Scalable Monitoring Support for Resource Management and Service Assurance

Abolghasem (Hamid) Asgari and Richard Egan, Thales Research & Technology (UK) Limited
Panos Trimintzios and George Pavlou, University of Surrey

Abstract

Continuous monitoring of network status and its resources are necessary to ensure proper network operation. Deployment of QoS-based value-added services in IP networks necessitates the employment of resource management techniques and specifically the use of traffic engineering. The latter typically relies on monitoring data for both offline proactive and dynamic reactive solutions. The variety of data to be collected and analyzed using different measurement methods and tools, and the extent of monitoring information to use demand a proper QoS monitoring infrastructure. A monitoring system should be scalable in terms of network size, speed, and number of customers subscribed to value-added services. This article investigates the requirements of scalable monitoring system architectures, proposes principles for designing such systems, and validates them through the design and implementation of a scalable monitoring system for QoS delivery in IP differentiated services networks. Experimental assessment results prove the accuracy and scalability of the proposed monitoring system.

Quality of service (QoS) monitoring is becoming crucial to Internet service providers (ISPs) for providing quantified QoS-based services and service assurance. Traffic engineering (TE) is the set of techniques that allow ISPs to maximize network resource utilization while at the same time meeting the QoS demands of services contracted to customers. Traffic engineering deals mainly with performance optimization of operational IP networks and encompasses the application of principles to the measurement, characterization, and control of traffic. Traffic engineering algorithms need knowledge of network status and history for their reactions. The functionality to capture this information is operational measurements, where the operational state of the network is monitored in order to assist traffic engineering functions.

In this article we assume that the performance requirements of a customer's requested service are described in a service level specification (SLS) [1], which is the technical part of a service level agreement (SLA) with the provider. In such a provider's network, QoS is supported in a scalable manner with the differentiated services (DiffServ) model [2], where routers aggregate traffic that belongs to several service classes according to predefined QoS policies. By QoS, we refer to a service offering where one or more performance parameters (i.e., throughput, delay, loss, delay variation) are quantified [1]. As the network attempts to offer several different service types, such as real-time traffic, virtual private networks (VPNs), and best effort services, network and service monitoring are important for providing end-to-end QoS and service assurance.

In this context, monitoring not only has a diagnostic role, but also becomes an important tool for supporting network operation and providing service auditing. Given the multitude of services with different performance requirements, measurement information needs to be collected at finer granularity than that of per ingress-egress node pairing, and the service type also needs to be taken into account. Thus, design and implementation of *scalable* monitoring systems capable of providing measurements for network provisioning, dynamic resource allocation, route management, and in-service verification of value-added services is a big challenge.

Overview of Network Monitoring Related Activities

There are a number of working groups in the Internet Engineering Task Force (IETF) [3] related to measurements and monitoring. The IETF IP Performance Metrics (IPPM) working group has defined a set of metrics that can be applied to the quality, performance, and reliability of Internet data delivery paths and services. The defined metrics are based on active performance measurements such as one-way delay, loss, delay variation, round-trip delay, and link bandwidth. The IETF IP Flow Information Export (IPFIX) working group was chartered to develop a common IP traffic flow measurement technology. It has defined a system architecture for IP flow information export that includes the flow definition, a metering process with sampling/filtering capabilities, a data export process, and a data export protocol between the observation and collection points. The IETF Real-Time Traffic Flow Measurement (RTFM) working group defined an architecture and methodology based on meters and a set of classification rules. The RTFM architecture consists of four main components: a powerful and flexible meter, meter reader, meter manager, and analysis application

for processing the collected data. NeTraMet is one implementation of the RTFM architecture, and although powerful and flexible with respect to the measured metrics, it is complicated to deploy and use.

High link speeds and growing fine-grained measurements increase the demand for data collection resources and could also place a large amount of processing overhead on network elements, impairing them in performing their main functions such as routing and packet forwarding. Deployment of packet sampling techniques can help to prevent overload of monitoring resources and limit measurement processing costs. The IETF Packet Sampling (PSAMP) working group was chartered to define a standard set of simple but rich capabilities for sampling packets through statistical and other methods. A packet sampling technique selects a representative subset of packets, which is used to infer knowledge about the whole set of observed packets without processing them all.

There exist numerous network performance monitoring tools focusing on path performance and network-wide measurements. The RIPE Network Coordination Center has implemented a number of the measurement protocols defined by IPPM. The RIPE Test Traffic Measurement (TTM) [4] project measures one-way delay and packet loss, bandwidth, and so on between installed measurement probes. NetFlow from Cisco includes a meter coupled with an unreliable data transport mechanism and a flow collector. The latter provides data filtering and aggregation capabilities. A flow analyzer application is then used as a post-processor. NetFlow reports statistics on byte and packet counts per micro or aggregated flows. National Internet Measurement Infrastructure (NIMI) [5] is a software system for building network measurement infrastructures and managing measurement tools. NIMI relies on servers being deployed at different network points. NIMI generates monitoring traffic sent among the NIMI servers and computes traffic characteristics based on active probes. Metrics such as available bandwidth, delay, and packet loss can be computed. The Measurement and Network Analysis Group of the National Laboratory for Applied Network Research (NLANR) has developed the Network Analysis Infrastructure (NAI) [6]. This includes two core projects, the Passive Measurement and Analysis Project that addresses workload profiles for a number of measurement points in high-speed environments, and the Active Measurement Project that addresses performance parameters such as round-trip time, packet loss, topology, and throughput among the deployed number of active monitors. The Cooperative Association for Internet Data Analysis (CAIDA) [7] is engaged in several measurement-related projects for monitoring Internet traffic. CAIDA has developed a variety of tools such as cflowd, RTG, skitter, NeTraMet, CoralReef, and Beluga. Performance analysis and modeling are addressed by macroscopic topology analysis research at CAIDA. The skitter tool is used to actively probe the Internet in order to analyze topology and performance. The Sprint IP Monitoring (IPMON) project [8] designed a system to collect packet-level traffic statistics at up to OC-192 link speeds and provide offline detailed observation of network traffic characteristics to be used for network overprovisioning purposes, load balancing, detecting network abnormalities, and so on.

There is also ongoing work on monitoring and measurements for traffic engineering at the interdomain level by European research projects [9]. The objective of the IST-INTERMON project is to develop an integrated interdomain QoS monitoring, analysis, and modeling system to be used in multidomain Internet infrastructure for the purpose of planning, operational control, and optimization. The focus of the IST-MoMe project is the enhancement of interdomain real-time QoS architectures with integrated monitoring and measurement capabilities. The objective of the IST-SCAMPI project is to develop an open and extensible network monitoring architecture including a passive monitoring adapter at 10 Gb/s speeds, and other measurement tools to be used for denial-of-service detection, SLS auditing, QoS, TE, traffic analysis, billing, and accounting.

There has also been some work at the intradomain level to use measurement information for tackling network performance degradation and managing congestion in operational networks as well as addressing service level monitoring. NetScope [10] provides a unified set of software tools for TE of IP backbone networks by using network measurements to derive traffic demands and then update the network configuration setup in a non-real-time fashion. Rondo [11] is designed as an automated control system using a monitoring system to react to and manage congestion in multiprotocol label switching (MPLS) TE networks in near-real time. With respect to service level monitoring, a generic system for VPN monitoring is presented in [12]. Also, a mechanism for monitoring of SLAs based on aggregation and refinement is presented in [13] in order to reduce the amount of information exchange between distributed nodes and a central network management system. In [14] the authors studied a number of algorithms for minimizing the communication overhead of monitoring systems. Most of the above work is complementary to ours. Some preliminary results of our monitoring work were presented in [15].

The above measurement tools and systems collect, analyze, and visualize forms of Internet traffic data such as network topology, traffic load, performance, and routing. A large amount of information is needed for large operational IP networks, especially when used for TE and service-level monitoring. Scalability is thus a big challenge for monitoring systems and necessitates suitable system architectures for scalable monitoring in real time. This article addresses the scalability of monitoring systems, proposes principles for designing and assessing such systems, and describes an example system designed and assessed according to these principles. The system supports QoS-based service level monitoring and provides QoS-based real-time monitoring support for TE in MPLS DiffServ-capable networks.

Monitoring and Measurement Requirements for Resource and Service Management

Network capacity and traffic management are achieved through TE mechanisms and realized with capacity planning, routing control, resource management that includes buffer and queue management, and other functions that regulate and schedule traffic flows through the network. The target is to accommodate as many customer requests as possible while at the same time satisfying their QoS requirements. The state-dependent TE functions require the observation of network state through monitoring and apply control actions to drive it to a desired state. This can be accomplished by both reacting in response to the current network state and proactively using forecasting techniques to anticipate future traffic demand and preconfigure the network accordingly. The goal of monitoring is not only to measure QoS metrics, but also provide information in order to guarantee the contracted services by means of tuning and controlling network resources.

A monitoring system should provide information for the following three categories of tasks:
1. Assist dynamic online TE in making provisioning decisions for optimizing the usage of network resources according to

short- to medium-term changes. The ability to obtain statistics at the QoS-enabled route level is important and, as such, an essential requirement. This information can be used to take appropriate actions on setting up new routes, modifying existing routes, performing load balancing, and rerouting traffic. It is also used for performing node-level optimizations on resource reservations (bandwidth assignment and buffer management) to combat localized congestion.

2. Assist offline TE in providing analyzed traffic and performance information for long-term planning in order to optimize network usage and avoid undesirable conditions. The analyzed information includes traffic matrices and growth patterns, and long-term utilization indications.

3. Verify whether the QoS performance guarantees committed to in SLSs are in fact being met. SLSs can differ depending on the type of services offered, and different SLS types have different QoS requirements that need processing of different types of information [16]. In-service verification of traffic and performance characteristics per service type is required.

Traffic engineering is a continual and iterative process of network performance improvement. The optimization objectives may change over time as new requirements and policies are imposed, so monitoring and measurement systems must be generic enough to cope with such changes.

Monitoring can occur at different levels of abstraction. Measurements can be used to derive packet level, application level, user/customer level, traffic aggregate level, node level, and network-wide level information. In TE networks, monitoring takes place at the network layer for deriving all the above information. Relevant measurements include one-way delay, delay variation, one-way packet loss, traffic load, and throughput.

There are two methods to perform low-level measurements in a monitoring system: *active* and *passive* measurements. *Active measurements* inject synthetic traffic into the network based on scheduled sampling in order to observe network performance. Active measurement tools typically require cooperation from both measurement endpoints. For one-way delay measurements, methods such as the Network Time Protocol (NTP), Global Positioning System (GPS), or other code-division multiple access (CDMA)-based time sources can be used for time synchronization. *Passive measurements* are mainly used to observe actual traffic patterns in the network but can also be used for network performance monitoring. Traffic monitoring requires continuous collection of data and monitoring of links at full load, which can be problematic on very high-speed links as it demands computing resources. The quality of analyzed information depends on the granularity and integrity of collected data.

Principles for Designing Scalable Monitoring Systems

Scalability in QoS-enabled IP networks and their associated services has three aspects: size of network topology, number and granularity of classes of service supported, and number of subscribed customers. Network topologies are characterized by a number of parameters, such as number of nodes and links, speed of links, degree of physical and logical connectivity, and network diameter. In QoS-enabled IP networks, supported services are mapped to a number of classes according to the DiffServ model, which has an impact on the scale of the monitoring system. A large number of subscribed customers subsequently require a large amount of information to be gathered for service monitoring and assurance, as in service verification of QoS, performance guarantees are required for individual customers.

The scalability of a monitoring system is the ability of effectively deploying the system at the scale of a *large* network offering a number of services to a large number of customers. The monitoring system must have a number of design features for a wide range of monitoring tasks that ensure a scalable solution for delivering the expected performance. The monitoring tasks include data collection, data aggregation, and data analysis for providing feedback. A diverse variety of measurement data is needed in order to perform both network and service performance monitoring. The amount of measurement data increases in QoS-enabled networks because there are a number of per-class states (e.g., different queues) per interface and a large number of QoS-based routes that must be monitored. Hence, scalable monitoring architectures must adhere to the principles we describe below.

Defining the QoS Monitoring Process Granularity at Aggregated Levels

An SLS is associated with unidirectional flows where the flow is identified as in [1] by a number of attributes including the DiffServ information (DSCP value), source and destination IP address information, and any application-related information such as protocol and port numbers. A micro-flow is a correlated set of IP packets for which all the above attributes are specified. A macro-flow is a correlated set of packets that should be treated by the network in a prescribed way. A macro-flow consists of a set of micro-flows with similar requirements, but not all of the above attributes are specified. As an example at the macro-flow level, for DiffServ behavior aggregate classification, only the DSCP attribute is required. Similarly, for TE and monitoring purposes, we identify an aggregated-flow as the correlated set of packets that are treated in the network in a similar manner (e.g., packets treated by the same per-hop behavior, PHB; packets carried over the same label switched path, LSP; or packets routed through the same IP route). It should be noted that DiffServ defines router forwarding behaviors known as PHBs. A PHB includes the differential treatment individual packets receive, implemented by queue management disciplines. DiffServ specified the following PHBs: expedited forwarding (EF) for premium services, assured forwarding (AF) for Olympic services, and best effort (BE) as the default PHB.

In a DiffServ environment, the measurement methodology must be aware of different service classes. Traffic-engineered networks may rely on the use of IP routing protocols for the establishment of IP routes as well as the use of MPLS for the establishment of explicit LSPs. MPLS provides a mechanism to control the forwarding of network traffic. Paths allow control over routing of traffic flows requiring specific QoS within a domain. IP engineered paths (i.e., IP routes, LSPs) are used to carry aggregate user traffic belonging to several SLSs with similar performance requirements. Traffic engineering algorithms should not operate at the level of individual packets, since collecting packet-level micro-flow-related statistics is prohibitively expensive and nonscalable. Instead, statistics should be gathered at the aggregated flow level. In DiffServ, the measurement functions should operate at the level of DiffServ PHBs and TE edge-to-edge paths that carry traffic of similar service classes.

Dispersing the Measurement Data Collection System at the Node Level

To support dynamic operation, the monitoring system must be able to capture the operational status of the network without generating a large amount of overhead data and

degrading network performance. The variety of data, the magnitude of raw data at the node level, and the necessary processing close to the measurement source necessitate distributed data collection, typically consisting of one monitoring engine per router. The distributed monitoring engines must have low impact on the performance of the router and minimal effect on network bandwidth, adopting a flexible event-driven reporting approach (see below). In addition, by distributing the monitoring entities a view of the entire network is possible.

Minimizing Information Exchange by Processing Raw Data Close to the Source

A common approach to reduce the monitoring overhead is to vary the polling and sampling frequency based on the state and characteristics of variables being monitored. In addition, processing, aggregating, sampling, or filtering the raw data into accurate and reliable statistics, and reducing the amount of data near the observation point (source) are key functions for scalable dynamic operation. However, processing raw data close to the source reduces the flexibility of post-analysis. The monitoring system should provide automatic threshold detection by using notification of events in addition to summarized measurement information. Therefore, the following two forms of measuring data must be considered.

Events: Event notifications can be employed to avoid overloading the network with unnecessary interactions between components requiring monitoring information and network nodes. Their granularity can be defined on a per-class (or PHB) level and for edge-to-edge routes only. Raw measurement data are collected in short timescales from internal variables using measurement probes and processed to yield a statistically "smoothed" *rate*. The latter is compared with preconfigured thresholds, and an event notification is generated when a threshold is crossed. Depending on the measurement timescale, triggering might be postponed for instantaneous threshold crossings until successive or too frequent threshold crossings are observed, meaning that the problem persists for a specified time interval. This ensures that transient spikes do not contribute to unnecessary events, and also reduces the number and amount of information transfers.

Statistics: In order to improve scalability, monitoring nodes aggregate measurement data into summarized statistics. Summarization is the integration of measurement data over specified periods. The granularity of summarization periods must be suitably chosen based on the requirements of the management entity that requires monitoring information (i.e., the monitoring client). The granularity of statistics range from PHB and route level for TE functions to flow levels per SLS for customer service monitoring. Statistics should be provided in near real time to time-critical monitoring clients. Records of statistical information can be queued and multiple records can be exported in a single packet reducing the number of information transfers when there is no need for timely responses.

Utilizing Aggregated Path-Level Performance Measurements in Combination with Per-SLS Traffic Measurements

In order to ensure conformance to SLSs and detect violations, a service provider needs to gather QoS performance and traffic related measurements from multiple network nodes. For a network carrying a large number of flows, the flow/SLS level measurement information exchanged between network nodes and the monitoring system could be substantial. In addition, measurement granularity can be related to SLSs since not all of them have the same requirements. Generally, SLSs that belong to a premium class require measurement results with higher frequency, but monitoring SLSs at different levels of granularity by using different sampling frequencies makes the monitoring system far more complex. We propose to gather active performance information (e.g., delay, loss, delay variation) only at the path level, not at the micro-flow level per SLS. As several SLSs may use the same edge-to-edge route, a single performance monitoring action will suffice for all of them. Such aggregation and refinement of monitoring information can reduce the amount of information exchange. Thus, SLS monitoring is scalable provided that aggregate network performance measurements at the edge-to-edge path level are used in combination with per SLS ingress/egress traffic measurements (e.g., offered load, throughput). Passive traffic measurements have to be performed at the flow level per SLS at ingress/egress points of the network for the purpose of individual customers' traffic monitoring. Path-level performance monitoring is used for both service assurance and dynamic TE purposes (Table 1).

Reducing Synthetic Traffic Injection Using Hop-by-Hop Instead of Edge-to-Edge Measurements

Two distinct methods, edge-to-edge and hop-by-hop, may be used for performance measurements. Monitoring between two edge nodes for edge-to-edge measurements or between two neighboring nodes for per hop measurements can be used in order to determine the status of the attached links, interfaces, and associated queues. Monitoring scalability could be a serious concern with an MPLS-TE approach. If a full mesh logical network path is in place, an order of $O(N^2)$ unidirectional LSPs need to be monitored where N is the number of edge nodes. In fact, it is even worse than $O(N^2)$ since more LSPs may be established as multiple paths are used for load balancing and different services use different LSPs between an ingress-egress node pair. LSP monitoring is scalable and feasible only if a limited number of LSPs are selected for edge-to edge measurements based on specified criteria and policy decisions.

An active monitoring agent attached to a per-node monitoring engine is used to inject synthetic traffic. The edge-to-edge method directly provides edge-to-edge measurement results. The hop-by-hop method overcomes the scalability problem using per service class or PHB level measurements to calculate the edge-to-edge result. There are many different edge-to-edge paths that are routed through the same link and therefore share the resources of the same class on a single interface. Introducing synthetic traffic sent to quantify the behavior of this particular interface satisfies in part the performance monitoring requirements of all the paths using that hop. This results in significant reduction of synthetic traffic in the network. In addition to the scalability benefit of the hop-by-hop method, when problems such as large end-to-end delay or loss are observed, locating the contributing interfaces to this large delay/loss is straightforward.

Using the hop-by-hop method, the edge-to-edge one-way delay (EE_{OWD}) is additive and one-way packet loss ratio (EE_{OWL}) is multiplicative, as shown below:

$$EE_{OWD} = \sum_{i=l}^{n} d_i$$

$$EE_{OWL} = (1 - \prod_{i=l}^{n}(1 - p_i))$$

Metrics	Measurement Mechanism (Active/Passive)	Events and Measurement Statistics for:			
		Service Assurance (SLS Types & Services)			Resource Mgt. & Traffic Engineering
		Real-time Services	*Guarantee data (VPN) Services*	*Best-effort*	
• Performance		Per Path	Per Path	Per Path	Per PHB/Path
One-way delay	IPPM (A)	√	—	—	√
IP packet delay variation (ipdv)	IPPM (A)	√	—	—	√
One-way packet loss	IPPM (A)	√	√	—	√
• User Traffic Flow		Per SLS	Per SLS	Per SLS	Per macro-flow
Throughput per SLS/flows at egress	Flow-based (P)	√	√	√	√
Offered load per SLS/flows at ingress	Flow-based (P)	—	—	—	√
Packet rate per SLS/flows at egress	Flow-based (P)	—	—	—	—
• Network Workload					Per PHB/LSP
Throughput per PHB per link	COPS-PIB/ Metering (P)				√
Throughput per LSP	Flow-based (P)				√
Packet rate at PHB and LSP					—
Packet Discards per PHB per link	COPS-PIB/ SNMP MIB (P)				√
Link utilization In/Out	SNMP MIB (P)				(Per Link) —
• Availability					
Link & device	ICMP (A)				√
"A" is for Active and "P" is for passive measurements. √ necessary measurement. — desirable and potentially useful measurement. SLS types, services are explained in [1] and [16].					

■ Table 1. *Measurement data for service monitoring and traffic engineering functions.*

d_i and p_i are one-way delay and one-way packet loss ratio (probability), respectively, measured for hop i and its associated link, and n is the number of hop-by-hop values measured along the edge-to-edge route. It is assumed that the losses are independent and not correlated with respect to EE_{OWL} calculation. The loss measures are in fact the measuring events on different probability spaces. The assumption of independence might not hold if the losses occurred because of an overload in traffic that could cause correlated losses.

For some QoS metrics, such as delay and spatial composition, adding hop-by-hop measurements along the path in order to find an end-to-end equivalent result is not absolutely accurate [17]. In these cases the end-to-end result of the composition is only an estimation of the actual end-to-end value. In most cases the estimation error is not analytically quantifiable, which means that the end-to-end composed metric is not the same as the actual end-to-end one. This is why, for example, the definition of one-way delay does not include spatial composition, since the composed delay is a new metric reflecting the estimated one-way-delay.

In these cases we could still use the hop-by-hop measurements to induce an edge-to-edge result under two assumptions. Either the composition error is negligible and thus does give approximately the edge-to-edge result; or the network administrator could run benchmarking experiments to empirically deduce the difference of the two measurements, and then use these results to infer the actual edge-to-edge result from the composition of the hop-by-hop measurements. For example, conducting several benchmarking experiments may reveal to the network administrator that for a particular QoS metric, the difference of actual edge-to-edge measurements from the equivalent composition of the hop-by-hop ones over the same path is of a particular order. Later estimations of that QoS metric could rely only on the composition of hop-by-hop measurements, and use the average order of the difference as found in the benchmarks as an engineering parameter to fine-tune the result of the composition. The above example shows how an administrator can determine the difference of edge-to-edge from the composed hop-by-hop measurements, in order to eliminate the need to always use the edge-to-edge one. Other engineering alternatives may exist but are beyond the scope of this article.

Later we describe our benchmark experiments for measuring the actual difference of edge-to-edge vs. the equivalent composed hop-by-hop for both one-way delay and packet loss in a testbed. In these experiments we show that the difference is negligible and dependent mainly on the number of hops in the path.

Synchronizing the Hop-by-Hop Measurements — Performance measurements must be carried out in a relatively synchronized manner, and information must be retrieved from distributed measurement points when the hop-by-hop method is employed. This means that we should monitor the traffic flow at the same time interval as the flow is "seen" by each hop along the path. Thus, in order to achieve accuracy, per-hop measurements must be carried out in a relatively synchronized manner so that extrapolating to calculate the edge-to-edge result yields almost the same value as would be obtained if the edge-to-edge method were used. Therefore, mechanisms for performing, recording, retrieving, and consolidating measurement information are needed to ensure the integrity and validity of measurements.
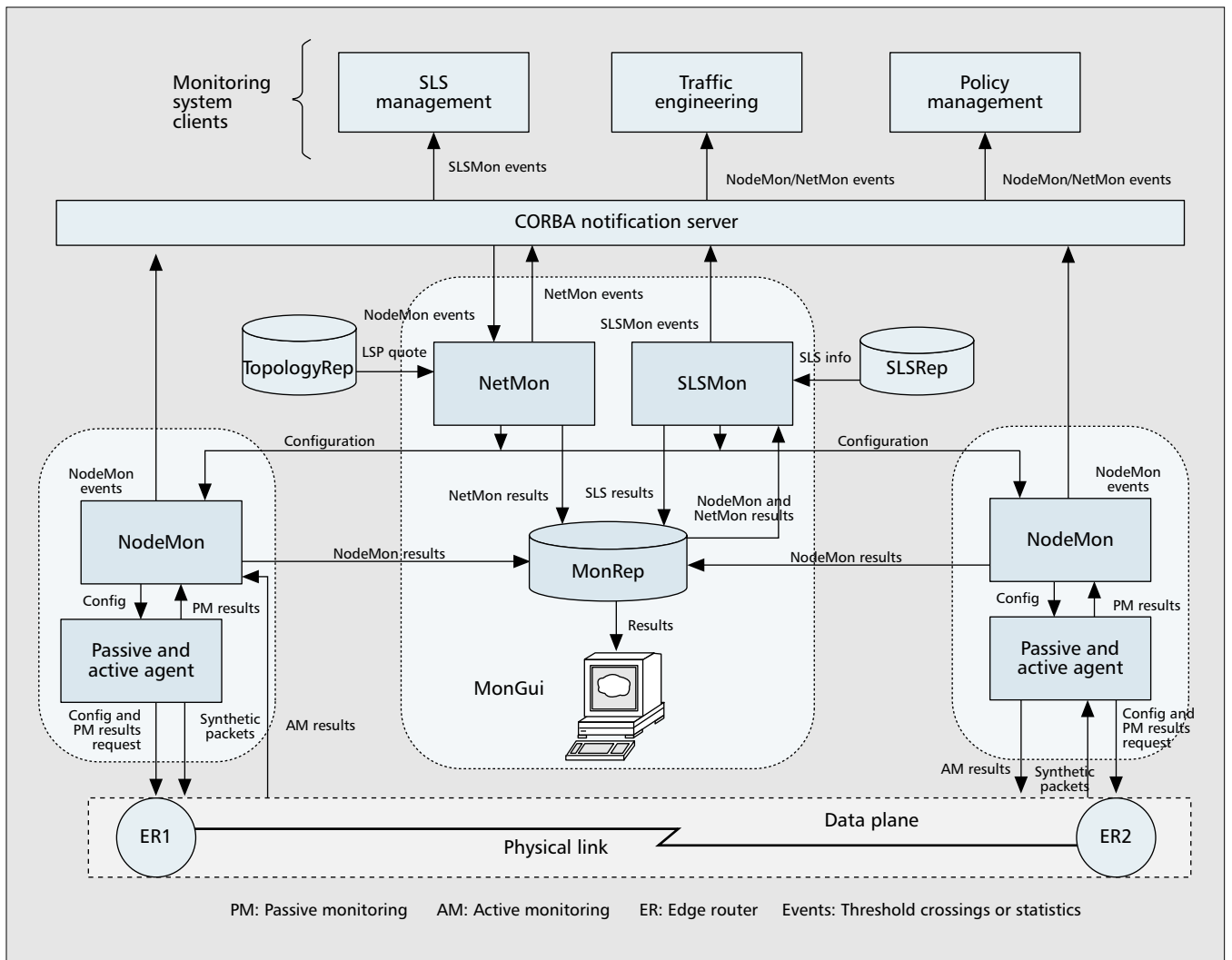
■ Figure 1. *The proposed monitoring system architecture and interactions with other subsystems.*

Limiting the Side-Effects on Network Performance by Controlling the Amount of Synthetic Traffic Insertion

Even when applying the principles described above, we may still need to further control the amount of synthetic traffic injected into the network for monitoring purposes. The requirements for the insertion of synthetic traffic are listed below:

- The synthetic traffic load should be small compared to the load on the connection under test. If not, the synthetic traffic will affect performance, and the measurements will be inaccurate.
- The sampling period should be small enough to study performance fluctuations.
- As the network changes over time, the amount and type of synthetic traffic should be configurable.
- The measurements should be randomly distributed to prevent synchronization of events as described in the IPPM recommendation [17] by using a pseudo-random Poisson sampling rate.

It should be noted that there is a trade-off relationship between the first two requirements. That is, smaller time intervals mean more synthetic traffic, but more synthetic traffic means a higher load on the network. It is common practice for network providers to keep the overhead produced by synthetic traffic in their network approximately below 1 percent of total network capacity.

A Scalable Monitoring System

There have been attempts to build network management and control systems that support traffic engineering and service differentiation [16, 18]. Here, we describe an intradomain QoS monitoring system for TE DiffServ networks that was designed based on the principles described in the previous section. Our monitoring system is tightly coupled with the overall system presented in [16] that includes *SLS management*, *TE*, and *policy management* subsystems in addition to *monitoring*.

A high-level view of the monitoring system, its components, and the interactions with the rest of the control and management system are shown in Fig. 1. In the following, we briefly describe the monitoring system components, while the detailed architecture and implementation issues are reported in [19]. The monitoring system has the following components:

1. **Node monitor** (NodeMon) is responsible for node-related measurements; there is one NodeMon per router. Node-Mon is hosted outside of the router on a dedicated computer, as the availability of required measurements is limited in currently available commercial routers. NodeMon is able to perform active measurements at the path or hop level, as well as passive monitoring of the router to which it is attached. NodeMon regulates and abstract various types of measured data. A NodeMon performs some short-term evaluation of results in addition to threshold crossing detection and notification.

2. **Network monitor** (NetMon) is responsible for network-wide post-processing of measurement data using a library of statistical functions. It is centralized and utilizes network-wide performance and traffic measurements collected by all the NodeMon entities in order to build a physical and logical network view (i.e., the view of the routes that have been established over the network). The analyzed data are mainly used for non-real-time proactive control of the network. Relatively frequent modifications to monitoring processes are required to accommodate any topological changes.

3. **SLS monitor** (SLSMon) is responsible for customer-related service monitoring, auditing, and reporting. SLSMon is centralized, since it keeps track of the compliance of the level of service provided to the customer of a domain. It utilizes information provided by NetMon and/or the various edge (ingress/egress) NodeMons.

4. **Monitoring repository** (MonRep) consists of two major parts for data cataloguing: a *data store* with database functionality for storing large amounts of data from monitoring components, and an *information store* for storing smaller amounts of configuration type information and information about active monitoring processes. Measurement data stored in the data store are used for subsequent analysis via the graphical user interface (GUI), NetMon, or SLSMon.

5. **Monitoring GUI** (MonGUI) is used to display measurement results and can be used in a network operations center. MonGUI presents a user interface allowing human operators to request graphical views of monitoring statistics extracted from the monitoring data store.

Various parts of the SLS management, TE, and policy management subsystems that require monitoring information must request information from one of the monitoring system components. In addition, parts of the monitoring system itself require some monitoring information. For example, SLSMon uses information from NodeMons or NetMon, and NetMon uses information from NodeMons. We collectively refer to all the components and subsystems requiring monitoring information from parts of the monitoring system as *monitoring clients*.

The monitoring system shown in Fig. 1 has been implemented in a modular fashion using an object-oriented approach. The monitoring system defines a set of common object request broker architecture (CORBA) interfaces to internal monitoring components for communicating with one another and to external components. The CORBA notification service has been used for delivering monitoring information to clients. Note that the CORBA notification service is centralized. The number of events pushed to the notification service is similar for all nodes; hence, the total event throughput processed by the notification server grows linearly with physical network size. For very large networks, the performance of the node hosting the notification server and the bandwidth of links around it could be limiting factors. Alternative solutions are to have a federated notification system, with one notification server per domain of nodes or even per node. In this case, monitoring clients will have to register with all the notification servers, which should share the overall event throughput, resulting in better scalability. The full monitoring system implementation details are presented in [19].

Within the scope of the proposed *Monitoring system*, a generic adaptation layer is used to provide two functions. The first is access to router statistics for passive monitoring. The second is installing traffic classifiers in the network elements to direct synthetic traffic to the required LSP/PHB. This generic software layer is designed to isolate a system's components from the type of routers actually used in the network.

Table 1 summarizes the data that can be monitored by the proposed monitoring system for both service-level assurance and resource management purposes. The proposed Monitoring system provides active performance measurements at the aggregated-flow (e.g., PHB, LSP, and IP route) granularity in addition to passive traffic measurements at the aggregated-flow micro/macro-flow (e.g., per SLS) granularities.

For performance measurements (delay, loss, jitter), synthetic traffic is injected into the network and the network behavior is observed. A pre-defined pattern of Type-P [17] synthetic packets is sent between two NodeMons. The time difference between receiver timestamp and sender timestamp is calculated as one-way delay. Synthetic packets are also sequence numbered. One-way packet loss is calculated from the packets that fail to arrive at the destination within a reasonable amount of time. The IP packet delay variation (ipdv) is measured by computing the difference of one-way delay of selected packets. Alternatively, it is also possible to calculate jitter as the absolute value of the difference between the interdeparture time of two subsequent Type-P packets at the NodeMon located at the ingress point of a path and the interarrival time of those two packets at the NodeMon located at the egress point of that path. For traffic measurement, continuous per-hop passive traffic monitoring is required for both service assurance and resource management. For traffic monitoring, we rely on the measured information and its granularity obtainable from the routers. The traffic measurement data is collected by routers and passed to the attached NodeMons for further analysis. Core routers collect traffic measurements on an aggregated basis at the PHB level. Ingress/egress routers collect traffic measurements (offered load and throughput) at both aggregated-flow and micro/macro-flow levels. Offered load is defined as the bit rate (in bits per second) at which the traffic that belongs to a flow is accepted by the network for delivery, measured at the ingress point. Throughput is defined as the bit rate at which the traffic that belongs to a flow is delivered by the network, measured at the egress point.

As the speed of network links increase, ingress/egress routers may not be able to perform full micro-flow measurement recording and will have to use sampling techniques to keep up with the speed. For full micro-flow measurement recording, costly high-speed dedicated probes (e.g., SCAMPI adapters) may be required. Core routers are only involved in aggregated-flow-level monitoring where there is no scalability concern. But in high-speed networks, traffic monitoring (e.g., throughput measurements at the egress router) at the micro-flow level for a large number of customers can cause scalability concerns.

Support for Resource Management

NodeMons collect information on the various classes at the PHB level and routes. NetMon deduces an end-to-end performance view by analyzing per-class and route-related measurements. During initialization, the TE subsystem informs NetMon of all the routes, the class of service associated with each route, and the associated PHBs that need to be monitored. NetMon has access to a topology repository (TopologyRep in Fig. 1) for retrieving topology-related information. While the network is in an operational state, the monitoring system performs the requested measurements and informs TE components about the occurred events (e.g., threshold crossings). Threshold crossings may trigger reconfiguring resources or redimensioning the network. PHB QoS performance measurements are used for managing link bandwidth and buffer space. The TE subsystem also needs to know the measured performance of the various routes in order to perform route management, load balancing, and dimensioning.
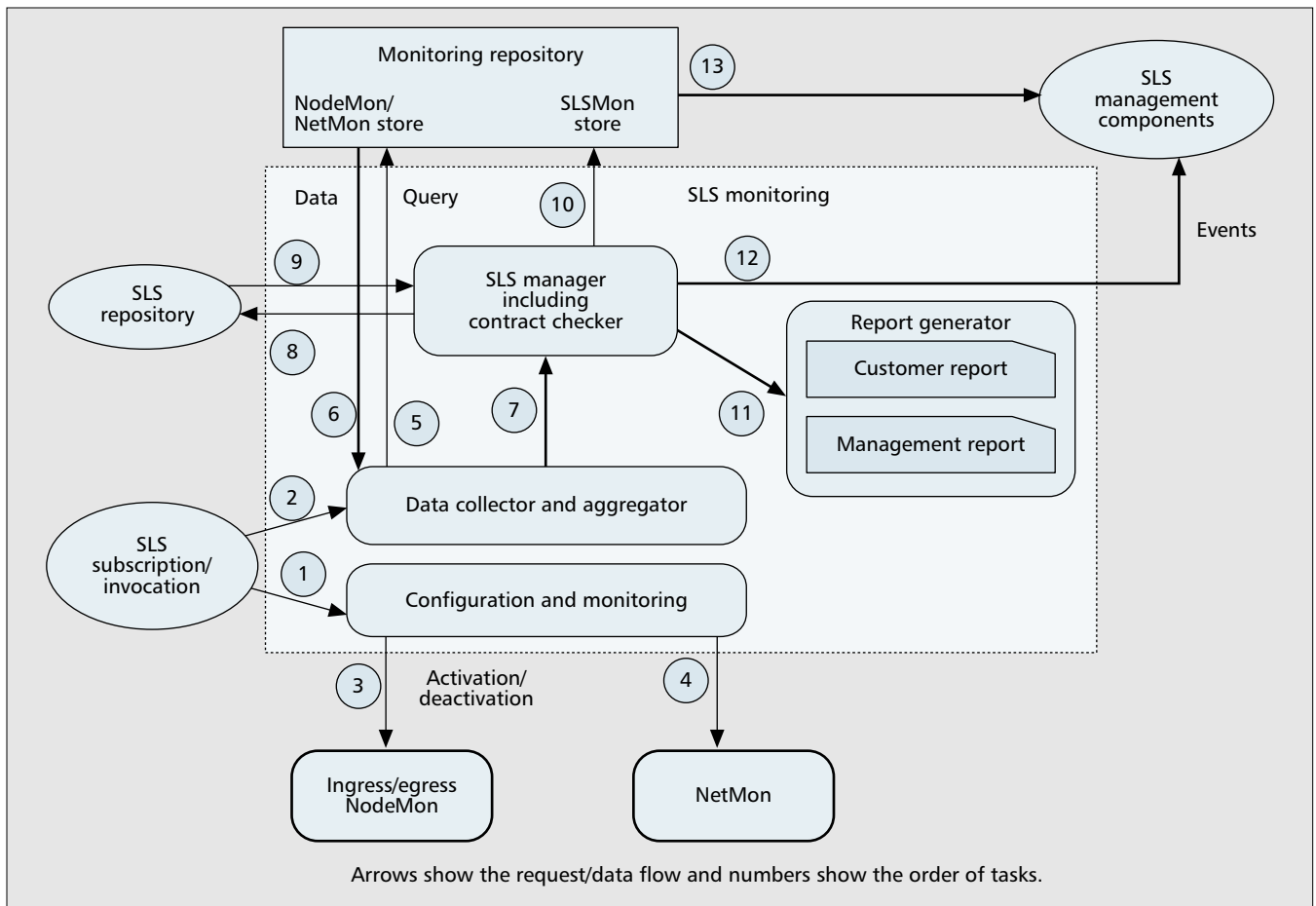
■ Figure 2. *SLS monitoring functions and interactions.*

Support for Service Assurance

SLS management, as the main client of SLSMon, requests the creation of necessary monitor instances when a customer SLS is invoked. SLSMon handles these requests for monitoring activation or deactivation. SLSMon acts as a client to NodeMons and NetMon. For scalability reasons, SLSMon was designed as a separate component rather than integrated with NetMon. Hence, it is hosted on a different platform from NetMon to avoid excessive load.

SLSMon retrieves SLS related information (e.g., SLS scope, i.e., ingress-egress points) from the SLS repository (SLSRep in Fig. 1). When an SLS is invoked, a specific route is used to forward the related traffic. SLSMon needs to obtain performance-related information (one-way delay, packet loss) on this specific route from NetMon and traffic-related information (throughput) from ingress/egress NodeMons. In order to make the system more scalable, if NetMon has already instructed the ingress/egress NodeMons to measure one-way delay and loss on the route/LSP to be used by this SLS, it simply uses the measurement information already available in the MonRep. During its operation, SLSMon accesses the MonRep for measurement data collected by NetMon and NodeMons. and combines the data for each individual SLS (i.e., path-level performance-related measurements and SLS-specific traffic related statistics). Since each service type has certain requirements, different metrics may need to be measured for each service type, as shown in Table 1.

SLSMon functions and its interactions with external components are shown in Fig. 2. SLSMon functions include the following. *Configuration and monitoring* handles the activation and deactivation requested by SLS subscription/invocation, and configures and activates the appropriate ingress and egress NodeMons and NetMon. A *data collector* accesses MonRep for measurement results collected by ingress/egress NodeMons and NetMon. The *SLS manager* detects SLS violation. The contract checker in the SLS manager checks each contracted SLS's performance and traffic-related values against measurement data, and activates the Report Generator and informs the SLS management subsystem of any performance violation. The measured values and the result from the contract checker are stored in the MonRep.

Evaluation of the Proposed Monitoring System

In this section we test the performance of the monitoring system described above in terms of measurement accuracy and scalability, and the ability to support per-class TE.

The assessment is based on experimental results obtained on a testbed (Fig. 3) consisting of four commercial routers connected through three 2 Mb/s serial links in a linear fashion (i.e., edge router ER1 is connected to core router R1 via link 1; core router R1 is connected to core router R2 via link 2; and core router R2 is connected to edge router ER2 via link 3). All routers have DiffServ capabilities for traffic classification, traffic conditioning, and various scheduling disciplines. For the scheduling disciplines, Low Latency Queuing (LLQ) together with Class-Based Weighted Fair Queuing (CBWFQ) are used. LLQ supports strict priority queuing together with the CBWFQ. Fair queuing provides a queue for each traffic flow and services these queues in such a way as to ensure that
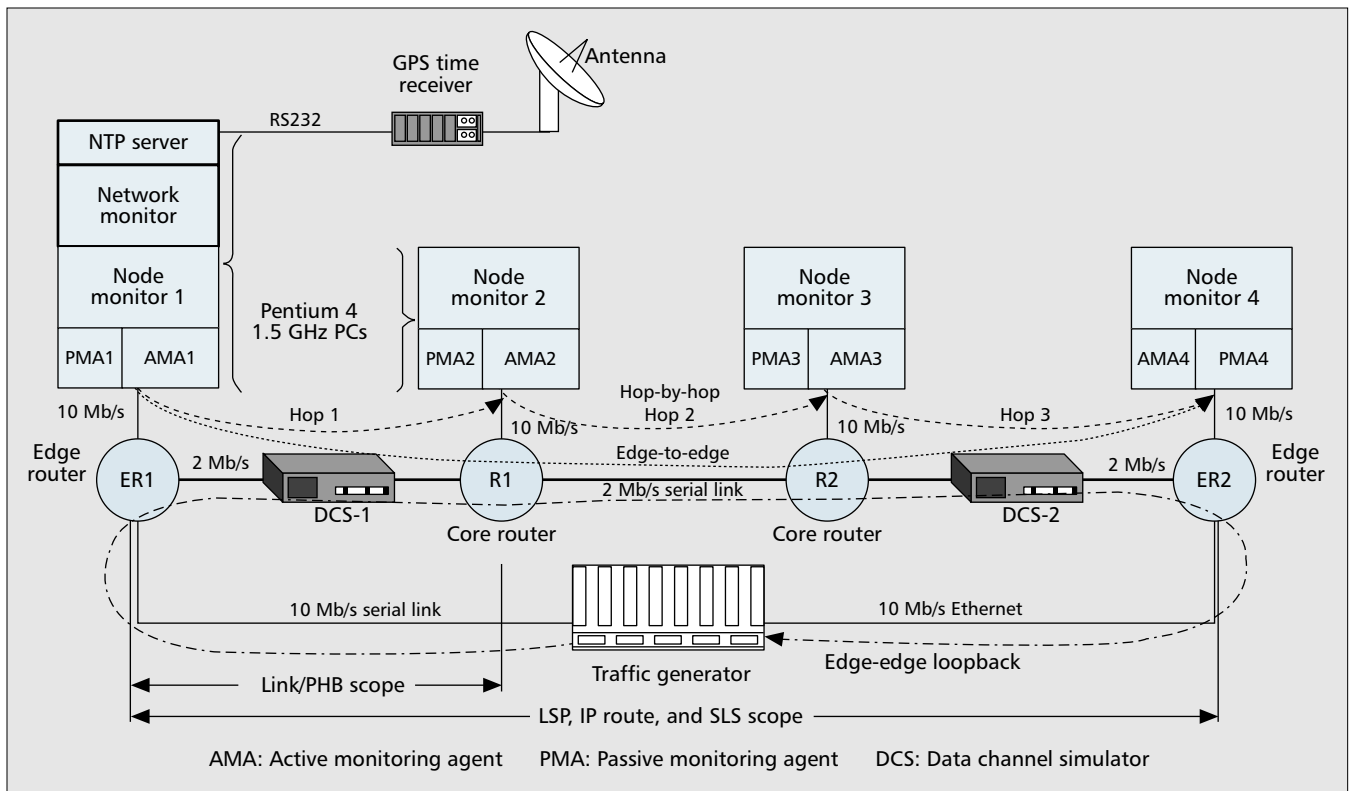
■ Figure 3. *The testbed configuration.*

no queue gets more than its fair share of available bandwidth during congestion. Weighted Fair Queuing (WFQ) places a weight on each flow. This allows the share of the bandwidth to be biased in favor of high-priority flows. CBWFQ extends WFQ functionality in order to support the configuration of user-defined traffic classes. In CBWFQ, a separate queue is associated with a defined traffic class. CBWFQ allows the exact amount of bandwidth to be specified for each traffic class.

A Pentium 1.5 GHz PC is attached to each router and hosts the node monitor with the PC attached to edge router ER1 also hosting the network monitor (note that NetMon is typically located in the network management center, but in this case it was attached to an edge router for convenience). Two data channel simulators (DCS) are used to introduce delays and independent losses into links 1 and 3. A commercial traffic generator is connected to both edge routers ER1 and ER2, and is used to generate synthetic traffic. The delay results measured by the traffic generator and the packet loss results programmed by the two DCSs are used to verify the results measured by the monitoring system.

Clock synchronization is essential for one-way measurements. It is important to use an accurate reference time clock. We encountered some problems in synchronizing the routers for getting accurate one-way delay results. The NTP settings were slightly inaccurate even for a less than 5 percent utilized network, yielding "jittery" one-way delay results. We had to dedicate a separate Ethernet segment for NTP traffic connected to all the PCs through second Ethernet cards. This was necessary in order to ensure that NTP traffic was not subjected to the network load, packet loss, and delay introduced by the DCSs. Thus, the NTP server is time-synchronized via a GPS time receiver and used to synchronize the NodeMons. In a real-world environment, a practical solution could be to have a dedicated NTP server synchronized with a GPS time receiver and a dedicated NTP Ethernet segment at every location that hosts a number of routers.

Measurement Accuracy and Scalability

Accuracy of monitoring systems is very important since the network operation relies on information that has to be accurate and reliable.

In a first series of tests, reported in [19], we addressed the one-way delay and packet loss from ER1 to ER2. The delay values measured by the monitoring system were very close to the ones measured by the traffic generator. This verified very good accuracy of monitoring results, even in configurations in which the monitoring agents are located outside the routers. We observed similar behavior with respect to packet loss. Programming 3.1 percent mean packet loss in DCS-2 at link 3 results in 3.28 percent mean packet loss measured by the monitoring system. As synthetic traffic is used for performance measurements, this result shows a very good approximation of packet loss measurements.

In a subsequent set of experiments, results of which we present below, we want to test the accuracy together with scalability by comparing edge-to-edge vs. hop-by-hop one-way delay and packet loss measurements. In these cases, measured edge-to-edge delay and packet loss were compared to aggregated values produced by NetMon based on per hop measurements.

Figure 4 shows the delay results. The mean difference between the edge-to-edge and aggregated hop-by-hop result is 1.1 ms. This small overestimation of delay was observed in all experiments and could result in a TE system that is slightly conservative. Note that underestimation would have more severe effects than overestimation when stringent delay constraints are required. The delay difference between hop-by-hop and edge-to-edge measurements is mainly dependent on the number of hops and is attributed to the fact that synthetic traffic crosses Ethernet segments, and more measurement processing is required in the hop-by-hop approach. It should be noted that longer end-to-end delays (congestion) do not introduce larger differences between the two approaches as

Accuracy test: one way delay hop-by-hop and edge-to-edge
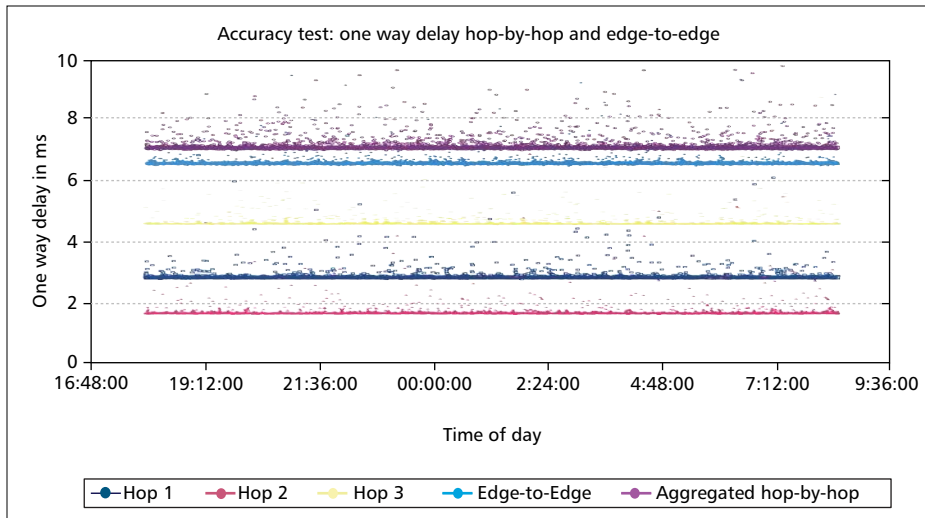
■ Figure 4. *Edge-to-edge and hop-by-hop one-way delay results.*

long as the number of hops remains the same. Hop-by-hop measurements are also transferred to NetMon for calculating edge-to-edge results.

If the active monitoring agents were embedded in the routers, the difference in delay measurements between the hop-by-hop and edge-to-edge methods would have been considerably reduced. Overall, we can state that comparable results are obtained by both methods, making hop-by-hop measurements more attractive because of the resulting scalability. Network administrators could judge the trade-off between accuracy and scalability gains and, for example, use edge-to-edge measurements for longer paths. We expect that this trade-off will be minimized when routers inherently support the monitoring functionality and appropriate processing priority is given to monitoring tasks in order to avoid yielding incorrect results.

Figure 5 shows the packet loss results: one-way packet loss experienced over each hop, edge-to-edge, and aggregated hop-by-hop. The average measured results were 5.26 percent for edge-to-edge, 2.04 percent for hop 1 on link 1, 0.0 percent for hop 2 (there is no DCS on link 2), 3.19 percent for hop 3, and 5.16 percent for aggregated hop-by-hop. The difference of 0.1 percent is negligible and can be attributed to rounding errors. Overall, the hop-by-hop method gave comparable results to edge-to-edge, having the advantage of enhanced scalability. Similar to the results on delay, from our experiments we observed that the number of hops is the only factor behind differences in hop-by-hop vs. edge-to-edge measurements. In a similar manner, for engineering service classes that require very accurate packet loss results on very long routes, administrators could use the edge-to-edge method instead of relying on the hop-by-hop approach. As discussed above, the differences between the two approaches are expected to monotonically decrease as routers include monitoring functionality instead of relying on external processors, as we did in our testbed.

Finally, we should state that system administrators could quantify through experimental measure-

ments the added inaccuracy introduced for each additional hop in a path. They could then use their estimations to deduce more accurate results from hop-by-hop measurements while preserving their inherent scalability properties.

Measurement-Based Dynamic Traffic Engineering

The experiments described in this section demonstrate the ability of our monitoring system to support online and dynamic per-class TE. We provide two example scenarios for measuring performance:
a) Of multiple traffic classes on a single interface
b) Simultaneously on multiple end-to-end tunnels (LSPs) carrying traffic from various classes

Online TE and dynamic resource management processes that optimize network utilization and performance require as input the performance-related events and statistics we gather in these cases. The results from the first scenario above could be utilized by an online process that dynamically changes the parameters (bandwidth, weight, and buffer size) of each class based on their measured performance and resource utilization (e.g., [20]). Measurement results from the second monitoring scenario could be utilized by LSP head-end load balancing processes that, depending on the performance of the various paths, (re)map traffic onto the best performing and least utilized LSPs (e.g., [21]).

For the experiments we configured in this section, we used the same testbed network shown in Fig. 3. For the purposes of our experiments we use three traffic classes, termed EF, AF1, and BE for differential treatment by the appropriate EF, AF, and BE PHBs implemented by LLQ along with CBWFQ. We configure the LLQ feature at the output interfaces of all routers. A physical queue is reserved for each class. With the LLQ feature, data belonging to the priority class (i.e., EF class) are served before packets from the other queues. In order to characterize a class, we assign bandwidth, weight, and maximum packet limit. The bandwidth assigned to a class is

One-way packet loss (edge-to-edge and hop-by-hop)
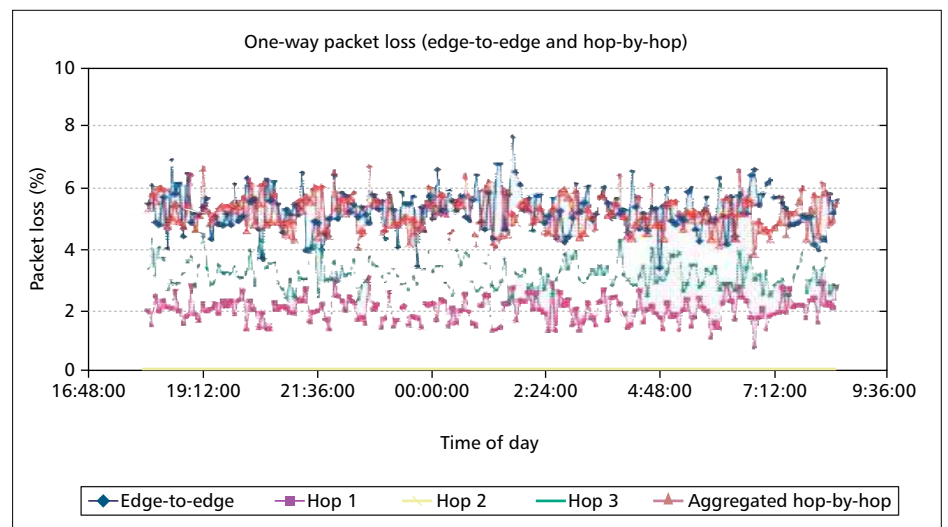
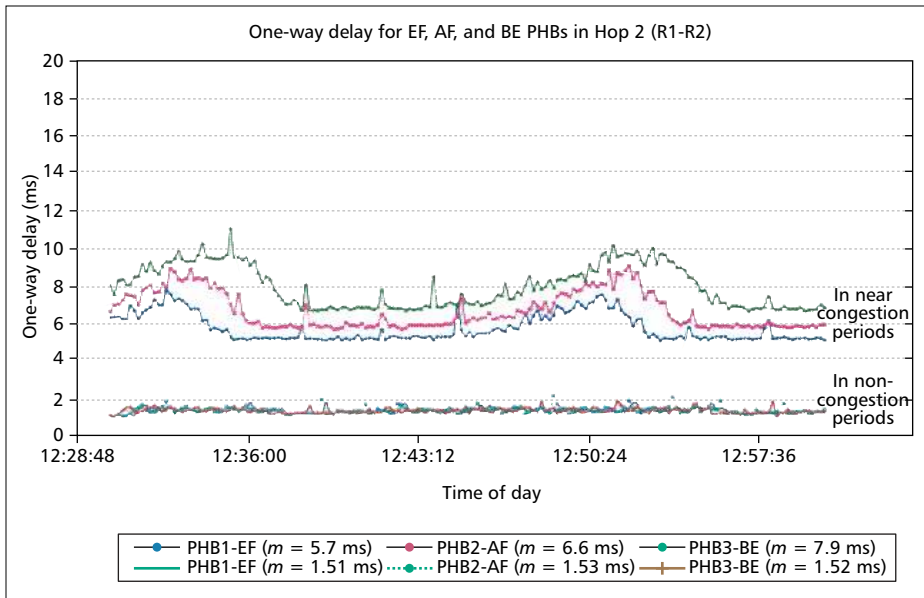■ Figure 5. *Edge-to-edge and hop-by-hop one-way packet loss results.*

Figure 6. *One-way delay for EF, AF1, and BE PHBs in Hop2 (R1-R2).*

the rate guaranteed to a class during congestion. EF traffic is directed to the priority queue, while AF1 and BE traffic use CBWFQ. The monitoring agents are configured for all three classes (EF, AF1, and BE).

We use the traffic generator to provide traffic sources marked as EF, AF1, and BE. We configured the packet interarrival times to be exponentially distributed, while the size of synthetic (test traffic) packets is configured to be of similar size to user traffic generated by the traffic generators. This ensures that the CBWFQ treats the synthetic and user traffic similarly; therefore, the delay measured for synthetic traffic is comparable to that experienced by user traffic.

In Fig. 6 we show the one-way delay, as measured by our monitoring system, packets experience at Hop2 (R1 to R2 in Fig. 3). We show the delay in two scenarios: during noncongested periods and when there is congestion. In the first case, when the network is not congested, the one-way per-hop delays shown in Fig. 6 are very low for all three traffic classes. The one-way per-hop delay comprises only transmission delay on the link's interface, which is around 0.51 ms, the link propagation delay, around 1.0 ms, and some negligible queuing delay. In the second case we measured the one-way delay results in Hop2 during congestion periods. As the network is congested, the one-way delay results in all three classes are higher than that experienced in a not fully loaded network. In addition to the transmission and propagation delay, the queuing delay is not negligible in this case, but we can see that there is a difference between the classes with EF having an average total delay of about 5.7 ms, AF1 about 6.6 ms, and 7.9 ms for BE.

Similarly, we have defined monitoring agents and can measure the packets lost, the average number of packets in a queue, and utilization of each of the classes. All the mon-

itoring results can be made available to dynamic per-class management and TE processes.

In the second set of experiments we looked at monitoring support for different edge-to-edge tunnels. We set up unidirectional LSP tunnels to carry traffic between the two edge routers ER1 and ER2 (Fig. 3). In the first test, the monitoring system (NodeMons) configured the appropriate active monitor jobs at the two endpoints to monitor one-way delay and packet loss experienced by the traffic using the LSP tunnels. Note that in this test we do not use the hop-by-hop approach to derive the end-to-end results, as described in the previous section. In order to measure the end-to-end one-way packet loss within a tunnel, the amount of BE traffic injected to the network from ER1 to ER2 is set to be higher than the amount of bandwidth reserved for the queue belonging to the BE PHB. Thus, we are able to map traffic onto the LSP in order to bring the link into congestion.

In Fig. 7 we show the measured one-way packet losses for the three end-to-end LSPs carrying EF, AF1, and BE traffic. The packet loss was very low in the tunnel carrying EF traffic and high in the LSP carrying BE traffic. The latter is due to the fact that we injected more traffic into the BE LSP than the bandwidth reserved for the BE PHBs.

The second test shows that the proposed monitoring system is capable of performing passive measurements. Three passive monitor jobs were set up to monitor throughput on three LSPs. Traffic sources were used to generate EF and AF1 marked traffic at 600 kb/s rates and BE traffic at 800 kb/s. The results are shown on Fig. 8. This figure shows that the average measured throughput for the EF, AF1, and BE LSP tunnels is about 600, 600, and 700 kb/s, respectively. This was expected as BE traffic was subjected to loss. It should be noted that one-way delay measured values are averaged in 2 s intervals, repre-
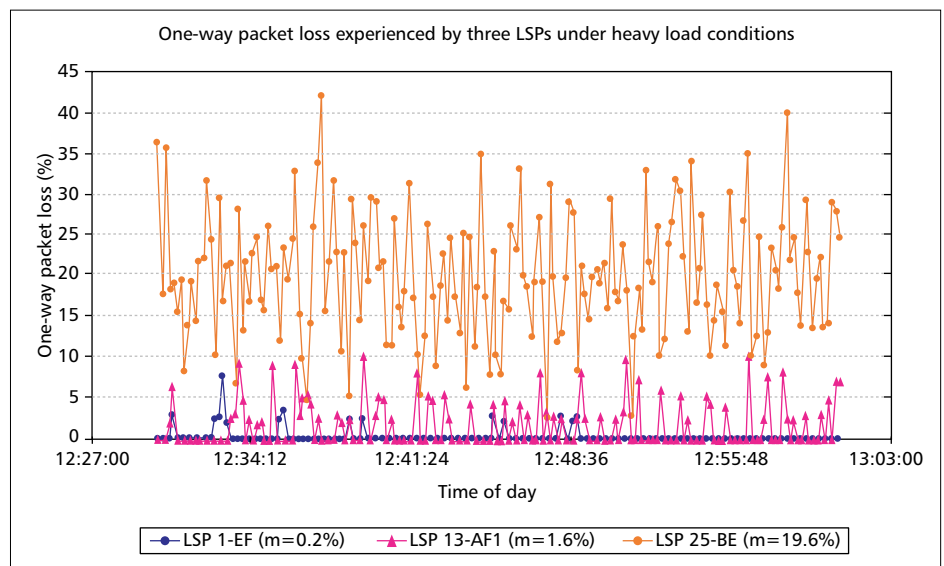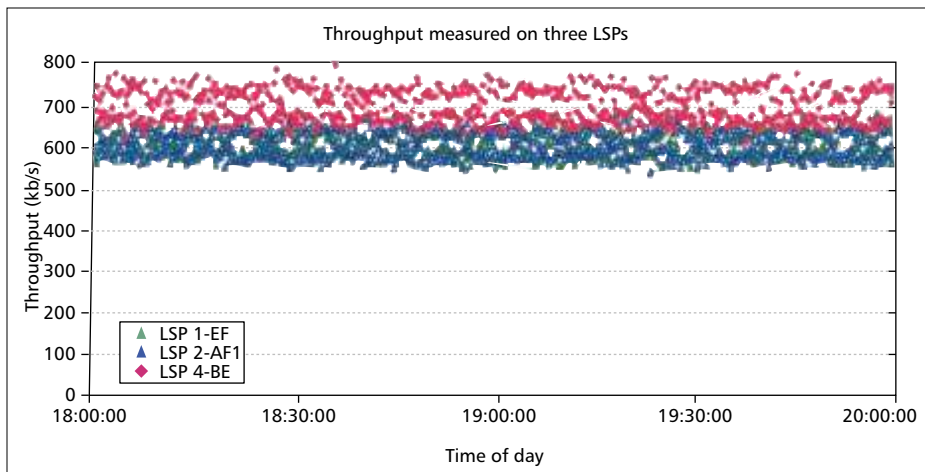

Figure 7. *Packet losses experienced by LSPs carrying EF, AF, and BE traffic.*

■ Figure 8. *Throughput measured on three different LSPs.*

senting a data point on Fig. 7. In addition, the readout period for retrieving the throughput values was 10 s, representing a data point on Fig. 8. These timescales are quite small, and the measurements can aid short-term TE processes.

Monitoring System Deployment

The proposed monitoring system is implemented in a modular fashion [19] in order to provide and visualize the measurement information collected from MPLS/IP TE networks. The monitoring system is distributed throughout the network, and has both per-node and centralized components. A number of hardware and software components are required to deploy the monitoring system in a provider's domain. A processor hardware component is required for each network element in order to host the per-node software components. This is due to the incapability of today's network elements to perform one-way delay and loss measurements and analysis. They are required at core routers if we want to perform per-hop measurements. They are also required at ingress and egress points in order to perform edge-to-edge performance and traffic-related measurements. A further computer platform is required to host the centralized software components, including network and SLS monitors, CORBA name and notification servers, MonGUI, database server, and NTP server. A database server is required for the MonRep and topology repository. The topology repository is used to populate the details of the network physical topology. It is also used to store the logical topology of a network in terms of computed TE paths and associated PHBs.

The proposed monitoring system provides passive measurements at PHB, LSP, and SLS granularity in addition to active measurements at the PHB, LSP, and IP route levels. These measurements are necessary to assist both offline proactive and dynamic reactive operation of TE DiffServ networks. Most commercial network management solutions currently support both MPLS TE and DiffServ capabilities. The proposed monitoring system can be thought of as a value-added feature to currently available solutions to provide the means of continuous optimization of network performance.

In the proposed monitoring system, the active measurement agents are located outside the routers, and could result in measurement inaccuracy and extra cost. In order to offer reactive TE solutions, it is beneficial for router manufacturers to integrate the following functionality within their routers:

• The capability of performing active one-way delay and packet loss measurements
• The ability of perform accurate time synchronization (in milliseconds) for one-way delay measurements

• Provide passive measurement information at finer granularity (i.e., at the PHB level) with configurable reporting periods

Summary

When delivering QoS-based value-added IP services, careful engineering of the network and its traffic are essential for efficiency of resource usage while meeting the required performance targets. Traffic engineering relies on measured data for offline proactive and dynamic reactive measures. In this article we first identify the measurement requirements for TE. We subsequently present requirements for a scalable monitoring system that gathers real-time data to reflect the current state of the network. We then present principles for designing scalable monitoring systems and methodologies for scalable event monitoring used for network operation and in-service performance verification. We finally present a scalable monitoring system designed and built based on those principles and its assessment.

The presented system is distributed in order to guarantee quick response times and minimize necessary management traffic. This ensures small reaction times and helps maintain scalability as the network size increases. Based on experimental assessment results, we show that the proposed monitoring system provides good accuracy for one-way delay and packet loss as well as highly comparable edge-to-edge and hop-by-hop results. We also show the ability of the proposed monitoring system to support dynamic TE. In summary, we believe that the presented principles result in scalable monitoring systems that can contribute to operationally optimized traffic engineered networks that can support a large number of customers.

References

[1] D. Goderis *et al.*, "Service Level Specification Semantics and Parameters," Internet draft, draft-tequila-sls-02.txt, Expired Aug. 2002.
[2] S. Blake *et al.*, "An Architecture for Differentiated Services," RFC 2475, Dec. 1998.
[3] For information on the various IETF working groups including DiffServ, IPPM, IPFIX, RTFM, and PSAMP, visit http://www.ietf.org/
[4] Test Traffic Measurements (TTM) project of Réseaux IP Européens (RIPE) Network Coordination Centre (NCC); http://www.ripe.net/ttm/
[5] V. Paxson *et al.*, "An Architecture for Large-Scale Internet Measurement," *IEEE Commun. Mag.*, vol. 36, no. 8, Aug. 1998, pp. 48–54.
[6] Network Analysis Infrastructure (NAI) projects of NLANR; http://mna.nlanr.net/infrastructure.html
[7] CAIDA) projects; http://www.caida.org
[8] C. Fraleigh *et al.*, "Packet-Level Traffic Measurements from the Sprint IP Backbone," *IEEE Network*, vol. 17, no. 6, Nov./Dec. 2003.
[9] IST research projects: http://www.cordis.lu/ist/; IST-INTERMON: http://www.ist-intermon.org/; IST-MoMe: http://www.ist-mome.org/; IST-SCAMPI: http://www.ist-scampi.org/
[10] A. Feldman *et al.*, "NetScope: Traffic Engineering for IP Networks," *IEEE Network*, vol. 14, no. 2, Mar./Apr. 2000, pp. 11–19.
[11] J. L. Alberi *et al.*, "Using Real-Time Measurements in Support of Real-Time Network Management," *RIPE-NCC 2nd Wksp. Active and Passive Measurements*, Amsterdam, The Netherlands, Apr. 2001.

[12] F. De Turck *et al.*, "Design and Implementation of a Generic Connection Management and Service Level Agreement Monitoring Platform Supporting the Virtual Private Network Service," *Proc. 7th IFIP/IEEE Integrated Management Symp.*, Seattle, WA, May 2001.

[13] Y.J. Lin, and M. C. Chan, "A Scalable Monitoring Approach Based on Aggregation and Refinement," *IEEE JSAC*, vol. 20, no. 4, May 2002.

[14] M. Dilman and D. Raz, "Efficient Reactive Monitoring," *IEEE JSAC*, vol. 20, no. 4, May 2002.

[15] A. Asgari *et al.*, "A Scalable Real-Time Monitoring System for Supporting Traffic Engineering," *Proc. IEEE Wksp. IP Ops. and Mgmt.*, Dallas, TX, Oct. 2002, pp. 202–97.

[16] P. Trimintzios *et al.*, "A Management and Control Architecture for Providing IP Differentiated Services in MPLS-Based Networks," *IEEE Commun. Mag.*, vol. 39, no. 5, May 2001.

[17] V. Paxson *et al.*, "Framework for IP Performance Metrics," IETF RFC-2330, May 1998.

[18] P. Aukia *et al.*, "RATES: A Server for MPLS Traffic Engineering," *IEEE Network*, vol. 14, no. 2, Mar./Apr. 2000, pp. 34–41.

[19] A. Asgari *et al.*, "Building Quality of Service Monitoring Systems for Traffic Engineering and Service Management," *J. Net. and Sys. Mgmt.*, vol. 11, no. 3, Dec. 2003.

[20] N. Christin and J. Liebeherr, "A QoS Architecture for Quantitative Service Differentiation," *IEEE Commun. Mag.*, vol. 46, no. 6, June 2003, pp. 38–45.

[21] A. Elwalid *et al.*, "MATE: MPLS Adaptive Traffic Engineering," *Proc. IEEE INFOCOM 2001*, Anchorage, AK, Apr. 2001.

Biographies

ABOLGHASEM (HAMID) ASGARI (Hamid.Asgari@thalesgroup.com) received his B.Sc. from Dr. Beheshti University in Tehran, M.Sc. (Honors) from the University of Auckland, New Zealand, both in computer science, and his Ph.D. in the design and analysis of ATM networks from the University of Wales, Swansea in 1997. He was with Iran Telecom Research Center (ITRC) from 1986 to 1990. He joined Thales (formerly Racal) Research & Technology, United Kingdom, in 1996 where he is an assistant chief engineer. He has been involved in simulation, design, and analysis of integrated IP services and network architectures. He has worked on the IST TEQUILA, MESCAL, and ENTHRONE projects. His main research interests are in network QoS management, QoS-aware monitoring, network and system-level performance evaluation, traffic engineering, and service management.

PANOS TRIMINTZIOS (P.Trimintzios@eim.surrey.ac.uk) holds a Ph.D. from the University of Surrey, United Kingdom, and received a B.Sc. in computer science and an M.Sc. in computer networks from the University of Crete, Greece, in 1996 and 1998, respectively. From 1995 to 1998 he was a research associate at ICS-FORTH, Greece, working on research projects involving high-speed network management and charging network and user services. Since 1998 he is a research fellow at the Center for Communication Systems Research (CCSR), University of Surrey. His main research interests include traffic engineering, QoS provisioning, network monitoring, network performance control, policy-based networking, and network and service management.

GEORGE PAVLOU (G.Pavlou@surrey.ac.uk) is a professor of communication and information systems at CCSR, Department of Electronic Engineering, University of Surrey, where he leads the activities of the Networks Research Group. He hold a Diploma in engineering from the National Technical University of Athens, Greece, and M.Sc. and Ph.D. degrees in computer science from University College London, United Kingdom. His research interests focus on network management, networking, and service engineering, covering aspects such as protocol performance evaluation, traffic engineering, quality of service management, policy-based systems, multimedia service control, programmable networks, and communications middleware.

RICHARD EGAN (Richard.Egan@thalesgroup.com) received a B.Eng. (Elect) from University College, Cork, Ireland in 1980. He worked for both GEC Telecommunications and Racal-Datacom on a variety of product developments, including the System X public switch. Since joining Thales (formerly Racal) Research &Technology in 1993, he has specialized in telecom and data communications system design, with particular emphasis on performance analysis. He is a chief engineer and leads the Ad Hoc Networks team, which specializes in research into self-organizing wireless networks. He also leads a team that participates in collaborative research programs and provides consultancy to other Thales companies. He has worked on IST Projects TEQUILA and MESCAL and he is Chairman of the Interworking of Networks Steering Group in the U.K. Mobile Virtual Center of Excellence. His main interests are in routing, QoS, and self-organization in next-generation wireless networks.