

◆ User-Level Billing and Accounting in IP Networks

Stephen M. Blott, Clifford E. Martin, Yuri J. Breitbart, José C. Brustoloni, Thomas R. Gramaglia, Henry F. Korth, David M. Kristol, Robert H. Liao, Eben L. Scanlon, and Avi Silberschatz

Internet protocol (IP) networks were originally designed by universities and government research agencies, not by network operators with commercial objectives. As a result, while aggregate accounting is possible for network management, detailed accounting at the level of individual users is often not possible. In today's networks, this limitation hinders the effective introduction of a variety of new IP-based services. Network operators lack the tools to monitor individual customers' use of network services, measure the type and quality of service that they receive, and ultimately establish service-level agreements and bill for those services. This paper presents an approach to accounting in IP networks that meets these needs based on a special-purpose network probe, which we term a NetCounter. The key functionality of a NetCounter is real-time, in-network correlation of network traffic with the individual users that generated it. This approach has several technical advantages over alternative approaches. Most importantly, the NetCounter achieves substantial in-network aggregation, reducing the volume of usage data generated by between two and four orders of magnitude (when compared to flow-logging systems). In addition, NetCounters capture usage data at the level of individual users and record detailed, user- and service-specific performance metrics.

Introduction

The use of the Internet has traditionally been free to end users, with the actual costs absorbed into the budgets of various research institutions, corporations, and government agencies. The delivery of data (that is, packets) has traditionally been accomplished exclusively on a best-effort basis with no guarantees of any specific quality of service. Although aggregate usage data is collected for network management,¹ there has been little or no requirement to collect usage data at the level of individual users.

Today's Internet increasingly deviates from this model. Service is provided by a variety of for-profit corporations, and the services carried increasingly

include time- or quality-sensitive data such as voice and video. Furthermore, the Internet model has been applied to the construction of virtual private networks with complex service-level agreements and private intranets that require accounting controls. As a consequence, commercial network operators now view the lack of detailed usage data for individual users as a serious shortcoming. For example, consumers' access to today's Internet in the United States is typically based on a flat monthly fee for unlimited access. Competition has tightened profit margins, and access providers are unable to differentiate their services or control their costs. To attack this problem, usage-based

pricing—where users are charged based on the amount and type of service—is seen by access providers as increasingly attractive (particularly for broadband access).

Changing technology is also creating the need for detailed accounting, most notably in the area of quality of service. Traditionally, IP networks have offered best-effort service only. However, today's networks increasingly offer value-added features such as guaranteed quality of service or differentiated services. These changes create two new accounting problems. First, there is the need to measure the level of service that individual users receive and to ensure that contracted service levels are met. Such accounting must be done at the level of individual users—for instance, monitoring the actual jitter or end-to-end latency on each voice or video call. Second, networks offering multiple service levels require more careful resource management and access control than do best-effort networks. Usage- and service-based pricing is one technique for controlling access to improved services.

An accounting solution for IP networks must solve three key problems:

- Handling the huge volume of events (either packets or flows),
- Associating usage with the users that generated it, and
- Providing detailed measurements of the service type and quality that individual users received.

Moreover, there can be an interesting and important tradeoff between the first two of these problems. In particular, some accounting systems reduce data volumes by aggregating within the network based on IP addresses. However, IP addresses are often assigned dynamically. As a result, these systems may inadvertently combine usage from two or more different users, thereby making it impossible to perform accounting at the level of individual users. This suggests that, in order to achieve effective aggregation within the network *and* retain detailed usage data for individual users, traffic must in fact be associated with users within the network prior to aggregating.

To address these problems, we developed a special-purpose, in-network device, which we call a *NetCounter*. The NetCounter is a dual-ported network

Panel 1. Abbreviations, Acronyms, and Terms

API—application programming interface
ATM—asynchronous transfer mode
BOOTP—bootstrap protocol
BPF—Berkeley Packet Filter
BSD—Berkeley Software Distribution
CPU—central processing unit
DHCP—dynamic host configuration protocol
DNS—data networking system
HTTP—hypertext transfer protocol
IETF—Internet Engineering Task Force
IPFW—IP firewall
IP—Internet protocol
LAN—local area network
Level 2—the network layer in the seven-layer OSI networking reference model
MAC—media access control
MTBF—mean time between failures
NFS—network file system
OC-3—optical carrier digital signal rate of 155 Mb/s in a SONET system
OC-12—optical carrier digital signal rate of 622 Mb/s in a SONET system
OSI—Open Systems Interconnection
PC—personal computer
PSTN—public switched telephone network
RADIUS—remote authentication dial-in user service
RAM—random access memory
RMON—remote monitoring
SMTP—simple mail transfer protocol
SNMP—simple network management protocol
SONET—synchronous optical network
TCP—transmission control protocol
ToS—type of service
UDP—user datagram protocol
VPN—virtual private network

device that, much like a bridge, is transparent to other devices at Level 2 and above. Transparency is important since it makes it possible to install and uninstall NetCounters without introducing transit subnets or reconfiguring adjacent switches and routers.² The key innovation in the NetCounter is the ability to associate packets with the individual users that generated them immediately within the network. This allows substantially more aggregation to be accomplished without the risk that user-level accounting detail will be lost.

NetCounters also record a number of novel measures of service type and quality, in addition to counts of packets, bytes, and flows.

The body of this paper is organized as follows. The next section sketches an IP mediation architecture currently being proposed by some vendors and explains the strengths and limitations of that architecture. The following section describes the NetCounter architecture in some detail, and a subsequent section evaluates the NetCounter's performance in terms of its impact on the performance of the network and the quantity of aggregation achieved. The last two sections compare our work with alternative approaches and present our conclusions.

Mediation in IP Networks and Problem Statement

Mediation systems collect usage data from network devices and systems and deliver that usage data to back-office applications (such as systems for billing, fraud management, or network planning). The BILLDATS® Data Manager is an example of a mediation product offered by Lucent Technologies.³ Mediation systems traditionally provide functionality in three main areas:

- *Interfacing to the network.* Traditional mediation systems support the multitude of—often proprietary—protocols and formats that define how usage data is exported by network devices. This avoids the need for back-office systems to communicate with the devices of the network directly.
- *Automated collection and delivery, and buffering.* Traditional mediation systems automate much of the collection and delivery process and provide temporary storage of usage data. As a result, in-network and back-office processing can be asynchronous.
- *Filtering, deduplication, and formatting.* Traditional mediation systems filter out data that is not pertinent to downstream processing and eliminate duplicate records. For example, nonbillable records can be filtered out even before they reach a back-office billing system.

Key Considerations for Mediation in IP Networks

Mediation in IP networks differs from mediation

in the PSTN in three main ways, since IP networks are characterized by:

- Substantial in-network aggregation to deal with huge data volumes,
- Greater use of open platforms, and
- Dynamic address assignment (which can make it difficult to identify individual users for accounting and billing purposes).

First, IP networks require substantial in-network aggregation. In the PSTN, the same granularity of accounting is used in the network and in the back office—namely, the telephone call. In IP networks, however, there may be a huge number of events within the network—far more than any back-office system either could or should process. Consider, for instance, the case of billing for a Web-hosting service. Web log files contain records corresponding, not to every page accessed, but to every hypertext transfer protocol (HTTP) transfer. A user accessing a single Web page commonly causes between ten and twenty log records to be generated. It is inconceivable that a billing system would produce a line item for each of these records. Rather, the mediation system must aggregate such logs, producing only summary data for back-office systems. For example, a billing system might receive, for each customer, only the total number of page hits and the total amount of data transferred, perhaps classified by the time of day.

Second, many IP services operate not on proprietary hardware and operating systems but on open, off-the-shelf platforms. Moreover, open formats are used for logging, often using textual representations. The use of open platforms makes it possible to develop “mediation probes”—that is, mediation software that executes on the service platform itself. Continuing the Web-hosting example above, aggregation of Web logs can be accomplished by mediation probes that run on the Web servers themselves. This approach allows only lower-volume, summarized data to be delivered to the mediation platform.

Third, IP addresses are often assigned dynamically, which can make it difficult to associate usage with individual users. In particular, protocols such as remote authentication dial-in user service (RADIUS) and dynamic host configuration protocol (DHCP)

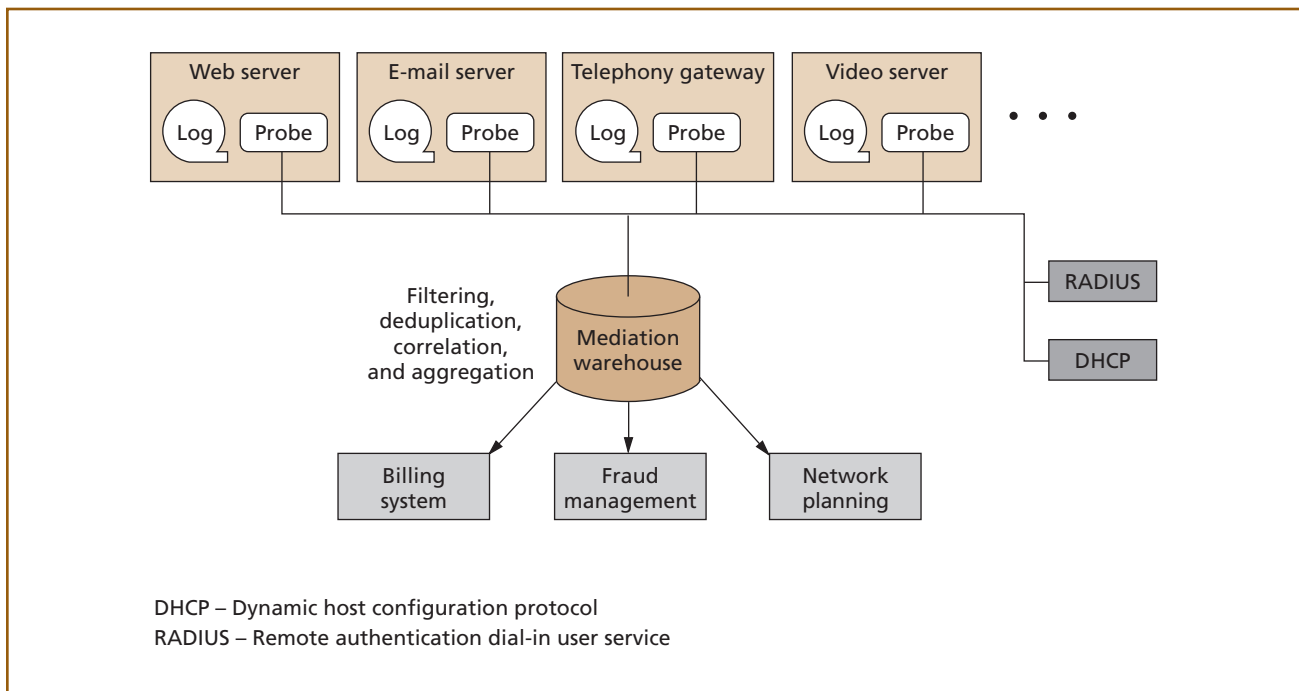


Figure 1.
A general mediation architecture for IP networks.

assign network addresses to users only for the time periods during which they are logged onto the network.^{4,5} As a result, IP addresses are associated with different users at different times. To overcome this difficulty, the authentication logs mapping IP addresses to individual users (from servers such as RADIUS and DHCP) must be incorporated into the mediation architecture. By correlating information from both usage and authentication logs, it is possible to map usage back to the individual users that generated it.

This type of correlation must usually be done before aggregation. To see why, consider the situation in which usage data generated for some IP address is aggregated for the period between 3:00 and 4:00. Assume that this address was assigned to one user from 3:00 until 3:23 and to another user from 3:27 until 4:00. In this case, the aggregation process discards information that is necessary in order to calculate individual users' network usage.

Based on these three considerations, an architecture for mediation in IP networks is sketched in **Figure 1**. The key features of this architecture are as follows. Usage data is collected by mediation probes

that run directly on service platforms themselves. These probes read usage data from system logs and (if necessary) perform filtering, correlation, and aggregation, before shipping usage data to the mediation warehouse. The mediation warehouse is a data-storage system, which may be either centralized or distributed. Further filtering, correlation, and aggregation might, if necessary, also be performed at the mediation warehouse. This architecture is broadly comparable with that used by XACCT Technologies⁶ and by Hewlett-Packard* in its Smart Internet Usage product.⁷ The "Related Work" section of this paper provides a discussion of these and other products.

Problem Statement: The Need for In-Network Aggregation

The mediation architecture shown in Figure 1 works well for services such as e-mail, Web hosting, voice, and video that are offered over an IP network. It does not account, however, for IP transport—that is, for raw use of the network itself.

Consider, for instance, the example illustrated in **Figure 2**. A cable operator provides IP access for a user. However, the user accesses a video server offered by another service provider. Video traffic is carried

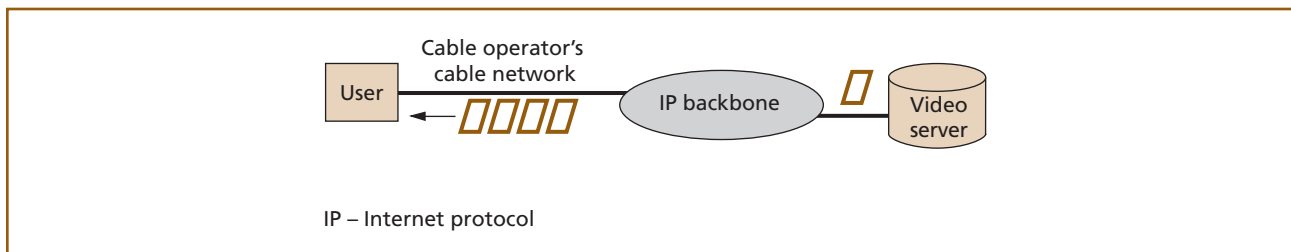


Figure 2.
An example of the need for in-network aggregation.

from the video server to the user over the cable operator's cable network.

In this scenario, the other service provider may bill the user for the video content that he/she received. Presumably, the video server generates a log of the content it delivers, and that log can be collected by a mediation probe and handled within a mediation architecture such as that discussed above. However, because video requires high-bandwidth transmission, the cable operator may bill the user for transport of the video data (especially if the user contracted for a guaranteed service level). Billing for IP transport—that is, for raw use of the IP network itself—does not fit easily into the mediation architecture sketched above for the following reasons:

- By almost any measure—the number of bytes, packets, or flows—IP networks generate huge volumes of data. For example, a gigabit router may handle up to 10,000 new flows every second. Therefore, in order to reduce data volumes to a manageable level, aggregation must be performed within the network.
- To associate traffic with users, usage data based on IP addresses must be correlated with authentication logs. Moreover, as discussed above, such correlation must be accomplished before aggregation. Hence, correlation, too, must be done within the network.
- Given the mediation architecture sketched above, this could be achieved through the use of a mediation probe running on some element within the network. However, the switches and routers that constitute today's IP network are generally based on proprietary hardware and software platforms. It is neither

possible nor desirable to run arbitrary third-party software on such devices.

To meet all these constraints simultaneously, it is necessary to have a device that correlates users with usage directly within the network. However, no existing network technology—including flow logging, simple network management protocol (SNMP), remote monitoring (RMON), and various types of traffic analyzers—can provide this capability.

In addition, the usage data captured by today's network devices includes only simple counts of the number of bytes, packets, and flows transmitted within each type of service. While these metrics are useful, we believe that more detailed accounting measures will be required for quality- and time-sensitive services. For example, consider the problem of accounting for a video call on an IP network. It is necessary to know the amount of data transferred and the duration of the call. However, it would also be valuable to record the level of jitter and the peak bandwidth consumed by the call. These metrics reflect the quality of service that the user received and may be relevant for either billing or verifying service-level agreements. These types of detailed usage metrics are not recorded within today's IP networks.

The NetCounter: An In-Network, User-Based Aggregator

This section describes the NetCounter, an extension of the mediation architecture sketched above for efficient accounting of network usage in an IP network.

A NetCounter is a special-purpose device that operates within an IP network, collecting usage data for individual users. This is illustrated in **Figure 3**. In particular, when users log onto an IP network, they are assigned an IP address. In the example, the IP

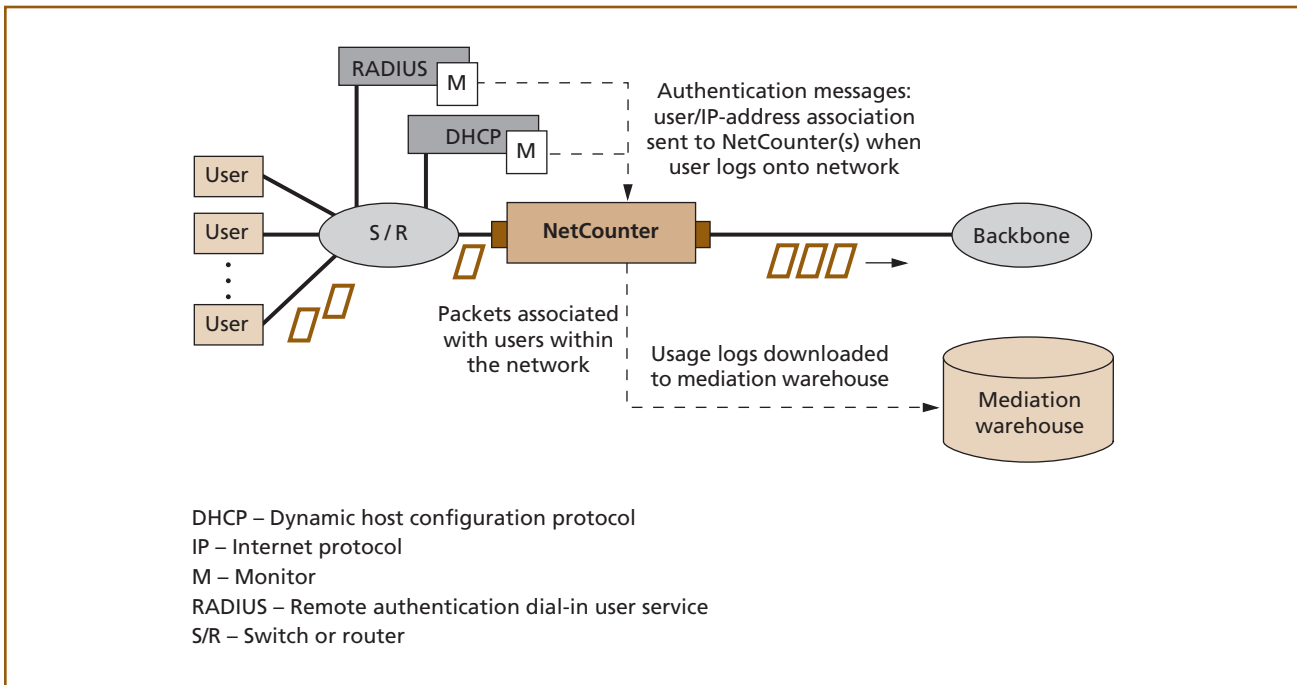


Figure 3.
NetCounter—a dual-ported accounting device within an IP network.

address is assigned when they register themselves with either a DHCP or a RADIUS server. The association of a user to an IP address is then sent immediately to the NetCounter, in what we term an *authentication message*. In the example, the NetCounter sits on the network between the users and the backbone. As users send and receive packets, the NetCounter associates each packet with a user. This allows the NetCounter to record highly aggregated yet detailed accounting data for each individual user.

The NetCounter is a network device with (at least) two network interfaces, as illustrated in Figure 3. NetCounters can operate in one of two modes. First, a Netcounter can operate as a bridge, as illustrated in Figure 3. In this mode, all packets arriving on one interface are transmitted on the other and vice versa. The NetCounter is transparent to other devices at Layer 2 and above; in the case of Figure 3, it appears as if the link goes directly from the switch/router to the backbone. This simplifies installation, since adjacent devices do not need to be reconfigured and no transit subnets are required.² In the second mode (which is not illustrated), a NetCounter can be configured to

accept data mirrored out of one or more switch or router ports. In this configuration, sufficient switch or router ports must be available and, when the NetCounter is installed, the switch or router must be reconfigured to mirror the required traffic to the NetCounter port(s).

NetCounters can communicate directly over their bridging interfaces. However, for security reasons, it is sometimes preferable to have a third dedicated interface for communication with downstream systems or a management station.

The basic usage data collected by NetCounters is the number of bytes, packets, and flows aggregated by user and service. Collection is based on a time interval. The time interval is configurable, but it is typically on the order of one minute to a few hours. When data ages, the aggregated usage data is downloaded to a mediation warehouse. Usage data is aggregated by user and service. The service recorded by a NetCounter is based on well-known port numbers. For example, an HTTP server customarily runs on transmission control protocol (TCP) port 80, while a network file system (NFS) server customarily runs on

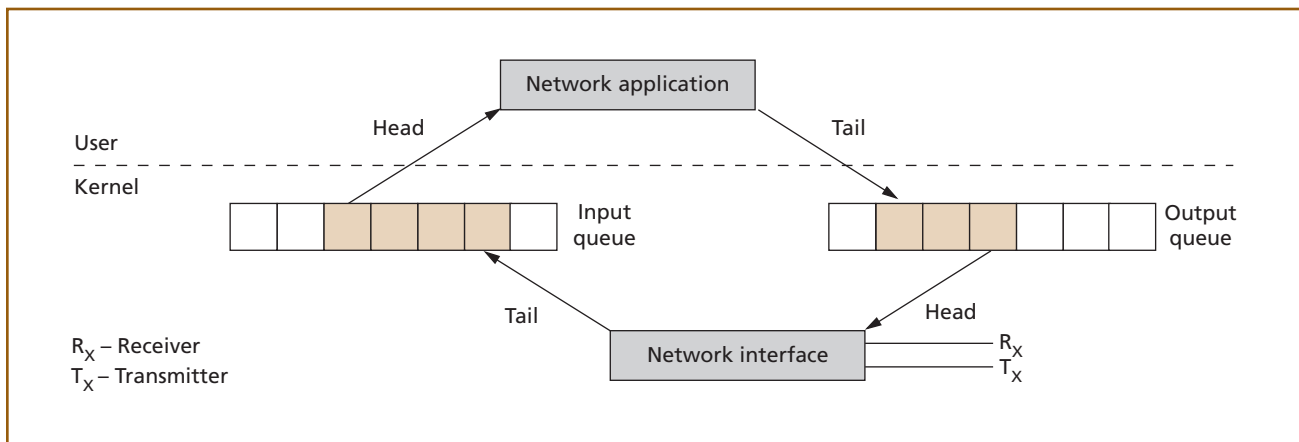


Figure 4.
The NetTap software architecture of the NetCounter.

user datagram protocol (UDP) port 2049. While this is a valuable indicator of the type of usage, it is not infallible. For this reason, NetCounters also capture additional usage metrics that describe the type and quality of service received (see below).

An important consideration is where NetCounters should be placed. The general rule is that at least one NetCounter must be placed on each network path for which accounting is required. A simple corollary is that no accounting data is collected for traffic that (only) traverses a path on which no NetCounter is placed. The goal of a commercial access provider is usually to monitor user access to its backbone. In this case, it generally suffices to install a single NetCounter on each uplink between a point of presence and the IP backbone.

NetCounter Implementation: Software Architecture

The NetCounter could be implemented in a variety of ways—namely, on a general-purpose computing platform, as line cards in a rack-mounted system, or by integration into switch or router packet processing. While we believe that integration with switches and routers may be the ultimate implementation of NetCounter, this method would prevent deployment of NetCounter technology in existing, multivendor networks. Our initial implementation was done as a bridge in a manner transparent to existing network elements, using a general-purpose computing platform with custom network interface cards. We expect that network capacities of OC-12 and higher will require a line-card implementation in which the NetCounter application is

executed on the network interface card itself.

In this paper, we describe our initial bridge-based implementation as a user-level application on a PC-based platform running the FreeBSD* operating system.^{8,9} Three factors motivated these choices. First, applications are usually easier to develop, debug, and maintain at user level than at kernel level. Second, the economies of scale of the PC market result in low PC costs and continuously improving performance. Third, FreeBSD incorporates protocol implementations that are widely regarded as de facto standards, offers state-of-the-art compilers and other development tools, and has a good track record for up-to-date peripheral support.

However, the decision to develop an application such as NetCounter at user level usually comes with a significant performance penalty. This penalty is caused by the inefficiency of FreeBSD's application programming interfaces (APIs), which may require a system call and one or more packet copies each time a network application receives or sends a packet.

We therefore added to FreeBSD a more efficient API, which we call the *NetTap* API.¹⁰ NetTap, illustrated in **Figure 4**, has the following main features:

- All packet buffers are mapped to both system and network applications (in our case, specifically, the NetCounter);
- Instead of passing *copies* of packets, system and network applications exchange *pointers* to packets; and

- System and network applications communicate asynchronously via a number of circular queues, thus avoiding system call overheads in most cases.

Because NetTap avoids packet copying and system calls, it imposes essentially no penalties for network processing done at the user level instead of the kernel level. Details of performance improvements due to the NetTap API are provided in the “Throughput” section.

NetCounter Implementation: Algorithm

The NetCounter follows the familiar input-process-output model, processing each packet as it arrives from the network. When it operates as a bridge, the NetCounter receives each packet on one of two interfaces and transmits that packet out the other one. Note that, with this model, the NetCounter can choose to drop a packet based on the processing. Since the NetCounter uses the NetTap architecture, the input part of the program requires simply obtaining a pointer to the next packet from the NetTap API’s input queues. Similarly, output requires simply passing a pointer to the NetTap API’s output queues.

The processing part of the program follows several steps:

1. *Classify the packet.* Determine what type of packet it is (for example, TCP or UDP), and extract information from the protocol headers.
2. *Look for in-packet user identification.* Check the packet to determine whether it is an authentication message containing information to associate a new IP address or a TCP or UDP data stream with a particular user. If user identification is present, create an entry in the user identification table.
3. *Identify the (UDP or TCP) packet’s “service.”* For a UDP or TCP packet, determine whether the source or destination port matches one that has been configured to represent a service. For example, the default HTTP port is 80. If a TCP packet has a destination port of 80, it is assumed to be part of an HTTP session.
4. *Identify the users.* Based on information in the user identification table, try to identify the user that sent the packet and/or the user to whom the packet is addressed. If both are identified, then

the packet is accounted against both the sender *and* the receiver.

5. *Accumulate usage statistics.* A statistics table is indexed by both the service and the user. If either the sender or the receiver is identified (or if both are identified), accumulate in the statistics table the counts of the number of bytes, packets, and flows generated for that user.

A flow cache is used to accelerate these processing steps for the case in which similar packets arrive close together. In particular, the flow cache allows steps 3 and 4 above to be skipped entirely and step 5 to be simplified in that the flow cache contains pointers directly into the statistics table. That is, no further table look-up is required for hits in the flow cache. As entries in the statistics table age, they are written to a log file. Log processing is done in small incremental steps and only when no packets are waiting to be processed. In this way, log processing does not interfere with the performance-critical task—that of actually collecting usage data and forwarding packets.

The NetCounter determines services by tracking well-known TCP and UDP port numbers. There are several limitations to this approach. In particular, tunneling techniques, such as those used by virtual private networks (VPNs), hide port numbers within encrypted packet payloads. Moreover, the use of well-known ports is merely a convention that cannot be enforced or relied upon. To address these difficulties, the NetCounter also records a variety of other metrics that reflect the type and quality of service that the user received. These include measures of the jitter, the active bit rate received, and the difference between the requested and carried loads.

NetCounter Implementation: Reliability

As previously stated, NetCounter can be operated in either bridge mode or mirrored mode. In particular, in bridge mode and in the absence of the reliability features we built into the NetCounter, a failure would cause failure of the network link itself. This section summarizes the steps we took to ensure reliability for NetCounters operating in active mode and avoid the failure of network links.

Hardware measures. We designed PC network cards with two network interfaces, a watchdog

timer, and a bypass switch, as shown in **Figure 5**.

The bypass mechanism switch automatically bridges traffic on the link if the device has no power. The watchdog timer controls the bypass switch. If the watchdog timer counts down to zero before receiving a “heartbeat” signal, the card automatically enters bypass mode. The watchdog timer is used primarily to handle software failures, whereas the switch guards against loss of power.

Most failures of hardware components will cause the NetCounter to fail. The most likely components to fail are the power supply and the hard drive, in which case the system would not reboot after failure. However, these failures would not cause a loss of network connectivity, merely a loss of data monitoring.

The NetCounter uses a solid-state disk drive that has a higher mean time between failures (MTBF) than typical magnetic disk drives. The power supply in the NetCounter is a redundant, hot-swappable unit. Should one of the supplies fail, the other supply will provide power to the NetCounter and the failed supply can be removed without shutting down the system. These features ensure that even the loss of data monitoring is minimized.

Software measures. If the network monitoring software fails, the NetCounter operating system automatically fails-over to a mode whereby packets are bridged between interfaces but usage data is not collected. This ensures that network connectivity is not compromised.

In the event of an operating system failure, the heartbeat signal would not be sent to the hardware watchdog timer, and thus the NetCounter would automatically bridge between interfaces. The operating system should then automatically reboot and—after about two minutes—resume collecting usage data.

Associating Users with IP Addresses

NetCounters associate packets with users through a number of mechanisms, because no one method will work in every situation. For some protocols, user identification information is encoded within the packets’ payloads: examples include NFS and simple mail transfer protocol (SMTP). In general, however, the association of packets with users is derived from the information contained in packets’ headers. In particu-

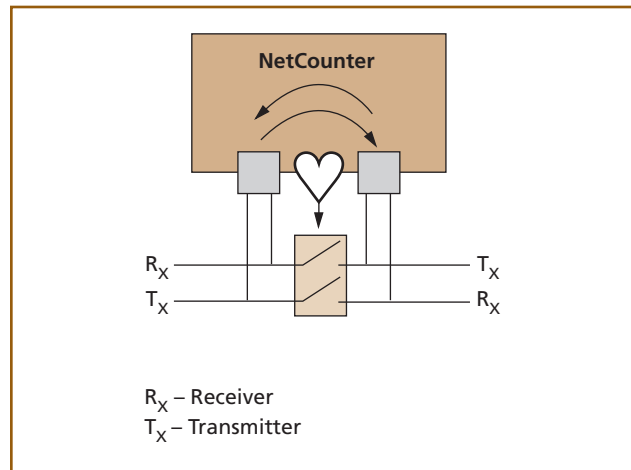


Figure 5.
NetCounter hardware fail-safe mode.

lar, NetCounters can use any combination of the following information: source and destination IP addresses, optionally masked; the IP protocol number (TCP or UDP); and source and destination TCP or UDP port numbers. For dial-up and broadband access, usually the IP address suffices to identify the user.

NetCounters receive authentication information from a number of sources:

- *Provisioned sources.* This case is most commonly used for dedicated access, if accounting is to be done at the level of the subnetwork or possibly in situations where IP addresses are fixed.
- *Off-the-wire sources.* In this case, authentication information is derived by analyzing the contents of critical authentication packets. To use DHCP as an example, consider the following. Assume that users have been assigned media access control (MAC) addresses, but IP addresses are assigned dynamically via DHCP. If the NetCounter is on the path between the users and the DHCP servers, then the NetCounter can intercept DHCP reply packets, analyze their contents, and use the information they contain to derive a mapping from IP addresses to MAC addresses. Since MAC addresses are statically assigned to users, a MAC address suffices to identify the individual user.
- *Authentication monitors.* Under this approach, NetCounters receive authentication informa-

tion from authentication monitors. These are software components running on and monitoring authentication systems such as DHCP and RADIUS servers. This approach is described in more detail below.

While the off-the-wire method is attractive for its simplicity, it has limitations. It relies on the topology of the network being such that authentication packets can be guaranteed to flow through a NetCounter. There is also a requirement that IP addresses not be shared by multiple users simultaneously (as in time-sharing systems). In addition, this approach is less secure than the authentication monitor approach described below.

Authentication monitors operate as follows. Each authenticator requires a small software component—denoted “M” for “monitor” in Figure 3—that monitors the authenticator and generates a real-time authentication feed to the NetCounters. For example, as soon as a user logs in at a RADIUS server, a message is dispatched to all relevant NetCounters providing the information necessary to identify that user’s packets on the network.

We developed several authentication monitors for different network environments, which we describe in some detail below. In general, authentication messages are broadcast using the bootstrap protocol (BOOTP)¹¹ in a manner much like the approach taken by DHCP.⁵ This enables the protocol to take advantage of BOOTP relay agents to pass messages across subnets.

Log monitors. We implemented authentication monitors for RADIUS, DHCP, and Windows NT* systems, all based on the principle of monitoring the server logs. In each case, a simple application was developed that reads the authenticator’s log file and sends out authentication messages to all NetCounters whenever a new authentication record appears. This works quite satisfactorily if records are logged in real time (which is the case for all the servers that we analyzed). An alternative to this approach, depending on the type of server, would be to develop a plug-in that is executed by the server itself at authentication time.

UNIX variants. UNIX* systems present a different type of problem since they are typically multiuser systems. Simply having the system’s IP address is insuffi-

cient to identify an individual user. In this case, the authentication monitor and the NetCounter also use the port number to identify the user. We developed a mechanism for TCP and most UDP sessions whereby a monitor on the shared system issues an authentication message exactly at the point at which each socket is created. This monitor is efficient in that it has only a marginal impact on performance. It is also practical in that neither applications nor the operating system must be changed in any way. Moreover, authentication messages are sent immediately before users’ processes send data over the socket. Thus, packets on the network can be authenticated in real time. Our method works on any UNIX system that supports dynamic linking and involves generating a new shared library that acts as a wrapper for the standard Berkeley Systems* socket calls. This library is automatically linked with standard network applications such as `ftp`, `telnet`, and `xterm`, and browsers such as Netscape Communicator* and Microsoft* Internet Explorer.

Performance

We measured the performance of our NetCounter implementation in terms of its impact on network traffic and the amount of aggregation that it achieves. We include only a summary of our results here. (Complete results are presented by Blott, Brustoloni, and Martin.¹⁰)

In our performance experiments, the applications ran on a PC with a 333-MHz Intel Celeron* CPU (SPECint95 rating of 12.3) with 32-KB L1 cache (16-KB instruction plus 16-KB data) and 128-KB L2 cache, 64-MB RAM, 80-MB SanDisk* Flashdrive, and two Intel Ether Express Pro 10/100 Ethernet adapters with the Intel 82559 chipset. The software consisted of the FreeBSD 3.2 operating system augmented with the NetTap API and running the NetCounter application. We measured throughput and latency using Netcom Systems’ Smartbits* SMB-2000 performance analyzer outfitted with two ML-7710 10/100 Layer 3 cards configured for full-duplex fast Ethernet operation. This analyzer follows the benchmarking methodology proposed by the Internet Engineering Task Force (IETF).¹²

Latency

The NetCounter increased packet latency by between 50 and 60 μ s, depending on the packet size; within this range, latency was slightly lower for smaller packets and higher for larger packets. We note that this is of the same order as (or slightly less than) the latency introduced by an Ethernet switch and that it is about 1/4,000th of the 200-ms end-to-end latency generally considered acceptable for IP telephony.

Throughput

We measured throughput as a function of packet size. Note that network links have limited bandwidth, and the maximum number of packets that a link can carry decreases as the packet size increases. Our throughput measurements correspond to the maximum sustained rates at which no packets are dropped in a 10-second interval. We present results for three cases. *Bridging kernel-level* is a reference bridge test that simply forwards packets between interfaces directly within the kernel. *Bridging on NetTap* is a reference bridge test with packet forwarding performed by an application using the NetTap interface. (Recall that the NetTap interface is an operating system API for efficient packet handling.) *NetCounter on NetTap* is the complete NetCounter implementation. We also provide throughput data for two standard Berkeley Software Distribution (BSD) network packages, namely, the Berkeley Packet Filter (BPF)¹³ and divert sockets (a feature of the FreeBSD IPFW firewall).

Our results are presented in **Figure 6**. First, we note that the two off-the-shelf mechanisms perform extremely poorly when compared with the experiments using the NetTap API. The critical experiment is that for the NetCounter (NetCounter on NetTap). For this experiment, for packets that are 384 bytes or longer, NetTap enables NetCounter to support the maximum throughput that the link can carry. However, for shorter packets, the per-packet processing costs of the NetCounter algorithms impact performance. The bridging results show that the NetTap interface completely eliminates the usual overhead associated with processing packets within user-level applications, as opposed to within the kernel. In fact, because of the additional buffering provided by the NetTap queues, throughput is

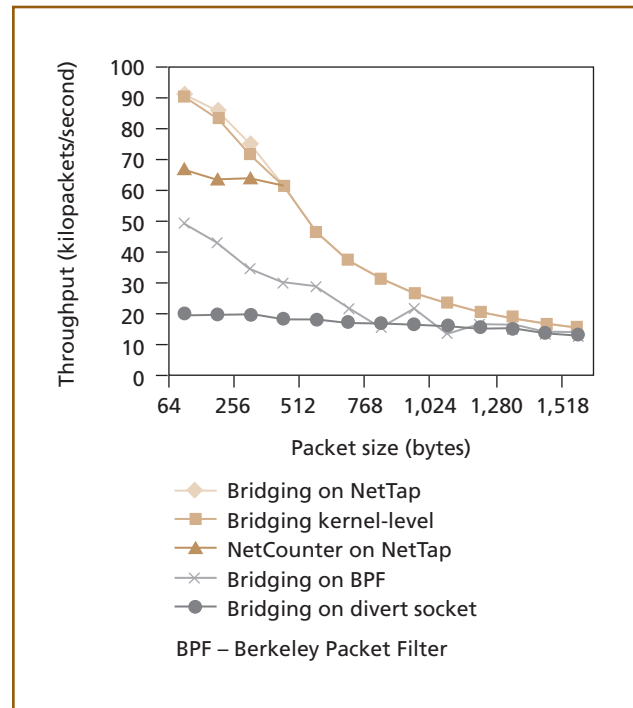


Figure 6. NetTap and NetCounter throughput results.

marginally improved when bridging packets with the NetTap API, when compared with bridging packets directly within the kernel.

Amount of In-Network Aggregation

The amount of aggregation achieved within the NetCounter has a substantial effect upon downstream systems. It affects the overhead in terms of the amount of bandwidth consumed downloading usage data to the mediation warehouse, the update rate that must be supported by that warehouse, and also the storage requirements at the warehouse.

To assess the degree of aggregation achieved within the NetCounter, we measured aggregation based on packet trace data from within our own office local area network (LAN). With the NetCounter, usage data is handled using a combination of two configurable modes. In the first mode, when an entry in the statistics table is created within the NetCounter, it lives for a fixed time interval during which usage is aggregated into that entry. At the end of the interval, an output record is generated and the table entry is deleted. This is a form of real-time operation in which downstream systems have guarantees as to the rate at

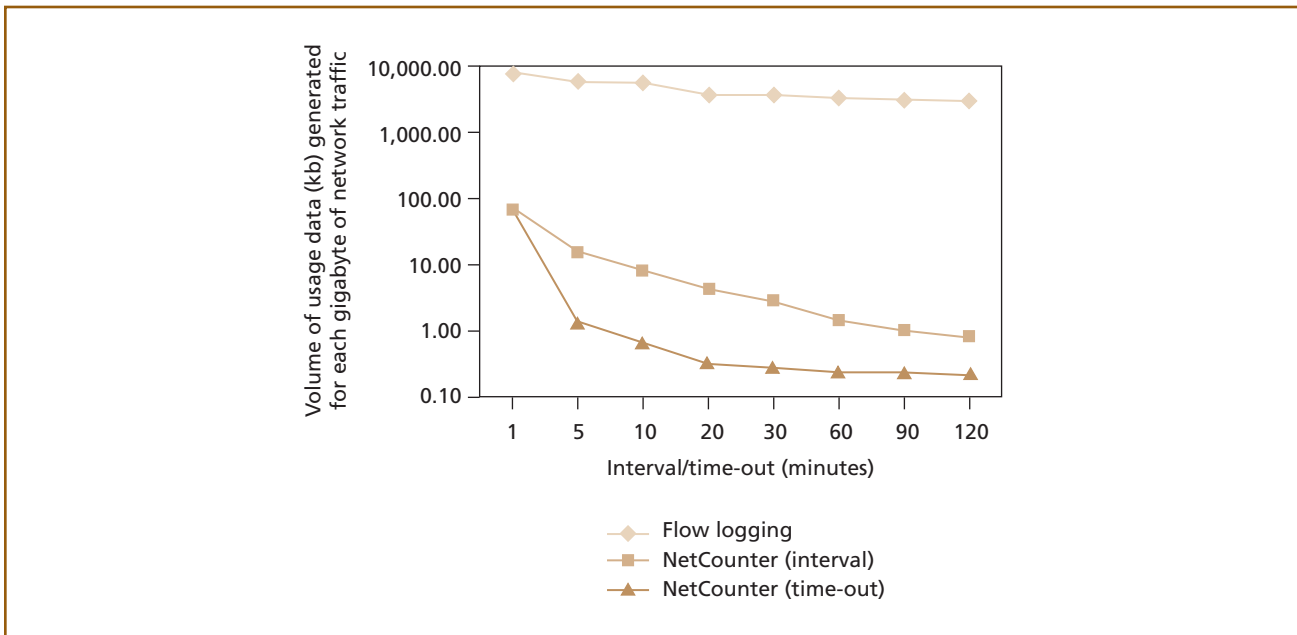


Figure 7.
Kilobytes of usage data generated for each gigabyte of network traffic.

which usage data is exported. With the second mode, time-outs are used to age entries in the statistics table. In particular, entries are purged and usage records are generated only when they have remained unused, and they time out. This mode is not considered to be “real time” since there is no bound on the amount of time between usage occurring and the corresponding usage records being generated. In practice, a combination of these two modes is usually required.

We measured aggregation for these two alternative NetCounter modes and also—as a reference—for flow logging. A flow is a sequence of packets within a protocol from one IP address and port to another. Many network devices maintain flow tables for switching and routing purposes, and some provide the ability to export usage logs aggregated into flows (recording the number of bytes and packets transferred with each flow). For UDP traffic, flow logs are generated based on a time-out.

The aggregation metric that we use is the number of kilobytes of usage data generated for each gigabyte of data carried on the network. In our experiments, we varied the collection interval and/or the time-out interval. Our results are presented in **Figure 7**. Three aspects of these results deserve comment:

- The amount of in-network aggregation achieved by the NetCounter is between two and four orders of magnitude greater than that achieved by flow logging. This improvement is substantial, making it feasible to collect usage data for high-bandwidth and large networks where flow logging is impractical.
- Our approach defines a design space that does not exist for flow logging. In particular, for flow logging, the amount of aggregation is only marginally affected by the collection interval. This is because most flows are short, so longer collection intervals simply include more unrelated flows that cannot be aggregated. However, with the NetCounter approach, increasing the collection interval generally further improves the amount of aggregation (by up to two orders of magnitude). That is, a design space exists within which an engineer can select a collection interval to achieve a target amount of aggregation.
- The cost of having real-time guarantees—that is, the difference between interval and time-out-based aggregation—can be up to an order of magnitude.

Related Work

One of the most widely available sources of usage data in an IP network is the simple network management protocol (SNMP).¹ SNMP is most commonly used to collect usage data at the aggregate level, not at the level of individual users. Thus, it does not meet our requirements here.

Many routers and switches, including those from Cisco Systems,* can capture statistics such as number of bytes and packets per flow. A packet's flow is identified by the packet header's source and destination IP addresses, the protocol, the source and destination port numbers and type of service (ToS), and the interface on which the packet was received. Flows are unidirectional. Cisco calls the capture of flow statistics *NetFlow services*.¹⁴ Configured routers and switches periodically export captured flow statistics to a workstation running the *NetCollector* application. NetCollector filters and aggregates flow statistics and stores records usable for billing and network planning.

The major shortcoming of NetFlow and flow logging in general is the massive volume of usage data generated. We have shown that flow logging generates up to four orders of magnitude more usage data than is generated by the NetCounter.

Another difficulty with NetFlow services is that they do not associate flows with users. In many access networks, users are assigned IP addresses dynamically, and therefore the same IP address corresponds at different times to different users. In such situations, NetCollector's simple aggregation scheme may not work. XACCT and Hewlett-Packard have independently offered alternative mediation platforms—XACCTusage⁶ and HP Smart Internet Usage,⁷ respectively—that overcome this NetCollector shortcoming by correlating flow usage with information from DHCP, RADIUS, and data networking system (DNS) servers. Such off-line correlation permits network usage to be correctly charged to the respective users, even when IP address assignment is dynamic. Our NetCounter achieves similar benefits by correlating packets and users on-line, directly at the point of capture of usage records.

In addition, flow logs may not capture necessary accounting information. For example, protocols such

as SMTP and NFS carry user information within application-layer data. Such information is necessary to individualize each user's usage, but it is not captured by NetFlow statistics. NetCounter, on the contrary, takes such information into account when it processes each packet. Moreover, for quality of service, flow logs contain information describing only the service quality that was requested, not the quality that was received.

NARUS* Incorporated is a start-up company that offers a data-collection and mediation solution for IP networks based on a network probe.¹⁵ NARUS's network probe differs from the NetCounter in three respects. First, the NARUS probe supports only the mirrored mode, not both the mirrored and bridged modes offered by the NetCounter. Second, the NARUS probe collects traffic on a single interface only. Thus, since most networks are full duplex, it cannot capture all the traffic on a highly utilized link. Third, because the NARUS probe does not correlate packets with users within the network, it achieves substantially less aggregation than the NetCounter achieves. Specifically, the NARUS probe generates about 10 megabytes of usage data for each gigabyte of network traffic.¹⁵ This is in excess of four orders of magnitude more usage data than that generated by the NetCounter.

Conclusion

Operators of IP networks lack the detailed, user-level accounting data that is commonly available in the PSTN. The lack of this information limits their ability to roll out new IP-based services, control their costs through usage-based billing, and prevent abuse of their networks (particularly in the case of broadband access providers). Moreover, detailed, user-level accounting is necessary to support future networks with differentiated services and quality-of-service guarantees.

In this paper, we described an IP accounting architecture and the NetCounter—a new, in-network device that captures network usage data aggregated by user and service. Unlike existing systems, NetCounters associate each packet with a user immediately within the network. This capability allows NetCounters to

perform substantial amounts of aggregation within the network, greatly reducing the network bandwidth, storage space, and CPU time necessary for processing usage records. In particular, the NetCounter generates between two and four orders of magnitude less usage data than that generated by flow-logging systems. Our experiments also show that our initial PC-based implementation is fast enough to keep up with fully utilized, full-duplex fast Ethernet links. We speculate that the same architecture will also suffice for OC-3 networks. Future NetCounter implementations will support faster links by using custom hardware or by integration into router or switch devices. We are currently developing several interface cards to support asynchronous transfer mode (ATM) and packet-over-SONET (synchronous optical network) on higher-bandwidth optical links.

Our initial NetCounter implementation uses NetTap—a new PC- and FreeBSD-based platform. The NetTap API eliminates copying and system call overheads, giving user-level network applications essentially the same performance as that of kernel-level ones. We therefore implemented NetCounter as a user-level application, which simplifies debugging, modifications, and maintenance. NetTap can be configured as a bridge, allowing easy NetCounter installation in existing networks without any subnet reconfiguration. The NetCounter features a watchdog timer and a bypass switch that preserve link connectivity in case of NetTap failure.

*Trademarks

Berkeley Systems is a registered trademark of Berkeley Systems, Inc.

Cisco Systems is a registered trademark of Cisco Systems, Inc.

FreeBSD is a registered trademark of FreeBSD, Inc., and Walnut Creek CDROM, Inc.

Hewlett-Packard is a registered trademark of Hewlett-Packard Company.

Intel and Celeron are registered trademarks of Intel Corporation.

Microsoft and Windows NT are registered trademarks of Microsoft Corporation.

NARUS is a registered trademark of NARUS, Inc.

Netcom is a registered trademark and Smartbits is a trademark of Netcom On-Line Communication Services, Inc.

Netscape is a registered trademark and Communicator is a trademark of Netscape Communications Corporation.

SanDisk is a registered trademark of SanDisk Corporation.

UNIX is a registered trademark of The Open Group.

XACCT Technologies is a registered trademark of XACCT Technologies.

References

1. W. Stallings, *SNMP, SNMPv2, and RMON*, Addison-Wesley, Reading, Mass., 1996.
2. T. A. Limoncelli, "Tricks You Can Do If Your Firewall Is a Bridge," *Proc. 1st USENIX Conf. on Network Administration (NETA '99)*, Santa Clara, Calif., Apr. 7–10, 1999, pp. 47–55.
3. <http://www.lucent.com/OS/billdatsdm.html>
4. C. Rigney, A. Rubens, W. Simpson, and S. Willens, "Remote Authentication Dial-In User Service (RADIUS)," IETF RFC 2138, Apr. 1997, <http://www.ietf.org/rfc/rfc2138.txt>
5. R. Droms, "Dynamic Host Configuration Protocol," IETF RFC 1531, Oct. 1993, <http://www.ietf.org/rfc/rfc1531.txt>
6. <http://www.xacct.com>
7. <http://www.hp.com/smartinternet>
8. G. Lehey, *The Complete FreeBSD*, 2nd ed., Walnut Creek CDROM Books, Walnut Creek, Calif., 1997.
9. M. McKusick, K. Bostic, M. Karels, and J. Quarterman, *The Design and Implementation of the 4.4 BSD Operating System*, Addison-Wesley, Reading, Mass., 1996.
10. S. Blott, J. Brustoloni, and C. Martin, "NetTap: An Efficient and Reliable PC-Based Platform for Network Programming," *Proc. IEEE OPENARCH 2000*, Tel Aviv, Israel (forthcoming Mar. 26–27, 2000).
11. B. Croft and J. Gilmore, "Bootstrap Protocol (BOOTP)," IETF RFC 951, Sept. 1985, <http://www.ietf.org/rfc/rfc0951.txt>
12. S. Bradner and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices," IETF RFC 1944, May 1996, <http://www.ietf.org/rfc/rfc1944.txt>
13. S. McCanne and V. Jacobson, "The BSD Packet Filter: A New Architecture for User-Level Packet Capture," *Proc. Winter 1993 USENIX Conf.*, San Diego, Calif., Jan. 25–29, 1993, pp. 259–269.
14. <http://www.cisco.com>
15. <http://www.narus.com>

(Manuscript approved December 1999)

STEPHEN M. BLOTT, a member of technical staff in the Network Systems Research Department at Bell Labs in Murray Hill, New Jersey, earned B.S. and Ph.D. degrees in computing science from the University of Glasgow in Scotland. Dr. Blott conducts research on database and network systems, particularly network accounting, network monitoring, quality of service, and billing.



CLIFFORD E. MARTIN is a member of technical staff in the Information Sciences Research Department at Bell Labs in Murray Hill, New Jersey, where he is currently working on several network management projects. His previous research projects included the CineBlitz™ multimedia file system and interactive television.



YURI J. BREITBART, a member of technical staff in the Internet Management Research Department at Bell Labs in Murray Hill, New Jersey, holds a D.Sc. degree in computer science from Technion, Israel Technological Institute, in Haifa. Dr. Breitbart has worked on Web data management, heterogeneous database management, and database replication and is currently working on Internet network topology, traffic monitoring, and fault management.



JOSÉ C. BRUSTOLONI, a member of technical staff in the Network Systems Research Department at Bell Labs in Murray Hill, New Jersey, holds a B.Sc. in electronics engineering from the Instituto Tecnológico de Aeronáutica in São José dos Campos, Brazil, an M.Sc. in electrical engineering from the Universidade de São Paulo in Brazil, and a Ph.D. in computer science from Carnegie Mellon University in Pittsburgh, Pennsylvania. His research interests include security in access networks, quality of service and billing, programmable networks, embedded systems, and protocol performance.



THOMAS R. GRAMAGLIA is currently director of Consulting Services at Lucent's Kenan Systems subsidiary in Cambridge, Massachusetts. He was formerly director of the Internet Business Unit, where he developed and executed the unit's corporate strategy in the Internet and electronic commerce market. Mr. Gramaglia holds a B.S. in physics from Yale University in New Haven, Connecticut, an M.S. in engineering from Stanford University in Palo Alto, California, and an M.B.A. from the Harvard Graduate School of Business Administration in Boston, Massachusetts.



HENRY F. KORTH, head of the Database Principles Research Department at Bell Labs in Murray Hill, New Jersey, holds a B.S. in mathematics from Williams College in Williamstown, Massachusetts, and M.S.E., M.A., and Ph.D. degrees in computer science from Princeton University in New Jersey. Dr. Korth is responsible for high-performance database systems, transaction processing, data replication, billing systems, and data warehousing.



DAVID M. KRISTOL, a member of technical staff in the Network Systems Research Department at Bell Labs in Murray Hill, New Jersey, holds a B.A. and a B.S. in electrical engineering from the University of Pennsylvania in Philadelphia and an M.S. and an M.E. in applied mathematics from Harvard University in Cambridge, Massachusetts. His research interests include Internet privacy and security.



ROBERT H. LIAO, a member of technical staff in the Development Department at Lucent's Kenan Systems subsidiary in Cambridge, Massachusetts, is working on software development for the Arbor Internet business unit. He holds a B.A. degree in economics and a J.D. degree from Harvard University in Cambridge, Massachusetts.



EBEN L. SCANLON is a member of technical staff at Lucent's Kenan Systems subsidiary in Cambridge, Massachusetts. He holds a B.A. degree in applied mathematics from Harvard College, also in Cambridge, Massachusetts. Mr. Scanlon, the co-director of the Research Enablement Center, is working on several Internet-related research projects.



AVI SILBERSCHATZ is the executive director of the Information Sciences Research Center at Bell Labs in Murray Hill, New Jersey. He holds a Ph.D. degree in computer science from the State University of New York at Stony Brook and is both an ACM and an IEEE Fellow. His research interests include operating systems, database systems, distributed systems, and real-time systems. Dr. Silberschatz directs work on operating systems and high-performance databases. ♦

