CS-301

Computer Networks Lab Report

Assignment - 4

BTech 5th Semester 2021

Department of Computer Science and Engineering,

National Institute of Technology Karnataka, Surathkal

October 2021

Submitted By:
Deepta Devkota - 191CS117
Akanksha More - 191CS106

Q1. Develop a basic port scanner to check if particular ports are open or closed for an input remote host.

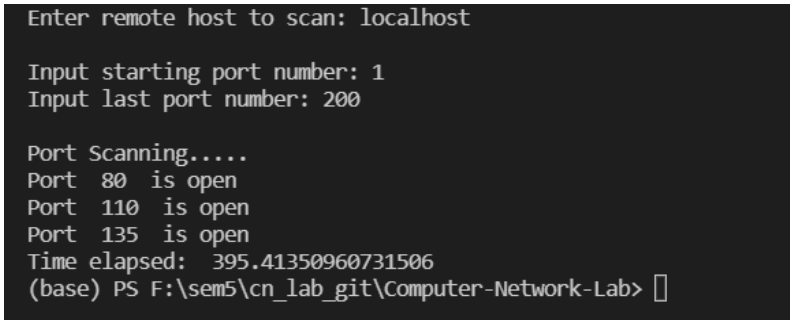
Port Scanner is a network scanner that helps in finding open ports for a given website or IP address.
Code for the given problem:

```
import socket
import time

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

target = input("Enter remote host to scan: ")

def portScan(port):
    try:
        s.connect((target,port))
        return True
    except:
        return False

start = int(input('\nInput starting port number: '))
end = int(input('Input last port number: '))
flag = 0

start_time = time.time()

print("\nPort Scanning.....")
for x in range(start, end+1):
    if portScan(x):
        print('Port ',x, ' is open')
        flag = 1

end_time  = time.time()

if flag==0:
    print('OOPS, no ports are open!')

print("Time elapsed: ", end_time-start_time)
```

For the above problem, we are using the socket library in python to connect the target to the port. In case the connection is successful, we infer that the port is open for the target else it is closed.

1) The flag indicates whether at least one port is open or else it is 0.
2) Start_time and end_time are used to calculate the total time required to scan ports in the given range.

Output with target=localhost:



Thus, we can observe that three ports: 80, 110, and 135 are open for localhost.

Port 80:

It is the port number assigned to the commonly used internet communication protocol, Hypertext Transfer Protocol (HTTP). This port is used by a computer or host to send and receive web-client-based communication and messages from a web server and to send and receive HTML data.

Port 110:

It is used by the Post Office Protocol (POP) which is commonly used by mail clients to retrieve internet mail.

Port 135:

It is used by RPC Endpoint Mapper Service that allows other systems to discover what services are advertised on a machine and what port to find them on.

Q2. Develop a threaded port scanner to check if particular ports are open or closed for a remote host. Determine which Port scanner is efficient.

```python
import socket
import threading
from queue import Queue
import time

print_lock = threading.Lock()
target = input("\nEnter remote host to scan: ")

def portScan(port):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        connection = s.connect((target,port))
        with print_lock:
            print('Port ', port, ' is open!')
        connection.close()
    except:
        pass

def threader():
    while True:
        worker = q.get()
        portScan(worker)
        q.task_done()
q = Queue()
for x in range(30):
    t = threading.Thread(target=threader)
    t.daemon = True
    t.start()
start = int(input('\nInput starting port number: '))
end = int(input('Input last port number: '))
start_time = time.time()
print("\nPort Scanning.....")
for worker in range(start,end+1):
    q.put(worker)
q.join()
end_time  = time.time()
print("\nTime elapsed: ", end_time-start_time, '\n')
```

Threads in python are an entity within a process that can be scheduled for execution. While implementing a threaded port scanner, we utilized the threading, queue, and time module of python. We initialized 30 threads for the above program. The 30 queues have scanned port numbers from start to end as given by the user, therefore the processing time drops down drastically.
In the above program, the concurrency issues were avoided by using the lock on print_lock.

Threading turns out to be more efficient because the process of scanning the ports for availability is distributed among the 30 threads so instead of a single process taking care of the function the workload is distributed.
We can observe the efficiency by the time taken by each of the programs, i.e as seen in the previous question, without threading for 200 port number the time taken was **395.4 secs,** however with threading the time taken was drastically reduced to **14.3 secs** for the same number of ports.

```
Enter remote host to scan: localhost

Input starting port number: 1
Input last port number: 200

Port Scanning.....
Port  80  is open!
Port  110  is open!
Port  135  is open!

Time elapsed:  14.29679012298584

(base) PS F:\sem5\cn_lab_git\Computer-Network-Lab>
```

Q3. Capture UDP packets and with the help of the captured UDP Packets

Dynamic Host Configuration Protocol (DHCP) is a network management protocol used to automate the process of configuring devices in IP networks, thus allowing them to use network services. It uses a connectionless service model, using UDP. It uses a client-server model as follows:

1. **Client(host)**
   A client can request an IP address or other configuration settings from a DHCP server.
2. **Server:**
   The server can then send IP addresses and many DHCP options to clients.
3. **Relay Agent:**
   Transmits DHCP messages between client and server.

a) Analyse UDP DHCP Packets

DHCP Request Packet being analyzed:

16	7.380505	192.168.1.104	192.168.1.1	DHCP	358	DHCP Request - Transaction ID 0x7f9db609

Various fields associated with the packet:

```
Frame 16: 358 bytes on wire (2864 bits), 358 bytes captured (2864 bits) on interface \Device\NPF_{E28C8894-8E61-4A9C-89D7-6D581BB98DA8}, id 0
Ethernet II, Src: IntelCor_83:e1:c2 (d0:ab:d5:83:e1:c2), Dst: BestITWo_6f:f2:28 (00:1e:a6:6f:f2:28)
Internet Protocol Version 4, Src: 192.168.1.104, Dst: 192.168.1.1
User Datagram Protocol, Src Port: 68, Dst Port: 67
Dynamic Host Configuration Protocol (Request)
```

In the Ethernet II field, we can observe destination and source addresses.
The destination address is the DHCP server's MAC address.

```
Destination: BestITWo_6f:f2:28 (00:1e:a6:6f:f2:28)
    Address: BestITWo_6f:f2:28 (00:1e:a6:6f:f2:28)
    .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
```

The source address is the MAC address of the host.

```
Source: IntelCor_83:e1:c2 (d0:ab:d5:83:e1:c2)
    Address: IntelCor_83:e1:c2 (d0:ab:d5:83:e1:c2)
    .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
```

We can view the IP details in the **Internet Protocol Version 4** field.

```
Internet Protocol Version 4, Src: 192.168.1.104, Dst: 192.168.1.1
   0100 .... = Version: 4
   .... 0101 = Header Length: 20 bytes (5)
 > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
   Total Length: 344
   Identification: 0xac0c (44044)
 > Flags: 0x00
   Fragment Offset: 0
   Time to Live: 128
   Protocol: UDP (17)
   Header Checksum: 0x0000 [validation disabled]
   [Header checksum status: Unverified]
   Source Address: 192.168.1.104
   Destination Address: 192.168.1.1
```

1. **The source** address is the IP address of the host.
2. **The destination** address is the IP address of the DHCP server.
3. We can observe that the transport protocol used is **User Datagram Protocol (UDP)** under the **Protocol** subfield.
4. **Flags** consist of three bits: **Reserved**, **Don't fragment** and **More fragments** as shown below.

```
Flags: 0x00
   0... .... = Reserved bit: Not set
   .0.. .... = Don't fragment: Not set
   ..0. .... = More fragments: Not set
```

5. **Time to Live (TTL)** is an 8-bit field that indicates the maximum time the Datagram will be live in the internet system.

UDP details can be viewed under the **User Datagram Protocol** field.

```
User Datagram Protocol, Src Port: 68, Dst Port: 67
   Source Port: 68
   Destination Port: 67
   Length: 324
   Checksum: 0x850f [unverified]
   [Checksum Status: Unverified]
   [Stream index: 1]
 > [Timestamps]
   UDP payload (316 bytes)
```

DHCP is implemented using two UDP port numbers which are the same as for the bootstrap protocol.
Source Port is the bootstrap protocol client port.
Destination Port is the bootstrap protocol server port.

DHCP ACK Packet being analyzed:

```
17 7.481879   192.168.1.1    255.255.255.…  DHCP      590            DHCP ACK      - Transaction ID 0x7f9db609
```

Various fields associated with the packet:

```
Frame 17: 590 bytes on wire (4720 bits), 590 bytes captured (4720 bits) on interface \Device\NPF_{E28C8894-8E61-4A9C-89D7-6D581BB98DA8}, id 0
Ethernet II, Src: BestITWo_6f:f2:28 (00:1e:a6:6f:f2:28), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 192.168.1.1, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 67, Dst Port: 68
Dynamic Host Configuration Protocol (ACK)
```

In the Ethernet II field, we can observe destination and source addresses.
The destination address is the MAC address of the host.

```
Destination: Broadcast (ff:ff:ff:ff:ff:ff)
    Address: Broadcast (ff:ff:ff:ff:ff:ff)
    .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory default)
    .... ...1 .... .... .... .... = IG bit: Group address (multicast/broadcast)
```

The source address is the DHCP server's MAC address.

```
Source: BestITWo_6f:f2:28 (00:1e:a6:6f:f2:28)
    Address: BestITWo_6f:f2:28 (00:1e:a6:6f:f2:28)
    .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
```

We can view the IP details in the **Internet Protocol Version 4** field.

```
Internet Protocol Version 4, Src: 192.168.1.1, Dst: 255.255.255.25
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 576
    Identification: 0x0000 (0)
  > Flags: 0x00
    Fragment Offset: 0
    Time to Live: 64
    Protocol: UDP (17)
    Header Checksum: 0xb704 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.1
    Destination Address: 255.255.255.255
```

1. The source address is the IP address of the DHCP server.
2. Destination address is the IP address of the host.
3. We can observe that the transport protocol used is **User Datagram Protocol (UDP)** under the **Protocol** subfield.
4. **Flags** consist of three bits: **Reserved**, **Don't fragment,** and **More fragments** as shown below.

```
Flags: 0x00
   0... .... = Reserved bit: Not set
   .0.. .... = Don't fragment: Not set
   ..0. .... = More fragments: Not set
```

5. **Time to Live (TTL)** is an 8-bit field that indicates the maximum time the Datagram will be live in the internet system.

UDP details can be viewed under the **User Datagram Protocol** field.

```
User Datagram Protocol, Src Port: 67, Dst Port: 68
   Source Port: 67
   Destination Port: 68
   Length: 556
   Checksum: 0x3c97 [unverified]
   [Checksum Status: Unverified]
   [Stream index: 2]
 > [Timestamps]
   UDP payload (548 bytes)
```

DHCP is implemented using two UDP port numbers which are the same as for the bootstrap protocol.
Source Port is the bootstrap protocol server port.
Destination Port is the bootstrap protocol client port.

Dynamic Host Configuration (ACK) field:

```
Dynamic Host Configuration Protocol (ACK)
   Message type: Boot Reply (2)
   Hardware type: Ethernet (0x01)
   Hardware address length: 6
   Hops: 0
   Transaction ID: 0x7f9db609
   Seconds elapsed: 0
 > Bootp flags: 0x8000, Broadcast flag (Broadcast)
   Client IP address: 192.168.1.104
   Your (client) IP address: 192.168.1.104
   Next server IP address: 0.0.0.0
   Relay agent IP address: 0.0.0.0
   Client MAC address: IntelCor_83:e1:c2 (d0:ab:d5:83:e1:c2)
   Client hardware address padding: 00000000000000000000
   Server host name not given
   Boot file name not given
   Magic cookie: DHCP
 > Option: (53) DHCP Message Type (ACK)
 > Option: (54) DHCP Server Identifier (192.168.1.1)
 > Option: (51) IP Address Lease Time
 > Option: (1) Subnet Mask (255.255.255.0)
 > Option: (3) Router
 > Option: (6) Domain Name Server
 > Option: (255) End
   Padding: 000000000000000000000000000000000000000000000000000000000000000000000000…
```

Upon accepting the DHCP Request message (for IP address renewal) received from the client, the DHCP server also unicasts a DHCP Ack message. It consists of information like client IP address, client MAC address, bootstrap protocol flags, DNS IP, lease time and default gateway along with some other information.

b) analyze UDP DNS Packets

After filtering the DNS packets using Wireshark, we get the response and request packets. The DNS queries below are obtained for the domain name 'www.facebook.com'. Each packet the request and the response have a time, source, and destination address associated with it.
Each query has a code that helps to identify the response message for that query, for example in the very first query the code is 0x46eb and the corresponding response has the same code.

	Time	Source	Destination	Protocol	Length	Info
7	1.468157756	192.168.1.105	192.168.1.254	DNS	83	Standard query 0x46eb A facebook.com OPT
8	1.468369365	192.168.1.105	192.168.1.254	DNS	83	Standard query 0x791f AAAA facebook.com OPT
26	1.479343733	192.168.1.254	192.168.1.105	DNS	99	Standard query response 0x46eb A facebook.com A 157.240.239.35 OPT
27	1.479343880	192.168.1.254	192.168.1.105	DNS	111	Standard query response 0x791f AAAA facebook.com AAAA 2a03:2880:f144:82:face:b00c:0:25de OPT
83	1.850245697	192.168.1.105	192.168.1.254	DNS	98	Standard query 0x7119 A star-mini.c10r.facebook.com OPT
84	1.850418712	192.168.1.105	192.168.1.254	DNS	98	Standard query 0x586f AAAA star-mini.c10r.facebook.com OPT
85	1.878447572	192.168.1.254	192.168.1.105	DNS	114	Standard query response 0x7119 A star-mini.c10r.facebook.com A 157.240.198.35 OPT
86	1.878447817	192.168.1.254	192.168.1.105	DNS	126	Standard query response 0x586f AAAA star-mini.c10r.facebook.com AAAA 2a03:2880:f144:82:face:b0
151	2.558354247	192.168.1.105	192.168.1.254	DNS	92	Standard query 0x23d4 A scontent.xx.fbcdn.net OPT
154	2.567532572	192.168.1.254	192.168.1.105	DNS	108	Standard query response 0x23d4 A scontent.xx.fbcdn.net A 157.240.198.15 OPT
155	2.567900474	192.168.1.105	192.168.1.254	DNS	92	Standard query 0x6040 AAAA scontent.xx.fbcdn.net OPT
156	2.574890693	192.168.1.254	192.168.1.105	DNS	120	Standard query response 0x6040 AAAA scontent.xx.fbcdn.net AAAA 2a03:2880:f044:10:face:b00c:0:3
425	2.870884070	192.168.1.105	192.168.1.254	DNS	88	Standard query 0xbf52 A beacons5.gvt3.com OPT
427	2.876684790	192.168.1.105	192.168.1.254	DNS	102	Standard query 0x1b5d AAAA content-autofill.googleapis.com OPT
429	2.891396782	192.168.1.254	192.168.1.105	DNS	131	Standard query response 0xbf52 A beacons5.gvt3.com CNAME beacons.gvt2.com A 142.250.67.195 OPT
430	2.891715872	192.168.1.105	192.168.1.254	DNS	87	Standard query 0x8096 AAAA beacons.gvt2.com OPT
431	2.892081605	192.168.1.254	192.168.1.105	DNS	130	Standard query response 0x1b5d AAAA content-autofill.googleapis.com AAAA 2404:6800:4009:82f::2
435	2.898935814	192.168.1.254	192.168.1.105	DNS	115	Standard query response 0x8096 AAAA beacons.gvt2.com AAAA 2404:6800:4009:813::2003 OPT

Analyzing the DNS query packet

The DNS request packet has the following fields:

```
▸ Frame 7: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface wlp3s0, id 0
▸ Ethernet II, Src: Chongqin_90:7c:6d (ac:d5:64:90:7c:6d), Dst: Shenzhen_31:47:20 (68:d4:82:31:47:20)
▸ Internet Protocol Version 4, Src: 192.168.1.105, Dst: 192.168.1.254
▸ User Datagram Protocol, Src Port: 41836, Dst Port: 53
▸ Domain Name System (query)
```

As DNS uses UDP in the transport layer we can observe the UDP as one of the fields.

The attributes in the DNS field of the DNS request packet:

```
▾ Domain Name System (query)
    Transaction ID: 0x46eb
  ▾ Flags: 0x0100 Standard query
      0... .... .... .... = Response: Message is a query
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... .0.. .... = Z: reserved (0)
      .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
  ▾ Queries
    ▸ facebook.com: type A, class IN
  ▾ Additional records
    ▸ <Root>: type OPT
    [Response In: 26]
```

The **Transaction ID** is there to uniquely identify a query.
The **Flags** are there to have additional information about the type of query.

- It helps to know whether the packet is a **response or a query**
- The **opcode** is used to know the type of query (standard or reverse)
- **Truncated** implies whether the packet has been truncated to multiple packets. A packet is truncated because of its large size.
- The **recursion desired** signifies whether the query is can be done recursively.

As the packet, we are analyzing is a query message the
Questions: 1, implies that it is a single query message.
The
Answer RRs field is set to zero as this is not a response message.

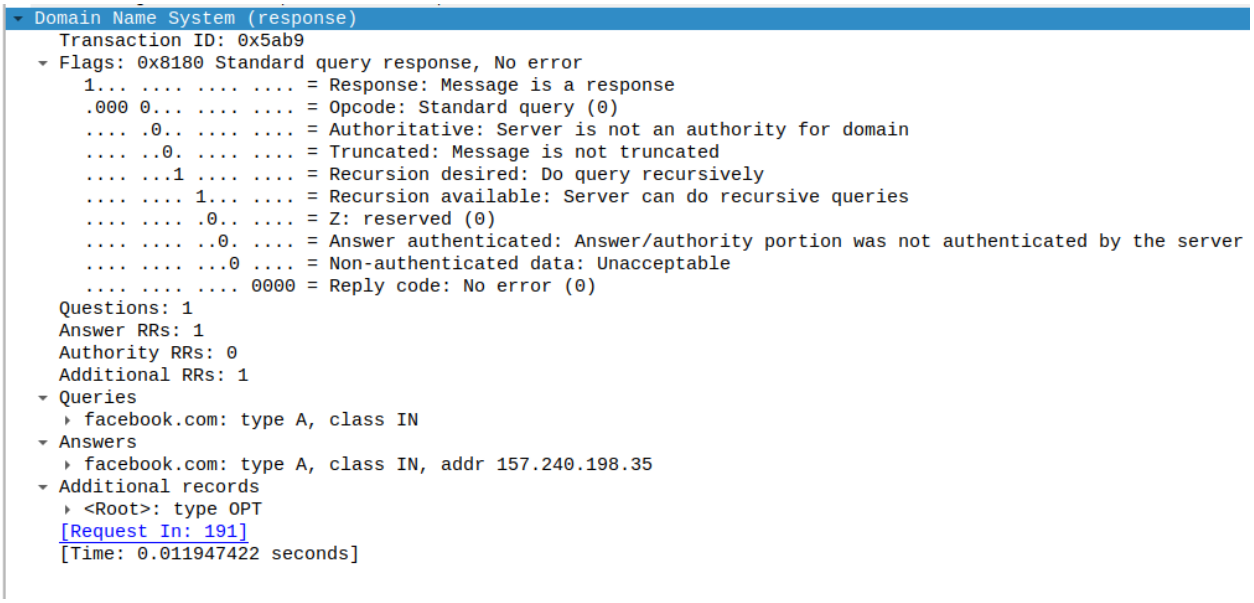
```
▾ Queries
  ▾ facebook.com: type A, class IN
      Name: facebook.com
      [Name Length: 12]
      [Label Count: 2]
      Type: A (Host Address) (1)
      Class: IN (0x0001)
```

The **Queries** field has the query message:
It has the following format:
Query name, **query type**, and the **query class**.
In the above query, our query name is **facebook.com**
The **query type is A**, which is a standard hostname-to-IP address mapping

And the **query class IN** stands for INTERNET

Analyzing the DNS response packet

Additional RRs field implies that the number of additional responses.



As discussed above in the query packet, we have flags in the response packet as well. The **Flags** are there to have additional information about the type of response.

- It helps to know whether the packet is a **response or a query**
- The **opcode** is used to know the type of query (standard or reverse)
- **Authoritative** helps to know whether the server is authoritative for the given domain.
- **Truncated** implies whether the packet has been truncated to multiple packets. A packet is truncated because of its large size.
- The **recursion desired** signifies whether the query is can be done
- The **recursion available** signifies whether the server can perform recursive queries.
- **Answer authenticated** implies whether the answer was authenticated by the server.
- **Reply code: No error**, implies that the DNS query was successfully completed

Answer RRs field is set to one as this is packet contains a single response message.
Additional RRs field implies that the number of additional responses.

```
▾ Answers
  ▾ facebook.com: type A, class IN, addr 157.240.198.35
      Name: facebook.com
      Type: A (Host Address) (1)
      Class: IN (0x0001)
      Time to live: 102 (1 minute, 42 seconds)
      Data length: 4
      Address: 157.240.198.35
```

The **Answers** field has the response message:
It has the following format:
Query name, **query type**, **query class,** and the **response**
In the above query, our query name is **facebook.com**
The **query type is A**, which is a standard hostname-to-IP address mapping
And the **query class IN** stands for INTERNET
As the query type is A the **response message** is the corresponding **IP address** for the domain name facebook.com, here the response is **addr : 157.240.198.35**
