### **Course Number :- F303**

### **Course Title :- Computer Networks**

### **Group Number :- G5**

### **Evaluation Component :- III**

**PROTOCOL ANALYST**



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI**

### **Group Members:-**

### **AKHIL KUMAR AGRAWAL 2014A7PS024P GAURANG BANSAL 2014A7PS128P SUBHAM KUMAR 2014A7PS121P**

### Github Project Link : - <https://github.com/Gaurang18/ComputerNetworkProject.git>

### 19 April 2017

# **Problem Statement**

A packet analyzer is a tool which is used to intercept, parse, gather, report and display the statistics of various packets coming and going. There already exist many protocol analysers such as Wireshark, and one may question the need of another packet capturing system. Although a wide variety of packet analysers exist but generally they are quite expansive in terms of memory. For eg. Wireshark has about 18 MB installation file which on installation takes about 81 MB of space. Generally, a user does not require all the functionalities that the wireshark provides. So it’s required to design a packet capturing tool which passively captures the packets and record it with very little system overhead.

**Scope of Work**

|  |  |  |
| --- | --- | --- |
| **Topic** | **Date** | **Mentor** |
| Literature Search and Review | 18th February - 3rd March | Akhil Gaurang Subham |
| Analysis and Modeling | 13th March - 16th March | Subham |
| Pseudo Code Generation & Socket Networking | 17th March - 20th March | Gaurang |
| Designing a Simple Prototype | 20th March - 24th March | Shubham & Gaurang |
| Network monitor  Ethernet Layer Packet Capturing | 25th March - 29th March | Akhil |
| Protocol Capturing  ( Network Layer & Transport Layer) | 25th March - 28st April | Gaurang & Subham |
| Packet Filtering  (Application Layer) | 28th March - 1st April | Subham |
| Network Utilities & Code Optimization | 1st April - 5th April | Gaurang |
| Packet Log File & User Display | 7th April - 10th April | Akhil |
| System testing and Evaluation | 10th April - 16th April | Gaurang |
| Buffer Space (Completion of Previous tasks) | 10th April - 13th April | Akhil & Subham |
| Final Project Report | 16th April - 18th April | Akhil |

**Design Challenges**

Some of Design Challenges we faced are:

* How to capture raw packets and process them quickly enough to ensure that they are not dropped before analysis.
* Filtering of the Packets

**Implementation Challenges**

We have constructed packet headers in a separate file. The headers have been formed using the well known headers format.

* Creating Header Structures for Protocols
* Extracting the relevant information from the application layer protocols and the transport layer protocols.
* Filtering of Captured Packets

**Architecture Details**

We are implementing a simple Packet-Capturing and Analysis Tool (“Protocol Analyst”) that will be able to capture application, transport, network and data link level structured PDUs. We have provided a brief overview of Design aspects of our model, its execution and implementation details.

* We have used raw sockets to capture and analyse packets from the ethernet interface. When the user runs the protocol analyst, the protocol analyst starts to capture the packets of data from ethernet interface using the raw socket. Raw packets are passed to the socket and to the analyst process.
* The whole packet is stored in a buffer which is used for processing. We have created headers for different protocols based on the different attributes they have. We use a pointer to capture the header from the beginning and then use our constructed header structures to extract information, first of the data link layer such as Host MAC address, Destination MAC Address and then upper layer protocols.
* After extracting the ethernet header we extract the payload of the packet and use it to find the Network Layer Protocols such as Internet Protocol (IP) ,Internet Control Message Protocol (ICMP) and Internet Group Management Protocol (IGMP).The Internet Protocol is used to extract some of the most crucial information such as source IP address, destination IP address, which transport layer protocol is used (We have calculated the type of protocol using the protocol field in IP header and comparing it with IANA protocols number existing). It also provides information related to Code , Type and Checksum attributes for ICMP and IGMP (reference for type number is IANA).
* The payload of IP contains Transport layer data unit. Using the information from the underlying network layer, we know which transport layer protocol used. For our Protocol Analyst, it analyses the two most popular transport layer protocols TCP and UDP. TCP being denoted by protocol number 6 and UDP by 17. The most important information from transport layer protocol is the source and destination port number. Since the first 1024 ports are reserved, these ports can help us identify the type of Application Layer protocol that is used. For example port number 80 in TCP header indicates HTTP is the application layer protocol in use.
* The application protocols that have been analysed are HTTP (Hypertext Transfer Protocol) and DNS (Domain Name System). DNS works generally using UDP as its transport layer protocol. The structures of DNS and HTTP header are designed to capture the necessary information. There are various attributes like the id, counts of the answer, additional and authoritative information in the DNS packet. The HTTP fields like Accept, Connection,Cookie are extracted and printed in the log file.

**Structural Design and Implementation Details**

The Protocol Analyst is designed to capture all the packets coming to and going via the ethernet interface. The program for “Protocol Analyst” is written in C Language and is designed for Linux.

* We have used raw sockets to capture packets as they are i.e all incoming packets are passed without any modifications as raw packets, at the link layer to the socket. They are processed to extract the required information.
* We have created structures for each of the packet headers, containing all the details available in each of the headers in the form of variables present in the structure. The packet headers are processed bottom up from the data link layer where protocols like ARP are present all the way up to HTTP and DNS at the application layer.
* All the details are dumped to a logfile which is a text file with all the relevant information as captured by the analyzer.The user has an option to filter and examine the details of each protocol separately in a different log file customized to his requirements. For example,if the UDP option is selected all the UDP packets and the information present in those headers like source port, destination port are printed in a log file.The number of packets captured per protocol is computed in real time and displayed on the console as they are arriving at the raw socket.
* The user has three options for capturing packets. The first is to capture a specified number of packets which will be specified beforehand. The second option is capturing packets for a specified amount of time entered in seconds. The third option is to run the analyzer indefinitely till user wishes to stop it.
* User can run the “Protocol Analyst” using various command line options.Then the user can open any network application like a browser along with the “Protocol Analyst” running in a terminal. He can analyse data based on:
  + Number of Packets :- User can specify the number of packets he wishes to capture.
  + Maximum Time :- User can specify the maximum time limit for capturing packets.
  + Key stop :- Press a particular key to stop the capturing of packets whenever required.

Some of linux libraries used are as following:

* netinet/in.h: The header file in.h contains constants and structures needed for internet domain addresses. These include standard IP addresses and TCP and UDP port numbers..
* errno.h: It defines macros for reporting and retrieving error conditions through error codes stored in a static memory location called errno (short for "error number").
* netdb.h: The <netdb.h> header may define the in\_port\_t type and the in\_addr\_t type as described in <netinet/in.h>
* sys/socket.h: This header has functions and variables to help communicate with sockets.

## **Limitations of Work Done**

* Output is Flat File containing text output
* It runs via command line only
* The packets which are captured are extracted through sockets. Since the data can’t be processed as fast as packets are being received there is possibility of loss of packets
* The packets captured are processed using buffer (array of Char in C program) which has its own size constraints.

## **Major Takeaways and Learning Aspects**

1. Gaurang Bansal

For me it was great opportunity to learn, explore and understand the protocols. Although protocol headers are already available in the linux libraries, yet we created our own headers so it gave us insight into how much memory is used by each header, how it can be used. Moreover it was link between theoretical information and practical information. Personally I could understand the insights of Packet Analysis although a mere scratch yet has given me deep impetus to research in this field more. I remember the chinese proverb given at end of Chapter 1 of Kurose book

“Tell me and I forget. Show me and I remember. Involve me and I understand”

2. Akhil Kumar Agrawal

It was a great learning experience for me to know about protocols and get an in depth understanding of how they function. We created our own headers and functions and learnt a lot about their functionalities. It has also intrigued me to further study about networks. I am thankful to the professors for giving this wonderful opportunity and my group mates for bearing with me.

3. Subham Kumar

When you go through challenges you learn. It was great experience I would say, capturing packets and analysing them. This project was a great stepping stone making the bridge between theoretical information and practical application. It also proved that in reality everything does not goes as ideally as planned. I would like to express my gratitude to teachers and group mates for a wonderful experience.

## 

## **Conclusions**

The packet capturing and Analysis tool “Protocol Analyst” is a simple application which is designed to capture packets from Ethernet interface providing analysis of some of the most popular protocols. The program is designed to run (execute) with least expense in terms of memory and time, as compared to other protocol analysers such as tcpdump or Wireshark. The packet capturing, analysis, extracting header information in least amount of time has been the primary objective over the use of any sort Graphical User Interface. The list of protocols that can be analysed is as following:

* Address Resolution Protocol (Data Link Layer)
* Internet Protocol (Network Layer)
* Internet Control Message Protocol (Network Layer)
* Internet Group Management Protocol (Network Layer)
* Transmission Control Protocol (Transport Layer)
* User Datagram Protocol (Transport Layer)
* Domain Name System (Application Layer)
* Hypertext Transfer Protocol (Application Layer)

So “Protocol Analyst” is a simple tool which tries to fulfill most of the aspects of good Protocol Analyser.

The analyser has capability to filter the packets as the user so desires which are printed in separate file.

## **Possible Future Extensions**

* Increase in number of Protocols at each layer that can be analysed.
* Extension of application to capture packets from other interfaces such as Wifi, Bluetooth.
* Extension to Windows or Other Operating Systems
* Provision of Lightweight customisable Graphical User Interface

**Appendix**

Research Paper, Books & References Reviewed

1. Network Traffic Analysis and Intrusion Detection using Packet Sniffer  
   <http://ieeexplore.ieee.org/document/5437681/>
2. Review of Sockets Used for Communication Purpose  
   International Journal of Advanced Research in Computer Science and Software Engineering (Volume 3, Issue 3, March 2013) ISSN: 2277 128X  
   <https://www.ijarcsse.com/docs/papers/Volume_3/3_March2013/V3I3-0155.pdf>
3. James F. Kurose, Keith W. Ross, “Computer Networking: A Top-Down Approach”
4. Unix-Socket-FrequentlyAskedQuestions <http://www.softlab.ntua.gr/facilities/documentation/unix/unix-socket-faq/unix-socket-faq.html>
5. What Are Packet Sniffers And How Do They Work?  
   <https://www.lifewire.com/what-is-a-packet-sniffer-2487312>
6. Packet Sniffing Basics  
   <http://www.linuxjournal.com/content/packet-sniffing-basics>
7. Packet Sniffers - The Tech-FAQ  
   <http://www.tech-faq.com/packet-sniffer.html>
8. Sniffing (network wiretap, sniffer)  
   [http://cs.baylor.edu/~donahoo/tools/sniffer/sniffingFAQ.htm](http://www.tech-faq.com/packet-sniffer.html)