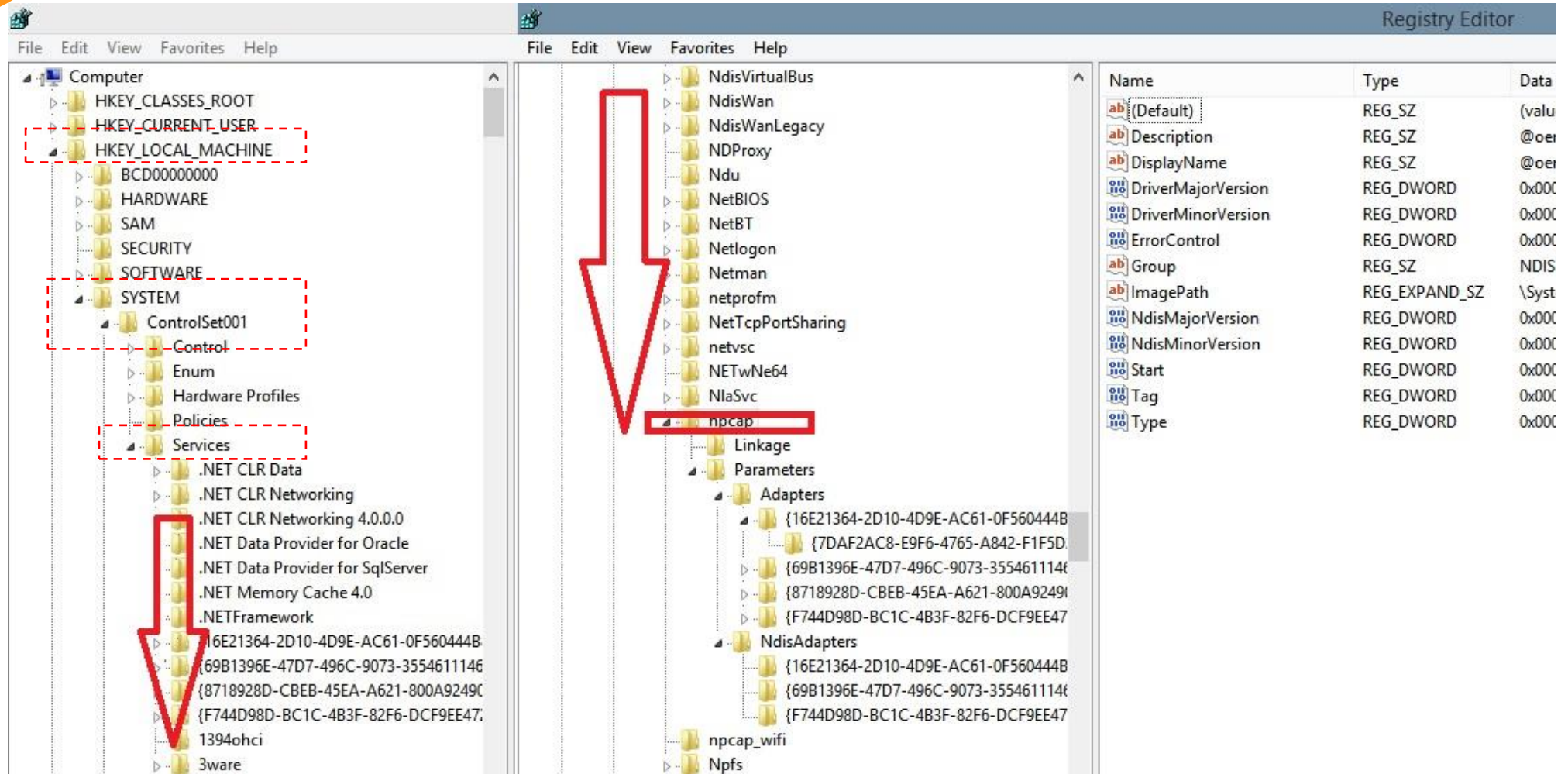
Packet capture library for Windows (*Nmap Packet Capture*)

- Npcap is the Nmap Project's packet capture (and sending) library for Microsoft Windows.
- This allows Windows software to capture raw network traffic (including wireless networks, wired ethernet, localhost traffic, and many VPNs) using a simple, portable API.
- Npcap allows for sending raw packets as well.
- Mac and Linux systems already include the Pcap API, so Npcap allows popular software such as [Nmap](#) and [Wireshark](#) to run on all these platforms (and more) with a single codebase.
- Wireshark uses this library to capture live network data on Windows.

- check the installation options of an already installed Npcap by reading the registry key:

Cmd → regedit →

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\npcap\Parameters





- You need the following software in order to install Scapy on Windows:
 - Python: Python 2.7.X or 3.4+. After installation, add the Python installation directory and its Scripts subdirectory to your PATH.
 - Install scapy library by: `python -m pip install scapy`
 - Wireshark
 - Npcap
- <https://www.systemconf.com/2021/05/06/how-to-install-wireshark/>
- Open a command prompt (cmd.exe) and run Scapy by typing scapy
- Or, open terminal in PyCharm and run Scapy by typing scapy



```
from scapy.all import *  
  
packets = rdpcap("2.pcap")  
packets.summary()  
packets.show()  
  
for packet in packets:  
    if packet.haslayer(UDP):  
        print('It is UDP')
```



```
from scapy.all import *

capture = input("Enter file path of pcap file: ")
pcap = rdpcap(capture)
ports = 53

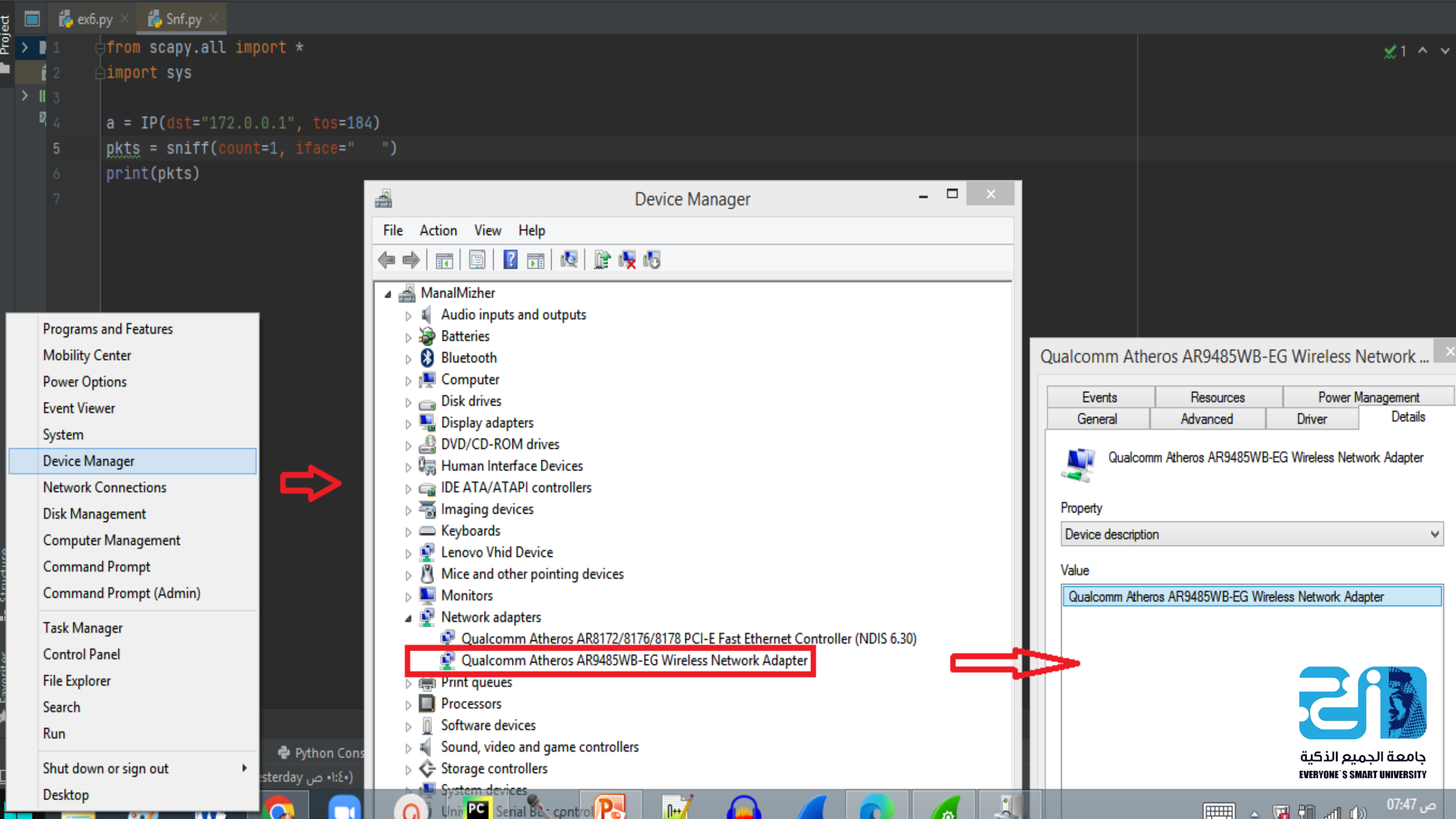
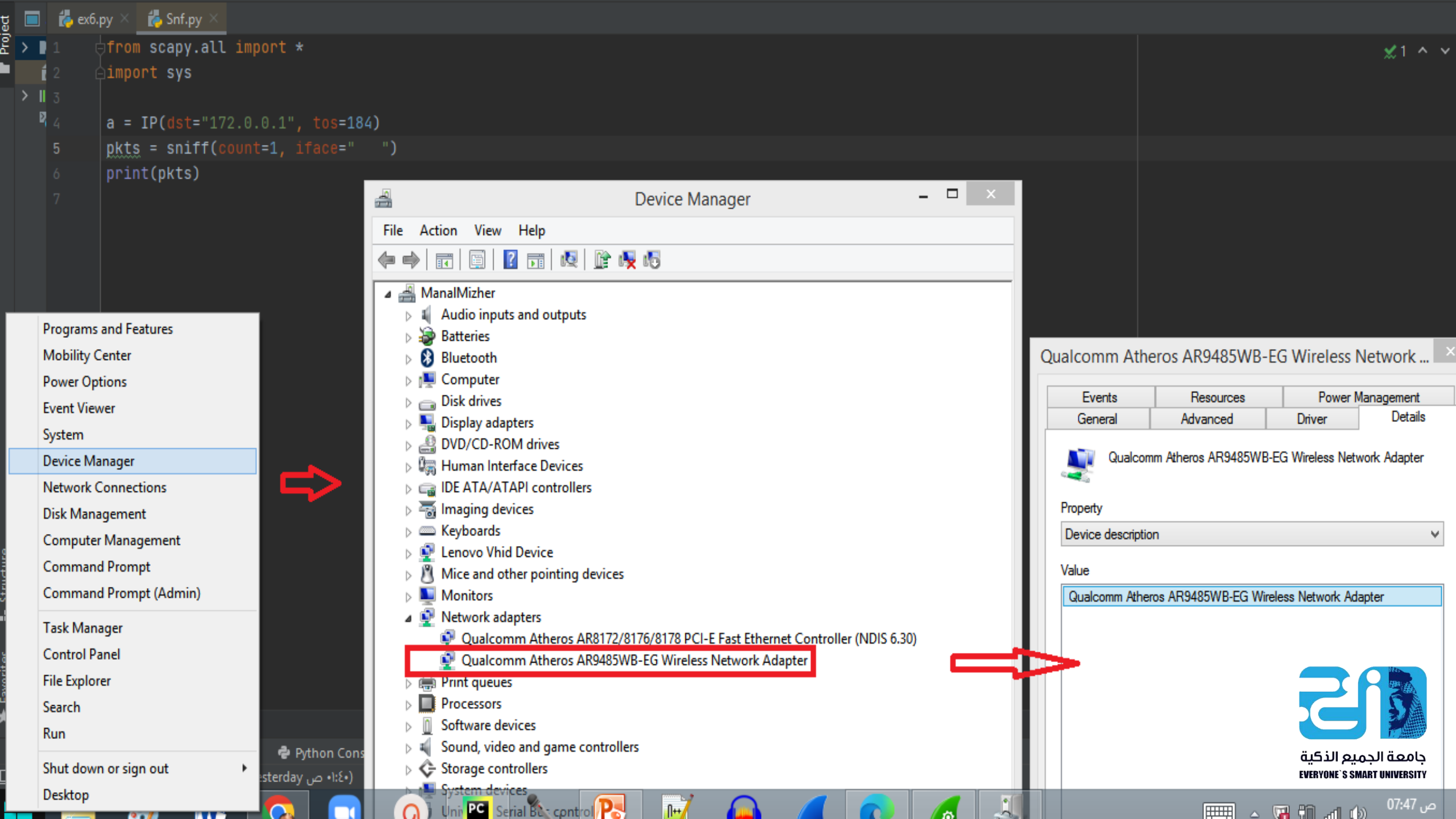
def write(pkt):
    wrpcap('filtered.pcap', pkt, append=True) # appends packet to output file

for pkt in pcap:
    if pkt.haslayer(UDP) and pkt.getlayer(UDP).sport == ports: # checks for UDP layer
        write(pkt) # sends the packet to be written if it meets criteria
        print('pkt == ', pkt)
    else:
        pass
```



- Capture up to 10 packets (or stop with ctrl-c)

```
from scapy.all import *  
import sys  
  
pkts = sniff(count=10, iface="Qualcomm Atheros AR9485WB-EG Wireless Network Adapter")  
print(pkts)  
pkts.show()
```





- Send one or more packets at layer three
`send(pkt, inter=0, loop=0, count=1, iface=N)`
- Send one or more packets at layer two
`sendp(pkt, inter=0, loop=0, count=1, iface=N)`
- Send packets much faster at layer two using tcpreplay
`sendpfast(pkt, pps=N, mbps=N, loop=0, iface=N)`

Note:

In Scapy, you have two ways of sending packets. The "layer 3 way", and the "layer 2 way".



```
from scapy.all import *  
import sys  
  
NetInf = "Qualcomm Atheros AR9485WB-EG Wireless Network Adapter"  
pkts = sniff(count=3, iface= NetInf)  
  
send(pkts, inter=0, loop=0, count=1, iface=NetInf)  
send(IP(dst="192.0.2.1")/UDP(dport=53))  
  
sendp(pkts, inter=0, loop=0, count=2, iface=NetInf)  
sendp(Ether()/IP(dst="192.0.2.1")/UDP(dport=53))
```





- Send packets and receive replies

`sr(pkt, iface=N)`

`srp(pkt, iface=N)`

- Send packets and return only the first reply

`sr1(pkt, inter=0, loop=0, count=1, iface=N), srp1(...)`

- Send packets in a loop and print each reply

`srloop(pkt, timeout=N, count=N), srploop(...)`





```
from scapy.all import *  
import sys  
  
a, b = sr(IP(dst="www.google.com") / ICMP())  
a.show()  
print(a)
```

Note :

/ is the encapsulation operator. That means ICMP encapsulated in IP.





- Scapy allows the setting of DSCP or ToS bits in IP packets, using the upper 6 bits of the 8 bit Type of Service field.

```
from scapy.all import *
import sys

a = IP(dst="172.0.0.1", tos=184)
print('a = ', a)
a.show()

b = IP(dst="172.0.0.1", ttl=(4, 25), id=100, tos=184)/TCP(flags=0x2)
print('b = ', b)
b.show()

print(" b.dst= ", b.dst)
print(" b.ttl= ", b.ttl)
print(" b.tos= ", b.tos)
```



- 1) Generate packets with various QoS markings with scapy.
- 2) Mimic the markings typical to VoIP phones and verify that the markings remain intact on the other end using Wireshark.
- 3) There are two QoS values we need to test:
 - IEEE 802.1Q priority marking (class of service, or CoS)
 - IP differentiated services control point (DSCP) field.

(Both of these have a default value of zero).



Notes:

- 1) An Ethernet frame can only be marked at layer two if it has an 802.1Q header.
- 2) Packets traveling out an access port or inside the untagged native VLAN on a trunk don't receive an 802.1Q header and thus have no priority field, and must rely on layer three QoS markings.

```
from scapy.all import *  
import sys
```

```
a = Dot1Q().display()  
print(a)
```

```
IP().display()
```



Notes:

- 3) An Ethernet frame can only be marked at layer two if it has an 802.1Q header.
- 4) Packets traveling out an access port or inside the untagged native VLAN on a trunk don't receive an 802.1Q header and thus have no priority field, and must rely on layer three QoS markings.

```
from scapy.all import *  
import sys
```

```
a = Dot1Q().display()  
print(a)
```

```
IP().display()
```



Notes:

- 5) build the packet with an Ethernet header with a specified destination MAC address and an 802.1Q header with a VLAN ID and priority (CoS) of five.

```
l2 = Ether(dst="00:23:7d:00:d0:a8")/Dot1Q(vlan=10, prio=5)
```



Notes:

- 6) DSCP values usually are referring to only the first six bits of an eight-bit field. This allows for a decimal value range of 0 through 63. A complete list of DSCP values is shown on the QoS cheat sheet (<https://packetlife.net/media/library/19/QoS.pdf>).
- 7) Scapy requires to provide an eight-bit value when setting the field in the packet.
- 8) How do we convert our decimal DSCP value of 46 (EF) to an eight-bit value?

EX: find 46 in binary, add two trailing zeros, and convert it back to decimal.

46 => 101110

101110**00** => 184

<https://www.rapidtables.com/convert/number/binary-to-decimal.html>



Notes:

- 9) Set the packet's DSCP field equal to 184 decimal. Scapy displays the DSCP field as a hexadecimal value.

```
l3 = IP(dst="192.168.0.1", src="192.168.0.2", tos=184)
print("Layer 3 = ", l3)
```

- 10) Top off the packet with an empty UDP payload and see what it looks like glued together.

```
l4 = UDP()
print("Layer 4 = ", l4)
b = l2/l3/l4
print("The three Layers = ", b)
```



Notes:

- 9) Set the packet's DSCP field equal to 184 decimal. Scapy displays the DSCP field as a hexadecimal value.

```
l3 = IP(dst="192.168.0.1", src="192.168.0.2", tos=184)
print("Layer 3 = ", l3)
```

- 10) Top off the packet with an empty UDP payload and see what it looks like glued together.

```
l4 = UDP()
print("Layer 4 = ", l4)
b = l2/l3/l4
print("The three Layers = ", b)
```



Notes:

11) Send the packet out on the wire.

- Remember that since we're working at layer two, we need to use the `sendp()` function, not `send()`, to transmit our packet.
- Scapy requires root or administrator privileges to send raw packets.

```
sendp(l2/l3/l4, iface="Qualcomm Atheros AR9485WB-EG Wireless Network Adapter")
```

12) Use Wireshark to verify that the packet has been marked with the appropriate CoS and DSCP values at the receiving end.





```
from scapy.all import *
import sys

a = Dot1Q().display()
print(a)
IP().display()

l2 = Ether(dst="09:10:6c:0b:a0:a0")/Dot1Q(vlan=10, prio=5)
l3 = IP(dst="192.168.0.1", src="192.168.0.2", tos=128)
print("Layer 3 = ", l3)
l2.show()
l3.show()

l4 = UDP()
print("Layer 4 = ", l4)
l4.show()

b = l2/l3/l4
print("The three Layers = ", b)
sendp(l2/l3/l4, iface="Qualcomm Atheros AR9485WB-EG Wireless Network Adapter")
```

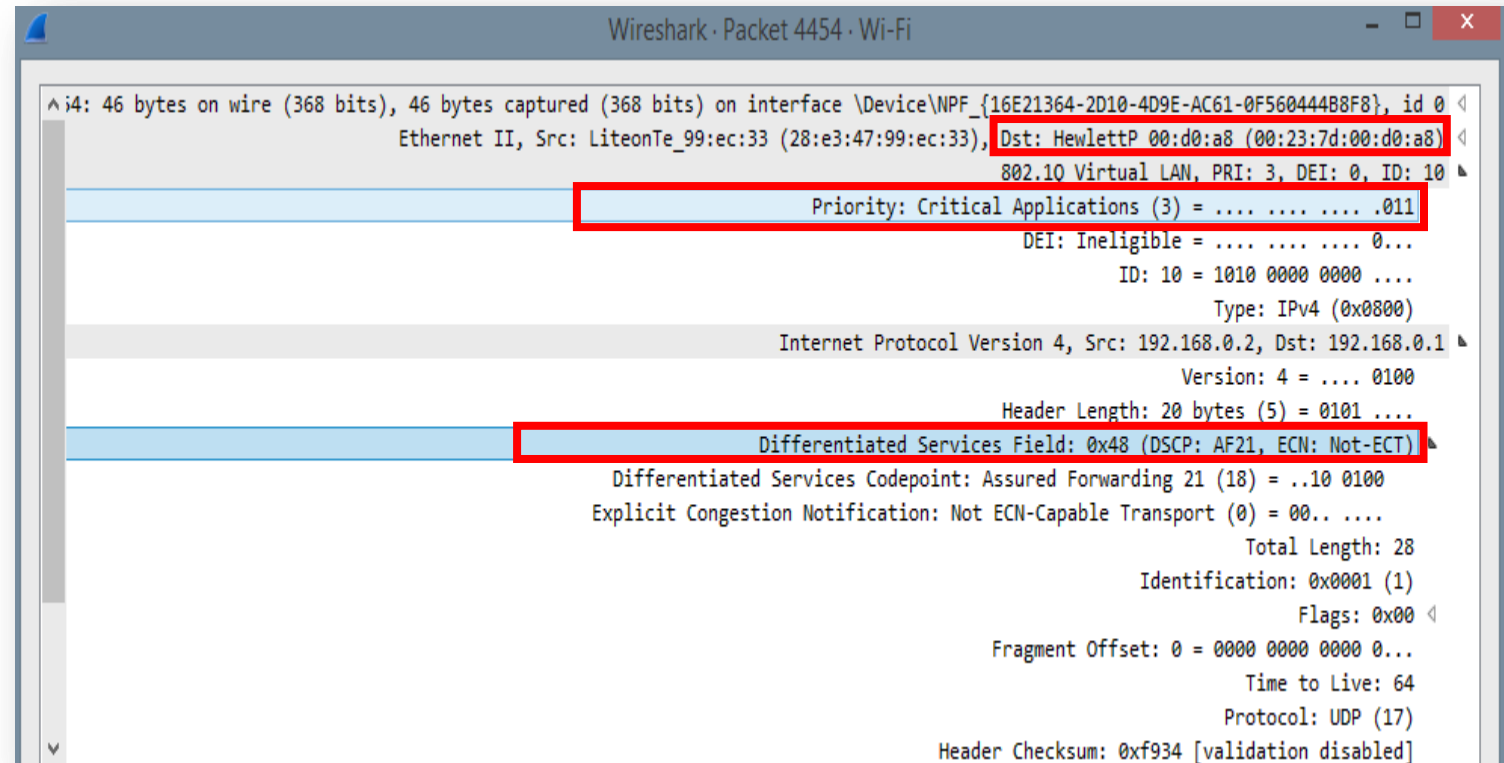


```
Wireshark · Packet 3595 · Wi-Fi
^ 15: 46 bytes on wire (368 bits), 46 bytes captured (368 bits) on interface \Device\NPF_{16E21364-2D10-4D9E-AC61-0F560444B8F8}, id 0
    Ethernet II, Src: LiteonTe_99:ec:33 (28:e3:47:99:ec:33), Dst: 09:10:6c:0b:a0:a0 (09:10:6c:0b:a0:a0)
        802.1Q Virtual LAN, PRI: 5, DEI: 0, ID: 10
            Priority: Voice, < 10ms latency and jitter (5) = .... .101
                DEI: Ineligible = .... .0...
                    ID: 10 = 1010 0000 0000 ....
                        Type: IPv4 (0x0800)
                            Internet Protocol Version 4, Src: 192.168.0.2, Dst: 192.168.0.1
                                Version: 4 = .... 0100
                                    Header Length: 20 bytes (5) = 0101 ....
                                        Differentiated Services Field: 0x80 (DSCP: CS4, ECN: Not-ECT)
                                            Total Length: 28
                                                Identification: 0x0001 (1)
                                                    Flags: 0x00
                                                        Fragment Offset: 0 = 0000 0000 0000 0...
```

Q1: Build the packet with an Ethernet header with:

- 1) Destination MAC address = **00:23:7d:00:d0:a8**
- 2) 802.1Q header
- 3) VLAN ID: **20**
- 4) Priority (CoS): **3**
- 5) DSCP: *PHB Class: AF21*

Q2: Screenshot your Wireshark using the filter as in the figure.



```
from scapy.all import *

Ether().show()

IP().display()
l3 = IP(dst="www.python.org", tos=104) / ICMP()
print("Layer 3 = ", l3)
l3.show()
send(l3)

def write(pkt):
    wrpcap('ICMPwriteOnly.pcap', pkt, append=True) # appends packet to output file

for pkt in l3:
    write(pkt) # sends the packet
    print('pkt == ', pkt)
```

Apply a display filter ... <Ctrl-/>									
DSCP	Info	Length	Protocol	Destination	Source	Time	.No		
Assured Forwarding 31	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)	28	ICMP	199.232.80.223	192.168.1.19	0.000000	1		

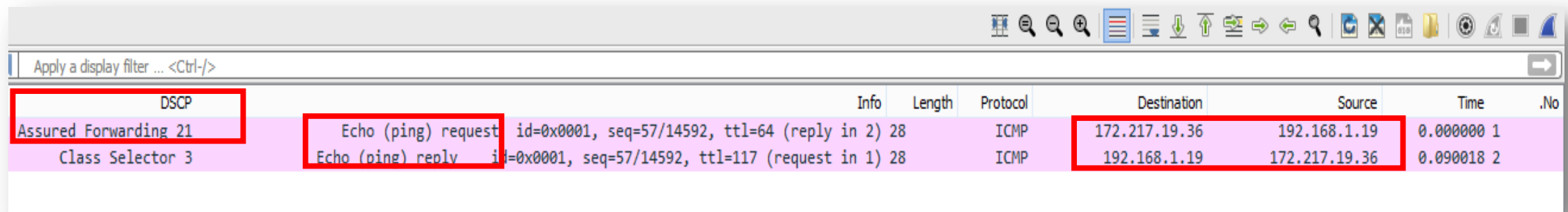
Q2 (a) : Use Python Scapy module to generate an ICMP Request with:

- ToS value 72
- ICMP(id=1, length=256, seq=57)
- Destination : www.google.com

(b) : send and receive ICMP request and reply by using sr() command.

(c) : write the returned value from sr() command to .pcap file.

(d) : open your .pcap file in wireshark and screenshot the wireshark window as the Figure.



	Info	Length	Protocol	Destination	Source	Time	.No
Echo (ping) request	id=0x0001, seq=57/14592, ttl=64 (reply in 2)	28	ICMP	172.217.19.36	192.168.1.19	0.000000	1
Echo (ping) reply	id=0x0001, seq=57/14592, ttl=117 (request in 1)	28	ICMP	192.168.1.19	172.217.19.36	0.090018	2

- 1) <https://networklessons.com/quality-of-service/how-to-configure-queuing-on-cisco-catalyst-3560-and-3750-switch>
- 2) https://www.youtube.com/watch?v=VbqIRBgemnM&ab_channel=KevinWallaceTraining%2CLLC
- 3) https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus1000/sw/4_0_4_s_v_1_3/qos/configuration/guide/n1000v_qos.pdf
- 4) https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus1000/sw/4_0_4_s_v_1_3/qos/configuration/guide/n1000v_qos/n1000v_qos_2classification.html
- 5) <http://what-when-how.com/ccnp-ont-exam-certification-guide/qos-implementation-methods-ip-quality-of-service/>

