Remote Monitoring System for Network Management

Project Report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology in

Computer Science & Engineering

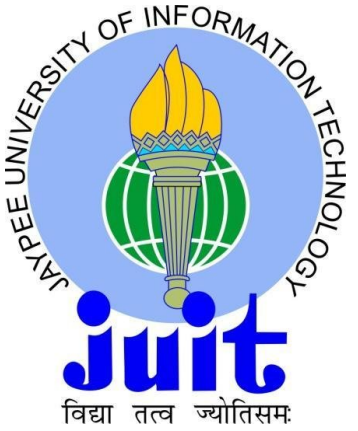
under the Supervision of

Mr. Punit Gupta

By

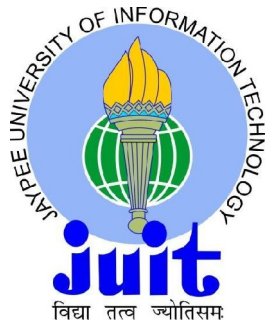
Anvit Sharma(111214)

to



JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

Wakanaghat, Himachal Pradesh, India



Jaypee University of Information Technology.
Waknaghat, Solan, HP

To Whom It May Concern,

This is to certify that project report entitled *Remote Monitoring System for Network Management*, submitted by Anvit Sharma, enrollment number 111214, in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan is being carried out under my supervision. This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date:

Supervisor's name: Punit Gupta
Designation: Assistant Professor

ACKNOWLEDGEMENT

I wish to express my gratitude to all those individuals who has contributed their ideas, time and energy to make this report a success. The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts. An understanding of the study like this, is never the outcome of the effort of a single person, rather it bears the imprint of a number of people who directly or indirectly helped me in completing this project. I would be failing in my duty if I don't say a word of thanks to all those people.

I would like to express my deepest gratitude to Mr. Punit Gupta, Senior Professor, JUIT for providing me with an opportunity to work on this project. It is my present privilege to acknowledge my profound gratitude and indebtedness toward him for his inspiration, constructive criticism & invaluable suggestion.

Finally, I would also like to thank my family and friends for their support throughout the project.

Anvit Sharma
111214, CSE
JUIT

TABLE OF CONTENTS

ABSTRACT	*1*
CHAPTER 1: GENERAL DEFINITIONS RELATED TO PROJECT	*4*
1.1 NETWORK MANAGEMENT	*5*
1.2 RMON	*8*
1.3 NETWORK MONITORING	*9*
1.4 INTERNET PROTOCOL (IP)	*10*
1.5 UDP(USER DATAGRAM PROTOCOL)	*13*
CHAPTER 2: LITERATURE REVIEW AND PREVIOUS WORK	*15*
CHAPTER 3: PROJECT RELATED DEPENDENCIES	*18*
3.1 PCAP	*19*
3.2 THE JPCAP LIBRARY	*20*
3.3 WINPCAP AND LIBPCAP	*26*
3.4 XAMPP	*27*
3.5 VMWARE	*29*
3.6 HIGHCHARTS	*30*
3.7 NETBEANS	*32*
3.8 JAVA (32 BITS)	*34*
CHAPTER 4: TOOLS AND TECHNIQUES	*36*
CHAPTER 5: RESULTS AND CONCLUSION	*46*
5.1 INTERPRETATION OF THE GRAPHS	*50*
5.2 CONCLUSION AND FUTURE DEVELOPMENT	*52*
REFERENCES	*53*

LIST OF FIGURES AND TABLES

Figure 1: IP Datagram format	*12*
Figure 2: Code obtaining the list of network interfaces	*22*
Figure 3: Code opening the network interfaces	*23*
Figure 4: Parameters when calling the JpcapCaptor.openDevice() method	*23*
Figure 5: Code for printing the captured packets	*24*
Figure 6: Code for printing the captured packets	*25*
Figure 7: Screenshot of XAMPP control panel	*28*
Figure 8: Screenshot of VMware software	*29*
Figure 9: Screenshot of Logarithmic chart	*31*
Figure 10: Screenshot of Netbeans IDE	*33*
Figure 11: Architectural diagram of NMS	*37*
Figure 12: The icon of jdk 32 bits is shown	*38*
Figure 13: The icon of Winpcap and Jpcap libraries are shown	*38*
Figure 14: Installation process of XAMPP server	*39*
Figure 15: Snapshot of libraries to be included	*40*
Figure 16: Database connection established	*40*
Figure 17: Table name packet created	*41*
Figure 18: Table name packet created in database phpmyadmin	*41*
Figure 19: Code for packet capture running	*42*
Figure 20: Table named packet in database phpmyadmin being	*42*

updated	
Figure 21: Copied contents of Highcharts in C:\xampp\htdocs\Highcharts-4.1.5	*43*
Figure 22: Graph_1: php code for creating graph of Packet Number vs length. Similar is for other graphs	*43*
Figure 23: HTML doc for main page	*44*
Figure 24: main html document source code	*45*
Figure 25: Remote monitoring for network management homepage	*47*
Figure 26: Number of packets vs length of the packet	*48*
Figure 27: Number of packets vs Timestamp Count	*48*
Figure 28: Number of packets vs. IP packet version	*49*
Figure 29: Number of packets vs. id field	*49*

ABSTRACT

Network is a key part of a business's IT system. To keep it running smoothly it's important to perform some basic network management tasks.

Why computer network maintenance matters?

There are several reasons why good network management is important.

- Prevent problems: Much like servicing a car, good network management will stop problems occurring and prolong your network's life.
- Work efficiently: Good management ensures your staff have access to the IT they need to do their jobs effectively.
- Maintain security: Even if you set your network up securely to begin with, you need good management to ensure it stays that way.
- Stay up to date: Although network technology doesn't move as fast as other areas of business IT, careful upgrades may help improve performance.

A well-managed network will serve business better, reducing the time one spends solving problems and allowing to get on with running the company more efficiently.

There are five important areas of network management-

1. *Software management:* This involves taking care of the software installed on your network. It includes keeping track of installed software, applying any important updates and deciding whether to upgrade when new versions become available.

2. *Hardware management:* You need to maintain the physical equipment which makes up your network. This might involve inspecting servers, cleaning dust from vents and testing key hardware like uninterruptible power supplies and backup drives.

3. *File management:* Particularly important if you operate a central file store on a server, file management involves keeping all the files on your system organised, deleting temporary files and archiving old data so there's room to save new files.

4. *Security management: I*f your business relies on its computer network, then keeping that network secure is very important. Security-related tasks include running and testing backups, regularly scanning for viruses and testing your firewall.

5. *User management:* You can reduce administration and boost your network's security further by giving employees different levels of access depending on what they need to do, removing access rights when staff leave and controlling what files people can change.

It's important you stay on top of these things. Performed regularly, network management and maintenance needn't take a great deal of time or cause much disruption.

Keeping on top of computer network maintenance

To make sure network management and maintenance tasks are carried out regularly, make someone in your business responsible for them and build the tasks into your company's schedule. If you leave them until someone has a spare moment then they'll probably never get done!

To minimise disruption, perform some computer network maintenance out of normal working hours. Tasks like defragmenting hard drives, running virus scans or testing backup systems can slow down a network server.

You may also be able to automate some management tasks. For instance, virus scanners can be set to perform a full scan at a specific time.

In network management terms, *network monitoring* is the phrase used to describe a system that continuously monitors a network and notifies a network administrator though messaging systems (usually e-mail) when a device fails or an outage occurs. Network monitoring is usually performed through the use of software applications and tools.

At the most basic level, ping is a type of network monitoring tool. Other commercial software packages may include a network monitoring system that is designed to monitor an entire business or enterprise network.

Some applications are used to monitor traffic on your network, such as VoIP monitoring, video stream monitoring, mail server (POP3 server) monitoring, and others.

CHAPTER 1: GENERAL DEFINITIONS RELATED TO PROJECT

1.1 NETWORK MANAGEMENT

In computer networks, network management is the operation, administration, maintenance, and provisioning (OAMP) of networked systems. Network management is essential to command and control practices and is generally carried out of a network operations centre.

- Operation deals with keeping the network (and the services that the network provides) up and running smoothly. It includes monitoring the network to spot problems as soon as possible, ideally before users are affected.

- Administration deals with keeping track of resources in the network and how they are assigned. It includes all the "housekeeping" that is necessary to keep the network under control.

- Maintenance is concerned with performing repairs and upgrades—for example, when equipment must be replaced, when a router needs a patch for an operating system image, when a new switch is added to a network. Maintenance also involves corrective and preventive measures to make the managed network run "better", such as adjusting device configuration parameters.

- Provisioning is concerned with configuring resources in the network to support a given service. For example, this might include setting up the network so that a new customer can receive voice service, real time communications etc.

A common way of characterizing network management functions is FCAPS—Fault, Configuration, Accounting, Performance and Security.

Functions that are performed as part of network management accordingly include controlling, planning, allocating, deploying, coordinating, and monitoring the resources of a network, network planning, frequency allocation, predetermined traffic routing to support load

balancing, cryptographic key distribution authorization, configuration management, fault management, security management, performance management, bandwidth management, Route analytics and accounting management.

Data for network management is collected through several mechanisms, including agents installed on infrastructure, synthetic monitoring that simulates transactions, logs of activity, sniffers and real user monitoring. In the past network management mainly consisted of monitoring whether devices were up or down; today performance management has become a crucial part of the IT team's role which brings about a host of challenges—especially for global organizations. Network management does not include user terminal equipment.

A network management system (NMS) is a set of hardware and/or software tools that allow an IT professional to supervise the individual components of a network within a larger network management framework.

Network management system components assist with:

- Network device discovery - identifying what devices are present on a network.

- Network device monitoring - monitoring at the device level to determine the health of network components and the extent to which their performance matches capacity plans and intra-enterprise service-level agreements (SLAs).

- Network performance analysis - tracking performance indicators such as bandwidth utilization, packet loss, latency, availability and uptime of routers, switches and other Simple Network Management Protocol (SNMP) -enabled devices.

- Intelligent notifications - configurable alerts that will respond to specific network scenarios by paging, emailing, calling or texting a network administrator.

Network management is a broad range of functions including activities, methods, procedures and the use of tools to administrate, operate, and reliably maintain computer network systems. Strictly speaking, network Management does not include terminal equipment (PCs, workstations, printers, etc.). Rather, it concerns the reliability, efficiency and capacity/capabilities of data transfer channels.

While there is no precise definition of the term due to it being such a broad concept, some of the main areas are summarized below:

Network Administration: This involves tracking and inventorying the many network resources such as monitoring transmission lines, hubs, switches, routers, and servers; it also involves monitoring their performance and updating their associated software – especially network management software, network operating systems, and distributed software applications used by network users.

Network Operation: This involves smooth network functioning as designed and intended, including close monitoring of activities to quickly and efficiently address and fix problems as they occur and preferably even before users are aware of the problem.

Network Maintenance: This involves timely repair and necessary upgrades to all network resources as well as preventive and corrective measures through close communication and collaboration with network administrators. Example work includes replacing or upgrading network equipment such as switches, routers and damaged transmission lines.

Network Provisioning: This involves configuring network resources to support the requirements of a particular service; example services may be voice capabilities or increasing broadband requirements to facilitate more users.

1.2 RMON

Remote Monitoring (RMON) is a standard monitoring specification that enables various network monitors and console systems to exchange network-monitoring data. RMON provides network administrators with more freedom in selecting network-monitoring probes and consoles with features that meet their particular networking needs. An RMON implementation typically operates in a client/server model. Monitoring devices (commonly called "probes" in this context) contain RMON software agents that collect information and analyze packets. These probes act as servers and the Network Management applications that communicate with them act as clients. While both agent configuration and data collection use SNMP, RMON is designed to operate differently than other SNMP-based systems:

- Probes have more responsibility for data collection and processing, which reduces SNMP traffic and the processing load of the clients.
- Information is only transmitted to the management application when required, instead of continuous polling.

In short, RMON is designed for "flow-based" monitoring, while SNMP is often used for "device-based" management. RMON is similar to other flow-based monitoring technologies such as NetFlow and SFlow because the data collected deals mainly with traffic patterns rather than the status of individual devices. One disadvantage of this system is that remote devices shoulder more of the management burden, and require more resources to do so. Some devices balance this trade-off by implementing only a subset of the RMON MIB groups. A minimal RMON agent implementation could support only statistics, history, alarm, and event.

1.3 NETWORK MONITORING

In network management terms, network monitoring is the phrase used to describe a system that continuously monitors a network and notifies a network administrator though messaging systems (usually e-mail) when a device fails or an outage occurs. Network monitoring is usually performed through the use of software applications and tools.

At the most basic level, ping is a type of network monitoring tool. Other commercial software packages may include a network monitoring system that is designed to monitor an entire business or enterprise network.

Some applications are used to monitor traffic on your network, such as VoIP monitoring, video stream monitoring, mail server (POP3 server) monitoring, and others.

While an intrusion detection system monitors a network for threats from the outside, a network monitoring system monitors the network for problems caused by overloaded and/or crashed servers, network connections or other devices. For example, to determine the status of a webserver, monitoring software may periodically send an HTTP request to fetch a page. For email servers, a test message might be sent through SMTP and retrieved by IMAP or POP3.

Commonly measured metrics are response time, availability and uptime, although both consistency and reliability metrics are starting to gain popularity. The widespread addition of WAN optimization devices is having an adverse effect on most network monitoring tools -- especially when it comes to measuring accurate end-to-end response time because they limit round trip visibility.[1]

Status request failures - such as when a connection cannot be established, it times-out, or the document or message cannot be retrieved - usually produce an action from the monitoring system. These actions vary -- an alarm may be sent (via SMS, email, etc.) to the resident sysadmin, automatic failover systems may be activated to remove the troubled server from duty until it can be repaired, etc. Monitoring the performance of a network uplink is also known as network traffic measurement, and more software are listed there.

1.4 INTERNET PROTOCOL (IP)

The Internet Protocol (IP) is the principal communications protocol in the Internet protocol suite for relaying datagrams across network boundaries. Its routing function enables internetworking, and essentially establishes the Internet.

IP has the task of delivering packets from the source host to the destination host solely based on the IP addressing the packet headers. For this purpose, IP defines packet structures that encapsulate the data to be delivered. It also defines addressing methods that are used to label the datagram with source and destination information.

Historically, IP was the connectionless datagram service in the original Transmission Control Program introduced by Vint Cerf and Bob Kahn in 1974; the other being the connection-oriented Transmission Control Protocol (TCP). The Internet protocol suite is therefore often referred to as TCP/IP.

The first major version of IP, Internet Protocol Version 4 (IPv4), is the dominant protocol of the Internet. Its successor is Internet Protocol Version 6 (IPv6).

The Internet Protocol is responsible for addressing hosts and for routing datagrams (packets) from a source host to a destination host across one or more IP networks. For this purpose, the Internet Protocol defines the format of packets and provides an addressing system that has two functions: identifying hosts; and providing a logical location service. Each datagram has two components: a header and a payload. The IP header is tagged with the source IP address, the destination IP address, and other meta-data needed to route and deliver the datagram. The payload is the data that is transported. This method of nesting the data payload in a packet with a header is called encapsulation. IP addressing entails the assignment of IP addresses and associated parameters to host interfaces. The address space is divided into networks and sub-networks, involving the designation of network or routing prefixes. IP routing is performed by all hosts, but most importantly by routers, which transport packets across network boundaries. Routers communicate with one another via specially designed routing protocols, either interior gateway protocols or exterior gateway protocols, as needed for the topology of the network.

IP routing is also common in local networks. For example, many Ethernet switches support IP multicast operations. These switches use IP addresses and Internet Group Management Protocol to control multicast routing but use MAC addresses for the actual routing.

The design of the Internet protocols is based on the end-to-end principle. The network infrastructure is considered inherently unreliable at any single network element or transmission medium and assumes that it is dynamic in terms of availability of links and nodes. No central monitoring or performance measurement facility exists that tracks or maintains the state of the network. For the benefit of reducing network complexity, the intelligence in the network is purposely mostly located in the end nodes of data transmission. Routers in the transmission path forward packets to the next known, directly reachable gateway matching the routing prefix for the destination address.

As a consequence of this design, the Internet Protocol only provides best effort delivery and its service is characterized as unreliable. In network architectural language, it is a connectionless protocol, in contrast to connection-oriented modes of transmission. Various error conditions may occur, such as data corruption, packet loss, duplication and out-of-order delivery. Because routing is dynamic, meaning every packet is treated independently, and because the network maintains no state based on the path of prior packets, different packets may be routed to the same destination via different paths, resulting in out-of-order sequencing at the receiver.

Internet Protocol Version 4 (IPv4) provides safeguards to ensure that the IP packet header is error-free. A routing node calculates a checksum for a packet. If the checksum is bad, the routing node discards the packet. The routing node does not have to notify either end node, although the Internet Control Message Protocol (ICMP) allows such notification. By contrast, in order to increase performance, and since current link layer technology is assumed to provide sufficient error detection, the IPv6 header has no checksum to protect it.

All error conditions in the network must be detected and compensated by the end nodes of a transmission. The upper layer protocols of the Internet protocol suite are responsible for resolving reliability issues. For example, a host may cache network data to ensure correct ordering before the data is delivered to an application. The dynamic nature of the Internet and the diversity of its components provide no guarantee that any particular path is actually capable

of, or suitable for, performing the data transmission requested, even if the path is available and reliable. One of the technical constraints is the size of data packets allowed on a given link. An application must assure that it uses proper transmission characteristics. Some of this responsibility lies also in the upper layer protocols. Facilities exist to examine the maximum transmission unit (MTU) size of the local link and Path MTU Discovery can be used for the entire projected path to the destination. The IPv4 internetworking layer has the capability to automatically fragment the original datagram into smaller units for transmission. In this case, IP provides re-ordering of fragments delivered out of order.

The Transmission Control Protocol (TCP) is an example of a protocol that adjusts its segment size to be smaller than the MTU. The User Datagram Protocol(UDP) and the Internet Control Message Protocol (ICMP) disregard MTU size, thereby forcing IP to fragment oversized datagrams.

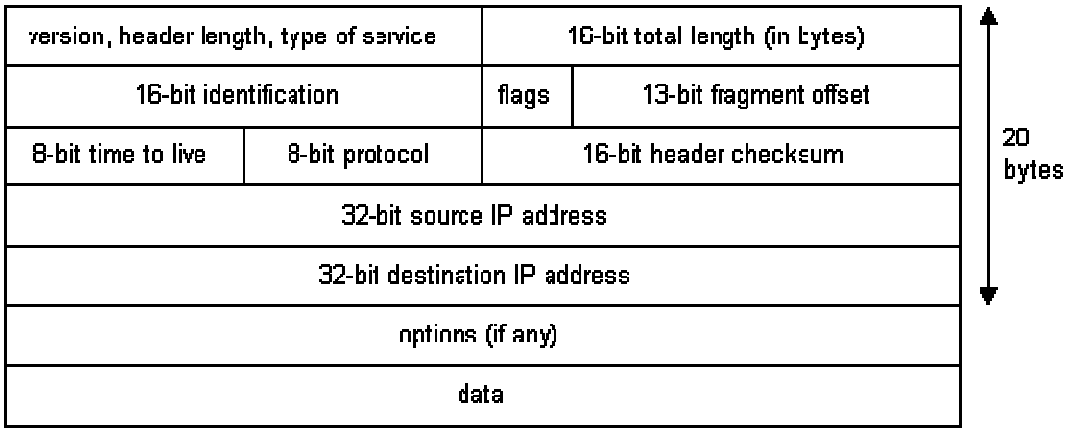


Figure 1: IP Datagram format

1.5 UDP(USER DATAGRAM PROTOCOL)

The User Datagram Protocol (UDP) is one of the core members of the Internet protocol suite. The protocol was designed by David P. Reed in 1980 and formally defined in RFC 768.

UDP uses a simple connectionless transmission model with a minimum of protocol mechanism. It has no handshaking dialogues, and thus exposes any unreliability of the underlying network protocol to the user's program. There is no guarantee of delivery, ordering, or duplicate protection. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram.

With UDP, computer applications can send messages, in this case referred to as datagrams, to other hosts on an Internet Protocol (IP) network without prior communications to set up special transmission channels or data paths. UDP is suitable for purposes where error checking and correction is either not necessary or is performed in the application, avoiding the overhead of such processing at the network interface level. Time-sensitive applications often use UDP because dropping packets is preferable to waiting for delayed packets, which may not be an option in a real-time system. If error correction facilities are needed at the network interface level, an application may use the Transmission Control Protocol (TCP) or Stream Control Transmission Protocol (SCTP) which are designed for this purpose.

A number of UDP's attributes make it especially suited for certain applications.

- It is transaction-oriented, suitable for simple query-response protocols such as the Domain Name System or the Network Time Protocol.
- It provides datagrams, suitable for modelling other protocols such as in IP tunnelling or Remote Procedure Call and the Network File System.
- It is simple, suitable for bootstrapping or other purposes without a full protocol stack, such as the DHCP and Trivial File Transfer Protocol.
- It is stateless, suitable for very large numbers of clients, such as in streaming media applications for example IPTV

- The lack of retransmission delays makes it suitable for real-time applications such as Voice over IP, online games, and many protocols built on top of the Real Time Streaming Protocol.
- Works well in unidirectional communication, suitable for broadcast information such as in many kinds of service discovery and shared information such as broadcast time or Routing Information Protocol.

  Applications use datagram sockets to establish host-to-host communications. An application binds a socket to its endpoint of data transmission, which is a combination of an IP address and a service port. A port is a software structure that is identified by the port number, a 16 bit integer value, allowing for port numbers between 0 and 65535. Port 0 is reserved, but is a permissible source port value if the sending process does not expect messages in response.

  The Internet Assigned Numbers Authority (IANA) has divided port numbers into three ranges.[2] Port numbers 0 through 1023 are used for common, well-known services. On Unix-like operating systems, using one of these ports requires super user operating permission. Port numbers 1024 through 49151 are the registered ports used for IANA-registered services. Ports 49152 through 65535 are dynamic ports that are not officially designated for any specific service, and may be used for any purpose. They also are used as ephemeral ports, from which software running on the host may randomly choose a port in order to define itself. In effect, they are used as temporary ports primarily by clients when communicating with servers.

CHAPTER 2: LITERATURE REVIEW AND PREVIOUS WORK

The literature referred for this project was mostly picked from research papers and pre-existing softwares based on Network management. Web based encyclopaedias were extensively used for the description of fundamental definitions related to the project. They were also used to understand certain concepts which were to be applied in the project. Literature regarding the project dependencies were read from their source sites. An extensive description of references will be provided in the references section of this report.

There is wide variety of literature available on Network management. There is a bevy of research papers and articles available on this topic. A systematic approach of reading relevant materials was adopted and a simultaneous conceptualization of the information gained was done via coding. A lot of time and resources were utilized to install and run project dependencies which are covered later in this report. Network management is a vast area and pinpointing the aspect to be developed was essential. In this project, it is endeavoured to capture UDP packets, store them into a database and analyze them thereby giving us details regarding the performance of the network.

Previous work was mostly aimed in understanding the nuances of the project problem and reading literature relevant to it. A code was also developed by establishing a client-server socket connection. This was later discarded and jpcap library was used instead to create a packet capturing application.

Existing software's on Network Management:

1. *OpenNMS:* Open NMS is an award winning network management application platform with a long track record of providing solutions for enterprises and carriers. OpenNMS can generate its own events or receive events from outside sources, such as SNMP Traps, syslog or TL/1. It is even easy to send custom events to OpenNMS: simply connect to a TCP port and, if you have permission, sent some XML-formatted text. OpenNMS can serve as the central repository for your network event stream. Able to handle bursts of thousands of events per second, OpenNMS also has a number of correlation methods to automatically clear events, translate one event into another, a reduce duplicate events into one alarm. OpenNMS was started during a time when Service Level Agreements (SLAs) were the focus of much management effort. The application comes with a large number of service monitors that perform synthetic transactions ranging from a simple ICMP request (ping) or port check, up through complex website monitoring and round trip e-mail testing. Detailed reports can be generated on the availability of the services,

and it is extremely easy to customize polling rates as well as to configure scheduled downtime.

2. *Paessler:* Adding to normal bandwidth monitoring capabilities based on SNMP, PRTG allows administrators to discern actual bandwidth usage based on multiple parameters, such as source and destination IP addresses, MAC addresses, port numbers, protocols, etc., using packet sniffing. Furthermore, PRTG's packet sniffing functionality can be used to generate top lists, which enable administrators to recognize detailed usage trends, sources and destinations of individual communications via the network, as well as the details of the traffic flowing within said network.

CHAPTER 3: PROJECT RELATED DEPENDENCIES

3.1 PCAP

In the field of computer network administration, pcap (packet capture) consists of an application programming interface (API) for capturing network traffic. Unix-like systems implement pcap in the libpcap library; Windows uses a port of libpcap known as WinPcap.

Monitoring software may use libpcap and/or WinPcap to capture packets travelling over a network and, in newer versions, to transmit packets on a network at the link layer, as well as to get a list of network interfaces for possible use with libpcap or WinPcap.

The pcap API is written in C, so other languages such as Java, .NET languages, and scripting languages generally use a wrapper; no such wrappers are provided by libpcap or WinPcap itself. C++ programs may link directly to the C API or use an object-oriented wrapper.

libpcap and WinPcap provide the packet-capture and filtering engines of many open source and commercial network tools, including protocol analyzers (packet sniffers), network monitors, network intrusion detection systems, traffic-generators and network-testers.

libpcap and WinPcap also support saving captured packets to a file, and reading files containing saved packets; applications can be written, using libpcap or WinPcap, to be able to capture network traffic and analyze it, or to read a saved capture and analyze it, using the same analysis code.

3.2 THE JPCAP LIBRARY

Introduction:

Jpcap is an open source library for capturing and sending network packets from Java applications. It provides facilities to:

- capture raw packets live from the wire.
- save captured packets to an offline file, and read captured packets from an offline file.
- automatically identify packet types and generate corresponding Java objects (for Ethernet, IPv4, IPv6, ARP/RARP, TCP, UDP, and ICMPv4 packets).
- filter the packets according to user-specified rules before dispatching them to the application.
- send raw packets to the network

Jpcap is based on libpcap/winpcap, and is implemented in C and Java. Jpcap has been tested on Microsoft Windows (98/2000/XP/Vista), Linux (Fedora, Ubuntu), Mac OS X (Darwin), FreeBSD, and Solaris.

Using Jpcap, you can develop applications to capture packets from a network interface and visualize/analyze them in Java. You can also develop Java applications to send arbitrary packets through a network interface. jpcap is a set of Java classes which provide an interface and system for network packet capture. A protocol library and tool for visualizing network traffic is included. jpcap hides the low-level details of network packet capture by abstracting many network packet types and protocols into Java classes. Internally, jpcap implements bindings to the libpcap system library through JNI (the Java Native Interface).

jpcap utilizes libpcap, a widely deployed shared-library for capturing user-level packets. libpcap must be installed on your system in order to use jpcap. jpcap consists of a small shared-library which wraps libpcap plus a collection of Java classes. The shared-library component provides event hooks, communication

and data conversion between a running Java VM and libpcap. The 'capture' package contains the core capture system.

The 'net' package contains abstractions for many network packet types and protocols. The 'simulator' package contains a network simulator. jpcap is licensed under the Mozilla Public License. Active jpcap development is taking place on Unix platforms. However, jpcap should run on any platform where libpcap is implemented.

Jpcap can be used to develop many kinds of network applications, including (but not limited to):

- network and protocol analyzers
- network monitors
- traffic loggers
- traffic generators
- user-level bridges and routers
- network intrusion detection systems (NIDS)
- network scanners
- security tools

Jpcap captures and sends packets independently from the host protocols (e.g., TCP/IP). This means that Jpcap does not (cannot) block, filter or manipulate the traffic generated by other programs on the same machine: it simply "sniffs" the packets that transit on the wire. Therefore, it does not provide the appropriate support for applications like traffic shapers, QoS schedulers and personal firewalls.

Library usage:

When you want to capture packets from a network, the first thing you have to do is to obtain the list of network interfaces on your machine. To do so, Jpcap providesJpcapCaptor.getDeviceList() method. It returns an array of NetworkInterface objects.A NetworkInterface object contains some information about the corresponding network interface, such as its name, description, IP and MAC addresses, and datatlink name and description. The following sample code obtains the list of network interfaces and prints out their information:

```
//Obtain the list of network interfaces
NetworkInterface[] devices =
JpcapCaptor.getDeviceList();

//for each network interface
for (int i = 0; i < devices.length; i++) {
  //print out its name and description
  System.out.println(i+": "+devices[i].name + "(" +
devices[i].description+")");

  //print out its datalink name and description
  System.out.println(" datalink:
"+devices[i].datalink_name + "(" +
devices[i].datalink_description+")");

  //print out its MAC address
  System.out.print(" MAC address:");
  for (byte b : devices[i].mac_address)
    System.out.print(Integer.toHexString(b&0xff) +
":");
  System.out.println();

  //print out its IP address, subnet mask and
broadcast address
  for (NetworkInterfaceAddress a :
devices[i].addresses)
    System.out.println(" address:"+a.address + " " +
a.subnet + " "+ a.broadcast);
}
```

Figure 2: Code obtaining the list of network interfaces

Once you obtain the list of network interfaces and choose which network interface to capture packets from, you can open the interface by using JpcapCaptor.openDevice()method. The following piece of code illustrates how to open an network interface:

```
NetworkInterface[] devices =
JpcapCaptor.getDeviceList();
int index=...;  // set index of the interface that
you want to open.

//Open an interface with openDevice(NetworkInterface
intrface, int snaplen, boolean promics, int to_ms)
JpcapCaptor
captor=JpcapCaptor.openDevice(device[index], 65535,
false, 20);
```

Figure 3: Code opening the network interfaces

When calling the JpcapCaptor.openDevice() method, you can specify the following parameters:

Name:	Purpose:
Network Interface	Network interface that you want to open.
int snaplen	Max number of bytes to capture at once.
boolean promics	True if you want to open the interface in promiscuous mode, and otherwise false. In promiscuous mode, you can capture packets every packet from the wire, i.e., even if its source or destination MAC address is not same as the MAC address of the interface you are opening. In non-promiscuous mode, you can only capture packets send and received by your host.
int to_ms	Set a capture timeout value in milliseconds.

Figure 4: Parameters when calling the JpcapCaptor.openDevice() method.

JpcapCaptor.openDevice() returns an instance of JpcapCaptor. You can then call several methods of the JpcapCaptor class to actually capture packets from the network interface.Once you obtain an instance of of JpcapCaptor, you can capture packets from the interface. There are two major approaches to capture packets using a JpcapCaptor instance: using a callback method, and capturing packets one-by-one.

Call back approach: In this approach, you implement a callback method to process captured packets, and then pass the callback method to Jpcap so that Jpcap calls it back every time it captures a packet. Let's see how you can do this approach in detail.
First, you implement a callback method by defining a new class which implements the PacketReceiver interface. The PacketReceiver interface defines a receivePacket()method, so you need to implement a receivePacket() method in your class.
The following class implement a receivePacket() method which simply prints out a captured packet:

```
class PacketPrinter implements PacketReceiver {
  //this method is called every time Jpcap captures
a packet
  public void receivePacket(Packet packet) {
    //just print out a captured packet
    System.out.println(packet);
  }
}
```

Figure 5: Code for printing the captured packets.

Then, you can call either JpcapCaptor.processPacket() or JpcapCaptor.loopPacket() methods to start capturing using the callback method. When calling processPacket()or loopPacket() method, you can also specify the number of packets to capture before the method returns. You can specify -1 to continue capturing packets infinitely. The two methods for callback, processPacket() and loopPacket(), are very similar. Usually you might want to use processPacket() because it supports timeout and non_blocking mode, while loopPacket() doesn't.

Capturing packets one-by-one: Using a callback method is a little bit tricky because you don't know when the callback method is called by Jpcap. If you don't want to use a callback method, you can also capture packets using the JpcapCaptor.getPacket() method. getPacket() method simply returns a captured packet. You can (or have to) call getPacket() method multiple times to capture consecutive packets. The following sample code also prints out captured packets:

```
JpcapCaptor
captor=JpcapCaptor.openDevice(device[index], 65535,
false, 20);

//call processPacket() to let Jpcap call
PacketPrinter.receivePacket() for every packet
capture.
captor.processPacket(10,new PacketPrinter());

captor.close();
```

Figure 6: Code for printing the captured packets.

3.3 WINPCAP AND LIBPCAP

WinPcap is the industry-standard tool for link-layer network access in Windows environments: it allows applications to capture and transmit network packets bypassing the protocol stack, and has additional useful features, including kernel-level packet filtering, a network statistics engine and support for remote packet capture.

WinPcap consists of a driver, that extends the operating system to provide low-level network access, and a library that is used to easily access the low-level network layers. This library also contains the Windows version of the well known libpcap Unix API.

Thanks to its set of features, WinPcap is the packet capture and filtering engine of many open source and commercial network tools, including protocol analyzers, network monitors, network intrusion detection systems, sniffers, traffic generators and network testers.

WinPcap is software that allows your network interface card to (NIC) operate in "promiscuous" mode. Normally if a NIC sees traffic addressed to another NIC on the network, it ignores it. If you are running a network sniffer application, you may have a need to capture that traffic for inspection. Putting a NIC in promiscuous mode allows your NIC to capture traffic addressed to another machine and pass it to the sniffer application.

Libpcap is a system-independent interface for user-level packet capture. libpcap provides a portable framework for low-level network monitoring. Applications include network statistics collection, security monitoring, network debugging, etc.

3.4 XAMPP

XAMPP is a free and open source cross-platform web server solution stack package, consisting mainly of the Apache HTTP Server, MySQL database, and interpreters for scripts written in the PHP and Perl programming languages.

XAMPP requires only one zip, tar, 7z, or exe file to be downloaded and run, and little or no configuration of the various components that make up the web server is required. XAMPP is regularly updated to incorporate the latest releases of Apache, MySQL, PHP and Perl. It also comes with a number of other modules including OpenSSL and phpMyAdmin.

Self-contained, multiple instances of XAMPP can exist on a single computer, and any given instance can be copied from one computer to another.It is offered in both a full, standard version and a smaller version.

Officially, XAMPP's designers intended it for use only as a development tool, to allow website designers and programmers to test their work on their own computers without any access to the Internet. To make this as easy as possible, many important security features are disabled by default.[2] In practice, however, XAMPP is sometimes used to actually serve web pages on the World Wide Web[citation needed]. A special tool is provided to password-protect the most important parts of the package.

XAMPP also provides support for creating and manipulating databases in MySQL and SQLite among others. Once XAMPP is installed, it is possible to treat a localhost like a remote host by connecting using an FTP client. Using a program like FileZilla has many advantages when installing a content management system (CMS) like Joomla or WordPress. It is also possible to connect to localhost via FTP with an HTML editor.

The default FTP user is "newuser", the default FTP password is "wampp". The default MySQL user is "root" while there is no default MySQL password.

In this project XAMPP has been used to create a localhost database called phpmyadmin. This database connects with the netbeans IDE and we can update the values on this table.

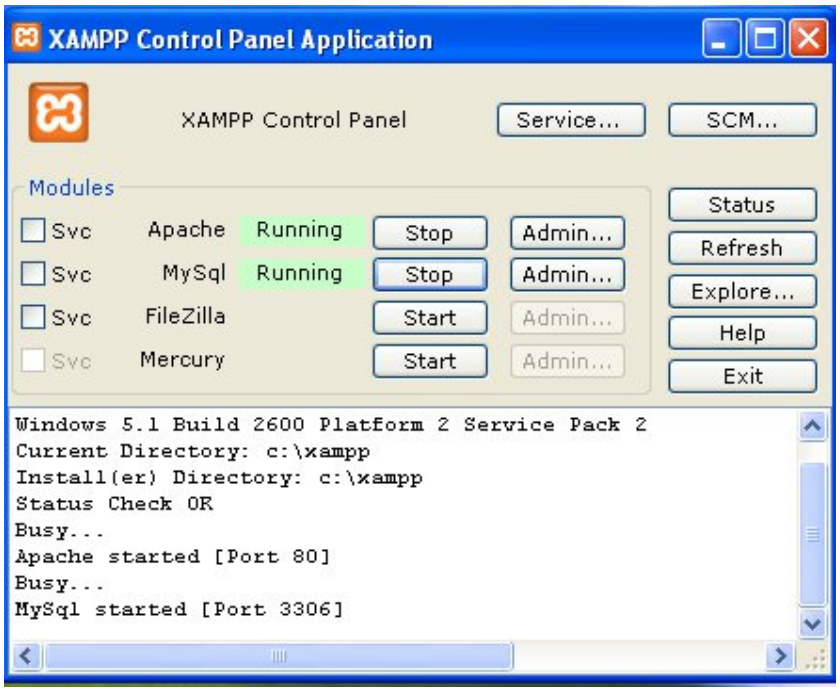


Figure 7: Screenshot of XAMPP control panel.

3.5 VMWARE

VMware Workstation is one of the best desktop virtualization applications available. If you need to run an operating system in a virtual machine, VMware Workstation is one of your best options. It is feature packed and offers support for tons of operating systems. VMware is dedicated to updating their applications to support the latest operating systems and hardware.

Windows XP has been used in VMware workstation for this project. The reason behind using VMware was that it improved the execution of the program for packet capture. Below is the snapshot of the software:

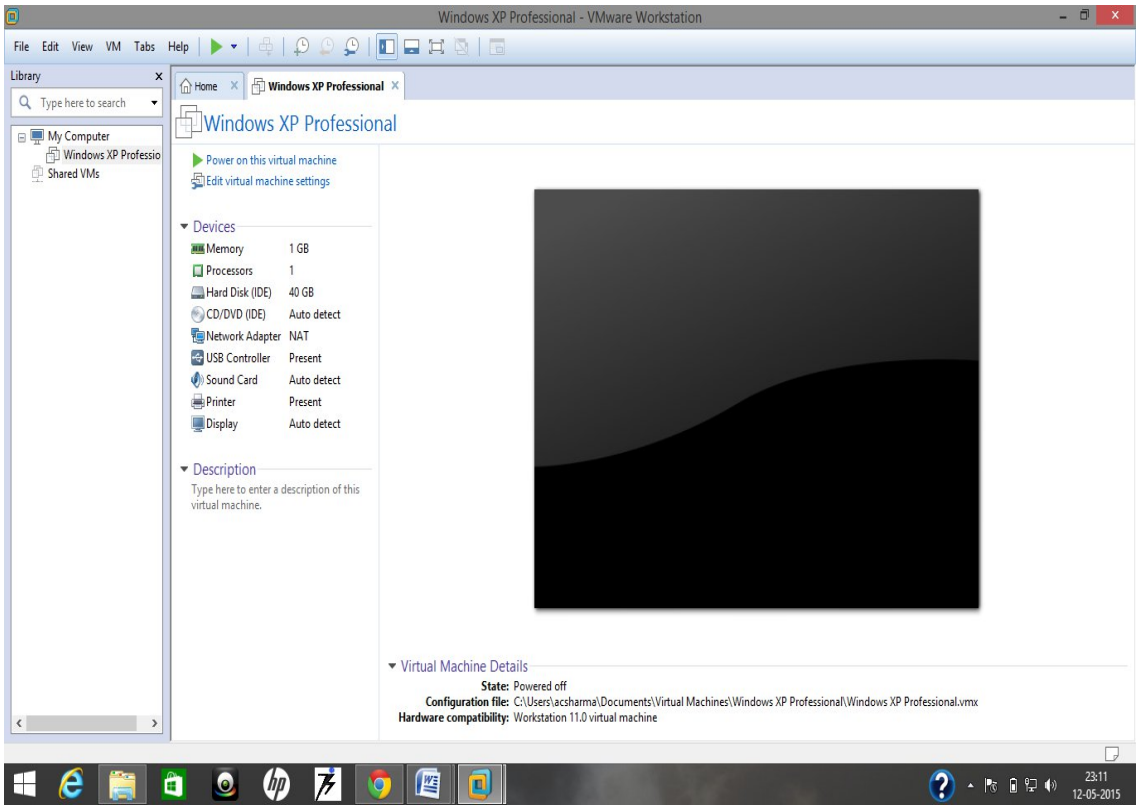


Figure 8: Screenshot of VMware software.

3.6 HIGHCHARTS

Highcharts is a charting library written in pure JavaScript, offering an easy way of adding interactive charts to your web site or web application. Highcharts currently supports line, spline, area, areaspline, column, bar, pie, scatter, angular gauges, arearange, areasplinerange, columnrange, bubble, box plot, error bars, funnel, waterfall and polar chart types. It works in all modern mobile and desktop browsers including the iPhone/iPad and Internet Explorer from version 6. On iOS and Android, multitouch support provides a seamless user experience. Standard browsers use SVG for the graphics rendering. In legacy Internet Explorer graphics are drawn using VML. One of the key features of Highcharts is that under any of the licenses, free or not, you are allowed to download the source code and make your own edits. This allows for personal modifications and a great flexibility. Highcharts is solely based on native browser technologies and doesn't require client side plugins like Flash or Java. Furthermore you don't need to install anything on your server. No PHP or ASP.NET. Highcharts needs only two JS files to run: The highcharts.js core and either the jQuery, MooTools or Prototype framework. One of these frameworks is most likely already in use in your web page. Highcharts supports line, spline, area, areaspline, column, bar, pie, scatter, angular gauges, arearange, areasplinerange, columnrange and polar chart types. Many of these can be combined in one chart. Setting the Highcharts configuration options requires no special programming skills. The options are given in a JavaScript object notation structure, which is basically a set of keys and values connected by colons, separated by commas and grouped by curly brackets. Through a full API you can add, remove and modify series and points or modify axes at any time after chart creation. Numerous events supply hooks for programming against the chart. In combination with jQuery, MooTools or Prototype's Ajax API, this opens for solutions like live charts constantly updating with values from the server, user supplied data and more. With the exporting module enabled, your users can export the chart to PNG, JPG, PDF or SVG format at the click of a button, or print the chart directly from the web page.

In this project Highcharts have been used to show relation between rows of database generated by the captured packets. Using these graphs a network administrator can easily guess about the condition of the network. The software created captures UDP packets.

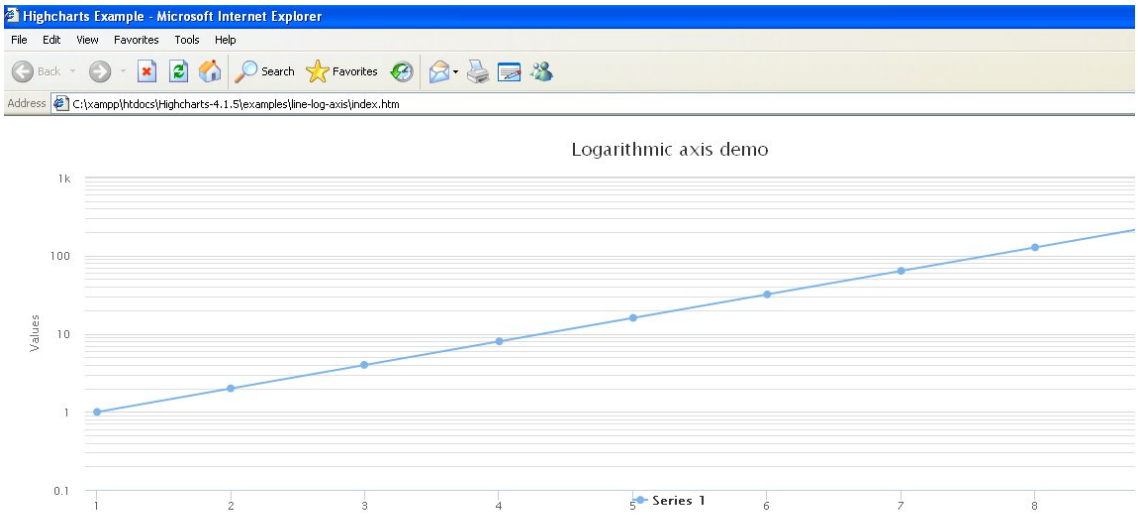


Figure 9: Screenshot of Logarithmic chart.

3.7 NETBEANS

NetBeans is a software development platform written in Java. The NetBeans Platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans Platform, including the NetBeans integrated development environment (IDE), can be extended by third party developers. The NetBeans IDE is primarily intended for development in Java, but also supports other languages, in particular PHP, C/C++ and HTML5. NetBeans is cross-platform and runs on Microsoft Windows, Mac OS X, Linux, Solaris and other platforms supporting a compatible JVM.

The NetBeans Team actively support the product and seek feature suggestions from the wider community. Every release is preceded by a time for Community testing and feedback. NetBeans IDE is an open-source integrated development environment. NetBeans IDE supports development of all Java application types (Java SE (including JavaFX), Java ME, web, EJB and mobile applications) out of the box. Among other features are an Ant-based project system, Maven support, refactorings, version
control (supporting CVS,Subversion, Git, Mercurial and Clearcase).

All the functions of the IDE are provided by modules. Each module provides a well defined function, such as support for the Java language, editing, or support for the CVS versioning system, and SVN. NetBeans contains all the modules needed for Java development in a single download, allowing the user to start working immediately. Modules also allow NetBeans to be extended. New features, such as support for other programming languages, can be added by installing additional modules. For instance, Sun Studio, Sun Java Studio Enterprise, and Sun Java Studio Creator from Sun Microsystems are all based on the NetBeans IDE.

An IDE is much more than a text editor. The NetBeans Editor indents lines, matches words and brackets, and highlights source code syntactically and semantically. It lets you easily refactor code, with a range of handy and powerful tools, while it also provides code templates, coding tips, and code generators.
The editor supports many languages from Java, C/C++, XML and HTML, to PHP, Groovy, Javadoc, JavaScript and JSP. Because the editor is extensible, you can plug in support for many other languages. Keeping a clear overview of large applications, with thousands of folders and files, and millions of lines of code, is a daunting task. NetBeans IDE provides different views of your data, from

multiple project windows to helpful tools for setting up your applications and managing them efficiently, letting you drill down into your data quickly and easily, while giving you versioning tools via Subversion, Mercurial, and Git integration out of the box. When new developers join your project, they can understand the structure of your application because your code is well-organized. The cost of buggy code increases the longer it remains unfixed. NetBeans provides static analysis tools, especially integration with the widely used FindBugs tool, for identifying and fixing common problems in Java code. In addition, the NetBeans Debugger lets you place breakpoints in your source code, add field watches, step through your code, run into methods, take snapshots and monitor execution as it occurs.

The NetBeans Profiler provides expert assistance for optimizing your application's speed and memory usage, and makes it easier to build reliable and scalable Java SE, JavaFX and Java EE applications. NetBeans IDE includes a visual debugger for Java SE applications, letting you debug user interfaces without looking into source code. Take GUI snapshots of your applications and click on user interface elements to jump back into the related source code.

NetBeans 32 bits has been used as it is compatible with Jpcap library.

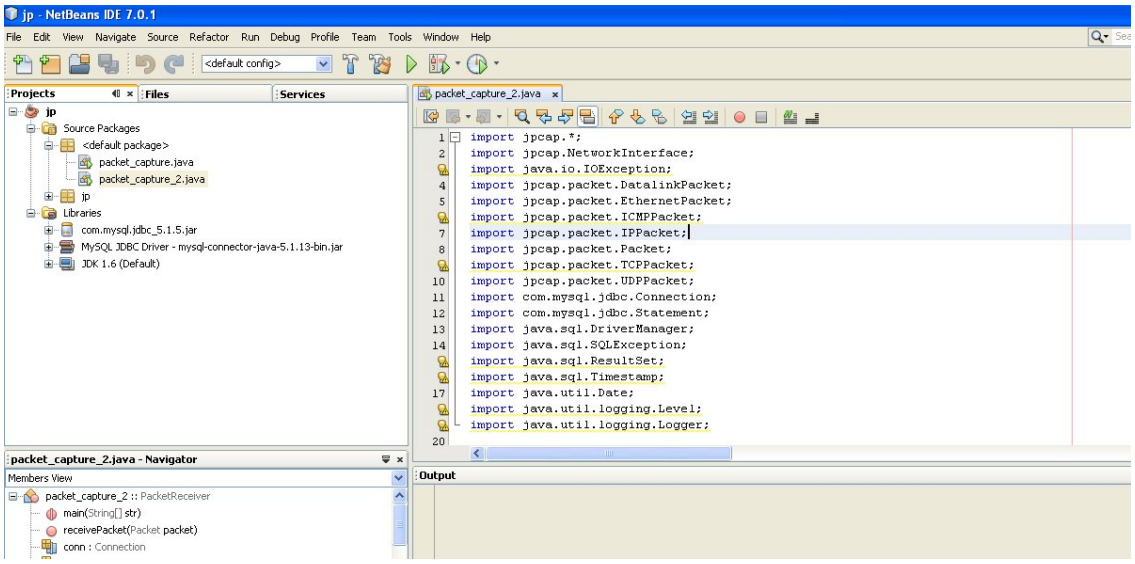


Figure 10: Screenshot of Netbeans IDE.

3.8 JAVA (32 BITS)

Java is a set of several computer software and specifications developed by Sun Microsystems, later acquired by Oracle Corporation, that provides a system for developing application software and deploying it in a cross-platform computing environment. Java is used in a wide variety of computing platforms from embedded devices and mobile phones to enterprise servers and supercomputers. While less common, Java applets run in secure, sandboxed environments to provide many features of native applications and can be embedded in HTML pages.

Writing in the Java programming language is the primary way to produce code that will be deployed as byte code in a Java Virtual Machine (JVM); byte code compilers are also available for other languages, including Ada, JavaScript, Python, and Ruby. In addition, several languages have been designed to run natively on the JVM, including Scala, Clojure and Groovy. Java syntax borrows heavily from C and C++, but object-oriented features are modeled after Smalltalk and Objective-C. Java eschews certain low-level constructs such as pointers and has a very simple memory model where every object is allocated on the heap and all variables of object types are references. Memory management is handled through integrated automatic garbage collection performed by the JVM. The Java platform is a suite of programs that facilitate developing and running programs written in the Java programming language. The platform is not specific to any one processor or operating system, rather an execution engine (called a virtual machine) and a compiler with a set of libraries are implemented for various hardware and operating systems so that Java programs can run identically on all of them.

The heart of the Java platform is the concept of a "virtual machine" that executes Java bytecode programs. This bytecode is the same no matter what hardware or operating system the program is running under. There is a JIT (Just In Time) compiler within the Java Virtual Machine, or JVM. The JIT compiler translates the Java bytecode into native processor instructions at run-time and caches the native code in memory during execution.

The use of bytecode as an intermediate language permits Java programs to run on any platform that has a virtual machine available. The use of a JIT compiler means that Java applications, after a short delay during loading and once they have "warmed up" by being all or mostly JIT-compiled, tend to run about as fast as native programs. Since JRE version 1.2, Sun's JVM implementation has included a just-in-time compiler instead of an interpreter.

Although Java programs are cross-platform or platform independent, the code of the Java Virtual Machines (JVM) that execute these programs is not. Every supported operating platform has its own JVM.

In this project Java 32 bits have been used as it is compatible with Jpcap library.

CHAPTER 4: TOOLS AND TECHNIQUES

Following is the architectural diagram of the project implementation. It gives a bird's eye view of the whole software.

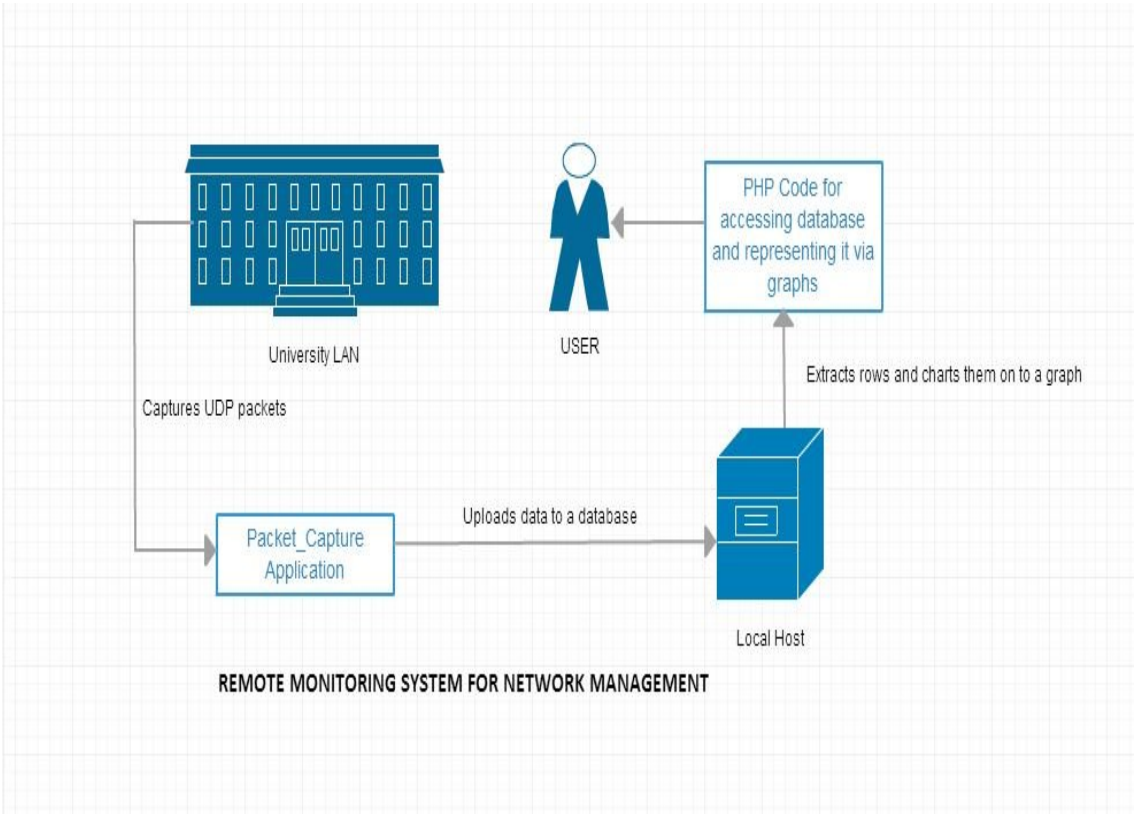


Figure 11: Architectural diagram of NMS.

Step 1: Install VMware on the system

The libraries used to run the project were better suited to Windows XP environment. As previous attempts to suit library execution in Windows 8 environment were futile, Windows XP was installed on VMware.

Step 2: Install Java 32 bit

The library jpcap is only compatible with 32 bit java.

Figure 12: The icon of jdk 32 bits is shown.

Step 3: Install Winpcap and Jpcap libraries via their setups

Jpcap and winpcap libraries are the core dependencies on which the packet capture code runs. Hence, it is very essential to ensure their proper installation.

Figure 13: The icon of Winpcap and Jpcap libraries are shown.

Step 4: Install XAMPP

XAMPP server provides us with a local host which can run MySql and Apache servers.

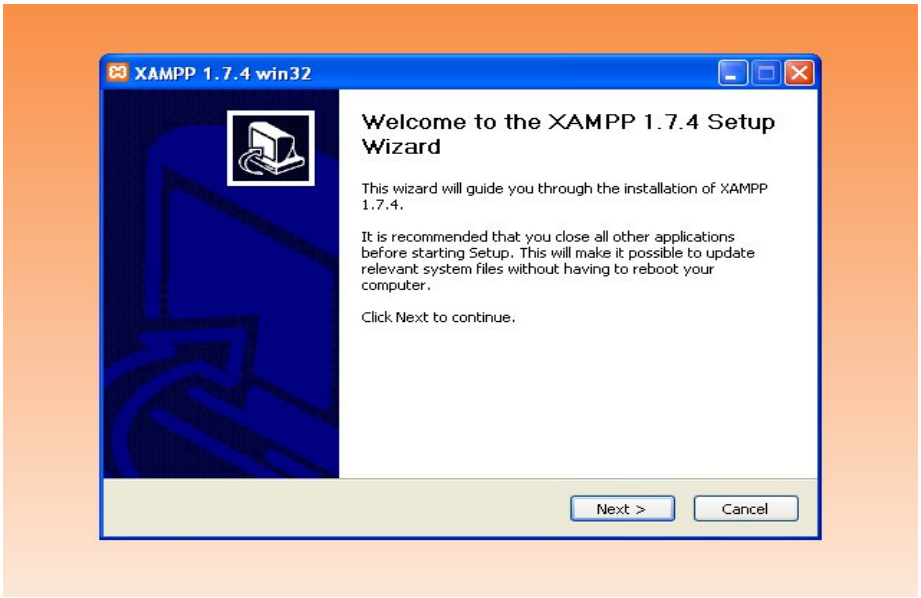


Figure 14: Installation process of XAMPP server.

Step 5: Install NETBEANS IDE

Netbeans is the editor used to run the packet capture program. Also, ensure that com.mysql.jdbc.jar, MYSQL JDBC driver and JDK 32 bit are included in the library of the project.

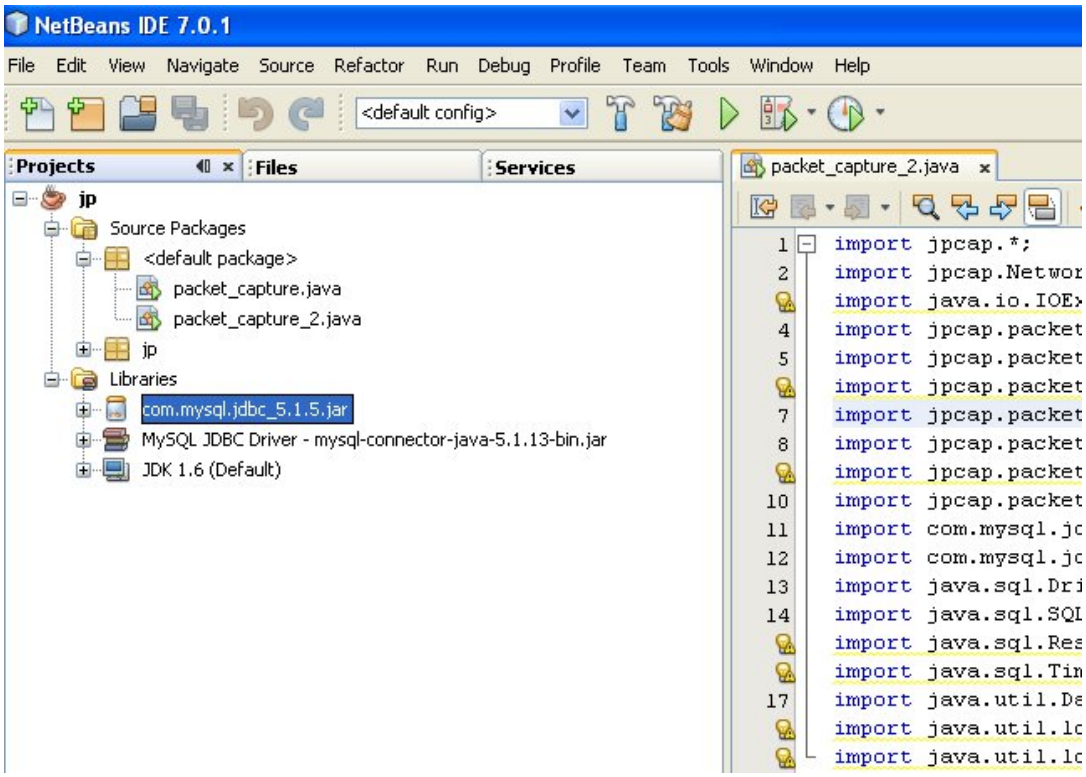


Figure 15: Snapshot of libraries to be included.

Step 6: Create a database connection and a table which will hold details of the information the program generates.

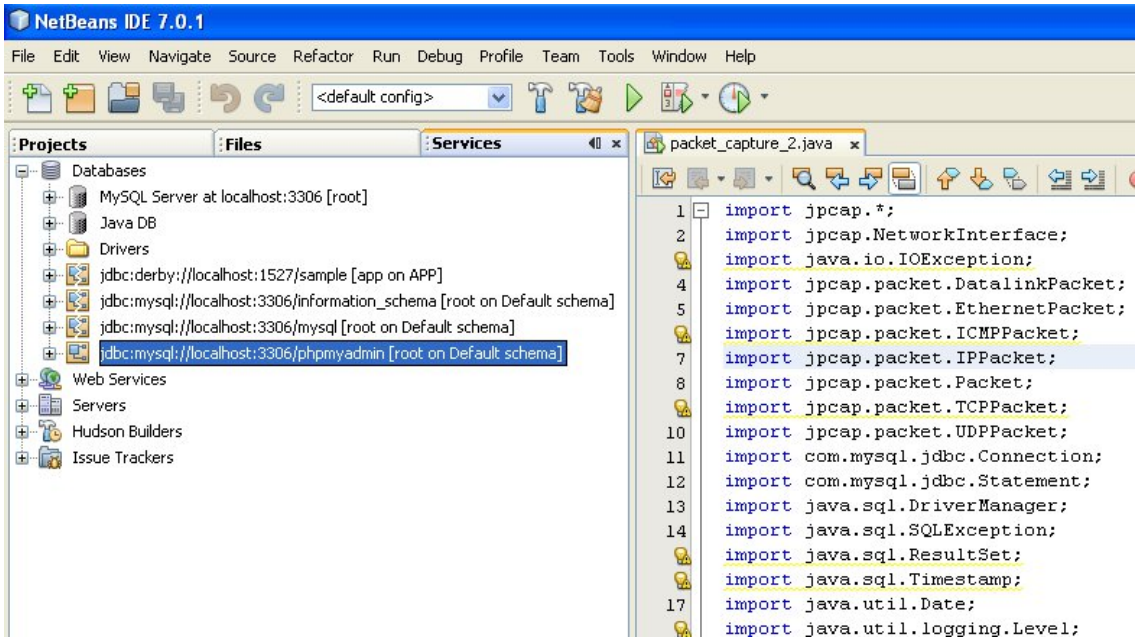


Figure 16: Database connection established.

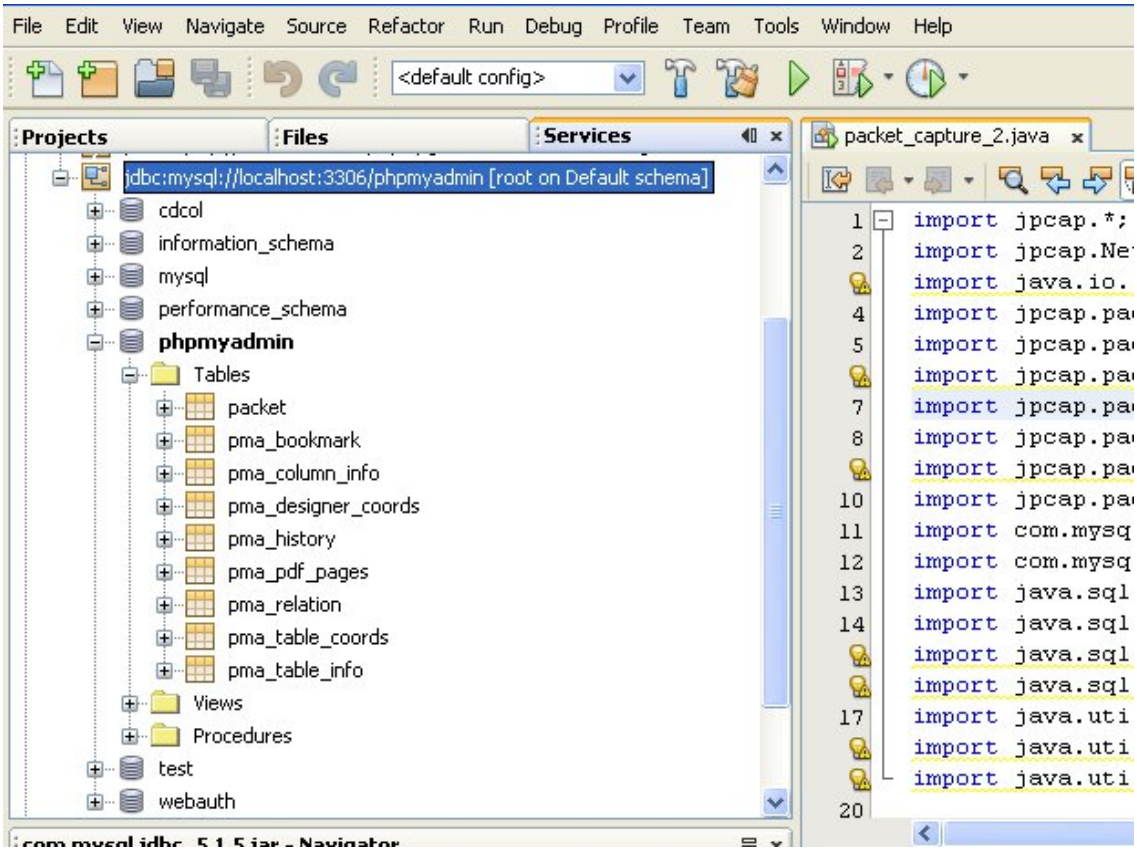


Figure 17: Table name packet created.

Step 7: Open the localhost server and verify whether the table has been created or not.

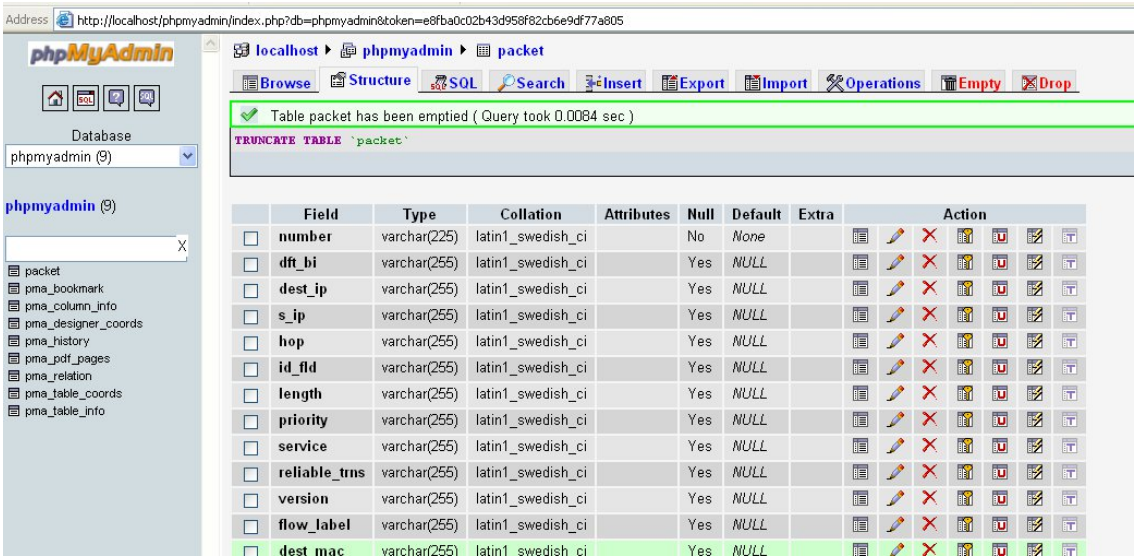


Figure 18: Table name packet created in database phpmyadmin.

Step 8: Run the code for packet capture

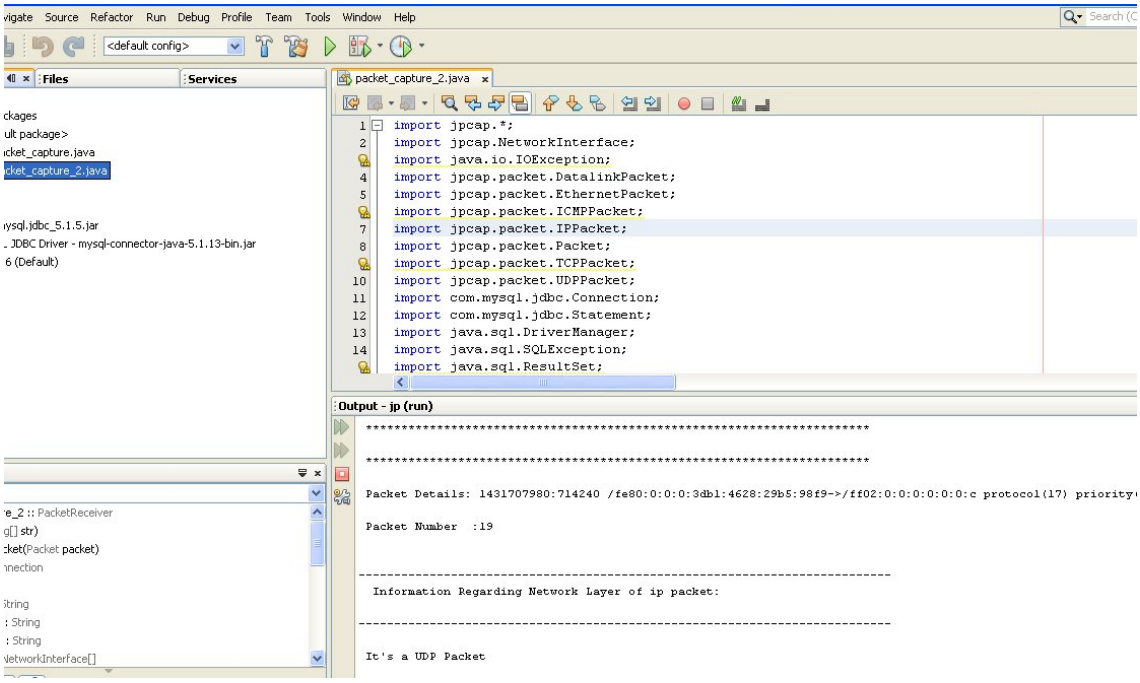


Figure 19: Code for packet capture running.

Step 9: Validate whether the data generated is being stored in database phpmyadmin.

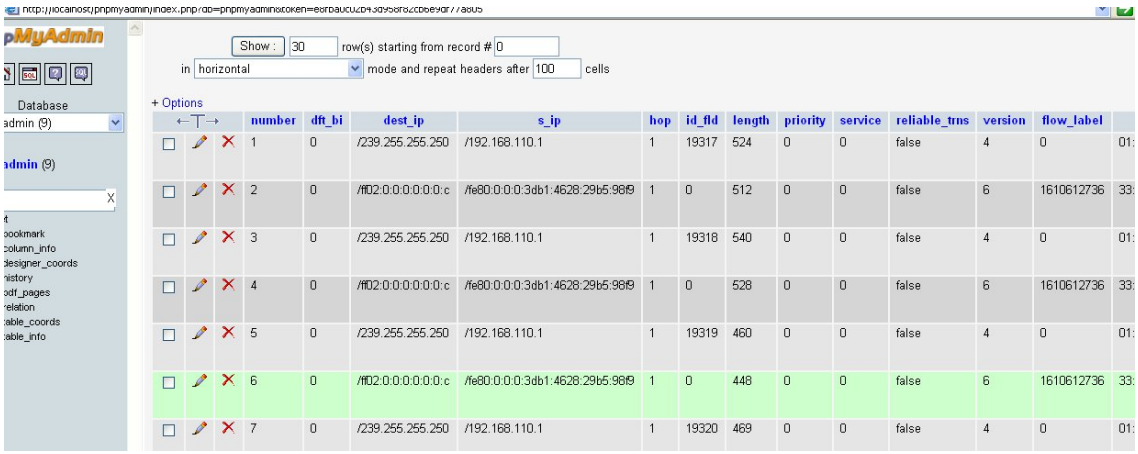


Figure 20: Table named packet in database phpmyadmin being updated.

Step 10: Create php code to generate graphs out of columns of database. For creating graphs, download Highcharts zip package and uninstall its contents in xampp/htdocs.

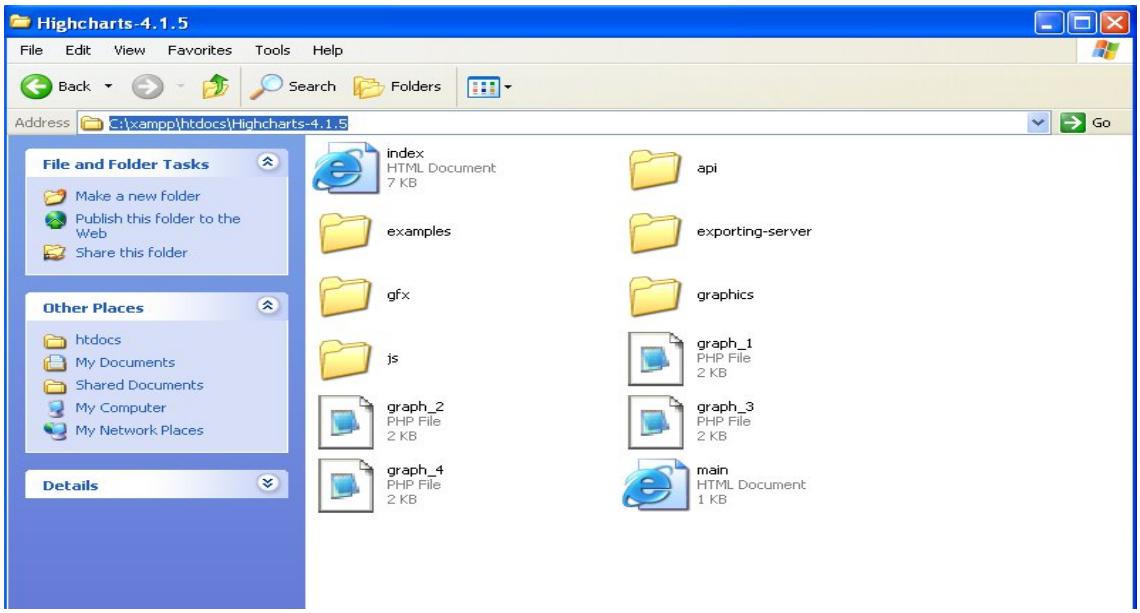


Figure 21: Copied contents of Highcharts in C:\xampp\htdocs\Highcharts-4.1.5.

(Here, graph_1,graph_2,graph_3,graph_4 and main are created later)

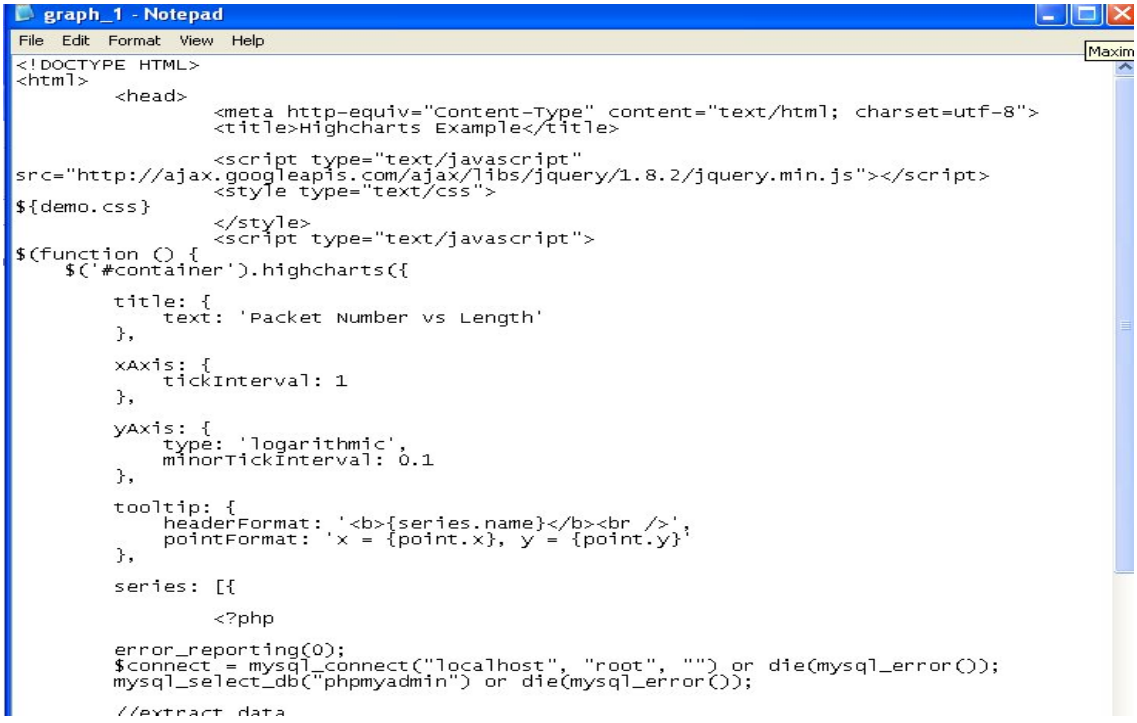


Figure 22: Graph_1: php code for creating graph of Packet Number vs length. Similar is for other graphs.

Step 11: Create a html doc which links all the graphs to a single page

File Edit Format View Help

```
<!DOCTYPE HTML>
<html>
<head> REMOTE MONITORING SYSTEM FOR NETWORK MANAGEMENT</head>


<body>


<FORM METHOD="LINK" ACTION="http://localhost/Highcharts-4.1.5/graph_1.php">
<INPUT TYPE="submit" VALUE="Graph 1">
</FORM>
<br>

<FORM METHOD="LINK" ACTION="http://localhost/Highcharts-4.1.5/graph_2.php">
<INPUT TYPE="submit" VALUE="Graph 2">
</FORM>
<br>

<FORM METHOD="LINK" ACTION="http://localhost/Highcharts-4.1.5/graph_3.php">
<INPUT TYPE="submit" VALUE="Graph 3">
</FORM>
<br>

<FORM METHOD="LINK" ACTION="http://localhost/Highcharts-4.1.5/graph_4.php">
<INPUT TYPE="submit" VALUE="Graph 4">
</FORM>
<br>



</form>

</body>
</html>
```

Figure 23: HTML doc for main page

Step 12: Run the main html doc. The user will be able to view the graphs.

File Edit Format View Help

```
<!DOCTYPE HTML>
<html>
<head>

<b><h1 style="color:gold">REMOTE MONITORING SYSTEM FOR NETWORK MANAGEMENT</h1><b>


</head>
<br>
<br>
<br>

<body>
<body style="background-color:Darkcyan">

<p style="color:white">Number of Packets vs Length of packets</p>

<FORM METHOD="LINK" ACTION="http://localhost/Highcharts-4.1.5/graph_1.php">
<INPUT TYPE="submit" VALUE="Graph 1">
</FORM>
<br>


<p style="color:white">Timestamp count vs Number of packets</p>

<FORM METHOD="LINK" ACTION="http://localhost/Highcharts-4.1.5/graph_2.php">
<INPUT TYPE="submit" VALUE="Graph 2">
</FORM>
<br>


<p style="color:white">Number of packets vs Version of packets</p>

<FORM METHOD="LINK" ACTION="http://localhost/Highcharts-4.1.5/graph_3.php">
<INPUT TYPE="submit" VALUE="Graph 3">
</FORM>
<br>


<p style="color:white">Number of packets vs Identification field</p>

<FORM METHOD="LINK" ACTION="http://localhost/Highcharts-4.1.5/graph_4.php">
<INPUT TYPE="submit" VALUE="Graph 4">
</FORM>
<br>
```

Figure 24: main html document source code

CHAPTER 5: RESULTS AND CONCLUSION

The final output would appear as follows:

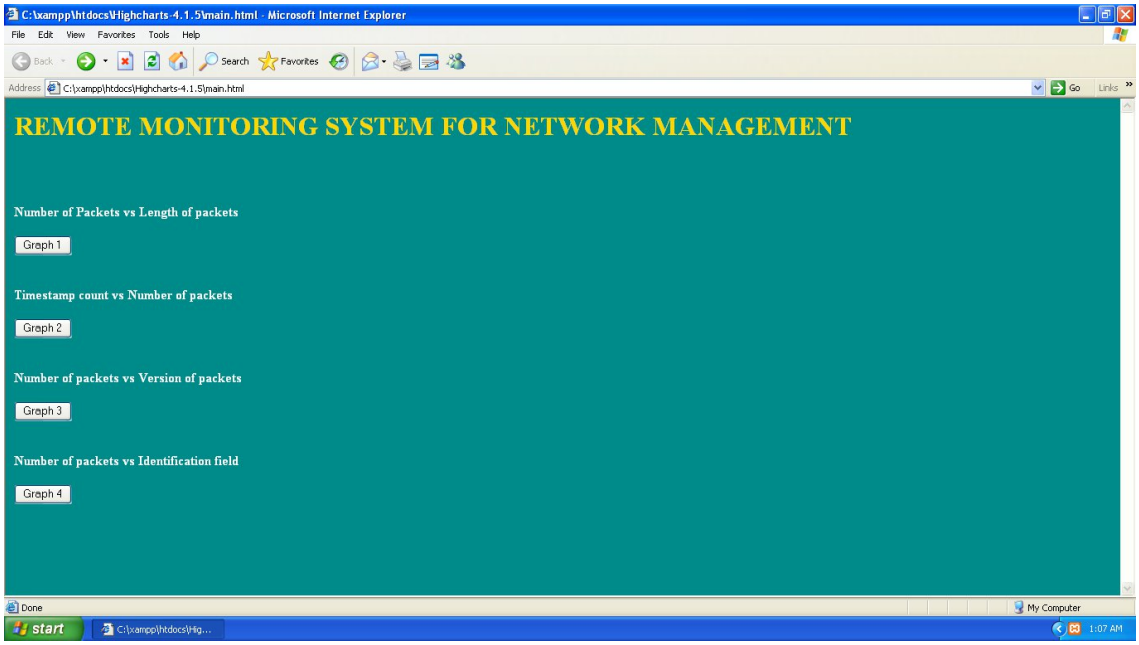


Figure 25: Remote monitoring for network management homepage

The user will see certain graphs being generated on his or her screen. They will be as follows:

1. Number of packets vs length of the packet

2.Timestamp count vs Number of packets

3. Number of packets vs version of ip packet

4.Number of packets vs identification field of the packet

These are shown below-

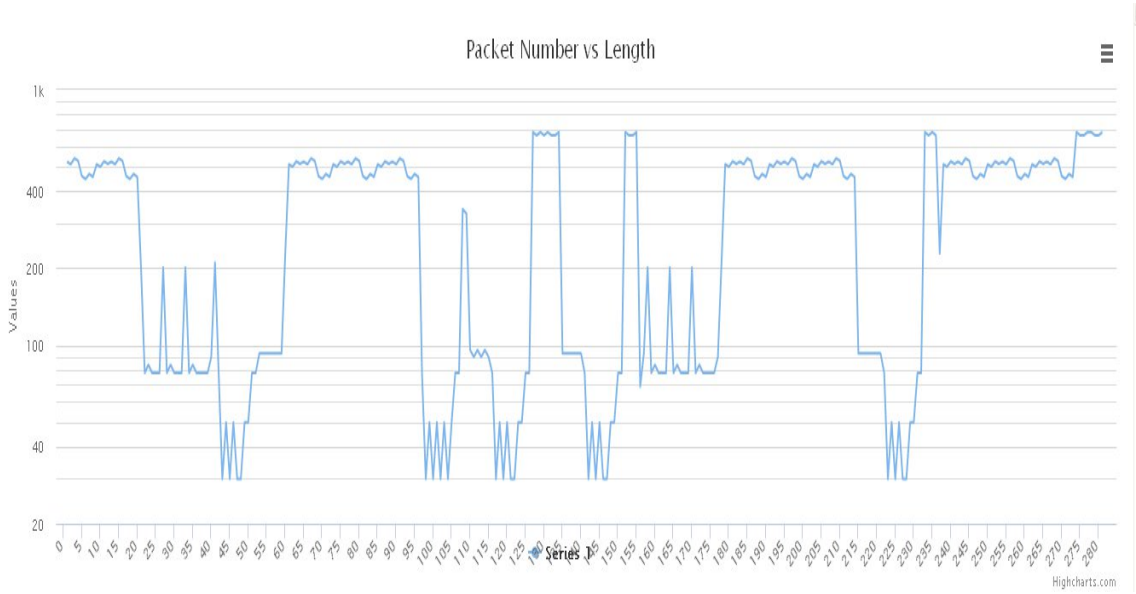


Figure 26: Number of packets vs length of the packet

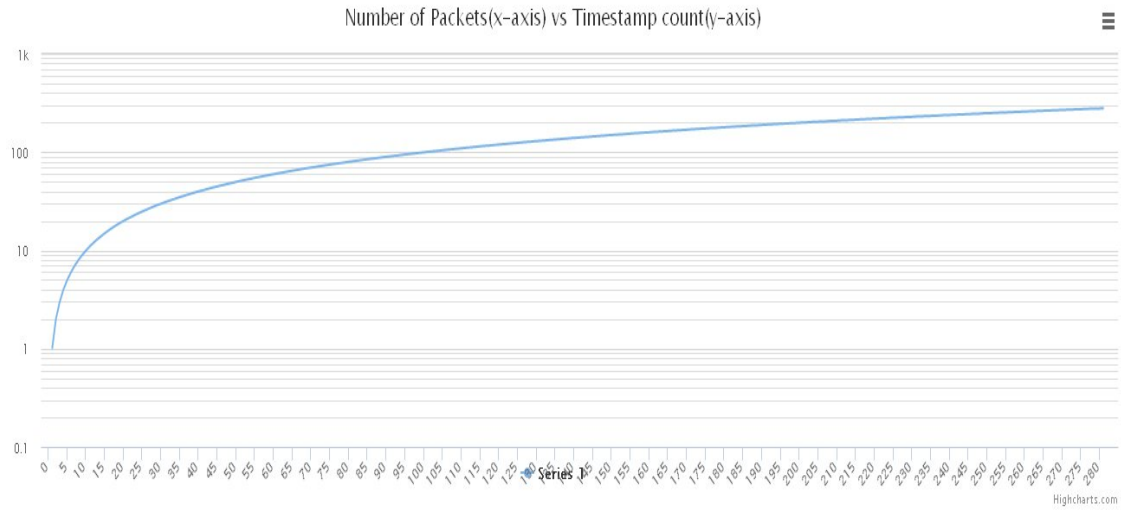


Figure 27: Number of packets vs Timestamp Count

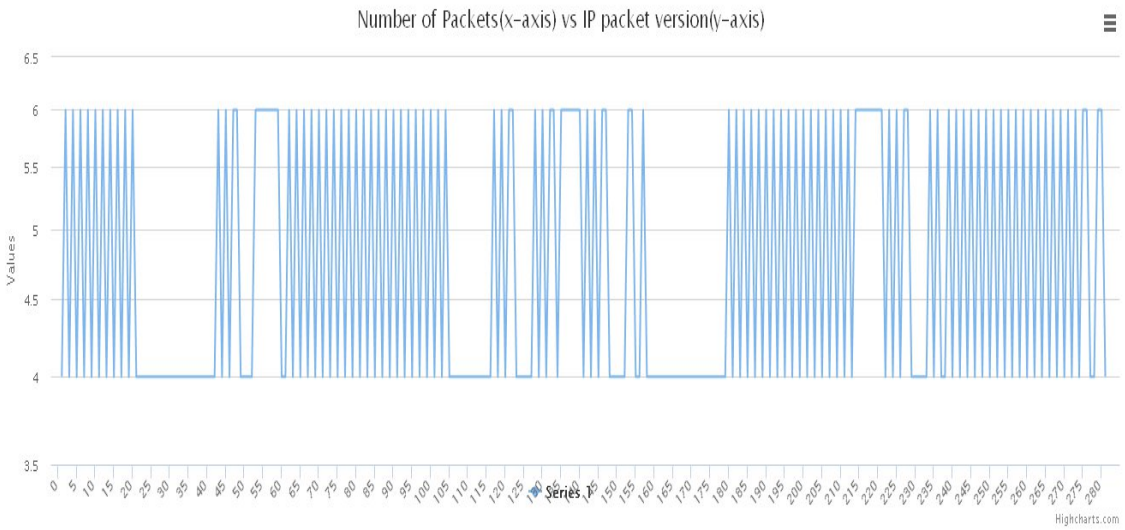


Figure 28: Number of packets vs. IP packet version

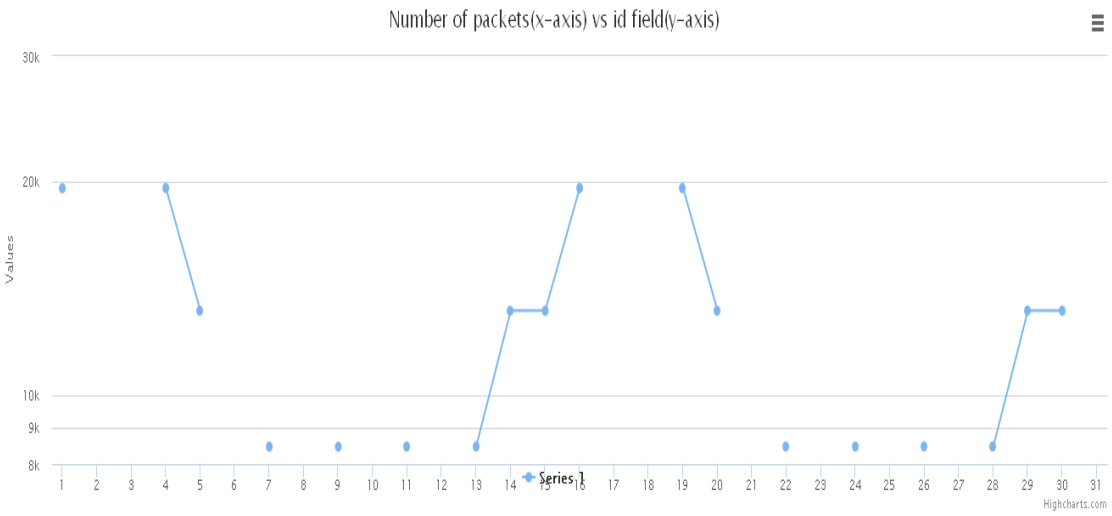


Figure 29: Number of packets vs. id field

5.1 INTERPRETATION OF THE GRAPHS

1. The first graph between Packet Number and Length of packet tells us which packet has what length. This graph can be useful to network managers to decipher about the *amount* of data travelling in the network. If the packet length is outside the admissible range the network administrator can terminate connections to reduce load on the network.

2. The second graph can also be termed as the *network performance graph*. It tells the relation between number of packets and timestamp count. As clear from the graph, the difference between two timestamps is initially large, which depicts that the rate of packet capturing is initially slow. As, we move further we see that the difference between consecutive timestamps decrease, denoting that the rate of packet capturing has now increased. A network manager can speculate from this graph the following possibilities:

- If the difference between timestamps is too large, then the graph will be very steep. This could denote that there is very sparse traffic or the network is not functioning properly.

- If the difference between timestamps is too less, then it can be inferred that the traffic is very dense and the network could be overloaded. In this case the graph will show a very gentle slope.

Using these results a network manager can take appropriate action on the network functioning.

3.The third graph is between number of packets vs the version of ip packet received. The network manager can guess from the graph about the type of version a packet is following. This can be a direct indication of the amount of information a packet is carrying(ipv4 carries less information than ipv6). Ipv6 can support real time audio and data. If there is large amount of ipv6 data then network changes can be incorporated to adjust this type of information.

4.The fourth graph is between number of packets and id field. Identification field is 16-bit field which identifies a datagram originating from the source host. The combination of the identification and source IPv4 address must

uniquely define a datagram as it leaves the source host. To guarantee uniqueness, the IPv4 protocol uses a counter to label the datagrams. The counter is initialized to a positive number. When the IPv4 protocol sends a datagram, it copies the current value of the counter to the identification field and increments the counter by 1. As long as the counter is kept in the main memory, uniqueness is guaranteed. When a datagram is fragmented, the value in the identification field is copied to all fragments. In other words, all fragments have the same identification number, the same as the original datagram. The identification number helps the destination in reassembling the datagram. It knows that all fragments having the same identification value must be assembled into one datagram.

So, this graph can indicate the identification number of packet, thus keeping a record of the packets flowing in the network.

5.2 CONCLUSION AND FUTURE DEVELOPMENT

The project covers the basic aspects of network management. Since the server can be accessed from anywhere, the database and the graphs can be viewed from anywhere, thus enabling remote monitoring. The project is based on capturing UDP packets and generation of graphs for network performance interpretation. This can be extrapolated to other type of packets such as TCP,ICMP etc which can give a more vivid picture of the network. More information can be gained from the tables generated and new graphs can be created. This could improve the accuracy of network analysis and pre-mediate problems in network.

Every software needs continuous development to sustain. The present project, with few enhancements can provide sufficient analysis about a small LAN network. If the database and graph generation is further improved ,a WAN can easily be monitored.

REFERENCES

1. https://www.winpcap.org/install/ (download link for winpcap)

2. http://jpcap.gitspot.com/index.html (download link for libcap)

3. https://www.vmware.com/products/ (download link for vmware)

4. https://www.apachefriends.org/download.html (download link for xampp)

5. https://java.com/en/download/ (download link for java 32 bit)

6. http://www.highcharts.com/download (download link for highcharts)

7. https://netbeans.org/downloads/ (download link for netbeans)

8. http://www.sipinspector.com/download/jpcap/JPcap_tutorial

9. Remote Configuration Monitoring of Autonomous Information Processing Machine on LAN by Hema Thomas1 Christ University, Bangalore.

10. Harsh Mittal, Manoj Jain and Latha Banda, Monitoring Local area Network using remote method invocation , International Journal of Computer Science and Mobile Computng, vol. 2, Issue. 5, May 2013.

11. Dinesh C. Verma, Simplifying Network Administration Using Policy-Based Management, IEEE Network, March/April 2002.

12. Network Management: Principles and Practice, Mani Subramanian