Effective Traffic Measurement Using ntop

Luca Deri, Finsiel S.p.A.

Stefano Suin, University of Pisa

ABSTRACT

Traffic measurements are becoming increasingly complex due to the variety of traffic types and the integration of different network media. Although traditional tools such as sniffers and network probes are still useful for traffic measurement, they often limit their scope to network packet analysis and visualization. This means that human operators need to manually perform specific actions when some anomalous traffic conditions happen since they have very limited, if any, support from the above tools. This article covers the design and implementation of ntop, an open-source Web-based traffic measurement and monitoring application. It enables users to track relevant network activities including traffic characterization, network utilization, network protocol usage, and congestion detection. Ntop's extensibility through dynamically loadable software components opens it to extensions by network administrators. In addition, its native security flaw detection facilities allow ntop to detect potentially dangerous traffic conditions, and hence to dynamically and autonomously adapt network configuration to tackle the identified problems.

BACKGROUND AND MOTIVATION

Network traffic measurement has been considered a necessary activity since the early days of networking. Administrators had to keep track of network traffic for several reasons, including detection of network bottlenecks and planning network extensions. In the last few years, this activity has become increasingly challenging for several reasons:

- The integration of several network media makes the identification of communication problems increasingly difficult, and requires network sniffers and probes able to keep up with the increased network speed.
- The interconnection of existing autonomous networks partially based on IP causes non-IP protocols such as NetBEUI, AppleTalk, and IPX to be silently propagated into existing IP networks, interfering with existing protocols and degrading network performance.
- Users administer their own PCs, making the network administrator's job even more difficult because he has limited control over such hosts.
- Network administrators need to constantly monitor the usage of protocols that make extensive use of the network bandwidth and hence considerably affect overall performance.

UNIX traditionally provides tools for testing basic connectivity problems as well as network sniffers such as `tcpdump` or *snoop*. These tools are very powerful for tracking network and protocol connectivity issues. However, they need offline analysis tools such as `tcpshow` and `tcptrace` for better analyzing and correlating captured data as well as identifying network flows. Similarly, network probes such as remote monitoring (RMON) agents are quite powerful, but unfortunately need sophisticated Simple Network Management Protocol (SNMP) managers able to configure and instrument them properly, and analyze collected data. Due to this complexity as well as the cost of such probes, RMON agents are basically used uniquely by network managers in large institutions. Other tools for network monitoring, such as NeTraMet [1], offer advanced programming languages for analyzing network flows and building statistical event records. Although these tools offer great advantages in terms of flexibility and user configuration with respect to RMON agents, they still require an SNMP manager to collect traffic data and do not usually provide administrators with facilities for triggering actions when specific network traffic patterns are detected.

On the Internet there are several tools designed to detect network security flaws and potential attacks in progress. Beside some exceptions such as Network Flight Recorder (NFR) [2], these tools are usually designed to detect attacks against a single host (usually the one where the tool has been activated) and do not provide network/subnet protection. Traffic measurement tools do not usually offer support for security; nor do they allow active actions to be taken when an attack happens; they simply notify the administrators when an attack has already occurred. This is because measurement tools classify network traffic according to some specified static rules with defined thresholds. These thresholds are often either not able to express

0163-6804/00/$10.00 © 2000 IEEE

complex traffic patterns (e.g., security attacks) or flexible enough to be able to cover a whole subnet without having to define the same rule for all the hosts of the subnet.

Ntop is a Web-based traffic measurement and monitoring application. It was initially written by the authors to tackle performance problems of the campus network backbone because available traffic monitoring tools were not satisfactory for the reasons listed above. Similar to the UNIX *top* tool that reports processes' CPU usage, the authors needed a simple tool able to measure network traffic and report information about captured packets. Ntop then evolved into a more flexible, extensible, and powerful tool as people over the Internet downloaded it and reported problems and suggestions.

The following section covers the ntop architecture design, its components and their interactions, and implementation details and tricks used to make the implementation efficient. Then we show how ntop can be used effectively for both traffic measurement and intrusion detection [3] in some real scenarios. Finally, some performance issues are analyzed and discussed.

NTOP ARCHITECTURE DESIGN GOALS

Ntop is an open-source application written in C, available free of charge under the GNU public license. This statement does not just mean that ntop's source code is freely available on the Internet, but also that many requirements came directly from early ntop adopters. The authors designed the first version of ntop, and then accommodated new requirements and extensions on the original architecture, which has been strongly influenced by the Webbin [4] architecture. Ntop's main design goals include:

- Portability across UNIX and non-UNIX (e.g., Win32) platforms
- Simple and efficient application kernel with low resource (both memory and CPU) usage
- The ability to monitor and manage a network from a remote location without the need to run specific client applications to analyze traffic information
- Minimal requirements (bare operating system), but capable of exploiting platform features if present (e.g., kernel threads)
- The bility to present data in both a character-based terminal and a Web browser
- Traffic analysis output rich in content and easy to read
- User extensibility via dynamically loadable software components (plugins)

The idea is to develop a simple and efficient application kernel able to handle general tasks, including:

- Packet capture and demultiplexing independent of the operating system (the same code should run unchanged on different platforms) and the network interface card (NIC) interface type used for packet capture
- Basic traffic analysis and protocol characterization
- Specification and measurement of network flows
- Embedded HTTP server for visualizing traffic data without the need to use an ad hoc client application

Ntop's design shares the UNIX philosophy: applications do not necessarily have to be large and monolithic, but can profitably be divided into small independent pieces that cooperate to achieve a common goal. The kernel is responsible for handling basic tasks efficiently and providing facilities to plugin developers that can exploit kernel services. This allows developers to keep the plugin complexity low and focus just on the plugin functionality, since the kernel will handle all other tasks. The use of plugins allows users to activate only the needed ones depending on the specific situation where ntop is being used, thus avoiding spending precious CPU cycles on plugins providing information that is not significant in a particular context. In addition, it leverages the plugin complexity since many basic services are provided by the ntop kernel, simplifying the implementation of new plugins.

Before analyzing the scenarios where ntop is used, it is worth having an overview of the application internals, its components, and its interactions.

PACKET CAPTURE

Packet capture is the ntop component that has potentially more portability issues than other components. In fact, unlike other facilities such as threads, there is no portable library for packet capture. Under UNIX the libpcap library provides a portable and unified packet capture interface, whereas other operating systems provide a proprietary capture facility. Due to the good design of libpcap, the authors decided to port libpcap even to non-UNIX platforms, embedding into it native platform capture facilities (e.g., NDIS on Win32). This allowed the ntop source code to be unique across various platforms.

Packet capture libraries often have small internal buffers that prevent applications from being able to handle burst traffic. In order to overcome this problem and hence reduce packet loss, ntop buffers captured packets. This allows the packet analyzer to be decoupled from the packet capture and not lose packets due to burst traffic.

PACKET ANALYZER

The packet analyzer processes one packet at a time. Packet headers are analyzed according to the network interface being used. Hosts information is stored in a large hash table whose entries contain several counters that keep track of the data sent/received by the host, sorted according to the supported network protocols. When necessary (e.g., periodically or if there are no entries left) ntop purges the host table in order to avoid exhausting all the available memory and creating huge tables that decrease overall performance. This guarantees that ntop's memory utilization does not grow indefinitely, and that packet processing time does not increase linearly with the number of active hosts.

Caching is performed in two steps. First-level caching is semi-persistent and based on GNU `gdbm`, whereas second-level caching is implemented using a Standard Query Language (SQL) database. Ntop locally caches semi-persistent information such as IP address resolution (mapping numeric/symbolic IP address) and remote host operating system [5] (computed using the

Packet capture libraries often have small internal buffers that prevent applications from being able to handle burst traffic. In order to overcome this problem and hence reduce packet loss, ntop buffers captured packets.

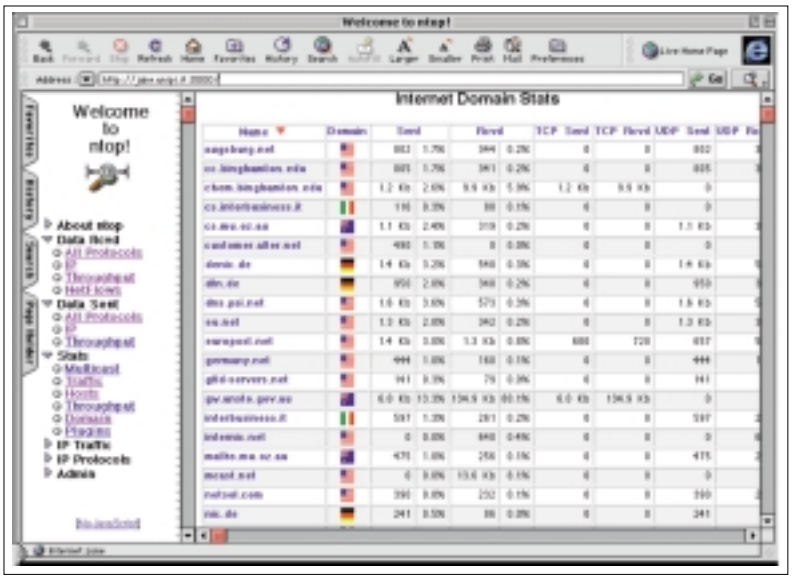

Figure 1. *The ntop Web interface.*

nmap tool). In order to reduce Domain Name Service (DNS) queries, ntop processes DNS reply packets and caches mappings for future use. Network events (e.g., TCP sessions), performance data, and other relevant information are stored permanently in the database. Storage happens either periodically or whenever the garbage collector has to purge some data. ntop talks with the database by means of a client application that interacts with the database using either Perl Database Interface (DBI) or Java Java Database Connectivity (JDBC), depending on the implementation language. This architecture allows ntop to be decoupled from a specific database and able to communicate with a remote database (e.g., the main company database) while having a very simple and light database client.

NETWORK FLOWS

Packet filtering is based on the Berkeley Packet Filter (BPF) [6] facility part of libpcap. BPF allows filters to be specified using simple English-like expressions such as those accepted by tcpdump. For better performance, filters are compiled and optimized prior to their storage inside ntop. A network flow is a stream of packets that matches a user-specified rule. Rules are specified using BPF expressions at ntop start up. Similar to NeTraMet flows, ntop network flows can be used to specify traffic of particular interest. Ntop applies all the stored flow filters to each captured packet. When a packet matches a filter, the flow counters are updated. Please note that packet processing time increases with the number of defined flows and the complexity of the associated filters.

HTTP SUPPORT

Although ntop can present traffic information in text-based terminals, it allows Web technologies to be taken advantage of. In fact, the ntop kernel contains an embedded HTTP/HTTPS server that offers users a view of traffic information richer than that offered by the terminal interface (Fig. 1).

The server offers HTTP/HTTPS authentication and allows administrators to specify users that have access to selected URLs. Users' passwords are stored in encrypted form in a database for greater security. When ntop is first started, sensitive traffic information is protected by default passwords. Administrators can then fully configure ntop using a Web browser.

PLUGINS

Plugins are shared libraries (dynamically loadable libraries, DLL, in Windows terminology) with a well-defined entry point stored in a specified directory (`plug-ins/` by default). At startup, ntop lists the stored plugins and loads them sequentially in alphabetical order. Developers can use plugins to extend the ntop kernel, define custom views of traffic information captured by ntop, and implement advanced traffic flow counters that perform additional operations besides basic traffic measurement.

MEASURING NETWORK TRAFFIC USING NTOP

Ntop has been developed to give us a simple, free, and portable tool for measuring traffic. Its development started because we were not satisfied with the existing traffic monitor tools, as described earlier. ntop focuses on:

- Traffic measurement
- Traffic characterization and monitoring
- Detection of network security violations
- Network optimization and planning

The following sections cover the above areas in depth and show what kind of information is provided by ntop to network administrators in order to help them to identify network flaws, optimize the network, and plan future extensions.

Since ntop makes use of some publicly available tools/libraries, the following table helps the reader to understand which services such tools and libraries provide, and for what purpose they are used by ntop.

TRAFFIC MEASUREMENT

In our view traffic measurement consists of measuring relevant traffic activities. Ntop associates each captured packet with the sender/receiver host. In this way, given a host (e.g., its name, NIC, or IP address), it is possible to find all traffic activities related to it observed by ntop. For each host, ntop records the following information:

- Data sent/received: The total traffic (volume and packets) generated or received by the host classified according to network protocol (IP, IPX, AppleTalk, etc.) and, when applicable, IP protocol (FTP, HTTP, NFS, etc.).
- IP multicast: The total amount of multicast traffic (volume and packets) generated or received by the host.
- TCP sessions history: The list of currently active TCP sessions established/accepted by the host and associated traffic statistics.
- UDP traffic: The total amount of UDP traffic (volume and packets) sorted by port. It is worth noting that it is possible to recognize simple portscan and protocol scan (e.g., an SNMP manager issued SNMP requests to a given host) when the host has received packets at a specified port but has

sent no data.
- TCP/UDP used services: The list of IP-based services (e.g., open and active ports) provided by the host with the list of the last five hosts that used them.
- Operating system (OS) type: Although it is rather simple to guess the OS of hosts running UNIX (e.g., it is sufficient to look at the banner displayed using the telnet command), in general OS guessing is a difficult task. Ntop makes use of nmap, which identifies the OS by sending some bogus packets and comparing the replies, if any, with a database of known patterns.
- Used bandwidth percentage: Actual, average, and peak bandwidth usage.
- Traffic distribution: Local (subnet) traffic, local vs. remote (outside specified/local subnet), remote vs. local.
- IP traffic distribution: UDP vs. TCP traffic; relative distribution of the IP protocols according to the host name.
- Local network usage: Statistics about open sockets, data sent/received, and contacted peers for each process running on the host where ntop is active.

In addition, ntop reports global traffic statistics, including:
- Traffic distribution: Local (subnet) traffic, local vs. remote (outside specified/local subnet), remote vs. local.
- Packet distribution: Total number of packets sorted by packet size, unicast vs. multicast vs. broadcast, and IP vs. non-IP traffic.
- Used bandwidth: Actual, peak, and average bandwidth usage.
- The list of active TCP sessions for each known host.
- Protocol utilization and distribution: Distribution of the observed traffic according to both protocol and source–destination (local vs. remote).
- Local subnet traffic matrix: 2D matrix where each cell (X, Y) contains the traffic sent by host X to host Y, where X and Y are hosts that belong to the local subnet of the host where ntop is running.
- Network Flows: Traffic statistics for each user-defined flow.

The current ntop version comes with a couple of plugins that provide detailed statistics about NFS/NetBIOS protocol usage and display the network bandwidth used by such protocols.

Ntop differs from many traffic monitoring tools because it transparently processes traffic data while capturing packets, and provides traffic information in a human-readable format. Other tools either capture data first, leaving traffic analysis to additional applications, or provide very basic information about captured data, pushing the user to write macros or scripts for extracting the information in which they are interested.

Traffic Characterization and Monitoring

Traffic monitoring is the ability to identify those situations where network traffic does not comply with specified policies or exceeds some defined thresholds. In general, network administrators specify some policies that all hosts must obey. Ntop natively provides support for detecting some network configuration problems, including:
- Use of duplicate IP addresses.
- Identification of all the subnet routers so that it is possible to find out whether a misconfigured host wrongly believes it acts as a router for the local subnet or a host is using a wrong netmask for the actual network.
- Identification of local hosts that have set the network card in promiscuous mode (see the "Spy Detection" section below).
- Misconfiguration of software applications: The analysis of some protocol traffic data allows administrators to guess that there is something wrong on a certain host. For instance, the use of ntop has allowed us to detect the installation of an unauthorized caching DNS which was filling up its cache very frequently, and a misconfigured NTP client that was asking the time of day once every 5 s.
- Service misuse detection: In order to reduce network traffic, administrators require that users make use of proxy applications (e.g., HTTP/FTP proxy) instead of direct connection to the remote site. Proper ntop configuration allows an administrator to identify hosts/users that do not make use of the specified proxies.
- Protocol misuse: Identification of those computers that speak unnecessary protocols. For instance, the Windows OS installs by default protocols such as NetBEUI and IPX, while most people use just TCP/IP.
- Excessive network bandwidth utilization: In organizations where the Internet connection has limited bandwidth, it is important to detect the hosts/users that use most of the available bandwidth. For instance, this can be done by either tracking traffic values for certain protocols (i.e., HTTP or FTP) or identifying hosts with connections established with remote hosts.

Ntop identifies the subnet routers by checking the association destination IP/medium access control (MAC) address in each captured packet (not just those directed to non-local IP addresses). Subnet routers are identified by the destination MAC address, whereas hosts with misconfigured netmasks are identified because they send a router those packets which are directed to hosts belonging to the local subnet.

The identification of duplicate IP addresses and the list of subnet routers is performed by the `arpwatch` plugin. This plugin keeps track of Address Resolution Protocol (ARP) packets, hence identifying IP address modification (e.g., a host changes its IP address either manually or via protocols such as DHCP or BOOTP) and MAC address clash for the same IP address (a host receives multiple replies for a single ARP request). ArpWatch is also used for packet spoofing detection, as explained below.

Detection of Network Security Violations

In networks most security attacks come from the network itself. For this reason ntop provides users with support for both tracking ongoing attacks and identifying potential security holes, including:
- Portscan detection: The classic (send a

Subnet routers are identified by the destination MAC address whereas hosts with misconfigured netmasks are identified because they send a router those packets which are directed to hosts belonging to the local subnet.

Using ntop, we have often identified OSPF and IGMP traffic in networks where these protocols were not used. In addition, protocols such as IPX, when used just by a single host in the network, generate some periodic broadcast traffic that is then propagated in the whole subnet.

packet to every port) and slow (a kind of portscan where port scan happens very slowly in order to make its detection more difficult) portscan (stealth scan) [7] can easily be detected. In fact, ntop reports the name of the last three hosts that sent a packet to each port less than 1024. We are currently adding support for negative port scan (e.g., NULL and XMAS scan). Please note that portscan is detected not only for the host where ntop is running but for all the hosts for which ntop can capture packets (usually the whole subnet). This means that ntop provides (sub)network portscan detection, whereas very few OSs support portscan detection, and then just for the host where the OS is running.

- Spoofing detection: *Spoofing* [8] happens when a host claims to be another host for the purpose of intercepting packets. In general, for packets that do not originate on the subnet where ntop runs, it is not possible to detect spoofing. Instead, spoofing can be detected at least for hosts belonging to the same subnet of the host where ntop is running. The arpWatch plugin part of ntop warns the user when two distinct IP addresses map to the same hardware address. Please note that spoofing detection should be used properly on networks where proxy ARP routers are installed or whenever a host has enabled multihoming support.
- Spy detection: A *spy* [9] is a host whose network card is set in promiscuous mode for capturing packets independent of whether they are directed to the card or not. Network Promiscuous Ethernet Detector (neped) is a tool that sends to each host X of the local subnet an ARP request containing the X IP address as the target IP address. Unfortunately, the algorithm just described does not work for all OSs; hence, nothing can be said about hosts whose NICs are apparently not set in promiscuous mode. Ntop periodically runs neped and warns the user about hosts whose cards are set in promiscuous mode.
- Trojan horse detection: Usually, users do not detect the presence of Trojan horse applications such as BO2K (http://www.bo2k.com/) until they experience severe problems. Because these kind of applications make use of well-known ports (e.g., the BO2K default port is 3777), ntop can detect their presence by periodically verifying whether there is some network traffic originated by/designated for these ports.
- Denial of service: *Synflood* [10] is the ability of an host to send packets with the SYN flag set (the SYN flag is used for opening a TCP connection) to a victim's open ports without proceeding further in the establishment of a connection. In this way the attacker fills all the victim's IP stack connection slots until the victim cannot accept new connections. Although some OSs natively offer synflood protection and hence are not affected by this problem, ntop can be used to detect attackers and report the problem to the network administrator. Please note that other kinds of attack, including *smurf* and *fraggle*, are also detected by analyzing the traffic sent/received by each host.

When a security violation or network misconfiguration/problem is identified, ntop offers facilities for:

- Reporting the problem to the network administrator
- Understanding where/how the attack originated by using the traffic information stored in the SQL database
- Performing specific actions (when applicable) in order to block the attack and hence limit its extension to the whole network

At the moment we are developing plugins for notifying network administrators about the problem via email, SNMP traps or GSM SMS (Short Messaging System) using some simple scripts that are executed once a well known security problem is identified.

Network Optimization and Planning

Often network performance is influenced by a suboptimal configuration of some hosts and by a non-efficient utilization of the available bandwidth. In particular ntop allows administrators to:

- Identify unnecessary protocols: Sometimes traffic is generated by hosts/routers that have not been configured properly and attempt to communicate with peers using protocols nobody else is using. Using ntop, we have often identified Open Shortest Path First (OSPF) and Internet Group Management Protocol (IGMP) traffic in networks where these protocols were not used. In addition, protocols such as IPX when used only by a single host in the network generate some periodic broadcast traffic which is then propagated in the whole subnet.
- Identify suboptimal routing: The `icmpwatch` ntop plugin is responsible for handling Internet Control Message Protocol (ICMP) packets. It is possible to identify machines that use nonoptimal routing just by keeping track of ICMP Redirect messages or periodically analyzing the list of subnet routers.
- Traffic characterization and distribution: Ntop allows administrators to understand how traffic is distributed with respect to the protocol and origin (local vs. remote traffic). The study of traffic patterns helps administrators to understand how the network is used both locally and from remote locations, and hence to improve, if possible, the global network topology and configuration. For instance, using ntop we have realized that our router had to route many DNS packets simply because the DNS was not placed in the best subnet.
- Reduction of the number of protocols used: In some cases two or more protocols are used for the same purpose. The use of ntop allowed us to see that in our network a few Windows computers made use of both NetBIOS and NetBIOS-over-IP, whereas the rest of the network used just NetBIOS-over-IP. By changing the network configu-

ration of these few hosts by removing NetBIOS, the number of protocols used has been reduced without losing any existing functionality.

- Wiser bandwidth usage: Network bandwidth is never sufficient; hence, it is valuable to try to avoid unnecessary communications. Studying how certain protocols are used helps administrators identify where to add applications such as proxies which allow traffic to be significantly reduced by caching information.

In general, ntop combines features otherwise present in various tools which are not always easy to integrate. Its unique user interface allows administrators to immediately take advantage of ntop without the need to purchase and manage the client applications necessary for tools such as RMON or NeTraMet. In addition, database support makes ntop suitable not only for network problem debugging, but also for long-standing network monitoring and problem backtracking.

Performance Issues

Ntop performance is quite good because:

- libpcap performance is excellent.
- Packet loss (if any) is very low because captured packets are buffered twice, both inside the kernel and in ntop.
- Potentially long-running actions (e.g., IP address resolution) are implemented asynchronously.
- ntop spawns several threads that prevent user interaction (e.g., HTTP user requests) from interfering with data collection.
- ntop makes extensive use of hash tables whose indexes are easy to compute yet fast during information retrieval due to the nature of network addresses (e.g., they are unique and already in 32/48-bit numeric format).

Users have tested ntop extensively on various network media running at different speeds. In general, ntop performance is greatly influenced by other running processes because some CPU-greedy applications may take up all the CPU cycles for a few seconds, causing packet loss. Assuming ntop is run on an averagely loaded host, tests have shown that ntop can work with very low (if any) packet loss on a 100 Mb Ethernet.

Final Remarks

This article attempts to show how ntop can be used for traffic measurement and monitoring. Features such as the embedded HTTP server, support for various network media types, light CPU utilization, portability across various platforms, storage of traffic information into an SQL database, extensibility via software components, and integration with many network tools make ntop suitable for those who want to analyze network traffic without having to pay for expensive tools that often have limited scope and lack many of the features offered by ntop.

Availability

Both ntop and libpcap for Win32 are distributed under the GPL2 license and can be downloaded free of charge from http://www.ntop.org/. Some UNIX distributions, including FreeBSD and Linux, come with ntop preinstalled.

References

[1] N. Brownlee, "NeTraMet v.4.2 Users' Guide," http://www.auckland.an.nz/net/Accounting/, 1998.

[8] "Computer Emergency Response Team: TCP SYN Flooding and IP Spoofing Attacks," CMU rep. CA-96:21, 1996.

[3] W. Cheswick and S. Bellovin, *Firewalls and Internet Security: Repelling the Wiley Hacker*, Addison-Wesley, 1994.

[4] L. Deri, "Surfin': Network Management Applications Across the Web," *Proc. 2nd Int'l. IEEE Wksp. Sys. and Network Mgmt.*, 1996.

[5] Fyodor, "Remote OS Detection via TCP/IP Stack Fingerprinting," http://www.insecure.org/nmap/nmap-fingerprinting-article.txt, 1998.

[7] Fyodor, "The Art and Detection of Port Scanning," *Sys Admin. Mag.*, Nov. issue, 1998.

[6] S. McCanne and V. Jacobson, "The BSD Packer Filter: A New Architecture for User-level Packet Capture," *Proc. 1993 Winter USENIX Conf.*, 1993.

[9] B. Mukherjee *et al.*, "Network Intrusion Detection," *IEEE Network*, vol. 8, no. 3, 1994.

[2] M. Ranum *et al.*, "Implementing a Generalized Tool for Network Monitoring," *Proc. LISA '97, USENIX 11th Sys. Admin. Conf.*, http://www.nfr.com/forum/publications/LISA-97.htm, 1997.

[10] C. Schuba *et al.*, *Analysis of a Denial of Service Attack on TCP*, COAST Laboratory, Purdue University, 1998.

Biographies

LUCA DERI (l.deri@finsiel.it) is currently sharing his time between Finsiel S.p.A. and the Centro Serra at the University of Pisa. He received his Ph.D. in computer science with a thesis on software components from the University of Berne in 1997. He previously worked as a research scientist at the IBM Zurich Research Laboratory, and as a research fellow at University College London. His professional interests include network management, software components, and object-oriented technology. His home page is http://www.tlcpi.finsiel.it/~deri/.

STEFANO SUIN (stefano@ntop.org) got its degree in computer science from the University of Pisa in 1986. After a short experience running its own company, he is currently heading Serra, the networking center of the University of Pisa. He co-designed the actual city backbone based on single-mode optical fiber, wireless connections, ATM, and Gigabit Ethernet network transport. Additionally, he is a member of several national research projects focusing on networking, and the creator and maintainer of the "it" Usenet hierarchy. His interests include network management, traffic measurement, and network security. His home page is http://realta.unipi.it/~stefano/.

ntop's unique user interface allows administrators to immediately take advantage of ntop without the need to purchase and manage the client applications necessary for tools such as RMON or NeTraMet.