A Scalable Real-time Monitoring System for Supporting Traffic Engineering

Abolghasem Asgari†, Panos Trimintzios‡, Mark Irons†, George Pavlou‡, Richard Egan†, Steven V. den Berghe§

†Thales Research & Technology (UK) Ltd
Worton Drive, Reading, RG2 0SB,
United Kingdom
email: Hamid.Asgari@uk.thalesgroup.com

‡Centre for Communication Systems Research, University of Surrey
Guildford, Surrey, GU2 7XH
United Kingdom

§Inter-university Microelectronics Centre (IMEC), Sint-Pietersnieuwstraat 41, 9000 Gent
Belgium

***Abstract*— Quality of service based value-added services in IP networks necessitate the use of traffic engineering. The latter relies typically on monitoring data for both offline, proactive and dynamic, reactive solutions. 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 are also presented.**

Keywords*— *IP, Monitoring, Traffic Engineering, Differentiated Services, Active/Passive Measurements, Scalability

I. INTRODUCTION

Monitoring systems are becoming increasingly important for providing quantified Quality of Service (QoS) based services and service assurance. Traffic Engineering (TE) can be defined as a set of techniques that allow service providers to maximize network resource utilization, while at the same time meet 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 [1]. Traffic engineering algorithms need typically an overview of the network status for their dynamic reactions. The functionality that delivers this information is viewed as operational measurements.

In Differentiated Services (DiffServ) [2] networks, routers process aggregate traffic that belongs to several service classes according to predefined QoS policies. In this paper, we assume that the QoS requirements of a customer are described in a Service Level Specification (SLS) [3], which is the technical part of a Service Level Agreement (SLA) between the customer and the provider. By QoS, we refer to a service offering where one or more traffic and performance parameters (i.e., throughput, delay, loss, and/or delay variation) are quantified [3]. As the network attempts to offer several service types by employing traffic engineering mechanisms, e.g., hard real-time traffic, Virtual Private Networks (VPNs), best-effort services, etc., service monitoring is important for providing end-to-end QoS and service assurance. In this context, monitoring does not just have a diagnostic role but becomes an important tool for supporting the network operation and providing service auditing for both traditional and value-added services. Given the multitude of services with different QoS requirements, measurement information needs to be collected at a finer granularity than per ingress-egress node pairing.

This work was supported by the Commission of the European Union under the Information Society Technologies (IST) TEQUILA (Traffic Engineering for QUality of service in the Internet at LArge scale) project (IST-1999-11253-TEQUILA).

A. Related work

A large amount of work has gone into developing mechanisms and protocols for performance and traffic measurements. This includes the work of Internet Engineering Task Force (IETF) groups such as IP Performance Metrics (IPPM). The developments on monitoring systems have mainly been focused on path performance analysis. Relevant activities include the RIPE Network Coordination Center that has implemented a number of the measurement protocols defined by IPPM; the CAIDA (Cooperative Association for Internet Data Analysis) measurement environment that uses the Skitter tool for gathering QoS related data; and the NLANR (National Laboratory for Applied Network Research) Network Analysis Infrastructure, and NIMI (National Internet Measurement Infrastructure) [4] that combine active and passive measurements for network path monitoring.

In comparison, little work has been done to use measurement information for tackling network performance degradation and managing congestion in operational networks in real time. NetScope [5] provides a set of software tools for traffic engineering of IP backbone networks by using network measurements to update the network configuration set-up in non-real time fashion. Rondo [6] is an automated control system that uses a monitoring system to react to and manage congestion in MPLS (Multi-Protocol Label Switching) traffic-engineered networks in *near* real time. An intra-domain monitoring system for DiffServ networks is described in [7] that supports resource control and end-to-end QoS evaluation and validation of network services based on SLAs and traffic classes.

In summary, a large amount of information is needed for traffic engineering large operational IP networks and for service level monitoring for a large number of customers. Scalability is thus a big challenge for monitoring systems and necessitates suitable system architectures for achieving scalable monitoring in near real-time.

This paper addresses the scalability issues of monitoring systems, proposes principles for designing and assessing such systems and describes an example system designed and

assessed according to those principles. The presented system supports both real-time service level monitoring and also monitoring for traffic engineering support in MPLS-based DiffServ-capable networks.

II. MONITORING REQUIREMENTS

Traffic engineering is achieved through capacity and traffic management. These two are realized with capacity planning, routing control, resource management including buffer and queue management and other functions that regulate and schedule traffic flow through the network in order to accommodate as many customer requests as possible, while at the same time satisfying their QoS requirements. The state dependent traffic engineering functions require the observation of the state of the network through a monitoring system and applying control actions to drive the network to a desired state. This can be accomplished by reacting through control actions in response to the current state of the network and/or by pro-actively using forecasting techniques to anticipate future traffic demand and pre-configure the network accordingly.

A monitoring system should provide information for the following three categories of tasks:

1. Assist traffic engineering 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 for taking appropriate actions on setting up new routes, modifying existing routes, performing load balancing among routes, and re-routing traffic.
2. Assist traffic engineering 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 growth patterns and congestion indications.
3. Verify whether the QoS performance guarantees committed 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 different types of information [8]. In-service verification of traffic and performance characteristics per service type is required.

Traffic engineering must be viewed as 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 systems must be generic enough to cope with such changes.

III. MEASUREMENT DATA AND METHODS

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. Measurements include one-way delay, packet delay variation, one-way packet loss, traffic load and throughput. There exist two types of 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 require co-operation from both measurement end-points. In the case of measuring one-way delay, end-point clocks need to be synchronized. Therefore, methods like the Network Time Protocol (NTP) [9], Global Positioning System (GPS) or other Code Division Multiple Access (CDMA) based time sources, can be used.

Passive measurements are used to observe actual traffic without injecting extra traffic into the network. While passive measurements do not require co-operation of end-points, they require continuous collection of data and monitoring of links at full load; the latter can be problematic on high-speed links. In both cases, the quality of analyzed information depends on the granularity and integrity of collected data.

IV. PRINCIPLES FOR SCALABLE MONITORING SYSTEMS

Scalability in QoS-enabled IP networks 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, number of links, speed of links, degree of physical and logical connectivity, network diameter, etc. In IP QoS-enabled networks, supported services are mapped to a number of classes according to the DiffServ model; the latter has an impact on the scale of the monitoring system. A large number of subscribed customers requires subsequently a large amount of information to be gathered for service assurance.

The scalability of the monitoring system is the ability of effectively deploying a 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, data analysis, and 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 exist a number of per class states (e.g. different queues) per interface and a large number of routes per class that must be monitored. Hence, scalable monitoring architectures must adhere to the principles described below and are summarized in Table I.

A. Defining the monitoring process granularity

In a DiffServ environment, the measurement methodology must be aware of different service classes. Traffic engineering algorithms should not operate at the level of individual packets, since collecting packet-level micro-flow related statistics is prohibitively expensive and non-scalable. Instead, statistics should be gathered at the aggregate macro-flow

level. In DiffServ, the measurement functions should operate at the level of Per Hop Behaviors (PHB) and traffic-engineered paths carrying traffic of similar service classes.

TABLE I.PRINCIPLES FOR BUILDING SCALABLE MONITORING SYSTEMS.

Principle	Scope and Action to be taken
Defining the monitoring process granularity	*At DiffServ Per Hop Behavior and path level*
Distributing data collection system	*At node level*
Minimizing the measurement transmission overhead	*By employing event notification and summarization of statistics*
Using aggregate performance measurements in combination with per-SLS traffic measurements	*By carrying out performance measurements at path level and traffic measurements at SLS level*
Reducing the amount of synthetic traffic	*By using hop-by-hop measurements*
Controlling the amount of synthetic traffic	*By having a trade-off between synthetic traffic load and sampling frequency*

B. *Distributing the data collection system*

To support dynamic operation, the monitoring system must be able to capture the operational status of the network without generating a large amount of data and without degrading network performance. The variety of data, the magnitude of raw data at node level and the necessary processing close to the measurement source necessitate distributed data collection, typically comprising one monitoring engine per router. The distributed monitoring engines must have low impact on the switching performance of the router and must have minimal effect on network bandwidth, adopting a flexible event-driven reporting approach (see section C).

C. *Minimizing the measurement transmission overhead by processing the raw data close to the source*

Processing and aggregating the raw data into accurate and reliable statistics and reducing the amount of data near the source is key to scalable dynamic operation. The monitoring system should provide automatic threshold detection by using notification of events in addition to summarized measurement information. Therefore, two forms of measurement data must be considered.

Events: Event notification can be employed to avoid overloading the network with unnecessary interactions between components requiring monitoring information and network nodes. The granularity of event notifications can be defined for PHBs and paths. Raw measurement data is collected in short-time scales from internal variables using measurement probes and processed to yield a statistically "smoothed" *rate*. The latter is compared with a previously configured threshold and an event notification is generated if the threshold is crossed. Depending on the measurement timescale, the triggering might be postponed on instantaneous threshold crossings until successive/frequent threshold crossings are observed, meaning that the problem persisted for a specified time interval. This ensures that transient spikes do not contribute to unnecessary events.

Statistics: in order to improve scalability, monitoring nodes aggregate the measurement data into summarized statistics. The granularity of summarization periods must be suitably chosen based on the requirements of the interested management functional entity. The granularity of statistics range from PHB and route level for traffic engineering functions to the aggregated flow levels for customer service monitoring. Statistics should be provided near real-time to time-critical functional entities. 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.

D. *Using aggregate performance measurements in combination with per-SLS traffic measurements*

SLSs may not need to be monitored in the same way. Generally, SLSs that belong to a premium class require measurement results with higher frequency but monitoring SLSs at different levels of granularity with different sampling frequencies makes the monitoring system more complex. SLS monitoring is scalable provided that aggregate network performance measurements at path level (e.g., delay, loss, delay variation) are used in combination with per SLS ingress/egress traffic measurements (e.g., throughput). As several SLSs use a single edge-to-edge path, a single monitoring action is enough for all of them. As an example, injecting synthetic traffic from an ingress point toward an egress point on a specific path for measuring one-way delay can satisfy the requirement of multiple SLSs using that path.

E. *Reducing the amount of synthetic traffic by using hop-by-hop instead of edge-to-edge measurements*

Two distinct methods may be used for performance monitoring. Monitoring between two edge nodes for edge-to-edge measurements or between two neighboring nodes for hop-by-hop measurements in order to determine the status of the attached links, interfaces, and associated queues.

Monitoring scalability could be a serious concern when a full mesh logical network is in-place, an order of $O\left(N^2\right)$. Path 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 Node Monitor 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 by using per hop (i.e., a PHB and its associated link) measurements to calculate the edge-to-edge result. There exist multiple edge-to-edge paths, which are routed through the same PHB. When these paths traverse the same hop they share the resources associated with that PHB. Introducing synthetic traffic sent to quantify the behavior of that hop satisfies the performance

monitoring requirements of all the paths using that hop. This results in significant reduction of the required synthetic traffic. Using the hop-by-hop method, the edge-to-edge one-way delay is additive and the one-way packet loss ratio multiplicative

F. *Controlling the amount of synthetic traffic insertion*

Even when applying the "hop-by-hop" measurement principle described above, there is still a need to control the amount of synthetic traffic. The requirements for the insertion of synthetic traffic are listed below:

1. The synthetic traffic load should be small compared to the load on the connection under test. If not, then the synthetic traffic will affect the performance and the measurement will be inaccurate.
2. The sampling period should be small enough to study performance fluctuations.
3. As the network changes over time, the amount and type of synthetic traffic should be configurable.
4. The measurements should be randomly distributed to prevent synchronization of events as described in the IPPM recommendation [10] by using a Poisson sampling rate.

It should be noted that the first two requirements should be as complementary as possible. That is, smaller time intervals means more synthetic traffic, but more synthetic traffic means a higher load on the network. A trade-off between these two requirements is necessary for controlling the amount of synthetic traffic. Practically, the rule used by some network operators is that synthetic traffic should not exceed approximately 1% of the total network capacity.

V. AN EXAMPLE SCALABLE MONITORING SYSTEM

Here, we describe an intra-domain QoS monitoring system for traffic-engineered DiffServ networks that was designed using the principles described in the previous section. Recently, there have been attempts to build network management and control systems that support traffic engineering and service differentiation e.g. [8] and [11]. Our monitoring system is tightly coupled with the system presented in [8] that includes *SLS Management*, *Traffic Engineering*, and *Policy Management* subsystems in addition to *Monitoring*.

All these subsystems require measurement information for their functionality. Monitoring large-scale traffic engineered networks requires mechanisms for data collection from a variety of network nodes, aggregation of these heterogeneous data sets, data mining of large data sets and analyzing this data to generate results for providing feedback to other functional subsystems requiring monitoring information. Our monitoring system architecture, its components, and the interactions with the rest of the management system are depicted in Figure 1.

A. *Monitoring system components*

The monitoring system has the following components:

Node Monitor (NodeMon) is responsible for node related measurements and there exists one NodeMon per router. NodeMon is hosted outside of the router on a dedicated machine, as the availability of required measurements is limited in currently available commercial routers. NodeMon is able to perform active measurements between itself and any other NodeMons, at path or hop level, as well as passive monitoring of the router it is attached to. A NodeMon collects measurement results from either meters or probes located at routers through *passive* or *active* monitoring agents. Another task of NodeMon is also to regulate and abstract various types of measured data. A NodeMon performs some short-term evaluation of results in addition to threshold crossing detection and notification.

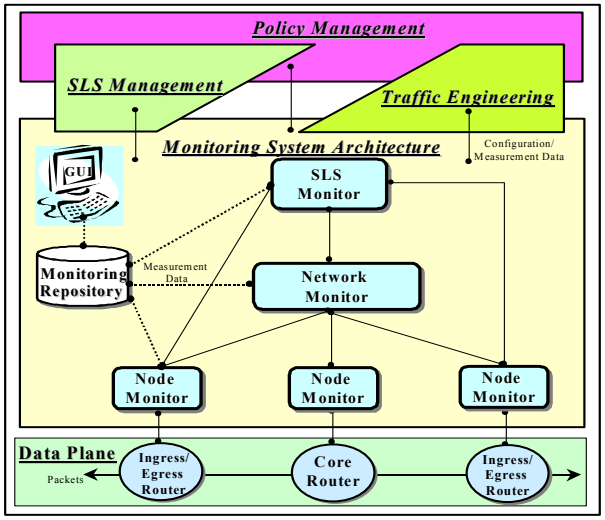

Figure 1. Monitoring system architecture and the interactions with other sub-systems.

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). There is no major scalability concern with NetMon, since the analyzed data are mainly used for non real-time, pro-active control of the network.

SLS Monitor (SLSMon) is responsible for customer related service monitoring, auditing and reporting. SLSMon is centralized, since it must keep track of the compliance of the level of service provided to the customer SLSs of a domain. It utilizes information provided by NetMon and/or various NodeMons. SLS Management requests the creation of the necessary monitors whenever a SLS is invoked. SLSMon handles the requests for activation or deactivation of monitoring a particular set of SLSs. During its operation, SLSMon accesses a repository for measurement data collected by NodeMons and NetMon and combines the data for each individual SLS, i.e. path level performance related statistics and SLS specific traffic related statistics. For each SLS, the performance parameters and the traffic-related values are

checked against measurement data to determine whether any violations occurred.

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.

Monitoring GUI (MonGUI) is used for displaying the measurement results and can be used in a Network Operations Center. It presents a user interface allowing human operators to request graphical views of monitoring statistics extracted from the monitoring data store. It also exposes an interface to allow other components to request the display of statistics.

B. Monitoring system implementation

The system has been implemented in a modular fashion using an object-oriented approach. The system is managed through policy-based high level configuration at node level, network level, and monitoring parameter level (such as specifying synthetic traffic injection rate, packets sizes, etc.). The monitoring system defines a set of CORBA (Common Object Request Broker Architecture) interfaces to internal monitoring components for communicating with one another and to external components. All the CORBA interfaces have been implemented using the Java language on a Java2 CORBA platform. Most components have been implemented in Java, apart from the NodeMon's active/passive monitoring agents that interact directly with the router. These agents set-up/retrieve monitoring data directly on the routers, were implemented in C++. The CORBA Notification service is used for delivering monitoring events to clients.

Various parts of the *SLS Management*, *Traffic Engineering* 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 sort of monitoring information, for example SLSMon uses information from NodeMon or NetMon, and NetMon uses information from NodeMon. We collectively refer to all the components and subsystems requiring monitoring information from parts of the monitoring system, as *monitoring clients* (or clients for short).

The monitoring operation is split into four phases:

Configuration: every client that requires monitoring information must register to one/more of the monitoring components (Node, Network, or SLS). The client must request monitoring actions by providing the necessary information (metric to be monitored, sampling and summarization periods, thresholds, etc). Clients have the option of requesting one/more aggregation functions to be applied to the data chosen from a set of available statistical functions. An XML schema has been defined that allows clients to specify their monitoring requirements.

Execution: NodeMons perform the measurements based on the received configuration. Passive measurements may be performed using SNMP; by feedback reports of the emerging Common Object Policy Service (COPS); or by proprietary polling mechanisms e.g. Command Line Interface. Active measurements (delay and loss) can be measured using the One Way Delay Active Protocol [12] defined by the IETF IPPM.

Reporting and exception: NodeMons send back the analyzed data and/or push the threshold crossing events to the interested monitoring clients. Network and SLS monitoring can provide both current and historical longer-term in-depth statistical analysis of monitoring data as requested by clients. System administrator may request the graphical display of any measurement data at node, network, and SLS levels.

VI. ASSESSMENT OF THE PROPOSED MONITORING SYSTEM

Here, we present the assessment of the monitoring system described in the previous section. We assess it in terms of *accuracy* and *scalability*. Accuracy is very important since the network operation relies on monitoring information, which has to be accurate and reliable. In addition, the monitoring system should scale with extending the network topological scope, increasing load, etc.

Assessment was based on experimental results obtained through a testbed shown in Figure 2, consisting of four routers connected through three 2Mbps serial links in a linear fashion.

A Pentium 1.5 MHz PC is attached to each router and hosts the node monitor with the PC attached to edge router PE1 also hosting the network monitor. Two Data Channel Simulators (DCS) are used to introduce delay and loss into links 1 and 3. A commercial traffic generator is connected to both edge routers PE1 and PE2 and is used to generate synthetic traffic in a loopback form. The delay results measured by the traffic generator and the packet losses programmed in DCSs are used to verify the results measured by the monitoring system.

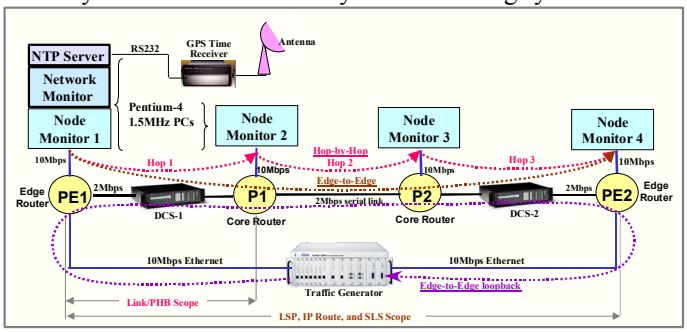

Figure 2. Experimental testbed.

The first test addressed one-way delay and packet loss from PE1 to PE2. The delay values measured by the monitoring system were in most of cases overlapped with the ones measured by the traffic generator. The one-way delays were measured in 0.5 second intervals and the delay values were about 12.6 msec in both cases over relatively long period. This verifies very good accuracy of monitoring results even in configurations like the one used here in which the monitoring agents are located outside the routers. We observed similar

behavior with respect to packet loss. Programming 3.1% packet loss in DCS-2 results in 3.28% packet loss measured by monitoring system.

Subsequent tests combined accuracy with scalability by comparing edge-to-edge vs. hop-by-hop one-way delay and packet loss. In this case, measured edge-to-edge (PE1 to PE2) delays and packet losses were compared to aggregated values produced by the network monitor based on per hop measurements (Hop1: PE1-P1, Hop2: P1-P2 Hop3: P2-PE2).

Figure 3 shows the delay results. The mean difference between edge-to-edge and aggregated hop-by-hop result is 1.1 msec which is mainly due to the fact that more measurements processing are required in the hop-by-hop method. If the active monitoring agents were embedded in the routers, the delay difference would have been considerably reduced.

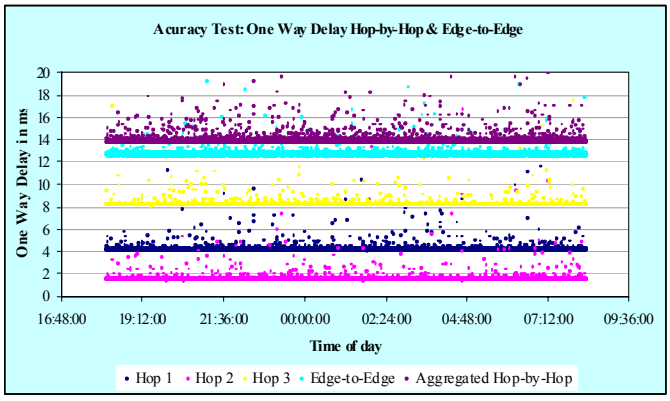

Figure 3. Edge-to-edge and hop-by-hop one way delay results.

Figure 4 shows the packet losses experienced over each hop and edge-to-edge and the aggregated per hop results. The mean packet losses programmed in DCS1 and DCS2 were 2.0% and 3.1% respectively.

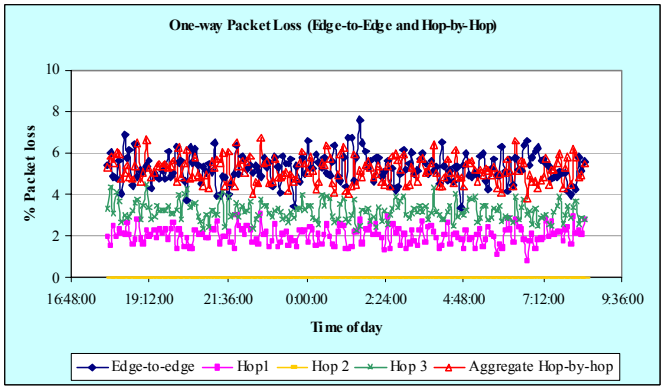

Figure 4. Edge-to-edge and hop-by-hop one-way packet loss.

The average measured results were *5.26%* for edge-to-edge, 2.04% for hop 1, 0.0% for hop 2 (no DCS on link 2), 3.19% for hop 3, and *5.16%* for aggregated hop-by-hop. The difference of 0.1% is negligible and can be attributed to rounding errors. Overall, we can state that comparable results are obtained by both methods, making hop-by-hop method more attractive because enhanced monitoring scalability.

VII. Conclusions

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 off-line proactive and dynamic reactive measures. In this paper, we identified first the measurement requirements for traffic-engineered networks. We subsequently presented requirements for a scalable monitoring system that gathers real-time data to reflect the current state of the network. We then presented principles for designing scalable monitoring systems and methodologies for scalable event monitoring used for network operation and in-service performance verification. We finally presented 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. Based on assessment results, we showed that the proposed monitoring system provides good accuracy for one-way delay and packet loss while it also provides highly comparable edge-to-edge and hop-by-hop results. In summary, we believe that the presented principles result in scalable monitoring systems that can contribute towards operationally optimized traffic-engineered networks.

References

[1] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, J. McManus, "Requirements for Traffic Engineering Over MPLS", IETF Informational RFC-2702, September 1999.

[2] S. Blake, D. Black, et al., "An Architecture for Differentiated Services", Informational RFC-2475, Dec. 1998.

[3] D. Goderis et al, "Service Level Specification Semantics, Parameters, and Negotiation Requirements", IETF Internet Draft: draft-tequila-sls-01.txt, work in progress, Dec. 2001, see: http://www.ist-tequila.org.

[4] V. Paxson, J. Mahdavi, A. Adams,, and M. Mathis "An Architecture for Large-Scale Internet Measurement" IEEE Communications Magazine, vol. 36 no. 8, pp. 48-54, August 1998.

[5] A. Feldman et al., "NetScope: Traffic Engineering for IP Networks", IEEE Network Magazine, Vol. 14, No. 2, pp. 11-19, March/April 2000.

[6] J. L. Alberi, Ta Chen, et al., "Using Real-Time Measurements in Support of Real-Time Network Management", RIPE-NCC 2nd Workshop on Active and Passive Measurements (PAM2001), Amsterdam April 2001.

[7] U. Hofmann, I. Milouchewa, "Distributed Measurement and Monitoring in IP Networks", In Proc. 5th World Multi -Conference on Systemics, Cybernetics and Informatics (SCI2001), Orlando, USA, July 2001.

[8] P. Trimintzios et al., "A Management and Control Architecture for Providing IP Differentiated Services in MPLS-based Networks", IEEE Communications Magazine, Vol. 39, No. 5, May 2001.

[9] D. L. Mills, "Network Time Protocol (Version 3) Specification, Implementation", IETF Draft Standard RFC-1305, March 1992.

[10] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis, "Framework for IP Performance Metrics", IETF Informational RFC-2330, May 1998.

[11] P. Aukia, et al., "RATES: A Server for MPLS Traffic Engineering", IEEE Network Magazine, Vol. 14, No 2, pp. 34-41, March/April 2000.

[12] S. Shalunov, B. Teitelbaum, M. Zekauskas, " A One-way Active Measurement Protocol ", Internet Draft, draft-ietf-ippm-owdp-04.txt, work in progress, July 2002.