Development of a Flexible Real-Time Monitor for an Enterprise Network

Mumtaz M. AL-Mukhtar
Information Eng. College
AL-Nahrain University
Baghdad-Iraq

ABSTRACT

This paper presents the design and implementation of a technique that can be used in a real-time network monitoring. It provides a Real-Time Network Monitoring System (RTNMS) that allows a host to capture any packets in a network by putting the host's Network Interface Card (NIC) into the promiscuous mode. Modular design has been devised. RTNMS consists of five modules that are integrated to give the required features. These features automatically give system administrators a helping hand in management and fault detection on a network. RTNMS is user-friendly because it has been augmented by a user interface to assist in navigating the capture and analysis of packets. It can be tapped into an enterprise network with high flexibility. It can significantly increase the efficiency of data capture and network monitoring through the association of various modules.

General Terms

Information Technology - Network Monitoring.

Keywords

Passive Monitoring, Network Packet Capture, Promiscuous Mode, Real-Time, Network Analysis.

1. INTRODUCTION

With the gradual expansion of network bandwidth, the traffic is also increasing in multiples with more and more complicated contents. As a result, network monitoring turns out to be a critical task in any serious network management solution.

Most of the current network monitoring tools are developed for showing specific aspects of network traffic. The way in which network information should be presented can although be different for each purpose. Generally, most of the tools attend specific requirements on how and which network information is presented [1, 2].

Network monitoring can be broadly classified into active monitoring and passive monitoring [3, 4]. Active or "intrusive" monitoring uses equipment that divides the circuit into two segments and allows the flow of traffic to be monitored, and actively transmitted from one side of the monitor point to another. Passive or "non-intrusive" monitoring uses equipment that taps into a network and does not interfere with the flow of network traffic. This could be accomplished by one or more sniffers placed at certain location(s) in the network to capture the traffic that they can hear. This method has the advantage of not needing to change the software running at hosts and hence easily deployable [5].

On a complementary plane, network monitoring can be classified into Offline analysis and Real-time analysis [6]. The former involves collecting the traffic trace in real networks and then analyzing the network semantics offline. On the other hand, Real-time monitoring and analysis help in significantly reducing the troubleshooting delays where time is critical. Though there are some tools [7, 8, 9] available for providing real-time network statistics, they are either for commercial use or provide insufficient information to the users regarding the local network.

In this paper, we propose a real-time monitoring approach that could find a way to run the analysis more quickly with smaller demands on system resources. We can obtain the quantitative and measured data of the network through network traffic capturing and analysis. It is aimed to help the network administrators to visualize and analyze the network's performance by providing statistics and graphs for the entire network in general – and each node in specific in real-time. This is critical for the efficient deployment and reliable performance of the network.

2. REAL-TIME NETWORK MONITORING SYSTEM DEVELOPMENT

The Real-Time Network Monitoring System (RTNMS) architecture relies on tapping a RTNMS host into the monitored network. The Network Interface Card (NIC) of the host monitor is set into the promiscuous mode in order to capture all packets going through the network in real-time.

The main steps in the development of the RTNM are:

1. Creating a socket stream.
2. Setting the NIC into a promiscuous mode.
3. Reading data from the open socket stream.
4. Real-Time analysis by interpreting the headers, formatting the data, and redirecting the data to the output stream.
5. The graphical visualize subsystem obtains the data from the data gathering subsystem and converts it to a form that can be displayed graphically.
6. Storing packets or relevant parts of packets in the secondary storage for later tracking or inspection by a network manager.

2.1 RTNMS Model and Sockets

In socket-based network programming, we do not directly access the network interface device to send or receive packets. Instead, an intermediary file descriptor is created to handle the programming interface to the network. The special file descriptors used to reference network connections are called sockets [10]. The socket defines the following:

- A specific communication domain, such as a network connection.
- A specific communication type, such as a stream or datagram.
- A special protocol, such as TCP or UDP.

After the socket is created, it must be bounded to either a specific network address or port on the system, or to a remote network address or port. Once the socket is bounded, it can be used to send or receive data from the network.

RTNMS Model is basically based on Windows Socket Model that consists of four main layers: Application layer, Protocol Stack TCP/IP layer, Network Driver layer and Network Interface layer. Windows Socket API works as an intermediate layer between WinSock Application and Network Layers. Figure 1 illustrates the Real-Time Network Monitoring System Model. Winsock library version 2.2 has been used for the development of the RTNMS which is a part of the current Windows operating systems releases.

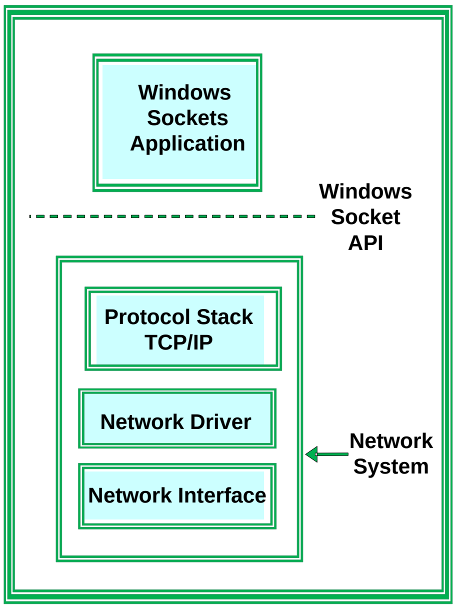


Fig 1: RTNMS Model Based on Windows Sockets

2.2 Promiscuous Mode

In a network, promiscuous mode allows a network device to intercept and read each network packet that arrives in its entirety [10]. In an Ethernet technology, promiscuous mode is a mode of operation in which every data packet transmitted can be received and read by an NIC even when they are not intended for it. The developed RTNMS sets the NIC of the RTNMS host machine into this mode. To do so, 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 for further treatment.

3. FUNCTIONS OF REAL-TIME NETWORK MONITORING SYSTEM

The RTNMS has diverse functions; these functions are depicted in figure 2.

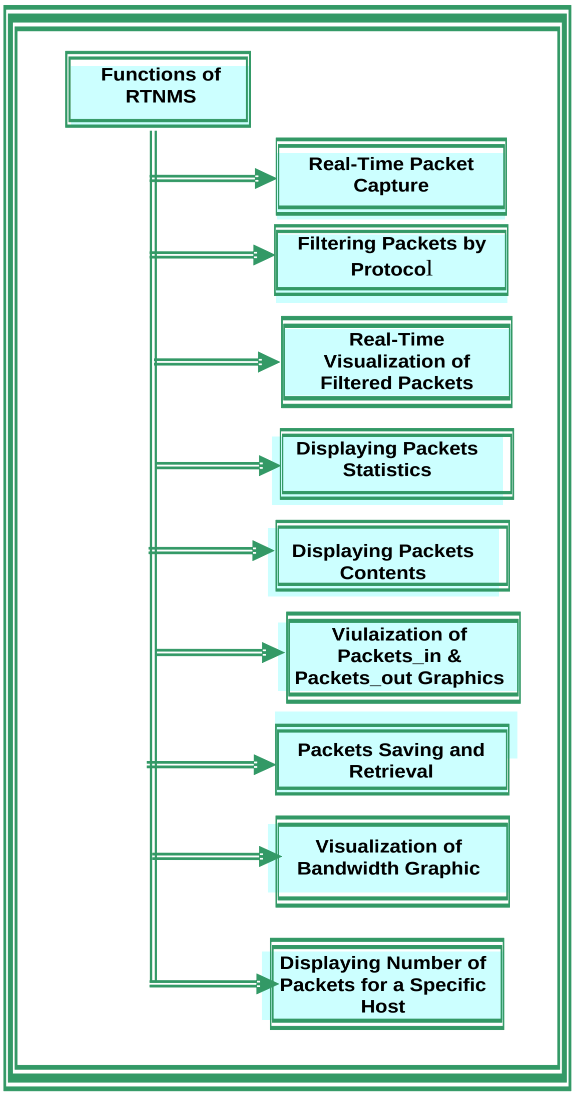

Functions of RTNMS
Real-Time Packet Capture
Filtering Packets by Protocol
Real-Time Visualization of Filtered Packets
Displaying Packets Statistics
Displaying Packets Contents
Viulaization of Packets_in & Packets_out Graphics
Packets Saving and Retrieval
Visualization of Bandwidth Graphic
Displaying Number of Packets for a Specific Host

Fig 2: Functions of RTNMS

4. STRUCTURE OF RTNMS

The Structure of RTNMS consists of five modules as shown in figure 3. Each module comprises procedures, functions and events that are declared via Flow controls in the following subsections.

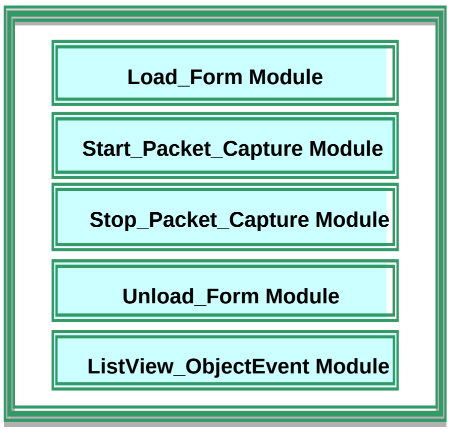

Load_Form Module
Start_Packet_Capture Module
Stop_Packet_Capture Module
Unload_Form Module
ListView_ObjectEvent Module

Fig 3: RTNMS Structure

4.1 Load_Form Module

It is the first module to be executed. It checks the availability of Windows Socket and fetches NIC's list of IP addresses of the RTNMS host as shown in figure 4. In case of failure, one of the following error messages will be displayed according to the type of error detected:

- The underlying network is not ready for network communication.
- The version of Windows Socket API support requested is not provided by this Windows Socket implementation.
- The Windows Sockets version specified by the application is not supported by this Dynamic Link Library (DLL).

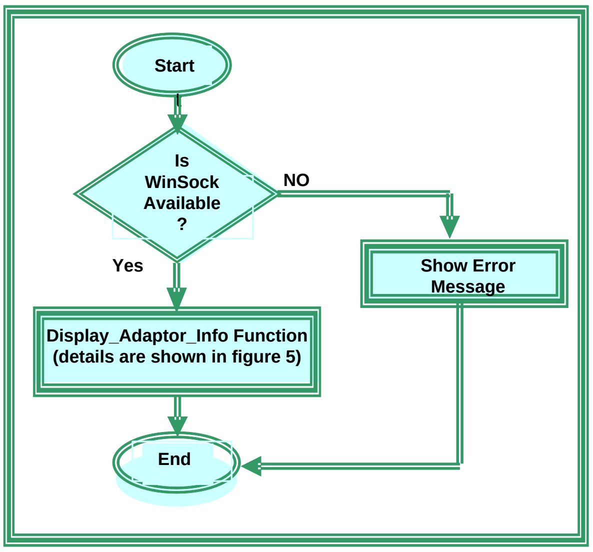

Start
Is WinSock Available ?
NO
Yes
Show Error Message
Display_Adaptor_Info Function (details are shown in figure 5)
End

Fig 4: Flow Control of Load_Form Module

Figure 5 shows the actions implemented by Display_Adaptor_Info function.

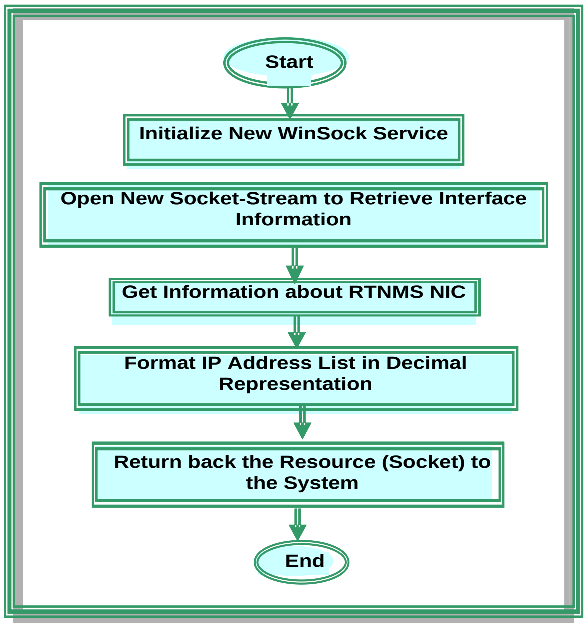

Start
Initialize New WinSock Service
Open New Socket-Stream to Retrieve Interface Information
Get Information about RTNMS NIC
Format IP Address List in Decimal Representation
Return back the Resource (Socket) to the System
End

Fig 5: Flow Control of Display_Adaptor_Info Function

4.2 Start_Packet_Capture Module

When the form has been loaded and list of IP Addresses are displayed, the first thing that executed by this module is checking the version of Windows Operating System compatibility with Winsock version used (version 2.2). The next step- Handling Window Message- uses a sub classing to enable the system to intercept every message that is sent to a window. Thereafter Winsock service is initialized to start capturing network packets by setting the NIC into promiscuous mode. This is illustrated in figure 6.

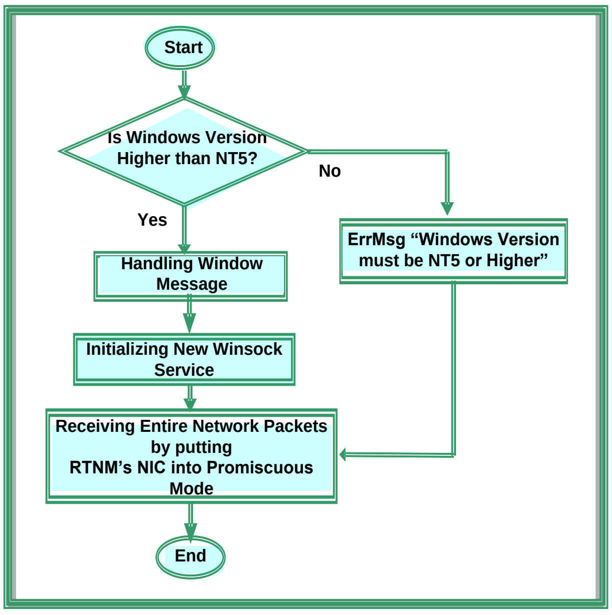

Start
Is Windows Version Higher than NT5?
No
Yes
ErrMsg "Windows Version must be NT5 or Higher"
Handling Window Message
Initializing New Winsock Service
Receiving Entire Network Packets by putting RTNM's NIC into Promiscuous Mode
End

Fig 6: Flow Control of Start_Packet_Capture

4.3 Stop_Packet_Capture Module

For stopping packet capturing process Stop Packet Capture Module is activated as shown in figure 7. First function that executes is requesting Windows message-based notification of network events for a socket. The connections are closed and the resources (socket and its functions) are returned to the system in order to be used by another new connection. Then ending Winsock session executes. Thereafter subclassing is stopped. These functions terminate the use of the Winsocket Dynamic Link Library (WS2_32.DLL).

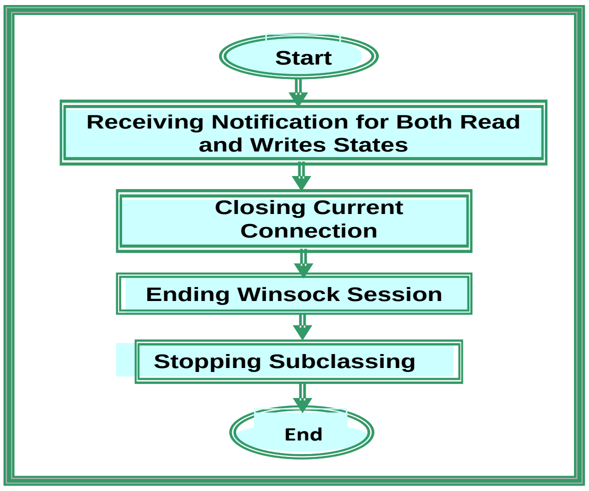

Start
Receiving Notification for Both Read and Writes States
Closing Current Connection
Ending Winsock Session
Stopping Subclassing
End

Fig 7: Flow Control of Stop_Packet_Capture Module

4.4 Unload_Form Module

The main function of unload_form is closing the Winsock connection, releasing the current used Windows Socket to the system and exiting the Network Packet Capture form. The Unload_ Form Module steps are the same as steps of Stop Capturing Module, but exiting the capture form.

4.5 ListView_ObjectEvent

The function of this module is showing details of each selected packet mainly Source IP Address, Destination IP Address, Source Port, Destination Port, Protocol, and packet Length. This module calls the *Display_Packet* function that formats the display text box to display the packet details in readable format (ASCII and Hexadecimal).

5. THE RTNM INTERFACE

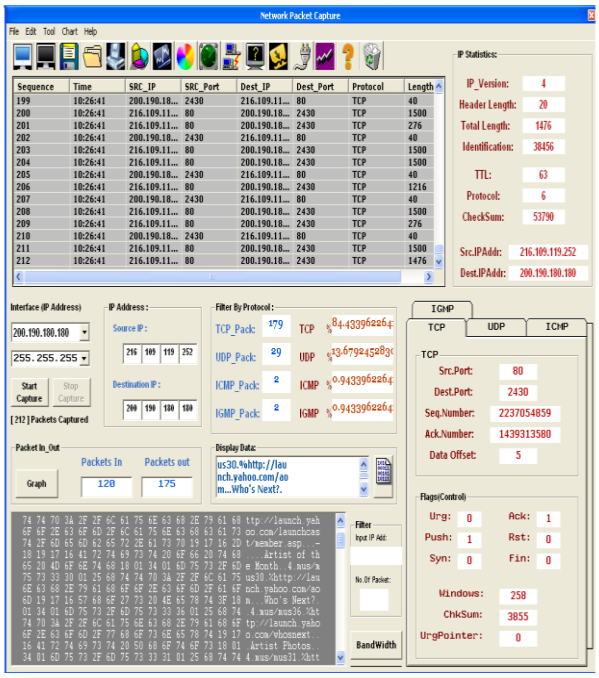
The main system form contains a command button: Packet Capture & Monitoring. This button would be clicked when a system initialization is required. This would activate the Network Packet Capture Form. This form can be used for real-time packet capturing and packet analyzing. The components of Network Packet Capture Form are:

- *Network Packet Capture Form*

Figure 8 depicts the display window of this form. Details of captured packets appear in the Captured Packet List View (Packet sequence, Time of capturing, Source IP address, Source Port, Destination IP Address, Destination Port, Protocols and Length of each packet). Moreover, some other useful information are displayed like: number of captured packets by each IP Address, filtered packets by protocols, number of packets passing into the RTNMS host, number of packets going out of RTNMS host, detailed statistics of each packet, total number of captured packets and percentages of filtered packets.

- *Packet Selection*

More details of captured Packets can be shown by clicking on each packet in the Captured Packet List View. When a specific packet is selected, its contents will appear in the text box (in hexadecimal and ASCII text format). The packet's header details are showed in a group of text boxes (IP Statistics frame) on the right hand side of the Network Packet Capture Form as shown in figure 8.



Sequence	Time	SRC_IP	SRC_Port	Dest_IP	Dest_Port	Protocol	Length
199	10:26:41	200.190.18...	2430	216.109.11...	80	TCP	40
200	10:26:41	216.109.11...	80	200.190.18...	2430	TCP	1500
201	10:26:41	216.109.11...	80	200.190.18...	2430	TCP	276
202	10:26:41	200.190.18...	2430	216.109.11...	80	TCP	40
203	10:26:41	216.109.11...	80	200.190.18...	2430	TCP	1500
204	10:26:41	216.109.11...	80	200.190.18...	2430	TCP	1500
205	10:26:41	200.190.18...	2430	216.109.11...	80	TCP	40
206	10:26:41	216.109.11...	80	200.190.18...	2430	TCP	1216
207	10:26:41	200.190.18...	2430	216.109.11...	80	TCP	40
208	10:26:41	216.109.11...	80	200.190.18...	2430	TCP	1500
209	10:26:41	216.109.11...	80	200.190.18...	2430	TCP	276
210	10:26:41	200.190.18...	2430	216.109.11...	80	TCP	40
211	10:26:41	216.109.11...	80	200.190.18...	2430	TCP	1500
212	10:26:41	216.109.11...	80	200.190.18...	2430	TCP	1476

IP Statistics:

IP_Version:	4
Header Length:	20
Total Length:	1476
Identification:	38456
TTL:	63
Protocol:	6
CheckSum:	53790
Src.IPAddr:	216.109.119.252
Dest.IPAddr:	200.190.180.180

Interface (IP Address): 200.190.180.180 / 255.255.255
Start Capture | Stop Capture
[212] Packets Captured

IP Address: Source IP: 216 109 119 252 — Destination IP: 200 190 180 180

Filter By Protocol:

TCP_Pack:	179	TCP	%84.433962264
UDP_Pack:	29	UDP	%13.679245283
ICMP_Pack:	2	ICMP	%0.943396226
IGMP_Pack:	2	IGMP	%0.943396226

Packet In_Out: Graph — Packets In 120 — Packets out 175

Display Data: us30.%http://launch.yahoo.com/aom...Who's Next?.

```
74 74 70 3A 2F 2F 6C 61 75 6E 63 68 2E 79 61 68 ttp://launch.yah
6F 6F 2E 63 6F 6D 2F 6C 61 75 6E 63 68 63 61 73 oo.com/launchcas
74 2F 6D 65 6D 62 65 72 2E 61 73 70 19 17 16 2D t/member.asp...-
18 19 17 16 41 72 74 69 73 74 20 6F 66 20 74 68 ....Artist of th
65 20 4D 6F 6E 74 68 18 01 34 01 6D 75 73 2F 6D e Month..4.mus/m
75 73 33 30 01 25 68 74 74 70 3A 2F 2F 6C 61 75 us30.%http://lau
6E 63 68 2E 79 61 68 6F 6F 2E 63 6F 6D 2F 61 6F nch.yahoo.com/ao
6D 19 17 16 57 68 6F 27 73 20 4E 65 78 74 3F 18 m...Who's Next?.
01 34 01 6D 75 73 2F 6D 75 73 33 36 01 25 68 74 .4.mus/mus36.%ht
74 70 3A 2F 2F 6C 61 75 6E 63 68 2E 79 61 68 6F tp://launch.yaho
6F 2E 63 6F 6D 2F 77 68 6F 73 6E 65 78 74 19 17 o.com/whosnext..
16 41 72 74 69 73 74 20 50 68 6F 74 6F 73 18 01 .Artist Photos..
34 01 6D 75 73 2F 6D 75 73 33 31 01 25 68 74 74 4.mus/mus31.%htt
```

Filter — Input IP Add: — No.Of Packet: — BandWidth

IGMP | TCP | UDP | ICMP

TCP

Src.Port:	80
Dest.Port:	2430
Seq.Number:	2237054859
Ack.Number:	1439313580
Data Offset:	5

Flags(Control)

Urg:	0	Ack:	1
Push:	1	Rst:	0
Syn:	0	Fin:	0

Windows:	258
ChkSum:	3855
UrgPointer:	0

Fig 8: Network Packet Capture *Form*

- *Packet Filter Chart*

Captured packets that are filtered by protocols (TCP, UDP, ICMP and IGMP) can be shown graphically by using Packet Filter Chart tool as shown in figure 9.

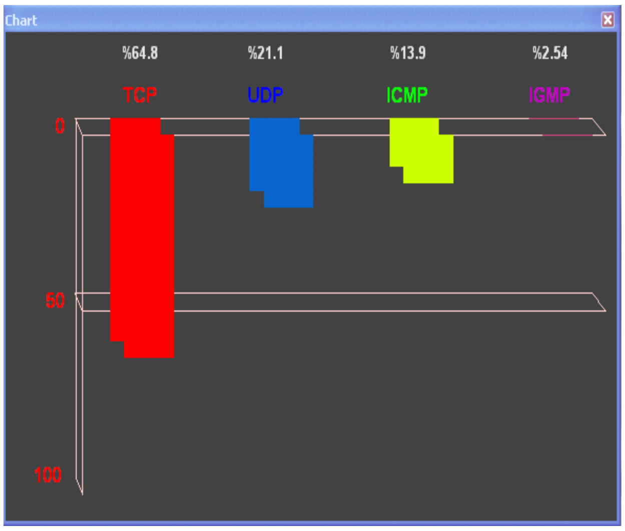


Fig: 9 Packet Filter Chart

- *Packets (IN_OUT)*

The number of packets that are sent to RTNM host and number of packets that went out of RTNMS host can be shown graphically by using Packets (In-Out) as shown in figure 10.

- *Filter By IP address*

Number of packets that each IP Address generates inside the network can be calculated using this tool.

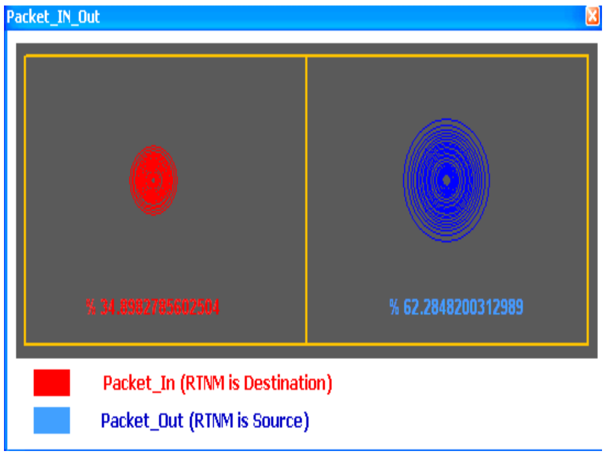
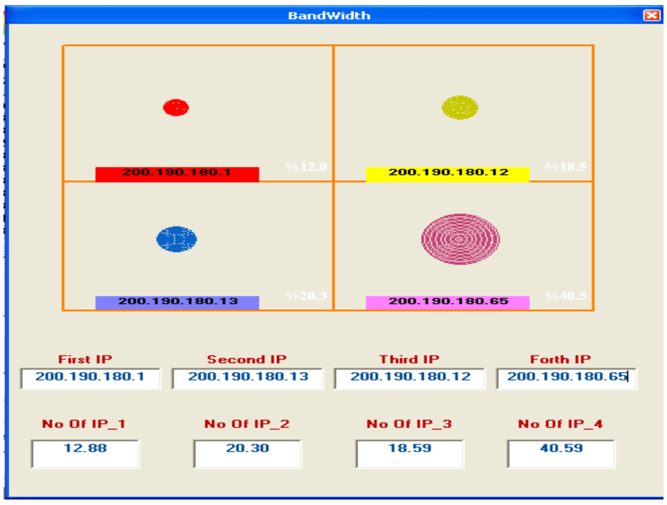

Packet_IN_Out
% 34.8982785602504
% 62.2848200312989
🟥 Packet_In (RTNM is Destination)
🟦 Packet_Out (RTNM is Source)

Fig: 10 Packets (IN_OUT)

- *Bandwidth Tester*

To get information about network bandwidth usage by hosts, this tool gives a clear view about how bandwidth is used by any host inside the network. Graphically the bandwidth used by four IP addresses that work within the network is shown in figure 11.



First IP	Second IP	Third IP	Forth IP
200.190.180.1	200.190.180.13	200.190.180.12	200.190.180.65

No Of IP_1	No Of IP_2	No Of IP_3	No Of IP_4
12.88	20.30	18.59	40.59

Fig 11: Used Bandwidth by Hosts

- *Control Tool Bar*

This is the main Toolbar that contains all required control buttons (i.e. Start Capture, Stop Capture, Save Packet, Open Packet file, Packet Filter Chart, Packet (In-Out), and Bandwidth) as shown in figure 12.

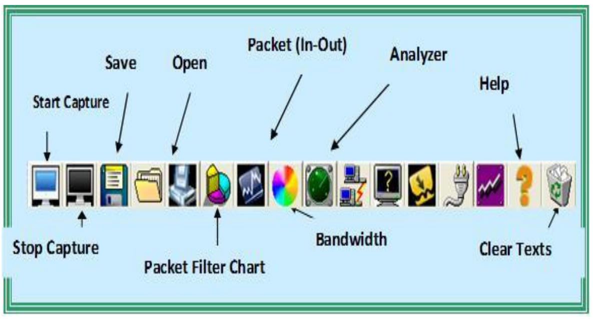


Fig: 12 Toolbar

6. CONCLUSION

RTNMS is a simple network monitoring tool that can be efficiently used for network administration. It has the capability of real-time analysis by analyzing the data as it comes of the wire. It provides a user with access to packets on a network through an easy-to-use graphical user interface.
RTNMS could be employed efficiently in networking management and troubleshooting processes. It enables network managers to evaluate and examine the data running through their network by troubleshooting network performance problems and identifying certain network faults in real time.
Besides its usage in technical environment, it can be used for educational and research purposes. It can be used to help understand packets' architecture and traffic patterns generated by common network applications. Moreover, it can be used to evaluate protocol performance and assist in protocol development thus improving the network performance as well as user experience.

7. REFERENCES

[1] Ying L., Ming W., Sidong Z., and Hongke Z. 2010. Research of the Network Information Monitoring System Based on P2DR Model. In Second International Conference on Computer Modeling and Simulation, pages: 190-194.

[2] Hofstede R., and Fioreze, T. 2009 .SURFmap: A Network Monitoring Tool based on the Google Maps API. In IFIP/IEEE International Symposium on Integrated Network Management (IM '09), pages: 676 – 690.

[3] Zangrilli M., and Lowekamp B. 2003. Comparing Passive Network Monitoring of Grid Application Traffic with Active Probes. In Proceedings of the Fourth International Workshop on Grid Computing (GRID'03), pages: 84-91.

[4] Papadogiannakis A, Vasiliadis G., Antoniades D., Polychronakis M., and Markatos E. 2012. Improving the Performance of Passive Network Monitoring Applications with Memory Locality Enhancements. Computer Communications Journal, Volume 35, Issue 1, pages: 129-140.

[5] Ansari S., Rajeev S.G., and Chandrasekhar H.S. 2003. Packet Sniffing: A Brief Introduction. IEEE Potentials, Volume: 21, Issue: 5, pages: 17–19.

[6] Kiszka J., Wagner B., Zhang, Y., and Broenink, J. 2005. RTnet – A Flexible Hard Real-Time Networking Framework. In 10th IEEE Conference on Emerging Technologies and Factory Automation (ETFA'05), pages: 456-464.

[7] Pande B., Gupta D. and Sanghi D. 2005. The Network Monitoring Tool-PickPacket. In Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05), pages: 191-196.

[8] Dabir A., and Matrawy A. 2007. Bottleneck Analysis of Traffic Monitoring Using Wireshark. In 4th International Conference on Innovations in Information Technology (IIT '07), pages: 158 – 162.

[9] Dashtbozorgi M. and Azgomi M. A. 2012. A High-Performance and Scalable Multi-core Aware Software Solution for Network Monitoring. The Journal of Supercomputing, Volume**:** 59 Issue: 2, Pages: 720-743.

[10] Johnson M. H. 2010. Windows Sockets Programming. Addison-Wesley Professional.