19.1. What is the network prefix for each of the following hosts: (1) 10.1.2.200/24 and (2) 10.1.2.200/26? (Pages 380~381)

1. 10.1.2.0/24
2. 10.1.2.192/26

19.2 When we send a packet, (1) How is the source port number of the packet decided? (2) How is the source IP of the packet decided?

1. The OS automatically selects the src port unless specified by the application.
2. The source IP of your machine is automatically used to decide the packet destination.

19.3 When we construct a packet, we need to set the address at each layer. What are the addresses called at the following layers: (1) transport layer, (2) network layer, and (3) data link layer?

1. Transport layer: Port number (52345)
2. Network layer: IP address (192.168.1.10)
3. Data link layer: MAC address (00:1A:2B:3C:4D:5E)

19.4 Please explain the purpose of each of the arguments in the following Scapy code snippet. (Page 391) pkt = sniff(iface='eth0', filter='icmp', prn=print_pkt)

Using the function sniff() to capture packets on a specified network, iface='eth0' choose the network interface to capture, in this case Ethernet 0 interface, which is the first Ethernet card on a device. Following the next argument, filter='icmp' is used to capture only ICMP packets to reduce unnecessary packet captures. Finally, the prn=print_pkt is a function to print the information of each captured packet.

19.5 The following result of the "ip address" command shows all the network interfaces on a host. Please write a Scapy program to sniff the packets transmitted over the 192.168.5.0/24 network. You are only interested in the UDP packets to 8.8.8.8's port 53. (Page 391. Test your answer on the Attacker machine.)

```
$ ip -br address
lo        UNKNOWN   127.0.0.1/8
enp0s3    UP        10.0.5.5/24
enp1s8    UP        192.168.5.0/24
```

```
from scapy.all import *
def print_pkt(pkt):
        print(pkt.summary())
pkt = sniff(iface= 'enpls8' filter = 'udp and dst host 8.8.8.8 and dst port 53', prn = print_pkt)
```

Victim:

```
root@859f4f35197d:/# nslookup google.com 8.8.8.8
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.217.78
Name:   google.com
Address: 2607:f8b0:400a:80a::200e
```

Attacker:

```
root@VM:/volumes# python3 test.py
Ether / IP / UDP / DNS Qry "b'google.com.'"
Ether / IP / UDP / DNS Qry "b'google.com.'"
```

19.6 In Scapy, we can use ls(IP) to list all the fields of the IP class. See the following. When we spoof an IP packet, we simply use ip = IP() without setting any argument, what will be the values of the ttl and src IP fields of the packet? How about if we do set the dst field: ip = IP(dst="1.1.1.1")? (Look into the headers in the command line to answer the questions.)

When we create an IP packet using ip = IP() without any arguments, the ttl field defaults to 64, and the src (source IP) field remains None until the packet is sent, at which point it is automatically assigned by the system. If we set ip = IP(dst="1.1.1.1"), the ttl remains 64, and the src is still None until it is sent.

```
>>> from scapy.all import *
>>> ls(IP)
len        : ShortField                          = (None)
id         : ShortField                          = (1)
ttl        : ByteField                           = (64)
proto      : ByteEnumField                       = (0)
chksum     : XShortField                         = (None)
src        : SourceIPField                       = (None)
...
```

```
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = hopopt
  chksum    = None
  src       = 127.0.0.1
  dst       = 127.0.0.1
  \options   \

None
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = hopopt
  chksum    = None
  src       = 10.0.2.15
  dst       = 1.1.1.1
  \options   \
```

19.7 In the following code snippet, we construct an Ethernet frame. (1) What type of object do we get from pkt.payload.payload? (2) What will happen if we do this pkt[UDP]? (3) How do we get the actual payload string hello? (Page 395~396. Use the terminal to get the answer.) pkt = Ether()/IP()/ICMP()/"hello"

1. pkt.payload.payload gives an ICMP object since the first payload refers to the IP layer, and its payload is ICMP.
2. pkt[UDP] will raise an error because the packet does not contain a UDP layer.
3. To get the actual payload string "hello", use pkt.load.

```
[01/31/25]seed@VM:~/.../volumes$ python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from scapy.all import *
>>> pkt = Ether()/IP()/ICMP()/"hello"
>>> pkt.payload.payload
<ICMP  |<Raw  load='hello' |>>
>>> pkt[UDP]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/local/lib/python3.8/dist-packages/scapy/packet.py", line 1185, in __getitem__
    raise IndexError("Layer [%s] not found" % lname)
IndexError: Layer [UDP] not found
>>> print(pkt[Raw].load)
b'hello'
```