Development of Java Based SDK for Network Protocol Analysis

Shridhar S Chini[1] Prof. Suma B. R[2] Suresh Aguru[3]
[1]M. Tech Student [2]Associate Professor [3]Monosek Lead
[1,2]R V College of Engineering, India [3]Nihon Communication Solutions Private Limited

Abstract— A protocol analyzer is a tool (software) used to capture and analyze signals and data traffic over a communication channel. Such a channel varies from a local computer bus to a satellite link that provides a means of communication using a standard communication protocol. Each type of communication protocol has a different tool to collect and analyze the data. A protocol analyzer (also known as a network analyzer, packet analyzer) is a computer program that can intercept and log traffic that passes over a digital network or part of a network as data streams flow across the network, the sniffer captures each packet and, if needed, decodes the packet's raw data, showing the values of various fields in the packet, and analyzes its content according to specifications.

Key words: Software Development Kit, Packet Analysis, Dialup / Ethernet Connection, Network Processor

I. INTRODUCTION

Security is at the forefront of most networks so as administration of the network, so it's necessary to analyze, debug, maintain and monitor the local networks and Internet connections. Network analysis is the process of listening to and analyzing network traffic. It also offers an insight into network communications to identify performance problems, locate security breaches, analyze application behavior, and perform capacity planning. Network analysis (aka "protocol analysis") is a process used by IT professionals who are responsible for network performance and security.

Network analysis is not brain surgery. Anyone can analyze network communications. However, need to acquire three basic skills to be a network analyst who can spot the cause of performance problems, evidence of breached hosts, misbehaving applications or the impending overload of the network.

1) A solid understanding of TCP/IP communications.
2) Comfort using Monosek.
3) Familiarity with packet structures and typical packet flows.

Depending on the network structure one can capture all or just parts of the traffic from a single machine within the network. However, there are many methods to avoid traffic narrowing by switches to gain access to traffic from other systems on the network. Our project focuses to presents the results of network analysis in a convenient and easily understandable format. And also allows us to defragment and reassemble network packets into streams. The program can fully decode and analyze network traffic [1] based on a number of different low level Internet protocols: ARP, DHCP, DNS, ICMP, IGMP, IP, IPv6, TCP, UDP and VLAN. It also performs full reconstruction of top-level protocols such as HTTP, SMTP, POP, FTP, and others under Windows and Linux platform and its use for support for Real time/Forensic analysis.

A. Working of network protocol analyzer

On wired broadcast LANs, such as Ethernet, Token Ring, and FDDI networks, depending on the network structure (hub or switch) one can capture traffic on all or parts of the network from a single machine on the network. However, some methods avoid traffic narrowing by switches to gain access to traffic from other systems on the network (e.g., ARP spoofing). For network monitoring purposes [2], it may also be desirable to monitor all data packets in a LAN by using a network switch with a so-called *monitoring port* that mirrors all packets that pass through all ports of the switch when systems are connected to a switch port. To use a network tap is an even more reliable solution than to use a monitoring port, since taps are less likely to drop packets during high traffic load.

On wired broadcast and wireless LANs, to capture traffic other than unicast traffic to the machine running the sniffer, multicast traffic to a multicast group that machine is monitoring, or broadcast traffic, the network adapter capturing the traffic must be in promiscuous mode. Some sniffers support this, but not all. On wireless LANs[3], even if the adapter is in promiscuous mode, packets not for the service set the adapter is configured for are usually ignored. To see those packets, the adapter must be in monitor mode

When traffic is captured, either the entire contents of packets are recorded, or the headers are recorded without recording the total content of the packet. This can reduce storage requirements, and avoid legal problems, yet provide sufficient information to diagnose problems.

Captured information is decoded from raw digital form into a human-readable format [4][5] that lets users easily review exchanged information. Protocol analyzers vary in their abilities to display data in multiple views, automatically detect errors, determine root causes of errors, generate timing diagrams, and reconstruct TCP and UDP data streams.

B. Motivation

Internet has become the information medium of first Resort each host on the network faces different threats in this environment like Malicious code, Denialofservice attacks, Attempts to hijack a machine.

The motivations to do this project are as follows:

- Capture the every packet passing through the Ethernet connection from as many different levels of the network stack: network, transport and application, as practical, process the packet and display it in readable form
- Provide supervise functionality, detect and prevent potential problems, develop effective counter measures for anomalies and sabotage
- Allow user to analyse, debug, maintain and monitor the local networks and Internet connections.

C. Objectives

Objective of the project is to build network protocol analyzer to process and captures the data passing through your dial-up connection or Ethernet network card, analyses this data and then represents it in an easily readable form for analyzing and monitoring the network.

This Network Protocol Analyzer will be useful tool for network administrators, security specialists, network application developers and anyone who needs a comprehensive picture of the traffic passing through their network connection or segment of a local area network.

II. MODEL

A. Overview of the design

Figure 1 shows the Data-Flow Diagram (DFD) Level 0, it shows the detailed visualization of flow of information in and out of each module. When some input is given to a processing module, some computation is carried out. The result is produced with the help of given input.

Level-0 is more general and high level view of input and output. The Level-0 DFD is also known as context diagram. Here the basic level data flow is shown. Here only the main module is included. Input given to the system at this level plays a vital role in understanding the nature of primary input set required for the system. Then the output produced in this Level-0 DFD is the final output and it remain same at all the levels.

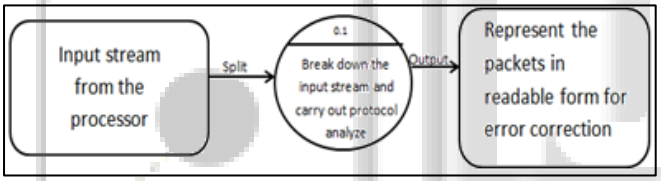

Fig. 1: Data Flow Diagram -Level 0

From the above figure it is obvious that network protocol analyser accepts the input data stream from the network processor and converts into suitable format that is easy to process and finally break down the input stream, assign values to the library variables and display input stream in readable format for monitoring and maintenance.

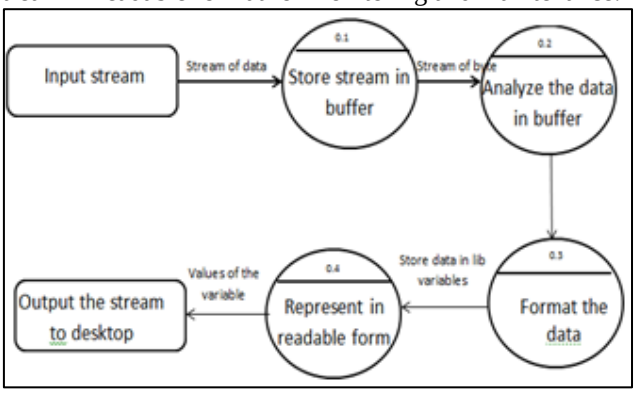

Fig. 2: Data Flow Diagram -Level 1

Figure 2 shows the Data-Flow Diagram (DFD) Level 1, it shows the immediate next sub-module required for computation. In the below figure the next module being explored are a set of independent sub-processes. These sub-processes run in series where output of one will be given as input to next module. The input data streams which are received from the network processor in string format that is accepted by network analyzing algorithm are stored in the network processor buffer and this data is passed to analyzer after analysis, the data formatted to represent it in human readable form.

Figure 3 shows the Data-Flow Diagram (DFD) Level 2, it shows all the sub-modules are explored. Also, the flow of data between each module is clearly explored that is Once the input stream is received it been broken into byte by byte and converted into integer value instead of hexadecimal and these values are stored in library variables.

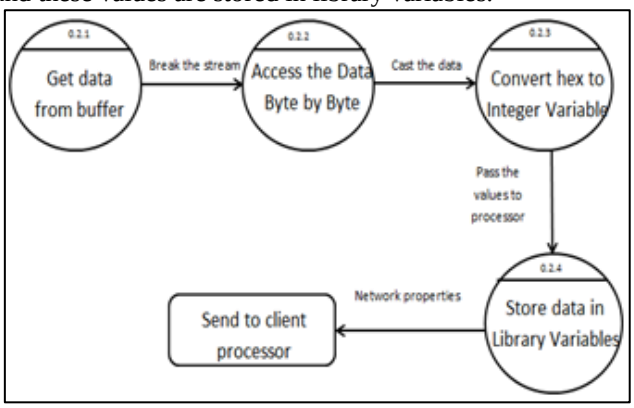

Fig. 3: Data Flow Diagram -Level 2

The process of filtering the packets where rules are applied from the rule base on the incoming packets and the packets are captured from the network traffic. If there is any mismatch that contradicts with the rule base then alarm is raised to the network administrator thereby indicating error in receiving the packets.

The working of the proposed system is based on capturing the network packets, once packets are received these converted into text format and sent to processing module as a string and this string is received by processor byte by byte and stored in library variables.

B. Workflow of RMA/RFID label generator

The workflow of the network protocol analyzer is depicted in Figure 4, This paper "Development of Java Based SDK for Network Based Protocol Analysis" is mainly aimed to develop the network processing tool. Starting from sniffing the network packets till the validation of it has been taken care. Here we have automated the logging part through a JAVA program. Whenever packets are transmitted from a system, Monosek will be invoked and start capturing the network packets. That will be stored automatically. To validate the contents the logic has been implemented to check particular pattern of packets or any specific string. These packets will be converted into a text format so that the validation can be accomplished through parsing the entire Monosek log. Based upon the parsing logic, pass/fail verdict will be indicated to user. The logic can always be extended depending upon the project requirements or a protocol and display these packets in readable form for a user/network admin.

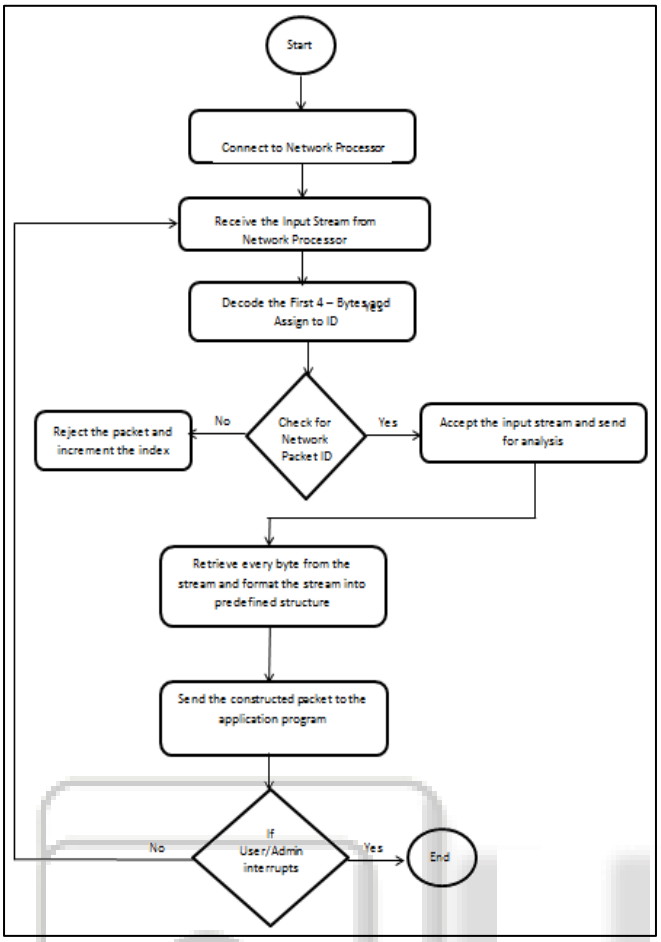

Fig. 4: Workflow of network protocol analyzer.

The next section discusses the implementation of various features of the Performance Analyzer.

III. IMPLEMENTATION

Any project, when it is ready to be implemented, many decisions must be taken considering the scalability, maintainability of the project. A wise decision will help to maintain the project with minimal efforts, as the requirements keep changing. A wrong selection of the programming language or design style or architecture will make the project code less adaptable to changes.

During the implementation of the SUPL server, similar decisions were made on the selection of the programming language, coding styles and platforms. This chapter gives the factors considered while making these decisions.

A. Programming Language Selection

The factors considered while selecting the programming language are, the ease of framing the protocol data units, the support provided by the language to implement security, the level of skills demonstrated by the people working on the project in a particular programming language.

The JAVA programming language provides the ability to build various data structures which are very useful in constructing the protocol data units. Using the class keyword we can construct custom data structures required to meet the needs of the project. Also the JAVA library can be extended to accommodate the security libraries. These security features are necessary to encrypt the data as it travels through the network. Considering all these factors, the language selected for development of the project is JAVA. This choice makes the project portable across different platforms with minimal or no configuration changes required.

B. Platform selection

The Monosek server design requirement states that the server should be available in the form of an exe file which will be easy to use across different teams. The visual studio is used with Install Shield to create an executable file of the server. This exe file can be passed to different teams who can execute the server without needing to build the project. The ease with which the development team can adapt to the platform and how quickly can the team leverage the features provided by the plat form is also taken into consideration. Considering these requirements, the platform selected for development is LINUX. However, as the programming language selected is C, the project can also be run on any other UNIX based system.

C. Integrated development environment (IDE)

Monosek - 1 enables Developers to access analysed packet information as well as raw packets in real time. This is extremely useful for Developer's as they can develop JAVA programs to work on real-time packets, Monosek - 1 provides for dynamic filters so that depending on how network traffic is behaving, the interfacing JAVA program can add, modify or disable any/some of the filters. This will help Developer's to dynamically vary network characterization parameters depending on results. Monosek - 1 operates in stealth mode. So there is absolutely no traffic injected or responded to or modified by Monosek - 1. This ensures that the analysis carried out on the network traffic does not reach to any wrong conclusions due to artificial additional network traffic created by the analyser itself. It also ensures that the card itself does not fall victim to network attacks. Using GCC based compiler the software is compiled and executed in Linux platform.

IV. EXPERIMENTAL EVALUATION

A. Test Environment

In most environments, before trying any of the practical research and tests proposed in this paper you must have written authorization to operate and play games over the tested systems and network. The same kind of authorization must be obtained to use the sniffing techniques required to analyze the network protocol behavior. For this reason it is recommended to have your own network and configure an isolated lab for all the proposed tests.

The packet analysis algorithm is tested under control environment with necessary security measures taken such that any ARP attacks initiated in the network does not affect the working of other system connected to the same LAN network connection. The host under testing is installed with Monosek server with valid license for capturing of data packets and the network interface card of the host machine is enabled with promiscuous mode such that it can sniff all the traffic through the network. Monosek - 1 operates in

stealth mode. So there is absolutely no traffic injected or responded to or modified by Monosek - 1. This ensures that the analysis carried out on the network traffic does not reach to any wrong conclusions due to artificial additional network traffic created by the analyzer itself. It also ensures that the card itself does not fall victim to network attacks.

B. Results

Experimental analysis refers to systematic evaluation carried out to find out specific features and behavior of the system. Experimental dataset consists of the different kind of data which is tested against the various network protocol analyzers that the system designed for. The various protocols analyzers used in the network protocol analyzer tool are listed below:

1) General analysis
2) TCP analysis
3) UDP analysis
4) SMTP analysis
5) POP3 analysis
6) HTTP analysis
7) Layer wise analysis

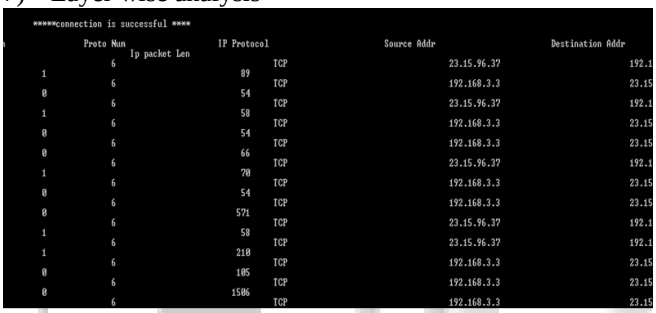

Fig 5: TCP analysis of the network stream

Figure 5 shows TCP analysis of the data stream where only tcp packets are captured and there attributes like packet number, packet length, protocol used and source and destination address are displayed.

Fig. 6: layer wise analysis of the network stream.

Figure 6 shows layer wise analysis of the data stream where every packets from dialup connection are captured and there attributes like packet number, packet length, protocol used and source and destination address are displayed layer wise.

Figure 7 shows the user interface of the network protocol analyzer tool where he can monitor and maintain the network

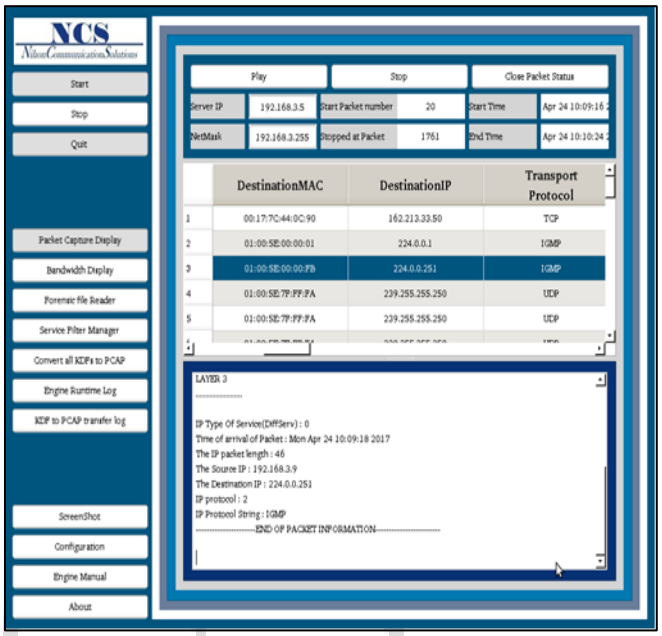

Fig. 7: Representation of the layer wise analysis at front end

V. CONCLUSION AND FUTURE WORK

Packet analyzer is not just a monitoring tool. It can be used for traffic analysis, troubleshooting and other useful purposes. When computers communicate over networks, they normally just listen to the traffic specifically for them. However, network cards have the ability to enter promiscuous mode, which allows them to listen to all network traffic regardless of if it's directed to them. Packet sniffers can capture things like clear-text passwords and usernames or other sensitive material. Sniffing is possible on non-switched and switched networks .There are many available tools used to capture network traffic that researcher used in their work, but there is a limitation in their work. Some tools only capture network traffic without analysis, therefore the researcher have to use another tools for analysis to get the traffic feature like it is need in his work. Some tools have large memory requirement. So we can design a tool that capture network traffic and analyze it and allows user to take only the feature as he need and store it in file to use it later in his work, then this will reduce the complexity and memory that is used to store the data.

ACKNOWLEDGMENT

We thank the Nihon Communication Solution Private Limited team members for their feedback in improving the tool and the managers for their support. Also, we thank the project committee, Dept. of CSE, R. V. College of Engineering for their timely guidance.

REFERENCES

[1] Xiaofan Lu,Weijia Sun and Shuangsheng Fu, "Network Protocol Analysis System Design and Implementation on Winpcap", Shenyang NormalUniversity,Vol.4,pp.514-516,2010.
[2] Meihua Xu, Zhenqi Wang, Xiujuan Han, "Capture Packets by Using WinPcap Technology", Network security technology and application of China,Vol.8, pp.31-33,2005
[3] Marcus J. Ranum, Kent Landfield, Mike Stolarchuk, Mark Sienkiewicz, Andrew Lambeth, and Eric Wall, "Implementing a Generalized Tool for Network Monitoring", LISA 97, San Diego, CA, October 26-31, 1997.
[4] F. Risso, L. Degioanni, "An Architecture for High Performance Network Analysis," in Proc. eh IEEE Symposium on Computers and Communications,pp. 1-15, 2001
[5] Vipin M, Srikanth S, "Analysis of Open Source Drivers for IEEE 802.11 WLANs", IEEE conference proceeding of ICWCSC 2010
[6] S.Suri, V. Batra, "Comparative Study of Network Monitoring Tools", in IJITEE, pp. 63-65, 2012.

