24

FORMAL DESIGN OF PACKET FILTERING SYSTEMS

G. Osman, M. G. Darwish, and M. Zaki
Research Developing Center (RDC)
Faculty of Computer and Information – Cairo Universty
Faculty of Engineering - Azhar Universty

Abstract: Observing network traffic is necessary for achieving different purposes such as system performance, network debugging and/or information security. Observations, as such, are obtained from low-level monitors that may record a large volume of relevant and irrelevant events. Thus adequate filters are needed to pass interesting information only.
This work presents a filtering mechanism that acquirs the intersting packets from the underlying network due to the user specifications. The packets are acquired according to specific grammar rules, and they are preserved in an observation file called log-file.

Key Words: packet filtering, monitoring, event filterng, sniffing, context free grammar.

1. INTRODUCTION

Network traffic contains a huge amount of events and a lot of useful information. Therefore, observing network traffic is necessary for achieving different purposes that may include performance, debugging, security or load analysis. The network traffic can be expressed in terms of traffic volume, composition, or packet sizes in order to characterize the underlying network behavior [1]. Global traffic trends can enable network architects to design a better class of networks that may respond properly to new technologies and protocols [2].

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35586-3_46

M. A. Ghonaimy et al. (eds.), *Security in the Information Society*

Performance studies are concerned with major traffic trends that could be obtained through statistical classification of the monitored network. Moreover, the nature of the dynamic documents exchanged among the network connections by different application protocols (http,smpt,ftp) can be subjected to such statistics. Another objective of network monitoring is the debugging of communication errors. Such systems respond to alarms as they occur on the network. They work directly in real-time environments, analyzing the situations as each new alarm comes in [3]. The information security monitoring is a third objective of network monitoring systems in which, the network traffic is analyzed [4].

As such, all monitoring activities include a large volume of events, therefore filtering mechanisms are needed to exclude the unwanted occurrences. A survey and evaluation of different event filtering mechanisms are discussed by [5], [6], and, [7]. These filtering mechanisms are interested in packet filtering. Such systems employ an interpretation model for event identification and filtering so that it could detect interesting events and then perform the appropriate action.

Also they present an effective means to observe the application's behavior at run-time and it provides status information by making use of certain debugging and management tools.

In this work the foucus will be on monitoring and filtering system, in that system, there are two main functions executed at two layers, first layer is an acquistion layer in which, the underlying packets are acquired from the network and logged in a log-file. The interesting packets are selected online according to user specifications. They are preserved in a specific file at the second layer (packet-filtering layer). The output log-file contains a set of selected packets that express the events of the acquisition duration on the underlying network.

2. THE ACQUISITION PROCESS

The acquisition process is responsible for collecting the network traffic and sending it to a log file. The information that could be monitored is classified as:

- **Static information**, which characterizes the current configuration and its elements.
- **Dynamic information**, which is related to the transmission of packet on the network [8]. In this work we are interested in dynamic monitoring information only.

The acquisition process needs a network-monitoring tool, to acquire and display packets exchanged between different hosts of a network [9]. The structure of the monitoring tool is presented in figure (1), which is

implemented using a probe machine. Such machine connected to the network segment, and consists of:

(1) Hardware Elements:

- Network Interface Card (NIC), attached to the probe machine. Actually the NIC is turned to the "promiscuous mode", to make the machine work as a listener to the network traffic [10], and [11]. Consequently the NIC can gather the underlying packets.
- Buffer, once the packets are acquired from the network, they are stored in a buffer.

(2) Software Elements:

- The monitoring program, which runs on the probe machine collects the Ethernet packets, figure (2), these packets are intercepted and stored in a structured buffer according to the Ethernet packet grammar rules, described in figure (3). The monitoring program is composed of five functional modules, as presented in figure (2), they are pointed out in the following.

(1) Device initialization: this module calls a subroutine, which initializes the network interface and puts it into promiscuous mode.

(2) Structure definition: in which a number of structures are defined. They are data link layer header structure (Ethernet-header), network layer header structure (IP-header) and transportation layer header structure (TCP-header). These structures then hold the packet. The IP-packet format is described in the grammar shown in figure (3).

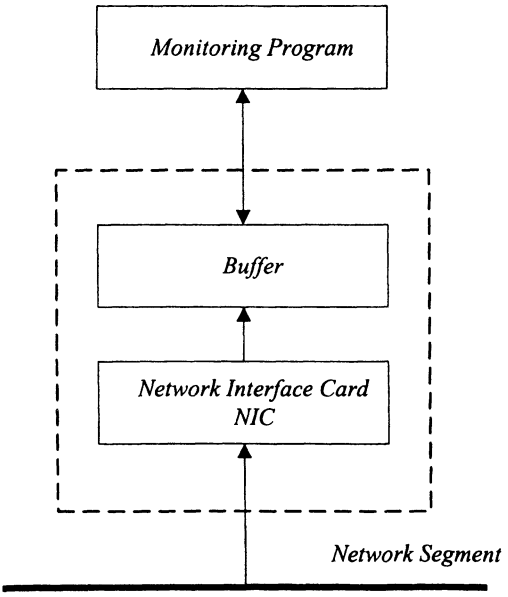

Figure (1) Structure of the monitoring tool

(3) Looping: the loop contains two parts, first is the main module of capturing process and the second is print module, the program goes into loop capturing data from the NIC and then printing it to the "log-file" and starting the loop again.
(4) Capturing: this module is concerned with reading data from NIC, when packet is sensed by the NIC, then data is read from NIC into a buffer, the buffer is the Ethernet packet structure, this structures defines the different layers of the TCP/IP protocol (Ether-header, IP-header, TCP-header, and the application layer protocol data).

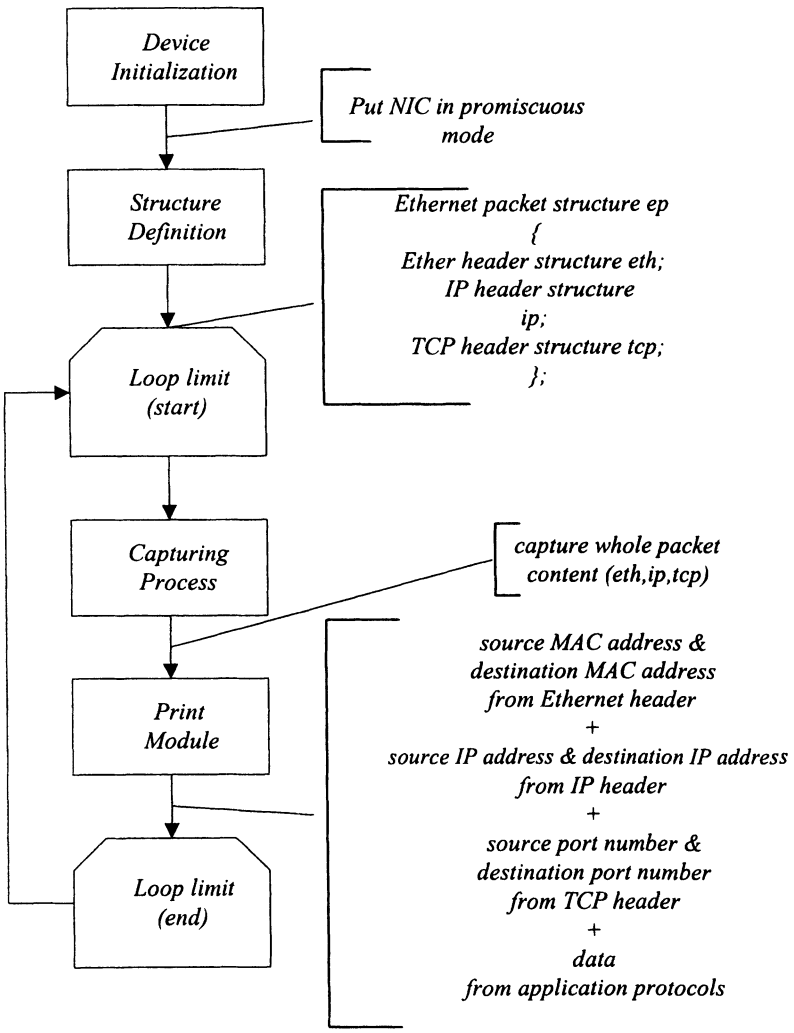

Figure (2) The monitoring program

(5) Print module: in which, the interested fields from the acquired headers (source and destination MAC addresses from Ether-header, source and destination IP-addressees from IP-header, source and destination TCP-port numbers from TCP-header, and data from application protocols) are printed into a log-file.

The produced "log-file", from the acquisition layer is structured as in figure (4), in that "log-file", there are a header part and a data part. The header part contains the selected header fields from the acquired packet, while the data part contains the selected header fields, while the the data part contains the corresponding packet data.

3. PACKET FILTERING

The second layer in the system, figure (5), is responsible for reducing the amount of data so that a user can receive only the desired information at suitable level of detail [7]. One of the best policies to minimize the unwanted information by conditional generation in which a monitoring report or log-file is generated when a certain predefined conditions are satisfied [12]. Filters will be defined by two components: filter expression and filter action. A packet filter expression describes all predicates (including the message fields and the operators), while the actions describe what will be done when the desired event is detected [5]. These filter predicates can be used to specify protocol header fields (such as TCP port numbers, IP source and destination addresses) [13].

<etherpacket>::=<ethhdr><iphdr><tcphdr><data>
<ethhdr>::=<h_dest><h_source><h_proto>
<iphdr>::=<ihl><version><tos><tot_len><id><frag_off><ttl><protocol><ipcheck>/
/<saddr><daddr><option>
<tcphdr>::=<source><dest><seq><ack_seq><doff><resv1><resv2><urg><ack>/
/<psh><rst><syn><fin><window><tcpcheck><urg_ptr>
*<data>::=(<byte>)**
<h_dest>::=<sixbyte>
<h_source>::=<sixbyte>
<h_port>::=<twobyte>
<ihl>::=<halfbyte>
<version>::=<halfbyte>
<tos>::=<byte>
<tot_len>::=<twobyte>
<id>::=<twobyte>
<frag_off>::=<twobyte>
<ttl>::=<byte>
<protocol>::=<byte>
<ipcheck>::=<twobyte>
<saddr>::=<fourbyte>
<daddr>::=<fourbyte>
*<option>::=(<byte>)**

<source>::=<twobyte>
< dest>::=<twobyte>
<seq>::=<fourbyte>
<ack_seqt>::=<fourbyte>
<doff>::=<halfbyte>
<resv1>::=<halfbyte>
<resv2>::=<twobit>
<urg>::=<bit>
<ack>::=<bit>
<psh>::=<bit>
<rst>::=<bit>
<syn>::=<bit>
<fin>::=<bit>
<window>::=<twobyte>
<tcpcheck>::=<twobyte>
<urg_ptr>::=<twobyte>
<sixbyte>::=<fourbyte><twobyte>
<fourbyte>::=<twobyte><twobyte>
<twobyte>::=<byte><byte>
<byte>::=<halfbyte><halfbyte>
<halfbyte>::=<twobit><twobit>
<twobit>::=<bit><bit>
<bit>::=0|1

Figure (3) Ethernet IP packet grammar

3.1 Header Matching

The input of the packet filter, figure (5), is a stream of Ethernet packets, which are defined according to the grammar rules in figure (3). Thus all fields for Ethernet-header, IP-header, TCP-header, and the data part of application protocol (http, ftp, or smtp) are defined. The filter expressions can be constructed on the basis of any field of these protocol header fields. The selection is based on a direct match so that a comparison is carried out between the current field value of the acquired packet and the pre-specified field value. Here, we focus only on filtering using any combinations of IP-address and TCP-port during packet acquisition.

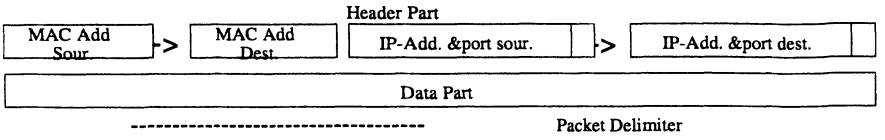

Figure (4) Layout of the produced "log-file"

3.2 Filtering Statements

The filtering process starts when a user describes his demands to the monitoring program. By this description user constructs the filter statement,

which, passed to the monitoring program, the packets monitored upon this filter statement. Table (1) demonstrates the possible filtering statements and the corresponding meaning.

Table (1) Filtering Statements

Filter Statement	Statement Meaning
Filter any	*The filter logs all monitored packets.*
Filter port [port]	*The filter only logs all packets concerned to the specified port number (80, 21, or 25).*
Filter ip <ip-add>	*The Filter only logs all packets concerned to the specified IP address.*
Filter ip port <ip-add> [port]	*Filter only logs packets concerned to the specified IP address and port number.*
Filter ip ip <ip-add1><ip-add2>	*Filter only logs packets concerned to specified two IP addresses.*

These filter statements are tested by the monitoring program to determine the value of specified filter arguments, which are used to activate the concerned filter case in the monitoring program, this process is implemented as in pseudo code which presented in figure (6). In which there are three arguments offered by the testing module, first one is argument-value[1], which accepts one string value to express the filter case ("ip", "port", "ipport", "any", and "ipip"), second is argument-value[2], which accepts the first value of argument-value[1], such as ip-address-value or tcp-port-value, and the third is argument-value[3], which is used to accept the second value of argument-value[1], such as tcp-port-value in condition 3 or ip-address-value[2] in condition 5.

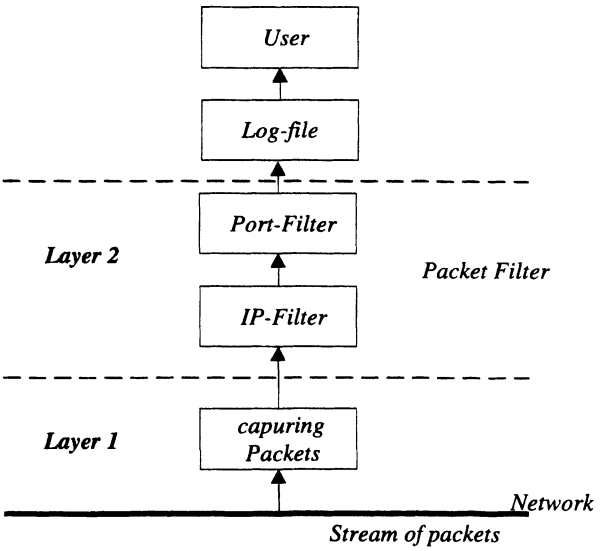

Figure (5) The proposed multilayer packet filtering model

3.3 Internal Representation of the Filter

Deterministic Finite Automata (DFA) is used to represents our event filter, figure (7), presents a finite state machine graph, in which, each state represents the history of the system environment either before or after the occurrence of an event. In this model, a filter consisting of a single primitive event is represented by a three state automaton consisting of start state, accept state, and non-acceptance (false) state, in figure (7), state 2, 3, 8, and state 10 are representing an example for a primitive event filter. While states 2, 5, 6, 7, 8 and 10 constitute an example of composite event filter, it's constructed by combining a DFAs of primitive events together into one DFA using the joining rules of finite automaton.

```
*/Start Testing of Arguments passed to Program/*
    if ((argument-value[1]="ip"))
        {    condition=1;
        activate case number 1
    using argument-value[2]=ip-address-value
    }
 if ((argument-value[1]="port"))
        {    condition=2;
        activate case number 2
    using argument-value[2]=tcp-port-value
        }
  if ((argument-value[1]="ipport"))
        {    condition=3;
        activate case number 3
    using argument-value[2]=ip-address-value
    using argument-value[3]=tcp-port-value
    }
  if (argument-value[1]="any"))
    {    condition=4;
    activate case number 4
        }
 if ((argument-value[1]="ipip"))
        {
        condition=5;
    activate case number 5
    using argument-value[2]=ip-address-value[1]
    using argument-value[3]=ip-address-value[2]
        }
/* End Testing of Arguments passed to Program */
```

Figure (6) pseudo code representing the arguments testing module

4. CASE STUDY

When a filter statement is constructed by the user, the accepted arguments is passed to filter module in order to activate the corresponding

filter case, each case has its filter expression and the desired action, the implementation of the filter core is implemented according to each case of table (1). These filter cases are presented by the pseudo code in figure (8). When certain filter case is activated by the previous argument testing module, a comparison is carried out between specified field value (arguments content) and the current field value of the acquired packet, and when the condition is satisfied, the capturing function is called to acquire the whole packet content (Ethernet-header, IP-header, TCP-header, and application protocol-data). Figure (8) presents all filtering cases and the desired action for each case. It is clear from figure (8) that both case 1 and case 2 are "one-value checking", in which one specified value is examined. But case 3 and case 5 are "two-value checking", in which compound conditions are formed. Actually, case 4 has no condition i.e. no filter. When the user specify a filter for certain IP address *(Filter ip <ip-add>)*, as in table (1), the argument is then tested by the argument testing module, which replays to activate the case 1 of the filter, figure (8). The filter expression is based on comparing both source IP address and destination IP address of the acquired packet with the specified IP address by the user, if one IP address matched the specified one, the action takes place by calling the capture module to acquire the Ether header, IP header, and TCP header, of only packets which contain a source or destination IP address that matches the user specified IP address. When case 3 is fired *(Filter ip port <ip-add> [port])*,. It is "two-value checking" case, in which, the filter expression has a compound condition, which contains two checking parts. The first part is comparing both source IP address and destination IP address of the acquired packet with the specified IP address by the user while the second part of filter expression is comparing both source TCP port number and destination TCP port number of the acquired packet with the user specified port number. If both the first and second parts of the filter expression are satisfied then the same action as case 1 is done. When the user specify the argument any *(Filter any)*, as in table (1), the argument is then tested by the argument testing module, which activates case 4 of the filter, figure (8). In this case there is no filter expression and consequently all packets will be acquired.

4.1 Packet Filter Output

The packet filter output is shown and discussed. The filtering statements in table (1) are used to fire three filters simultaneously, first filter upon port 25, second filter upon port 25 and IP-address of a client in the network, third filter is as same as second filter but on IP-address of another client on the network. We found that the first filter acquires all smtp packets that exchanged in that duration, a sample from the log-file shown in figure (9), in which there exist sample from

two different sessions. The second filter acquires only packets with TCP port number 25 and originated from certain IP-address. The third filter, acquires only the packets have also an TCP port number 25 and originated from another IP-address.

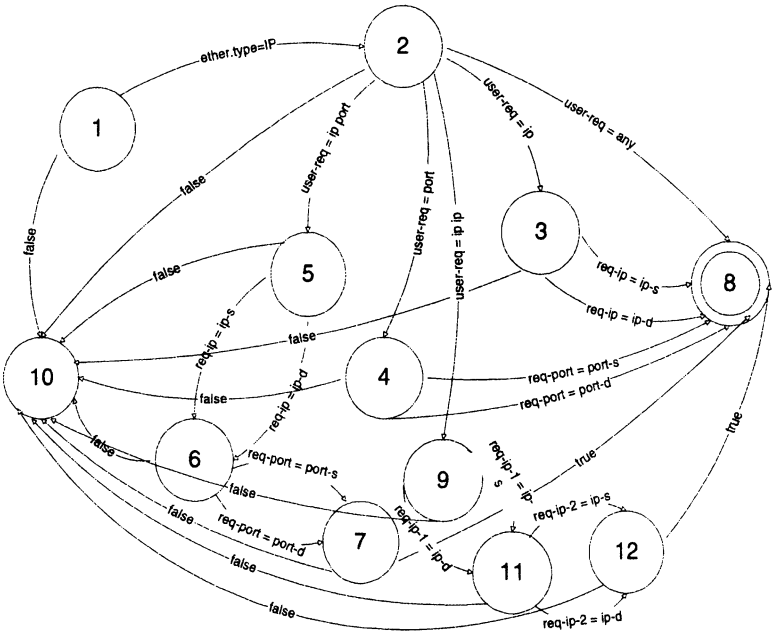

***Figure** (7) Deterministic finite automata representation of the filter*

```
case 1: if((ip-address-value=source-ip-address) OR
              (ip-address-value=destination-ip-address))
              capture (ether-header,ip-header,tcp-header,application-data);
          break;
case 2: if ((tcp-port-value=tcp-port-source) OR
              (tcp-port-value=tcp-port-destination))
              capture (ether-header,ip-header,tcp-header,application-data);
          break;
case 3: if ((ip-address-value=source-ip-address) OR
            (ip-address-value=destination-ip-address)) AND
        if ((tcp-port-value=tcp-port-source) OR
            (tcp-port-value=tcp-port-destination))
              capture(ether-header,ip-header,tcp-header,application-data);
          break;
case 4: capture(ether-header,ip-header,tcp-header,application-data);
          break;
case 5: if ((ip-address-value[1]=source-ip-address) OR
      (ip-address-value[1]=destination-ip-address)) AND
          if ((ip-address-value[2]=source-ip-address) OR
            (ip-address-value[2]=destination-ip-address))
```

```
    capture(ether-header,ip-header,tcp-header,application-data);
break;
```

Figure (8) pseudo code representing packet filter cases

00:c0:df:f0:a4:eb->00:80:48:81:0a:61 192.168.2.3[1027]->192.168.2.1[25]
Received: by client.filter.com with Microsoft Mail...id <01C09619.282A8D40@client.filter.com>; Wed, 14 Feb 2001 00:00:35 +0200..Message-ID: <01C09619.282A8D40@client.filter.com>..From: Gamal Ossman <gamal@gamal.filter.com>..To: "'sismaiel@gamal.filter.com'" <sismaiel@gamal.filter.com>..Subject: mail-1..Date: Wed, 14 Feb 2001 00:00:33 +0200..MIME-Version: 1.0..Content-Type: multipart/mixed; boundary="---- =_NextPart_000_01C09619.283B5620".....------ =_NextPart_000_01C09619.283B5620..Content-Type: text/plain; charset="us-ascii"..Content-Transfer-Encoding: 7bit.... ..------ =_NextPart_000_01C09619.283B5620..Content-Type: text/plain; name="economy.txt"..Content-Transfer-Encoding: quoted-printable....Egyptian economy is the focus of the world at the moment. the economic =..proffesionals have put a privitaization plan, and increased financial =..projects. This strengthed the monetary market of Egypt. Trade increased =..as well both inside egypt and outside. The costs of living decreased =..thus making a lot of ser
---
00:c0:df:f0:a4:eb->00:80:48:81:0a:61 192.168.2.3[1027]->192.168.2.1[25]
vices avialable to the common people. This =..revived the Egyptian market. ..------ =_NextPart_000_01C09619.283B5620--.....

00:80:48:81:0a:61->00:c0:df:f0:a4:eb 192.168.2.1[25]->192.168.2.3[1027]
250 BAA00165 Message accepted for delivery..

00:c0:df:f0:a4:eb->00:80:48:81:0a:61 192.168.2.3[1027]->192.168.2.1[25]
QUIT..

00:c0:df:49:af:9e->00:80:48:81:0a:61 192.168.2.2[1026]->192.168.2.1[25]
Received: by client2 with Microsoft Mail...id <01C061A4.CD11CAC0@client2>; Sat, 9 Dec 2000 05:56:40 +-200..Message-ID: <01C061A4.CD11CAC0@client2>..From: ghadeer <ghadeer@gamal.filter.com>..To: "'sismaiel@gaml.filter.com'" <sismaiel@gaml.filter.com>..Subject: mail-2..Date: Sat, 9 Dec 2000 05:56:22 +-200..MIME-Version: 1.0..Content-Type: text/plain; charset="us-ascii"..Content-Transfer-Encoding: 7bit....to my friend sherif ismaiel that is just for test an email exchange between client2 and server,,,,,, to check the monitoring program..check out and replay me.....gamal..

00:80:48:81:0a:61->00:c0:df:49:af:9e 192.168.2.1[25]->192.168.2.2[1026]
250 BAA00170 Message accepted for delivery..

00:c0:df:49:af:9e->00:80:48:81:0a:61 192.168.2.2[1026]->192.168.2.1[25]
QUIT..

Figure (9) sample of output from port filtering

5. CONCLUSION

In this work a packet filtering system is presented. This system exploits a monitoring system with packet filtering mechanism. In that system, the packet structure is defined according to context-free grammar rules. All the packet header fields are available (Ethernet header, IP header, and TCP header) for processing. The underlying packets are acquired from the network and logged in a log-file. The interesting packets are selected online according to some user specifications, such as IP- address, TCP-port, or both. The selected packets are preserved in a specific file at the packet-filtering layer.

REFERENCES

[1] Kevin Thompson, Gregory J. Miller, and Rick Wilder, "Wide-Area Internet Traffic patterns and characteristics", IEEE Network, November/December 1997.

[2] K. Claffy and Trace Monk, "What's Next for Internet Data Analysis? Status and Challenges Facing the Community.", Proceedings of the IEEE, October 1997.

[3] Sameh Rabie, Drew Rau-Chaplin, and Taro Shibahara, " DAD: A Real-Time Expert System for Monitoring of Data Packet Networks", IEEE Network , September 1996.

[4] Biswanath Mukherjee, L. Todd Heberlein, and Karl N. Levitt, "Network Intrusion Detection", IEEE Network, May/June 1994.

[5] Ehab S. Al-Shaer, " High-Performance Event Filtering for Distributed Dynamic Multi-Point Application: Survey and Evaluation ",Old Dominion University, Norfolk, VA,USA,Oct. 1997.

[6] Ehab Al-Shaer, " Event Filtering Framework : Key Criteria and Design Trade-Offs", The 21st IEEE International Conference on Computer Software and Applications, Pages 88-93, Washington, D.C., August 1997.

[7] Douglas C. Schmit, "High-Performance Event iltering for Dynamic Multi-Point Applications", In 1st Workshop on High Performance Protocol Architectures (HIPPARCH), Sophia Antipolis, France, December 1994, INRIA.

[8] William Stallings, SNMP, SNMPv2, and RMON: Practical Network Management, Addison-Wesly, 1996.

[9] Mahesh Jayaram, and Ron K. Cytron, " Efficient Demultiplexing of Network Packets by Automatic Parsing", National Science Foundation Grant NCR-9405444, july 19, 1995.

[10] Matt Blaze, "NFS Tracing by passive Network Monitoring" , Princeton University ,1992.

[11] Steven M. Bellovin , " Packets Found on an Internet", Computer Communications Review, July 1993.

[12] Masoud Mansouri-Samani and Morris Sloman, " Monitoring Distributed System", IEEE Network, November 1993.

[13] Guru Parulkar, Douglas, Eileen Kraemer, Jonathan Turner, And Anshul Kantawala, "An Architecture for Monitoring, Visualization, and control of Gigabit Networks", IEEE Network, September/October 1997.