Department Of Computer Science & Information Systems

Computer Networks Mini-Project

Packet Capture and Analysis Tool for Transport Stream Based Classification

Under the supervision of

Prof. Dr. Rahul Banerjee

CS F303 - Computer Networks

Submitted By

GROUP 11

Rishabh Garg (2014A7PS065P)

C Hemanth (2014A7PS075P)

M Sharat Chandra (2014A7PS108P)

Birla Institute of Technology & Science

Pilani

Table of Contents

List of Figures

List of Tables

A. Problem Statement

" **Design and Implementation of a simple Packet-Capturing and Analysis Tool with on-demand Stream Classification that would be able to capture application, transport, network and data link level structured PDUs and enable its simple visual analysis.** "

B. Software Requirement Specifications

1.0 Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the Packet Capture and Analysis Tool. It will elucidate the purpose and features of the system, the interfaces of the system, what the system is capable of, the constraints under which it must operate, scalability of the system and how the system will react to external stimuli. The document is intended for all of the developers and any other stakeholders involved in the mini-project along with the Professors and Teaching Assistants of the course CS F303 Computer Networks, BITS Pilani.

1.2 Scope

The final software developed will be a Tool to Capture Packets and Analyse for Transport Stream Based Classification. This Tool intends not only to simply capture the packets but also to provide a Graphical User Interface (GUI) which would facilitate the analysis and visualisation of the traffic generated at a network host. This being its primary purpose it also seeks to explain the how it can classify any particular capture stream on the basis of transport layer protocol.

1.3 Definitions, Acronyms and Abbreviations

Packet: A network packet (referred to as packet in our document) is a formatted unit of data.
Transport Stream: Stream of a particular transport layer protocol.
Network Host: A network host is a computer or other device connected to a computer network.
GUI: Graphical User Interface
GCC: GNU Compiler Collection
IETF: Internet Engineering Task Force
NIC: Network Interface Controller
Protocol Abbreviations:
HTTP: Hypertext Transfer Protocol
DNS: Domain Name System
TCP: Transport Control Protocol
UDP: User Datagram Protocol
IP: Internet Protocol
ICMP: Internet Control Message Protocol
ARP: Address Resolution Protocol

1.4 References

IEEE. *IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.* IEEE Computer Society, 1998.

1.5 Overview

The next section, the Overall Description section of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next section.

The third section, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product. Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language.

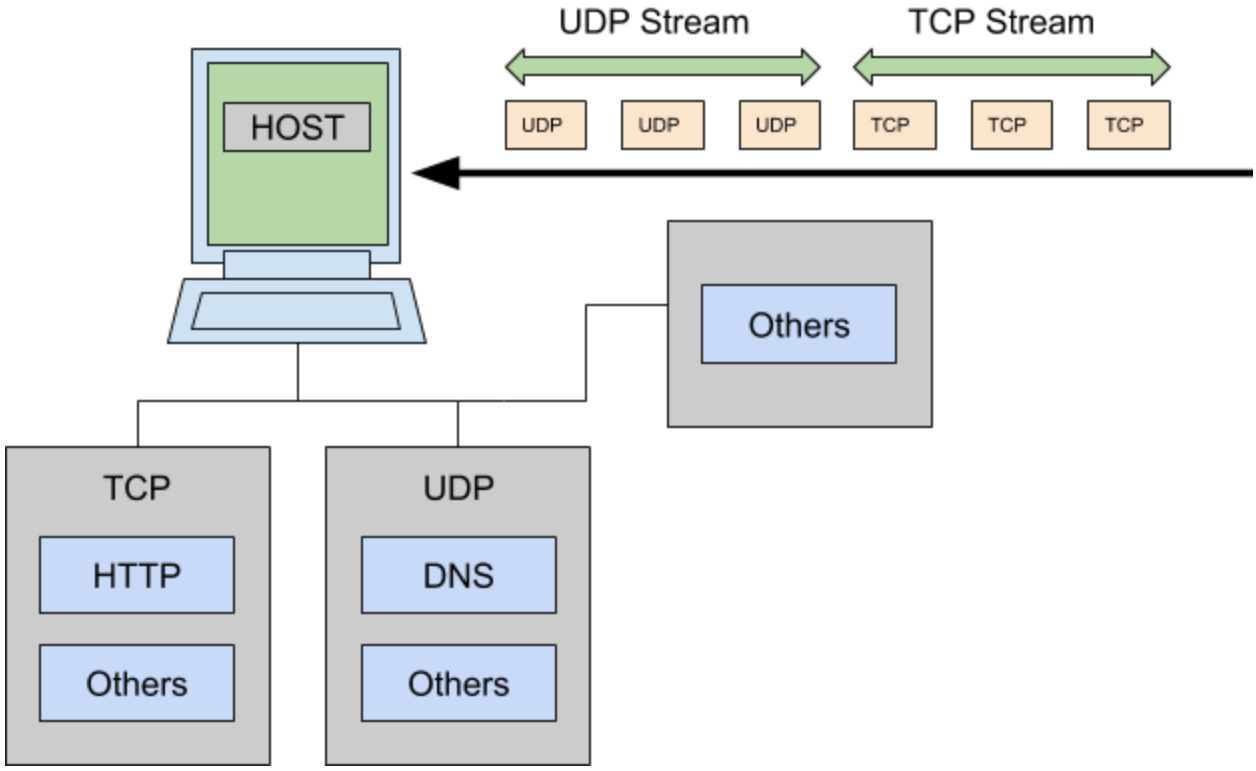

Figure 1: Overview of the working scenario of our application

2.0 Overall Description

2.1 Product Perspective

This product is a simple tool to capture packets and identify the broad type of packet among the common classifications of packets based on the protocols in a simple GUI, in contrast to Wireshark, which is a more

complicated and comprehensive tool that assimilates each and every packet and provides all accessible information to the end user.

2.1.1 System Interfaces

N/A

2.1.2 User Interfaces

The user facing end of the tool consists of a GUI which will enable the visual analysis and inspection of the captured packets. Since many packets keep flowing in the system, an option is provided to freeze last 10 packets on the GUI to analyse them. There is a section towards the bottom of the GUI, which keeps counts of packets segregated on the basis of various protocols.

2.1.3 Hardware Interfaces

A network socket is an internal endpoint for sending or receiving data at a single node in a computer network. Concretely, it is a representation of this endpoint in the networking software (protocol stack), on the base of which the Network Interface Controller(NIC) resides. The packets meant to reach the host enter through the NIC(Physical Layer) for further processing. Also, all the traffic generated by the host is pushed into the network through the Network Interface Card.

2.1.4 Software Interfaces

The application uses the standard socket Application Programming Interface provided by linux. This interface allows the application to create sockets to listen and capture the incoming network packets.

2.1.5 Communications Interfaces

The application uses raw sockets to capture all types of protocols. Thus, no specific protocol is supported or utilized by the application technically. That being said, the application decodes packets of certain protocols and presents them on a GUI for analysis purposes.

2.1.6 Memory Requirements

The primary memory requirements are estimated to be around 4MB and the secondary memory usage depends on the the time of execution as data in log files goes on increasing.

2.1.7 Operations

The default operation is to capture the incoming packets and segregate them on the basis of protocol headers. In addition to this, the user interface also provides an option to initiate a capture which will identify the dominant stream of traffic with respect to the transport layer.

2.1.8 Site Adaptation Requirements

No Site Adaptations are required for the Tool.

2.2 Product functions

The application captures the packets, segregates the packets into some common pre-specified classifications and displays the packet visually using a Graphical User Interface.

The application classifies the packet into the broad types of packets:

a) Application Layer
   i) HTTP
   ii) DNS
b) Transport Layer
   i) TCP
   ii) UDP
c) Internet Layer
   i) IP
   ii) ICMP
d) Data Link Layer
   i) ARP

2.3 User characteristics

The user is expected to be able to operate a system(computer) and have a knowledge of network sensitive applications to be able to generate packets. Knowledge of network protocol stack and layers associated with various commonly used protocols is also required.

The developer should have an understanding of operating system concepts and network programming to efficiently intermingle CPU and I/O bound jobs and capture packets.

2.4 Constraints

The advancement of the application is limited by the following constraints:

a) The scope and depth of the specifications mentioned by the course.
b) Lack of proper application based expertise in the subject.

2.5 Assumptions and dependencies

While developing the application it is assumed that:

1. The client is connected to a network and has a working connection before starting the application.
2. Operating system being used by the client is linux or one of its flavours.

2.6 Apportioning of Requirements

N/A

3.0 Specific requirements

3.1 External Interfaces

Name of Item: Packet
Description of Purpose: Input Information
Source of Input: Network or Host
Valid Range, Accuracy and/or Tolerance: No Range of packet size or type
Units of Measure: Size in octets of bits
Timing: No Specific Timing
Relationships to other inputs/outputs: Provides information about output data to be displayed in the GUI
Screen Formats: Standard application screen
Window Formats: A GUI which has user interactivity to display and analyse packets
Data Formats: Data should in the form of any valid packet as specified by IETF standards
Command Formats: Appropriate commands in the GUI will be present to stop/restart the analysis of packets and to close the application
End Messages: The number of packets of that type will be updated accordingly

3.2 Functions

3.2.1 List of specific functions

1. Main
    a. Inputs: None
    b. Outputs: Integer: 0 if successful execution, else -1
    c. Function: Creates Raw Sockets
2. ProcessPacket
    a. Inputs: Buffer, BufferSize
    b. Outputs: None
    c. Function: Identifies type and calls appropriate functions
3. WritetoLog
    a. Inputs: Buffer,BufferSize
    b. Outputs: None
    c. Function: Writes the buffer to output log file

4. ProcessDNS
    a. Inputs: Buffer, BufferSize
    b. Outputs: Struct populated with formatted headers
    c. Function: Identifies DNS Application Header

The following functions behave like ProcessDNS by identifying relevant headers.

5. ProcessHTTP
6. ProcessUDP
7. ProcessTCP
8. ProcessIP
9. ProcessICMP
10. ProcessEthernet

3.2.2 Error Handling

Appropriate handling of possible errors has been done in our software.

3.3 Performance Requirements

The number of packets captured is dependent on the traffic generated at the nde and also on the network speed. Typically, this is dependent on the nature of user activity which is the primary source of packet generation.

3.4 Logical Database Requirements

The packet data is captured and stored in a properly formatted format in the secondary storage in the form of a log file.

3.5 Design Constraints

The packet capturing is dependent on the host's network connectivity speed.

3.5.1 Standards compliance

N/A

3.6 Software System Attributes

3.6.1 Reliability

The factors required to establish the reliability of the application are a stable Linux system which provides a socket API and it being connected to a network which supports TCP/IP stack.

3.6.2 Availability

The host should be running Linux or any of its flavours.

3.6.3 Security

PCAT doesn't have provisions of security for the analysis and storage of data.

3.6.4 Maintainability

The application can be maintained very easily and does not pose any specific requirements for its maintenance.

3.6.5 Portability

The application is portable to any system running Linux or any of its flavours and should have GCC 5.4.0.

3.7 Organizing the Specific Requirements

3.7.1 System mode

The interfaces and performance are independent of the mode of the system.

3.7.2 User class

The application requires that the user in the Linux-based system to have elevated permissions.

3.7.3 Objects

1. Packets:- The network packets generated by the user will be captured in the application.
2. User:- The GUI provides user the capability to analyse the captured packets.

3.7.4 Feature

The application provides a complete interface to view the protocol headers and data as a packet enters the network stream. It also identifies the stream (UDP or TCP) for an initiated capture.

3.7.5 Stimulus

The application can encounter different types of packets as well as any commands from the User to stop analysis of packets or close the application as stimuli.

3.7.6 Response

The application captures, analyses and displays information about the packet according to appropriate stimuli. It also stops the analysis when required and closes the application appropriately.

3.7.7 Functional hierarchy

N/A

3.8 Additional Comments

N/A

4.0 Supporting information

4.1 Appendices

Type	Variable	Hex Code
Ethernet Loopback packet	ETH_P_LOOP	0x0060
Internet Protocol packet	ETH_P_IP	0x0800
Address Resolution packet	ETH_P_ARP	0x0806

Table 1: Ethernet Protocol ID's used in our application

C. Weekly Work Schedule

The proposed weekly schedule is as detailed below.

Week 1	Preliminary Literature Analysis
Week 2	Study of various protocols and their implementations
Week 3	Implementing a preliminary capture tool using raw sockets
Week 4	Decoding headers of various protocols
Week 5	Implementing Stream based Classification
Week 6	Developing GUI

Modular Ownership

Rishabh Garg:

1. Development of Graphical User Interface
2. Decoding Application layer Protocols

C Hemanth:

1. Integration of Information into the GUI
2. Decoding Transport Layer Protocols

M Sharat Chandra:

1. Stream based Classification
2. Decoding Network and Data Link Layer Protocols