Passively Monitoring Networks at Gigabit Speeds Using Commodity Hardware and Open Source Software

Luca Deri, *NETikos S.p.A.*

***Abstract*—Passive network monitoring is a complex activity that mainly consists in packet capturing and classification. Unfortunately this architecture often cannot be applied to gigabit networks, as the amount of data to capture is too large for the monitoring applications.**

This paper describes the challenges and lessons learnt by the author while monitoring networks at gigabit speeds and above. Finally, it describes an architecture able to successfully monitor networks at high speeds using commodity hardware and open source software.

***Index Terms*—Passive Network Monitoring, Packet Capture, NetFlow, sFlow.**

I. Passive Network Monitoring: State of the Art

Years ago, many computer specialists predicted the need to have more bandwidth available because people were supposed to need it mostly for multimedia applications. Time has proved that this statement was half true. Although there are many attempts to provide TV/radio access over the Internet, today most bandwidth demanding applications do not belong to the family of interactive multicast-based multimedia applications but rather to peer-to-peer (P2P) [3]. In fact while multicast transmission has not really taken off, the Internet has experienced a tremendous growth of P2P applications. Various attempts to ban this kind of applications failed because an application that has been stopped (e.g. Napster), many more appeared on the scene and new ones are appearing every month. When P2P applications were first introduced, they were mostly used for exchanging songs coded in MP3, whereas today many users share DVDs and movies. The result is that in a couple of years the average size of a file exchanged using P2P applications has risen from a few MB of an MP3 file to 700 MB of a compressed movie with a 100/200-x increase.

The use of P2P applications changed the Internet traffic not only in terms of volume but also in terms of type, as these applications redefined the concept of client and server. Usually a server is a host that provides a service to several clients. In terms of IP traffic:

- A client establishes one or more connections with a server, uses the server, and then closes the connections.
- Servers can exchange data with both clients and servers, whereas clients talk only with servers.
- Client to server connections are often asymmetrical (e.g. in HTTP the client sends little data with respect to the server).
- Packets such as ICMP port unreachable or TCP packets with the RST flag set are relatively rare, and they are used to indicate that there is a problem somewhere.
- Servers and applications usually listen on well-known ports.

With the advent of P2P, the above statements need to be revised as:

- There is not a true difference between a client and a server as a P2P application is both a client and a server.
- The number of permanently established TCP connections is very little (usually only those with master servers).
- Periodically (e.g. when a file need to be downloaded or at every new attempt to talk with additional master servers) applications attempt to establish new connections whose success rate is very little, producing quite some ICMP traffic used to report the problem.

The main consequence of the use of P2P applications is that many network monitoring applications to be modified are not rewritten. For instance a host that periodically:

- receives several ICMP Destination Unreachable or ICMP Administratively Prohibited packets;
- attempts to establish TCP connections mostly with no luck;

cannot be marked with a red exclamation mark on the management console, as this host is likely to run a P2P application.

This work was supported in part by the IST project SCAMPI (IST-2001-32404) funded by the European Union.

Luca Deri is with NETikos S.p.A., Via Matteucci 34/b, 56124 Pisa, Italy (email: luca.deri@netikos.com, http://luca.ntop.org/).

While networks changed significantly in terms of bandwidth available and type of traffic, network monitoring applications basically remained the same. Besides large companies that can afford to buy expensive network traffic monitoring applications, most of the people today monitor the traffic flowing across their network using the same tools they used three years ago, which in most of the cases means MRTG [5] polling traffic information out of network routers/switches interfaces via SNMP MIB-II variables [13].

Unfortunately today this way of monitoring networks is no longer effective because:

- The traffic has changed significantly from what it used to be a few years ago both in terms of protocols (HTTP is likely not to be the most used protocol anymore) being used and user (many end-user computers move more data than servers) behavior.
- It's not longer possible to predict what is flowing across the network using aggregate information such as the one provided by network interface counters.
- Security violation attempts are quite common and cannot be detected without using specialized tools.
- Well-known ports cannot be used anymore to identify a service (e.g. passive FTP and P2P use dynamic ports) making it difficult to calculate simple statistics such as how much FTP traffic is flowing across the local network?
- Standard monitoring protocols such as Cisco NetFlow [6] are often implemented very badly on modern network equipment. For instance, popular network appliances produced by companies such as Juniper Networks and Extreme Networks that can switch millions of packet/second, feature NetFlow implementations able to handle less than 10'000 packet/sec[1]. For this reason, network administrators cannot base their standard NetFlow-based monitoring tools on the equipment they use for operating the network unless they want to experience severe packet loss. This is yet another reason why many network administrator still reply on SNMP.

In recent years the speed of many networks has moved from the Mbit to the Gbit range. In LANs Gbit network adapters are very common in the last few years[2] and Gbit ethernet switches are now so cheap that basically every new network is going to be based on this technology. In WANs, many institutions are replacing existing ATM networks with high speed (1 Gbit and above) optical networks based on technologies such as MPLS [1] and DWDM [2] (Dense Wavelength Division Multiplexing). For instance in the past months, the local university campus backbone has migrated from 155Mbit ATM to a 1Gb MPLS network, and replaced the 45 Mbit leased line used for Internet connectivity with a 2.5 Gbit line. Finally, ADSL lines running at not less than 256Kbit have often replaced end-user Internet connectivity based on 56K/ISDN modem lines. Basically today a single user has often more bandwidth than the one that used to be available to a mid company two years ago.

A side effect of Gbit networks is that tools that were more than adequate in terms of performance for monitoring 100Mbit networks are not often able to keep up at high speeds. Most libpcap [7] based applications fall into this category. Recent studies [8] have shown that the problem is not the libpcap performance per-se, but the application that sits on top of it. For instance Snort [9], a very popular IDS (Intrusion Detection System) tool, is not able to analyze traffic above ~200 Mbit, whereas [10] has shown that simply changing the implementation of a core Snort routine the application performance could be almost doubled. Similarly, traffic sniffers such as tcpdump or ethereal cannot save traffic on disk at Gbit speeds without losing packets [8] not just because these applications are inefficient but also for other issues (e.g. the file system performance bottleneck). In order to overcome these performance limitations caused by both the operating system and the PCI bus bandwidth[3], some researchers started to use network cards equipped with NPU (Network Process Units) [11] such as the Intel IPX family. So, as to compute traffic statistics directly on the card without having to move all the traffic to the computer. The dark side of NPU cards is that they:

- are quite expensive and relatively difficult to purchase;
- are available only for a few media types, usually Ethernet and a few others;
- have little memory on board limiting dramatically the size of programs that can run on the card itself[4];
- can be programmed using very primitive tools and languages.

Therefore, the use of NPU cards solves some of the problems but introduces new limitations in terms of ease of programming and hardware availability

In conclusion, the combination of:

- High speed networks in both LAN and WAN environments.
- Performance bottleneck of many applications when used on Gbit networks.
- New traffic patterns mostly due to the use of P2P protocols.
- Slow, if any, evolution of existing monitoring tools.

made passive network monitoring at Gbit speeds quite challenging. The following chapters cover the design and implementation of an architecture, based on commodity hardware and open source software, which allowed the author to successfully monitor Gbit networks.

[1] Juniper M5 switch for instance can handle about 7'000 packet/sec.

[2] Apple manufactures Macintosh computers with 10/100/1000 Ethernet adapters since 1999

[3] Currently the 64-bit PCI bus has a bandwidth of 4 Gbit [12].

[4] Not all the NPU-based cards (e.g. Endace DAG cards) allow programmers to run programs directly on the board (Intel IXP allow this).

II. An Architecture for High Speed Traffic Measurement

A. Design Goals and Requirements

During the design phase, the author made the following decisions:

- All the hardware components need to be available on the market at a reasonable price.
- All the software measurement applications need to be open source either home grown or available on the Internet.
- The architecture should maximize the number of existing applications that can take advantage of it.
- The architecture should be easily replicable (i.e. it should not be tied to the author' environment) and really scalable.

When the author migrated the existing network to Gbit speed, he decided to purchase network equipment that feature:

- Rich monitoring capabilities.
- Ability to configure the core network switch for performing some measurements directly on it.
- Access to traffic counters using standard protocols such as SNMP.
- Ability to mirror traffic across different media types (e.g. mirror IP traffic encapsulated on ATM cells on Ethernet).
- High programmability using common languages such as C and Perl.

At the moment the switches that best fits these requirements belong to the M-series of Juniper Networks [15]. The key motivation in selecting this product has been the ability to use it both as a switch and as a sort of NPU card at a portion of the price[5]. In fact, the Juniper switches can be instrumented from remote using an XML-based language called JUNOScript [16] to:

- account traffic and access the traffic statistics via SNMP similar to NeTraMet [17];
- filter traffic using a flexible language such as BPF (Berkeley Packet Filter) [18] and either mirror, block, or account it.

The switch performs all the activities just described in ASICs (Application Specific Integrated Circuits), with no performance degradation, as the main CPU is not involved at all.

The use of commercial hardware has several advantages with respect to the use of custom hardware designed to run at high speeds:

- Cost savings: the switches avoid the use of custom cards (e.g. NPU-based cards) and they virtually cost nothing, as they are needed to run the network.
- Monitoring heterogeneous media: it is possible to use applications designed to play with IP on Ethernet to analyze IP traffic encapsulated on ATM (and other media), simply configuring the Juniper to mirror that traffic on a Gbit Ethernet interface (the switch takes care of all the conversions involved).

On the software side, the requirement to use open source software is motivated by the availability of the source code with no license constrains, that enables people to modify it in order to fully take advantage of the architecture. In particular as the author is actively developing *ntop* [19], an open source passive network traffic monitor, a major project goal is to see how ntop can be modified in order to run at Gbit speeds.

The following section describes in detail the architecture used as test bed.

B. Hardware Architecture

The picture below highlights the core architecture components.

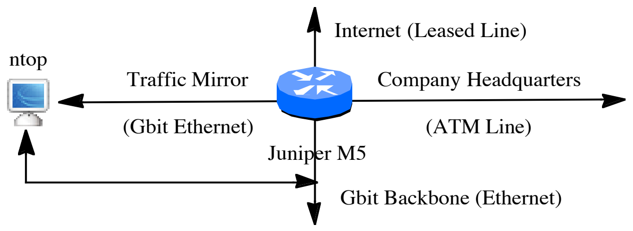

Fig. 1. Hardware Architecture Overview

The Juniper M5 is a low cost switch designed to handle more than 6 million packets/sec, that connects the local ethernet backbone with both the Internet (Leased Line) and the company headquarters (ATM). The switch is also responsible for mirroring traffic over a Gbit ethernet interface to which a PC (Dual 1.8 GHz AMD Athlon™ with 64 bit PCI ethernet adapters[6]) is connected. This PC has two interfaces: the first one is attached directly to the Juniper and is used to receive mirrored traffic, whereas the other interface is used to connect the PC to the local backbone. The PC is based on Linux and is used to run monitoring applications including ntop. Juniper traffic mirror is very flexible as it allows mirroring either all the traffic flowing across one or more interfaces or only those packets that match a filter expression. In addition, when mirroring the ATM traffic, the switch sends the IP packets without the ATM encapsulation so that monitoring applications need to support just ethernet to understand the ATM traffic. If properly instrumented, the switch can mirror a subset of the all the packets as it supports packet sampling.

C. Monitoring Architecture

This architecture is targeted at Gbit speeds as this is

[5] As of today a Juniper M5 switch with a Gbit card costs 1/4th of the popular Endace 64-bit DAG card.

[6] 64-bit PCI adapters are needed to capture traffic at Gbit speed. In fact 32 bit Gbit adapters cannot usually operate above 400 Mbit.

currently the top network speed available both at the author' premises and in several companies. Carriers currently use multi-Gbit speed networks and they will likely need a few more years before they become more widely used. The monitoring architecture has been designed starting from the following goal: provide the same metrics supported by ntop in LANs on WANs at Gbit speeds. Ntop currently performs several measurements including:

- RMON-like measurements (e.g. top senders/receivers, protocols/IP ports usage distribution, traffic matrix)
- NetFlow-like measurements (e.g. sessions tracking)
- Security measurements [20] (e.g. signature detection, ICMP traffic tracking, suspicious traffic detection).

Unfortunately not all these measurements are Gbit-friendly as some of them require several CPU cycles that could lead to severe packet loss. In order to overcome this problem the author decided to:

- instrument the switch to perform some measurements directly on the switch;
- use the traffic mirror facility to feed the Linux box with network traffic.

The measurements that can be performed on the switch make sense on the Juniper as they are performed directly on ASICs at wire speed and not handled by the main CPU. By means of JUNOScript, it is possible to implement and define some flows and associate a counter to them as shown below.

```
term anti-fragment {
        from {
             is-fragment;
             protocol icmp;
        }
        then {
             count FragmentAttack;
             syslog;
             discard;
         }
   }
```

Fig. 2. Definition of a Counter for Detecting ICMP Fragment Attacks

The value of the counter can be read using several methods including CLI (Command Line Interface), SNMP or XML-RPC over HTTP.

Complex measurements like those that are more complex than `if filtering(captured packet) is true then count` (e.g. NetFlow flow generation) cannot be implemented by the Juniper due to the way traffic is accounted. In this case the author decided to use the Juniper as an efficient packet filter and mirror appliance that forwards packets to a Linux box where complex measurement software is running. The design decision to decouple the overall implementation into Juniper + Linux has been motivated by the following facts:

The Juniper box:

- natively implements traffic mirror across different media types, so that a cheap Ethernet-based Linux box can monitor ATM or SONET traffic with no additional hardware;
- can account traffic that can be described using simple BPF-like filters.
- can be used to filter/sample uninterested traffic, hence reducing the load on the Linux box. For instance if the Juniper box is connected to a 10Gb link and a network administrator over such link wants to analyze the traffic generated by a specific host, the Juniper box can be instrumented to forward Linux just the traffic that really needs to be accounted. In this case the Linux box doesn't have to handle 10 Gbit (far too much traffic for a PC) but just a portion of it (i.e. only the traffic generated by the monitored host).

The Linux box can:

- run complex software such as a network probe or IDS that couldn't run natively on the Juniper;
- be used both as a network probe and collector for the counters implemented on the Juniper.

Basically the Juniper is used for accounting "easy to count" traffic (e.g. traffic volume), mirroring and filtering traffic (e.g. discard uninteresting packets), whereas the PC is deployed for complex accounting (e.g. session tracking, attack detection, traffic matrix).

Nevertheless, the above architecture is not the solution for all the problems whenever all the packets need to be analyzed, say for complex accounting (e.g. count the bandwidth used by P2P applications) or security purposes. In this case, the Juniper will mirror the full traffic stream to the Linux box without filtering out much traffic, causing severe packet loss when complex applications are activated on the Linux box. For instance, ntop has been designed not to handle traffic above a couple of hundred Mbit, hence it cannot operate at Gbit speeds. In order to overcome this performance bottleneck, the author decided to use a two-layer architecture based on a preprocessor plus the existing monitoring application. This solution has the advantage that the ntop code does not need to be changed nor traffic analysis facilities be removed in order to increase the application performance.

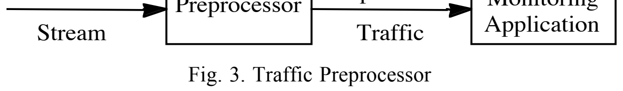

Fig. 3. Traffic Preprocessor

A preprocessor is responsible for reducing the amount of work that the monitoring application has to carry out. The simplest preprocessor is a traffic sampler that discards packets according to a specific rate similar to what sFlow-based probes [23] do. In this case the Juniper box can be used as a preprocessor as it can mirror a subset of the overall traffic at a specified rate. Although this solution is effective, it has some drawbacks:

- The monitoring application need to know at what rate it is receiving packets in order to scale traffic data.
- Some applications (e.g. IDSs) may be unable to operate properly as packets discarded by the traffic sampler may contain important information (e.g. an attack signature).

In the ntop case, a traffic sampler can be very well used for enabling ntop to cope with Gbit speeds and above. However the author decided to implement a smarter preprocessor in order to avoid using packet sampling at least for speeds in the Gbit range. The lessons learnt deploying ntop on several different environments and also using traffic generators [21] on a test network are:

- It is possible to capture traffic up to ~900 Mbit on a PC with a standard Linux distribution (Debian Linux 3.0 with no kernel tuning) based on a dual AMD Athlon XP 1.6 GHz and 64-bit PCI Gbit Ethernet adapters.
- Although many people say that libpcap performance could be a problem this statement is not completely correct. A simple
  ```
  pcap_open();
  while(1) capturePacketAndDiscardIt();
  pcap_close();
  ```
  is able to capture packets with no loss at full Gbit speed (see next chapter).
- The upper limit is ~250'000 packet/sec where CPU is used at about 85%.

The conclusion is that ntop is not able to run above ~200 Mbit because the time spent analyzing each captured packet is too long, hence the CPU is overloaded (both due to packet capture and anaysis) and packet loss occurs. For this reason the author has written *nProbe* [22], a light NetFlow v5 pcap-based open source traffic probe, to be used as preprocessor. This application is much faster than ntop as its architecture is very simple. nProbe has a large hash table that contains information about traffic probe. Each time a packet is captured, the corresponding hash bucket is updated and the captured packet discarded. Periodically the expired flows contained in hash buckets are asynchronously emitted and the bucket is empty, ready to accommodate a new flow. In nProbe the packet processing time is basically the time spent on hash lookup plus the bucket counter update. This means that nProbe is slightly more complex than the simple pcap-based test application used to study the libpcap performance. The advantage of using nProbe is that ntop does not have to handle all the packets but just the NetFlow flows sent by nProbe to ntop, which are a little portion of the initial traffic. The combination of ntop with nProbe has allowed the author to use ntop for analyzing Gbit networks mostly with no packet loss without packet sampling techniques and code changes (see next chapter). This result is very important as by configuring a light packet sampling (e.g. 1:10) on the Juniper box, it is possible to analyze multi-Gbit networks using the same architecture (scalability).

In order to simplify the Juniper instrumentation, the existing libpcap implementation, used by ntop to capture packets, has been extended as follows:

- Ability to specify packet filters in both BPF and Juniper syntax.
- Ability to specify a packet-sampling rate.
- Transparent instrumentation of the Juniper box whenever a filter or traffic rate is specified.

Juniper instrumentation is performed transparently by libpcap by means of JUNOScript. Whenever the libpcap is initialized or a new filter is specified, the library issues a JUNOScript request to set the filter into the switch so that only the traffic matching the filter is being mirrored.

Thanks to these enhancements, ntop and all existing libpcap-based applications can transparently instrument the Juniper so that mirrors only the specified traffic without the need to modify its configuration via CLI.

III. LESSONS LEARNT

This chapter describes the lessons learnt while monitoring high speed networks that have influenced the design of the architecture described in the previous chapter.

A. *What is Traffic Measurement Complexity?*

Many people believe that monitoring networks at Gbit speeds is challenging because of the high traffic volume. The author does not claim that this statement is wrong, but believes that it is half true. Recent DDOS (Distributed Denial Of Service) attacks have demonstrated that a 100Mbit stream of tiny fragmented traffic generated almost randomly, is much more difficult to handle than a few hundred Mbit of "normal" traffic. Therefore it is probably better to define the network measurement complexity in terms of number of packets/sec rather than in Mbit.

However packet rate is yet just one aspect of the problem. In fact, very often the type of measurement is influenced by the kind of traffic being measured. For instance per-host measurement (e.g. top 10 senders) requires that each packet being captured is analyzed and the host peers stored. In this case, high rate traffic between two hosts is easier to handle compared to a lower rate traffic distributed among several different hosts. This is because measurement applications need to:

- Allocate memory for each host bucket that contains host statistics.
- Spend a significant amount of time in host lookup that increases with the number of known hosts.
- Per-protocol measurement (e.g. top 10 protocols) is not influenced by the source/destination peers but just by the protocol type. In this case, as the number of different protocols is limited, the type of traffic does

not influence significantly the performance of the measurement application.

In summary, traffic measurement complexity is proportional to the packet capture rate and type of measurement required.

B. *How to Measure What?*

In high-speed networks, the time spent analyzing each single packet need to be small and limited. For instance long-standing operations (e.g. host name resolution) do not need to be performed in-line as they could lead to packet loss. nProbe has demonstrated that reducing the time spent for analyzing packets enables pcap-based applications, on adequate hardware, to cope with Gbit speeds. Of course the nProbe solution cannot be applied everywhere. For instance IDS systems need to analyze the raw packets so they cannot take advantage of NetFlow flows. In this case it is necessary to reduce the amount of packets the IDS needs to analyze. This can be achieved by configuring the Juniper so that some packets are not forwarded on the mirror interface. For instance packets coming from "safe" network interfaces and hosts could be dropped by the Juniper and not sent to the IDS. During the experiments, the author has realized that IDSs can usually operate analyzing just the packet header and initial data payload without the need to scan the rest of the packet.

For this reason the author has designed a modified version of NetFlow v5, named v5+, that also contain packet payload information as follows:

- v5+ flows have variable lengths, are compressed, support autentication, and contain fragmentation information.
- Non-ICMP, UDP and TCP flows[7] contain up to 64 bytes (user configurable) of the initial data payload.
- ICMP flows contain a bitmask that specifies the ICMP packet type and code.

In order to validate this work, nProbe has been enhanced with v5+ support, and a v5+ NetFlow collector based on Snort is currently being developed. An early prototype has been deployed on a real network and compared with the original Snort. The results are very interesting as the Snort-NetFlow application has identified most (but not all) of the problems reported by the original Snort. For instance Snort-NetFlow is not able to match signatures that:

- Have long binary strings to match with packet payload.
- Require IP packet fields (e.g. TOS) not part of the flows.
- Need per-packet TCP flags information (NetFlow reports aggregate flow information).

On the other hand this solution has the advantage of being able to run Snort at Gbit speeds without the need of additional hardware (e.g. traffic splitters).

C. *Divide et Impera Revised*

In computer science the "divide et impera" principle is very well known. The experiments described in this paper show that it can still be applied to Gbit traffic measurement. Some companies produce traffic splitters that allow an incoming high-speed link to be split into several low speed links (e.g. using products such as TopLayer's IDS Balancer). This solution has the limitation that does not solve the problem of analyzing traffic at high speeds as the amount of captured traffic is not reduced but just distributed across several probes. Instead:

- the use of traffic preprocessors such as nProbe;
- delegation of some measurements to the Juniper;
- traffic sampling;

are good ways to limit the amount of traffic that needs to be analyzed by monitoring applications and IDSs.

D. *Packet Loss*

Packet loss is not a problem for nProbe with real network traffic where packets are distributed in the 64-1500 byte ranges. Using a hardware traffic simulator that fills the network with small (64 byte) packets, nProbe experienced packet loss. Basically the probe is able to handle with no loss up to ~250'000 packet/sec. In order to decide whether this limit was due to libpcap or the operating system (OS), the author has ported nProbe directly on top of the OS using raw sockets (Linux) and /dev/bpf* (BSD) reusing code from the ettercap project [25]. The overall performance has increased a bit but the difference was negligible. Running the probe on different OSs (Linux and BSD) has produced more or less the same result[8]. The outcome of these experiments is that:

- Libpcap performance is not an issue, as it does not introduce significant computing overhead.
- The overall performance is basically the same on different OSs using the same hardware.
- Network cards and drivers significantly influence the overall performance.

IV. CONCLUSIONS

The proposed architecture is currently being used in the author's company and at the University of Pisa for monitoring the backbone. Using nProbe, ntop can be used to account traffic at Gbit speeds with no modification. Juniper and ntop counters are periodically saved on RRD databases [24] using homegrown software[9]. The traffic analysis in the past months has enabled network administrators to identify several network flaws and hosts

[7] On TCP flows, in most of the cases it is enough to add payload information only is the flow to emit has the SYN|ACK flags set (i.e. those that identify the session begin), in order to reduce the flow size.

[8] The flow collector (cflowd) performs (with respect to flow loss) much better on BSD than on Linux, probably due to a more efficient packet buffering.

[9] The site http://mrtg.unipi.it/ contains the current network map and provides access to network traffic statistics produced using the architecture here described.

that were producing unexpected traffic mostly due to misconfiguration and viruses. These problems were not identified using MRTG and SNMP interface counters on the border gateway.

All the software described in this paper is freely available on the Internet (http://www.ntop.org) under the GNU GPL license. Some Unix distributions including but not limited to FreeBSD and Linux, come with ntop preinstalled.

ACKNOWLEDGMENT

The author thanks Stefano Suin, Paolo Caturegli and Davide Vaghetti of the Centro Serra of the University of Pisa for providing requirements, and suggestions other than for letting him run some tests on their backbone. In addition he thanks Yuri Francalacci of NETikos for his help during the design and testing of the tools here described, and Fabio Palozza of Juniper for his support.

REFERENCES

[1] E. Rosen, A. Viswanathan, and R. Callon, “Multiprotocol Label Switching Architecture”, RFC 3031, January 2001.
[2] Cisco Systems, “Introducing DWDM”, http://www.cisco.com/univercd/cc/td/doc/product/mels/cm1500/dwdm/dwdm_fns.htm, September 2002.
[3] F. Manjoo, “Peer-to-Peering Into the Future”, Wired Magazine, February 2001.
[4] J. Kwan, “Peer-to-Peer Promises to Reshape the Net”, Mercury News, http://www0.mercurycenter.com/svtech/news/top/docs/peer021201.htm, December 2001.
[5] T.Oetiker, “Multi Router Traffic Grapher (MRTG)”, http://www.mrtg.org/.
[6] Cisco Systems, “NetFlow Services and Applications”, White Paper, http://www.cisco.com/warp/public/cc/pd/iosw/netlct/tech/napps_wp.htm, July 2002.
[7] Lawrence Berkeley National Labs, “libpcap”, Network Research Group, http://www.tcpdump.org/.
[8] F. Risso, and L. Degioanni, “An Architecture for High Performance Network Analysis”, Proceedings of ISCC 2001, Hammamet, Tunisia, July 2001.
[9] M. Roesch, “Snort - Lightweight Intrusion Detection for Networks”, Proceeding of LISA ‘99, Seattle, November 1999.
[10] E. Markatos, and others, “Exclusion-based Signature Matching for Instrusion Detection”, International Conference on Communications and Computer Networks (CCN 2002), October 2002.
[11] G. Memik, and W.H. Mangione-Smith, “Specialized Hardware for Deep Network Packet Filtering”, Proceeedings of FPL 2002, Montpellier, France, September 2002.
[12] Quatech Inc., “PCI Bus Overview”, Technical Report, http://www.quatech.com/Application_Objects/FAQs/comm-over-pci.htm, 2001.
[13] K. McCloghrie, and M. Rose, “Management Information for Network Management of TCP/IP-based Internets: MIB-II”, RFC 1213, March 1991.
[14] W. Betts, “Defying Denial of Service Attacks”, White Paper, Network Magazine, May 2001.
[15] Juniper Networks, “M-Series Routers”, http://www.juniper.net/products/, 2002.
[16] Juniper Networks, “JUNOScript”, http://www.juniper.net/support/JUNOScript/, 2002.
[17] N. Brownlee, “NeTraMet - A Network Traffic Flow Measurement Tool”, http://www.caida.org/tools/measurement/netramet/, 2002.
[18] S. McCanne, and V. Jacobson, “The BSD Packet Filter: A new architecture for user-level packet capture”, Proceedings of 1993 Winter USENIX Conference, San Diego, January 1993.
[19] L. Deri, R. Carbone, and S. Suin, “Monitoring Networks Using ntop, Proceedings of IM 2001”, Seattle, May 2001.
[20] L. Deri, S. Suin, and G. Maselli, “Design and Implementation of an ADS: an Empirical Approach”, Proceedings of Terena TNC 2003 Conference, Zagreb, Croatia, May 2003 (to appear).
[21] A. Götje, “nProbe Performance Evaluation”, Technical Report, http://www.ntop.org/nProbe.html, Hauman Technologies Inc., 2002.
[22] L. Deri, “nProbe: an Open Source NetFlow Probe for Gigabit Networks”, Proceedings of Terena TNC 2003 Conference, Zagreb, Croatia, May 2003 (to appear).
[23] InMon Corp., “sFlow: A Method for Monitoring Traffic in Switched and Routed Networks”, RFC 3176, September 2001.
[24] T.Oetiker, “Round Robin Database (RRD) Tool”, http://www.rrdtool.org/.
[25] A. Ornaghi, and M. Valleri, “Ettercap 0.6.9”, http://ettercap.sourceforge.net/, 2002.