An Effective Wireless Monitoring Tool (EWMT)

Dr. Mumtaz M. AL-Mukhtar*, Dr.Shaimaa H. Shaker**
& Afnan Adil Abd Alraheem ***

Received on:24/6/2009
Accepted on:3/6/2010

Abstract

As computer networking has expanded and matured, there has been a growing requirement to collect traffic statistics for understanding the current load and planning future connectivity. This requirement leads to consider the general problem of a software packet monitor for analyzing traffic patterns and gathering statistics.

This Paper presents a software monitoring system named "Effective Wireless Monitoring Tool" (EWMT) which has been built to help the network administrator to discover network problems as soon as possible.

The proposed system achieves its goal by sniffing, capturing, and analyzing the network traffic data packets using a passive monitoring scheme without affecting or interfering with the network information flow.

This is achieved by configuring the Network Interface Card (NIC) in promiscuous mode. The EWMT extracts and views different valuable detailed information concerning the network in human- readable format. It taps network protocols that run at the IP level, TCP level and other protocols at the application level as DNS, HTTP, and FTP.

EWMT system is constructed using Microsoft Visual C++ V8.0 development environment along with 3.5.NET framework.

Keywords: Wireless Monitoring, IEEE 802.11, Promiscuous Mode.

نظام مراقبة شبكة لاسلكية فعّال

الخلاصة

نظرا لاتساع استخدام شبكات الكومبيوتر ظهرت الحاجة الى تجميع الاحصائيات عن انسياب المعلومات لاغراض التخطيط والتطوير لانظمة الاتصالات. هذه الحاجة ادت الى اعتبار ان المشكلة الاساسية تكمن في مراقبة وتحليل سير حزمة البيانات وتجميع الاحصائيات.

يتضمن البحث بناء نظام المراقبة للشبكة اللاسلكية سمي EWMT لغرض مساعدة مدير الشبكة لاكتشاف مشاكل الشبكة وبالسرعة الممكنة. ان النظام المقترح حقق هدفه بواسطة استراق وتحليل سير حزمة البيانات بدون تاثيراوتداخل لانسياب شبكة المعلومات وذلك بجعل الشبكة تعمل في صيغه الحاله غير الشرعيه "Promiscuous Mode"(وهي الصيغه التي تجعل بطاقة الشبكه الالكترونية لا تقوم بفحص عنوان المرسل اليه) ويقوم النظام باستخلاص المعلومات المفصلة المتعلقة بالشبكة بصيغة واضحة و مفهومة للمستخدم. ان النظام المقترح يعمل على البروتوكولات في طبقةIP و TCP بالاضافه الى البروتوكولات التطبيقيه مثل DNS,HTTP,FTP.

* Engineering College, University of AL-Nahrain / Baghdad
** Computer Engineering and Inf. Technology Department, University of Technology/Baghdad
*** Engineering College, University of Al Mustansiriya /Baghdad

https://doi.org/10.30684/etj.28.16.8
2412-0758/University of Technology-Iraq, Baghdad, Iraq

Introduction

Wireless technology is the method of delivering data from one point to another without using physical wires, which includes radio, cellular, infrared, and satellite. Wireless communication is one of the fastest growing technologies. The demand for connecting devices without cable is increasing everywhere; therefore Wireless LANs are found on college campuses, office buildings, and public areas [1, 2].

Wireless networks are classified into three main categories: WWAN (Wireless Wide Area Network), WLAN Wireless Local Area Network), and WPAN (Wireless Personal Area Network). WWAN includes technologies such as 2G cellular and Global System for mobile communication. WLAN includes IEEE 802.11g standard and others. WPAN includes Bluetooth and IR devices. Wireless communication uses the wavelengths from the radio frequency (RF) band up to the Infra-red (IR) band [3].

The most popular network in the wireless domain is the IEEE 802.11 wireless local area network. This is mainly because of the advantages that its systems possess. Its key characteristics, such as interoperability, mobility, flexibility, and cost-effective deployment, have led to its gaining vast support across enterprises, the public sector, homes, and data service providers [4]. Because of the complexity of networks and attacks on networks, analysts require as much information as possible about their own network infrastructure. Knowledge of host connectivity and traffic details is essential when one must analyze, investigate or react to computer or network performance or security attacks therefore EWMT (Effective Wireless Monitoring Tool) is proposed.

1. Related works

Several related network monitoring systems are presented in this field.

- Ngo et al [5] proposed a novel resource usage monitoring tool for ad-hoc wireless Ethernet networks, called WANMon (Wireless Ad-hoc Network Monitoring Tool).WANMon can be installed on a wireless node to monitor resource usage (such as network usage , power usage, memory usage, and CPU usage) at the node.

- Pande et al [6] designed a network monitoring tool that handles the conflicting issues of network monitoring and privacy through its judicious use. PickPacket has four components "The PickPacket Configuration File Generator" for assisting the user in setting up the parameters for capturing packets, the "PickPacket Packet Filter" for capturing packets, the "PickPacket Post-Processor" for analyzing packets, and the "PickPacket Data Viewer" for showing the captured data to the user.

- Karygiannis et al [7] presented a collection of host-based kernel modules for the GNU/Linux operating system. These kernel modules monitor the network traffic traversing the host's wireless network interface and write a number of network traffic metrics into the Linux"/proc" virtual directory.

- Arun Chhetri and Rong Zheng [8] introduce WiserAnalyzer, a passive monitoring tool to analyze WLANs for inference of nodal relationships, detection of malicious usage or incompliance to the IEEE 802.11 specification. WiserAnalyzer can determine the relation among wireless nodes and finegrained device characteristics. Such information can be utilized as the basis for cross-layer diagnosis of WLANs.

2. IEEE 802.11 Architecture

Two types of network structures are defined by the IEEE 802.11 standard: the independent (Ad-hoc) network and the infrastructure network [9].

3.1 Peer-to-peer (Ad-hoc) Mode

A BSS (Basic Service Set) consists of a set of stations and that communicate directly with each other without access point [10].

3.2 Infrastructure (Dependent) Mode

BSS (Basic Service Set) includes stations and an AP Access point). An AP is a special station which can access a DS (Distributed System) that connects the BSSs [9].

4. Promiscuous Mode

Normally, a computer's network interface hardware checks the destination address of each incoming frame to determine whether the frame should be accepted. However, a computer's network interface hardware can be configured by software into promiscuous mode, which overrides the conventional address checking. Once in promiscuous mode, the network interface does not check the destination address of the incoming frame, but accepts all frames [11].

5. Mirror Port

A switch operates very differently from a hub. When a switch receives information from a computer it does not just blindly send it to all other computers; instead it sends unicast traffic to the desired destination computer. Therefore, the sniffer does not see this traffic but configuring a switch with a mirror port makes this possible. Most switches come with a feature known as port mirroring, or port spanning [12]. To mirror ports, we need to configure the switch to duplicate the traffic from a port we want to monitor to a port we are connected to with our network analyzer. This feature was implemented just for this purpose that is to analyze the network traffic for troubleshooting.

6. Proposed Network Layout Architecture

The proposed network layout architecture relies on tapping EWMT into the monitored network. Connecting the EWMT host on a mirror port would make it possible to capture the packets being sent and received through the access point. The NIC of EWMT host is configured in the promiscuous mode in order to capture all packets going and coming inside the network without caring if packets are destined to its IP address or not. A network layout illustrating this architecture is shown in figure 1.

7. EWMT Hierarchy

The EWMT is built on a network library (Netlib), which has been designed and implemented for the proposed framework. To build a network library (Netlib); .Net Socket class of System and .Net namespace are used. This system consists of three modules: GUI which is responsible for all user interface related operation, Mainwindow that is responsible for all the client area related operations, and Tools which deals with all none GUI operations.

These modules collaborate with each other to aggregate the system. Figure 2 demonstrates the main system modules.

8. EWMT Processing Stages

The EWMT comprises four basic processing stages as demonstrated in figure 3. Through these basic stages the EWMT will satisfy the system goals in network monitoring. Each stage includes several functions that are implemented in sequence or are synchronism on executing.

8.1 Initialization Stage:

This stage is responsible for creation and initialization of sockets. A socket is the programming technique used in EWMT for packet capturing. This stage involves the following functions:

1- Create socket, using the specified address family and a socket type, enables the network to receive the designated packets.

Socket (Address Family, Socket Type, Protocol Type)

- ***Address Family =InterNetwork***, specifies the standard address families used by the Socket class to resolve network addresses. InterNetwork indicates that an IP version 4 address is expected when a Socket connects to an endpoint.
- ***Socket Type = raw***, supports access to the underlying transport protocol. Specifically the application must provide a complete IP header when it is sending packets. Received datagrams are returned with the IP header.
- ***Protocol Type=IP***, specifies the network protocol to use when communicating on the Socket**.**

2- Bind a sniffer socket with the computer IP by following these steps:

- First create the local IPEndPoint (.Net class) from which we intend to communicate data. IPEndPoint is used to represent a network endpoint as an IP address and a port number; 0 is used for the port number which means any port**.**
- Bind method associates a socket with a local end point.

3- Check if any error has occurred while binding. A Socket Exception is thrown by the socket when an error occurs with the network to send a report to the user.

4- Setup socket by using SetSocketOption method (standard.Net Socket method) to receive a packet at the IP layer including its header.

Set Socket Option (SocketOptionLevel, SocketOptionName, Boolean)

- ***SockeOptionLevel*** is set to IP and indicates that the application provides the socket options only to IP sockets in order to receive a packet at the IP layer.

- ***SocketOptionName*** is set to Header Included and indicates that the application provides the IP header for outgoing datagrams; in other words is set to receive a packet including its header.
- **Boolean** is set to True to enable the option.

5- Set the socket to receive all packets from any IP on any port (set the socket to promiscuous mode) by using IO_Control method (standard.Net Socket method).
IO_Control code is set to ***SIO_RCVALL*** to enable receiving all IPv4 packets on the network. The socket must have an address family Internet work. The socket type must be Raw, and the protocol type must be IP. If a return value equals zero, this means the process has succeeded.

6- Check if any error occurs with socket setting. If ***IO_Control*** method returns none zero value that means an error has occurred therefore a report is sent to the user.

7- Enforce sniffer socket to begin receiving data on its buffer by using ***Begin Receive ( )*** method (standard.Net Socket method). The ***Begin Receive ( )*** method begins to asynchronously receive data from a connected socket; it registers an AsyncCallback method to run when data is available on the socket. When data is available, the registered method is called, and the***End Receive ()*** method reads it.

8- Call the ***End Receive*** method to complete the asynchronous read operation started in the Begin Receive method. The End Receive method (standard.Net Socket method) will read as much data as is available up to the number of bytes that are specified in the size parameter of the Begin Receive method.

9- Check if any error occurs with ending captured data. If no error occurs the process stage begins; else send the report message to the user.

8.2 Process Stage:

This stage comprises the following functions:

1- Parses out an IPv4 datagram which includes IP header part and data part by using ***HandleDatagram ( ) function.***

2- Extract IP version value from IP4 datagram by using specific methods in ***HeaderParser*** class:

To Byte (array <Byte> datagram, int offset, int length); this method returns Byte (8bit) value and take three parameters:

Array <Byte>**:** Represent array of byte

Offset: Represents the bit position at which the convert operation starts from it.

Length: Represents the number of bits which participate in the convert operation.

3- Captured packet has two parts: header and data; function ***getIPv4Header ( )*** is used to extract IP header from captured packet.

4- Extract IP header's information for each field such as type of service

TOS, total length, identification ID, protocol , time to live TTL using ***Netlib::HeaderParser*** methods which includes:

- ***ToByte***: this method returns a Byte value and takes three parameters as described in step 2.
- ***To Int***: it is the same as ***ToByte*** but it returns Int (32bit) instead of Byte.
- ***ToShort***: it is the same as ***ToByte*** but it returns Int (16bit) instead of Byte.
- ***ToUInt***: it is the same as ***ToByte*** but it returns unsigned int (32bit) value instead of Byte.

During the extraction of protocol field, **ToByte** method is used with setting the offset value to 72 and length to 8 because protocol field starts in bit No.72 and has a length of 8 bits.

5- Check protocol type by checking the extracted value of its field:

a- If the extracted value is equal to 6, it is a TCP protocol and then extract TCP header information using ***ToTcpHeader( )*** function.

b- If the extracted value is equal to 17, it is a UDP protocol then extract UDP header information by using ***ToUdpHeader ( )*** function.

c- If the extracted value is equal to 1, it is an ICMP protocol then extract ICMP header information by using ***ToIcmpHeader( )*** function.

8.3 Store and Display Stage:

The third stage of EWMT includes the following functions:

1- Store the brief packet information found in ***database list*** in a data store class by calling ***add ( )*** function. A Statistical stage makes use of this stored information.

2- Store all the captured packets IP header portion, and data portion as a binary format in ***packetstore list*** in a data store class by calling ***addPacket ( )*** function. These stored packets are used when details of packets are displayed in hexa format.

3- Get page of type ALL from a detail panel by calling ***get page ( )*** function. Display a brief packet information that is not restricted to a single protocol type by adding a brief packet information to the view list part of ALL pages as a table by calling ***AddPacketInfo ( )*** function.

4- If the protocol is TCP then get TCP ports by calling ***GetTcpPorts ( )*** function. Consider a port number as integer and returns its equivalent name as a string from the hashtable of TCP ports in the constant class. Get the TCP page from the detail panel by calling ***get page ( )*** function then add the extracted TCP header information and TCP ports name to the view list part of TCP page to be displayed as a table by calling ***AddPacketInfo ( )*** function.

5- If protocol is UDP then get UDP ports by calling ***GetUdpPorts( )*** function. Consider a port number as integer from UDP header and return its equivalent name as string from hashtable of UDP ports in the constant class. Get the UDP page from the detail panel by calling ***get page( )*** function then add the extracted UDP header information and UDP ports name to the view list part of UDP page to be displayed as a table by calling ***AddPacketInfo ( )*** function.

6- If protocol is ICMP then get ICMP ports by calling **GetICMPorts** ( ) function that takes port number as integer from ICMP header and return its equivalent name as string from hashtable of ICMP ports in the constant class. Get UDP page from the detail panel by calling ***get page ( )*** function then add the extracted ICMP header information and ICMP port's name to the view list part of ICMP page to be displayed as a table by calling ***AddPacketInfo ( )*** function.

7- Display details of each captured packet in packet data viewer part of each page in hexa format by Calling ***GetBinaryPacket ( )*** function which retrieves captured packets that are stored in packetstore list. It takes an index of each captured packet in packet store list and returns data as array of bytes. Then call ***PrintHexBytes ( )*** function to display captured packet in Hexa format.

8.4 Statistical Viewer Stage:

This stage demonstrates how statisticals are calculated. This is achieved by using a brief packet information for each captured packet which is stored in a database list and calling some functions and methods of data store class that perform particular calculations out of this information. This stage includes the following:

1- Create the main IPchart of type PIE graph by using ***Add Chart ( )*** function.

2- For each element in a brief packet information that is stored in database list, get list of captured IPs by using ***List Of IP ( )*** function: ***List <String^>^ List Of IP ( ).***

3- For each element in a brief packet information, calculate counts of all available ports using ***CountOfPortsPacket ( )*** method. Get a list of available ports and return their counts for each port to be stored in ports Packets Counts hashtable (.Net class). The hashtable class represents a collection of key/value pairs that are organized based on the hash code of the key. The key represents port's number and value represents its count: ***CountOfportsPacket().***
Hashtable^ portsPacketsCounts.

4- For each pair of key/value in ***portsPacketsCounts*** hashtable:

- get the name of the port as a string by calling **GetPorts ( )**function ,which uses the port number as a parameter and returns its name as a string.**String^ Get Ports (int port)**
- Update IP chart by calling **Update IP Chart()**function which creates a list of chart items. Each chart item has a port count and its name where these chart items are added to the IP chart.

5- For each element of a brief packet information that is stored in the database list, the number of packets of TCP protocol is calculated by using ***Count Of Packet ( )*** method, which returns the number of packets of TCP protocol. Also UDP and ICMP protocols are calculated.

- If there is a count for TCP; then create TCP chart of a type single bar by using ***Add Chart ( )*** function. Update the chart by calling the main function ***Update Chart ( )*** which considers port or protocol number and list of captured IPs. Each IP in the list calls ***Count Of IP ( )*** function which returns count of each specific IP and TCP protocol.

- If there is a count for UDP, then create UDP chart of type single bar by using ***Add Chart ( )*** function. Update the chart by calling the main function ***Update Chart( )*** which inputs the port or protocol number and list of captured IPs. Each IP in the list calls ***Count Of IP()*** function which returns a count of each specific IP and UDP protocol.
- If there is a count for ICMP, then create an ICMP chart of a type single by using ***Add Chart ( )*** function. Update chart by calling the main function ***UpdateChart()*** which inputs the port or protocol number and list of captured IPs. Each IP in the list calls ***CountOfIP( )*** which returns count of each specific IP and ICMP protocol.

6- If statistical icon is selected, then clear all charts and all steps above are repeated.

9. EWMT Interface Components

EWMT is so easy to be used through its friendly user interface that consists of two parts as shown in figure 4.

9.1 ToolBar

This includes three buttons which are self explanatory:

- Run.
- Stop.
- Restart Monitoring.

9.2 Main Window

It consists of three parts:

A. **Main Project Component Panel** has been built with three option Icons:

- Network Information.

It is used to view all the detailed information for each network interface card (NIC) that exists in the EWMT host machine. In addition it gives information about the network availability, updates current interface information whenever selection changes, determines if the network is available at startup and its operational status.

- **Packet Captured.**

Details of the captured packet will appear in four types of tap pages; each page has a list view and details viewer to view packet in Hexa format as shown in Figure 5.

- **Statisticals**

This Icon is selected from the Main Project Component Panel to display all the possible collected statistical information concerning protocols for all the captured packets in the network. This is implemented dynamically in a PI graph in the main chart as shown in figure 6. In addition the relations between each IP and every protocol are displayed in a dynamic single bar chart as shown in figures 7. It is possible to display more protocols depending on the type of the captured packets.

In both PI graph chart and single bar chart, by right clicking on the image in each chart, the following features will be available:

- Image can be copied to clipboard.
- Image can be saved.

- Image page can be setup to appropriate size.
- Show point values.
- Un-Zoom.
- Set scale to default.

Figures 8 and 9 show these capabilities.

B. **Details Panel** is responsible for hosting system pages, and updating itself to select the appropriate page to be displayed according to the user selected option Icon.

C. **Output Panel** is used to display information about the user selected option Icon.

10. Proposed EWMT Versus Available Monitoring Tools:

Network Stumbler is an open source software tool that is used for monitoring and assessing security threats for wireless networks. It can actively detect wireless networks by periodically sending probe requests. Network Stumbler works best with network interface cards that use the hermes chipset; this basically refers to ORiNOCO Gold or Silver "Classic" cards or any "re–badged" version [3].

Kismet is another wireless network detector that works passively. This means that without sending any loggable packets, Kismet works with wireless cards which support raw monitoring mode, and can sniff 802.11a, 802.11b and 802.11g traffic [13].

WANMon [6] is oriented towards monitoring wireless Ad-hoc type of networks.

However, EWMT does not demand a specific type of wireless NICs. EWMT is based on monitoring the traffic between wireless stations and Internet using a standard, non-wireless monitor on a mirror port. This would make it possible to capture the packets being sent and received through the access point. In addition EWMT is used to monitor packets in both wire and wireless networks while the above mentioned tools are used only for wireless network. Using a switch with mirror port makes it possible to monitor switched network as well as hub network.

11. Conclusions

Promiscuous monitoring of wireless networks has often been a source of confusion. It appears logical that if any Ethernet adapter can be used for promiscuous mode monitoring in a wired Ethernet network, then any Wireless Ethernet adapter is equally good for doing the same in a 802.11 a, b, or g network. Theoretically, this is true, but in practice, this is very far from reality because not all wireless NICs support promiscuous mode. Therefore the dilemma is to monitor the traffic between wireless stations and Internet. The proposed system presents a specific layout by using a standard, non-wireless monitor on a mirror port, which would make it possible to capture the packets being sent and received through the access point.

EWMT runs on Windows 32-bit based platforms including Windows XP operating system and later releases. It is driver-independent and supports every Network adapter that can be put into promiscuous mode. It is basically based on a socket class of

3.5 .NET framework which consolidates advanced features.
EWMT can be used by Network Administrators to troubleshoot network problems and for educational purposes to trace network traffic patterns.

12. References

[1] Jeffrey Wheat, Randy Hiser, Jackie Tucker, Alicia Neely, and Andy McCullough, ***Designing A Wireless Network***, Syngress Publishing, Inc., 2001.

[2] Behrouze A. Forouzan, **"*Data Communictions and Networking*"**, 4th edition, McGraw-Hill, Inc., 2007.

[3] Clincy,Victor, Sitaram, and Ajay Krithi, **"*Evaluation And Illustration of A Free Software (FS) Tool For Wireless Network Monitoring And Security*"**, CCSC Rocky Mountain Conference, JCSC 21, pp. 19-29, 2006.

[4] Dionysios Skordoulis, Qiang Ni, Hsiao-Hwa Chen, Adrian P. Stephens, Changwen Liu, and Abbas Jamalipour, **"*IEEE 802.11N MAC Frame Aggregation Mechanisms for Next Generation High-Throughput WLANS*"**, IEEE Wireless Communications, Volume 15 , Issue 1, pp. 40-47, 2008.

[5] Don Ngo, Naveed Hussain, Mhbub Hassan, and Jim Wu, **"*WANMon :A Ressource Usage Monitoring Tool for Ad Hoc Wireless Networks*"**, Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks (LCN'03) , 2003.

[6] Brajesh Pande, Deepak Gupta, Dheeraj Sanghi, and Sanjay Kumar Jain, **"*The Network Monitoring Tool – PickPacket*",** Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05), pp.191 - 196, 2005.

[7] A. Karygiannis, E. Antonakakis, A. Apostolopoulos, **"*Host-Based Network Monitoring Tools for MANETs*"**, Proceedings of the 3rd ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor and Ubiquitous Networks, Terromolinos, Spain, pp. 153 - 157, 2006.

[8] Arun Chhetri and Rong Zheng, "***WiserAnalyzer: A Passive Monitoring Framework for WLANs***", Fifth International Conference on Mobile Ad-hoc and Sensor Networks, pp. 495-502, 2009.

[9] Hongyan Lei, and Arne A. Nilsson **," *A Power Management Scheme for the IEEE 802.11 Based WLANs*** ", Performance, Computing, and Communication Conference, IPCCC 2005, pp.123-127, 2005.

[10] Woo-Yong Choi, **"*Clustering Algorithm for Hidden Node Problem In Infrastructure Mode IEEE 802.11 Wireless LANs*",** 10th International Conference on Advanced Communication Technology , ICACT 2008, pp.1335-1338, 2008.

[11] Xiaohong Yuan, Percy Vega, Jinsheng Xu, Huiming Yu, and Yaohang Li, **"*Using Packet Sniffiner Simulator in the class: Experience and Evaluatiion*"**, ACM Southeast Regional Conference: Winston-Salem, NC, USA, ACM, pp.116-121, 2007.

[12] Angela Orebaugh et al, ***Ethereal packet sniffing***, Syngress Publishing, Inc, 2004.

[13] "Kismet's home page and documentation," http://www.kismctwireless.net

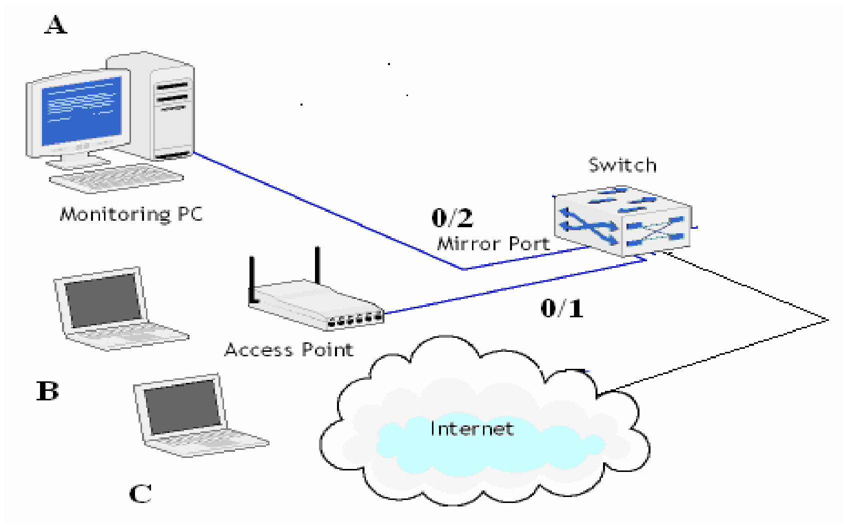


Figure (1) Proposed Network Layout Architecture

.

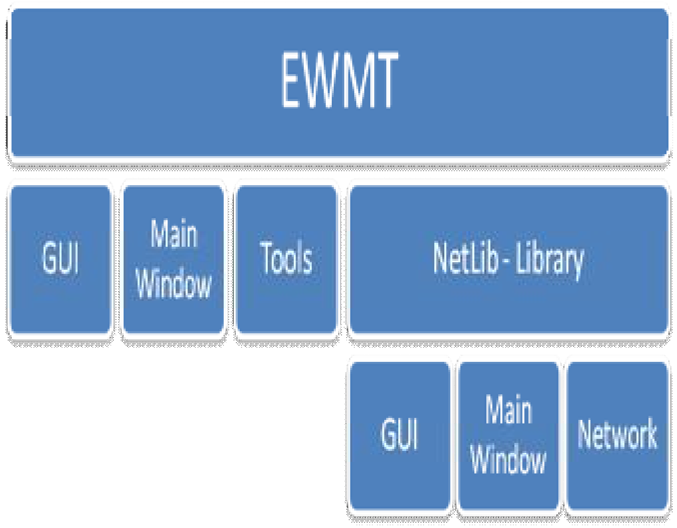


Figure (2) EWMT Hierarchy

.

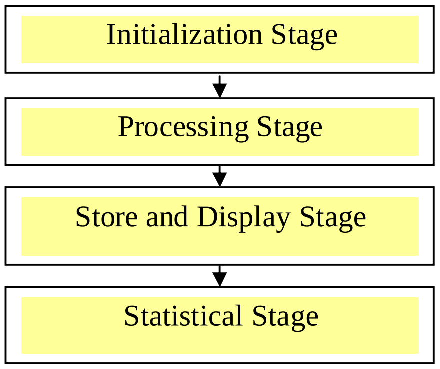
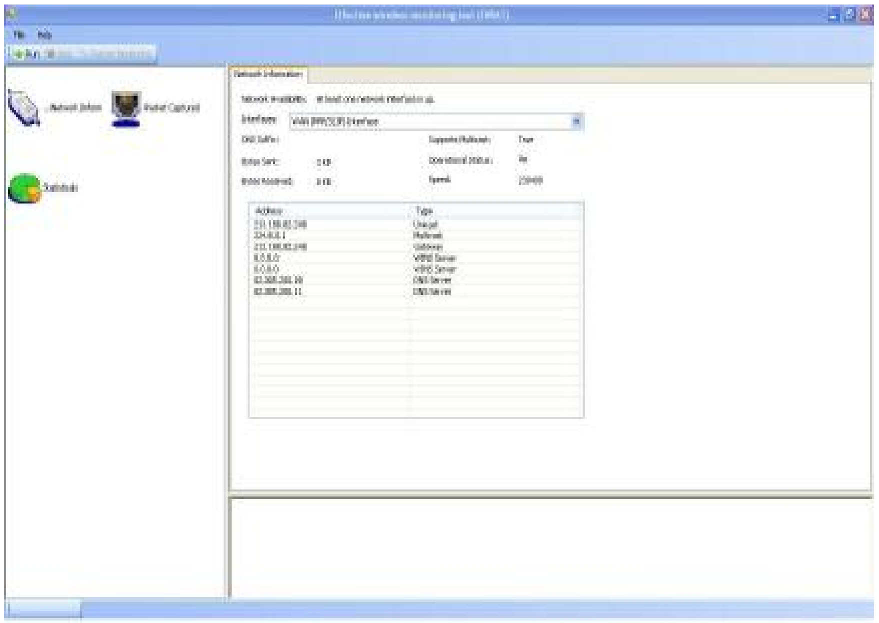


Figure (3) EWMT Stages



Network Availability: At least one network interface is up.

Interfaces: WAN (PPP/SLIP) Interface

DNS Suffix :

Bytes Sent: 1 KB

Bytes Received: 1 KB

Supports Multicast: True

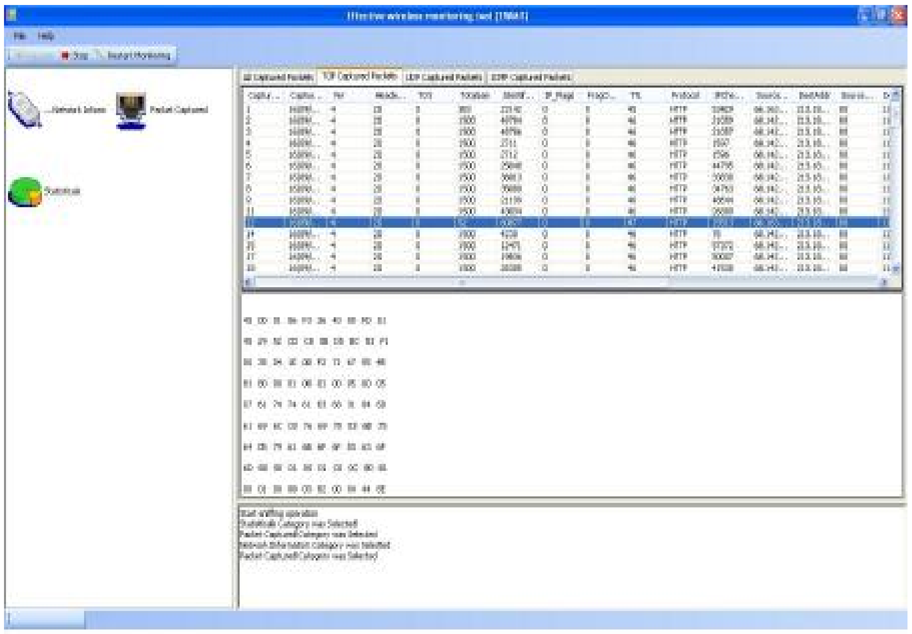
Operational Status: Up

Speed: [illegible]

Address	Type
111.188.82.248	Unicast
224.0.0.1	Multicast
111.188.82.248	Gateway
0.0.0.0	WINS Server
0.0.0.0	WINS Server
82.205.200.10	DNS Server
82.205.200.11	DNS Server

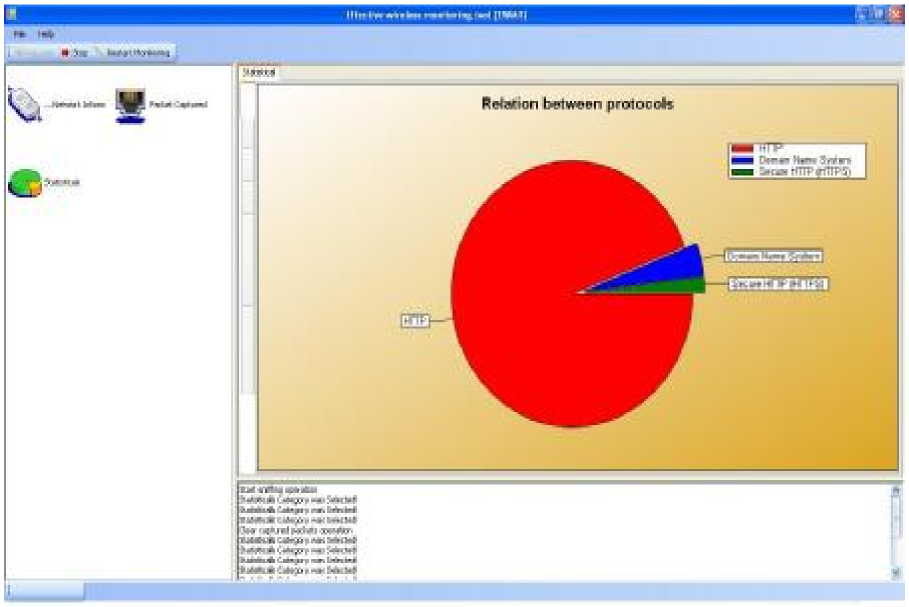
Figure (4) EWMT Interface Components

.



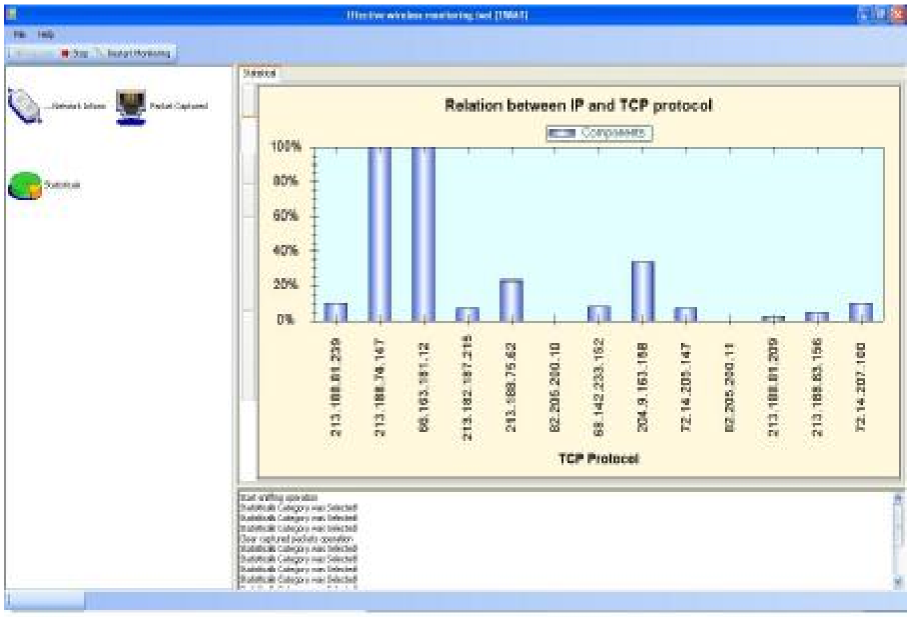
Start sniffing operation
Statistical Category was Selected
Packet Captured Category was Selected
Network Information Category was Selected
Packet Captured Category was Selected

Figure (5) Packet Selection



```
Start sniffing operation
Statisticals Category was Selected
Statisticals Category was Selected
Statisticals Category was Selected
Clear captured packets operation
Statisticals Category was Selected
Statisticals Category was Selected
Statisticals Category was Selected
Statisticals Category was Selected
```

Figure (6) Relation between Protocols in PI Graph Chart



```
Start sniffing operation
Statisticals Category was Selected
Statisticals Category was Selected
Statisticals Category was Selected
Clear captured packets operation
Statisticals Category was Selected
Statisticals Category was Selected
Statisticals Category was Selected
Statisticals Category was Selected
```

Figure (7) Relation between IP and TCP Protocol in Single Bar Chart

.

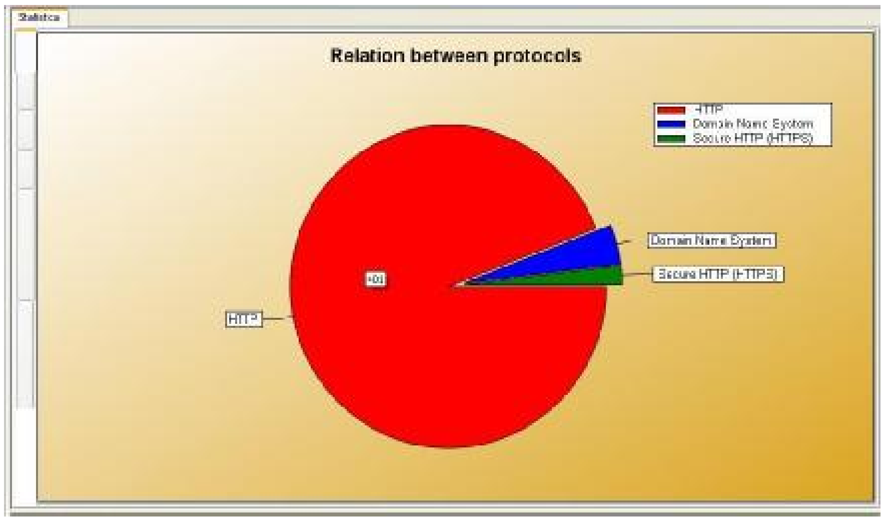
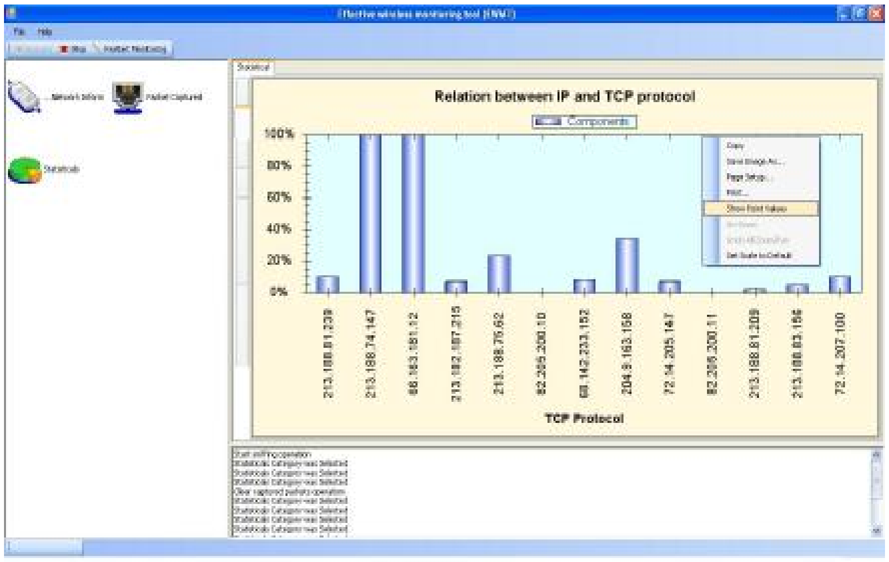


Figure (8) Value of HTTP Part



```
Start sniffing operation
Statistical Category was Selected
Statistical Category was Selected
Statistical Category was Selected
Clear captured packets operation
Statistical Category was Selected
Statistical Category was Selected
Statistical Category was Selected
Statistical Category was Selected
```

Figure (9) Capabilities in Single Bar Chart