Practical Report File

COMPUTER NETWORKS

(CSPC-26)

Computer Engg. Department

Submitted to: Dr. Ankit Jain

Submitted by: HARSH MALIK

12012014

CS-A-02

Index Table for List of the experiments

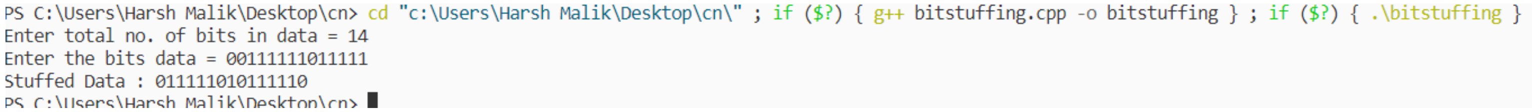
Experiment No: 1

Write a C/C++ Program for bit stuffing.

Code :

```
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;    // if 5 concecutive 1 then push 1 extra bit of 0
vector<char> bits_stuffing(int n, vector<char> data)  {
    vector<char> stuffed_data;
    int count = (data[0]=='1'? 1: 0);
    for(int i=1; i<n; i++){    stuffed_data.push_back(data[i]);
        if(data[i]=='1'){
            count++;
            if(count==5){
                stuffed_data.push_back('0');  count=0;
            }
        }
        else  count=0;
    }
    return stuffed_data;
}
int main()  {
    int n;    cout<<"Enter total no. of bits in data = ";      cin>>n;
    vector<char> data(n);
    cout<<"Enter the bits data = ";
    for(int i=0; i<n; i++)  cin>>data[i];
    vector<char> ans = bits_stuffing(n, data);   cout<<"Stuffed Data : ";
    for(auto it: ans )  cout<<it;
    return 0;
}
```

Sample Output:



```
PS C:\Users\Harsh Malik\Desktop\cn> cd "c:\Users\Harsh Malik\Desktop\cn\" ; if ($?) { g++ bitstuffing.cpp -o bitstuffing } ; if ($?) { .\bitstuffing }
Enter total no. of bits in data = 14
Enter the bits data = 00111111011111
Stuffed Data : 011111010111110
PS C:\Users\Harsh Malik\Desktop\cn> █
```

Experiment No: 02 ***

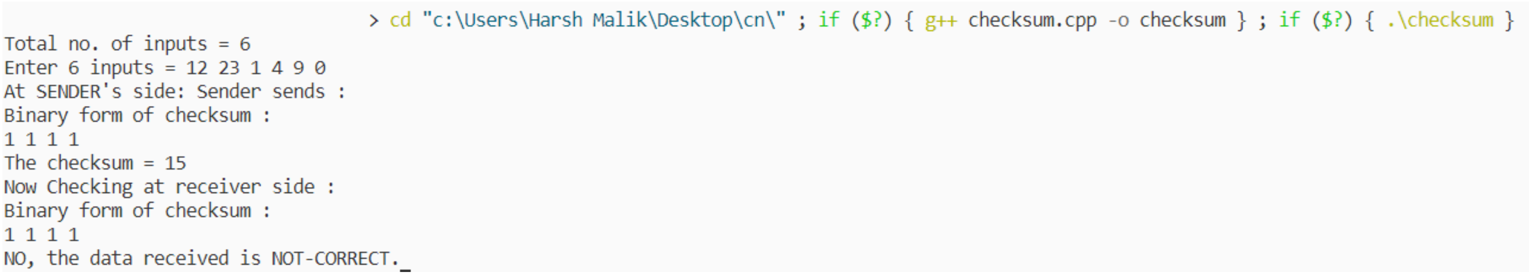
Write a C/C++ program to validate data using Checksum error detection technique.

Source Code:

```
#include<bits/stdc++.h>
using namespace std;
vector<int> decimal_to_binary(int checksum)  {
    vector<int> binary;
    while(checksum!=0)  {
        int rem = checksum%2;
        binary.push_back(rem);   checksum /= 2;
    }
    reverse(binary.begin(), binary.end());   return binary;
}
vector<int> add(vector<int>& binary, vector<int>& extra)  {
    vector<int> sum;
    int n1 = binary.size() , n2 = extra.size();
    int i=n1-1, j=n2-1 , carry=0 , result;
    while(i>=0 && j>=0)  {
        if(binary[i]==0 && extra[j]==0 && carry==0)   result = 0;
        else if(binary[i]==0 && extra[j]==0 && carry==1)   result=1;
        else if((binary[i]==0 && extra[j]==1 && carry==0) || (binary[i]==1 && extra[j]==0
&& carry==0)){ result = 0 ; carry = 1;  }
        else if((binary[i]==1 && extra[j]==1 && carry==0))  {   result = 0;  carry = 1;  }
        else if(binary[i]==1 && extra[j]==1 && carry==1)  {   result = 1;  carry = 1;  }
        sum.insert(sum.begin(), result);   i--;  j--;
    }
    while(i>=0){
        sum.insert(sum.begin(),binary[i]);
        i--;
    }
    while(j>=0){
        sum.insert(sum.begin(),extra[j]);
        j--;
    }
    return sum;
}
int binary_to_decimal(vector<int>& sum_) {
    int num = 0 , count = 0;
    for(int i=sum_.size()-1; i>=0; i--) {
        num = num + sum_[i]*pow(2,count);
        count++;
    }
    return num;
}
void complement(vector<int>& sum_){
    int n = sum_.size();
    for(int i=0; i<n; i++){
        if(sum_[i]==0)  sum_[i] = 1;
        else if(sum_[i]==1)  sum_[i] = 0;
    }
```

```cpp
}
int find_checksum(int n, vector<int>& inputs)  {
    int checksum = 0 , sum= 0;
    for(int i=0; i<inputs.size(); i++)  sum += inputs[i];
    checksum += sum;
    vector<int> binary = decimal_to_binary(checksum);   vector<int> extra;
    if(binary.size() > 4)  {
        for(int i=0; i<binary.size()-4; i++)   extra.push_back(binary[i]);
        binary.erase(binary.begin(), binary.begin()+binary.size()-4);
    }
    vector<int> sum_ = add(binary,extra);
    complement(sum_);
    cout<<"Binary form of checksum : \n";
    for(int i=0; i<sum_.size(); i++)  cout<<sum_[i]<<" ";
    cout<<endl;
    int ans = binary_to_decimal(sum_);   //binary to decimal
    return ans;
}
int main()  {
    int n;
    cout<<"Total no. of inputs = ";  cin>>n;
    vector<int> inputs;   cout<<"Enter "<<n<<" inputs = ";
    for(int i=0; i<n; i++){
        int x;   cin>>x;   inputs.push_back(x);
    }
    cout<<"At SENDER's side: Sender sends : \n";
    int ans = find_checksum(n, inputs);
    cout<<"The checksum = "<<ans<<"\nNow Checking at receiver side : \n";
    inputs.push_back(ans);  ans = find_checksum(n,inputs);
    if(ans == 0)  cout<<"Yes, the data received is CORRECT.\n";
    else  cout<<"NO, the data received is NOT-CORRECT.\n";
    return 0;
}
```

Sample Output:



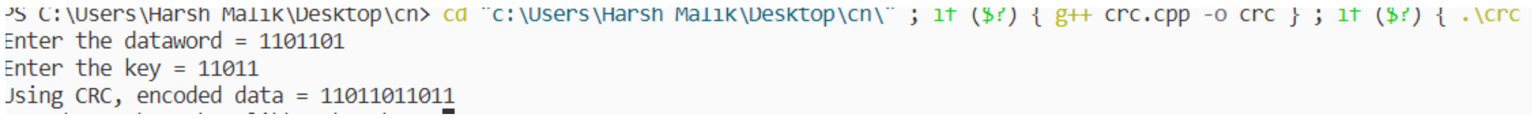
```
                                        > cd "c:\Users\Harsh Malik\Desktop\cn\" ; if ($?) { g++ checksum.cpp -o checksum } ; if ($?) { .\checksum }
Total no. of inputs = 6
Enter 6 inputs = 12 23 1 4 9 0
At SENDER's side: Sender sends :
Binary form of checksum :
1 1 1 1
The checksum = 15
Now Checking at receiver side :
Binary form of checksum :
1 1 1 1
NO, the data received is NOT-CORRECT._
```

Experiment No: 03

Write a C/C++ program to validate data using CRC error detection technique.

```
#include<iostream>
using namespace std;
string do_XOR(string dividend, string key)  {
    string result="";
    for(int i=0; i<key.length(); i++){
        if(dividend[i]==key[i])  result += '0';
        else  result += '1';
    }
    return result;
}
string CRC(string dataword, string key)  {
    string original = dataword;
    for(int i=1; i<=key.length()-1; i++)  dataword += '0';
    string special_divisor="";
    for(int i=1; i<=key.length(); i++)  special_divisor += '0';
    string dividend="";
    for(int i=0; i<key.length(); i++)   dividend += dataword[i];
    if(dividend[0]=='0')  dividend = do_XOR(dividend, special_divisor);
    else  dividend = do_XOR(dividend, key);
    for(int i=key.length(); i<dataword.length(); i++)   {
        dividend.erase(dividend.begin()+0);   dividend += dataword[i];
        if(dividend[0]=='0')  dividend = do_XOR(dividend, special_divisor);
        else  dividend = do_XOR(dividend, key);
    }
    dividend.erase(dividend.begin() + 0); //this is the final remainder.
    original.append(dividend); //this will be the required encoded data.
    return original;
}
int main()  {
    string dataword;
    cout<<"Enter the dataword = ";   cin>>dataword;   string key;
    cout<<"Enter the key = ";   cin>>key;
    string encoded_data = CRC(dataword, key);   cout<<"Using CRC, encoded data = 
"<<encoded_data;
}
```

Sample Output:



```
PS C:\Users\Harsh Malik\Desktop\cn> cd "c:\Users\Harsh Malik\Desktop\cn\" ; if ($?) { g++ crc.cpp -o crc } ; if ($?) { .\crc
Enter the dataword = 1101101
Enter the key = 11011
Using CRC, encoded data = 11011011011
```

Experiment No: 04

Write a C/C++ program to simulate Stop and wait ARQ protocol.

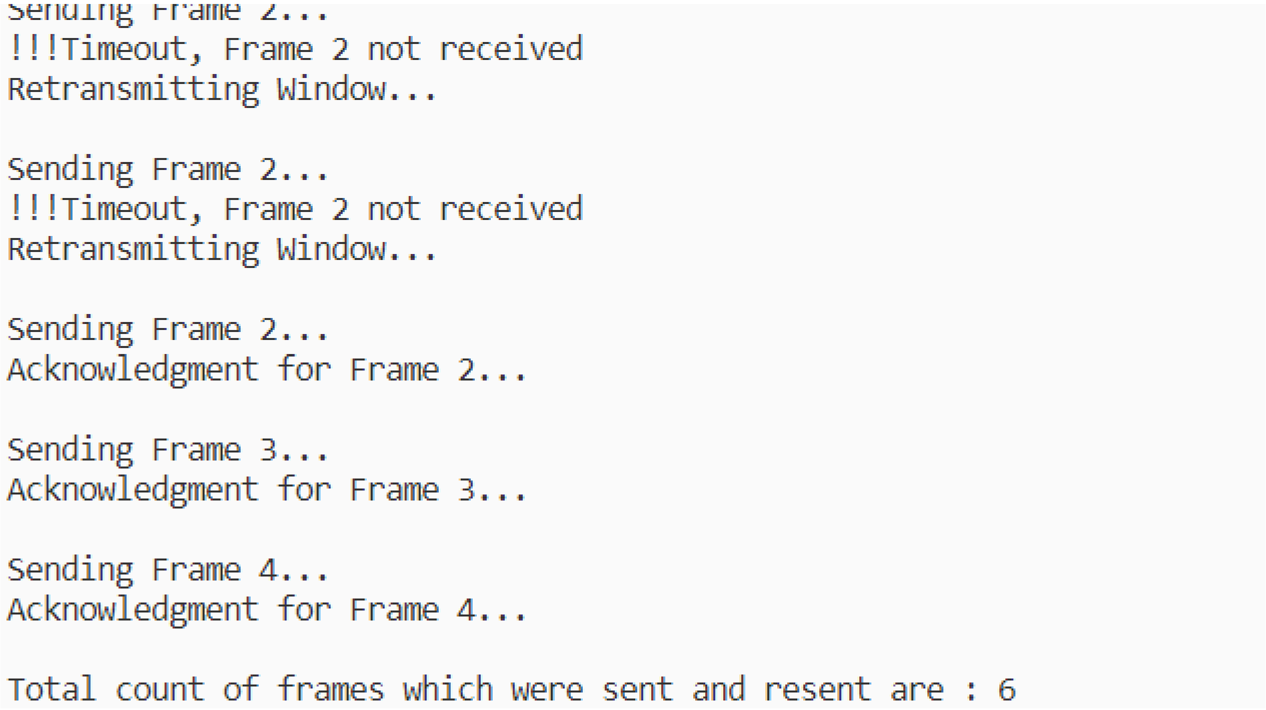
Source Code:

```
#include <bits/stdc++.h>
#include <ctime>
using namespace std;
int transmission(int windowsize, int totalframes, int count) {
int i = 1;
    while (i <= totalframes) {
        int shft = 0;
        for (int k = i; k < i + windowsize && k <=
totalframes; k++) {
            cout << "Sending Frame " << k << "...\n";
count++;  }

        for (int k = i; k < i + windowsize && k <=
totalframes; k++) {
            double f = (double)rand() / RAND_MAX;
            if (f > 0.3) {   // Considering probability of
failure to send ACK = 0.3
                cout << "Acknowledgment for Frame " << k <<
"...\n";   shft++;
            }else {
                cout << "!!!Timeout, Frame " << k << " not
received\n";   cout << "Retransmitting Window...\n";
                break;
            }
        }
        cout << "\n";  i += shft;
    }
    return count;
}
int main() {
    int totalframes, windowsize = 1, count = 0;
    srand(time(NULL));
    cout << "Enter total number of frames : ";
    cin >> totalframes;
    count = transmission(windowsize, totalframes, count);
    cout << "Total count of frames which were sent and resent
are : " << count;
```

```
    return 0;
}
```

Sample Output:



```
Sending Frame 2...
!!!Timeout, Frame 2 not received
Retransmitting Window...

Sending Frame 2...
!!!Timeout, Frame 2 not received
Retransmitting Window...

Sending Frame 2...
Acknowledgment for Frame 2...

Sending Frame 3...
Acknowledgment for Frame 3...

Sending Frame 4...
Acknowledgment for Frame 4...

Total count of frames which were sent and resent are : 6
```

Experiment No: 05

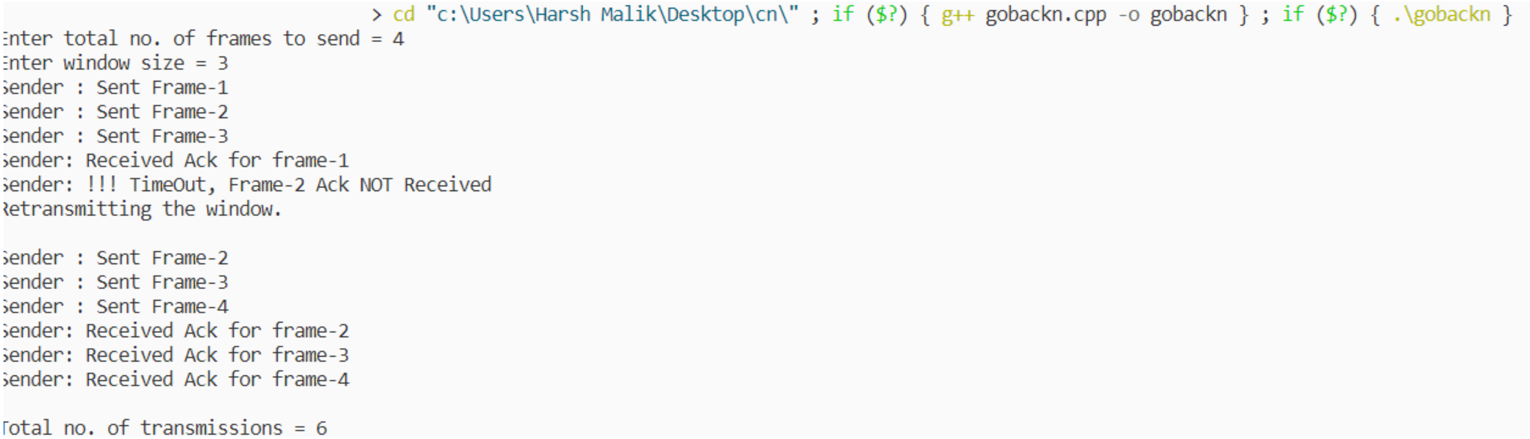
Write a C/C++ program to simulate Go Back to N ARQ.

Source Code:

```
#include<bits/stdc++.h>
using namespace std;
int main()  {
    srand(time(NULL));
    int fn , N , tr = 0 , i = 1 ;
    cout<<"Enter total no. of frames to send = ";   cin>>fn;   cout<<"Enter window size =
"; cin>>N;
    while(i<=fn){   //for each frame starting from 1st frame.
        int x = 0;
        for(int j=i; j<=fn && j<=i+(N-1); j++)   {
            cout<<"Sender : Sent Frame-"<<j<<endl;   tr++;
        }
        for(int j=i; j<i+N && j<=fn; j++){          //Now randomly, get acknowledgement
for some frames.
            int flag = rand()%2;
            if(!flag){
                cout<<"Sender: Received Ack for frame-"<<j<<endl;
                x++;
```

```
      }else{
        cout<<"Sender: !!! TimeOut, Frame-"<<j<<" Ack NOT Received\nRetransmitting the window."<<endl;   break;
      }
    }
    cout<<endl;  i += x;
  }
  cout<<"Total no. of transmissions = "<<tr<<endl;
  return 0;
}
```

Sample Output:



```
                              > cd "c:\Users\Harsh Malik\Desktop\cn\" ; if ($?) { g++ gobackn.cpp -o gobackn } ; if ($?) { .\gobackn }
Enter total no. of frames to send = 4
Enter window size = 3
Sender : Sent Frame-1
Sender : Sent Frame-2
Sender : Sent Frame-3
Sender: Received Ack for frame-1
Sender: !!! TimeOut, Frame-2 Ack NOT Received
Retransmitting the window.

Sender : Sent Frame-2
Sender : Sent Frame-3
Sender : Sent Frame-4
Sender: Received Ack for frame-2
Sender: Received Ack for frame-3
Sender: Received Ack for frame-4

Total no. of transmissions = 6
```

Experiment No: 6

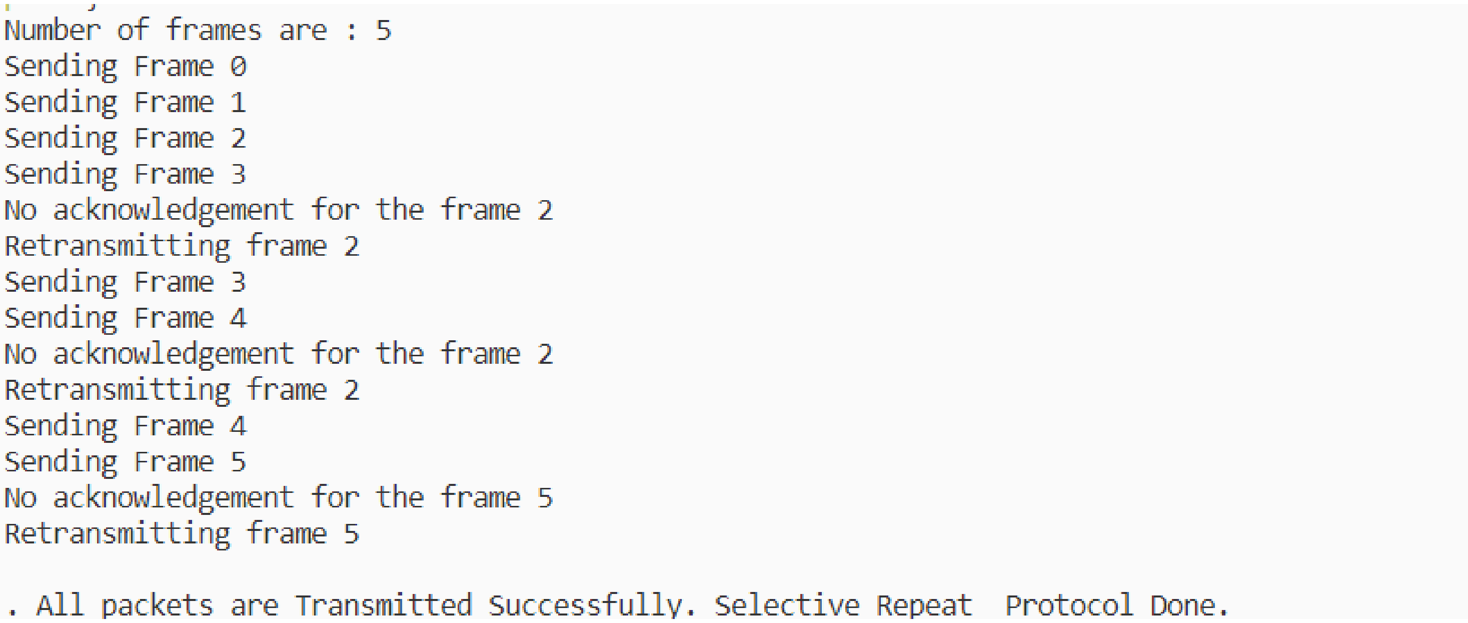
Write a C/C++ program to simulate Selective repeat ARQ

Source Code:

```
#include <iostream>
int tmp1, tmp2, tmp3, tmp4, tmp5, i, windowsize = 4, noofPacket, morePacket;  using namespace std;
int receiver(int tmp1)  {
  int i;
  for  (i = 0; i < 5; i++)   rand();
  i = rand() % tmp1;
  return i;
}
int negack(int tmp1)  {
  int i;
  for  (i = 0; i < 5; i++)    rand();
  i = rand() % tmp1;
  return i;
}
int simulate(int windowsize)  {
  int tmp1, i;
  for  (i = 0; i < 5; i++)   tmp1 = rand();
```

```
    if (tmp1 == 0)  tmp1 = simulate(windowsize);
    i = tmp1 % windowsize;
    if (i == 0)  return windowsize;
    else  return tmp1 % windowsize;
}
int main() {
    for (int i = 0; i < 10; i++)  rand();
    noofPacket = rand() % 10;  cout << "Number of frames are : " << noofPacket;
morePacket = noofPacket;
    while (morePacket >= 0) {
        tmp1 = simulate(windowsize);  windowsize -= tmp1;  tmp4 += tmp1;
        if (tmp4 > noofPacket)   tmp4 = noofPacket;
        for (i = noofPacket - morePacket; i <= tmp4; i++)   cout << "\nSending Frame " <<
i;
        tmp2 = receiver(tmp1);   tmp3 += tmp2;
        if (tmp3 > noofPacket)  tmp3 = noofPacket;  tmp2 = negack(tmp1);  tmp5 += tmp2;
        if (tmp5 != 0)  cout << "\nNo acknowledgement for the frame " << tmp5;   cout <<
"\nRetransmitting frame " << tmp5;
        morePacket -= tmp1;
        if (windowsize <= 0)  windowsize = 4;
    }
    cout << "\n\n. All packets are Transmitted Successfully. Selective Repeat  Protocol
Done.";
}
```

Sample Output:



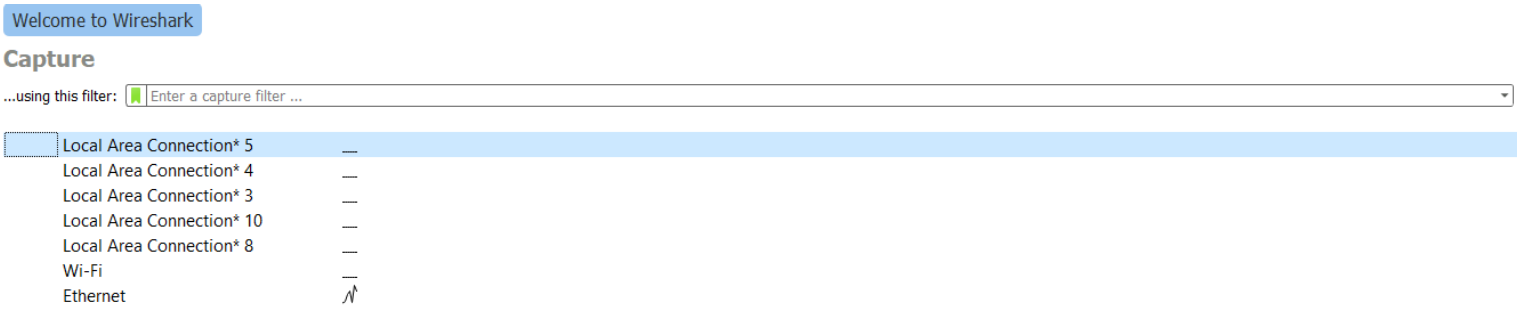
```
Number of frames are : 5
Sending Frame 0
Sending Frame 1
Sending Frame 2
Sending Frame 3
No acknowledgement for the frame 2
Retransmitting frame 2
Sending Frame 3
Sending Frame 4
No acknowledgement for the frame 2
Retransmitting frame 2
Sending Frame 4
Sending Frame 5
No acknowledgement for the frame 5
Retransmitting frame 5

. All packets are Transmitted Successfully. Selective Repeat  Protocol Done.
```

Experiment No: 07 ***

Analyse Network Packets using Wireshark Procedure and sample output:

i) Select internet network from interface window:



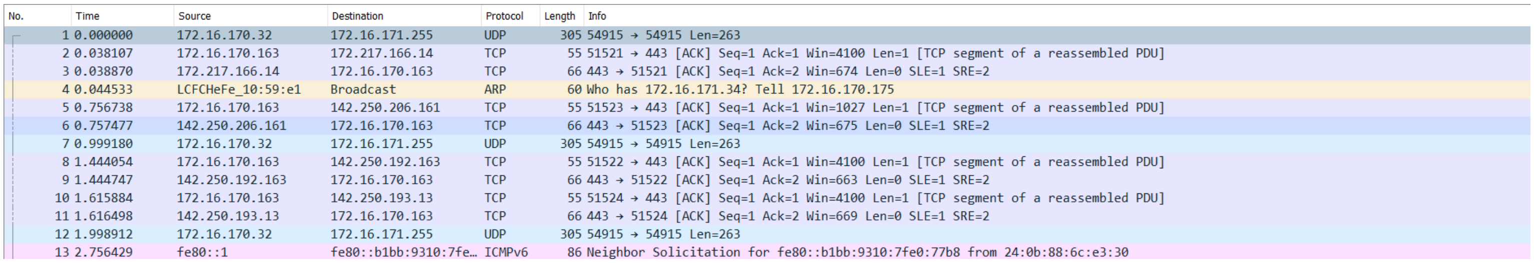
Welcome to Wireshark

Capture

...using this filter: Enter a capture filter ...

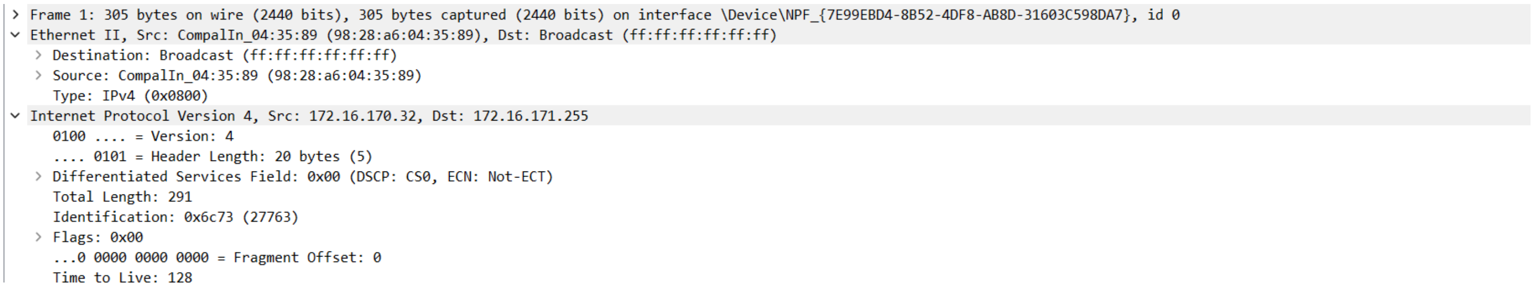
- Local Area Connection* 5
- Local Area Connection* 4
- Local Area Connection* 3
- Local Area Connection* 10
- Local Area Connection* 8
- Wi-Fi
- Ethernet

ii) It will start capturing the packets, now just search any website like ‘nitkkr.ac.in’ in browser and then Stop packet capturing and then analyse the packets in different prtocols like tcp, http , see their ip addresses, data-size, other details etc. Various packet captured are :



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.170.32	172.16.171.255	UDP	305	54915 → 54915 Len=263
2	0.038107	172.16.170.163	172.217.166.14	TCP	55	51521 → 443 [ACK] Seq=1 Ack=1 Win=4100 Len=1 [TCP segment of a reassembled PDU]
3	0.038870	172.217.166.14	172.16.170.163	TCP	66	443 → 51521 [ACK] Seq=1 Ack=2 Win=674 Len=0 SLE=1 SRE=2
4	0.044533	LCFCHeFe_10:59:e1	Broadcast	ARP	60	Who has 172.16.171.34? Tell 172.16.170.175
5	0.756738	172.16.170.163	142.250.206.161	TCP	55	51523 → 443 [ACK] Seq=1 Ack=1 Win=1027 Len=1 [TCP segment of a reassembled PDU]
6	0.757477	142.250.206.161	172.16.170.163	TCP	66	443 → 51523 [ACK] Seq=1 Ack=2 Win=675 Len=0 SLE=1 SRE=2
7	0.999180	172.16.170.32	172.16.171.255	UDP	305	54915 → 54915 Len=263
8	1.444054	172.16.170.163	142.250.192.163	TCP	55	51522 → 443 [ACK] Seq=1 Ack=1 Win=4100 Len=1 [TCP segment of a reassembled PDU]
9	1.444747	142.250.192.163	172.16.170.163	TCP	66	443 → 51522 [ACK] Seq=1 Ack=2 Win=663 Len=0 SLE=1 SRE=2
10	1.615884	172.16.170.163	142.250.193.13	TCP	55	51524 → 443 [ACK] Seq=1 Ack=1 Win=4100 Len=1 [TCP segment of a reassembled PDU]
11	1.616498	142.250.193.13	172.16.170.163	TCP	66	443 → 51524 [ACK] Seq=1 Ack=2 Win=669 Len=0 SLE=1 SRE=2
12	1.998912	172.16.170.32	172.16.171.255	UDP	305	54915 → 54915 Len=263
13	2.756429	fe80::1	fe80::b1bb:9310:7fe…	ICMPv6	86	Neighbor Solicitation for fe80::b1bb:9310:7fe0:77b8 from 24:0b:88:6c:e3:30

iii) By analysing these packets we can get details of our searched website like :



```
> Frame 1: 305 bytes on wire (2440 bits), 305 bytes captured (2440 bits) on interface \Device\NPF_{7E99EBD4-8B52-4DF8-AB8D-31603C598DA7}, id 0
v Ethernet II, Src: CompalIn_04:35:89 (98:28:a6:04:35:89), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
   > Destination: Broadcast (ff:ff:ff:ff:ff:ff)
   > Source: CompalIn_04:35:89 (98:28:a6:04:35:89)
     Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 172.16.170.32, Dst: 172.16.171.255
     0100 .... = Version: 4
     .... 0101 = Header Length: 20 bytes (5)
   > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
     Total Length: 291
     Identification: 0x6c73 (27763)
   > Flags: 0x00
     ...0 0000 0000 0000 = Fragment Offset: 0
     Time to Live: 128
```

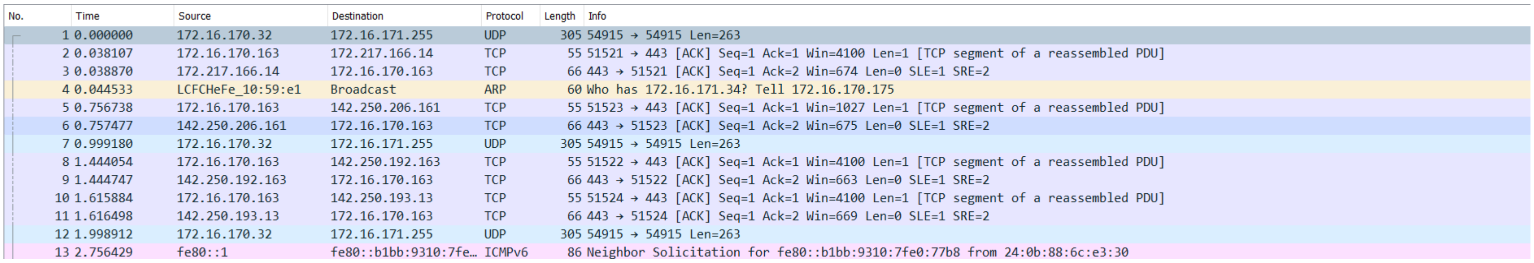
iv) This above screenshot contains, varios details like Host in http, source port, destination port, their ip addresses, ip version, payload and many more details are there about various packets.

Experiment No: 08

To study TCP Three-Way-Handshake using Wireshark

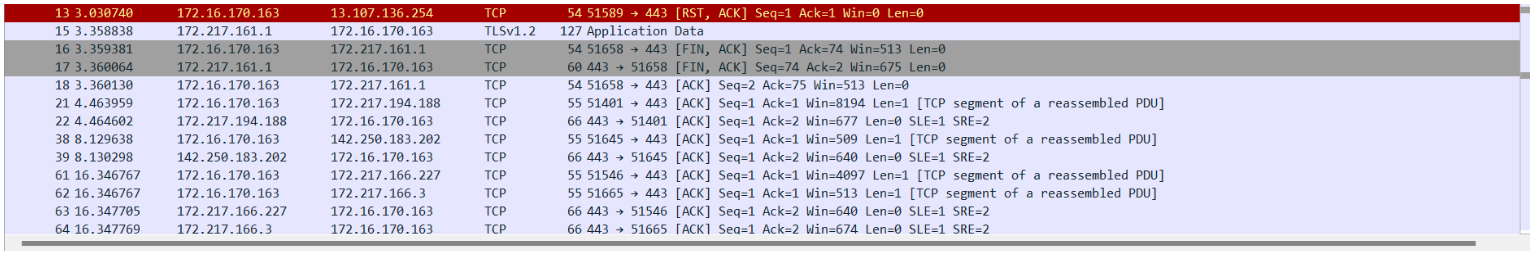
i)Selecting our Internet network from the interface window as shown in above answer

ii) The packet capture starts, then search any website like ‘Google.com’ in browser start and then stop capturing



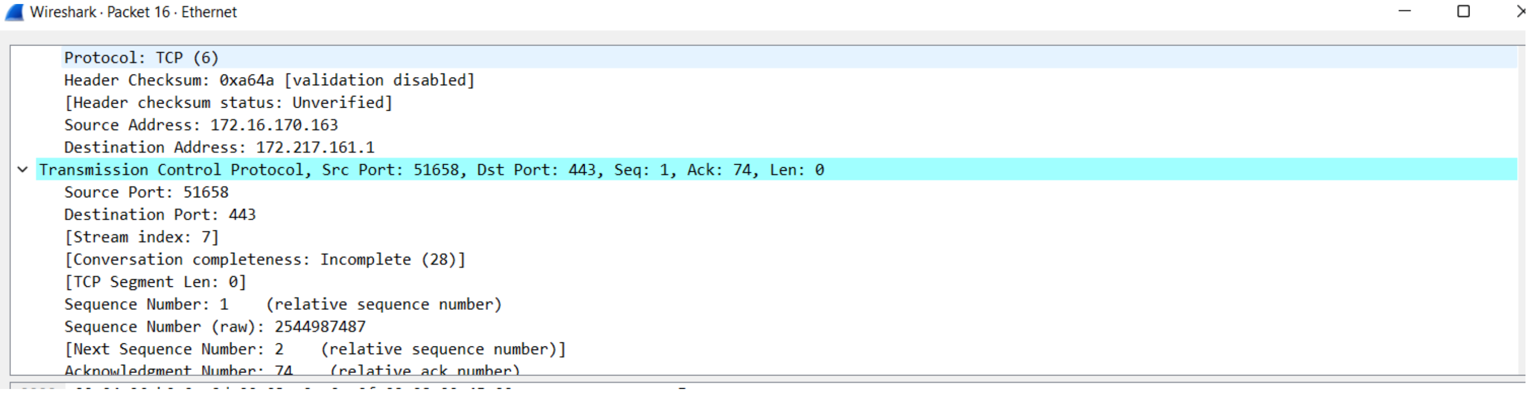
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.170.32	172.16.171.255	UDP	305	54915 → 54915 Len=263
2	0.038107	172.16.170.163	172.217.166.14	TCP	55	51521 → 443 [ACK] Seq=1 Ack=1 Win=4100 Len=1 [TCP segment of a reassembled PDU]
3	0.038870	172.217.166.14	172.16.170.163	TCP	66	443 → 51521 [ACK] Seq=1 Ack=2 Win=674 Len=0 SLE=1 SRE=2
4	0.044533	LCFCHeFe_10:59:e1	Broadcast	ARP	60	Who has 172.16.171.34? Tell 172.16.170.175
5	0.756738	172.16.170.163	142.250.206.161	TCP	55	51523 → 443 [ACK] Seq=1 Ack=1 Win=1027 Len=1 [TCP segment of a reassembled PDU]
6	0.757477	142.250.206.161	172.16.170.163	TCP	66	443 → 51523 [ACK] Seq=1 Ack=2 Win=675 Len=0 SLE=1 SRE=2
7	0.999180	172.16.170.32	172.16.171.255	UDP	305	54915 → 54915 Len=263
8	1.444054	172.16.170.163	142.250.192.163	TCP	55	51522 → 443 [ACK] Seq=1 Ack=1 Win=4100 Len=1 [TCP segment of a reassembled PDU]
9	1.444747	142.250.192.163	172.16.170.163	TCP	66	443 → 51522 [ACK] Seq=1 Ack=2 Win=663 Len=0 SLE=1 SRE=2
10	1.615884	172.16.170.163	142.250.193.13	TCP	55	51524 → 443 [ACK] Seq=1 Ack=1 Win=4100 Len=1 [TCP segment of a reassembled PDU]
11	1.616498	142.250.193.13	172.16.170.163	TCP	66	443 → 51524 [ACK] Seq=1 Ack=2 Win=669 Len=0 SLE=1 SRE=2
12	1.998912	172.16.170.32	172.16.171.255	UDP	305	54915 → 54915 Len=263
13	2.756429	fe80::1	fe80::b1bb:9310:7fe…	ICMPv6	86	Neighbor Solicitation for fe80::b1bb:9310:7fe0:77b8 from 24:0b:88:6c:e3:30

iv) Just search ‘tcp’ via filter, then all tcp details will be there:



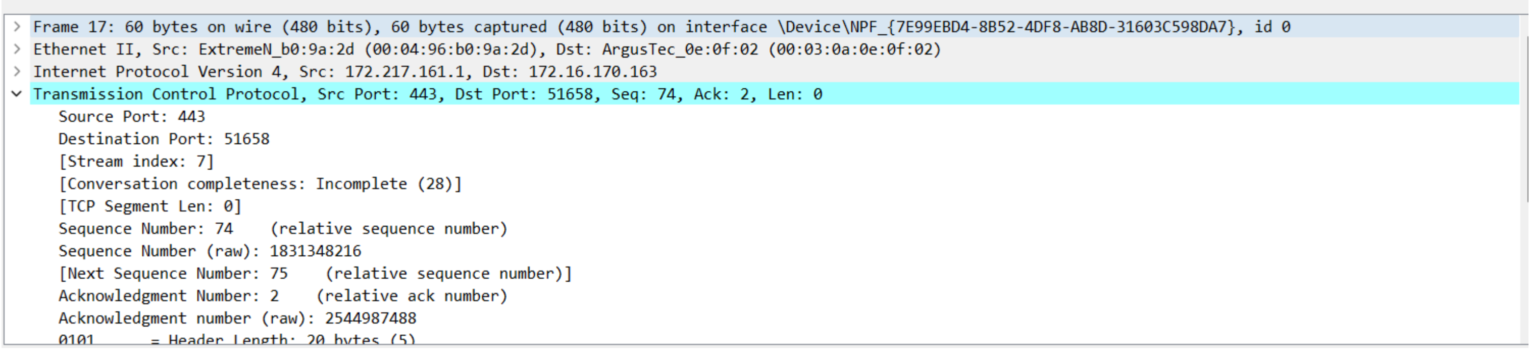
```
13 3.030740   172.16.170.163   13.107.136.254   TCP      54 51589 → 443 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
15 3.358838   172.217.161.1    172.16.170.163   TLSv1.2 127 Application Data
16 3.359381   172.16.170.163   172.217.161.1    TCP      54 51658 → 443 [FIN, ACK] Seq=1 Ack=74 Win=513 Len=0
17 3.360064   172.217.161.1    172.16.170.163   TCP      60 443 → 51658 [FIN, ACK] Seq=74 Ack=2 Win=675 Len=0
18 3.360130   172.16.170.163   172.217.161.1    TCP      54 51658 → 443 [ACK] Seq=2 Ack=75 Win=513 Len=0
21 4.463959   172.16.170.163   172.217.194.188  TCP      55 51401 → 443 [ACK] Seq=1 Ack=1 Win=8194 Len=1 [TCP segment of a reassembled PDU]
22 4.464602   172.217.194.188  172.16.170.163   TCP      66 443 → 51401 [ACK] Seq=1 Ack=2 Win=677 Len=0 SLE=1 SRE=2
38 8.129638   172.16.170.163   142.250.183.202  TCP      55 51645 → 443 [ACK] Seq=1 Ack=1 Win=509 Len=1 [TCP segment of a reassembled PDU]
39 8.130298   142.250.183.202  172.16.170.163   TCP      66 443 → 51645 [ACK] Seq=1 Ack=2 Win=640 Len=0 SLE=1 SRE=2
61 16.346767  172.16.170.163   172.217.166.227  TCP      55 51546 → 443 [ACK] Seq=1 Ack=1 Win=4097 Len=1 [TCP segment of a reassembled PDU]
62 16.346767  172.16.170.163   172.217.166.3    TCP      55 51665 → 443 [ACK] Seq=1 Ack=1 Win=513 Len=1 [TCP segment of a reassembled PDU]
63 16.347705  172.217.166.227  172.16.170.163   TCP      66 443 → 51546 [ACK] Seq=1 Ack=2 Win=640 Len=0 SLE=1 SRE=2
64 16.347769  172.217.166.3    172.16.170.163   TCP      66 443 → 51665 [ACK] Seq=1 Ack=2 Win=674 Len=0 SLE=1 SRE=2
```

v)Firstly, client (my desktop) send SYN request to server('Google.com'), now it will show all necessary details, here 1st line in screenshot shows 'SYN' :



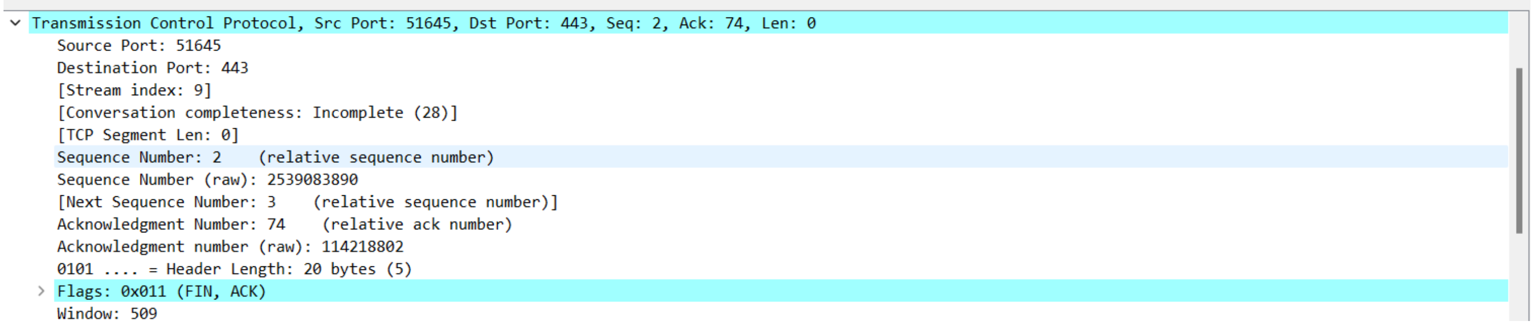
```
    Protocol: TCP (6)
    Header Checksum: 0xa64a [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.16.170.163
    Destination Address: 172.217.161.1
⌄ Transmission Control Protocol, Src Port: 51658, Dst Port: 443, Seq: 1, Ack: 74, Len: 0
    Source Port: 51658
    Destination Port: 443
    [Stream index: 7]
    [Conversation completeness: Incomplete (28)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 2544987487
    [Next Sequence Number: 2    (relative sequence number)]
    Acknowledgment Number: 74    (relative ack number)
```

vi) Now, 2nd is SYN+ACK sent by server('Google.com') to client (My desktop) :



```
> Frame 17: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{7E99EBD4-8B52-4DF8-AB8D-31603C598DA7}, id 0
> Ethernet II, Src: ExtremeN_b0:9a:2d (00:04:96:b0:9a:2d), Dst: ArgusTec_0e:0f:02 (00:03:0a:0e:0f:02)
> Internet Protocol Version 4, Src: 172.217.161.1, Dst: 172.16.170.163
⌄ Transmission Control Protocol, Src Port: 443, Dst Port: 51658, Seq: 74, Ack: 2, Len: 0
    Source Port: 443
    Destination Port: 51658
    [Stream index: 7]
    [Conversation completeness: Incomplete (28)]
    [TCP Segment Len: 0]
    Sequence Number: 74    (relative sequence number)
    Sequence Number (raw): 1831348216
    [Next Sequence Number: 75    (relative sequence number)]
    Acknowledgment Number: 2    (relative ack number)
    Acknowledgment number (raw): 2544987488
    0101 .... = Header Length: 20 bytes (5)
```

vii) Now again, our computer i.e . client will send ACK to server ('Google.com') :



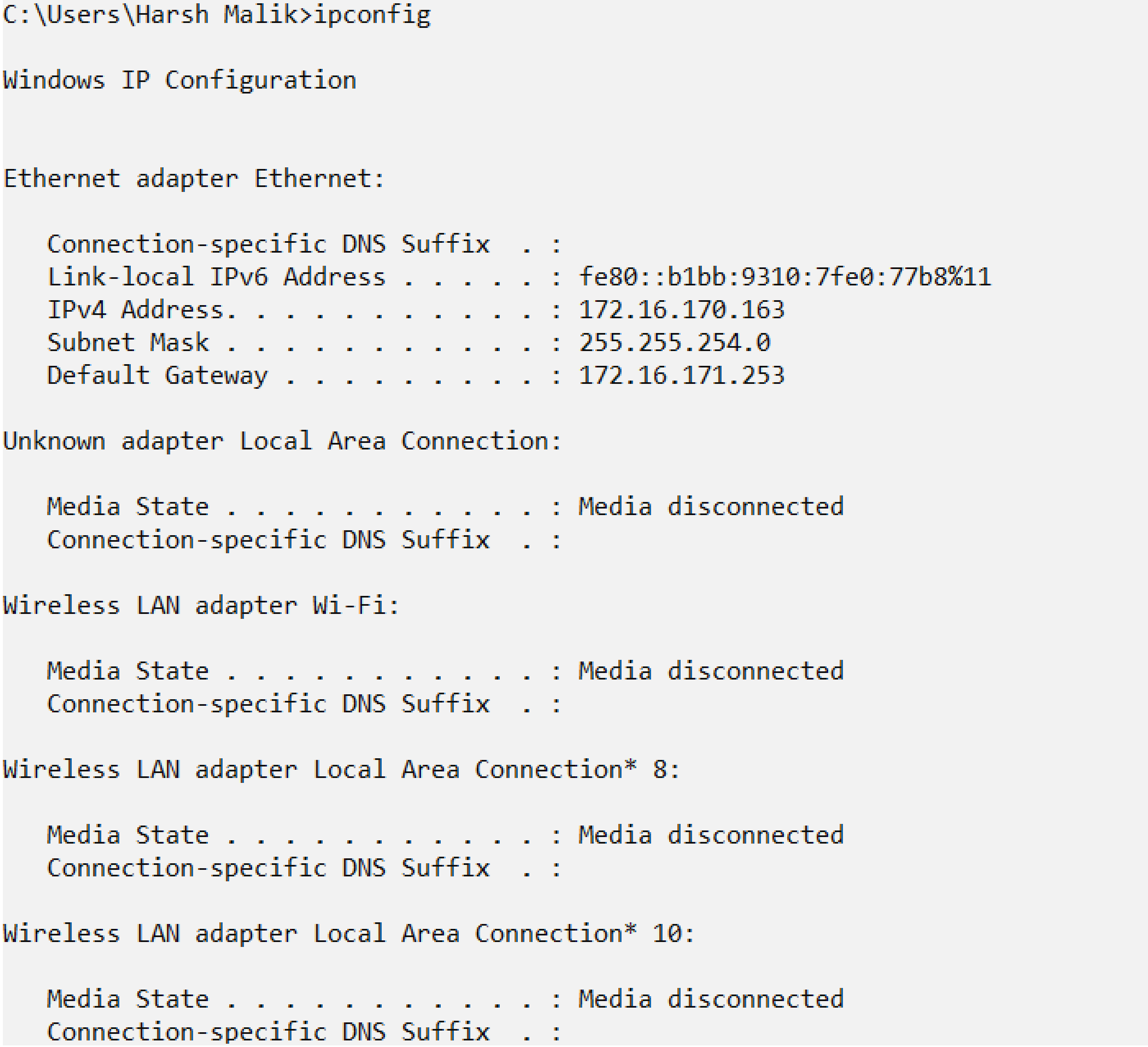
```
⌄ Transmission Control Protocol, Src Port: 51645, Dst Port: 443, Seq: 2, Ack: 74, Len: 0
    Source Port: 51645
    Destination Port: 443
    [Stream index: 9]
    [Conversation completeness: Incomplete (28)]
    [TCP Segment Len: 0]
    Sequence Number: 2    (relative sequence number)
    Sequence Number (raw): 2539083890
    [Next Sequence Number: 3    (relative sequence number)]
    Acknowledgment Number: 74    (relative ack number)
    Acknowledgment number (raw): 114218802
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x011 (FIN, ACK)
    Window: 509
```

(For verification purpose, if we go to browser and just search the destination IP address as shown in above figure during SYN+ACK then that IP address will take us to that website which we have searched e.g. Google.com)

Experiment No: 09

ipconfig Analysis in command prompt.

Sample Output:



```
C:\Users\Harsh Malik>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::b1bb:9310:7fe0:77b8%11
   IPv4 Address. . . . . . . . . . . : 172.16.170.163
   Subnet Mask . . . . . . . . . . . : 255.255.254.0
   Default Gateway . . . . . . . . . : 172.16.171.253

Unknown adapter Local Area Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 8:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 10:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
```