# Network System Design
# CS6100
# Tutorial - 06

**Student Details**
Name: R Abinav
Roll No: ME23B1004
Date: February 27, 2026

## 1 Source Code

### 1.1 Bash Script - Web Visitor

Listing 1: bash script to generate tcp traffic

```bash
#!/bin/bash

sites=(
    "https://www.reddit.com"
    "https://www.google.com"
    "https://www.github.com"
    "https://www.youtube.com"
    "https://www.wikipedia.org"
    "https://www.amazon.com"
    "https://www.twitter.com"
    "https://www.linkedin.com"
    "https://www.netflix.com"
    "https://www.apple.com"
    "https://www.microsoft.com"
    "https://www.stackoverflow.com"
    "https://www.cloudflare.com"
)

for site in "${sites[@]}"; do
    echo "hitting $site"
    curl -s -o /dev/null "$site"
done

echo "done"
```

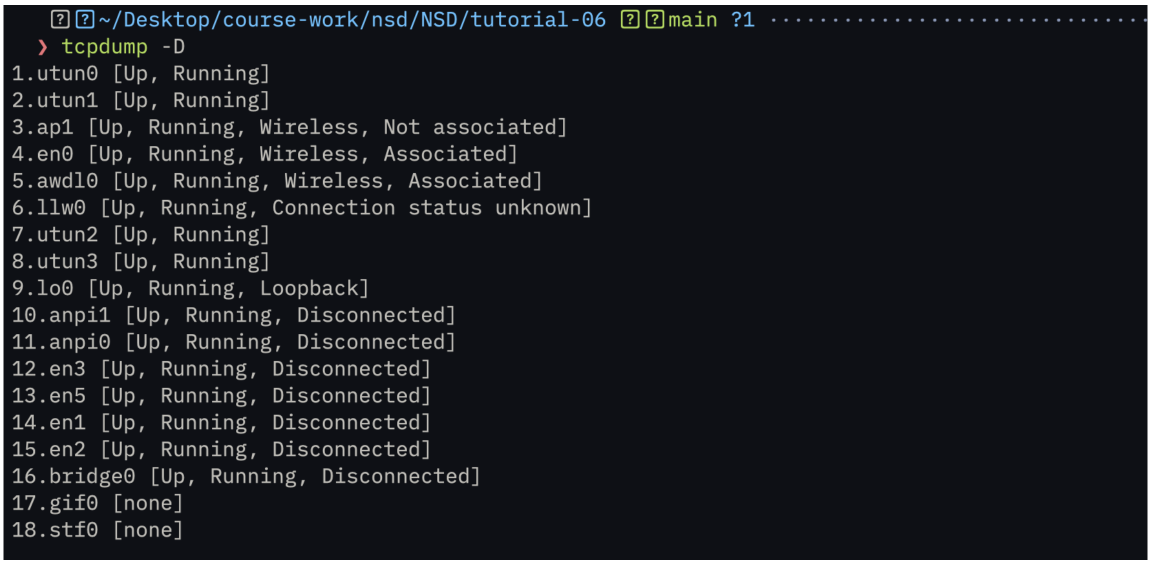
## 1.2 Python Script - TCP Analyzer

Listing 2: tcp connection analyzer using scapy

```python
from scapy.all import rdpcap, TCP, IP

def analyze(file):
    packets = rdpcap(file)
    connections = {}

    for pkt in packets:
        if IP not in pkt or TCP not in pkt:
            continue

        src = pkt[IP].src
        dst = pkt[IP].dst
        sport = pkt[TCP].sport
        dport = pkt[TCP].dport
        flags = pkt[TCP].flags

        #normalize key so both directions map to same connection
        key = tuple(sorted([(src, sport), (dst, dport)]))

        if key not in connections:
            connections[key] = {'syn': 0, 'synack': 0, 'ack': 0}

        if flags & 0x02 and flags & 0x10:
            connections[key]['synack'] += 1
        elif flags & 0x02:
            connections[key]['syn'] += 1
        elif flags & 0x10:
            connections[key]['ack'] += 1

    print(f"total connections found: {len(connections)}\n")

    completed = 0
    failed = 0

    for key, counts in connections.items():
        (a_ip, a_port), (b_ip, b_port) = key
        syn = counts['syn']
        synack = counts['synack']
        ack = counts['ack']

        if syn >= 1 and synack >= 1 and ack >= 1:
            status = "completed"
            completed += 1
        else:
            status = "incomplete/failed"
            failed += 1

        print(f"{a_ip}:{a_port} <-> {b_ip}:{b_port} | syn={syn} synack={synack} ack={ack} | {status}")

    print(f"\ncompleted: {completed} | failed/incomplete: {failed}")

analyze("capture.pcap")
```

## 2 Output

### 2.1 Available Network Interfaces



```
  ??~/Desktop/course-work/nsd/NSD/tutorial-06 ??main ?1 ·····························
  ❯ tcpdump -D
1.utun0 [Up, Running]
2.utun1 [Up, Running]
3.ap1 [Up, Running, Wireless, Not associated]
4.en0 [Up, Running, Wireless, Associated]
5.awdl0 [Up, Running, Wireless, Associated]
6.llw0 [Up, Running, Connection status unknown]
7.utun2 [Up, Running]
8.utun3 [Up, Running]
9.lo0 [Up, Running, Loopback]
10.anpi1 [Up, Running, Disconnected]
11.anpi0 [Up, Running, Disconnected]
12.en3 [Up, Running, Disconnected]
13.en5 [Up, Running, Disconnected]
14.en1 [Up, Running, Disconnected]
15.en2 [Up, Running, Disconnected]
16.bridge0 [Up, Running, Disconnected]
17.gif0 [none]
18.stf0 [none]
```

Figure 1: tcpdump -D showing available network interfaces, en0 selected (wifi, associated)

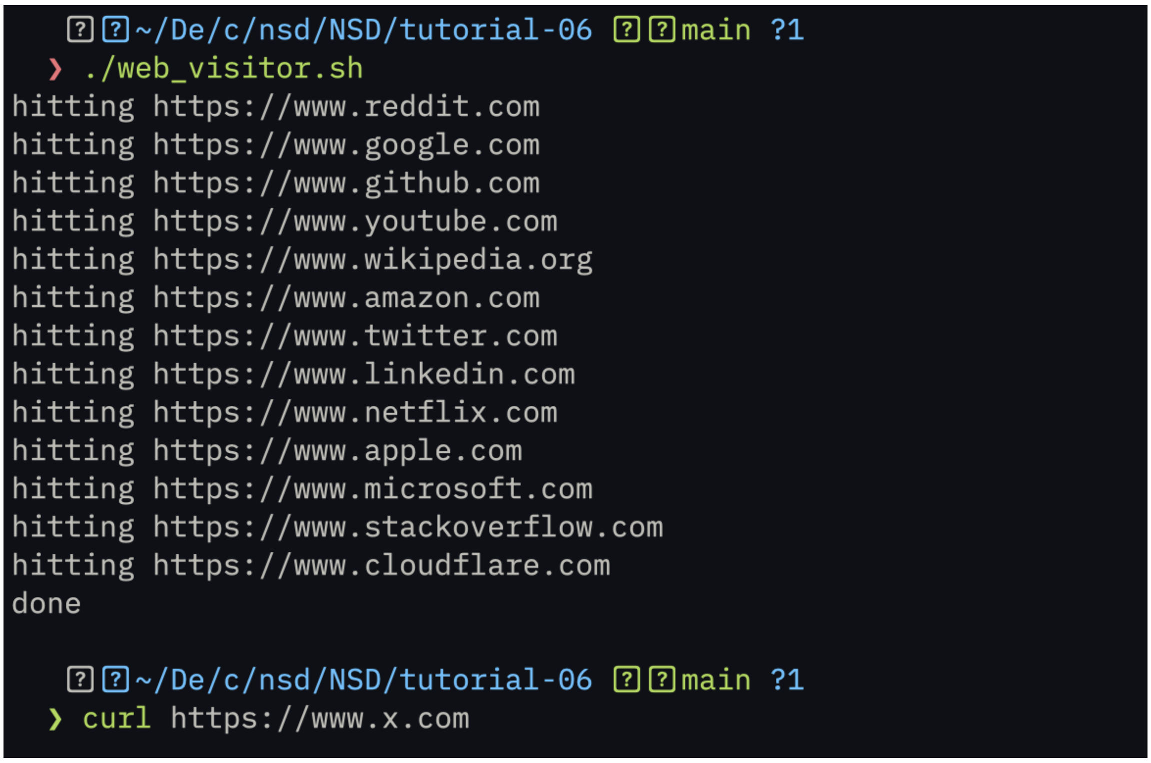
### 2.2 Packet Capture Command

Packets were captured using the following command, filtering only TCP traffic on port 80 and 443 to reduce background noise:

```
sudo tcpdump -i en0 -w capture.pcap -c 10000 'tcp port 443 or tcp
    port 80'
```

The capture was stopped manually after the web visitor script completed. Packets were saved to `capture.pcap`.

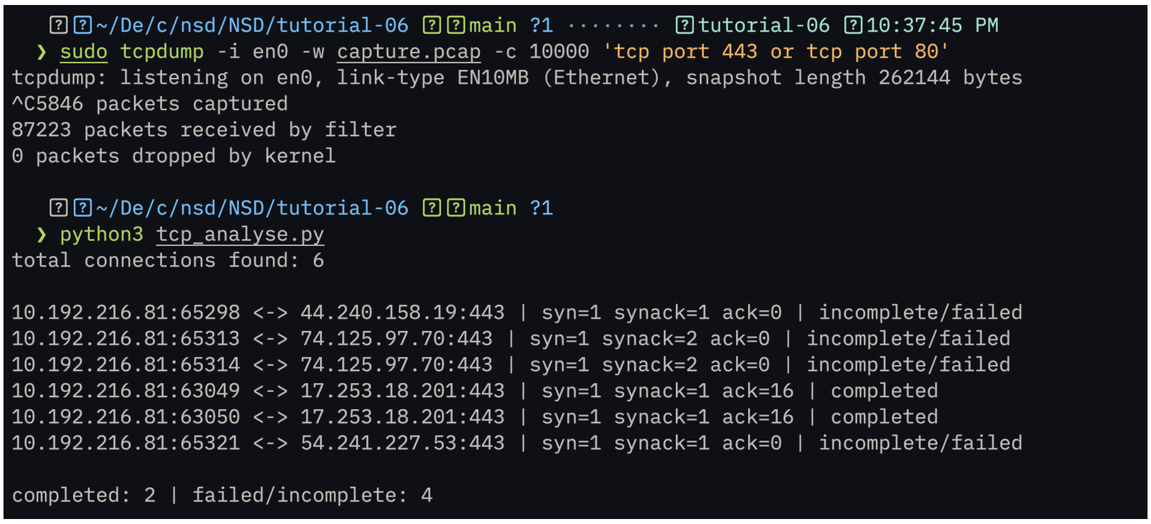
### 2.3 Web Visitor Script Output



```
   ?? ~/De/c/nsd/NSD/tutorial-06 ?? main ?1
  ❯ ./web_visitor.sh
hitting https://www.reddit.com
hitting https://www.google.com
hitting https://www.github.com
hitting https://www.youtube.com
hitting https://www.wikipedia.org
hitting https://www.amazon.com
hitting https://www.twitter.com
hitting https://www.linkedin.com
hitting https://www.netflix.com
hitting https://www.apple.com
hitting https://www.microsoft.com
hitting https://www.stackoverflow.com
hitting https://www.cloudflare.com
done

   ?? ~/De/c/nsd/NSD/tutorial-06 ?? main ?1
  ❯ curl https://www.x.com
```

Figure 2: output of web visitor bash script making curl requests to 13 websites

### 2.4 TCP Analyzer Output



```
  ?? ~/De/c/nsd/NSD/tutorial-06 ?? main ?1 ········ ? tutorial-06 ? 10:37:45 PM
  ❯ sudo tcpdump -i en0 -w capture.pcap -c 10000 'tcp port 443 or tcp port 80'
tcpdump: listening on en0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
^C5846 packets captured
87223 packets received by filter
0 packets dropped by kernel

  ?? ~/De/c/nsd/NSD/tutorial-06 ?? main ?1
  ❯ python3 tcp_analyse.py
total connections found: 6

10.192.216.81:65298 <-> 44.240.158.19:443 | syn=1 synack=1 ack=0 | incomplete/failed
10.192.216.81:65313 <-> 74.125.97.70:443 | syn=1 synack=2 ack=0 | incomplete/failed
10.192.216.81:65314 <-> 74.125.97.70:443 | syn=1 synack=2 ack=0 | incomplete/failed
10.192.216.81:63049 <-> 17.253.18.201:443 | syn=1 synack=1 ack=16 | completed
10.192.216.81:63050 <-> 17.253.18.201:443 | syn=1 synack=1 ack=16 | completed
10.192.216.81:65321 <-> 54.241.227.53:443 | syn=1 synack=1 ack=0 | incomplete/failed

completed: 2 | failed/incomplete: 4
```

Figure 3: output of python tcp analyzer showing connections, handshake flags, and status