SPL-1 Project Report

HOSTMAN: A TCP/IP Packet Analysis Software

Course: SE 305: Software Project Lab I

Submitted by

Rafed Muhammad Yasir

BSSE Roll No. : 733

BSSE Session: 2014-2015

Supervised by

Dr B M Mainul Hossain

Assistant Professor

Institute of Information Technology

University of Dhaka

Submitted to

Rezvi Shahariar

Assistant Professor

Institute of Information Technology

University of Dhaka

Institute of Information Technology

University of Dhaka

01-05-2016

Acknowledgement

At first I would like to thank almighty for helping me complete this project.

I would like to express my deepest gratitude to all those who provided me the support and encouragement to complete this project. Special thanks to my project supervisor Dr. B M Mainul Hossain, whose continuous suggestions and guidance has been invaluable to me. Without such stimulating suggestions the project could not have made this much progress.

I would also like to thank my teacher, Professor Mohammad Zulfiquar Hafiz, whose classes on networking has been very informative and helpful for working on this project.

I am grateful to the Institute of Information Technology for giving me the opportunity to do such a project.

Lastly I would like to thank my classmates. Discussing and sharing ideas with them provided me valuable insights from time to time.

Sincerely,
Rafed Muhammad Yasir
BSSE 0733

Abstract

‘Hostman’ is a simple TCP/IP packet analyzing software that can analyze captured network packets with the most common protocols of the TCP/IP suite. This report discusses in detail how Hostman is designed, how it has been implemented and how it works. The report also discusses the basic concepts needed behind building a packet analyzer and their importance.

Table of Contents

1. Introduction 1
1.1. Broad Domain 1
1.2. Project Scope 2
1.3. Challenges 3
1.4. Software Requirements 3
1.5. Methodology 4
1.6. Achievements 5
2. Background Study 6
3. Analysis and Design 7
4. Implementation 8
4.1. Header files 8
4.1.1. packetHeaders.h 8
4.1.2. cmdOptions.h 9
4.1.3. packetBasics.h 9
4.1.4. miscFunctions.h 10
4.2. C Source files 10
4.2.1. main.c 10
4.2.2. miscFunctions.c 10
4.2.3. packetBasicInfo.c 11
4.2.4. packetDetails.c 11
4.2.5. packetDump.c 11
4.2.6. packetSearch.c 11
4.2.7. printUsage.c 12
4.2.8. followStream.c 12
5. User Manual 13
6. Program Outputs 14
6.1. Print usage 14
6.2. Print basic information 14
6.3. Print packet dump 15
6.4. Print all packet details 16

6.5. Print a particular packet details 17

6.6. Print packet details in range 17

6.7. Search packet by IP 17

6.8. Search packet by text 18

6.9. Follow packet stream 18

7. Conclusion 19

Reference 20

List of Figures

Figure 1: PCAP file format 4
Figure 2: Working pattern of Hostman 5
Figure 3: Design pattern of Hostman 7
Figure 4: Output of usage 14
Figure 5: Output of packet basics 14
Figure 6: Output of packet dump 15
Figure 7: Output of packet details 16
Figure 8: Output of particular packet details 17
Figure 9: Output of search packet by IP 17
Figure 10: Output of search packet by text 18
Figure 11 Follow packet stream 18

1. Introduction

The main aim of this project is to make a simple packet analyzer called the Hostman. The term Hostman comes from the two words 'host' and 'postman'. Just like a postman reads data associated with a post or a mail, the Hostman reads information associated with a network host. A network host is a computer or other device (also known as a network node) connected to a computer network.

There are many professionally written packet analyzers available in the internet such as Wireshark, tshark, ettercap and tcpdump. Most of the professionally written analyzers are GUI based. So they cannot be used in low level systems. And the ones that are CLI based are more of 'packet capturers' and less of 'analyzers'. So there are not many suitable tools that has the sore capability of analyzing packets in CLI environment. Hostman is made to fulfill this purpose. It is a console based tool made for UNIX systems to analyze captured network packets from a pcap file.

1.1. Broad Domain

The internet is a big network of interconnected computers. Everything that we do in the internet involves packets. A packet is simply a unit of data that is routed between an origin and a destination on the internet. Every web page that we receive comes as a series of packets and every e-mail we send leaves as a series of packets.

When a file is to be sent across a network it is broken down into several packets to an efficient size for routing. The number of packets that are formed and the way they are formed depends on the protocol that is used. A packet formed by breaking down not only contains the data requested by the user but also some other information which are added by the protocol such as the address of the source computer and the destination computer. The individual packets are then routed across the network. Each packet may take different routes. When they are all arrived, they are reassembled into the original file.

A packet analyzer is a networking tool that can decode the content of a packet and see its data. It analyzes the network traffic and generates a customized report to assist organizations in managing their networks. A packet analyzers can be used to do things such as:

1. Analyzing network issues and problems
2. Monitoring network security by detecting unauthorized attempts to hack the network

3. Monitoring bandwidth
4. Filtering unwanted contents
5. Preventing unauthorized access

1.2. Project Scope

Hostman is targeted to analyze packets of the TCP/IP protocol. The TCP/IP suite itself contains over hundreds of protocols. That is why it requires a lot of time investment to make a packet analyzer that supports all the protocols. Hostman is designed to understand only the most common protocols that are used. The software can understand details of the following protocols:

- Data link layer:
    - Ethernet frame
- Network layer
    - Internet protocol (IPv4)
    - Address resolution protocol (ARP)
- Transport layer:
    - Transmission control protocol (TCP)
    - User datagram protocol (UDP)
- Application layer:
    - Hypertext transmission protocol (HTTP)
    - Hypertext transmission protocol secured (HTTPS)

Apart from these, there are some other protocols that Hostman can identify but not understand the details, such as:

- Network layer:
    - Internet Protocol (IPv6)
    - Reverse address resolution protocol (RARP)
- Transport layer:

  - Internet control message protocol (ICMP)
  - Internet group management protocol (IGMP)
- Application layer:
  - File transfer protocol (FTP)
  - Secure shell (SSH)
  - Telnet
  - Simple mail transfer protocol (SMTP)
  - Domain name system (DNS)
  - Border gateway protocol (BGP)
  - Dynamic host configuration protocol (DHCP)

The scope of Hostman is limited within these protocols.

1.3. Challenges

The key challenges of this project are:

1. Parsing packet: In each layer of a packet data are stored in the format defined by a particular protocol. Parsing packets according to different protocols is challenging because while handling large files even if a single byte of information is misread then the whole output will be wrong.
2. Working with bits: The lowest size of data we can work with in most machines is a single byte. However packets contains information that are stored in bits. These information are read using special techniques.
3. Handling endianness: Endianness refers to the order of bytes. Most modern computers follow the little endian format i.e. least significant bytes are stored first. However data communication over the internet occurs in big endian format, most significant bytes are transferred and stored first. So when the packets are to be interpreted it should be ensured that they are being read in the correct order.

1.4. Software Requirements

The software requirements are as follows:

1. View of hex dumps of the packets along with their corresponding printable characters.

2. Short description of all the packets in the capture file– source address, destination address, packet size and used protocol.

3. Detailed description of the packets in the capture file. The user will have the option to choose the packets which he wants to see.

4. A search option that can search in the packets for

   - IP addresses

   - Strings (in the application layer)

   Search results will show short descriptions of the packets if they are found.

5. An option that can see the conversation between two hosts separately.

1.5. Methodology

Hostman works on pcap (packet capture) files. The structure of a pcap file is as follows:

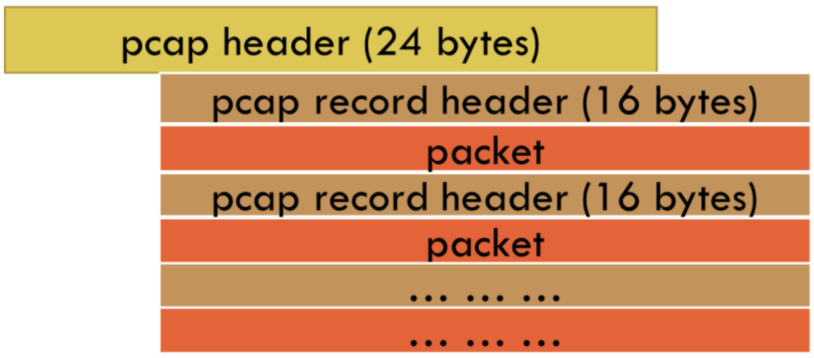


Figure 1: PCAP file format

At the start of the file there is a 24 byte pcap header. After that there is a series of pcap record headers and packet information which continues until the end of file.

The working structure oh Hostman is based upon this file format. The program has to read byte by byte according to the structure of the file. A basic flow diagram how each module of Hostman works is shown below:

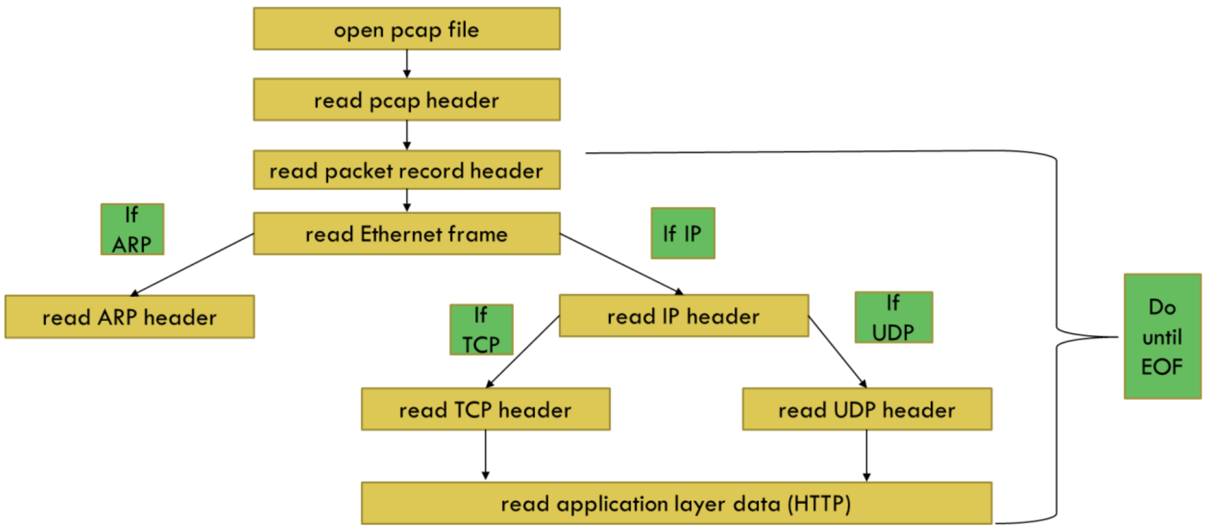


Figure 2: Working pattern of Hostman

1.6. Achievements

There were many achievements of this project:

- A deeper understanding of how the internet works
- Gaining basic networking skills
- Learning how to analyze packets
- Learning new coding techniques
- Leaning how to do a software project

2. Background Study

There are several basic networking concepts that are required to fully understand the project. Also some concepts of memory management in machines is required to understand some techniques.

OSI model: The Open Systems Interconnection (OSI) model is a standard "reference model" created by the International Organization for Standardization (ISO) to describe how the different software and hardware components involved in network communication should divide labor and interact with one another. It defines a seven-layer set of functional elements, ranging from the physical interconnections at Layer 1 (also known as the physical layer) all the way up to Layer 7, the application layer.

TCP/IP: The internet protocol suite is the computer networking model and set of communications protocols used on the internet and similar computer networks. It is commonly known as TCP/IP. TCP/IP provides end-to-end data communication specifying how data should be packetized, addressed, transmitted, routed and received.

Packets: A packet is simply a chunk of data enclosed in one or more wrappers that help to identify the chunk of data and route it to the correct destination. *Destination* in this sense means a particular application or process running on a particular machine. These wrappers consist of headers and trailers. Headers are simply bits of data added to the beginning of a packet. Trailers are added to the end of a packet.

Protocols: Networking protocols specify what types of data can be sent, how each type of message will be identified, what actions must be taken by participants in the conversation, precisely where in the packet header or trailer each type of required information will be placed, and more.

Endianness: Endianness refers to the order of the bytes, comprising a digital word, in computer memory. It also describes the order of byte transmission over a digital link. Words may be represented in big-endian or little-endian format.

3. Analysis and Design

The concept behind analyzing packets is quite simple; you just have to read byte by byte and interpret the binary information in human readable form and print it. As per the requirements, the program must have options for different output formatting. Also it requires an option so that the user can choose what kind of output formatting he wants. To do this a modular code structure will be the best for management. Each module will handle a specific output format. The design diagram is shown below:

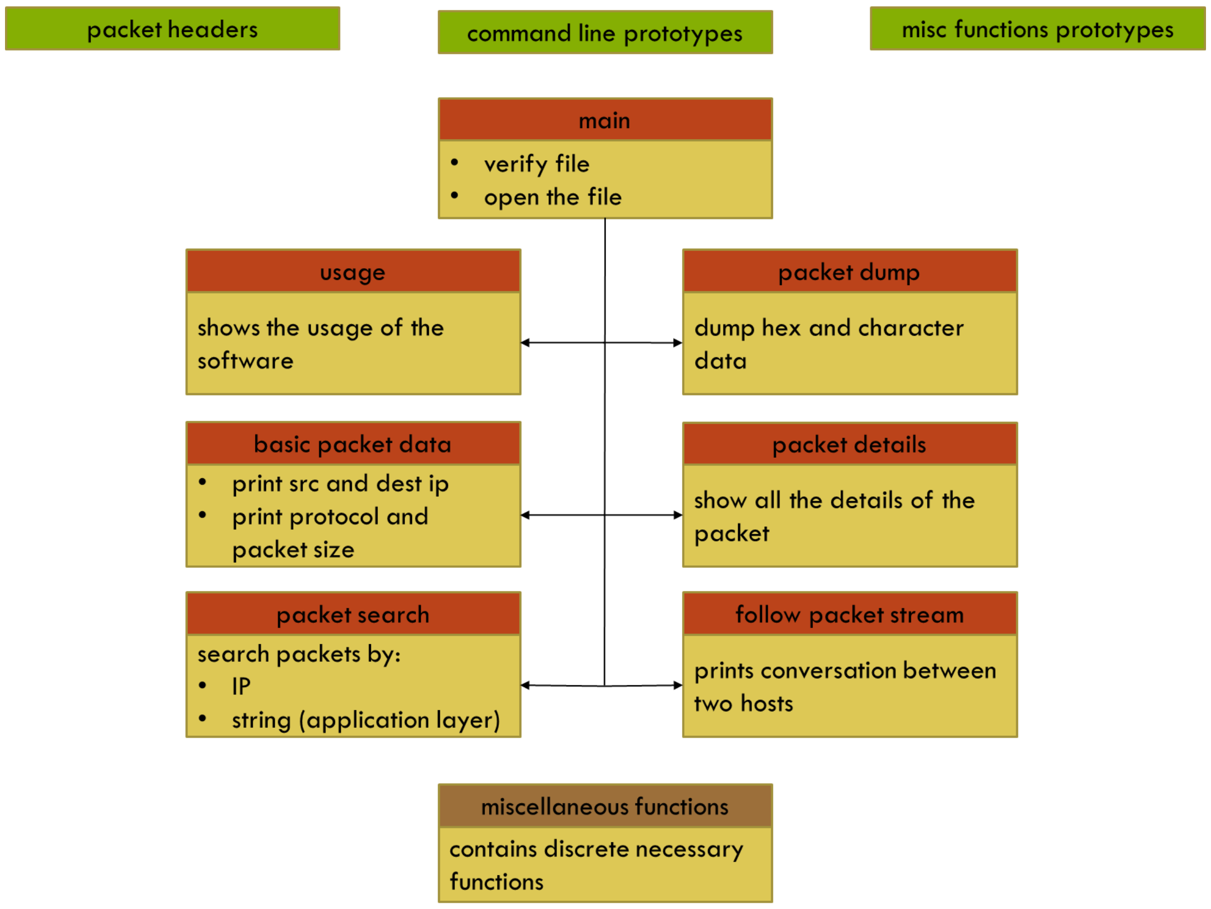

packet headers
command line prototypes
misc functions prototypes
main
verify file
open the file
usage
shows the usage of the software
packet dump
dump hex and character data
basic packet data
print src and dest ip
print protocol and packet size
packet details
show all the details of the packet
packet search
search packets by:
IP
string (application layer)
follow packet stream
prints conversation between two hosts
miscellaneous functions
contains discrete necessary functions

Figure 3: Design pattern of Hostman

4. Implementation

Hostman is developed in ANSI C. As a result the style of coding style is structural. It's a command line driven program. The whole program has been compiled from 12 source files:

Header files:

- packetHeaders.h
- packetBasics.h
- cmdOptions.h
- miscFunctions.h

C files:

- main.c
- miscFunctions.c
- packetBasicInfo.c
- packetDetails.c
- packetDump.c
- packetSearch.c
- printUsage.c
- followStream.c

4.1. Header files

The header files contain structures of the TCP/IP protocol, different macros and the prototypes of some funcitons.

4.1.1. packetHeaders.h

This header file is the heart of the whole program. It is included in every other module. This file contains the structs of the different headers: pcap file header, pcap record header, Ethernet frame header and ARP, IP, TCP and UDP headers.

The file also has some macros defined such as the size of different headers and hex values for doing bit masking.

The Ethernet frame header and ARP header is shown below as a sample:

```
typedef struct ethernetFrame_s {
        uint8_t destinationMac[6];
        uint8_t sourceMac[6];
        uint16_t type;
} ethernetFrame_t;

typedef struct arpHeader_s {
    uint16_t hardwareType;          /* Hardware Type  */
    uint16_t protocolType;          /* Protocol Type    */
    uint8_t hwAddrLength;           /* Hardware Address Length */
    uint8_t protoAddrLength;        /* Protocol Address Length */
    uint16_t opcode;                /* Operation Code */
    uint8_t senderHwAddr[6];        /* Sender hardware address */
    uint8_t senderIpAddr[4];        /* Sender IP address  */
    uint8_t targetHwAddr[6];        /* Target hardware address */
    uint8_t targetIpAddr[4];        /* Target IP address */
} arpHeader_t;
```

To parse a packet, first it is stored in an uint8_t array. Then a pointer type ethernetFrame_t struct (ethernetFrame_t*) is taken and the array is pointed to it. This parses out the information of the first layer. We can know the protocol of the next layer because it is mentioned in the Ethernet frame. Then depending on the type of the next layer a pointer type struct of the next layer is taken and the array where the packet is stored is pointed to that struct. In this way the whole packet is parsed and the data in it is printed.

4.1.2. cmdOptions.h

The program is command line driven. The prototypes of the functions that are invoked by the user are in this header.

4.1.3. packetBasics.h

This file contains the function prototypes that are required to show basic packet information. This file s included in the packetSearch.c file so that the search results show basic data of the found packets.

4.1.4. miscFunctions.h

Some functions were separated from source files to miscFunctions.c so that those source files remain organized. These functions were too small to make a separate source file for them. As a result the source file miscFunctions.c was made which contains discrete important functions. miscFunctions.h contains the prototypes of these functions.

4.2. C Source files

These files contain the actual functions that make the core of Hostman. They are explained below in detail.

4.2.1. main.c

The program starts executing from here. It opens a pcap file and checks it validity. If it is a valid pcap file it checks for the validity of the passed command line arguments. If the arguments are valid then the corresponding function that the user chose is invoked. After the operation is completed the file is closed.

4.2.2. miscFunctions.c

This file contains various discrete functions required throughout the whole program in different places. The implemented functions are:

- uint16_t swap_uint16(uint16_t), swaps 16 bit integers
- uint32_t swap_uint32(uint32_t), swaps 32 bit integers
- void invalidArguments(), prints an error message when the command line arguments are wrong
- uint8_t checkIfNumber(char*), checks if a string contains digits
- const char* checkIP(char*), checks if the ip address is valid or not
- const char* checkExtensions(char*), checks if the pcap file format is correct
- const char* determineEthernetProtocol(uint16_t), determines the protocol of the layer after the Ethernet frame
- const char* determineIpProtocol(uint8_t), determines the protocol of the layer after the network later
- const char* determinePort(uint16_t), determines the port name
- const char* flagSetOrNot(uint8_t), returns the string value whether a bit is set or not

4.2.3. packetBasicInfo.c

To print basic data of the packets it is necessary to parse the packet only up to from the Ethernet frame to the network frame. When the function, void packetBasicInfo(FILE*) is called it parses the packet up to the network layer and shows:

- source address
- destination address
- protocol
- packet length

4.2.4. packetDetails.c

The functions that print the packet details are written in this file. There are manly three functions that do this:

- void allPacketDetails(FILE*), prints the details of all the packets present in the file
- void selectedPacketDetails(FILE* , uint32_t), prints the details of a particular packet that the user can specify
- void selectedPacketRangeDetails(FILE* , uint32_t , uint32_t), prints the details of a specified packet range

4.2.5. packetDump.c

This file contains the function, void packetDump(FILE*) which prints the hexdump of the packets and their corresponding ASCII characters (if they are printable).

4.2.6. packetSearch.c

Two search functions are implemented in this source file:

- void ipSearch(FILE*), gives users the option to search packets that contain a particular IP address
- void textSearch(FILE*), gives users the option to search strings in the application layer

Packet capture files can be very large in size. So searching strings can be a very time consuming process. For faster searching of strings the KMP algorithm has been used which can search texts in linear time.

The KMP function is defined as void kmpSearch(uint8_t* data, int32_t dataLen, char *pattern).

4.2.7. printUsage.c

This file contains, void printUsage(), which prints the usage of the program in case the user needs to know the command line arguments in case he forgets them.

4.2.8. followStream.c

This is perhaps the most important feature of this program. When the function followStream() with the specified arguments is called it prints the sole conversation between two hosts (the application layer data). The data of the two hosts are printed with a separate color code for each of them for easier readability of the conversation.

5. User Manual

The commands that invoke the different functions of Hostman are given below:

- prints usage: ./Hostman
- print packet dump: ./Hostman <filename> -d
- print basic information: ./Hostman <filename> –b
- print all packet details: ./Hostman <filename> -v
- print a particular packet details: ./Hostman <filename> -v <packet no.>
- print packet details in a range: ./Hostman <filename> -v <start no.> <number of packets>
- search packets by IP: ./Hostman <filename> ip
- search packets by text: ./Hostman <filename> -t
- follow packet stream: ./Hostman <filename> -f

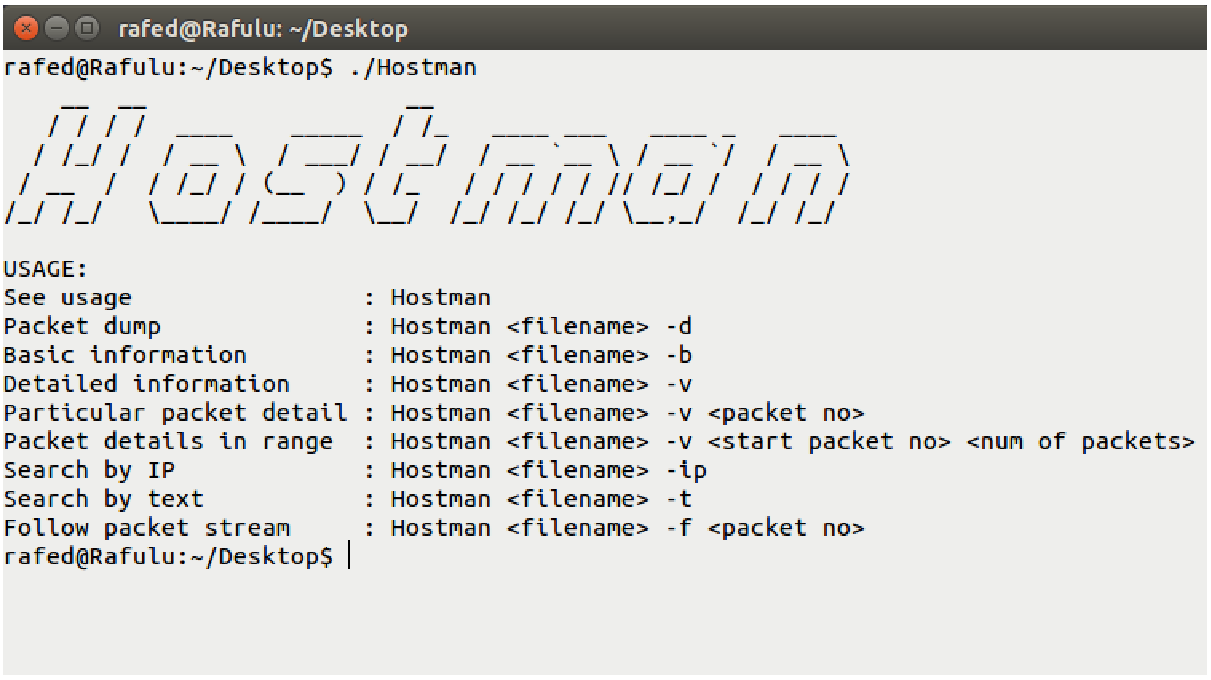
Sample output of each command is given in the next section.

6. Program Outputs

Some sample outputs of the program for the file “test.pcap” is shown below:

6.1. Print usage

Command in terminal: ./Hostman

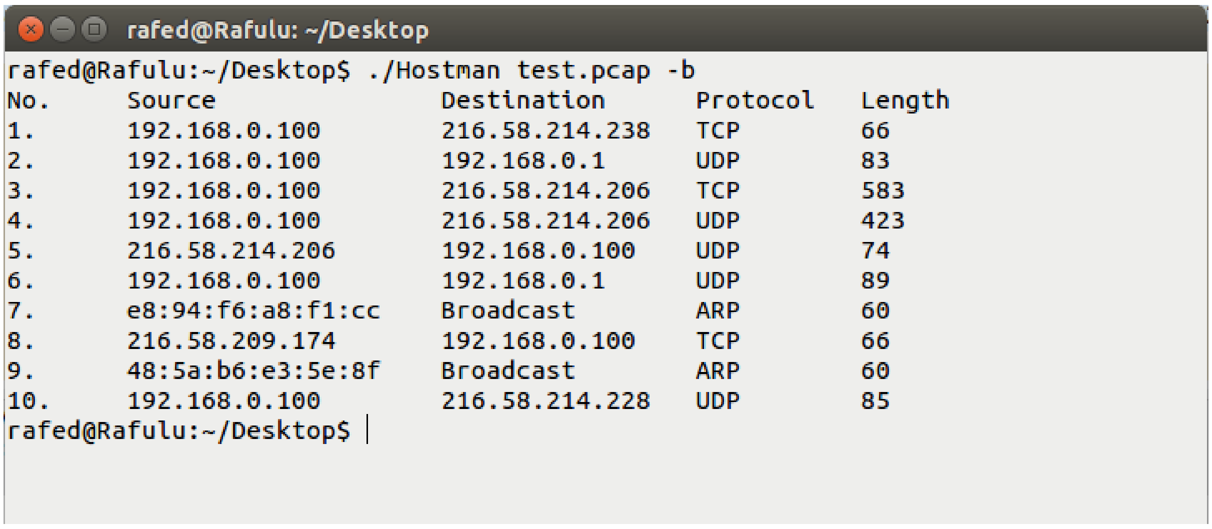


```
rafed@Rafulu:~/Desktop$ ./Hostman
    __  __                  __
   / / / /  ____   _____  / /_  ____ ___   ____ _  ____
  / /_/ /  / __ \ / ___/ / __/ / __ `__ \ / __ `/ / __ \
 / __  /  / /_/ / (__  ) / /_  / / / / / // /_/ /  / / / /
/_/ /_/   \____/ /____/  \__/  /_/ /_/ /_/ \__,_/  /_/ /_/

USAGE:
See usage                 : Hostman
Packet dump               : Hostman <filename> -d
Basic information         : Hostman <filename> -b
Detailed information      : Hostman <filename> -v
Particular packet detail  : Hostman <filename> -v <packet no>
Packet details in range   : Hostman <filename> -v <start packet no> <num of packets>
Search by IP              : Hostman <filename> -ip
Search by text            : Hostman <filename> -t
Follow packet stream      : Hostman <filename> -f <packet no>
rafed@Rafulu:~/Desktop$
```

Figure 4: Output of usage

6.2. Print basic information

Command in terminal: ./Hostman test.pcap –b

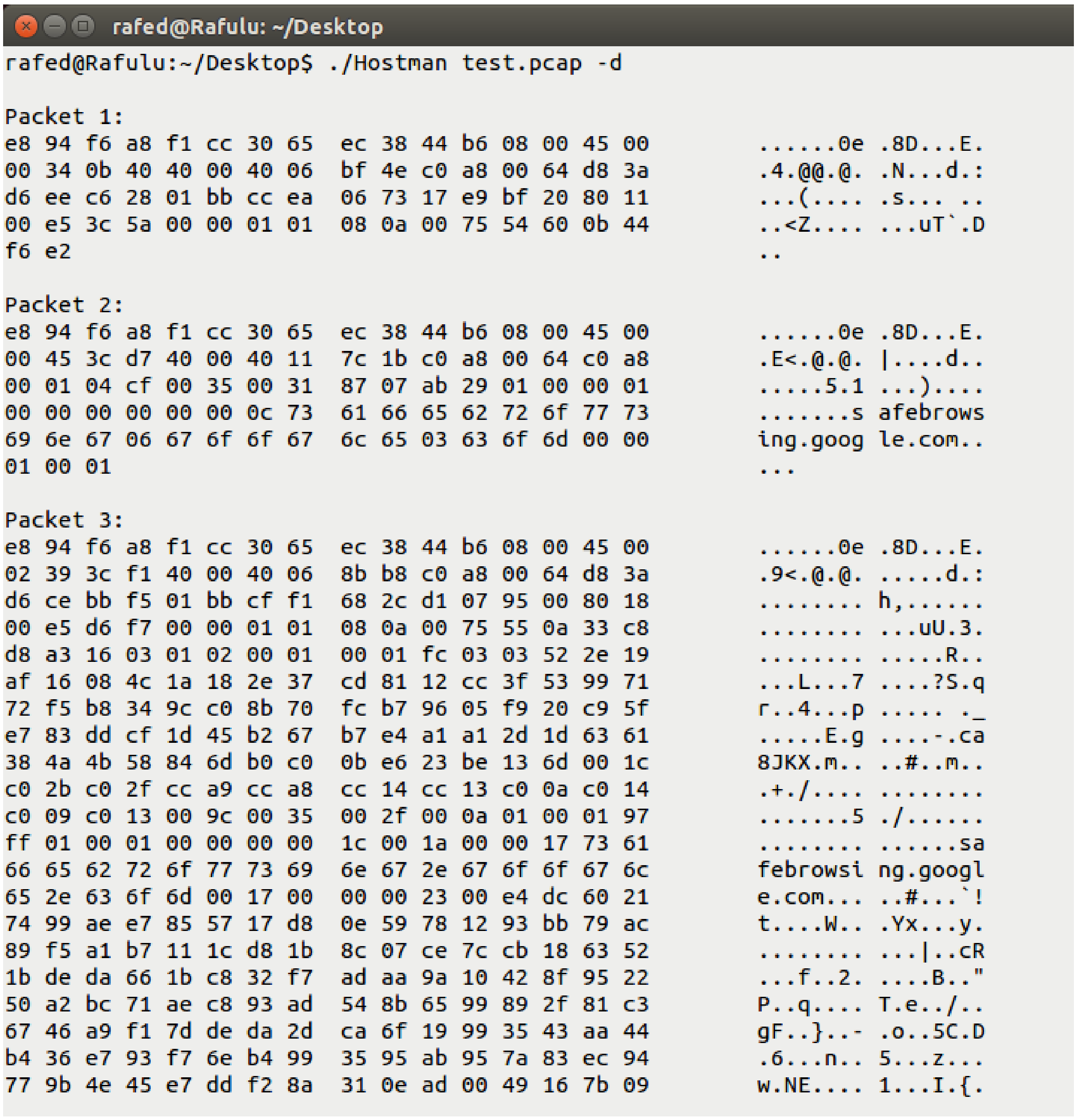


```
rafed@Rafulu:~/Desktop$ ./Hostman test.pcap -b
No.     Source              Destination      Protocol   Length
1.      192.168.0.100       216.58.214.238   TCP        66
2.      192.168.0.100       192.168.0.1      UDP        83
3.      192.168.0.100       216.58.214.206   TCP        583
4.      192.168.0.100       216.58.214.206   UDP        423
5.      216.58.214.206      192.168.0.100    UDP        74
6.      192.168.0.100       192.168.0.1      UDP        89
7.      e8:94:f6:a8:f1:cc   Broadcast        ARP        60
8.      216.58.209.174      192.168.0.100    TCP        66
9.      48:5a:b6:e3:5e:8f   Broadcast        ARP        60
10.     192.168.0.100       216.58.214.228   UDP        85
rafed@Rafulu:~/Desktop$
```

Figure 5: Output of packet basics

6.3. Print packet dump

Command in terminal: ./Hostman test.pcap –d

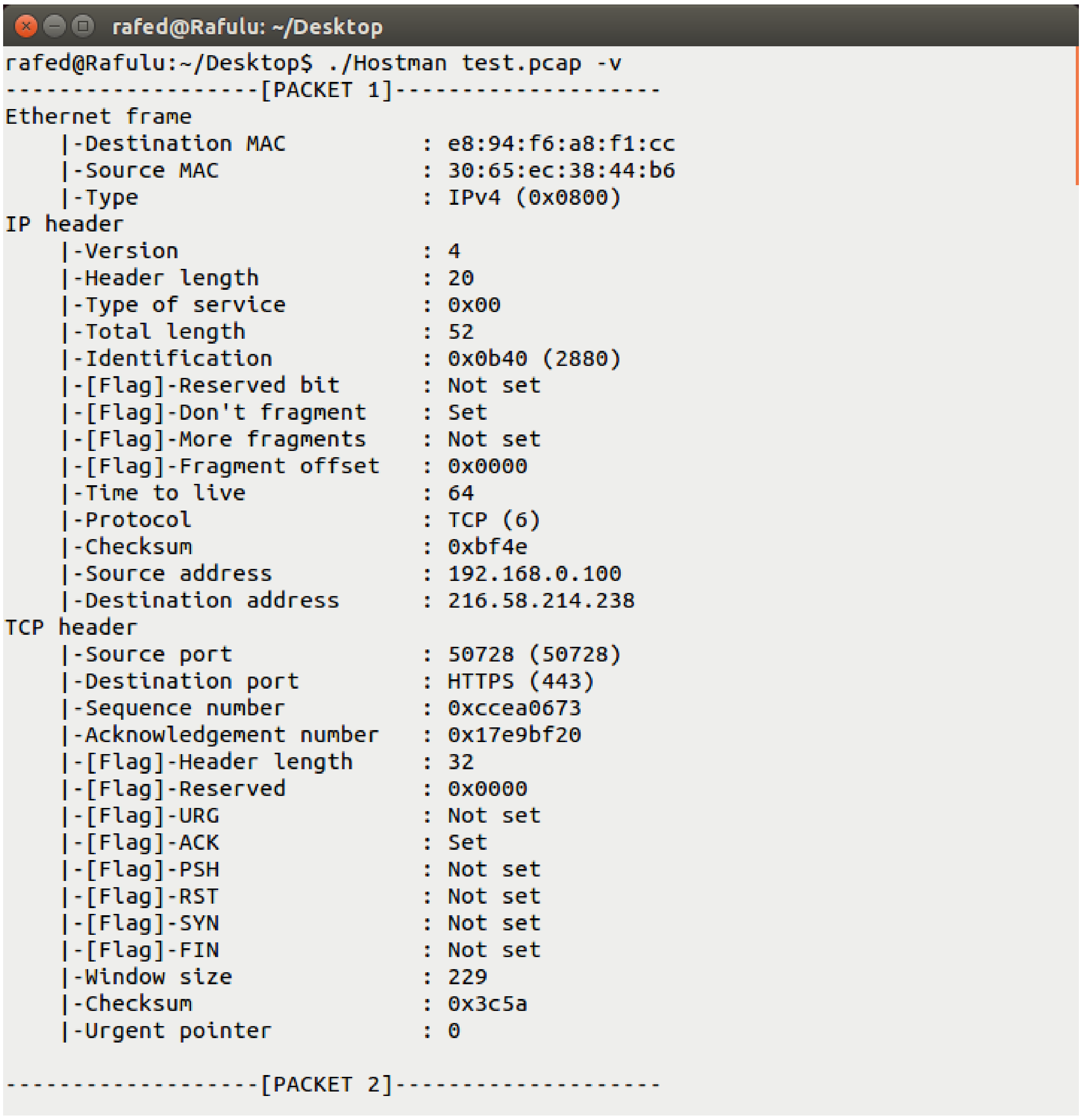


```
rafed@Rafulu: ~/Desktop
rafed@Rafulu:~/Desktop$ ./Hostman test.pcap -d

Packet 1:
e8 94 f6 a8 f1 cc 30 65  ec 38 44 b6 08 00 45 00      ......0e .8D...E.
00 34 0b 40 40 00 40 06  bf 4e c0 a8 00 64 d8 3a      .4.@@.@. .N...d.:
d6 ee c6 28 01 bb cc ea  06 73 17 e9 bf 20 80 11      ...(.... .s... ..
00 e5 3c 5a 00 00 01 01  08 0a 00 75 54 60 0b 44      ..<Z.... ...uT`.D
f6 e2                                                 ..

Packet 2:
e8 94 f6 a8 f1 cc 30 65  ec 38 44 b6 08 00 45 00      ......0e .8D...E.
00 45 3c d7 40 00 40 11  7c 1b c0 a8 00 64 c0 a8      .E<.@.@. |....d..
00 01 04 cf 00 35 00 31  87 07 ab 29 01 00 00 01      .....5.1 ...)....
00 00 00 00 00 00 0c 73  61 66 65 62 72 6f 77 73      .......s afebrows
69 6e 67 06 67 6f 6f 67  6c 65 03 63 6f 6d 00 00      ing.goog le.com..
01 00 01                                              ...

Packet 3:
e8 94 f6 a8 f1 cc 30 65  ec 38 44 b6 08 00 45 00      ......0e .8D...E.
02 39 3c f1 40 00 40 06  8b b8 c0 a8 00 64 d8 3a      .9<.@.@. .....d.:
d6 ce bb f5 01 bb cf f1  68 2c d1 07 95 00 80 18      ........ h,......
00 e5 d6 f7 00 00 01 01  08 0a 00 75 55 0a 33 c8      ........ ...uU.3.
d8 a3 16 03 01 02 00 01  00 01 fc 03 03 52 2e 19      ........ .....R..
af 16 08 4c 1a 18 2e 37  cd 81 12 cc 3f 53 99 71      ...L...7 ....?S.q
72 f5 b8 34 9c c0 8b 70  fc b7 96 05 f9 20 c9 5f      r..4...p ..... ._
e7 83 dd cf 1d 45 b2 67  b7 e4 a1 a1 2d 1d 63 61      .....E.g ....-.ca
38 4a 4b 58 84 6d b0 c0  0b e6 23 be 13 6d 00 1c      8JKX.m.. ..#..m..
c0 2b c0 2f cc a9 cc a8  cc 14 cc 13 c0 0a c0 14      .+./.... ........
c0 09 c0 13 00 9c 00 35  00 2f 00 0a 01 00 01 97      .......5 ./......
ff 01 00 01 00 00 00 00  1c 00 1a 00 00 17 73 61      ........ ......sa
66 65 62 72 6f 77 73 69  6e 67 2e 67 6f 6f 67 6c      febrowsi ng.googl
65 2e 63 6f 6d 00 17 00  00 00 23 00 e4 dc 60 21      e.com... ..#...`!
74 99 ae e7 85 57 17 d8  0e 59 78 12 93 bb 79 ac      t....W.. .Yx...y.
89 f5 a1 b7 11 1c d8 1b  8c 07 ce 7c cb 18 63 52      ........ ...|..cR
1b de da 66 1b c8 32 f7  ad aa 9a 10 42 8f 95 22      ...f..2. ....B.."
50 a2 bc 71 ae c8 93 ad  54 8b 65 99 89 2f 81 c3      P..q.... T.e../..
67 46 a9 f1 7d de da 2d  ca 6f 19 99 35 43 aa 44      gF..}..- .o..5C.D
b4 36 e7 93 f7 6e b4 99  35 95 ab 95 7a 83 ec 94      .6...n.. 5...z...
77 9b 4e 45 e7 dd f2 8a  31 0e ad 00 49 16 7b 09      w.NE.... 1...I.{.
```

Figure 6: Output of packet dump

6.4. Print all packet details

Command in terminal: ./Hostman test.pcap –v

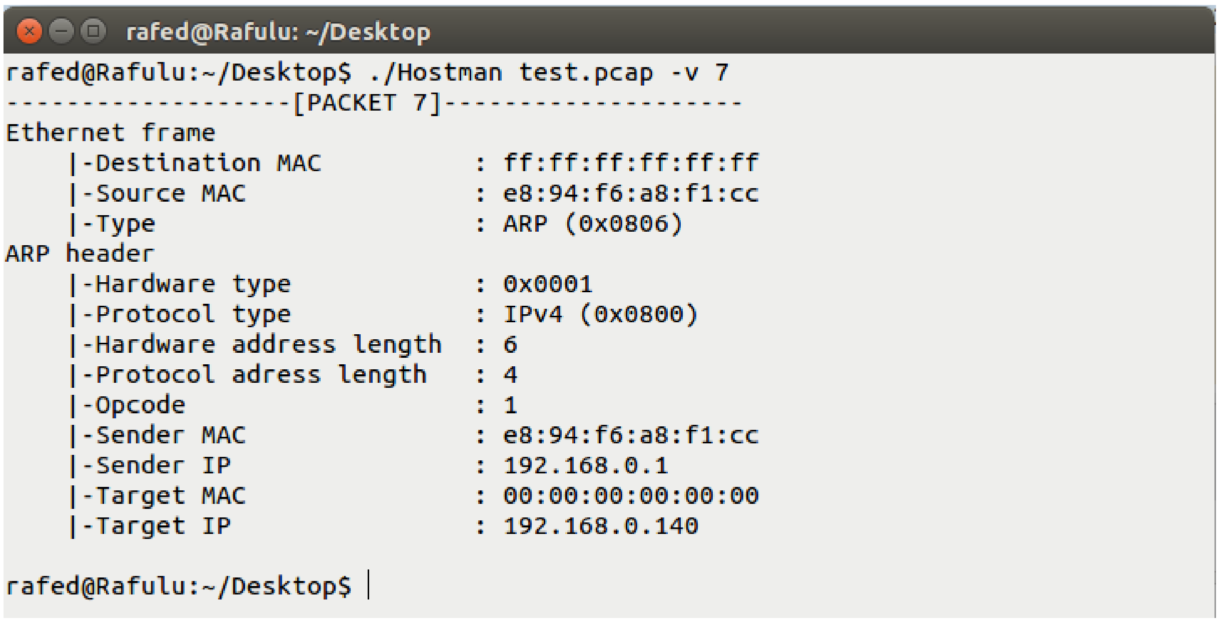


```
rafed@Rafulu: ~/Desktop
rafed@Rafulu:~/Desktop$ ./Hostman test.pcap -v
-------------------[PACKET 1]--------------------
Ethernet frame
    |-Destination MAC            : e8:94:f6:a8:f1:cc
    |-Source MAC                 : 30:65:ec:38:44:b6
    |-Type                       : IPv4 (0x0800)
IP header
    |-Version                    : 4
    |-Header length              : 20
    |-Type of service            : 0x00
    |-Total length               : 52
    |-Identification             : 0x0b40 (2880)
    |-[Flag]-Reserved bit        : Not set
    |-[Flag]-Don't fragment      : Set
    |-[Flag]-More fragments      : Not set
    |-[Flag]-Fragment offset     : 0x0000
    |-Time to live               : 64
    |-Protocol                   : TCP (6)
    |-Checksum                   : 0xbf4e
    |-Source address             : 192.168.0.100
    |-Destination address        : 216.58.214.238
TCP header
    |-Source port                : 50728 (50728)
    |-Destination port           : HTTPS (443)
    |-Sequence number            : 0xccea0673
    |-Acknowledgement number     : 0x17e9bf20
    |-[Flag]-Header length       : 32
    |-[Flag]-Reserved            : 0x0000
    |-[Flag]-URG                 : Not set
    |-[Flag]-ACK                 : Set
    |-[Flag]-PSH                 : Not set
    |-[Flag]-RST                 : Not set
    |-[Flag]-SYN                 : Not set
    |-[Flag]-FIN                 : Not set
    |-Window size                : 229
    |-Checksum                   : 0x3c5a
    |-Urgent pointer             : 0

-------------------[PACKET 2]--------------------
```

Figure 7: Output of packet details

6.5. Print a particular packet details

Command in terminal: ./Hostman test.pcap –v 7



```
rafed@Rafulu: ~/Desktop
rafed@Rafulu:~/Desktop$ ./Hostman test.pcap -v 7
-------------------[PACKET 7]--------------------
Ethernet frame
    |-Destination MAC           : ff:ff:ff:ff:ff:ff
    |-Source MAC                : e8:94:f6:a8:f1:cc
    |-Type                      : ARP (0x0806)
ARP header
    |-Hardware type             : 0x0001
    |-Protocol type             : IPv4 (0x0800)
    |-Hardware address length   : 6
    |-Protocol adress length    : 4
    |-Opcode                    : 1
    |-Sender MAC                : e8:94:f6:a8:f1:cc
    |-Sender IP                 : 192.168.0.1
    |-Target MAC                : 00:00:00:00:00:00
    |-Target IP                 : 192.168.0.140

rafed@Rafulu:~/Desktop$
```

Figure 8: Output of particular packet details

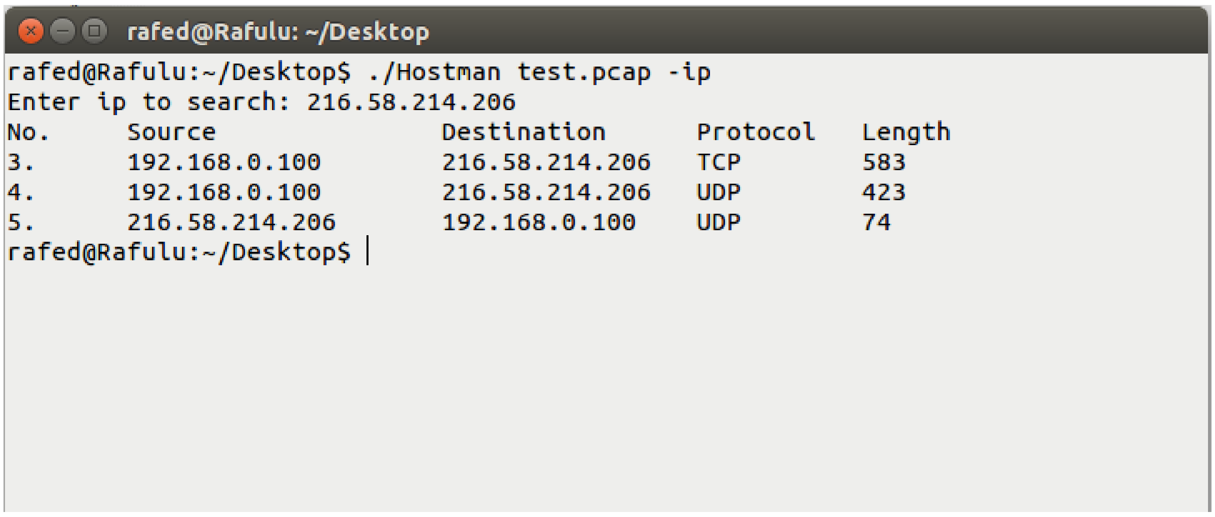
6.6. Print packet details in range

Command in terminal: ./Hostman test.pcap –v 7 1

Produces similar output to Figure 7 and Figure 8.

6.7. Search packet by IP

Command in terminal: ./Hostman test.pcap -ip

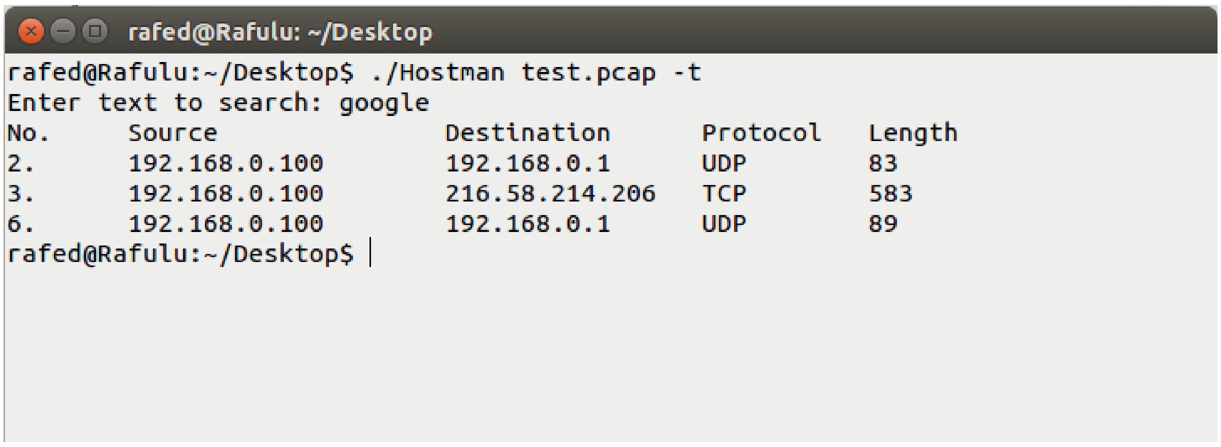


```
rafed@Rafulu: ~/Desktop
rafed@Rafulu:~/Desktop$ ./Hostman test.pcap -ip
Enter ip to search: 216.58.214.206
No.     Source                  Destination      Protocol   Length
3.      192.168.0.100           216.58.214.206   TCP        583
4.      192.168.0.100           216.58.214.206   UDP        423
5.      216.58.214.206          192.168.0.100    UDP        74
rafed@Rafulu:~/Desktop$
```

Figure 9: Output of search packet by IP

6.8. Search packet by text

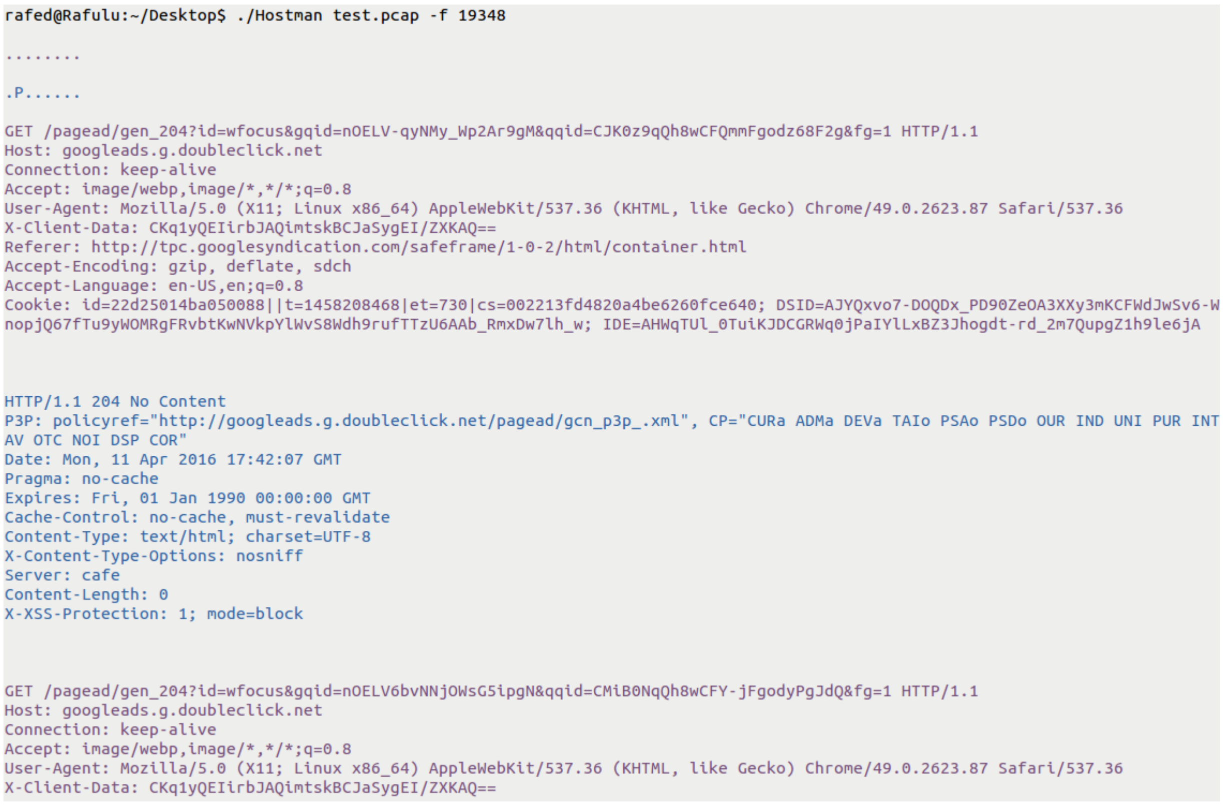
Command in terminal: ./Hostman test.pcap -t



```
rafed@Rafulu: ~/Desktop
rafed@Rafulu:~/Desktop$ ./Hostman test.pcap -t
Enter text to search: google
No.     Source              Destination       Protocol    Length
2.      192.168.0.100       192.168.0.1       UDP         83
3.      192.168.0.100       216.58.214.206    TCP         583
6.      192.168.0.100       192.168.0.1       UDP         89
rafed@Rafulu:~/Desktop$
```

Figure 10: Output of search packet by text

6.9. Follow packet stream



```
rafed@Rafulu:~/Desktop$ ./Hostman test.pcap -f 19348

........

.P......

GET /pagead/gen_204?id=wfocus&gqid=nOELV-qyNMy_Wp2Ar9gM&qqid=CJK0z9qQh8wCFQmmFgodz68F2g&fg=1 HTTP/1.1
Host: googleads.g.doubleclick.net
Connection: keep-alive
Accept: image/webp,image/*,*/*;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.87 Safari/537.36
X-Client-Data: CKq1yQEIirbJAQimtskBCJaSygEI/ZXKAQ==
Referer: http://tpc.googlesyndication.com/safeframe/1-0-2/html/container.html
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Cookie: id=22d25014ba050088||t=1458208468|et=730|cs=002213fd4820a4be6260fce640; DSID=AJYQxvo7-DOQDx_PD90ZeOA3XXy3mKCFWdJwSv6-W
nopjQ67fTu9yWOMRgFRvbtKwNVkpYlWvS8Wdh9rufTTzU6AAb_RmxDw7lh_w; IDE=AHWqTUl_0TuiKJDCGRWq0jPaIYlLxBZ3Jhogdt-rd_2m7QupgZ1h9le6jA


HTTP/1.1 204 No Content
P3P: policyref="http://googleads.g.doubleclick.net/pagead/gcn_p3p_.xml", CP="CURa ADMa DEVa TAIo PSAo PSDo OUR IND UNI PUR INT
AV OTC NOI DSP COR"
Date: Mon, 11 Apr 2016 17:42:07 GMT
Pragma: no-cache
Expires: Fri, 01 Jan 1990 00:00:00 GMT
Cache-Control: no-cache, must-revalidate
Content-Type: text/html; charset=UTF-8
X-Content-Type-Options: nosniff
Server: cafe
Content-Length: 0
X-XSS-Protection: 1; mode=block


GET /pagead/gen_204?id=wfocus&gqid=nOELV6bvNNjOWsG5ipgN&qqid=CMiB0NqQh8wCFY-jFgodyPgJdQ&fg=1 HTTP/1.1
Host: googleads.g.doubleclick.net
Connection: keep-alive
Accept: image/webp,image/*,*/*;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.87 Safari/537.36
X-Client-Data: CKq1yQEIirbJAQimtskBCJaSygEI/ZXKAQ==
```

Figure 11 Follow packet stream

7. Conclusion

Packet analyzers are the pocket knife of any network engineer. Although the scope of Hostman is not so big and its functionality is limited it can still be used by network admins to analyze TCP/IP packets in their network. Information such as which ip addresses are generating traffic, what kind of payload are being sent by users can be seen using Hostman.

Also anyone who has interest in making packet analyzers can use Hostman as a reference. By reading this report and studying the source code of Hostman a basic concept behind building packet analyzers can be obtained.

Reference

[1] Cormen, Leiseson, Rivest, Stein. Introduction to Algorithms. 3rd edition. Pg-1002

[2] Forouzan. Data Communications and Networking. 5th edition.

[3] Kurose, Ross. Computer Networking. 6th edition.

[4] http://searchnetworking.techtarget.com/definition/packet, TechTarget, last accessed on 14-04-2016

[5] https://www.techopedia.com/definition/25323/packet-analyzer, TechoPedia, last accessed on 14-04-2016

[6] http://www.itinfo.am/eng/software-development-methodologies/, Software Development Methodologies, last accessed on 14-04-2016

[7] http://searchnetworking.techtarget.com/answer/What-is-the-difference-between-OSI-model-and-TCP-IP-other-than-the-number-of-layers, TechTarget, last accessed on 15-04-2016

[8] http://www.wildpackets.com/resources/compendium/ethernet/ethernet_packets, Wildpackets, last accessed on 15-05-2016

[9] https://en.wikipedia.org/wiki/Endianness, Wikipedia, last accessed on 16-04-16

[10] https://en.wikipedia.org/wiki/Knuth%E2%80%93Morris%E2%80%93Pratt_algorithm, Wikipedia, 16-04-16