NETCAP (A NETWORK TRAFFIC MONITORING SNIFFER)

LISA ANN PHUAN HUI JING

A thesis submitted in fulfillment of the requirements for the award of the degree of Bachelor of Computer Science (Computer Systems & Networking)

Faculty of Computer Systems & Software Engineering
Universiti Malaysia Pahang

MAY, 2011

ABSTRACT

From hundreds to thousands of computers, hubs to switched networks, network administrators need more sophisticated network traffic monitoring tools in order to deal with the increase. These network monitoring tools are needed in order to perform their work, to obtain the data transiting on a network and capturing it while the network is working. The capture process consists of listening on the network, every transiting frame, independently from its source or destination. However, most of the network traffic monitoring sniffer display data packets that are captured in a less-readable format which are difficult to be analyzed by newbie network administrators or students that are learning to analyze network traffic flow. Moreover, most of the sniffers only use command-line to run. Thus, NetCap is developed to display a more readable and detailed data packets information that are captured and to display a more user-friendly graphical interface. The information about sniffer is searched to analyze the functions of sniffer and research methodology is done to plan, design and implement the sniffer. Software capture architecture for the Microsoft's Win32 operating system family is analyzed. This architecture includes a low-level framework, winpcap that adds to Win32 operating systems the ability to efficiently capture data from the most used network families. Based on that, NetCap is design and implemented using Netbeans 6.9.1. Furthermore, it also uses JCommon 1.0.16 and JFreeChart 1.0.13, library for Java platform to support and develop a real-time packets flow graph in NetCap. This paper uses jpcap, a Java library for capturing packets which is used to develop applications to capture packets from network interfaces plus analyze them in Java, and winpcap, a windows version of libpcap library which includes driver to support capturing packets on windows to design NetCap which is compatible to be used in windows platform.

ABSTRAK

Dari ratusan hingga ribuan komputer, hub ke rangkaian dihidupkan, administrator memerlukan sniffer yang lebih canggih untuk menangani peningkatan. Sniffer ini diperlukan dalam rangka untuk melaksanakan pekerjaan mereka, untuk mendapatkan data transit pada rangkaian dan menangkap sementara rangkaian bekerja. Proses menangkap terdiri daripada mendengar pada rangkaian, setiap frame transit, bebas daripada sumber. Namun, sebahagian besar trafik data rangkaian monitoring pakej paparan yang ditangkap dalam format kurang-dibaca yang sukar untuk dianalisa oleh administrator newbie atau pelajar yang sedang belajar untuk menganalisa trafik rangkaian sniffer. Sebahagian besar sniffer hanya menggunakan command-line untuk menjalankan. Dengan demikian, NetCap dibangunkan untuk memaparkan maklumat data yang lebih mudah dibaca dan untuk memaparkan grafik antara muka yang lebih user-friendly. Maklumat tentang sniffer dicari untuk menganalisis fungsi sniffer dan metodologi kajian dilakukan untuk merancang, merancang dan melaksanakan sniffer. Software menangkap arsitektur untuk keluarga sistem Win32 di operasi Microsoft dianalisis. Arsitektur ini merangkumi rangka peringkat rendah, WinPcap yang menambah untuk sistem operasi Win32 kemampuan untuk secara cekap menangkap data dari keluarga rangkaian yang paling digunakan. NetCap dilaksanakan menggunakan Netbeans 6.9.1, Jcommon 1.0.16 dan JFreeChart 1.0.13 digunakan untuk platform Java untuk menyokong grafik real-time mengalir di NetCap. Java menggunakan jpcap untuk menangkap pakej-pakej yang digunakan untuk mengembangkan aplikasi untuk menangkap pakej-pakej dari peranti rangkaian ditambah analisis mereka, dan WinPcap, versi windows perpustakaan libpcap yang menyokong menangkap pakej-pakej pada windows untuk merancang NetCap yang serasi untuk digunakan pada platform windows.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Nowadays, network monitoring has become more and more important in a modern complicated network. In the past, network administrators might only monitor a few network devices or less than a hundred computers. The network bandwidth may be just 10 or 100 Megabit per second; but now administrators have to deal with not only higher speed wired network and Asynchronous Transfer Mode (ATM) network but also wireless networks. Therefore, they need more sophisticated network traffic monitoring and analysis tools in order to maintain the network system stability and availability such as to fix network problems on time or to avoid network failure, to ensure the network security strength and to make good decisions for network planning. Network administrators have to regularly check the network performance if the network devices are overloaded. Before a failure due to the overload, information about network usage can be used to make a network plan for short-term and long-term future improvement.

There are various kinds of tools dealing with the network monitoring and analysis, such as tools used by Simple Network Management Protocol (SNMP), Windows Management Instrumentation (WMI), Sniffing, and Network flow monitoring and analysis (Chakchai, 2010)[7]. Aside from network flow information from network devices, the local traffic or host-bed traffic flow information is described in this paper. Instead of requesting network devices to send the flow information to the monitoring host, packet sniffer locally collects the flow information in local network. Initially, sniffer is a registered trademark of Network Associates, Inc. used on their network analyzing products, but today sniffer is a well-known name for network monitor and analyzer (Chakchai, 2010) [7].

Furthermore, packet sniffer is also known as network analyzer, network sniffer, packet analyzer, protocol analyzer or just sniffers. It can intercept and log traffic passing over a digital network or part of a network. As data streams flow across the network, it captures each packet and eventually decodes and analyses its content according to the appropriate RFC or other specifications. Besides that, it is also a tool which utilizes network interfaces of computer to capture data packets which destination is other computers. Packet sniffers are basically applications. They are programs used to read packets that travel across the network layer of the Transmission Control Protocol/Internet Protocol (TCP/IP) layer (Sabeel, A., Rajeev, S.G. and Chandrashekar, H.S., 2003) [19]. Mainly, the packets are retrieved from the network layer and the data is interpreted.

A sniffer can be either hardware or software, which mainly intercept and collect the local traffic. After recording the traffic, the sniffer provides the function to decode and simply analyze the content of the packets in human readable. The traffic flow information in this category is local, that is, sniffer can capture the packet only from the network that sniffer attaches to (Chakchai, 2010)[7]. Therefore, in order to capture more traffic from several networks, some techniques have to be enabled or the network infrastructure might be changed. For example, due to the widespread of installing switch rather than hub, a port mirroring technique has to be enabled in order to make switches forward all the data packets to the sniffer. Another technique is to place sniffer in a core network where all packets the administrator concerns are passed.

Consider the nature of broadcasting network; a network adapter discards a packet, which the destination address does not belong to. However, to capture all traffic, the network adapter will be placed in to promiscuous mode (Chakchai, 2010)[7]. This means that the NIC will then pass all traffic it receives to the kernel rather than just frames addressed to it. The program then can constantly read all the information entering the computer via network card. The amount of traffic largely depends on the location of the computer in the network. A client system out on an isolated branch of the network sees only a small segment of the network traffic, while the main domain server sees almost all of it. A packet sniffer can usually be set up in one of two ways which are unfiltered and filtered. An unfiltered packet sniffer will capture all of the packets where as a filtered network monitoring tool will capture only those packets containing specific data elements. Packets that contain targeted data are copied onto the hard disk as they pass through. These copies can then be analyzed carefully for specific information or patterns.

A packet sniffer can be used in many ways whether it is for good or bad. A packet sniffer can be used to monitor network activities. For example, when a network monitoring tool is located at one of the servers of your ISP would potentially be able to monitor all of your network activities, such as which Web sites you visit, what you look at on the site, whom you send e-mail to, what's in the e-mail you send, what you download from a site and also what streaming events you use.

Positive usage of packet sniffer is to maintain network and system working normally by capturing packets, recording and analyzing traffic, decrypting packets and displaying in clear text, converting data to readable format, showing relevant information like IP, protocol, host or server name and so on. The most common protocols analyzed by sniffer are TCP/IP, IPX, DECNet. Normally, sniffer is used as an assistant of network management for its monitoring and analyzing features can help us to troubleshoot network, detect intrusion, control traffic or supervise network contents. However, such features may also be utilized by hackers as a snooping tool to break into other computers.

Though having sniffer installed can benefit a lot in term of network troubleshooting, network intrusion detection, network usage, and so on, the limitation of sniffer is that it cannot read the encrypted packets. Big issue is about the privacy reason since the administrator can see the content of the packet (Chakchai, 2010)[7]. Negative usage of packet sniffer is well known as its harms to network security such as catching password, which is the main reason for most illegal uses of sniffing tool, capturing special and private information of transactions, like username, credit ID, account, and password, recording email or instant message and resuming its content and also disserving the security of network places or to gain higher level authority.

The purpose of developing NetCap is to display a more readable and detailed data packets information that is captured by packet sniffer and to display a more user-friendly graphical interface in packet sniffer. NetCap is a network traffic monitoring sniffer that sniffs packets from the LAN network to analyze its traffic. Moreover, it is developed using Java. Besides that, it uses jpcap, a standard libpcap library with Java wrapper as an application programming interface to capture the packets in user level, WinPcap in windows platform. Furthermore, it uses JCommon and JFreeChart to develop the real-time graph of ongoing packets transfer generated by NetCap. NetCap can help newbie network administrators to understand the packets transferred in the network flow due to its user-friendly graphical interface. Besides that, it also make sure administrators understand what kind of packets contain in the network flow and their information based on the table display and tree display generated by NetCap.

In a nut shell, despite the negative usages of sniffer, where it can be misused for illegal purposes such as hacking or invading someone's privacy; it also has benefits such as it can be used by network administrators to monitor network activities in order to maintain the performance of the network. In addition, it can also be used by beginners who have just started learning how to monitor their own network flow and learn about the data packets that flow through their network.

1.2 Problem Statement

(i) The data packets information captured by packet sniffer are in a less-readable format.

(ii) Most of the packet sniffer have less user-friendly graphical interface or sometimes use command-line tool only.

1.3 Objective

(i) To display a more readable and detailed data packets information that are captured by packet sniffer.

(ii) To display a more user-friendly graphical interface in packet sniffer.

1.4 Scope

(i) Network administrator can detect network problems by just analyzing more readable and understandable data packets information.

(ii) Network administrator has no problem interpreting the network problem by just analyzing the graphical form of the network traffic captured

CHAPTER 2

LITERATURE REVIEW

2.1 What is Packet Sniffer?

Packet sniffing or network monitoring is a method of tapping each packet as it flows across the network. Moreover, it is a technique in which a user sniffs data belonging to other users of the network. Packet sniffers can operate as an administrative tool or for malicious purposes. It depends on the user's intent. Network administrators use them for monitoring and validating network traffic. According to Sabeel, A., Rajeev, S.G. and Chandrashekar, H.S. (2003)[19], packet sniffers are basically applications. They are programs used to read packets that travel across the network layer of the Transmission Control Protocol/Internet Protocol (TCP/IP) layer. They also state that packet sniffers are not limited to Local Area Networks (LANs). Similar packet sniffers exist for Wide Area Networks (WANs). If a machine is in the path of two connected machines (A and B) on a WAN, the machine can listen to the traffic flowing from A to Similarly, Trabelsi, Z., Rahmani, H., Kaouech, K. and Frikha, M. (2004)[21] also comments that Sniffers are programs that allow a host to capture any packets in an Ethernet network, by putting the host's Network Interface Card (NIC) into the promiscuous mode. When a host's NIC is in the normal mode, it captures only the packets sent to the host. Since many basic services, such as FTP and SMTP, send passwords and data in clear text in the packets, Sniffers can be used by hackers to capture passwords and confidential data.

Broadway, J., Turnbull, B. and Slay, J. (2008)[6] states that packet sniffers are software tools that can be used to monitor network activity. They are installed on a computer in a network, once activated they make copies of all network traffic packets that are sent and received by the host computer. Network administrators are the primary users of packet sniffers. They use packet sniffers for a variety of reasons, including: as a problem-solving tool to fix network problems, as a performance tool to identify bottlenecks in the network and areas where efficiency can be improved. Likewise, Phang, S.Y., Lee, H.J. and Lim, H.T. (2008)[14] also mention that packet sniffing is an important technique in network monitoring for network administration or security management. According to them, the network administrator or the security manager can grab those packets that passing thru the network to log or trace any access to their network. Packet sniffer is a tool which can be utilized to sniff or grab packets off the network interface. A packet sniffer is useful for detecting messages being sent and received from a network interface, detecting an error implementation in network software, collecting statistics on the network traffic and etc. Moreover, they also stated that packet sniffer is important in IPv6 where the more and more network nodes and application will be supporting this protocol. Hence, packet sniffer can play an important role here to help in every aspects start from the designation of IPv6 supported network software up to the implementation and deployment of IPv6.

Moreover, Qadeer, M.A., Żahid, M., Iqbal, Arshad and Siddiqui, M.R. (2010)[15] indicate that computer software that can intercept and log traffic passing over a digital network or part of a network is better known as packet sniffer. The sniffer captures these packets by setting the NIC card in the promiscuous mode and eventually decodes them. The decoded information can be used in any way depending upon the intention of the person concerned who decodes the data (i.e. malicious or beneficial purpose). Depending on the network structure one can sniff all or just parts of the traffic from a single machine within the network. Correspondingly, Zhang, S.F., Chen, X.B., Zhang, S.A. and Yan, H.C. (2009)[22] note that in the sharing type Ethernet the Sniffer principle is based on the Ethernet in all correspondences all is broadcast this fact. In local area network, because the Ethernet sharing type characteristic had decided smells searches can succeed. Because the Ethernet is based on the broadcast way transmission data, all physical signal can transmit to each main engine node, in addition the network card may establish combination receive pattern (Promiscuous), under this kind of pattern, regardless of monitors the data frame destination address how, can the network card all give to receive. But in the TCP/IP agreement stack application agreement majority definite orders transmit in the network, in these definite orders data, often contains some sensitive information for example password, account number and so on, therefore uses Sniffer to be possible to be quietly monitors in all local area network the data communication, obtains these sensitive information. Simultaneously the Sniffer confidentiality is good, it only is "passive" the receive data, but not outward transmits the data, therefore in the transmission data process, is unable to perceive radically monitors to some people. Thus has realized to the network data monitor, namely Packet Sniffing.

2.2 Usages of Packet Sniffer

Sabeel, A., Rajeev, S.G. and Chandrashekar, H.S. (2003)[19] define that an analogy to a packet sniffer is a telephone wiretap. A person can tap a telephone line if he or she wants to snoop on another person. Similarly, packet sniffers can be used to snoop on other people's data that is currently being transmitted across the network. Network sniffers can capture passwords and other sensitive pieces of information passing through the network. Sometimes, sniffers can retain anonymity if they are launched from another system on the network. The packets may be sniffed from some intermediary hosts to which the launcher has access. Sniffing programs can be classified under two categories. Commercial packet sniffers used by network and Slay, J., 2008)[6]. Packet sniffers are very useful for these purposes because they provide the network administrator with a direct view of exactly how the network is behaving. Hackers and crackers use packet sniffers because of their ability to give the user a view of the data transmitted over networks. This data can then be used to gain personal information such as usernames and passwords from network users.

Phang, S.Y., Lee, H.J. and Lim, H.T. (2008)[14] state that the most common usage of packet sniffing will be the network protocol analyzer where those packets that being grabbed will be further analyzed and interpreted in a human readable form instead of displaying tons of hexadecimal characters. Some examples of the existing network protocol analyzer are TCPDUMP, Wireshark, Kismet and etc. These variations of packets sniffer has been existed since IPv4 and minimal a support for IPv6. These applications are a large piece of software or codes which have bunch of functionalities. Therefore, they appear to be complex and less efficient where IPv6 packets sniffing and analyzing is given the priority. Qadeer, M.A., Zahid, M., Iqbal, Arshad and Siddiqui, M.R. (2010)[15] mention that the packet sniffer captures the data that is addressed to other machines, saving it for later analysis. It can be used legitimately by a network or system administrator to monitor and troubleshoot network traffic. Using the information captured by the packet sniffer an administrator can identify erroneous packets and use the data to pinpoint bottlenecks and help maintain efficient network data transmission. Packet sniffers were never made to hack or steal information (Qadeer, M.A., Zahid, M., Iqbal, Arshad and Siddiqui, M.R., 2010)[15]. They had a different goal, to make things secure. But then everything has a dark side.

2.3 Packet Sniffer Procedure

Sabeel, A., Rajeev, S.G. and Chandrashekar, H.S. (2003)[19] mention that the technique behind packet sniffing on shared bus broadcast LANs is explained with the following example IEEE 802.3 Ethernet LANs employ a broadcast technology, i.e.

when a message is sent to another machine on the LAN, the message is sent to all the machines that are connected to the network. The machine's Network Interface Card (NIC) checks the destination address of the arriving packet. The card accepts the packet if it has the latter machine's address; otherwise, it is discarded. What a packet sniffer does is put the NIC into a "promiscuous mode." The NIC now does not discard packets that are not addressed to its machine. It silently accepts the packets. They also demonstrate the steps for creating a Linux packet sniffer. The main steps are creating a socket stream, setting the NIC into promiscuous mode and reading data from the open socket stream. They also state that the rest of the steps only deal with interpreting the headers and formatting the data and redirecting the data to the output stream. First step is creating a socket. On UNIX and its clones, communication points called sockets can be created. These help in the communication of end systems present in the network. When a socket is created, a socket stream, similar to the file stream, is created, through which data is read. Second step is setting the NIC to promiscuous mode. First, a reference to a structure called ifreq is needed. This is done by the statement, struct ifreq ifr. The ifreq structure is a rather large one that is passed on to the standard ioctl's to configure the network devices. The ifreq is an interface request structure used for socket ioctl. The ioctl provides an interface for controlling the behavior of devices, their descriptors and configuring the underlying services. Next, the NIC name, viz. eth0 usually is copied to the structure member ifr_name. The next step is to get the flags of the specified interface using the ioctl system call. The ioctl system call takes three arguments, viz. 1) The socket stream descriptor 2) The function that the ioctl function is supposed to perform. Here, the macro used is SIOCGIFFLAGS. 3) Reference to the ifreq member (Sabeel, A., Rajeev, S.G. and Chandrashekar, H.S., 2003)[19]. In the next step, the promisc flagmust be set. The promisc flag is a structure element of the ifreq structure. Setting this flag makes the NIC accept all the packets on the network. The IFF_PROMISC is a predefined macro. It is set using the statement, Ifr.ifr_flags |= IFF_PROMISC; the last step in this process is to set these flags to the NIC using the ioctl call. The same first and third parameters are used, but the second parameter to ioctl is changed to SIOCSIFFLAGS. Now comes the cumbersome part: protocol interpretation. To do this, the user is required to have some basic knowledge about the protocol that he or she intends to sniff. The protocol contents, its fields, field lengths must be known and the user must know what kind of data to expect in those fields.

Trabelsi, Z., Rahmani, H., Kaouech, K. and Frikha, M. (2004)[21] state that sniffers capture all packets in a network. To achieve this, the Sniffer sets the Network Interface Card (NIC) of the computer into a mode called "promiscuous mode". Then the NIC will blindly receive all packets and pass them to the system kernel. Packets that are not supposed to arrive to that PC are no longer blocked by the NIC. Moreover, they also indicate that many basics services, such as FTP, Telnet and SMTP, send clear text data in the packets. A Sniffer captures all packets and displays their contents on the hacker's computer screen, for examples the passwords used to authenticate during an FTP session, or the message of an email in SMTP packets (Trabelsi, Z., Rahmani, H., Kaouech, K. and Frikha, M., 2004)[21]. Besides that, Broadway, J., Turnbull, B. and Slay, J. (2008)[6] also mention that a packet sniffer is placed at strategic areas of a network to capture data packets sent between nodes of the network. They work by putting the Network Interface card (NIC) of a computer in 'promiscuous' mode. While in this mode the NIC will accept all packets it receives, instead of only accepting the packets addressed to its host machine. The packet sniffer then duplicates all packets and stores them to disk for later processing. When used on a broadcast network or placed at an appropriate location in the network a packet sniffer can capture all of the packets being sent across the network.

Rajan, S.B.R., Nirmelt, R.A., Rahuman, S.A.A., Kader, S.M.A. and Ganesh, S. (2009)[16] mention that network sniffer is interfaced with the detection system which captures all kind of packets (UDP, TCP, ICMP) in binary forms passing through the network via the Network Interface Card (NIC). First, it displays the packet information and then it sends the information to the worm detection system. It streamlines the binary data into the fields of the IP packet and stores the values in the corresponding variables of IP packet class (Rajan, S.B.R., Nirmelt, R.A., Rahuman, S.A.A., Kader, S.M.A. and Ganesh, S., 2009)[16]. The IP Header parsing function would parse the packets into two header information such as IP header and Transport layer headers. All parsed header information will be displayed. Additionally, Hong, D., Chi, M.and Xu, S.L. (2010)[11] also indicate that there are many kinds of methods to get the network packets. The raw socket is simple but it is not efficient and short of many functions like getting the network link layer protocol packets. As there are more packets in one time, the high performance must be assured. The WinPcap[4] is the most popular package which includes BPF and it is used for capture the network raw packet. It provides implementation-independent access to the underlying packet capture facility. The filter rules is used to get the special interesting network packets which can get more high efficiency. The system can get more portability because WinPcap[4] is a high level API which can transported into other system API.

Qadeer, M.A., Zahid, M., Iqbal, Arshad and Siddiqui, M.R. (2010)[15] define that when a packet is received by a NIC, it first compares the MAC address of the packet to its own. If the MAC address matches, it accepts the packet otherwise filters it. This is due to the network card discarding all the packets that do not contain its own MAC address, an operation mode called non promiscuous, which basically means that each network card is minding its own business and reading only the frames directed to it. In order to capture the packets, NIC has to be set in the promiscuous mode. Packet sniffers which do sniffing by setting the NIC card of its own system to promiscuous mode, and hence receives all packets even they are not intended for it (Qadeer, M.A., Zahid, M., Iqbal, Arshad and Siddiqui, M.R., 2010)[15]. So, packet sniffer captures the packets by setting the NIC card into promiscuous mode. To set a network card to promiscuous mode, all we have to do is issue a particular ioctl ( ) call to an open socket on that card and the packets are passed to the kernel. When the packets are sent from one node to another in the network, a packet has to pass through many intermediate nodes (Qadeer, M.A., Zahid, M., Iqbal, Arshad and Siddiqui, M.R., 2010)[15]. A node whose NIC is set in the promiscuous mode tends to receives the packet. The packet arriving at the NIC are copied to the device driver memory, which is then passed to the kernel buffer from where it is used by the user application. In Linux kernel, libpcap uses "PF_PACKET" socket which bypasses most packet protocol processing done by the kernel.

Furthermore, Zhang, S.F., Chen, X.B., Zhang, S.A. and Yan, H.C. (2009)[22] emphasize that the sniffer procedure is one kind uses the Ethernet the characteristic the network adaptive card (NIC) for the promiscuous pattern condition tool, once establishes with the card as this kind of pattern, it can receive the transmission in the network each information packet. In the ordinary situation, it only receives and own address related information packet, namely transmits the local main engine information packet. Must enable Sniffer to receive and to process this way the information, the system needs to support under BPF, Linux to need to support

SOCKET-PACKET. But in the ordinary circumstances, the network hardware and the TCP/IP storehouse does not support the receive or the transmission the data packet which has nothing to do with the local computer, therefore, in order to bypass the standard the TCP/IP storehouse, the network card must establish the promiscuous pattern which just started for us to say. In the ordinary circumstances, must activate this way, the essence must support this kind of false equipment BP filter, moreover needs the root jurisdiction to move this kind of procedure, therefore sniffer needs the root status installment, if only were enters the human system by local subscriber's status, then not impossible to call searches root the password, because could not move Sniffer. Based on the Sniffer such pattern, may analyze each kind of information packet and describe the network structure and the use machine, because it receives any the data packet which transmits in the identical webpage, therefore also has the capture password, each kind of information, the secret documents and so on some encryption information possibility (Zhang, S.F., Chen, X.B., Zhang, S.A. and Yan, H.C., 2009)[22].

Moreover, according to Phang, S.Y., Lee, H.J. and Lim, H.T. (2008)[14], Libpcap is an open source C library to put our Network Interface Card in promiscuous mode. In other words, libpcap is an application programming interface (API). This packet capture library provides an interface to packet sniffing software for all packets on the network whether is destined to own or other hosts. The advantage of using this API is that the code is portable and machine independent. This is one of the approaches that can be used to design and implement a packet sniffer. Another approach will be designing and implementing the packet sniffer using the Raw Socket API. By creating a Raw Socket and bind to the network card interface driver, any packet will be received by this binding interface and being passed to the sniffing application. Blindly received any packets without filter accordingly of the intended and interested packets is not a good practice where it will be a resource and time consuming process. As Sniffing program run as user-level processes, packets must be copied across the kernel to user space and this copying can be minimized by deploying a kernel agent called a packet filter, which discards unwanted packets as early as possible. Berkeley Packet Filter (BPF) is the first facility that available in BSD variants. Several works and studies have been done over the past in optimizing the packet filter. System architectures are also designed specifically for network monitoring in. From these works, these architectures provide rich sets of functions. These architectures provide great options for developing packet monitoring software.

2.4 Existing Packet Sniffer

2.4.1 Raw Socket

Raw Socket is a software interface through which a process can fully control the packets it sends or receives to or from the network. This is done by bypassing the network stack of the host and communicating directly with the device driver. Besides that, it only works on Windows 2000/XP or greater. It also allows you to capture TCP/IP packets on your network without installing a capture driver. However, raw Sockets have been known to have problems with portability because of which one might frequently run into problems when switching platforms. Moreover, it also cannot capture outgoing UDP and ICMP packets and your program needs administrator rights on the machine to use raw sockets.

2.4.2 WinDump

WinDump[3] is the porting to the Windows platform of tcpdump (Mezzalama, 2000)[12]. WinDump[3] is fully compatible with tcpdump but introduces some extensions to work better in the Win32 environments. The WinDump.exe executable file is linked with libpcap for Win32, therefore can run both under Windows 95/98 and under Windows NT/2000. To run WinDump[3], the correct version of the BPF