# Table of Contents

1. Introduction ............................................................................................................ 1

2. Related Work ........................................................................................................ 4  
   2.1 MRTG ............................................................................................................... 4
   2.2 Packet Capturing Tools ................................................................................. 6  
       2.2.1 Etherfind .................................................................................................. 6
       2.2.2 NFSwatch ................................................................................................. 7
       2.2.3 TCPdump ................................................................................................. 8
   2.3 Argus ............................................................................................................... 9
   2.4 Web Technology ............................................................................................. 11

3. System Requirements .......................................................................................... 13  
   3.1 Platform Independence .................................................................................. 13
   3.2 Web-based User Interfaces .............................................................................. 13
   3.3 Guaranteed Packet Capturing .......................................................................... 14
   3.4 Classification of Protocol Information ............................................................ 14
   3.5 Mobility ........................................................................................................... 14
   3.6 Security ........................................................................................................... 15
   3.7 Viewing of Real-Time and Historical Data ...................................................... 15

4. Design of WebTrafMon ....................................................................................... 16  
   4.1 Probe ............................................................................................................. 17
       4.1.1 MAC Layer ............................................................................................... 17
       4.1.2 Network Layer ......................................................................................... 18
List of Figures

Figure 1. Traffic Load Statistics of a Network Segment Using MRTG........ 5
Figure 2. Principle Architecture of a Common Gateway Interface.......... 12
Figure 3. Design Architecture of a Web-based Traffic Monitoring & Analysis System............................................................... 16
Figure 4. Medium Access Layer Packet......................................... 18
Figure 5. WebTrafMon Homepage.................................................. 25
Figure 6. Source Host Traffic Analysis View.................................... 26
Figure 7. Destination Host Traffic Analysis View................................. 27
Figure 8. Source to Destination Traffic Analysis View.......................... 27
Figure 9. Network Layer Traffic Analysis View................................ 28
Figure 10. Transport Layer Traffic Analysis View................................. 29
Figure 11. Application Layer Traffic Analysis View.............................. 30
Figure 12. Real-Time Traffic Monitoring Using WebTrafMon............... 30
List of Tables

Table 1. Usage Example of Etherfind .................................................. 6
Table 2. Sample Output of TCPdump .................................................... 8
Table 3. Features of Traditional Network Monitoring Tools .................. 10
Table 4. Sample Log File .................................................................. 23
Table 5. Accuracy of Packet Sampling (5-minute test) ....................... 32
1. Introduction

Today, more than ever, enterprise computing relies increasingly on computer networks. Network environments are growing, fueled by extensive enhancements of computer hardware and software as well as the rapid growth of the Internet and WWW. More systems are being connected to networks, rapidly increasing traffic. Such growth impacts on the performance of many network-related user applications. These conditions give birth to various problems and solutions for network traffic monitoring and analysis.

Why monitor and analyze network traffic? With more systems are connected to the network especially Internet and more network-related applications are being developed, network itself is treated as an indispensable resource that should be managed like local disk or memory. So, managing various network resources and analyzing network traffic usage became a heavy burden to the information technology managers.

Network traffic monitoring provides collected data and analysis of this data. Raw status data is gained by probing network packets, and analysis provides extended data based on the raw data. To use limited network resources effectively, we need to obtain accurate and reliable information from our networks, such as:

- how much traffic is transferred,
- what type of traffic is transferred,
- how much traffic is generated from which system,
- which system or application is causing bottlenecks and
- how high is peak traffic and when does it peak.

If network managers cannot provide reliable answers to these kinds of questions, valuable network resources will be wasted.
A number of automated tools have been developed in order to help network managers monitor and analyze network traffic. Multi-Router Traffic Grapher (MRTG) [7], Argus [1], Etherfind [12], NFSwatch [9] and TCPdump [13] are examples of such tools.

For example, MRTG provides hourly, daily, weekly, monthly and yearly statistics of network traffic loads using Simple Network Management Protocol (SNMP) [14] through the use of Web-based graphical user interfaces. But unfortunately, MRTG does not provide details concerning protocols used or system traffic origin. Although MRTG is an excellent tool with easy-to-use Web-based user interfaces, network traffic managers need more sophisticated and user-friendly tools to analyze detailed traffic information.

This thesis presents the design and implementation of a new Web-based network traffic monitoring and analysis system called WebTrafMon. The proposed system shows network traffic in detail, allowing users to see statistics at each communications layer, such as source and destination, total number of packets transferred, and other data. WebTrafMon employs Web interfaces so users can access traffic information via generic, easy-to-use, ubiquitous Web browsers. Web interfaces are system and location-independent and the most generic way of showing what we have done, and what WebTrafMon can do. Users only need to be connected to the Internet and have a Web browser. No special training is required. No additional program installation or configuration is needed on the client side. Anyone can access the data and analysis, although we can restrict access by using a common Web-based user authentication.

WebTrafMon extracts information using network packets from headers at the lower MAC level, to that of the upper application level. No additional network information exchange or overhead is needed to obtain the information. The information is extracted in real time, so we can see the network’s activities, and which system or protocol is becoming dominant - from the very first time we
use WebTrafMon.

Our tool differs from similar tools because it can show traffic information according to source and destination host through any Web interface; and it can show the traffic status according to each protocol layer, from the network layer to the application layer. Often when networks are heavily loaded, traffic originates from a number of specific hosts. Network managers of enterprise network environments should be able to ascertain quickly which host is making a bandwidth peak. Once identified, the responsible host causing the bottleneck, can be contacted via the system administrator who then corrects the situation.

Traditional network traffic analysis tools are unable to show traffic per host. Traffic information per host, however, nowadays is important data. For example, only one unauthorized VOD(Video On Demand) or AOD(Audio On Demand) server can cause an entire subnet to peak. More and more multimedia services are being offered on the Internet, some consuming a great deal of bandwidth. Thus information about protocols from each level is vital, but also, host traffic data becomes more and more critical. WebTrafMon can show both types of information to the users.

Our system, WebTrafMon can be divided into two parts: a probe and a viewer. The probe extracts data from network packets and composes log files. Analysis results are based on these log files. The viewer interacts with users, showing the analysis results to the user through Web browsers.

The remainder of this thesis is organized as follows. Section 2 examines other existing network traffic monitoring solutions. Section 3 examines the requirements for traffic monitoring and analysis systems. Section 4 presents the design architecture of WebTrafMon. Section 5 describes implementation details of WebTrafMon. Section 6 examines the use of this system for analyzing enterprise networks. Finally, Section 7 summarizes the work and examines future research.
2. Related Work

Network monitoring and analysis have become important topics in networking environments, encouraging worldwide research and development. Many organizations have developed automated network monitoring and analysis systems, some of which are described below.

2.1 MRTG

The Multi-Router Traffic Grapher (MRTG) [7] is a tool for monitoring traffic loads on network-links. MRTG generates HyperText Markup Language (HTML) pages containing GIF images that provide a live, visual representation of network traffic. MRTG is implemented using Perl and C languages and it can be operated under various UNIX platforms and Windows NT. MRTG is successfully used on many sites around the world.

MRTG consists of a Perl script which uses SNMP to read the traffic counters of routers and a fast C program which logs the traffic data and creates beautiful graphs representing the traffic on the monitored network connection. These graphs are embedded into Web pages which can be viewed from any modern Web-browser.

In addition to a detailed daily view, MRTG also creates visual representations of the traffic seen during the last seven days, the last four weeks and the last twelve months. This is possible because MRTG keeps a log of all the data it has pulled from the router. This log is automatically consolidated, so that it does not grow over time, but still contains all the relevant data for all the traffic seen over the last two years. This is all performed in an efficient manner. Therefore it can monitor 50 or more network links from any halfway decent UNIX box.

MRTG is not just limited to monitoring network traffic. It can be used to
monitor any Simple Network Monitoring Protocol (SNMP) MIB [20] variable. MRTG can even supply analysis results using data gathered from an external program. People are using MRTG to monitor such information as system load, login sessions, modem availability and more. MRTG even accommodates two or more data sources into a single graph. Figure 1 shows a screen shot of network load monitoring using MRTG.

Figure 1. Traffic Load Statistics of a Network Segment Using MRTG
Despite all of these attractive capabilities, MRTG cannot provide information that shows which host or application may be causing a traffic bottleneck. SNMP MIB variables are not appropriate for such use and, for traffic, it can only show traffic load. MRTG does not provide information about traffic type or protocol statistics.

2.2 Packet Capturing Tools

Various packet capturing tools have been developed to date. Here are some examples of these tools that are widely used by network administrators to analyze network statistics.

2.2.1 Etherfind

The SunOS operating system provides Etherfind [12], which is a software packet monitor. The software opens the network card in promiscuous mode and writes a summary line of each packet to a file. Information includes protocol type, size, and sending and receiving addresses.

<table>
<thead>
<tr>
<th>Table 1. Usage Example of Etherfind</th>
</tr>
</thead>
<tbody>
<tr>
<td># etherfind -p -i le0 -src nyssa -o -dst nyssa</td>
</tr>
<tr>
<td>icmp type</td>
</tr>
<tr>
<td>lhth proto source destination src port dst port</td>
</tr>
<tr>
<td>60 tcp leela.acs.ohio nyssa login 1021</td>
</tr>
<tr>
<td>118 udp tardis nyssa 652 684</td>
</tr>
<tr>
<td>60 tcp leela.acs.ohio nyssa login 1021</td>
</tr>
</tbody>
</table>
This tool extracts information from each packet. Data is supplied as a text-based user interface, and only users with root permission can access the tool.

For example, to examine all traffic originating or terminating at the host called "nyssa", see the usage shown in table 1.

2.2.2 NFSwatch

NFSwatch [8, 9] monitors all incoming network traffic destined to NFS file servers, and divides it into several categories. The number and percentage of packets received in each category is displayed on the screen, which is continuously updated.

By default, NFSwatch monitors all packets destined for the current host. An alternate destination host to watch for may be specified using the -dst argument. If a source host is specified with the -src argument, then only packets arriving at the destination host which were sent by the source host are monitored. Traffic between a specific server and its clients may be watched by specifying the name of the server with the -server argument. If the -all argument is given, then all NFS traffic on the network is monitored.

To monitor NFS traffic to files and file systems, NFSwatch must extract information from the NFS file handle. The file handle is a server-specific item, and its contents vary from vendor to vendor and operating system to operating system. Unfortunately, instead of using libpcap [15], a portable network library, NFSwatch uses a server-dependent way to extract information from a file handle.

This tool, like Etherfind, is inappropriate for our needs too because it was originally designed to monitor a single host moreover, does not provide Web interface.
2.2.3 TCPdump

Originally written by Van Jacobson, TCPdump [13] was developed to research and improve TCP and Internet gateway performance. Recent versions make use of the system-independent packet capture library (libpcap) [15].

<table>
<thead>
<tr>
<th>Time</th>
<th>Source IP</th>
<th>Port</th>
<th>Dest IP</th>
<th>Port</th>
<th>Action</th>
<th>Seq1</th>
<th>Seq2</th>
<th>Ack</th>
<th>Ack2</th>
<th>Win</th>
</tr>
</thead>
<tbody>
<tr>
<td>22:27:52</td>
<td>ohhara.postech.ac.kr.6255</td>
<td>&gt; crystal.hiper.co.kr.7871</td>
<td>S</td>
<td>2171642993:2171642993(0)</td>
<td>win 512</td>
<td>&lt;mss 1460</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22:27:52</td>
<td>ohhara.postech.ac.kr.12382</td>
<td>&gt; xgs21.postech.ac.kr.x11</td>
<td>P</td>
<td>1289:1433(144)</td>
<td>ack 1824</td>
<td>win 31744 (DF)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22:27:52</td>
<td>crystal.hiper.co.kr.7871</td>
<td>&gt; ohhara.postech.ac.kr.6255</td>
<td>R</td>
<td>0:0(0)</td>
<td>ack 2171642994</td>
<td>win 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22:27:52</td>
<td>ohhara.postech.ac.kr.6256</td>
<td>&gt; crystal.hiper.co.kr.7872</td>
<td>S</td>
<td>11194989</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Sample Output of TCPdump

TCPdump prints the headers of packets on a network interface. And the users should manually analyze network status using this header information.
Analyzing the output manually to monitor overall, log-term network status is very hard work. Although TCPdump has many options for capturing raw data, it does not provide any analysis capability of the captured data.

Table 2 shows a sample output of TCPdump. As you can see, the format is very complex.

2.3 Argus

Argus [1] is a generic IP network transaction auditing tool that has been used by many sites worldwide, performing many powerful network management tasks. It runs as an application level daemon, promiscuously reading network packets from a specified interface. It then generates network traffic audit records for the network activity that it encounters. The manner in which Argus categorizes and reports on network activity makes this tool unique and powerful.

Argus enables a site to generate comprehensive network transaction audit logs, in a fashion that provides for high degrees of data reduction, and high degrees of semantic preservation. This has allowed us to perform extensive analysis of our network traffic, historically. By processing these historical network logs, we have been able to, among other things:

- Verify that network security access control policies are actually being enforced and detect attempts to break through firewall and host based mechanisms.
- Perform grade of service analysis for every IP based network service that is offered in our network infrastructure.
- Identify and troubleshoot difficult transient network problems such as intermittent service failure, denial of service attacks and host and network configuration problems.
The data that Argus generates makes possible the ability to analyze network activity and performance in ways that have not been possible before. In short, it extracts information from each packet in promiscuous mode and saves the information to a file later analyzes that file. This tool skillfully shows information about protocols, but does not show source or destination host information. Further, it only provides a text-based user interface.

We have examined the above-mentioned tools carefully, but none of them provided the capability we desire. This motivated us to develop WebTrafMon.

Table 3 summarizes the features of the tools we have discussed so far.

<table>
<thead>
<tr>
<th></th>
<th>MRTG</th>
<th>Etherfind</th>
<th>NFSwatch</th>
<th>TCPdump</th>
<th>Argus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web-based?</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Analysis capability?</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Per Host Traffic Information?</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Per Protocol Traffic Information?</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 3. Features of Traditional Network Monitoring Tools
2.4 Web Technology

The World-Wide Web (WWW or the Web) [17] was developed originally by researchers at CERN. The Web is a way of organizing and accessing on-line content in multiple media using hyperlinked relations on the Internet. The Web also attempts to integrate the major protocols and applications of the Internet into one easy-to-use interface.

The native protocol of the Web is HyperText Transfer Protocol (HTTP) [6], used on the Web since 1990. HTTP is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, object-oriented protocol that uses an extension of its request methods (commands) for name servers and distributed object management systems. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred. HyperText Markup Language (HTML) is the simple data format used to create hypertext documents that are portable from one platform to another. HTML documents use Standard Generalized Markup Language (SGML) with generic semantics that are appropriate for representing information from a wide range of domains.

The Web is based on the client/server architecture and basically operates as follows. Hypertext documents that are to be made available on the Web are prepared in HTML and made accessible by the Web server. Users wishing to retrieve the documents can do so by using a Web client (Web browsers such as Netscape Navigator or Internet Explorer) to connect to the Web server which contains the documents.

The Common Gateway Interface (CGI) [18] is the most popular method for interfacing external applications with Web servers. An HTML document that the Web daemon retrieves is static, which means that it exists in a constant state: a text file that does not change. A CGI program, on the other hand, is executed
in real-time, so that it can output dynamic information. If a person wants to hook up their Unix database to the Web so people worldwide may query it, then the person needs a CGI program. The CGI program must allow the Web daemon to transmit information to the database engine, receive the results back again, and display them to the client.

Figure 2. Principle Architecture of a Common Gateway Interface

Figure 2 illustrates the architecture of the Web with a CGI. Using CGIs, what can be hooked up to the web is practically limitless. Many currently available Web-based management solutions use CGIs as gateways to management agents.
3. System Requirements

The following are the most important requirements that should be satisfied in an enterprise network monitoring and analysis system.

3.1 Platform Independence

Low-level packet capturing routines should be platform-independent. Because each platform provides a different low-level network device, there should be an abstraction above the base network layer. If the code is written for one specific platform, porting the program to different platforms would be hard, and users of those platforms will not be able to use it. For Unix, BSD networking code is the common basis of each proprietary operating system. Thus, packet-capturing routines can be based on a common code. Porting this program to a PC Windows platform would require additional work because on low level network devices the Unix platform and Windows platform have almost nothing in common.

3.2 Web-based User Interfaces

User interface should be easily understood and manipulated. For this reason a Web based interface can be the best solution. Web-based user interfaces are easy to use and ubiquitous because the Web is not dependent on a specific operating system. As well, Web browsers are available for most operating systems. Web supports multimedia data. This feature enables developers to make a beautiful user interface easily.

A Web-based user interface has one more specific strong point. To use a
Web-based system, a person only needs access to the network and use of a Web browser. Anywhere, anytime, anyone can access the system using a universal Web browser.

3.3 Guaranteed Packet Capturing

On a fast network, the number of packets transmitted can be astronomical. Analyzing such data requires a great deal of processing time. Thus the efficiency of a packet capturing code is essential.

On a fast network, the system may not be able to handle all the packets in time. If the processing speed, or the system itself, acts at speed insufficient to capture all packets, the analysis result is unreliable. As a matter of fact, ensuring that all packets are properly captured is extremely challenging [11].

3.4 Classification of Protocol Information

Many communications protocols are currently used on a typical enterprise network. For example, numerous application protocols exists, such as HTTP, FTP, Telnet, SNMP, MP3, RealAudio, RealVideo, etc. All packets are delivered using some specific protocol and the protocol can be classified into a certain layer. Ideally, a monitoring tool should be able to classify and display all possible protocols in each layer.

3.5 Mobility

The packet capturing tool should be easy to install and use on any network
segment. If someone wants to monitor a specific network segment, he should be able to install the monitoring system on a notebook or desktop and connect it to that segment easily.

3.6 Security

Security is vital. Securing internal data is necessary to prevent illegal access and potential damage to valuable data. Sometimes, pranksters or hackers crack systems that neglect security measures. We must then restrict users to only those who are authorized. Accordingly, a Web security mechanism (such as username and password checking) can be used to provide such access to the monitoring and analysis system.

3.7 Viewing of Real-Time and Historical Data

The system should be able to show the online real-time status data and accumulated historical status data easily. From the historical data, the user can analyze long-term traffic trends and from the real-time data, short term traffic trends. This will help the user to detect problems of the network easier and faster.
4. Design of WebTrafMon

Based on the requirements discussed in the previous section, we have designed a Web-based enterprise network monitoring and analysis system. The design architecture of our system is illustrated in Figure 3.

![Figure 3. Design Architecture of a Web-based Traffic Monitoring & Analysis System](image)

The overall system consists of two parts: a probe and a viewer. The probe extracts data from network packets in each layer and saves it to a pre-configured log file. The log file is then processed by the analyzer component of the viewer. The user interacts with the output analysis data stored in the data repository, which the Web server sends to the user’s Web browser through the view controller.
4.1 Probe

The probe is used to retrieve data from each captured packet and store the data in a log file for further processing. To capture all network packets, it acts in promiscuous mode. This is due to the property of the Ethernet itself: packet broadcasting. Capturing all packets is the most important and basic operation of the probe. Therefore it should be implemented using an operating system independent code. A detailed description of the structure of the probe for each data communications layer follows.

But first, here is the role of each component of the probe. The packet capturer captures packets from the network. The data filter extracts the information of each packet. The data logger writes the information that the data filter extracted to the log file.

4.1.1 MAC Layer

Medium Access Control (MAC) layer packets should be captured at full speed. However, it is practically impossible to capture all packets reliably because this is not a hardware solution but a software solution, of which the packet capturing routine should be concise and efficient. Figure 4 shows the packet structure of the MAC level [10].

At this layer, the abstraction of some specific hardware drivers (e.g., Ethernet card) is important. If we use specific network device drivers directly on our packet capturing code, porting this program to another network device may require additional work.
We extract all traffic-related information from this layer and the size of each packet from its header.

### 4.1.2 Network Layer

This layer is related to IP (Internet Protocol), ARP (Address Resolution Protocol) [10], RARP (Reverse ARP) [10], and so on. If a packet is IP-based, we process it further to analyze the source and destination host information. If not, we need not analyze it further because it does not contain the source or destination host information. We must then inform the viewer that the packet is not IP-based so it knows that the destination or host information is not available.

As for determining which host sent a packet, information from this layer is used. A standard IP packet has two IP addresses in it: source and destination. We use this valuable information to analyze per-host traffic [10].

### 4.1.3 Transport Layer

This layer is the host-to-host transport layer. The two most prevalent protocols in the transport layer are TCP (Transmission Control Protocol) [3] and UDP (User Datagram Protocol) [3]. TCP provides reliable data delivery service with end-to-end error detection and correction. UDP provides low-overhead, connectionless, best-effort datagram delivery service.
The probe extracts the protocol that was used at this level for any given packet, and saves the information to the log file for further use by the viewer.

4.1.4 Application Layer

The application layer is directly related to the source and destination ports, providing information about the application protocols and destination port contained in the packet [4]. For example, port number 23 is used by Telnet. This port number is reserved for Telnet use only. The RFC 1700 [2] contains the list of port number assignments from the Internet Assigned Numbers Authority (IANA). IANA classifies the port numbers into three categories [22].

- Well-known ports: 0 through 1023. These port numbers are controlled and assigned by the IANA. When possible, the same port is assigned to a given service for both TCP and UDP. For example, port 80 is assigned for a Web server, for both protocols, even though all implementations currently use only TCP.

- Registered ports: 1024 through 49151. These are not controlled by the IANA, but the IANA registers and lists the uses of these ports as a service to the Internet community. When possible, the same port is assigned to a given service for both TCP and UDP. For example, ports 6000 through 6063 are assigned for an X Window server, for both protocols, even though all implementations currently use only TCP.

- Dynamic or private ports: 49151 through 65535. The IANA says nothing about these ports. These are what we call ephemeral ports.

A large number of application layer protocols exist. For example, HyperText Transfer Protocol (HTTP) related traffic has increased due to the popularity of the World Wide Web, WWW. HTTP is a Multipurpose Internet Mail Exchange
(MIME) [4] based protocol. Any data that can be defined using MIME can be transferred through HTTP [5, 6]. This gave birth to multimedia Web services, which typically require an excess of bandwidth. Such services have attracted a lot of users to the Internet and hence, increased network traffic (both Internet and Intranet) has substantially in recent years. New Internet applications (and thus net application protocols) are being introduced to the Internet rapidly these days. A network monitoring system should be able to detect such new application protocols and analyze them.

4.2 Viewer

The WebTrafMon viewer consists of three components: data reader, analyzer and view controller. The data reader reads packet information from the log file that the probe has generated. The analyzer analyzes information that the view controller requested. Finally, the view controller interacts with the user to provide the information the user requests.

The viewer analyzes the log file generated by the probe and shows all possible information. This includes information on the protocols being used on the network, but it mainly concerns the network bandwidth use of:

- each network layer protocol
- each transport layer protocol
- each application layer protocol
- traffic between each source and destination
- traffic from each source host
- traffic to each destination host
For ease and efficiency, we have used a Web-based interface, although any interface can be used. A Web-based interface eliminates problems of porting, while a single script provides uniform results, regardless of operating system, wherever the user is located.

The viewer reads the log file generated by the probe and interacts with the user to determine what the user wants. Depending on the user interaction, the viewer processes the packet data from the log file and shows the result to the user through a Web browser.

In designing the viewer, we have considered all possible security problems. This demands that we authenticate users so only those with permission can access and use the system. Password checking is the most general way of authentication. If password checking is too frequent though, the user may regard it as a nuisance. Therefore, checking passwords at the initial login stage is enough.
5. Implementation

We have used Web technology to implement our design. In this section, we present implementation details of WebTrafMon.

5.1 Probe Implementation

We have implemented the packet-capturing component using libpcap [15], a system-independent interface for user-level network packet capturing developed by the Network Research Group at the Lawrence Berkeley Laboratory. The general libpcap API makes porting, and supporting of multi-vendor systems easy. The libpcap interface supports a filtering mechanism based on the architecture of the BSD packet filter (BPF) [15]. Although most packet capture interfaces require in-kernel filtering, libpcap uses in-kernel filtering only for the BPF interface. Systems lacking a BPF, use all packets that are read into user space. Then the BPF filters are evaluated in the libpcap library. BPF is standard in 4.4BSD, BSD/386, NetBSD, and FreeBSD. DEC OSF/1 uses the packet filter interface but has been extended to accept BPF filters (which libpcap utilizes).

To analyze packet information, we have defined a log format according to the TCP/IP protocol layer and the size of each packet. Table 4 is a sample part of a log file that we store. The significance of each row and column is as follows. Each row represents a captured packet, with the first field showing the size, in bytes, including header of each packet. This information is extracted from the MAC layer. As for Ethernet packets, this information appears under the Ethernet header.

The second field is the source host of the packet, if it is an IP (Internet Protocol) based packet. If it is not based on IP, it may be based on ARP
(Address Resolution Protocol) [10], RARP (Reverse ARP) [10], or other protocols in this layer. For example, lines 3 and 4 from the sample are such cases. This information is extracted from the network layer.

The third field is the destination host of the packet if it is IP-based. The source and destination host information goes hand-in-hand. This information is very important in order to learn from which host the traffic originated.

The fourth field is the transport layer protocol information. If it is TCP-based, the flag "tcp" is printed; and if it is UDP-based, the flag "udp" is printed. If it is not based on TCP or UDP, the appropriate protocol name is printed. The last line is such a case.

The last field of each TCP or UDP-based packet is the application layer information. This is determined using the port number information of each TCP or UDP-based packet.

<table>
<thead>
<tr>
<th></th>
<th>IP Address</th>
<th>IP Address</th>
<th>Protocol</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>346</td>
<td>164.124.96.18</td>
<td>141.223.82.4</td>
<td>udp</td>
<td>telnet</td>
</tr>
<tr>
<td>64</td>
<td>141.223.82.4</td>
<td>141.223.82.26</td>
<td>tcp</td>
<td>http</td>
</tr>
<tr>
<td>112</td>
<td>Rarp</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>Arp</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>141.223.99.99</td>
<td>141.223.82.28</td>
<td>icmp</td>
<td></td>
</tr>
</tbody>
</table>

Table 4. Sample Log File

5.2 Viewer Implementation

The viewer analyzes the log files generated by the probe. We used Perl script
to analyze the log file, and CGI to show the final output on the user’s browser. For each column, the Perl script retrieves information on each network layer and shows it graphically. All scripts gather source-host information, destination-host or protocol-related information, and packet size information.

Showing such data is fairly simple. All that is needed is to add HTML code. We can now show the following traffic information for a given log file:

- source host
- destination host
- source to destination pair
- network layer protocol
- transport layer protocol
- application layer protocol

The viewer extracts one or two of the fields that is actually needed from the log file, and sorts it according to the total traffic. The traffic and bandwidth related information originates from the first field of the log file, the size of a given packet.

For security, we added a password-checking interface at the initial stage to authenticate the user. This can be done through the viewer component itself or by modifying the Web server side. We chose modifying the configuration of the Web server. The Apache Web server [16] that we use provides a simple and easy password authentication mechanism.
6. Experience

We use WebTrafMon on a demand basis. As long as they have security access, anyone who feels that the network has peaked or has some problems, can connect to our homepage to interact with this tool. The analysis can be done in real time or in traffic history. Figure 5 shows the main window of WebTrafMon.

![WebTrafMon Homepage](image)

Figure 5. WebTrafMon Homepage

The left window shows what information WebTrafMon can display for the user. “Data Received” displays the destination host information. “Data Sent” displays the source host information. “Data Exchanged” displays the traffic information according to both the source and destination. “Protocol Information” displays the protocol information classified according to each network layer. And “Real-Time Monitoring” displays monitor the network status in real-time, online.
Using a Web browser, users may select a monitoring time interval or review the existing traffic history. During that time period, WebTrafMon collects the network packets from a network segment and produces Web-based output results of current network status.

Figure 6 shows sample “Data Sent” traffic history information. It can show both the most recent traffic status and accumulated daily traffic status. Each frame window can be resized to be viewed by the user.

Figure 7 shows the destination host traffic analysis view. The interface is almost the same as figure 6. In this example, we can easily see that the host “salt.postech.ac.kr” received the most data.
Figure 7. Destination Host Traffic Analysis View

Figure 8. Source to Destination Traffic Analysis View
If we want to know which host pairs generated the most network traffic, then “Data Exchanged” will show the user the data exchange for the source to destination host pair. Figure 8 is such an example.

WebTrafMon can show the protocol information too. Figure 9 shows the network layer traffic analysis view. In this example, we can easily see that most packets were IP-based. Figure 10 shows the transport layer traffic analysis view. This example shows that most packets were UDP-based. From this information, we can know the network protocol usage status more thoroughly.

![Network Layer Traffic Analysis View](image)

Figure 9. Network Layer Traffic Analysis View
Figure 10. Transport Layer Traffic Analysis View

Figure 11 shows the application layer traffic analysis view. In this example, we can easily see that the service name “afs3-fileserver” dominated the network. The “unknown” means that the packet uses the port number which is not registered to the IANA [22].

Figure 12 is the main control panel for the real-time monitoring process. Users can check network statistics from this user interface by clicking on the buttons. The output form is almost the same as that of traffic history shown recently. The only difference is that it is operated in real-time.
Figure 11. Application Layer Traffic Analysis View

Figure 12. Real-Time Traffic Monitoring Using WebTrafMon
We have experienced one big bandwidth killer, an AFS [21] file server. When we analyzed that server, it always appeared as the dominant host that crowded our network. These days, many AOD servers offer mp3 (Mpeg Layer 3) music files [19]. Listening to music through an mp3 site causes the network bandwidth to soar. If a number of users listen to music through an mp3 site online, network traffic can peak. In this situation, we can use our tool to find which server consumes the most network resources.

Configuring WebTrafMon to run automatically and periodically gave us much information for long term analysis. We configured our system to probe the network automatically and after probing the data, collect and analyze it. This feature showed us a long-term trend and provided greater insight into our enterprise network.

Capturing and storing network packets being transferred on a 10 Mbps or 100 Mbps requires a large amount of disk space for logging, even for one minute capturing. In order to capture and store network packet information for a longer period (e.g., tens of minutes or hours) requires hundreds of megabytes (MB) of disk space. This is not much of a problem since most PCs or workstations are equipped with gigabytes (GB) of disk space nowadays. What is the major problem is quickly processing the log file of such large size and presenting the network traffic information users want to see. Thus, we have designed our logging based on sampling in order to reduce the log file size and expedite the log processing.

It is true that this may yield inaccurate information about the network use. We have experimented various sampling techniques and found that in almost all cases sampling yielded fairly accurate information. This techniques is especially useful on a high speed network and using a not so powerful PC or workstation for monitoring. If we just want to monitor the overall network status, capturing every Nth packet will reduce the system overhead without
hurting the accuracy too much.

Table 5 shows the result of each sampling rate. In this table, we compare 5-minute test. The table shows that the system doesn’t handle all packets reliably at high speed. If we did not use packet sampling, packet loss rate will be higher so the result will be more unreliable in a slow system. The reason is that the kernel drops all the forthcoming packets if the internal network buffer is full. In a slow system, the buffer is quickly filled with packets waiting for operation but the system cannot manipulate these packets fast, packet loss rate becomes high.

And even on a fast system, if the network is overloaded, then trying to capture all the packets will result in unreliable information too. So in an average case, using this sampling technique (dropping packets intentionally) will make the overall result more accurate.

<table>
<thead>
<tr>
<th>N</th>
<th># of packets</th>
<th>total traffic (bytes)</th>
<th>Approximate traffic (bytes)</th>
<th>avg. packet size</th>
</tr>
</thead>
<tbody>
<tr>
<td>No sampling</td>
<td>37504</td>
<td>26516592</td>
<td>26516592</td>
<td>707</td>
</tr>
<tr>
<td>Every 5th</td>
<td>12751</td>
<td>9026523</td>
<td>45132615</td>
<td>707</td>
</tr>
<tr>
<td>Every 10th</td>
<td>6983</td>
<td>4948933</td>
<td>49489330</td>
<td>708</td>
</tr>
<tr>
<td>Every 15th</td>
<td>4798</td>
<td>3462837</td>
<td>51942555</td>
<td>721</td>
</tr>
<tr>
<td>Every 30th</td>
<td>2448</td>
<td>1715834</td>
<td>51475020</td>
<td>700</td>
</tr>
</tbody>
</table>

Table 5. Accuracy of Packet Sampling (5-minute test)

Table 5 illustrates such situation very well. We have experimented four different sampling intervals: 1) every 5th, 2) every 10th, 3) every 15th, and 4) every 30th packet. The number of total packets and traffic (in bytes) captured and stored obviously decreases as we increase the sampling interval. However,
the approximate traffic (i.e., total traffic multiplied by N) does not decrease as we increase the sampling interval. We can see that sampling at every 15th packet yields the highest approximate traffic. This shows that the system does not capture all the packets and can get a fairly accurate information using sampling, with the reduced burden on the system resources.
7. Conclusion and Future Research

We developed WebTrafMon as a Web-based network traffic monitoring and analysis system. Why Web-based? Today’s extensive growth of computer networks has mostly been caused by the explosive use of the Internet. The Internet opened the door to a new way of searching for, and using information. What makes the Internet so popular is the Web. Using the Web is easier than any other text-based traditional Internet service (for example, Telnet, FTP, Gopher etc.). The Web-based system is now familiar to anyone who has surfed the Internet using the Web. Anyone who is connected to the Internet can use the Web-based system anywhere, anytime. The only requirement is a connection to the Internet. These days this is not a problem as most computer operating systems support a Web browser. These facts convinced us that the ideal traffic monitoring and analysis system should be Web-based. We eliminated problems that arise from various operating system: the location of access, and the time of access.

We did our best to make our system easy and powerful. Besides adopting a Web interface, we defined the traffic analysis structures according to each data communications layer. We are sure this guarantees understanding the usage our system easier because it is based on standard TCP/IP network layering.

Showing and analyzing source and destination host information is another feature that makes our system unique. For enterprise network managers, it is often more important to know which host is consuming network resources, than it is to know which protocol is prevailing. Our system is able to show such information according to source and destination hosts. This ability provides more comprehensive insight into the network’s use.

All of the above information is available to users of WebTrafMon. When the WebTrafMon system is initiated, the probe component of our system captures
the packets and extracts information of each packet and logs the information onto a log file using a predefined format. When the user wants to view the data, the user interacts with the view controller to request the information. After the view controller knows what the user wants, it reads the log file, analyzes it, and sends the analyzed data to the user’s Web browser.

Although basic features and operations were implemented for WebTrafMon, improvement remains a vast, unexplored territory. For example, we should be able to analyze a larger size of log file which the probe generates. What makes this objective challenging is that one cannot precisely predict network status.

Integrating with MRTG will also be useful. Although MRTG is an excellent tool, it can not handle source or destination host information. Configuring MRTG to run WebTrafMon when network traffic peaks or satisfies a certain condition that the user has previously defined will be useful, providing network status details at any specific time.

More work on packet sampling technique and analysis is required. We simply tested for 5 minutes once but if we test it using more time interval and more sampling period and analyze it, we will be able to know the overall network status with more accuracy and less system load.

Adding long-term analysis facility will make WebTrafMon more powerful network monitoring system. Currently it supports hourly and daily analysis only but if we add weekly, monthly, and yearly analysis facility like MRTG, the user will be able to overview the network status more easily.

And finally, making WebTrafMon support other type of network like switched Ethernet, FDDI, or ATM is required. Currently it supports shared Ethernet only.
References


http://hoohoo.ncsa.uiuc.edu/cgi/.


Appendix

A. Sample source code of WebTrafMon Viewer-hourly analysis

The following source code shows the source host traffic to the user and stores its statistics to specified format. Other traffic information can be shown almost the same as this one because all are based on the same log file. Lines that begin with "#" are comments.

#!/usr/bin/perl

# determine the file number.
# file number is used to get the time information
open(IDX, "../idx");

while(<IDX>)
{
 $NAME=hour.$_
}

close(IDX);

# open the log file that the probe has generated
# open the file handle to store its analysis data
open(LOG, ".//log");
open(HTML, "./index.html");
open(HTML_DET, "./index_det.html");
open(OUT, ">../day/$NAME");

# initial value set to 0
$totalbytes = 0;
$totalpackets = 0;

# read the log file and extract information
while(<LOG>)
{
    if(/^(\w*)\s(\w*\.*\w*\.*\w*)$/)
    {
        $array{$2} += $1;
        $totalbytes += $1;
        $totalpackets += 1;
    }
}

close(LOG);

# save analysis data to file
require "../../misc/print-html.pl";

&init_html;
print HTML "Total size of data = $totalbytes 
";
print HTML "<P>
print HTML "Total number of packets = $totalpackets 
";
print HTML "<P>
print HTML "<P>

&init_table;

foreach $item(sort { $array{$b} <=> $array{$a} } (keys %array)){
# convert IP address to domain name
use Socket;

@ = split /\./, $item;

$addr = pack('C4', @);
($addr) = gethostbyaddr($addr, AF_INET);

if($addr eq '')
{
    print HTML_DET "$item\n";
}
else
{
    print HTML_DET " $addr \n";
}

print HTML_DET "<TD VALIGN=MIDDLE ALIGN=MIDDLE

#$width = int( 400 * ($array{$item}/$totalbytes) );
print HTML_DET "<IMG SRC="../../img/m.gif" WIDTH=$width HEIGHT=20

print HTML_DET "<TD VALIGN=MIDDLE ALIGN=MIDDLE>
"n";

print HTML_DET "<TD VALIGN=MIDDLE ALIGN=MIDDLE>
"n";
$percentage = (100 * ($array{$item}/$totalbytes));
($n2) = $percentage =~ /\d+\.\d\d\)/;
print HTML_DET "$array{$item} bytes ($n2 \%)";

# write upper layer log
print OUT "$array{$item} $item\n";
}

@apparray = (sort { $array{$b} <=> $array{$a}} (keys %array));

for($i = 0; $i < 5; $i++)
{

$item = $apparray[$i];

print HTML "<TR\n";

print HTML "<TD VALIGN=MIDDLE ALIGN=MIDDLE BGCOLOR="#DDDDDD">\n";

use Socket;

@a = split '/\./', $item

$addr = pack('C4', @a);
($addr) = gethostbyaddr($addr, AF_INET);

if($addr eq "")
{
    print HTML "$item\n";
}
} else {
    print HTML "\n";
}

print HTML "\n";
$width = int( 400 * ($array{item}/$totalbytes) );
print HTML "\n";
print HTML "$array{item} bytes ($n2 \%"");

\end_table;

print HTML "\n";
print HTML "\n";
print HTML "$array{item} bytes ($n2 \%"");

\end_html;
close(OUT);
close(HTML);
close(HTML_DET);

chop($NAME);
$NAME_CUR= $NAME . "." . "html" ;
exec "cp ./index_det.html ./$NAME_CUR";

B. Sample source code of WebTrafMon Viewer-daily analysis

Daily analysis procedure is almost the same as that of hourly analysis. The only difference is that daily analysis reads the file which the hourly analysis procedure has generated and does not read the log file directly which was generated by the probe.

#!/usr/bin/perl

system "cat hour* > log";

# open the log file that the probe has generated
open(LOG, "./log");
open(HTML, "> ./index.html");
open(HTML_DET, "> ./index_det.html");

$total_bytes = 0;

while(<LOG>)
{
  if(/\gamma\\w+/s(/\w+.\w+.\w+.\w+$/)
{  
$array[2] += $1;
$total bytes += $1;
}

close(LOG);

require "../../misc/print-html.pl";

&init_html;

print HTML "Total size of data = $total bytes \n";
print HTML "<P>";

&init_table;

foreach $item(sort { $array{$b} <=> $array{$a}} (keys %array)){

print HTML_DET "<TR\n";

print HTML_DET "<TD VALIGN=MIDDLE ALIGN=MIDDLE  
BGCOLOR="#DDDDDD">\n";

use Socket;

@split \./, $item

$addr = pack('CA', @);

44
($addr) = gethostbyaddr($addr, AF_INET);

if($addr eq "")
{
    print HTML_DET "$item\n";;
}
else
{
    print HTML_DET "$addr \n";;
}

print HTML_DET "<TD VALIGN=MIDDLE>

$width = int( 400 * ($array{$item}/$totalbytes) );
print HTML_DET "<IMG SRC= "/img/m.gif" WIDTH=$width HEIGHT=20 BORDER=0>\n";

print HTML_DET "<TD VALIGN=MIDDLE>

$percentage = ( 100 * ($array{$item}/$totalbytes));
#print HTML_DET "$array{$item} bytes ($percentage \%)";
($n2) = $percentage =~ /\d+\.\d{2}/;
print HTML_DET "$array{$item} bytes ($n2 \%)";

}

@apparray = (sort { $array{$b} <=> $array{$a}} (keys %array));

for($i = 0; $i < 5; $i++)
{

$item = $apparray[$i];

print HTML "<TR \n";

print HTML "<TD VALIGN=MIDDLE ALIGN=MIDDLE BGCOLOR="#DDDDDD">\n";

use Socket;

@=split /\./, $item;

$addr = pack('C4', @a);
($addr) = gethostbyaddr($addr, AF_INET);

if($addr eq "")
{
    print HTML "$item \n";
}
else
{
    print HTML "$addr \n";
}

print HTML "<TD VALIGN=MIDDLE ALIGN=MIDDLE>\n";

$width = int( 400 * ($array{$item}/$totalbytes) );
print HTML "<IMG SRC="../../img/m.gif" WIDTH=$width HEIGHT=20 BORDER=0>\n";

print HTML "<TD VALIGN=MIDDLE ALIGN=MIDDLE>\n";

$percentage = ( 100 * ($array{$item}/$totalbytes));
($n2) = $percentage =~ /\d+\s{0,1}\d\{2\}/;
print HTML "$array{$item} bytes ($n2 \%)";

} &end_table;

print HTML "<p>";
print HTML_DET "<p>";

print HTML "<a href="/index_det.html">Detailed View</a>\n";
&end_html;

close(CUT);
close(HTML);
close(HTML_DET);
WebTrafMon (viewer) and WebTrafMon (probe) have different responsibilities.

WebTrafMon (viewer) collects and aggregates data from the traffic sent via the Internet and stores it in a database.

WebTrafMon (probe) acts as a traffic generator and simulates user behavior on the Internet to test network performance.

Both tools complement each other in the monitoring and analysis of web traffic.
WebTrafMonは、インターネットトラフィックの監視と分析を強化します。