UNIVERSITY INSTITUTE OF ENGINEERING AND TECHNOLOGY, PANJAB UNIVERSITY, CHANDIGARH

NETWORK SECURITY AND CRYPTOGRAPHY

PRACTICAL FILE

SUBMITTED BY:
TRISHAN PREET SINGH
UE218105
IT 3rd year

SUBMITTED TO:
SUKHVIR KAUR

PRACTICAL – 1

What is Packet Sniffer?

Packet sniffing is done by using tools called packet sniffer. It can be either filtered or unfiltered. Filtered is used when only specific data packets have to be captured and Unfiltered is used when all the packets have to be captured. WireShark, SmartSniff are examples of packet-sniffing tools.

The act of capturing data packet across the computer network is called packet sniffing. It is similar to as wire tapping to a telephone network. It is mostly used by crackers and hackers to collect information illegally about network. It is also used by ISPs, advertisers and governments.

What is the purpose of Packet Sniffer?

- **Examine the slowdown of the system:** Sometimes, users experience a slowdown of a system, and in such times, packet sniffing will help you dig deep into the problem. It measures the network response time, which is also known as network latency. Thus, it will determine the time required for specific information to travel from the sender to the receiver. Packet sniffing can identify the affected applications so that administrators can fix the problem.
- **Analyze traffic by type**: It is paramount to have a firm grasp over the traffic of your network. When packet sniffers are categorized into types based on destination served IP addresses, you can manage your traffic volume for each type. Thus, you can identify the non-business traffic and minimize it or eliminate it from your traffic.
- **Improve bandwidth**: Business growth is at stake if users complain of slow internet or if your network is entirely down. The key to set back on track is to understand where your network bandwidth is used. A Wi-Fi packet sniffer can retrieve performance and monitor your network security. Hence, you can detect potential issues and resolve the downtime.
- **Enhance security**: A packet sniffer can identify if there is an unusual spike in your network traffic. It can indicate if some intruder is trying to apply for illegal communication or transfer a large amount of data. Thus, it provides you with network security and minimizes cybercriminal work.

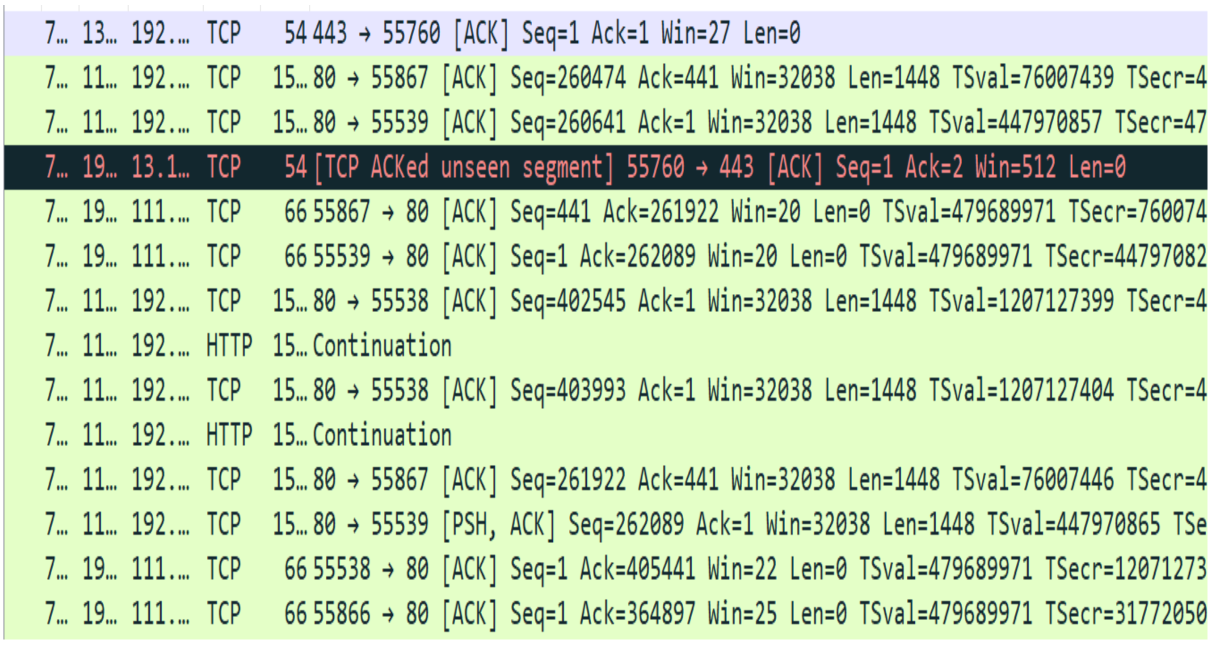
What are the popular Packet Sniffer?

- **SolarWinds Network Packet Sniffer**: SolarWinds Network Packet Sniffer (formerly known as Network Performance Monitor Packet Sniffer) is a network monitoring and analysis tool that captures and analyzes packets transmitted over a network. It allows you to see the data being transmitted over the network, as well as identify issues and troubleshoot problems. With Network Packet Sniffer, you can capture packets in real-time, view packet data in a variety of formats, and apply filters to narrow down the packet data you want to see. You can also use the tool to capture and analyze packets from specific interfaces, protocols, or conversations. Network Packet Sniffer is a powerful tool for network administrators and IT professionals, as it allows you to monitor and analyze network traffic, identify potential issues, and optimize network performance. It is part of the SolarWinds Network Performance Monitor (NPM) suite of network monitoring and analysis tools.

- **Wireshark:** The Wireshark tool is one of the most widely common software as known and uses packet sniffers. It offers an unlimited number of features designed to implement and assist in the dissection and analysis of traffic for it. The Wireshark packet sniffing tool is known for both its data capture and analysis capabilities. We can apply filters to limit the scope of data as well as Wireshark collecting through it, or simply let it collect all traffic passing through your selected networks. Thus, it can only collect the data on a web server with a desktop installed. Since the desktop is not common on the servers many sysadmins choose to use the tcpdump or the WinDump to capture traffic to a file, which they load into Wireshark in-depth analysis.
- **Paessler PRTG:** Paessler PRTG network monitor is a professional all-in-one packet sniffing tool. It will provide valuable insights into your infrastructure and network performance. It supports Windows. It has various possibilities for monitoring everything like bandwidth and traffic. PRTG makes the use of various technologies like SNMP, NetFlow, WMI, network sniffing, etc. while monitoring the data packets.

Components found in capturing the packets of UIET Website:

- **Collecting Packets:**

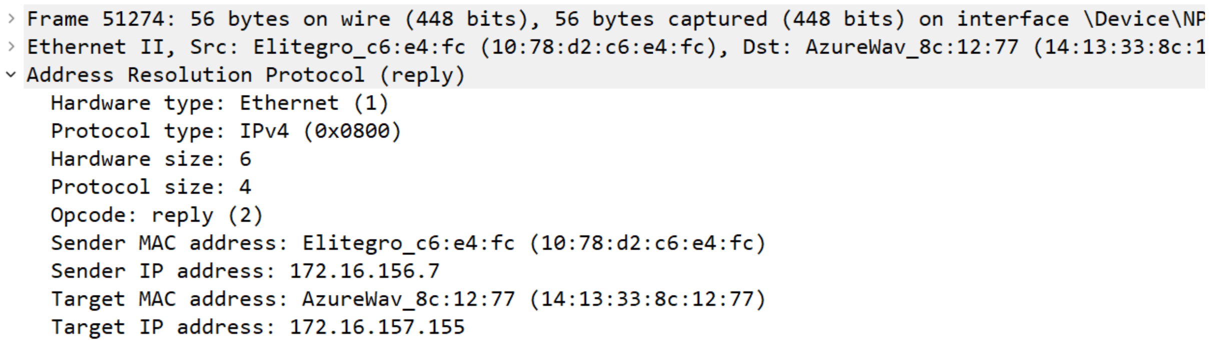


```
7… 13… 192.… TCP    54 443 → 55760 [ACK] Seq=1 Ack=1 Win=27 Len=0
7… 11… 192.… TCP   15… 80 → 55867 [ACK] Seq=260474 Ack=441 Win=32038 Len=1448 TSval=76007439 TSecr=4
7… 11… 192.… TCP   15… 80 → 55539 [ACK] Seq=260641 Ack=1 Win=32038 Len=1448 TSval=447970857 TSecr=47
7… 19… 13.1… TCP    54 [TCP ACKed unseen segment] 55760 → 443 [ACK] Seq=1 Ack=2 Win=512 Len=0
7… 19… 111.… TCP    66 55867 → 80 [ACK] Seq=441 Ack=261922 Win=20 Len=0 TSval=479689971 TSecr=760074
7… 19… 111.… TCP    66 55539 → 80 [ACK] Seq=1 Ack=262089 Win=20 Len=0 TSval=479689971 TSecr=44797082
7… 11… 192.… TCP   15… 80 → 55538 [ACK] Seq=402545 Ack=1 Win=32038 Len=1448 TSval=1207127399 TSecr=4
7… 11… 192.… HTTP  15… Continuation
7… 11… 192.… TCP   15… 80 → 55538 [ACK] Seq=403993 Ack=1 Win=32038 Len=1448 TSval=1207127404 TSecr=4
7… 11… 192.… HTTP  15… Continuation
7… 11… 192.… TCP   15… 80 → 55867 [ACK] Seq=261922 Ack=441 Win=32038 Len=1448 TSval=76007446 TSecr=4
7… 11… 192.… TCP   15… 80 → 55539 [PSH, ACK] Seq=262089 Ack=1 Win=32038 Len=1448 TSval=447970865 TSe
7… 19… 111.… TCP    66 55538 → 80 [ACK] Seq=1 Ack=405441 Win=22 Len=0 TSval=479689971 TSecr=12071273
7… 19… 111.… TCP    66 55866 → 80 [ACK] Seq=1 Ack=364897 Win=25 Len=0 TSval=479689971 TSecr=31772050
```

- **IP V4:**

Pv4 is a version 4 of IP. It is a current version and the most commonly used IP address. It is a 32-bit address written in four numbers separated by 'dot', i.e., periods. This address is unique for each device.

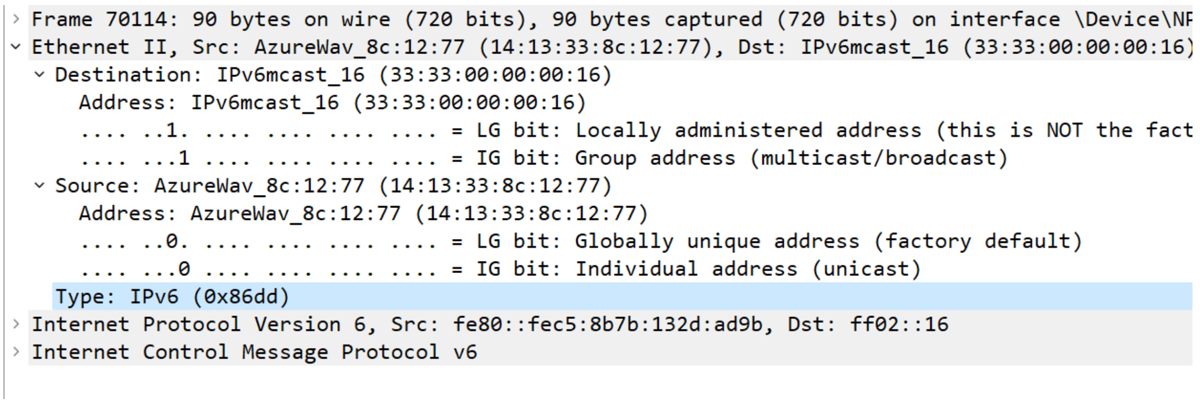
For example, **172.16.254.1**



```
› Frame 51274: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NP
› Ethernet II, Src: Elitegro_c6:e4:fc (10:78:d2:c6:e4:fc), Dst: AzureWav_8c:12:77 (14:13:33:8c:1
⌄ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: Elitegro_c6:e4:fc (10:78:d2:c6:e4:fc)
    Sender IP address: 172.16.156.7
    Target MAC address: AzureWav_8c:12:77 (14:13:33:8c:12:77)
    Target IP address: 172.16.157.155
```

- **IP V6:**

IPv4 produces 4 billion addresses, and the developers think that these addresses are enough, but they were wrong. IPv6 is the next generation of IP addresses. The main difference between IPv4 and IPv6 is the address size of IP addresses. The IPv4 is a 32-bit address, whereas IPv6 is a 128-bit hexadecimal address.

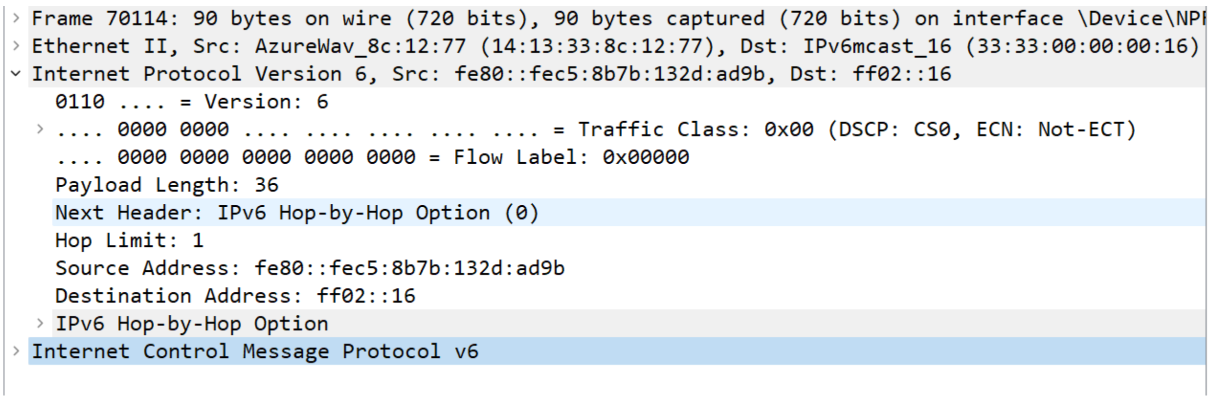


```
› Frame 70114: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface \Device\NP
⌄ Ethernet II, Src: AzureWav_8c:12:77 (14:13:33:8c:12:77), Dst: IPv6mcast_16 (33:33:00:00:00:16)
  ⌄ Destination: IPv6mcast_16 (33:33:00:00:00:16)
      Address: IPv6mcast_16 (33:33:00:00:00:16)
      .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the fact
      .... ...1 .... .... .... .... = IG bit: Group address (multicast/broadcast)
  ⌄ Source: AzureWav_8c:12:77 (14:13:33:8c:12:77)
      Address: AzureWav_8c:12:77 (14:13:33:8c:12:77)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv6 (0x86dd)
› Internet Protocol Version 6, Src: fe80::fec5:8b7b:132d:ad9b, Dst: ff02::16
› Internet Control Message Protocol v6
```

- **ICMP V6:**

ICMPv6 is used by IPv6 nodes to report errors encountered in processing packets, and to perform other internet-layer functions, such as diagnostics (ICMPv6 "ping"). ICMPv6 is an integral part of IPv6, and the base protocol must be fully implemented by every IPv6 node.

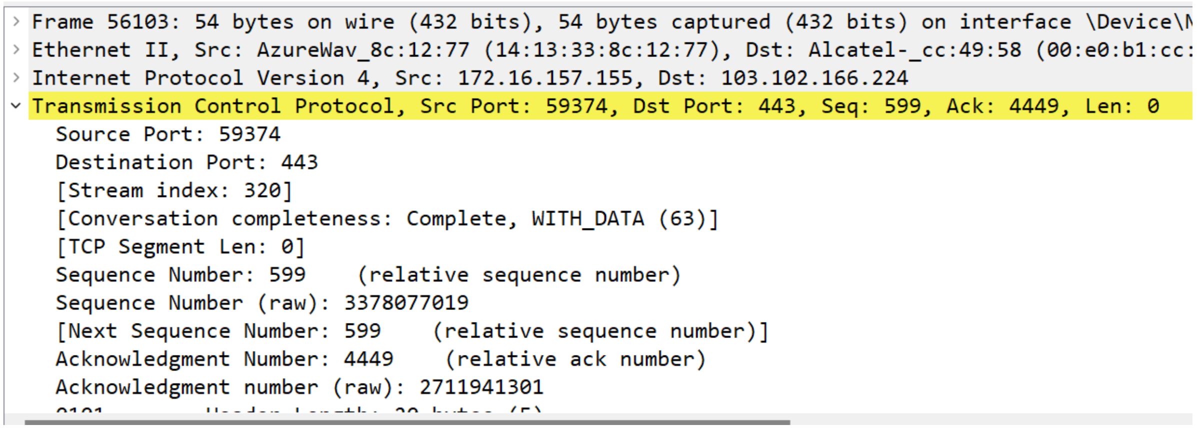
Internet Control Message Protocol (both ICMPv4 and ICMPv6) is a protocol which acts as a communication messenger protocol between the communicating devices in IP network. ICMP messages provide feedback, error reporting and network diagnostic functions in IP networks which are necessary for the smooth operation of IPv6.



```
> Frame 70114: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface \Device\NPI
> Ethernet II, Src: AzureWav_8c:12:77 (14:13:33:8c:12:77), Dst: IPv6mcast_16 (33:33:00:00:00:16)
v Internet Protocol Version 6, Src: fe80::fec5:8b7b:132d:ad9b, Dst: ff02::16
    0110 .... = Version: 6
  > .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
    Payload Length: 36
    Next Header: IPv6 Hop-by-Hop Option (0)
    Hop Limit: 1
    Source Address: fe80::fec5:8b7b:132d:ad9b
    Destination Address: ff02::16
  > IPv6 Hop-by-Hop Option
> Internet Control Message Protocol v6
```

- **TCP:**

TCP (Transmission Control Protocol) is one of the main protocols of the Internet protocol suite. It lies between the Application and Network Layers which are used in providing reliable delivery services. It is a connection-oriented protocol for communications that helps in the exchange of messages between different devices over a network. The Internet Protocol (IP), which establishes the technique for sending data packets between computers, works with TCP.

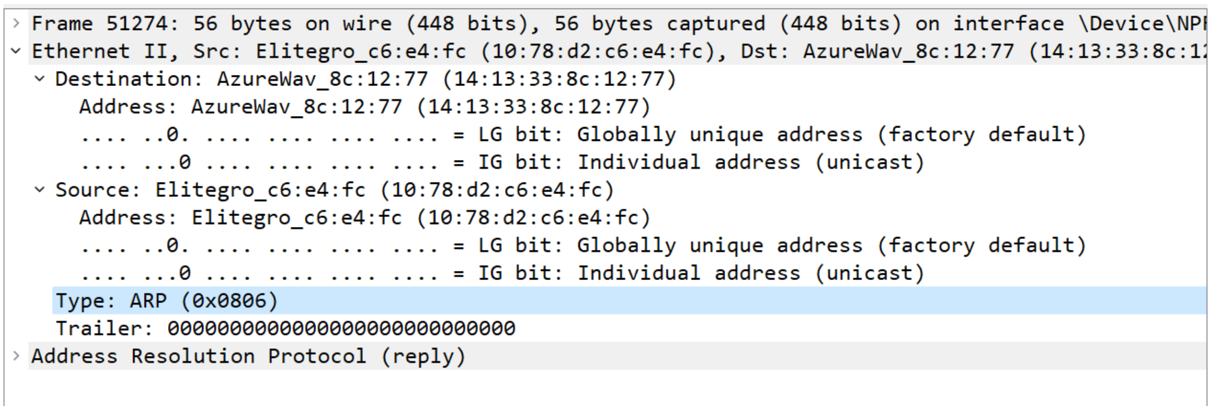


```
> Frame 56103: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\N
> Ethernet II, Src: AzureWav_8c:12:77 (14:13:33:8c:12:77), Dst: Alcatel-_cc:49:58 (00:e0:b1:cc:
> Internet Protocol Version 4, Src: 172.16.157.155, Dst: 103.102.166.224
v Transmission Control Protocol, Src Port: 59374, Dst Port: 443, Seq: 599, Ack: 4449, Len: 0
    Source Port: 59374
    Destination Port: 443
    [Stream index: 320]
    [Conversation completeness: Complete, WITH_DATA (63)]
    [TCP Segment Len: 0]
    Sequence Number: 599    (relative sequence number)
    Sequence Number (raw): 3378077019
    [Next Sequence Number: 599    (relative sequence number)]
    Acknowledgment Number: 4449    (relative ack number)
    Acknowledgment number (raw): 2711941301
```

- **ARP:**

The acronym ARP stands for **Address Resolution Protocol** which is one of the most important protocols of the Data link layer in the OSI model. It is responsible to find the hardware address of a host from a known IP address. There are three basic ARP terms. ARP finds the hardware address, also known as the Media Access Control (MAC) address, of a host from its known IP address.
There are 3 types of ARP:

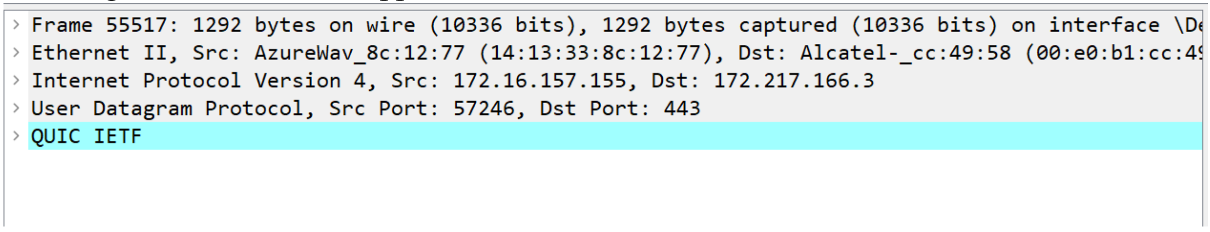
- Reverse ARP
- Proxy ARP
- Inverse ARP



```
> Frame 51274: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPI
v Ethernet II, Src: Elitegro_c6:e4:fc (10:78:d2:c6:e4:fc), Dst: AzureWav_8c:12:77 (14:13:33:8c:1
  v Destination: AzureWav_8c:12:77 (14:13:33:8c:12:77)
      Address: AzureWav_8c:12:77 (14:13:33:8c:12:77)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  v Source: Elitegro_c6:e4:fc (10:78:d2:c6:e4:fc)
      Address: Elitegro_c6:e4:fc (10:78:d2:c6:e4:fc)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: ARP (0x0806)
    Trailer: 000000000000000000000000000
> Address Resolution Protocol (reply)
```

- **QUIC:**

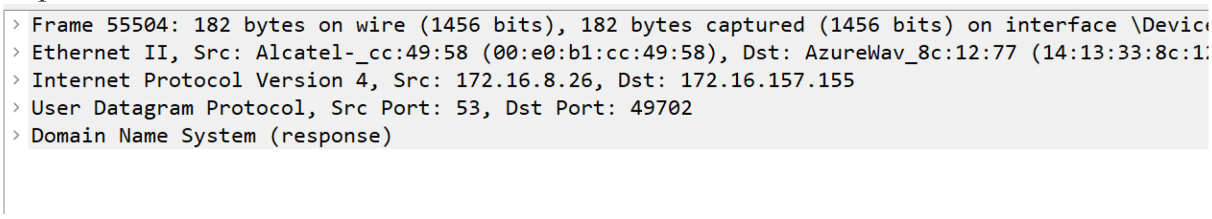
QUIC (**Q**uick **U**DP **I**nternet **C**onnections, pronounced *quick)* is an experimental transport layer network protocol designed by Google. The overall goal is to reduce latency compared to that of TCP. Think of QUIC as being similar to TCP+TLS+HTTP/2 implemented on UDP. Because TCP is implemented at the lowest levels of machinery (operating systems, routing firmware), making changes to TCP is next to impossible given the amount of upgrades that would need to soccur. Since QUIC is built on top of UDP, it suffers from no such limitations and can be integrated into end host applications.



```
> Frame 55517: 1292 bytes on wire (10336 bits), 1292 bytes captured (10336 bits) on interface \De
> Ethernet II, Src: AzureWav_8c:12:77 (14:13:33:8c:12:77), Dst: Alcatel-_cc:49:58 (00:e0:b1:cc:49
> Internet Protocol Version 4, Src: 172.16.157.155, Dst: 172.217.166.3
> User Datagram Protocol, Src Port: 57246, Dst Port: 443
> QUIC IETF
```

- **DNS:**

Domain Name System (DNS) is a hostname for **IP address** translation service. DNS is a distributed database implemented in a hierarchy of name servers. It is an application layer protocol for message exchange between clients and servers. It is required for the functioning of the Internet. It is very difficult to find out the IP address associated with a website because there are millions of websites and with all those websites we should be able to generate the IP address immediately, there should not be a lot of delays for that to happen organization of the database is very important.



```
› Frame 55504: 182 bytes on wire (1456 bits), 182 bytes captured (1456 bits) on interface \Devic
› Ethernet II, Src: Alcatel-_cc:49:58 (00:e0:b1:cc:49:58), Dst: AzureWav_8c:12:77 (14:13:33:8c:1
› Internet Protocol Version 4, Src: 172.16.8.26, Dst: 172.16.157.155
› User Datagram Protocol, Src Port: 53, Dst Port: 49702
› Domain Name System (response)
```

PRACTICAL-2

AIM: Socket Programming

SERVER

Code :-

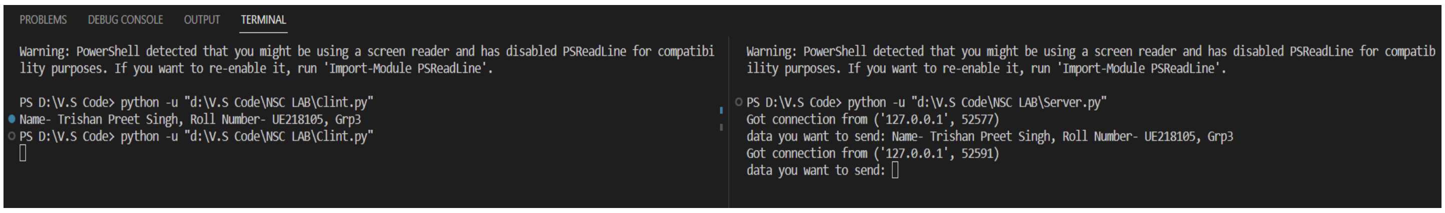
```
import socket
s = socket.socket()
host = "127.0.0.1"
port = 12348
s.bind((host, port))
s.listen(5)
while True:
   c, addr = s.accept()
   print('Got connection from', addr)
   k=input("data you want to send: ")
   c.send(k.encode())
   c.close()
```

CLIENT

Code :

```
import socket              # Import socket module
s = socket.socket()         # Create a socket object
port = 12348               # Reserve a port for your service.
s.connect(("127.0.0.1", port))
print(s.recv(1024).decode())
s.close()
```

Output :-



```
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibi
lity purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.

PS D:\V.S Code> python -u "d:\V.S Code\NSC LAB\Clint.py"
Name- Trishan Preet Singh, Roll Number- UE218105, Grp3
PS D:\V.S Code> python -u "d:\V.S Code\NSC LAB\Clint.py"
```

```
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatib
ility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.

PS D:\V.S Code> python -u "d:\V.S Code\NSC LAB\Server.py"
Got connection from ('127.0.0.1', 52577)
data you want to send: Name- Trishan Preet Singh, Roll Number- UE218105, Grp3
Got connection from ('127.0.0.1', 52591)
data you want to send: 
```

PRACTICAL-3

AIM: Socket Programming

SERVER

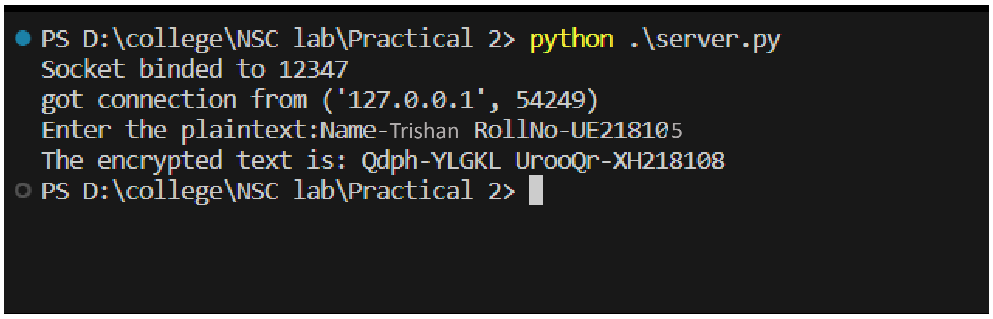
```
import socket
def ecrypt(text,shift):
    result=""
    for i in range(len(text)):
        char=text[i]
        if(char.isalpha()):
            if(char.isupper()):
                result+=chr((ord(char)+shift-65)%26 + 65)
            if(char.islower()):
                result+=chr((ord(char)+shift-97)%26 + 97)
        else:
            result+=char
    return result

s=socket.socket()
port=12347
s.bind(('127.0.0.2',port))
print("Socket binded to %s"%(port))
s.listen(5)
while True:
    c,addr=s.accept()
    print("got connection from",addr)
    k=input("Enter the plaintext:")
    shift=3
    encrypted_text=ecrypt(k,shift)
    print("The encrypted text is:",encrypted_text)
    c.send(encrypted_text.encode())
    c.close()
    break
```

CLIENT

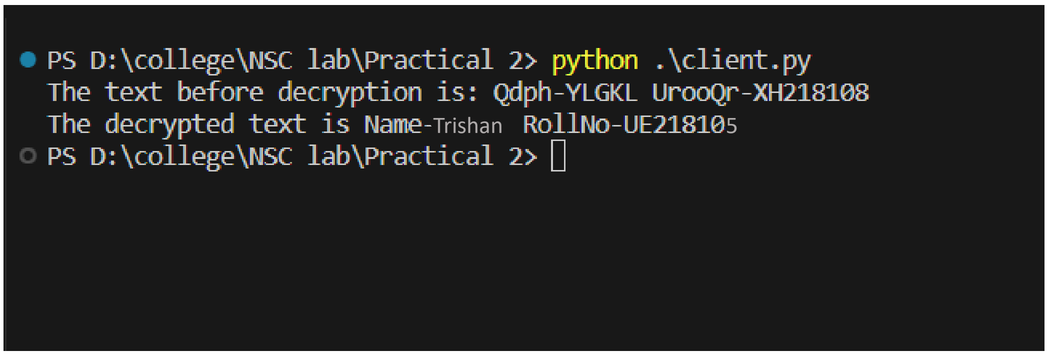
Code :

```
import socket
def decrypt(text,shift):
    result=""
    for i in range(len(text)):
        char=text[i]
        if(char.isalpha()):
            if(char.isupper()):
                result+=chr((ord(char)-shift-65)%26 + 65)
            if(char.islower()):
                result+=chr((ord(char)-shift-97)%26 + 97)
        else:
            result+=char
    return result
import socket
s=socket.socket()
port=12347
s.connect(('127.0.0.2',port))
text=s.recv(1024).decode()
print("The text before decryption is:",text)
shift=3
decrypted_text=decrypt(text,shift)
print("The decrypted text is",decrypted_text)
s.close()
```

Output :-



```
● PS D:\college\NSC lab\Practical 2> python .\server.py
  Socket binded to 12347
  got connection from ('127.0.0.1', 54249)
  Enter the plaintext:Name-Trishan RollNo-UE218105
  The encrypted text is: Qdph-YLGKL UrooQr-XH218108
○ PS D:\college\NSC lab\Practical 2> 
```



```
● PS D:\college\NSC lab\Practical 2> python .\client.py
  The text before decryption is: Qdph-YLGKL UrooQr-XH218108
  The decrypted text is Name-Trishan RollNo-UE218105
○ PS D:\college\NSC lab\Practical 2> 
```

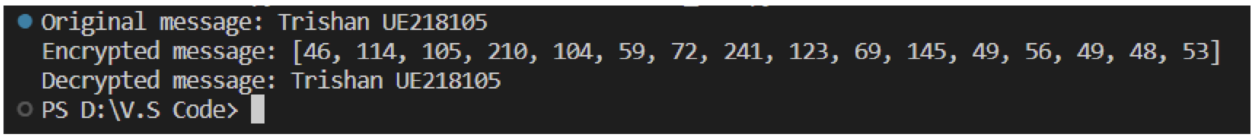
PRACTICAL-4

AIM: Asymmetric Encryption Using Asymmetric Algo.

SERVER

```
import random
def generate_keypair(p, q):
   n = p * q
   phi = (p - 1) * (q - 1)

   e = random.randrange(1, phi)
   while gcd(e, phi) != 1:
      e = random.randrange(1, phi)

   d = mod_inverse(e, phi)
   return ((n, e), (n, d))

def gcd(a, b):
   while b:
      a, b = b, a % b
   return a

def mod_inverse(a, m):
   m0, x0, x1 = m, 0, 1
   while a > 1:
      q = a // m
      m, a = a % m, m
      x0, x1 = x1 - q * x0, x0
   return x1 + m0 if x1 < 0 else x1

def encrypt(public_key, plaintext):
    n, e = public_key
    cipher = [pow(ord(char), e, n) for char in plaintext]
    return cipher

def decrypt(private_key, ciphertext):
    n, d = private_key
    plain = [chr(pow(char, d, n)) for char in ciphertext]
    return ''.join(plain)

p = 13
q = 19
public_key, private_key = generate_keypair(p, q)

message = "Vidhi UE218108"
encrypted_message = encrypt(public_key, message)
decrypted_message = decrypt(private_key, encrypted_message)

print(f"Original message: {message}")
print(f"Encrypted message: {encrypted_message}")
print(f"Decrypted message: {decrypted_message}")
```

Output :-



```
Original message: Trishan UE218105
Encrypted message: [46, 114, 105, 210, 104, 59, 72, 241, 123, 69, 145, 49, 56, 49, 48, 53]
Decrypted message: Trishan UE218105
PS D:\V.S Code>
```