♦ Monitoring Infrastructure for Converged Networks and Services

Shipra Agrawal, C. N. Kanthi, K. V. M. Naidu, Jeyashankher Ramamirtham, Rajeev Rastogi, Scott Satkin, and Anand Srinivasan

Network convergence is enabling service providers to deploy a wide range of services such as Voice over Internet Protocol (VoIP), Internet Protocol television (IPTV), and push-to-talk on the same underlying IP networks. Each service has unique performance requirements from the network, and IP networks have not been designed to satisfy these diverse requirements easily. These requirements drive the need for a robust, scalable, and easy-to-use network management platform that enables service providers to monitor and manage their networks to provide the necessary quality, availability, and security. In this paper, we describe monitoring mechanisms that give service providers critical information on the performance of their networks at a per-user, per-service granularity in real time. This allows the service providers to ensure that their networks adequately satisfy the requirements of the various services. We present various methods to acquire data, which can be analyzed to determine the performance of the network. This platform enables service providers to offer carrier grade services over their converged networks, giving their customers a high-quality experience. © 2007 Alcatel-Lucent.

Introduction

Service providers all over the world have started deploying services such as Voice over Internet Protocol (VoIP) and Internet Protocol television (IPTV) on their IP-based networks to increase their revenues. Additionally, deploying new services over an IP network offers a reduction in their capital and operational expenditure. Emerging paradigms such as the IP Multimedia Subsystem (IMS) [5] allow service providers easily to deploy new services on their IP-based networks. A converged network demands a wide range of requirements from the underlying network depending on the services. For example, VoIP places stringent requirements on the delay, loss, and jitter performance and requires high availability from the network (five nines availability); however, it has relatively low bandwidth requirements [20]. On the other hand, streaming applications are more tolerant of delay but generally require higher bandwidth from the network. These requirements drive the need for a robust, scalable, and easy-to-use network management platform, which enables service providers to monitor and manage their networks to provide the necessary quality, availability, and security.

Bell Labs Technical Journal 12(2), 63–78 (2007) © 2007 Alcatel-Lucent. Published by Wiley Periodicals, Inc. Published online in Wiley InterScience (www.interscience.wiley.com). • DOI: 10.1002/bltj.20236

Traditionally, IP networks have been managed by measuring aggregate parameters, such as link utilization and packet losses, over interfaces of routers or other network elements. While this is sufficient to manage best-effort services, managing new services based on voice and video that have diverse requirements requires measurements of finer granularity. In this paper, we describe a next-generation monitoring infrastructure that offers service providers critical information on the performance of their networks at a per-user, per-service granularity in real time. This allows service providers to ensure that their networks adequately satisfy the requirements of the various services. We present various methods to acquire data that can be analyzed to determine the performance of the network. Data acquisition methods include active and passive probes, software agents for mobile devices, and flow information from IP routers. This infrastructure enables service providers to offer carrier grade services over their converged networks, giving their customers a high-quality experience.

Panel 1. Abbreviations, Acronyms, and Terms

API—Application programming interface
BA—Bangalore machine
CDMA—Code division multiple access
CPU—Central processing unit
DAG—Data acquisition and generation
DDoS—Distributed denial of service
DNS—Domain name system
DoS—Denial of service
GPS—Global Positioning System
HTTP—Hypertext Transfer Protocol
IMS—IP Multimedia Subsystem
IP—Internet Protocol
IPTV—Internet Protocol television
ISP—Internet service provider
ITU—International Telecommunication Union
MH—Murray Hill machine
MOS—Mean opinion score
NIC—Network interface card
NOC—Network operations center
NTP—Network Time Protocol
PC—Personal computer
PESQ—Perceptual evaluation of speech quality
PPP—Point-to-Point Protocol
PSTN—Public switched telephone network
RF—Radio frequency
RTP—Real Time Transport Protocol
SIP—Session Initiation Protocol
SLA—Service level agreement
SNMP—Simple Network Management Protocol
TCP—Transmission Control Protocol
ToS—Type of service
UDP—User Datagram Protocol
URL—Uniform resource locator
VoIP—Voice over IP
XML—Extensible Markup Language

Monitoring Infrastructure for Converged Networks

Our proposed monitoring infrastructure supports three primary functions: service assurance, traffic profiling, and fault detection and diagnosis. By monitoring the performance of various services on the underlying network, the platform must be able to detect service quality degradations and identify the cause of the problems. This information can then be used by the service provider to take remedial actions, minimizing the impact of degradations on the quality of user experience.

The broad set of requirements of a monitoring platform for a converged network can be summarized as follows:

- *Extensibility.* As new services are deployed on the network, it should be possible to easily and seamlessly deploy new monitoring mechanisms for these services.
- *Scalability.* Growing link speeds and the corresponding increase in the amount of information that must be processed to deduce the performance of the network at these rates place enormous stress on the management system. The monitoring platform must be able to handle increasing network speeds and a large number of devices in the network. As we discuss later, scalability can be achieved by reducing the information collected using efficient filtering, sampling, and aggregation algorithms.
- *Real time operation.* For many monitoring applications, real time reports on the performance of the network are essential to allow timely remedial

action by the service provider. This requires the monitoring platform to support continuous and real time mechanisms that detect problems in the network as they happen.

- *Granularity.* Each service utilizes a number of network protocols (e.g., Session Initiation Protocol (SIP), Hypertext Transfer Protocol (HTTP), Domain Name System (DNS)), and monitoring the performance of a service requires capturing the performance of each component protocol. This lets the service provider easily isolate the root cause of degradation. For instance, call setup times of a VoIP call can be excessive for a variety of reasons such as an overloaded SIP proxy, overloaded routers, or high loss rates in the network. The monitoring mechanisms must be fine-grained in order to make this distinction.
- *Diversity.* A converged network has a large number of network elements from multiple vendors, protocols, and applications pieced together to provide the user with the "bundle" of services. Consequently, any monitoring platform needs to support this diversity.
- *Low cost.* Finally, the cost of deploying and operating the monitoring infrastructure must be low to provide value for service providers. This implies that the system must use the least amount of computing, storage, and communication resources.

Figure 1 presents a proposed monitoring infrastructure for converged networks. A converged network typically consists of the core IP network that carries the network traffic, a session layer that handles the establishment of sessions between end hosts (e.g., SIP [21]), and an application layer that hosts application servers and handles application protocols. Monitoring this network requires monitoring network elements as well as protocols at each layer.

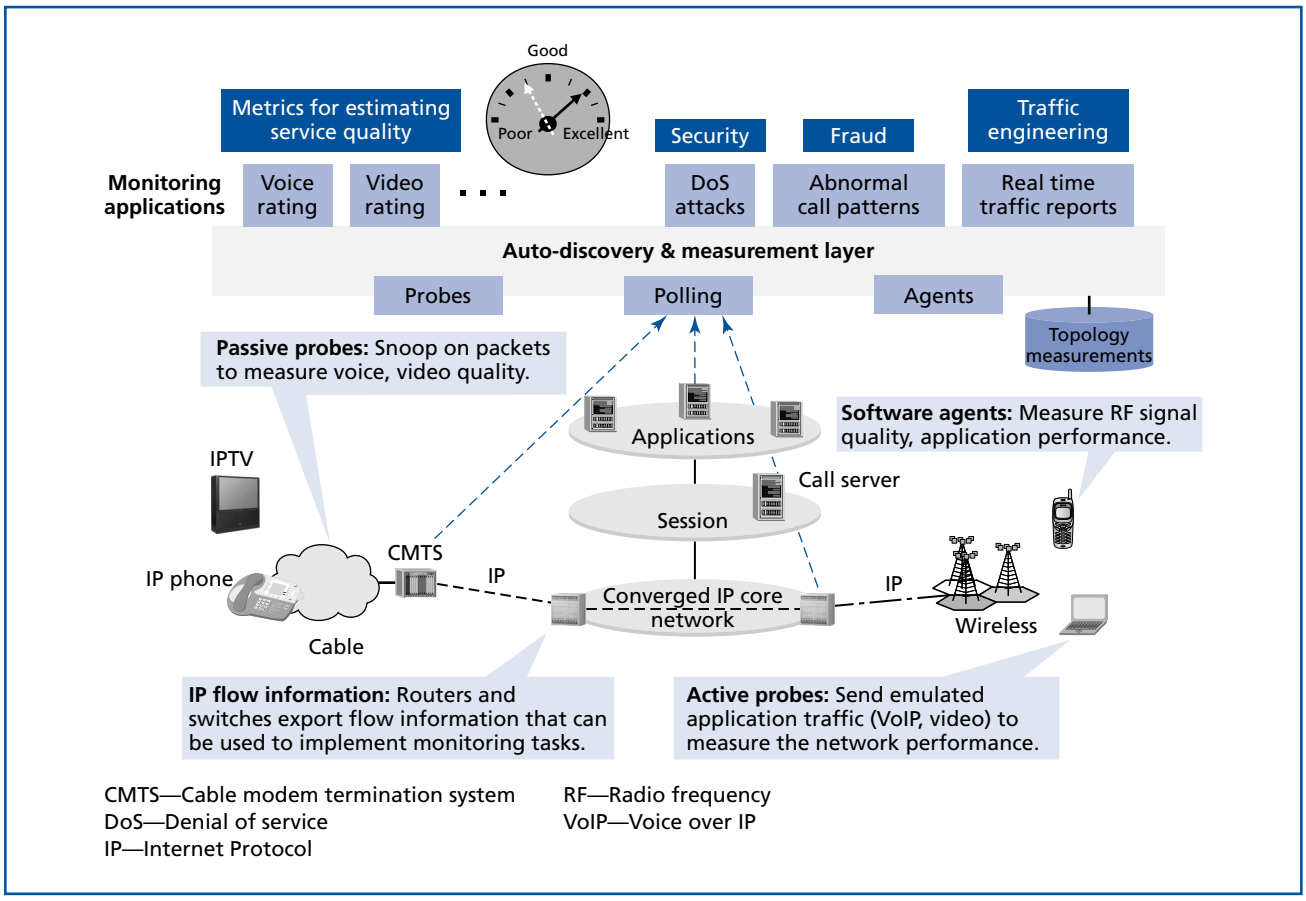


Figure 1.
Monitoring infrastructure for next generation converged networks.

The proposed monitoring infrastructure has the following components:

1. *Measurement sources.* These elements provide the monitoring applications with the necessary measurements from the network. The monitoring applications use this information to deduce the performance of the network and to identify faults. In addition to traditional SNMP-based measurements, we present four data acquisition methods that provide the monitoring applications with fine-grained, real time information: active probes, passive probes, software agents on wireless devices, and flow information from IP routers.

2. *Topology inventory.* The data collected using the measurement sources can be used to identify problems in the network. In order to identify the location and root cause of the problem, a key requirement is the knowledge of the network topology. Technologies like the NetInventory system [4] can provide the topology information necessary for fault diagnosis.

3. *Measurement layer.* This layer provides an interface that can be used by monitoring applications to gain access to the data collected by the various measurement sources.

4. *Monitoring applications.* Finally, the various monitoring applications use the data from the measurement layer to offer service providers insight into the performance of their networks for each service. Applications include monitoring the quality of voice or video services, identifying denial of service attacks, and providing traffic profiles of the service provider network (e.g., total amount of HTTP traffic, peer-to-peer traffic).

We now describe implementations of the various data acquisition sources and a few illustrative applications that are implemented on this platform.

Active Probes

Active probes [3] are software agents that run on designated end systems in the network and are used to measure the end-to-end service quality provided by the network for an application. Active probes send emulated application traffic through the network and measure the service quality of the network. This measured service quality is indicative of the quality that users would see when using the network. Service providers can program the active probes to monitor the network and report degradations in the network performance continuously and automatically.

To measure the performance of an application on the network, the Network Operations Center (NOC) issues a request to the active probes. The active probes establish connections among themselves, exchange artificially generated traffic that mimics the traffic pattern of the application, measure network performance parameters, and send the measurement report for that application to the NOC. For instance, if the NOC needs measurement reports for VoIP performance, it issues VoIP measurement requests to the active probes and the active probes measure parameters like call setup time, delay, loss, and jitter for the artificially generated VoIP traffic and report them to the NOC.

The active probing system uses two basic operations to perform monitoring, measurement task and measurement report. A measurement task is used by the NOC to get information about the network's performance for VoIP. The NOC identifies two active probes in the network to be a "caller" and a "callee," respectively. It then assigns a task to be performed by this pair of probes. The task consists of establishing a sequence of VoIP calls from the caller to the callee for a certain duration and measuring the network performance. A measurement task is composed using XML and sent to the probes. A task gives the addresses of the caller and the callee and other details about the task. For instance, if the measurement task is for VoIP, the task specifies the type of VoIP codec to be used, the number of VoIP calls that need to be established, the duration of each call, and other parameters.

Our active probes are multithreaded systems implemented in Java* and support SIP as the connection establishment protocol. Service providers can set up measurement tasks among a large set of probes (organized as a mesh). This gives service providers a networkwide view of performance on a service-by-service basis. Once the NOC issues a measurement task, the active probes handle all aspects of performing the measurements and send the reports to the NOC. The NOC does not have to coordinate the tasks in a fine-grained manner. The only responsibilities of the NOC are to issue measurement tasks and process

the returned reports, a great reduction of the load on the NOC.

Thus, active probes provide a "black box" measurement mechanism of the network and can be used to detect service degradations. It cannot, however, help pinpoint the root cause of failure or congestion. For example, while an active probe can detect an excessive delay through the network, it cannot identify which link's congestion is causing the delay. Similarly, if there is a link or a server failure, the active probes can detect the failure but cannot help identify the cause of the failure.

Passive Probes

Passive probes [3] provide a different perspective of the network as compared to active probes. Passive probes are installed on links within the network, and they snoop the traffic that flows through the links being monitored. They can be used continuously to monitor the performance of the network for the actual application traffic, as opposed to active probes that perform measurements for synthetic traffic. They also can be used to segment the network to identify sources of failures or congestion. However, if passive probes are used in isolation, they do not give an end-to-end perspective of network performance. They can compute the performance of the network only between the points of installation of the probes.

We now describe our implementation of a high-performance passive probe architecture that supports many monitoring applications.

Scalable High-Performance Implementation

Passive probes can be used to implement a large number of monitoring applications such as distributed denial of service (DDoS) attack detection, service quality monitoring, and traffic profiling. In order to support a large number of applications, the choice of the platform used to implement passive probes plays a crucial role. Hardware implementations provide little flexibility in adding new features. A platform that uses network processors allows us to deploy new monitoring applications easily; however, developing new applications is cumbersome and time consuming.

A commercial off-the-shelf PC or server provides maximal flexibility in terms of developing and deploying new applications. However, transferring data into the system memory from the network interface card (NIC), for example, interrupt handling and data copying, consumes most of the CPU resources, leaving very little for processing data. High-performance data capture cards such as the data acquisition and generation (DAG) cards from Endace* [7] as well as Xyratex* cards [27] significantly reduce the amount of CPU resources used to transfer data from the card to the system memory by reducing or eliminating interrupts and data copies. Hence, most of the CPU capacity can be used by the applications to process the data. For links that have low traffic volumes (10 Mbps to 50 Mbps), we use commercial network interface cards to perform the snooping. For higher rates, we use high-performance capture cards.

A passive probe performs the following functions for any application: capturing packets in system memory, application processing, and exporting the measurement information to the NOC. Our implementation uses multiple threads to implement these functions in order to use the CPU resources efficiently and to be able to scale to higher link speeds by using multiple processors. We implement concurrency control using wait-free and lock-free mechanisms.

Data Reduction

It is impractical for a passive probe to report information about every packet it sees to the NOC, and so, probes need techniques to reduce the amount of information transmitted to the NOC. A passive probe uses three methods to perform this data reduction: filtering, sampling, and aggregation.

Filtering. The NOC can install filters at passive probes to avoid processing packets that are not used by the application. For example, if a passive probe is used to perform VoIP monitoring, packets belonging to SIP sessions and Real Time Protocol (RTP) sessions are the only ones that are useful, and the rest of the packets can be discarded. Note that in addition to reducing the amount of information sent to the NOC, filtering reduces the information that needs to be processed at the probes. Libpcap [14] based implementations use filtering capabilities provided by the library. Specialized capture cards provide additional filtering capabilities such as regular expression matching in the payload.

Aggregation. A passive probe aggregates information into records on the basis of certain properties of the packets and sends a report for the collection of packets. For example, one option that is commonly used is to send a single record to the NOC for all packets belonging to a "flow," defined by a flow identifier. A flow identifier is defined differently for each protocol. For example, TCP/IP packets use the five tuple (source IP address, destination IP address, source port, destination port, and protocol) fields to define a flow identifier. However, a SIP flow identifier is defined by the TO, FROM, and CALL-ID fields of the SIP header. Records can be further aggregated by using other properties such as source IP address (e.g., if we are only interested in the total amount of traffic from users) or the requested URLs of HTTP requests (to measure the popularity of various Web sites: Yahoo*, Google*, and others.)

Sampling. Another technique to reduce the amount of traffic processed by the probe and sent to the NOC is to sample traffic and use only a fraction of it. For example, the probe can reduce the amount of traffic by picking one in every five packets at random, thus reducing the load on the probe by a fifth. Obviously, by sampling packets, we lose information and the reports received by the NOC are not accurate. Hence, the NOC has to be able to tolerate limited inaccuracy in the reports received from the probes. An alternate method of sampling is to process one in every five "flows" at random. While this method preserves the accuracy of the reports belonging to flows, it causes inaccuracy in terms of discarding entire flows.

Software Agents on Wireless Devices

Software agents can be deployed on mobile wireless devices to monitor and measure application-level performance on these devices. These agents use two different methods to obtain performance measurements: they can operate in the active mode or the passive mode. In the active mode, they operate in a manner similar to active probes and generate application traffic (as requested by the NOC) in order to measure end-to-end performance and diagnose problems. However, it is impractical to do this continuously because wireless bandwidth is at a premium. Rather, such measurements should be triggered only when some fault is suspected. This can be achieved either by having passive probes in the wireless network or by passively monitoring the traffic seen at that mobile handset. We have taken the second approach here, because it allows us to obtain a more accurate view of the end-user experience as well as obtain measurements such as signal coverage, which cannot be obtained by in-network probes.

These agents can be used to measure various kinds of statistics ranging from radio frequency (RF) and device level to application level. For instance, it can be used to monitor the number of lost calls, and the perceived bandwidth and latency. These RF-level measurements, along with location information, can be extremely useful to the service provider in identifying bottleneck areas. Device-level measurements such as usage statistics of device features give insight into the user preferences and are potentially useful in the design of new devices and applications. Finally, the protocol and application level measurements help in improving application design and allow detection of service bottlenecks. Thus, these agents provide various statistics that allow the service provider to obtain critical information for network monitoring and maintenance.

For handsets that run advanced operating systems such as Windows* CE or Symbian*, we have implemented these agents at the kernel level, thus making them completely transparent to the user. In particular, the IP traffic is passively monitored via hooks on the network stack that allow us to peek into each packet. **Figure 2** shows the implementation of the agents on the Symbian operating system, which consists of four components:

- A coordinator that handles various control and data messages,
- Inbound and outbound hooks that snoop, respectively, on incoming packets and outgoing packets, and
- A controller that acts as the intermediary between the agent and the NOC and effectively provides a local abstraction of the NOC.

Mobile handsets are severely constrained in terms of battery, memory, and CPU. Hence, the agents must have extremely low overhead to ensure that the user's activities are not impacted in any manner. Similar to

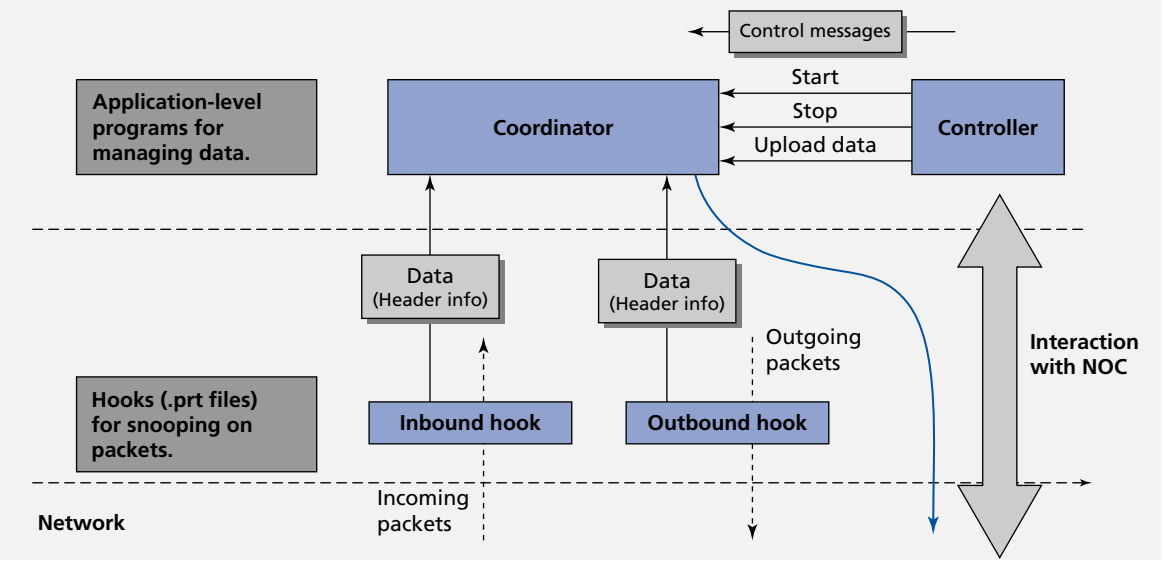


Figure 2.
Implementation architecture of agents on wireless devices.

passive probes, we use sampling to keep the CPU overheads low and maintain data summaries to reduce the memory usage. In addition, since wireless bandwidth is at a premium, we need to keep the number of message transmissions low as well as keep the message sizes small. The former is achieved by using alarm- or event-driven reporting, i.e., the agents report statistics to the NOC only when anomalous events occur. We have also designed the agents so that the NOC can configure the period at which the agent reports measurements; this period can be determined appropriately on the basis of the fraction of bandwidth that the service provider is willing to reserve for monitoring purposes. Message sizes can be kept small by sending data summaries and sending only deltas instead of complete information.

Flow Information From IP Routers

Often, network administrators rely on flow data to determine their networks' properties. Routers and switches throughout their networks continuously stream data summarizing each flow; however, these data can be overwhelming, often exceeding hundreds of gigabytes each day. Therefore, it is necessary to have a flow collector to aggregate the data in real time and provide the administrator with the desired information in a useful format. With a flow collector, an administrator can define queries describing the information of interest to him or her. For example, a network administrator may want to know which users are connecting to each other, and how much data they are transferring. In this situation, he would create an aggregator with source IP address and destination IP address as the key elements, and packet count or byte count as the value elements. The collector output can be sorted to see easily which users are sending or receiving the most data. **Figure 3** shows the architecture of a flow collector.

Flow information is the summarized traffic statistics, exported from routers in standard industry formats like NetFlow [17], J-Flow [11], and sFlow* [23]. A flow is typically defined as a unidirectional sequence of packets that have the same destination

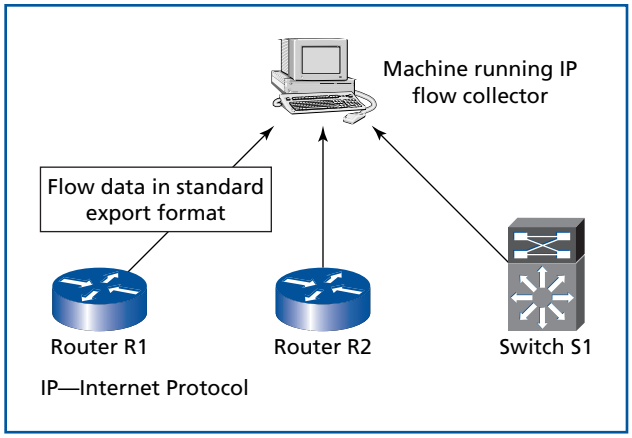


Figure 3.
Architecture of the flow collector.

and source address, transport level information, type of service (TOS) bits, and protocol information. Technologies like NetFlow, JFlow, and sFlow enable routers and switches to collect information about all the flows passing through them and export these "flow records" to a central collector, where they are used for various purposes like network monitoring and planning, traffic analysis, accounting, and data mining. A flow record typically contains the source and destination IP addresses, source and destination port numbers, protocol, ToS byte, next-hop router IP address, input and output interface numbers on the router, packet count, byte count, start time, and end time of the flow.

In a typical Internet service provider (ISP) network, there are hundreds of routers generating flow records at very high rates. Thus, building a flow collector that is scalable and can handle large amounts of data is a key requirement for any real world deployment. In addition, users may specify a diverse set of aggregation queries to meet the needs of their network management applications such as traffic engineering, traffic demands estimation, top-N reports identifying the top sources, destinations and hosts, SLA verification, anomaly detection, and fault monitoring and diagnosis. Hence it is important to provide a flexible query format, which can support large varieties of aggregation queries that are of interest to network management applications.

Query Format

An aggregation query has three main components:

- *Filters.* Filters are conditions that specify whether a record should be considered for further aggregation. They are range specifications on various fields in the flow records. Thus, each filter is like a multidimensional box whose boundaries along a dimension coincide with the range specified for the field corresponding to the dimension. Only records that fall within this box are considered for aggregation. Filters can be specified on any field including IP addresses, port numbers, and the protocol field. We support nested filters that can be combined with Boolean operations (and, or, not).
- *Aggregators.* An aggregator has two parts:
  - Key fields, on which the aggregation is performed. All records with the same combination of values for the key fields are grouped together in the aggregated output.
  - Value fields, which are the fields that are accumulated for each unique set of key fields. The value specification also includes an operation to define how to aggregate the information. These operations include sum, max, min, average, and rate.
- *Period.* This is the time interval over which aggregation is performed. After each time period, tuples comprising key fields and the aggregated value fields are output.

Performance of the Multithreaded NetFlow Collector

Our NetFlow collector is multithreaded with basic load balancing of aggregators so that it can easily run on multicore machines using parallelism at full capacity. For efficiency, concurrency control is primarily done using nonlocking synchronization primitives. Furthermore, our system uses heuristics to coalesce filters and merge aggregation schemes to reduce computational expense.

Input to the collector is a configuration file that governs the specification of keys, values, filters and aggregation queries. The collector has three main modules—listener, aggregator, and output—each

optimized for performance. Each of these modules is implemented as a separate thread. We associate one listener for each exporting device. The listener threads transfer the arriving NetFlow records into the system memory; the aggregator threads process these records one query at a time; and the output thread periodically writes out the results. Nonblocking operations ensure that the system operates at maximal capacity and utilizes the CPU efficiently to deliver maximal throughput.

Monitoring Applications

We have described our monitoring infrastructure consisting of various measurement methods. This provides a powerful platform to develop and deploy a wide variety of monitoring applications. In this section, we describe a few monitoring applications that can be implemented on this infrastructure.

VoIP Service Quality Monitoring

VoIP is a key revenue generating service on converged networks because of its lower cost per call than traditional telephone networks. Internet-based voice applications like Skype* [24], Yahoo! Voice* [26], and Net2Phone* [16] and VoIP services by Vonage* [25] are very popular. Thus, assuring good quality of service for VoIP services is very critical for service providers to maximize their revenues. Network parameters that affect the quality of a voice call are call set-up time, tear-down time, delay, loss, and jitter. These measured parameters are used to compute a voice quality metric that subjectively represents user perceived quality. In this section, we discuss the voice quality metrics that we use and the implementation of a VoIP monitoring system using our probes.

VoIP service quality metric. The quality of a voice call is affected by three sets of impairments: signal processing impairments, network impairments, and environmental impairments. Signal processing impairments are caused by the voice filters, quantizers, and codecs employed in the system. Network impairments like delay, loss, and jitter determine the quality of a voice call. Environmental impairments include factors like the ambient noise levels in the user's location.

Measurements through monitoring mechanisms are mapped by voice quality metrics to the actual quality of the end-user experience, referred to as mean opinion score (MOS). At present, there are no standardized voice quality metrics for VoIP. The International Telecommunication Union (ITU) has defined two voice quality metric standards for the circuit-switched public switched telephone network (PSTN), the E model [8], and perceptual evaluation of speech quality (PESQ) [19]. Both these models were originally designed to do transmission planning and attempt to address all the different impairments. The PESQ algorithm predicts the subjective MOS values by comparing the received signal distortion to a standard reference signal. It assumes that the distorted signal should take into account the network impairments, and therefore network performance does not need to be specified explicitly. PESQ does not take into account frequency responses and loudness, which are two important factors affecting perceived quality. The E model uses an additive impairment model that factors in all impairments including environmental factors that affect the voice quality. It, however, is not as accurate as PESQ from a signal processing perspective. The additive nature of the E model makes it more amenable to explicitly accounting for network impairments on a per-call basis. Therefore, we use a metric based on the E model that has been adapted for VoIP as the voice quality metric to determine the MOS. References [3, 6, 18] describe the E model in more detail.

Active measurements. Active measurements of VoIP service quality are initiated by the NOC by issuing measurement tasks to the probes. The measurement task issued by the NOC contains a call profile that describes each of the task's parameters, such as the total number of calls that need to be made between the caller and the callee as part of the measurement task, the duration of each call, the time that the probes wait after completing a call before initiating the next call, and the type of codec that is used for the task. Both active probes as well as agents on wireless devices are capable of executing active measurement tasks. We refer to both as probes from here on. The probes establish VoIP calls based on requests from

the NOC; send artificially generated traffic based on the codec; measure parameters like call set-up time, call tear-down time, delay, loss, and jitter, which affect call quality; and compute the perceived call quality using this information. This information is then reported to the NOC, which tracks the service quality degradation across the network.

Our implementation requires the NOC to identify a caller/callee pair and send the measurement task to the probes. The probes coordinate among themselves to execute the task, requiring no coordination by the NOC. This reduces the load on the NOC and enables scaling the system to handle a large network from the NOC. Reference [3] describes the system in more detail.

Passive measurements. Passive probes, flow information from IP routers, and software agents can act as sources for passive measurements. These measurement methods observe the traffic flowing through the interface(s) they are monitoring and report information about the packets flowing through. The NOC uses the aggregate information sent by the probes to correlate them and compute service quality parameters such as delay, loss, and jitter. For these measurements, the probes synchronize their local clocks using mechanisms such as Network Time Protocol (NTP) [15] or Global Positioning System (GPS)/code division multiple access (CDMA)–based clock synchronizers.

NetFlow collection and analysis allow gathering of statistics regarding delay, loss, and other attributes. Packet loss can be computed by correlating packet counts at routers. Delay can be estimated by maintaining the start time and end time of flows at all routers and calculating the time difference at the first and last router of the flow. However, when the losses are high in the network, this estimate of the delay can be inaccurate. Note that the most straightforward method of correlating NetFlow information to applications is through the destination port numbers. In the case of VoIP, where RTP is used to transport voice packets, the two sides negotiate a dynamic port to exchange the RTP packets using a signaling protocol like SIP. Thus, it is not straightforward to identify a VoIP flow with an arbitrary port number.

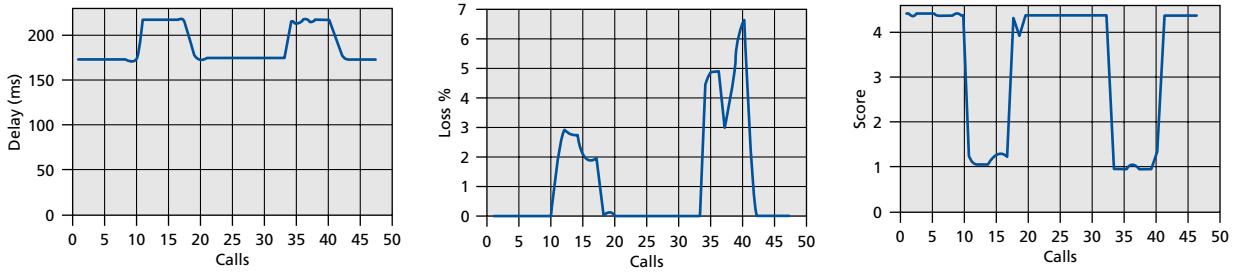
Passive probes and software agents provide more flexibility in terms of looking into the packet to find requisite information as well as to provide better estimates of delay in the network. SIP [21] is an ASCII-based protocol and, hence, SIP packets can be easily identified by looking for patterns such as "SIP/2.0" in the packet. RTP packets, on the other hand, cannot be identified as easily. The RTP request for comments (RFC) [22] prescribes the following, which we use to validate the packets as an RTP stream:

- The version number of the packet must be 2,
- The length of the RTP packet should be the same as the length header, and
- Three consecutive sequence numbers should be received at some point in the flow's duration.

Passive probes (and software agents) compute loss in the network by looking at the total number of packets received by the probes belonging to the same flow; the difference gives us the loss in the network between the probes. In order to compute the delay between two probes, we estimate the average timestamp of the set of packets at a probe. When packets are lost, the timestamp information can become inaccurate. Hence, we compensate for lost packets by adding timestamps artificially for lost packets. This gives us more accurate estimates of the delay in the network. Reference [3] describes this procedure in more detail.

Reports. We now present the reports generated by the probes for a particular test scenario. We installed the active probe software on two machines at two enterprise locations, one in Bangalore, India, and the other in Murray Hill, New Jersey, in the United States. We denote the two machines as BA and MH, respectively. We performed a measurement test between the two sites and set up 50 VoIP calls using the G.711 codec and then measured the performance of the network between the two sites. We also used passive probes to snoop on the calls made by the active probes at the same two locations. While the calls were in progress, we set up User Datagram Protocol (UDP) sources to send continuous traffic from MH to BA. This fills the router buffers and results in packet drops. We start one source around call number 10 and stop it at call 18. We then start two sources around call 33 and stop the sources at call 42.

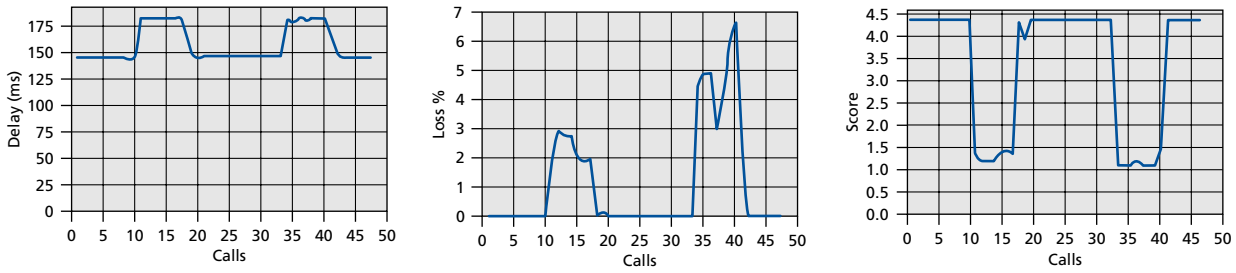
Figure 4 shows the results from the active probes and **Figure 5** shows the results from the



(a) One-way delay (b) Packet loss percentage (c) MOS score

MOS—Mean opinion score
ms—milliseconds
VoIP—Voice over Internet Protocol

Figure 4.
VoIP service quality reports using active probes.



(a) One-way delay (b) Packet loss percentage (c) MOS score

MOS—Mean opinion score
ms—milliseconds
VoIP—Voice over Internet Protocol

Figure 5.
VoIP service quality reports using passive probes.

passive probes. The delay measurements from the active probes include the packetization delay introduced at the source and are, hence, around 25 milliseconds (ms) higher than the delays measured by the passive probes. The passive probes measure the losses and the mean opinion score as accurately as the active probes.

Wireless Service Management

Agents on mobile handsets enable service providers to deploy sophisticated wireless network management applications. Service providers can compile highly accurate signal coverage maps using these agents. In particular, if the mobile handset allows location computation, then the agent can correlate the signal strength measurements with the location information to determine the locations with poor signal quality. Examples include GPS-enabled handsets and handsets with operating systems that provide application programming interfaces (APIs) to calculate location via triangulation. Since the agents can continuously monitor the signal strength, the NOC can periodically collect the information from multiple agents to generate real time coverage maps. This can be used to quickly identify a failing RF subsystem, as well as to expand the network and ensure full coverage in all areas.

In addition, the agents can help in remote troubleshooting. For instance, if a user is experiencing some issues with his phone, then the customer service representative can send messages to the agent on the phone to request statistics or to invoke specific active tests that can be used to isolate the root cause. Thus, issues can be potentially diagnosed and solved remotely.

The agents can also be useful in identifying certain kinds of attacks that are specific to wireless networks; for instance, battery-draining attacks can be detected by identifying the period for which a Point to Point Protocol (PPP) connection is established and the number of such occurrences. The agent can detect any anomalous behavior in real time and report the issue to the NOC, allowing quicker identification of the attacker.

SLA Verification

Service level agreement (SLA) performance monitoring involves measuring and reporting against designated SLAs. Each SLA can have various parameters that must be measured, including delay, loss, throughput, and availability. By cleverly utilizing active probes, passive probes, and flow collectors, we are able to determine each of these statistics. Flow collectors are ideal for accurately reporting throughput by aggregating the amount of data transmitted from each source. Other information such as delay and loss are more easily determined using active or passive probes or wireless software agents.

Fault Detection and Diagnosis

Link congestion can cause significant delays in the network that can lead to degradation of service quality. Similarly, network element failures can result in dramatic degradation of service quality. Using active probes and passive probes, we can detect this degradation. However, it is essential to identify the cause (link or network element) of this failure, allowing service providers to take immediate remedial action. We implement fault diagnosis by carefully choosing a set of overlapping paths in the network and monitoring the service quality received on these paths. If a link becomes congested (or a network element fails), all paths that use that link (or network element) simultaneously notice degradation in the service quality received. This allows us to focus on the link (or network element) that causes the degradation. Service providers can then take remedial action by routing traffic on an alternate path. Reference [2] describes our algorithms in more detail.

Application Profiling

Application profiling includes characterizing the network load by classifying the flow records into the applications generating them. Port number based classification is often insufficient since many applications do not use standard port numbers. Hence, some application specific information and heuristics must be used to classify traffic. With the flow data, we can estimate the average packet interarrival time and average packet size of the flow concerned. Additional information about communication pattern (e.g., most popular destination or cliques in the network [12]) can also be determined from the correlation of flow data across the network. This can be used to classify traffic into corresponding applications.

Distributed Constraint Monitoring

Many monitoring applications do not need to know the exact state of the network; rather, they require knowing only whether the network exhibits unexpected or anomalous behavior. For example, if a service provider requires the delay experienced by VoIP service to be less than 180 ms, it is not essential to track the exact delay experienced by each VoIP session. It is sufficient to notify the service provider if the delay of some VoIP session has exceeded the threshold of 180 ms.

Distributed constraint monitoring (also known as distributed triggers) is a mechanism that decomposes networkwide constraints, such as the VoIP delay constraint, into a set of "local" constraints at each measurement device in the network. As long as the local constraints are satisfied at the measurement devices, we know that the network performance conforms to expectation. Local constraints act as filters that reduce the amount of information communicated on the network and, thus, reduce the monitoring burden on the network.

Distributed constraint monitoring can be used to implement many different monitoring applications,

which are interested only in anomalous behavior of the network. Examples include congestion detection, SLA violation detection, and distributed denial of service (DDoS) attack detection. References [1, 9, 10, 13] discuss algorithms to implement distributed constraint monitoring efficiently.

Conclusion

Service providers are moving toward providing many different services such as VoIP and IPTV on their IP networks. The performance requirements of these services are very different from those of traditional data services. Thus, traditional network management systems are not suitable to managing converged networks. Specifically, these systems do not have monitoring mechanisms that provide fine-grained and real time information about the traffic flowing through the network. In this paper, we proposed a monitoring infrastructure that enables service providers to easily deploy monitoring applications for a wide variety of services. The monitoring infrastructure is composed of various methods that monitor the performance of a diverse set of protocols and elements: active probes, passive probes, software agents on wireless devices, and flow collectors. Service providers can use this information to deploy new monitoring applications for new services and, thus, manage their converged networks at a per-user, per-service granularity.

References

[1] S. Agrawal, S. Deb, K. V. M. Naidu, and R. Rastogi, "Efficient Detection of Distributed Constraint Violations," Proc. IEEE 23rd Internat. Conf. on Data Engineering (ICDE '07) (Istanbul, Tur., 2007), pp. 1320–1324.

[2] S. Agrawal, K. V. M. Naidu, and R. Rastogi, "Diagnosing Link-Level Anomalies Using Passive Probes," Proc. 26th IEEE Internat.Conf. on Computer Commun. (IEEE INFOCOM '07) (Anchorage, AK, 2007), pp. 1757–1765.

[3] S. Agrawal, P. P. S. Narayan, J. Ramamirtham, R. Rastogi, M. Smith, K. Swanson, and M. Thottan, "VoIP Service Quality Monitoring Using Active and Passive Probes," Proc. First Internat. Conf. on Commun. System Software and Middleware (IEEE Comsware '06) (New Delhi, India, 2006), pp. 1–10.

[4] Y. Breitbart, M. Garofalakis, B. Jai, C. Martin, R. Rastogi, and A. Silberschatz, "Topology Discovery in Heterogeneous IP Networks: The NetInventory System," IEEE/ACM Trans. Networking, 12:3 (2004), 401–414.

[5] G. Camarillo and M. A. Garcia-Martin, The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds, John Wiley, Chichester, Eng., 2005.

[6] R. G. Cole and J. H. Rosenbluth, "Voice Over IP Performance Monitoring," ACM SIGCOMM Computer Commun. Rev., 31:2 (2001), 9–24.

[7] Endace, <http://www.endace.com>.

[8] International Telecommunication Union, Telecommunication Standardization Sector, "The E-Model, A Computational Model for Use in Transmission Planning," ITU-T Rec. G.107, Mar. 2003, <http://www.itu.int>.

[9] A. Jain, J. Hellerstein, S. Ratnasamy, and D. Wetherall, "A Wakeup Call for Internet Monitoring Systems: The Case for Distributed Triggers," Proc. 3rd Workshop on Hot Topics in Networks (ACM SIGCOMM HotNets III) (San Diego, CA, 2004).

[10] N. Jain, P. Yalagandula, M. Dahlin, and Y. Zhang, "INSIGHT: A Distributed Monitoring System for Tracking Continuous Queries," Work in Progress Session, Proc. 20th ACM Symposium on Operating Syst. Principles (ACM SOSP) (Brighton, Eng., 2005), pp. 1–7.

[11] J-Flow, <http://www.juniper.net>.

[12] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: Multilevel Traffic Classification in the Dark," Proc. Conf. on Applications, Technologies, Architectures, and Protocols for Computer Commun.

(ACM SIGCOMM) (Philadelphia, 2005), pp. 229–240.
[13] R. Keralapura, G. Cormode, and J. Ramamirtham, "Communication-Efficient Distributed Monitoring of Thresholded Counts," Proc. ACM SIGMOD Internat. Conf. on Management of Data (Chicago, 2006), pp. 289–300.
[14] Libpcap, <http://www.tcpdump.org>.
[15] D. L. Mills, "Network Time Protocol (Version 3): Specification, Implementation and Analysis," IETF RFC 1305, Mar. 1992.
[16] Net2Phone, <http://www.net2phone.com>.
[17] NetFlow, <http://www.cisco.com>.
[18] M. E. Perkins, C. A. Dvorak, B. H. Lerich, and J. A. Zebarth, "Speech Transmission Performance Planning in Hybrid IP/SCN Networks," IEEE Commun. Mag., 37:7 (1999), 126–131.
[19] PESQ, "Perceptual Evaluation of Speech Quality," <http://www.pesq.org>.
[20] G. Prabhakar, R. Rastogi, and M. Thottan, "OSS Architecture and Requirements for VoIP Networks," Bell Labs Tech. J., 10:1 (2005), 31–45.
[21] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," IETF RFC 3261, June 2002.
[22] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," IETF RFC 3550, July 2003.
[23] sFlow, <http://www.sflow.org>.
[24] Skype, <http://www.skype.com>.
[25] Vonage, <http://www.vonage.com>.
[26] Yahoo! Voice, <http://voice.yahoo.com>.
[27] Xyratex, <http://www.xyratex.com>.

(Manuscript approved March 2007)

SHIPRA AGRAWAL worked as a member of technical

staff at Bell Labs Research India in Bangalore from 2004 to 2006. She is currently a doctoral candidate at the Department of Computer Science, Stanford University, in California; she received her master's degree from the Indian Institute of Science. Her interests are in the areas of data streaming, datamining, and privacy.

C. N. KANTHI is a member of technical staff at Bell Labs

Research India in Bangalore. She received her master's degree in computer science and engineering from the Indian Institute of Technology, Delhi, and her bachelor's degree in computer science and engineering from SJCE, Mysore, India. Prior to her tenure at Bell Labs, she also worked with Cisco Systems India in Bangalore. Her research interests include wireless networks, routing protocols, and operating systems.

K. V. M. NAIDU is a member of technical staff at

Bell Labs Research India in Bangalore. He received his B.Tech. degree in computer science and engineering from the Indian Institute of Technology in Guwahati, and his master's degree in computer science and engineering from the Indian Institute of Science in Bangalore. His research interests span the areas of algorithms, multimedia networking, and software systems for mobile handsets.

JEYASHANKHER RAMAMIRTHAM is a member of

technical staff at Bell Labs Research India in Bangalore. He has a B.Tech. from the Indian Institute of Technology, Madras, and a doctor of science from Washington University in St. Louis, Missouri. His primary interests are in the areas of high-speed switching and routing, and management of next-generation networks.

RAJEEV RASTOGI is the executive director of Bell Labs

Research India in Bangalore. He received his B.Tech. degree in computer science from the Indian Institute of Technology, Bombay, and his master's and Ph.D. degrees in computer science from the University of Texas at Austin. Dr. Rastogi was named a Bell Labs Fellow in 2003. He is active in the fields of networking and databases and has served as a program committee member for several conferences in the database area. His writings have appeared in a number of ACM and IEEE publications, and other professional conferences and journals. His research interests include database systems, network management, and knowledge discovery. His most recent research has focused on the areas of network topology discovery, monitoring, configuration and provisioning, XML publishing, approximate query answering, and data stream analysis.

SCOTT SATKIN is an associate at Bell Labs Research India in Bangalore. He received his bachelor's degree in computer science engineering from Washington University in St. Louis, Missouri. His research interests include sensor networks, geographic information systems, and computer vision.

ANAND SRINIVASAN is a member of technical staff at Bell Labs Research India in Bangalore. He received his B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Bombay, and his master's and Ph.D. degrees in computer science from the University of North Carolina at Chapel Hill. His research interests span the areas of real time scheduling, operating systems, multimedia networking, and peer-to-peer systems. ◆