CSN-503 | ACN Project

PACKET HEADER ANALYSIS

Group 3

Members:

- Dayanand Raut (20535010)
- Swapnil Naik (20535029)
- Sagar Gaur (20535023)
- Atul Sharma (20535007)
- Mithlesh Kumar Yadav (20535016)
- Sajal Chourasiya (20535024)
- Gaurish Mishra (20535013)

SN	Name	Enrollment No	Brief Contribution	Page
1	Dayanand Raut	20535010	Analysis of ipv4, design of data structure to store fields of packet, extracting mac addresses of source and destination, code cleanup	5,6,10 , 24, 25
2	Swapnil Naik	20535029	Carray file parsing, protocol parsing and analysis of protocols used in each layer.	11,13, 14,15, 20,22, 23
3	Sagar Gaur	20535023	Extracting source and destination ports and IP addresses, Analysis of ipv6 and ARP packet. Contribution in report.	4,6,7, 17,18, 19
4	Atul Sharma	20535007	Extracting the Ethertype field , discarding the packets other than ipv4, ipv6 and ARP packets , debugging of code	10,16, 21
5	Mithlesh Kumar Yadav	20535016	Extracting port number ,checking of packet whether it is ARP,IPV4,IPV6 or not ,code optimization and debugging ,contribution in ppt and report	16,19, 23
6	Sajal Chourasiya	20535024	Extracting the application layer protocol using port number, worked on application layer part of print summary. Worked on presentation.	13,18, 23
7	Gaurish Mishra	20535013	Printing the summary , Testing the code , Refining report	11

TABLE OF CONTENTS

Problem Statement

Given a file containing some packets captured (in C array format). Analyze the packets and determine all the fields in the packet header like source/destination MAC, IP addresses, port numbers, and different protocols used in different layers, etc. The application should take the c array format as input and display all the parameters as an output. It should also display which are the most used protocols in different layers.

Description

Using Wireshark for generating the packets and exporting captured packets in C array format.

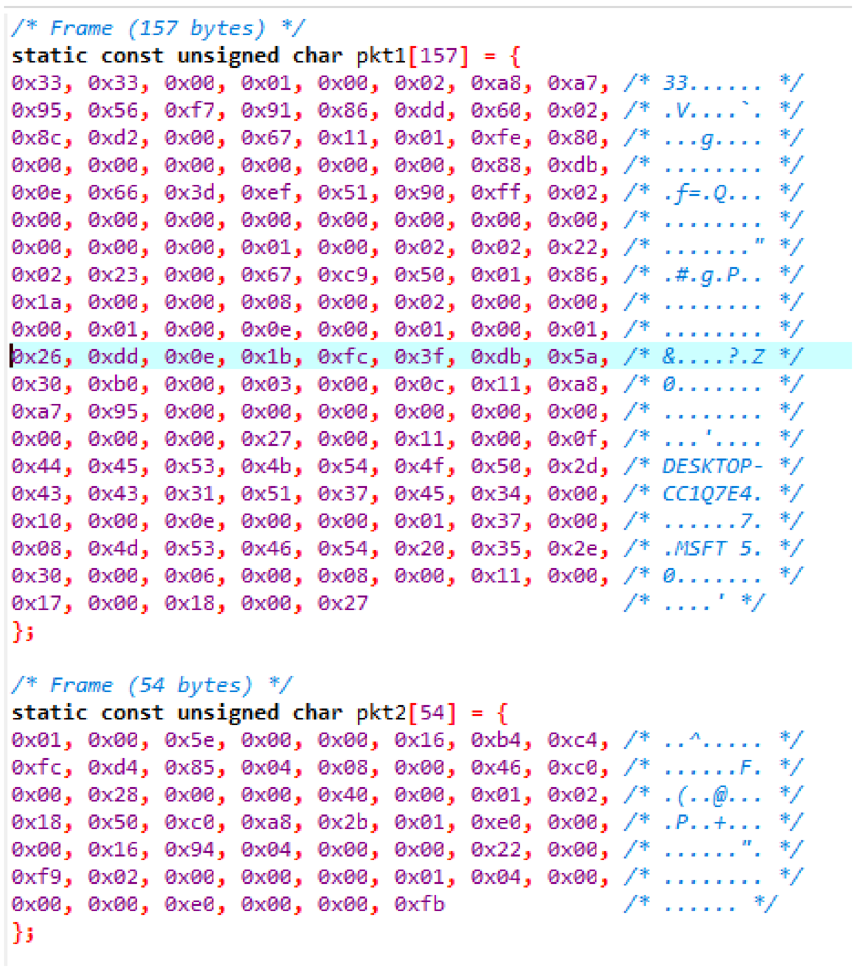
Wireshark: Wireshark is a network packet analyzer which presents captured packet data in as much detail as possible. It is an open-source software for analyzing the packets.

It can capture traffic from many different network media types, including Ethernet, Wireless LAN, Bluetooth, USB, and more. It can save captured packets in many formats including pcapng, libpcap and many more.

Wireshark provides a variety of options for exporting packet data. The formats available for exporting are plain text, C arrays, CSV, PSML, PDML, JSON.

Analysis and Discussion

Analysis of C array packet



```
/* Frame (157 bytes) */
static const unsigned char pkt1[157] = {
0x33, 0x33, 0x00, 0x01, 0x00, 0x02, 0xa8, 0xa7, /* 33...... */
0x95, 0x56, 0xf7, 0x91, 0x86, 0xdd, 0x60, 0x02, /* .V....`. */
0x8c, 0xd2, 0x00, 0x67, 0x11, 0x01, 0xfe, 0x80, /* ...g.... */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x88, 0xdb, /* ........ */
0x0e, 0x66, 0x3d, 0xef, 0x51, 0x90, 0xff, 0x02, /* .f=.Q... */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* ........ */
0x00, 0x00, 0x00, 0x01, 0x00, 0x02, 0x02, 0x22, /* ......." */
0x02, 0x23, 0x00, 0x67, 0xc9, 0x50, 0x01, 0x86, /* .#.g.P.. */
0x1a, 0x00, 0x00, 0x08, 0x00, 0x02, 0x00, 0x00, /* ........ */
0x00, 0x01, 0x00, 0x0e, 0x00, 0x01, 0x00, 0x01, /* ........ */
0x26, 0xdd, 0x0e, 0x1b, 0xfc, 0x3f, 0xdb, 0x5a, /* &....?.Z */
0x30, 0xb0, 0x00, 0x03, 0x00, 0x0c, 0x11, 0xa8, /* 0....... */
0xa7, 0x95, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* ........ */
0x00, 0x00, 0x00, 0x27, 0x00, 0x11, 0x00, 0x0f, /* ...'.... */
0x44, 0x45, 0x53, 0x4b, 0x54, 0x4f, 0x50, 0x2d, /* DESKTOP- */
0x43, 0x43, 0x31, 0x51, 0x37, 0x45, 0x34, 0x00, /* CC1Q7E4. */
0x10, 0x00, 0x0e, 0x00, 0x00, 0x01, 0x37, 0x00, /* ......7. */
0x08, 0x4d, 0x53, 0x46, 0x54, 0x20, 0x35, 0x2e, /* .MSFT 5. */
0x30, 0x00, 0x06, 0x00, 0x08, 0x00, 0x11, 0x00, /* 0....... */
0x17, 0x00, 0x18, 0x00, 0x27                    /* ....' */
};

/* Frame (54 bytes) */
static const unsigned char pkt2[54] = {
0x01, 0x00, 0x5e, 0x00, 0x00, 0x16, 0xb4, 0xc4, /* ..^..... */
0xfc, 0xd4, 0x85, 0x04, 0x08, 0x00, 0x46, 0xc0, /* ......F. */
0x00, 0x28, 0x00, 0x00, 0x40, 0x00, 0x01, 0x02, /* .(..@... */
0x18, 0x50, 0xc0, 0xa8, 0x2b, 0x01, 0xe0, 0x00, /* .P..+... */
0x00, 0x16, 0x94, 0x04, 0x00, 0x00, 0x22, 0x00, /* ......". */
0xf9, 0x02, 0x00, 0x00, 0x00, 0x01, 0x04, 0x00, /* ........ */
0x00, 0x00, 0xe0, 0x00, 0x00, 0xfb              /* ...... */
};
```

Fig. 1

C array file contains the packet in different arrays. Like as shown in figure 1 packets are captures and stored in character array.

<u>Analysis of the fields of ipv4 based on the hexacode from captured frame</u>.

Field Name	Byte position
Ethernet	
Destination mac	0-5
Source mac	6-11
Type	12-13
IP	
Version	14th first nibble

Header length	(14th second nibble) * 4
Source address	26-29
Destination address	30-33
Flags	20
Fragment Offset	20-21
TTL	22
Protocol	23
Total Length	16-17
UDP / TCP	
Source port	34-35
Destination port	36-37
Length	38-39

<u>In case of IPv6 in IP byte positions described below</u>:

Version: 14th byte first nibble
Priority: 14th byte second nibble and 15th byte first nibble
Flow label: 15th byte second nibble and 16-17
Payload length: 18-19
Next header: 20
Hop limit: 21
Source address: 22-37
Destination address: 38-53
TCP/UDP
Source port: 54-55
Destination port: 56-57
Length: 58-59

ARP Packet

ARP packet is of 42 bytes total. ARP protocol is identified by **type field** (12-13th byte position) of packet. Type field value for ARP protocol is 0806.

Analyzing the fields based on the hexacode from captured packet with byte position

Field: byte position
Destination MAC: 0-5
Source MAC: 6-11
Type: 12-13
Hardware type: 14-15
Protocol type: 16-17

Hardware size: 18
Protocol size: 19
Opcode: 20-21
Sender IP address: 28-31
Destination IP address: 38-41

ARP packet does not contain source and destination ports. So, in the code source port and destination port fields are stored –1

Conclusion

As a part of this project, packets were captured using Wireshark and exported to c array file. The file was parsed in c++ and various fields were analyzed from each packet. The results of analysis were displayed. Finally, the overall summary was prepared to show the different protocols used in different layers and most commonly used protocols were identified.

After analyzing various packets, we found that at various layers following protocols were most frequently used:

Application Layer	:	https
Transport Layer	:	TCP
Network Layer	:	IPv4
Data Link Layer	:	Ethernet

References

https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xml
https://en.wikipedia.org/wiki/Transport_layer
https://en.wikipedia.org/wiki/Network_layer
https://blog.packet-foo.com/2015/01/determining-network-protocols/
https://www.howtogeek.com/104278/how-to-use-wireshark-to-capture-filter-and-inspect-packets/

Source Code

packet_analyser.cpp

```
#include<bits/stdc++.h>
using namespace std;

map<string,string> aplProtocolType; // to store application protocols
map<string,string> protocolType; // stores TYPE OF PROTOCOL - (hex, name_of_protocol)
/**
Data structure to hold the values of different fields of packet.
*/
class Packet{
public:
   string srcIPAddress;
   string destIPAddress;
   string srcMacAddress;
   string destMacAddress;
   int srcPort;
   int destPort;
   string protocol;
   string ethType;
   string aplProtocol;

   void printPacket(int counter){

cout<<counter<<"\t"<<protocol<<"\t"<<srcIPAddress<<"\t"<<destIPAddress<<"\t"<<srcMacAd
dress<<"\t"<<destMacAddress<<"\t"<<srcPort<<"\t"<<destPort<<"\t"<<ethType<<"\t"<<aplPro
tocol<<endl;
   }
};

/**
function to print the packets from the list of packet object.
*/
void printPackets(vector<Packet> packets){
   cout<<"Printing packets"<<endl;

cout<<"Packet"<<"\t"<<"Protocol"<<"\t"<<"srcIPAddress"<<"\t"<<"destIPAddress"<<"\t"<<"src
MacAddress"<<"\t"<<"destMacAddress"<<"\t"<<"srcPort"<<"\t"<<"destPort"<<"\t"<<"ethType
"<<"\t"<<"aplProtocol"<<endl;
```

```
    cout<<"---------------------------------------------------------------------------------------------------------------
------------------"<<endl;
    for(int i=0; i<packets.size(); i++){
        packets[i].printPacket(i+1);
    }
}

/**
function to generate the summary of protocols used in different layer.
*/
void printSummary(vector<Packet> packets) {
    cout<<"--------------------SUMMARY----------------------"<<endl;
    map<string, int> nwproto;
    map<string, int> transproto;
    map<string, int> aplproto;

    vector<string> nwprotolist = {"ICMP","IGMP","DDP","IPv6-Frag","IPv6-Route","IPv6-
ICMP","IPv6-NoNxt","EGP","EIGRP","IPX-in-IP","OSPF","PIM"};
    for(int i=0; i<packets.size(); i++){
        if(find(nwprotolist.begin(), nwprotolist.end(), packets[i].protocol) != nwprotolist.end()) {
            if(nwproto.count(packets[i].protocol))
                nwproto[packets[i].protocol]++;
            else
                nwproto[packets[i].protocol] = 1;
        }
        else {
            if(nwproto.count(packets[i].ethType))
                nwproto[packets[i].ethType]++;
            else
                nwproto[packets[i].ethType] = 1;
        }
    }

    map<string, int>::iterator itr;
    string mostUsednwProtocol = "";
    int nwmax = 0;
    cout<<"Protocols used in network layer: "<<endl;
    for(itr= nwproto.begin(); itr!=nwproto.end(); itr++) {
        cout<<itr->first<<" "<<itr->second<<endl;
        if(nwmax < itr->second) {
            mostUsednwProtocol = itr->first;
            nwmax = itr->second;
        }
    }

    cout<<"Most used protocol on network layer: "<<mostUsednwProtocol<<endl;
    cout<<"--------------------------------------------------"<<endl;

    //-------- transport layer part
    vector<string> transprotocolList =
{"TCP","UDP","DCCP","FC","IL","RDP","SCTP","TCP","UDP","UDPLite"};
    for(int i=0; i<packets.size(); i++){
        if(find(transprotocolList.begin(), transprotocolList.end(), packets[i].protocol) !=
transprotocolList.end()) {
            // it means it is some transport layer protocol
            if(transproto.count(packets[i].protocol))
                transproto[packets[i].protocol]++;
            else
                transproto[packets[i].protocol] = 1;
        }
        else {
            if(transproto.count("OTHER")) {
                transproto["OTHER"]++;
            }
            else
                transproto["OTHER"] = 1;
        }
    }

    string mostUsedtransProtocol = "";
    int transmax = 0;
    cout<<"Protocols used in Transport layer: "<<endl;
    for(itr= transproto.begin(); itr!=transproto.end(); itr++) {
        cout<<itr->first<<" "<<itr->second<<endl;
        if(transmax < itr->second && itr->first.compare("OTHER") != 0) {
            mostUsedtransProtocol = itr->first;
            transmax = itr->second;
        }
    }
    cout<<"Most used protocol on Transport layer: "<<mostUsedtransProtocol<<endl;
    cout<<"------------------------------------------------"<<endl;
    //--------- application layer part

    for(int i=0; i<packets.size(); i++){
        if(aplproto.count(packets[i].aplProtocol)) {
            aplproto[packets[i].aplProtocol]++;
        }
        else {
            aplproto[packets[i].aplProtocol] = 1;
        }
    }

    string mostUsedaplProtocol = "";
    int aplmax = 0;
    cout<<"Protocols used in Application layer: "<<endl;
    for(itr= aplproto.begin(); itr!=aplproto.end(); itr++) {
        cout<<itr->first<<" "<<itr->second<<endl;
        if(aplmax < itr->second && itr->first.compare("OTHER") != 0 && itr->first.compare("NONE")
!= 0) {
            mostUsedaplProtocol = itr->first;
            aplmax = itr->second;
        }
    }
    cout<<"Most used protocol on Application layer: "<<mostUsedaplProtocol<<endl;
    cout<<"--------------------------------------------------"<<endl;
}

/**
function to read a c array file and returns it in string format
*/
string readcFile(string name) {
     string line;
     string result = "";
     ifstream myfile (name);
    if (myfile.is_open()) {
        while (getline(myfile,line))
          result += line;
        myfile.close();
      }
     else
         cout<<"Unable to open file";
    return result;
}

/**
function to read protocols from given file with list of all protocols
format is of the form - HEX NUM NAME
*/
void readprotocolFile() {
    string line;
    string result = "";
    ifstream myfile ("listofprotocol.txt");
   if (myfile.is_open()) {
       while (getline(myfile,line)) {
           string temp;
           int i=0;
           string hex = "";
           while(line[i] != ' ' && i<line.length()) {
               hex += string(1,line[i]);
               i++;
           }
           i++;
           string num;
           while(line[i] != ' ' && i<line.length()) {
               num += string(1,line[i]);
               i++;
           }
           i++;
           string protocol;
           while(line[i] != ' ' && i<line.length()) {
               protocol += string(1,line[i]);
               i++;
           }
           transform(hex.begin(), hex.end(), hex.begin(), ::tolower);
           protocolType[hex] = protocol;
       }
       myfile.close();
     }
    else
        cout<<"Unable to open file";
}
```

```
/**
function to split packet based on ,
*/
vector<string> splitData(string packet) {
   int n = packet.length();
   vector<string> result;
   string temp;
   for(int i=0; i<n; i++) {
      if(packet[i]!=',')
         temp += string(1,packet[i]);
      else {
         result.push_back(temp);
         temp = "";
      }
   }
   if(temp!="" || temp!= " ")
      result.push_back(temp);

   return result;
}
/**
function to get mac addresses
*/
void getSourceDestMac(vector<string> fields, Packet *pkt){

   int MACsourceAddressStartIndex = 6, MACdestAddressStartIndex = 0;
   string sourceMAC =
fields[MACsourceAddressStartIndex]+fields[MACsourceAddressStartIndex+1]+fields[MACsourc
eAddressStartIndex+2]+fields[MACsourceAddressStartIndex+3]+fields[MACsourceAddressStartI
ndex+4]+fields[MACsourceAddressStartIndex+5];
   string destMAC =
fields[MACdestAddressStartIndex]+fields[MACdestAddressStartIndex+1]+fields[MACdestAddres
sStartIndex+2]+fields[MACdestAddressStartIndex+3]+fields[MACdestAddressStartIndex+4]+fiel
ds[MACdestAddressStartIndex+5];
   string sourcefinal = sourceMAC.substr(2, 2)+":"+sourceMAC.substr(6,
2)+":"+sourceMAC.substr(10, 2)+":"+sourceMAC.substr(14, 2)+":"+sourceMAC.substr(18,
2)+":"+sourceMAC.substr(22, 2);
   string destfinal = destMAC.substr(2, 2)+":"+destMAC.substr(6, 2)+":"+destMAC.substr(10,
2)+":"+destMAC.substr(14, 2)+":"+destMAC.substr(18, 2)+":"+destMAC.substr(22, 2);;
   pkt->srcMacAddress = sourcefinal;
```

```
    pkt->destMacAddress = destfinal;

}

/**
function to check whether the packet is ARP
*/
int checkARP(vector<string> fields, Packet *pkt){
    string str1 = fields[12]+fields[13];  // checking ARP on basis of 13th and 14th byte.
    string check = str1.substr(2,2) + str1.substr(6, 2);
    string compare1 = "0806";
    string compare2 = "0800";
    string compare3 = "86dd";
    if((check.compare(compare1)) == 0)
        return (1);
    else if((check.compare(compare2)) == 0)
      return (2);
    else if((check.compare(compare3)) == 0)
      return (3);
    else
        return (4);
}

/**
function to get various fields from ARP packet
*/
void ARPdetails(vector<string> fields, Packet *pkt){
    {
    int sourceAddressStartIndex = 28, destAddressStartIndex = 38;
    string src = to_string(stoi(fields[sourceAddressStartIndex], 0,
16))+"."+to_string(stoi(fields[sourceAddressStartIndex+1], 0,
16))+"."+to_string(stoi(fields[sourceAddressStartIndex+2], 0,
16))+"."+to_string(stoi(fields[sourceAddressStartIndex+3], 0, 16));
    string des = to_string(stoi(fields[destAddressStartIndex], 0,
16))+"."+to_string(stoi(fields[destAddressStartIndex+1], 0,
16))+"."+to_string(stoi(fields[destAddressStartIndex+2], 0,
16))+"."+to_string(stoi(fields[destAddressStartIndex+3], 0, 16));
    pkt->srcIPAddress = src;
    pkt->destIPAddress = des;
    pkt->srcPort = -1;
    pkt->destPort = -1;
    pkt->protocol = "ARP";
    }
}



/**
function to get SOURCE and DESTINATION ip addresses based on HEX value
*/
void getSourceDest(vector<string> fields, Packet *pkt) {
    int n = 14,version;
    string str = fields[n];
    string str2= str.substr(2,1); // for extracting IP version
    version = stoi(str2, 0, 10);
    if(version == 4){
    int sourceAddressStartIndex = 26, destAddressStartIndex = 30;
    string src = to_string(stoi(fields[sourceAddressStartIndex], 0,
16))+"."+to_string(stoi(fields[sourceAddressStartIndex+1], 0,
16))+"."+to_string(stoi(fields[sourceAddressStartIndex+2], 0,
16))+"."+to_string(stoi(fields[sourceAddressStartIndex+3], 0, 16));
    string des = to_string(stoi(fields[destAddressStartIndex], 0,
16))+"."+to_string(stoi(fields[destAddressStartIndex+1], 0,
16))+"."+to_string(stoi(fields[destAddressStartIndex+2], 0,
16))+"."+to_string(stoi(fields[destAddressStartIndex+3], 0, 16));
    pkt->srcIPAddress = src;
    pkt->destIPAddress = des;
    }
    else{
        // for IPv6 address.
    int sourceAddressStartIndex = 22,i, destAddressStartIndex = 38;
    string
strsource1,strsource2,strsource11,strdest1,strdest2,strdest11,sourcefinal="",destfinal="",comp
arestring="ff";
    strsource1 = fields[sourceAddressStartIndex];
    strsource2 = fields[sourceAddressStartIndex+1];
    strsource11 = strsource1.substr(2, 2)+strsource2.substr(2, 2);
    sourcefinal = sourcefinal+strsource11;
    strdest1 = fields[destAddressStartIndex];
    strdest2 = fields[destAddressStartIndex+1];
    strdest11 = strdest1.substr(2, 2)+strdest2.substr(2, 2);
    destfinal = destfinal+strdest11;
    for(i=2;i<=14;i=i+2)
    {
    strsource1 = fields[sourceAddressStartIndex+i];
    strsource2 = fields[sourceAddressStartIndex+(i+1)];
    strsource11 = strsource1.substr(2, 2)+strsource2.substr(2, 2);
    sourcefinal=sourcefinal+":"+strsource11;
    strdest1 = fields[destAddressStartIndex+i];
    strdest2 = fields[destAddressStartIndex+(i+1)];
    strdest11 = strdest1.substr(2, 2)+strdest2.substr(2, 2);
    destfinal=destfinal+":"+strdest11;
    strsource1="", strsource11 ="";
    strdest1="" ,strdest11="";
    }
    pkt->srcIPAddress = sourcefinal;
    pkt->destIPAddress = destfinal;
    }

}

/**
function to identify application layer protocol based on port number
*/
void getApplicationLayerProtocol(int sourceportnum, int destportnum, Packet *pkt) {

    string sport = to_string(sourceportnum);
    string dport = to_string(destportnum);

    if(aplProtocolType.count(sport) || aplProtocolType.count(dport)) {
        if(aplProtocolType.count(sport)) {
            pkt->aplProtocol = aplProtocolType[sport];
        }
        else {
            pkt->aplProtocol = aplProtocolType[dport];
        }
    }
    else{
        pkt->aplProtocol = "OTHER";
    }
}

/**
```

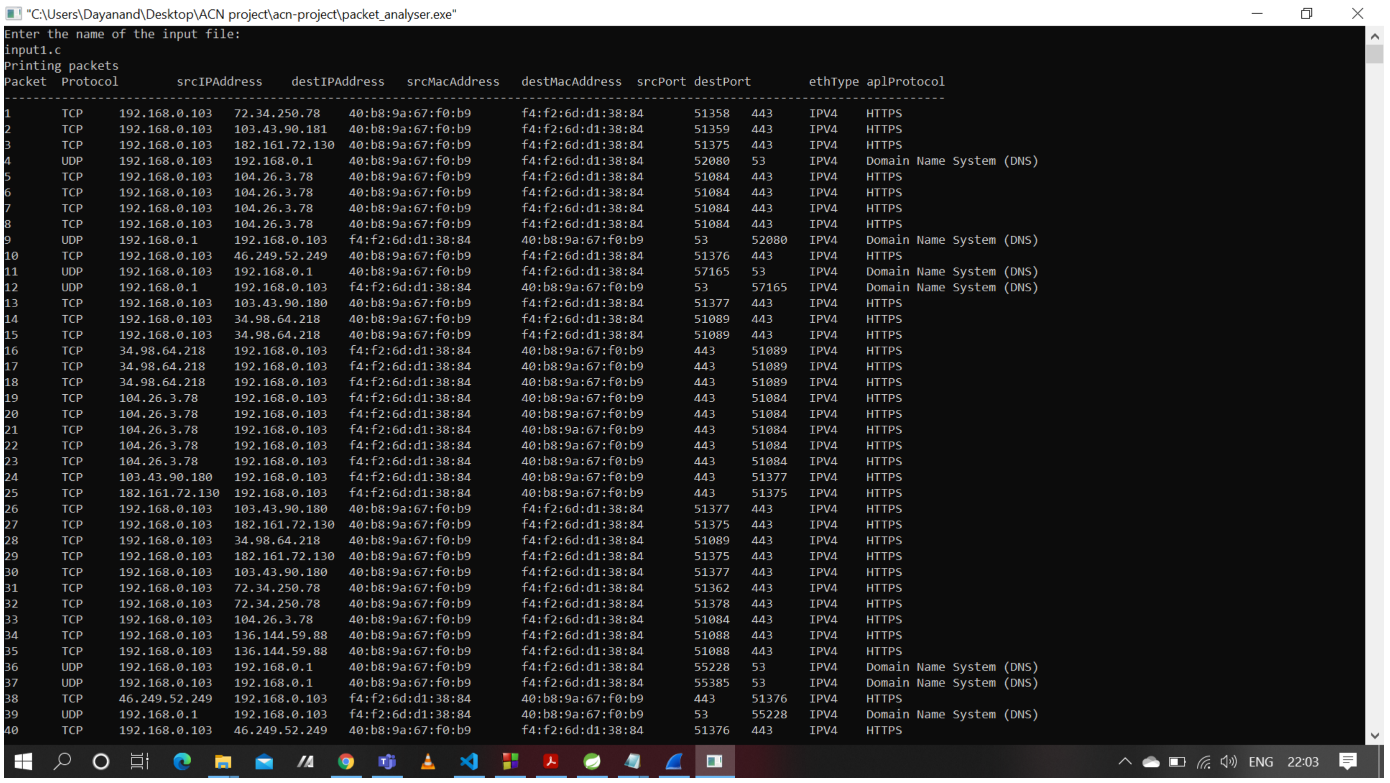
```
function to get source and destination ports
*/
void getPorts(vector<string> fields, Packet *pkt){
   int n = 14,m; // used for storing location of header version and header length
   int version, headerlength,sourceportnumber,destportnumber;
   string sourceport="",destport="";
   string str = fields[n];
   string str2= str.substr(2,1); // for extracting IP version
   string str3= str.substr(3,1);
   version = stoi(str2, 0, 10);
   headerlength = stoi(str3, 0, 10);
   if(version == 4)
   {
      headerlength*=4;
      m= n+headerlength; // for storing source port index.
      sourceport+= fields[m]+ fields[m+1];
      sourceport = sourceport.substr(2,2) + sourceport.substr(6,2);
      sourceportnumber = stoi(sourceport, 0, 16);
      destport+=fields[m+2] + fields[m+3];
      destport = destport.substr(2,2) + destport.substr(6,2);
      destportnumber = stoi(destport, 0, 16);
      pkt->srcPort = sourceportnumber;
      pkt->destPort = destportnumber;
      getApplicationLayerProtocol(sourceportnumber,destportnumber,pkt);
      sourceport=""; destport="";
   }
   else
   {
      m = n+ 40;
      sourceport+= fields[m]+ fields[m+1];
      sourceport = sourceport.substr(2,2) + sourceport.substr(6,2);
      sourceportnumber = stoi(sourceport, 0, 16);
      destport+=fields[m+2] + fields[m+3];
      destport = destport.substr(2,2) + destport.substr(6,2);
      destportnumber = stoi(destport, 0, 16);
      pkt->srcPort = sourceportnumber;
      pkt->destPort = destportnumber;
      getApplicationLayerProtocol(sourceportnumber,destportnumber,pkt);
      sourceport=""; destport="";
   }
}
```

```
/**
function to get NAME OF THE PROTOCOL based on HEX value
*/
void getProtocol(vector<string> fields, Packet* pkt) {
   int location = 23;
   string protocol = "";
   if(protocolType.count(fields[location])){
      protocol = protocolType[fields[location]];
   }
   else{
      protocol = "Unassigned/Experimental";
   }
   pkt->protocol = protocol;

}

/* get SIZE of the packet in bytes */
void getPacketSize(vector<string> fields) {
   cout<<"Packet Size: "<<fields.size()<<" bytes"<<endl;
}

/**
Main function to analyse fields values
*/
void analysefields(vector<string> fields, int packetCount, Packet *pkt, int *print) {

   int check;   // for checking ARP packet.
   /* Implement functions here */

   check = checkARP(fields, pkt);
   *print=1;
   if(check == 1)
   {
      pkt->ethType = "ARP";
      pkt->aplProtocol = "NONE";
      getSourceDestMac(fields, pkt);
      ARPdetails(fields, pkt);
   }
   else if(check == 2){
      pkt->ethType = "IPV4";
      getSourceDestMac(fields, pkt);
      getSourceDest(fields, pkt);
      getProtocol(fields, pkt);
      getPorts(fields, pkt);
   }
   else if(check == 3){
      pkt->ethType = "IPV6";
      getSourceDestMac(fields, pkt);
      getSourceDest(fields, pkt);
      // ipv6 has different offset
      int location = 20;
      string protocol = "";
   if(protocolType.count(fields[location])){
      protocol = protocolType[fields[location]];
   }

   else{

      protocol = "Unassigned/Experimental";
   }

   pkt->protocol = protocol;
   getPorts(fields, pkt);
   }
   else
   {
      pkt->ethType = "OTHER";
      getPacketSize(fields);
      *print=0;
   }

}

/**
function to divide c array data into packets
*/
void getPackets(string data) {
   vector<Packet> packets;
   int packetCount = 0;
   string packet = "";
```

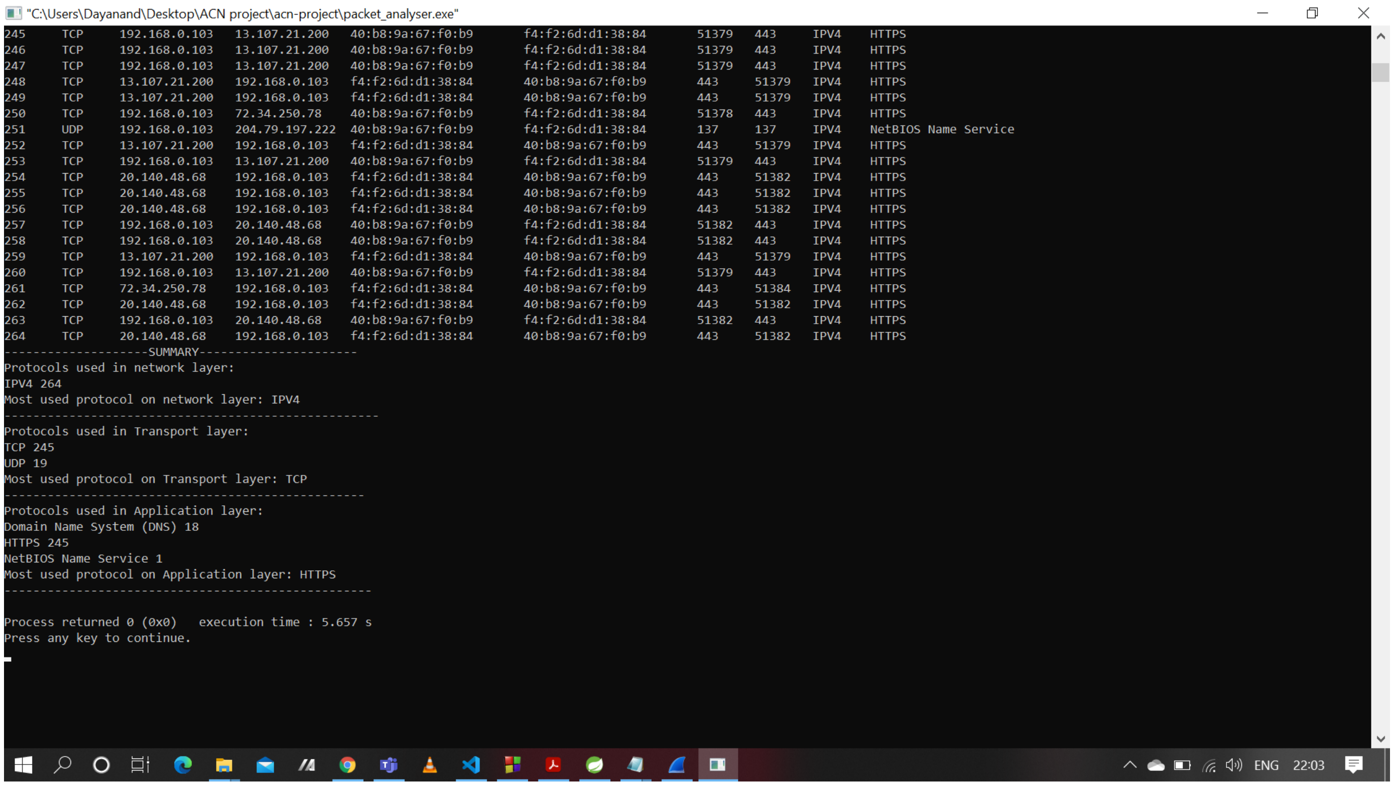
```
bool ignorePacket = false;
for(int i=0; i<data.length(); i++) {
   char curr = data[i];

   if(i+11<data.length() && data.substr(i,11).compare("Reassembled")==0) {
      ignorePacket = true;
   }
   if(curr=='{') {
      if(!ignorePacket)
         packetCount++;
      char next = '{';
      while(next != '}') {
         i++;
         next = data[i];
         if(next != ' ') {
            // remove comment from c array file
            if(data[i] == '/' && data[i+1]=='*') {
               i+=2;
               int valid = 1;
               while(valid) {
                  i++;
                  if(data[i] == '*' && data[i+1] == '/')
                     valid = 0;
               }
               i++;
            }
            else if(next != '}' && next != ' ' && data[i] != '\n' && data[i] != '\r')
               packet += next;
         }
      }
      vector<string> fields = splitData(packet); // split data based on ,
      packet = "";

      if(!ignorePacket) {
         // analyse each packet
         Packet pkt;
         int print; // print is 1 when ethertype is ipv4,ipv6 or arp otherwise 0
         analysefields(fields,packetCount,&pkt, &print);
         if(print==1)
            packets.push_back(pkt);
         else
```

```
                packetCount--;
            }
            else
               ignorePacket = false;
        }
    }
    printPackets(packets);
    printSummary(packets);
}

void readApplicationLayerProtocolList() {
    string line;
    ifstream myfile("service-names-port-numbers.txt");
    if(myfile.is_open()) {
        while(getline(myfile, line)) {
            vector<string> result = splitData(line);
            aplProtocolType[result[0]] = result[1];
        }
        myfile.close();
    }
    else
        cout<<"unable to open file"<<endl;
}

int main () {
    cout<<"Enter the name of the input file: "<<endl;
    string temp;
    cin>>temp;
    readprotocolFile(); // read a list of protocol hex and their name
    readApplicationLayerProtocolList();
    string data = readcFile(temp); // read c array file
    getPackets(data); // divide data into packets
    return 0;
}
```

Output Screenshots

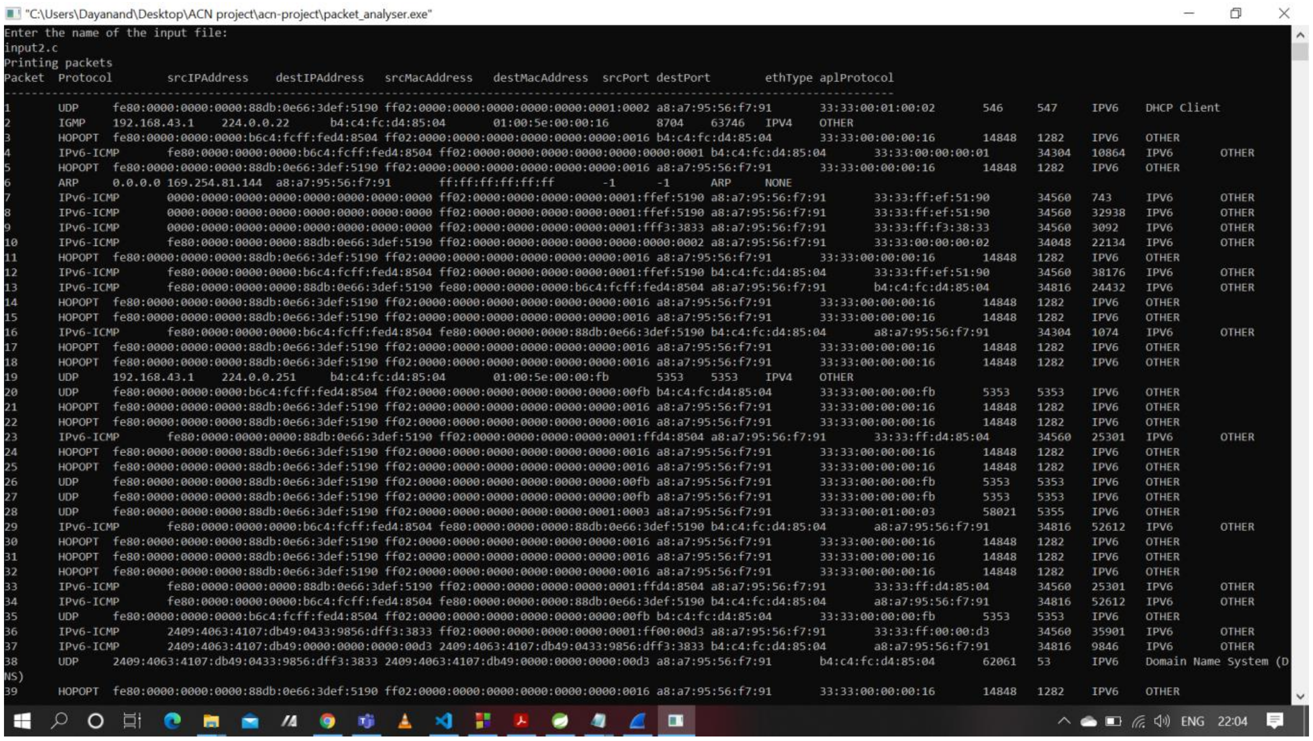


```
"C:\Users\Dayanand\Desktop\ACN project\acn-project\packet_analyser.exe"
Enter the name of the input file:
input1.c
Printing packets
Packet  Protocol        srcIPAddress    destIPAddress   srcMacAddress   destMacAddress  srcPort destPort        ethType aplProtocol
-----------------------------------------------------------------------------------------------------------------------------------
1       TCP     192.168.0.103   72.34.250.78    40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51358   443     IPV4    HTTPS
2       TCP     192.168.0.103   103.43.90.181   40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51359   443     IPV4    HTTPS
3       TCP     192.168.0.103   182.161.72.130  40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51375   443     IPV4    HTTPS
4       UDP     192.168.0.103   192.168.0.1     40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       52080   53      IPV4    Domain Name System (DNS)
5       TCP     192.168.0.103   104.26.3.78     40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51084   443     IPV4    HTTPS
6       TCP     192.168.0.103   104.26.3.78     40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51084   443     IPV4    HTTPS
7       TCP     192.168.0.103   104.26.3.78     40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51084   443     IPV4    HTTPS
8       TCP     192.168.0.103   104.26.3.78     40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51084   443     IPV4    HTTPS
9       UDP     192.168.0.1     192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       53      52080   IPV4    Domain Name System (DNS)
10      TCP     192.168.0.103   46.249.52.249   40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51376   443     IPV4    HTTPS
11      UDP     192.168.0.103   192.168.0.1     40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       57165   53      IPV4    Domain Name System (DNS)
12      UDP     192.168.0.1     192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       53      57165   IPV4    Domain Name System (DNS)
13      TCP     192.168.0.103   103.43.90.180   40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51377   443     IPV4    HTTPS
14      TCP     192.168.0.103   34.98.64.218    40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51089   443     IPV4    HTTPS
15      TCP     192.168.0.103   34.98.64.218    40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51089   443     IPV4    HTTPS
16      TCP     34.98.64.218    192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51089   IPV4    HTTPS
17      TCP     34.98.64.218    192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51089   IPV4    HTTPS
18      TCP     34.98.64.218    192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51089   IPV4    HTTPS
19      TCP     104.26.3.78     192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51084   IPV4    HTTPS
20      TCP     104.26.3.78     192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51084   IPV4    HTTPS
21      TCP     104.26.3.78     192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51084   IPV4    HTTPS
22      TCP     104.26.3.78     192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51084   IPV4    HTTPS
23      TCP     104.26.3.78     192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51084   IPV4    HTTPS
24      TCP     103.43.90.180   192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51377   IPV4    HTTPS
25      TCP     182.161.72.130  192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51375   IPV4    HTTPS
26      TCP     192.168.0.103   103.43.90.180   40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51377   443     IPV4    HTTPS
27      TCP     192.168.0.103   182.161.72.130  40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51375   443     IPV4    HTTPS
28      TCP     192.168.0.103   34.98.64.218    40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51089   443     IPV4    HTTPS
29      TCP     192.168.0.103   182.161.72.130  40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51375   443     IPV4    HTTPS
30      TCP     192.168.0.103   103.43.90.180   40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51377   443     IPV4    HTTPS
31      TCP     192.168.0.103   72.34.250.78    40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51362   443     IPV4    HTTPS
32      TCP     192.168.0.103   72.34.250.78    40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51378   443     IPV4    HTTPS
33      TCP     192.168.0.103   104.26.3.78     40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51084   443     IPV4    HTTPS
34      TCP     192.168.0.103   136.144.59.88   40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51088   443     IPV4    HTTPS
35      TCP     192.168.0.103   136.144.59.88   40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51088   443     IPV4    HTTPS
36      UDP     192.168.0.103   192.168.0.1     40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       55228   53      IPV4    Domain Name System (DNS)
37      UDP     192.168.0.103   192.168.0.1     40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       55385   53      IPV4    Domain Name System (DNS)
38      TCP     46.249.52.249   192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51376   IPV4    HTTPS
39      UDP     192.168.0.1     192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       53      55228   IPV4    Domain Name System (DNS)
40      TCP     192.168.0.103   46.249.52.249   40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51376   443     IPV4    HTTPS
```



```
"C:\Users\Dayanand\Desktop\ACN project\acn-project\packet_analyser.exe"
245     TCP     192.168.0.103   13.107.21.200   40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51379   443     IPV4    HTTPS
246     TCP     192.168.0.103   13.107.21.200   40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51379   443     IPV4    HTTPS
247     TCP     192.168.0.103   13.107.21.200   40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51379   443     IPV4    HTTPS
248     TCP     13.107.21.200   192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51379   IPV4    HTTPS
249     TCP     13.107.21.200   192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51379   IPV4    HTTPS
250     TCP     192.168.0.103   72.34.250.78    40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51378   443     IPV4    HTTPS
251     UDP     192.168.0.103   204.79.197.222  40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       137     137     IPV4    NetBIOS Name Service
252     TCP     13.107.21.200   192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51379   IPV4    HTTPS
253     TCP     192.168.0.103   13.107.21.200   40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51379   443     IPV4    HTTPS
254     TCP     20.140.48.68    192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51382   IPV4    HTTPS
255     TCP     20.140.48.68    192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51382   IPV4    HTTPS
256     TCP     20.140.48.68    192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51382   IPV4    HTTPS
257     TCP     192.168.0.103   20.140.48.68    40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51382   443     IPV4    HTTPS
258     TCP     192.168.0.103   20.140.48.68    40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51382   443     IPV4    HTTPS
259     TCP     13.107.21.200   192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51379   IPV4    HTTPS
260     TCP     192.168.0.103   13.107.21.200   40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51379   443     IPV4    HTTPS
261     TCP     72.34.250.78    192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51384   IPV4    HTTPS
262     TCP     20.140.48.68    192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51382   IPV4    HTTPS
263     TCP     192.168.0.103   20.140.48.68    40:b8:9a:67:f0:b9       f4:f2:6d:d1:38:84       51382   443     IPV4    HTTPS
264     TCP     20.140.48.68    192.168.0.103   f4:f2:6d:d1:38:84       40:b8:9a:67:f0:b9       443     51382   IPV4    HTTPS
--------------------SUMMARY---------------------
Protocols used in network layer:
IPV4 264
Most used protocol on network layer: IPV4
---------------------------------------------------
Protocols used in Transport layer:
TCP 245
UDP 19
Most used protocol on Transport layer: TCP
-------------------------------------------------
Protocols used in Application layer:
Domain Name System (DNS) 18
HTTPS 245
NetBIOS Name Service 1
Most used protocol on Application layer: HTTPS
---------------------------------------------------

Process returned 0 (0x0)   execution time : 5.657 s
Press any key to continue.
```

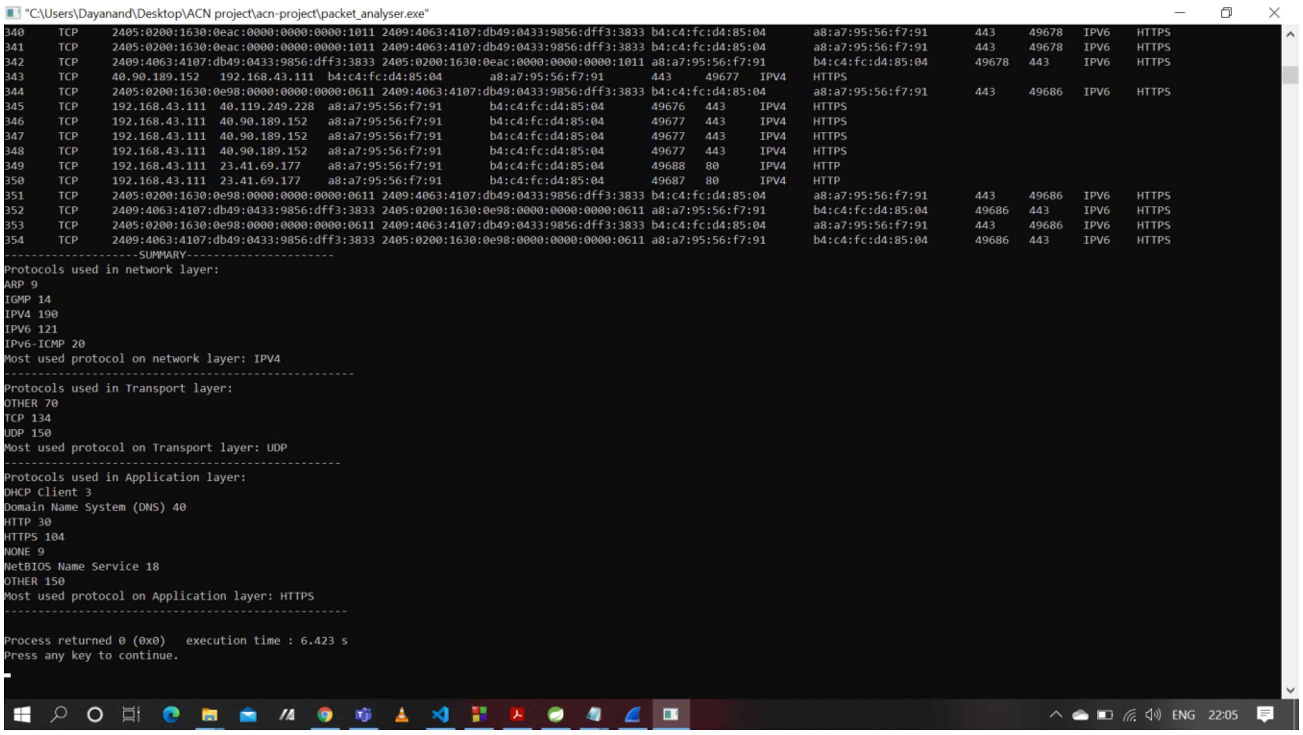
Output for input1.c



"C:\Users\Dayanand\Desktop\ACN project\acn-project\packet_analyser.exe"

```
Enter the name of the input file:
input2.c
Printing packets
Packet  Protocol        srcIPAddress    destIPAddress   srcMacAddress   destMacAddress  srcPort destPort        ethType aplProtocol
------------------------------------------------------------------------------------------------------------------------------------
1       UDP     fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0000:0001:0002 a8:a7:95:56:f7:91       33:33:00:01:00:02       546     547     IPV6    DHCP Client
2       IGMP    192.168.43.1    224.0.0.22      b4:c4:fc:d4:85:04       01:00:5e:00:00:16       8704    63746   IPV4    OTHER
3       HOPOPT  fe80:0000:0000:0000:b6c4:fcff:fed4:8504 ff02:0000:0000:0000:0000:0000:0000:0016 b4:c4:fc:d4:85:04       33:33:00:00:00:16       14848   1282    IPV6    OTHER
4       IPv6-ICMP       fe80:0000:0000:0000:b6c4:fcff:fed4:8504 ff02:0000:0000:0000:0000:0000:0000:0001 b4:c4:fc:d4:85:04       33:33:00:00:00:01       34304   10864   IPV6    OTHER
5       HOPOPT  fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0000:0000:0016 a8:a7:95:56:f7:91       33:33:00:00:00:16       14848   1282    IPV6    OTHER
6       ARP     0.0.0.0 169.254.81.144  a8:a7:95:56:f7:91       ff:ff:ff:ff:ff:ff       -1      -1      ARP     NONE
7       IPv6-ICMP       0000:0000:0000:0000:0000:0000:0000:0000 ff02:0000:0000:0000:0000:0001:ffef:5190 a8:a7:95:56:f7:91       33:33:ff:ef:51:90       34560   743     IPV6    OTHER
8       IPv6-ICMP       0000:0000:0000:0000:0000:0000:0000:0000 ff02:0000:0000:0000:0000:0001:ffef:5190 a8:a7:95:56:f7:91       33:33:ff:ef:51:90       34560   32938   IPV6    OTHER
9       IPv6-ICMP       0000:0000:0000:0000:0000:0000:0000:0000 ff02:0000:0000:0000:0000:0001:fff3:3833 a8:a7:95:56:f7:91       33:33:ff:f3:38:33       34560   3092    IPV6    OTHER
10      IPv6-ICMP       fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0000:0000:0002 a8:a7:95:56:f7:91       33:33:00:00:00:02       34048   22134   IPV6    OTHER
11      HOPOPT  fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0000:0000:0016 a8:a7:95:56:f7:91       33:33:00:00:00:16       14848   1282    IPV6    OTHER
12      IPv6-ICMP       fe80:0000:0000:0000:b6c4:fcff:fed4:8504 ff02:0000:0000:0000:0000:0001:ffef:5190 b4:c4:fc:d4:85:04       33:33:ff:ef:51:90       34560   38176   IPV6    OTHER
13      IPv6-ICMP       fe80:0000:0000:0000:88db:0e66:3def:5190 fe80:0000:0000:0000:b6c4:fcff:fed4:8504 a8:a7:95:56:f7:91       b4:c4:fc:d4:85:04       34816   24432   IPV6    OTHER
14      HOPOPT  fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0000:0000:0016 a8:a7:95:56:f7:91       33:33:00:00:00:16       14848   1282    IPV6    OTHER
15      HOPOPT  fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0000:0000:0016 a8:a7:95:56:f7:91       33:33:00:00:00:16       14848   1282    IPV6    OTHER
16      IPv6-ICMP       fe80:0000:0000:0000:b6c4:fcff:fed4:8504 fe80:0000:0000:0000:88db:0e66:3def:5190 b4:c4:fc:d4:85:04       a8:a7:95:56:f7:91       34304   1074    IPV6    OTHER
17      HOPOPT  fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0000:0000:0016 a8:a7:95:56:f7:91       33:33:00:00:00:16       14848   1282    IPV6    OTHER
18      HOPOPT  fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0000:0000:0016 a8:a7:95:56:f7:91       33:33:00:00:00:16       14848   1282    IPV6    OTHER
19      UDP     192.168.43.1    224.0.0.251     b4:c4:fc:d4:85:04       01:00:5e:00:00:fb       5353    5353    IPV4    OTHER
20      UDP     fe80:0000:0000:0000:b6c4:fcff:fed4:8504 ff02:0000:0000:0000:0000:0000:0000:00fb b4:c4:fc:d4:85:04       33:33:00:00:00:fb       5353    5353    IPV6    OTHER
21      HOPOPT  fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0000:0000:0016 a8:a7:95:56:f7:91       33:33:00:00:00:16       14848   1282    IPV6    OTHER
22      HOPOPT  fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0000:0000:0016 a8:a7:95:56:f7:91       33:33:00:00:00:16       14848   1282    IPV6    OTHER
23      IPv6-ICMP       fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0001:ffd4:8504 a8:a7:95:56:f7:91       33:33:ff:d4:85:04       34560   25301   IPV6    OTHER
24      HOPOPT  fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0000:0000:0016 a8:a7:95:56:f7:91       33:33:00:00:00:16       14848   1282    IPV6    OTHER
25      HOPOPT  fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0000:0000:0016 a8:a7:95:56:f7:91       33:33:00:00:00:16       14848   1282    IPV6    OTHER
26      UDP     fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0000:0000:00fb a8:a7:95:56:f7:91       33:33:00:00:00:fb       5353    5353    IPV6    OTHER
27      UDP     fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0000:0000:00fb a8:a7:95:56:f7:91       33:33:00:00:00:fb       5353    5353    IPV6    OTHER
28      UDP     fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0000:0001:0003 a8:a7:95:56:f7:91       33:33:00:01:00:03       58021   5355    IPV6    OTHER
29      IPv6-ICMP       fe80:0000:0000:0000:b6c4:fcff:fed4:8504 fe80:0000:0000:0000:88db:0e66:3def:5190 b4:c4:fc:d4:85:04       a8:a7:95:56:f7:91       34816   52612   IPV6    OTHER
30      HOPOPT  fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0000:0000:0016 a8:a7:95:56:f7:91       33:33:00:00:00:16       14848   1282    IPV6    OTHER
31      HOPOPT  fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0000:0000:0016 a8:a7:95:56:f7:91       33:33:00:00:00:16       14848   1282    IPV6    OTHER
32      HOPOPT  fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0000:0000:0016 a8:a7:95:56:f7:91       33:33:00:00:00:16       14848   1282    IPV6    OTHER
33      IPv6-ICMP       fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0001:ffd4:8504 a8:a7:95:56:f7:91       33:33:ff:d4:85:04       34560   25301   IPV6    OTHER
34      IPv6-ICMP       fe80:0000:0000:0000:b6c4:fcff:fed4:8504 fe80:0000:0000:0000:88db:0e66:3def:5190 b4:c4:fc:d4:85:04       a8:a7:95:56:f7:91       34816   52612   IPV6    OTHER
35      UDP     fe80:0000:0000:0000:b6c4:fcff:fed4:8504 ff02:0000:0000:0000:0000:0000:0000:00fb b4:c4:fc:d4:85:04       33:33:00:00:00:fb       5353    5353    IPV6    OTHER
36      IPv6-ICMP       2409:4063:4107:db49:0433:9856:dff3:3833 ff02:0000:0000:0000:0000:0001:ff00:00d3 a8:a7:95:56:f7:91       33:33:ff:00:00:d3       34560   35901   IPV6    OTHER
37      IPv6-ICMP       2409:4063:4107:db49:0000:0000:0000:00d3 2409:4063:4107:db49:0433:9856:dff3:3833 b4:c4:fc:d4:85:04       a8:a7:95:56:f7:91       34816   9846    IPV6    OTHER
38      UDP     2409:4063:4107:db49:0433:9856:dff3:3833 2409:4063:4107:db49:0000:0000:0000:00d3 a8:a7:95:56:f7:91       b4:c4:fc:d4:85:04       62061   53      IPV6    Domain Name System (D
NS)
39      HOPOPT  fe80:0000:0000:0000:88db:0e66:3def:5190 ff02:0000:0000:0000:0000:0000:0000:0016 a8:a7:95:56:f7:91       33:33:00:00:00:16       14848   1282    IPV6    OTHER
```

ENG 22:04



"C:\Users\Dayanand\Desktop\ACN project\acn-project\packet_analyser.exe"

```
340     TCP     2405:0200:1630:0eac:0000:0000:0000:1011 2409:4063:4107:db49:0433:9856:dff3:3833 b4:c4:fc:d4:85:04       a8:a7:95:56:f7:91       443     49678   IPV6    HTTPS
341     TCP     2405:0200:1630:0eac:0000:0000:0000:1011 2409:4063:4107:db49:0433:9856:dff3:3833 b4:c4:fc:d4:85:04       a8:a7:95:56:f7:91       443     49678   IPV6    HTTPS
342     TCP     2409:4063:4107:db49:0433:9856:dff3:3833 2405:0200:1630:0eac:0000:0000:0000:1011 a8:a7:95:56:f7:91       b4:c4:fc:d4:85:04       49678   443     IPV6    HTTPS
343     TCP     40.90.189.152   192.168.43.111  b4:c4:fc:d4:85:04       a8:a7:95:56:f7:91       443     49677   IPV4    HTTPS
344     TCP     2405:0200:1630:0e98:0000:0000:0000:0611 2409:4063:4107:db49:0433:9856:dff3:3833 b4:c4:fc:d4:85:04       a8:a7:95:56:f7:91       443     49686   IPV6    HTTPS
345     TCP     192.168.43.111  40.119.249.228  a8:a7:95:56:f7:91       b4:c4:fc:d4:85:04       49676   443     IPV4    HTTPS
346     TCP     192.168.43.111  40.90.189.152   a8:a7:95:56:f7:91       b4:c4:fc:d4:85:04       49677   443     IPV4    HTTPS
347     TCP     192.168.43.111  40.90.189.152   a8:a7:95:56:f7:91       b4:c4:fc:d4:85:04       49677   443     IPV4    HTTPS
348     TCP     192.168.43.111  40.90.189.152   a8:a7:95:56:f7:91       b4:c4:fc:d4:85:04       49677   443     IPV4    HTTPS
349     TCP     192.168.43.111  23.41.69.177    a8:a7:95:56:f7:91       b4:c4:fc:d4:85:04       49688   80      IPV4    HTTP
350     TCP     192.168.43.111  23.41.69.177    a8:a7:95:56:f7:91       b4:c4:fc:d4:85:04       49687   80      IPV4    HTTP
351     TCP     2405:0200:1630:0e98:0000:0000:0000:0611 2409:4063:4107:db49:0433:9856:dff3:3833 b4:c4:fc:d4:85:04       a8:a7:95:56:f7:91       443     49686   IPV6    HTTPS
352     TCP     2409:4063:4107:db49:0433:9856:dff3:3833 2405:0200:1630:0e98:0000:0000:0000:0611 a8:a7:95:56:f7:91       b4:c4:fc:d4:85:04       49686   443     IPV6    HTTPS
353     TCP     2405:0200:1630:0e98:0000:0000:0000:0611 2409:4063:4107:db49:0433:9856:dff3:3833 b4:c4:fc:d4:85:04       a8:a7:95:56:f7:91       443     49686   IPV6    HTTPS
354     TCP     2409:4063:4107:db49:0433:9856:dff3:3833 2405:0200:1630:0e98:0000:0000:0000:0611 a8:a7:95:56:f7:91       b4:c4:fc:d4:85:04       49686   443     IPV6    HTTPS
--------------------SUMMARY---------------------
Protocols used in network layer:
ARP 9
IGMP 14
IPV4 190
IPV6 121
IPv6-ICMP 20
Most used protocol on network layer: IPV4
-----------------------------------------------
Protocols used in Transport layer:
OTHER 70
TCP 134
UDP 150
Most used protocol on Transport layer: UDP
-----------------------------------------------
Protocols used in Application layer:
DHCP Client 3
Domain Name System (DNS) 40
HTTP 30
HTTPS 104
NONE 9
NetBIOS Name Service 18
OTHER 150
Most used protocol on Application layer: HTTPS
-----------------------------------------------

Process returned 0 (0x0)   execution time : 6.423 s
Press any key to continue.
```

ENG 22:05

Output for input2.c