Journal of Network and Systems Management, Vol. 11, No. 4, December 2003 (© 2003)

Building Quality-of-Service Monitoring Systems for Traffic Engineering and Service Management

Abolghasem (Hamid) Asgari,[1,3] Panos Trimintzios,[2] Mark Irons,[1] Richard Egan,[1] and George Pavlou[2]

Deployment of quality-of-service (QoS) based value-added services in IP networks necessitates the use of traffic engineering. Traffic engineering allows service providers to use the network resources efficiently, according to the different quality levels associated with the range of services they offer. Traffic engineering relies typically on monitoring data for both "offline proactive" and "dynamic reactive" approaches. Monitoring data may be used for network provisioning, dynamic resource allocation, route management, and in-service performance verification for value-added IP services. A monitoring system should scale with the *network size*, the *network speed*, and the *number of customers* subscribed to use value-added IP services. This paper investigates the requirements of scalable monitoring system architectures, proposes principles for designing such systems and validates these principles through the design and implementation of a scalable monitoring system for traffic engineering and QoS delivery in IP Differentiated Services networks. Methods for assessing the relative merits of such monitoring systems are proposed. Experimental assessment results prove the scalability, accuracy, and also demonstrate the benefits of the proposed monitoring system.

KEY WORDS: IP; QoS monitoring; differentiated services; passive/active measurements.

1. INTRODUCTION

Monitoring systems are becoming increasingly important for providing quantified Quality-of-Service (QoS) based services and service assurance. Until today the Internet has been delivering a single class best-effort IP service without any performance guarantees. The measurement functions in current best-effort networks

[1]Thales Research & Technology (UK) Limited, Worton Drive, Worton Grange, Reading RG2 0SB, United Kingdom. Emails: {*Hamid.Asgari,Mark.Irons,Richard.Egan*}*@thalesgroup.com*
[2]Center for Communication Systems Research (CCSR), University of Surrey, Guildford, Surrey, GU2 7XH, United Kingdom. E-mails: {*p.trimintzios,g.pavlou*}*@eim.surrey.ac.uk*
[3]To whom correspondence should be addressed.

mostly have a diagnostic role. They evaluate the status of the network, analyze the network behavior during a certain time period, and provide feedback reports to a management system. The measurement information is normally collected on both a per-traffic flow basis for accounting and on a per-link basis for diagnostic purposes.

Traffic Engineering (TE) can be defined as the collection of techniques that allow service providers to maximize their network resource utilization, while at the same time meet the customers' demands as they are specified in contractual agreements. Traffic engineering deals mainly with performance optimization of operational IP networks and encompasses the application of principles to the measurement, characterization, modeling and control of traffic [1]. When adding traffic-engineering capabilities to a network, the algorithms used need typically an overview of the network status for their dynamic reactions. The functionality that delivers this status 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], the technical part of a Service Level Agreement (SLA) between the provider and the customer. 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, e.g., real-time, Virtual Private Network—VPN, best-effort services, etc., by employing traffic engineering mechanisms, service monitoring is becoming increasingly 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 assisting the network operation and providing service auditing for both traditional and value-added services. In addition, traffic of each service type or class has certain requirements and exhibits certain behavior (e.g., Voice over IP service requires low delay and low loss, while data services can tolerate medium delay). Having a single measurement result is not adequate when having traffic of different service classes. It should be noted that, in best-effort networks, a single measurement (e.g., round-trip/one-way delay) is performed between a given source-destination pair irrespective of different traffic flows sent between these end-points. Given the multitude of services with different QoS requirements, measurement information needs to be collected at a finer granularity than per source-destination pairing; i.e., the service class needs to be also taken into account.

In summary, a large amount of information is needed for service level monitoring and for traffic engineering of large operational IP networks. It is also equally important to be able to adapt the monitoring processes to the frequent changes in the network's state and the continuously evolving network configuration. All these create a big challenge to monitoring systems and necessitate scalable architectures in order to monitor the network state in real-time. This paper addresses the scalability issues of monitoring systems and proposes novel principles for designing and assessing such systems. Furthermore, it describes the architecture of such a system, that adheres to the scalability requirements and takes into account in real-time both service-level monitoring; and monitoring for traffic engineering purposes in DiffServ-capable networks.

This paper is organized as follows. Section 2 describes measurement requirements that need to be taken into account when designing a monitoring system for use in traffic-engineered IP networks. Section 3 presents the common measurement methods. Section 4 explains some novel scalability principles in designing monitoring system architectures. It looks at how measurements can be organized in a proper way for traffic engineering and service monitoring purposes. Section 5 describes the architecture of a monitoring system where we applied the principles described in Section 4. Section 6 includes the fundamental properties, namely the accuracy of measurements, benefits/costs, and scalability that we need to test in order to assess the performance of a monitoring system. Section 6 also provides performance results related to the monitoring system. An overview of related work is given in Section 7, and finally, a summary is presented in Section 8.

2. MONITORING REQUIREMENTS FOR TRAFFIC-ENGINEERED NETWORKS AND CUSTOMER SERVICES

Traffic engineering is achieved through capacity and traffic management [4]. 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 and 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 automatically reacting and taking actions in response to the current state of the network and adaptively optimizing network performance, and/or pro-actively by using forecasting techniques to anticipate future trends and applying action to prevent any undesirable future conditions. Ideally, control actions should involve the modification of traffic management parameters, parameters associated with routing, and constraints associated with resources [1].

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

1. Assist traffic engineering in taking provisioning decisions for optimizing the usage of network resources according to short to medium term changes. This is to assist the efficient and effective allocation of resources i.e., to

queues and paths over which routes will be established. The ability to obtain statistics at the QoS-enabled route level is important and should be considered an essential requirement for traffic engineering. This information can be used for taking appropriate engineering actions on setting up new routes, modifying existing routes, performing load balancing among routes, and re-routing traffic for optimization purposes or work around congestion. It is also used to perform node-level optimization to resource reservations (bandwidth assignment and buffer management) to combat localized congestion.

2. Assist traffic engineering in providing analyzed traffic and performance information for long-term planning in order to optimize the network and to avoid undesirable network conditions. The analyzed information includes traffic growth patterns and congestion issues. This is extremely helpful for proactive network control.
3. Verify whether the QoS performance guarantees (negotiated between a customer and a provider) 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 [3]. In-service verification of traffic and performance characteristics per service type is required for monitoring the continuity and quality-of-services offered to customers, auditing the services, and preparing reports.

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 and measurement systems must be generic enough to cope with such changes at the minimal cost.

3. MEASUREMENT DATA AND MEASUREMENT 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 characteristics. In traffic-engineered networks, monitoring occurs at the network layer for deriving all these characteristics (except packet and application level characteristics). These include performance-related measurements such as one-way delay, packet delay variation, one-way packet loss, and traffic-related measurements such as traffic load, and throughput.

There are typically two types of methods to perform low-level measurements in a monitoring system. *Active measurements* inject synthetic traffic into networks based on a 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, both end-point clocks need to be synchronized. Therefore, methods like the Network Time Protocol (NTP) [5], Global Positioning System (GPS) or other Code Division Multiple Access (CDMA) based time sources should be used. GPS provides high precision but its deployment makes it a relatively expensive solution.

Passive measurements are used to simply observe data traffic transmitted through the network. While passive measurements do not require co-operation of end-points, they require continuous collection of data and monitoring the link at full load, which can be problematic for high-speed links. In both cases, the quality of analyzed information depends on the granularity and integrity of collected data.

4. PRINCIPLES FOR DESIGNING SCALABLE MONITORING SYSTEMS

Scalability is the ability of a system to be deployed and used in a large scale. Scalability in QoS-enabled IP networks has three aspects: size of network topology, number and granularity of the supported service classes, 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 require 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 *large* IP networks offering a number of services to a *large* number of customers. The monitoring system must have a number of design features for performing a wide range of monitoring tasks that ensure a scalable solution to deliver the expected performance at a large scale. The monitoring tasks include data collection, data aggregation, data analysis, and providing feedback results. A diverse variety of measurement data is needed in order to perform both network and customer service performance monitoring. In QoS-enabled networks, the amount of measurement data increases with the number of class states (e.g., different queues) per interface and the large number of edge-to-edge routes per class that must be monitored. Hence, scalable monitoring system architectures must adhere to the principles described later and summarized in Table I.

In Table I, we summarize the six principles and the corresponding optimal scope of information and required action that we have identified for a monitoring system to be scalable. The first principle is to define the monitoring process granularity at the aggregate level of PHB and LSP/IP route level (and not at packet level) for data gathering since collecting packet level micro-flow statistics is expensive and nonscaleable. The second principle is to distribute the data collection

Table I. Principles for Building Scalable Monitoring System Architectures

Principle	Optimal Scope and Required Action
Defining the monitoring process granularity	At Per Hop Behavior and path (LSP/IP route) level
Distributing the data collection system	At node level for processing and aggregating data at source
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 the path level and traffic measurements at the SLS level
Reducing the amount of synthetic traffic	By using hop-by-hop measurements and calculating edge-to-edge results
Controlling the amount of synthetic traffic	By having a trade-off between the synthetic traffic load and sampling frequency

system at node level for processing and aggregating data at the source. The third principle is to minimize the measurement transmission overhead by employing event notification and summarization of statistics. In the fourth principle, we identify the need for carrying out *performance* measurements at the LSP/IP route and *traffic* measurements at the SLS levels. This principle aims to reduce the amount of synthetic traffic injection for carrying out the SLS monitoring as several SLSs may use a single IP route/LSP. The fifth principle is about reducing the amount of synthetic traffic by using per-hop measurements and calculating edge-to-edge results in case of MPLS-TE where there might be a huge number of LSPs to be monitored. We classify active monitoring as either edge-to-edge, per-hop, or hop-by-hop. The edge-to-edge approach considers an entire path through the domain and involves injecting synthetic traffic at a domain ingress node and receiving it at a domain egress node. The per-hop approach considers a single hop within the domain, and involves injecting traffic at the start of the hop and receiving it at the end of the hop. The hop-by-hop approach is an extension of the per-hop approach in which the results of several per-hop measurements are aggregated to derive an edge-to-edge result. The motivation behind the hop-by-hop approach is to reduce the quantity of synthetic traffic (when compared to the edge-to-edge approach), at the expense of increased processing effort. The reduction in synthetic traffic comes about because each per-hop measurement is potentially a constituent of several edge-to-edge paths. The hop-by-hop approach is only viable if it has an accuracy that is comparable to that achieved using the edge-to-edge approach. Finally, the last principle states that we need to control the amount of synthetic traffic by having a trade-off between the synthetic traffic load and sampling rate. That is, smaller time intervals mean injecting more synthetic traffic into the network, but injecting more synthetic traffic mean introducing higher load in the network that affect the network performance.

5. THE ARCHITECTURE OF A SCALABLE MONITORING SYSTEM

Recently, there have been attempts to design and build network management and control systems that support traffic engineering and service differentiation (see [10, 11] for two examples). In this work, we describe an intra-domain monitoring system for traffic-engineered DiffServ networks that was designed by incorporating and adhering to the principles described in Section 4. Our monitoring system is tightly coupled with the control and management system presented in [10] that includes the following functionality, and corresponding subsystems: *SLS Management*, *Traffic Engineering,* and *Policy Management,* in addition to *Monitoring. SLS Management* is responsible for subscribing and negotiating SLSs with customers, a customer being possibly a service provider. *Traffic Engineering* is responsible for provisioning in long to medium timescales according to the projected demands, and in the medium to short term timescales for establishing and dynamically maintaining the network configuration that meets the SLS demands at a minimum cost. *Policy Management* adds flexibility to the operation of the other subsystems by driving their behavior according to the administrators' high-level policies.

All these subsystems require one way or another some monitoring information for their functionality. Monitoring large-scale traffic engineered networks entails mechanisms for data collection from a variety of network nodes, aggregation of these heterogeneous data sets, data mining of large data sets and analyzing the data sets to generate results for providing feedback to the other functional subsystems requiring monitoring information. We have designed a *Monitoring* system to meet all these requirements. The detailed Monitoring system architecture, its components, and the interactions with the rest of the control and management system are shown in Fig. 1.

5.1. Monitoring System Components

The *Monitoring* system has the following main components:

1. **Node Monitor** (NodeMon) is responsible for node related measurements and there exists one NodeMon per router. In our implementation, NodeMon is hosted outside of the router on a dedicated machine, as the availability of required measurements is limited by what is currently supported by the 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 on the router to which it is attached. NodeMons are configured with information about the variable to be monitored, the sampling and summarization periods, and, if required, threshold parameters. A NodeMon collects measurement results from either meters or probes located at routers through passive monitoring agents or active monitoring agents. Probes present the data they collect in a variety of ways. Another

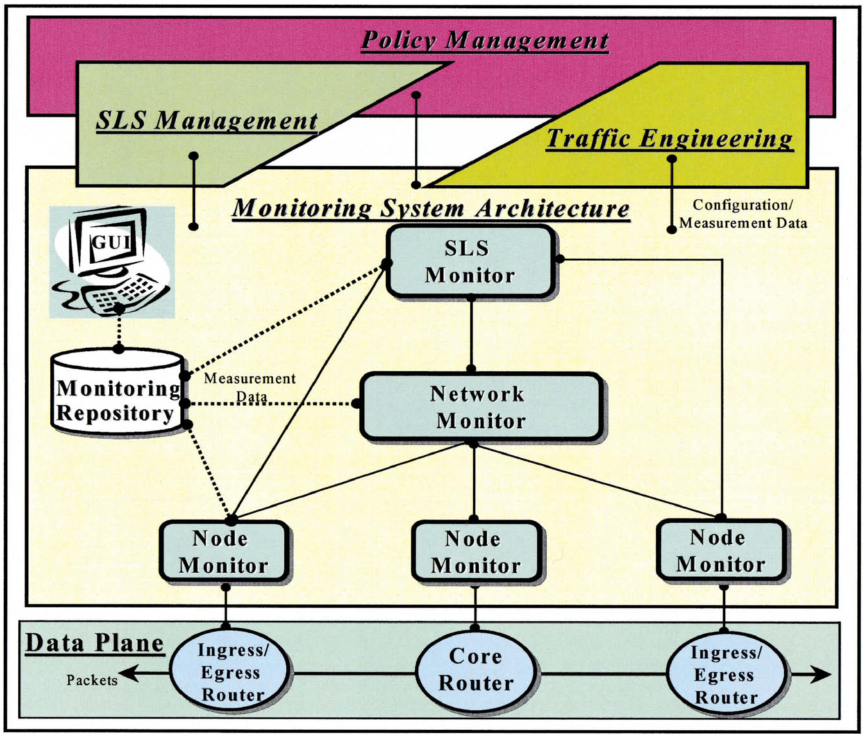

Fig. 1. The proposed Monitoring system architecture.

task of NodeMon is to regulate and re-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 by definition centralized and it 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 nonreal-time, pro-active control of the network. NetMon also uses hop-by-hop measurements for calculating edge-to-edge results. Since relatively frequent topological changes are common, the monitoring system must be flexible enough to adapt to such changes; modifications to monitoring processes are thus required to accommodate rapid network changes. This is achieved through NetMon access to a Topology

Repository (TopologyRep) that contains updated information about the network physical topology and the current network configuration as resulted from the provisioning processes [10].

3. **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 SLS instances in a domain. It utilizes information provided by the centralized NetMon and/or the various distributed NodeMons.

   The *SLS Management* subsystem is the main client of SLSMon. It 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 monitoring 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 contracted SLS, the performance parameters and the traffic-related values stored in SLS Repository (SLSRep) are checked against measurement data to determine whether any violations occur and then reports are generated.
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 for displaying 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. It also exposes an interface to allow other components to request display of statistics.

5.2. Monitoring System Implementation

This system has been implemented in a modular fashion using an object-oriented approach. Figure 2 shows the implemented *Monitoring* system, components and the communications between components. The *Monitoring* 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 communicating with one another and to external components. All the

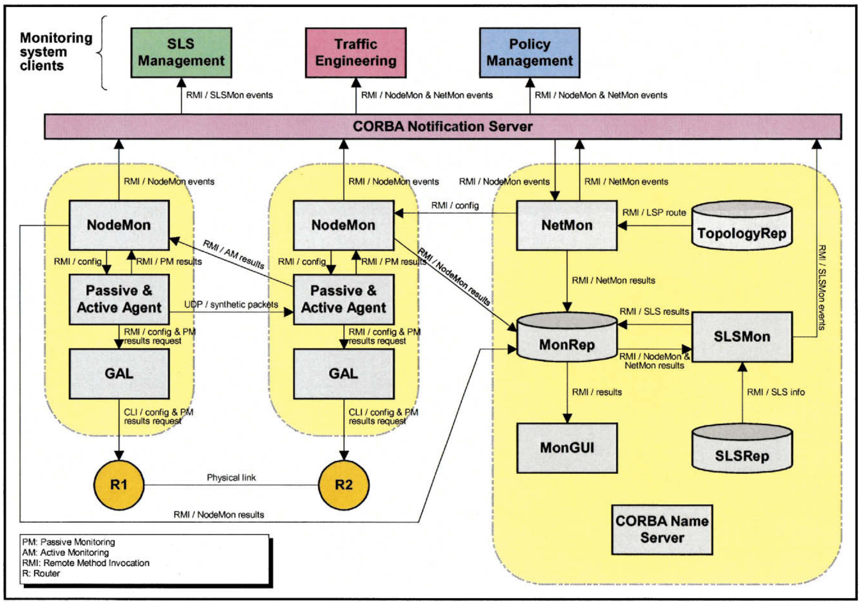

Fig. 2. The Monitoring system components and their interaction with internal and external components.

CORBA interfaces have been implemented using the Java language on a Java2 CORBA platform. All components have been implemented in Java, except the NodeMon sub-component that interacts directly with the router, i.e., the active and passive monitoring agents that set-up and retrieve monitoring data directly on the router; which have been implemented in C++ to enhance performance and response time. The monitoring agent is a per-node component. The passive monitoring agent is responsible for requesting monitoring data from the router. The active monitoring agent is responsible of generating, transmitting, and receiving synthetic packets. The receiving agent calculates one-way packet delay and detects occurrences of one-way packet loss.

The CORBA Notification Server is used by the *Monitoring* system to decouple monitoring clients from the sources of monitoring events. By event, we mean either threshold crossing or statistical data. The NodeMon, NetMon, and SLSMon push events into Notification Service Event Channels, that take responsibility for delivering events to the registered monitoring clients. Monitoring components access a database to retrieve configuration information, network topology information, and to store monitoring results. The CORBA Name Server is also used in order for components to be able to locate other components in the network. It should be noted that the GAL (Generic Adaptation Layer) is a per-node component used to provide a generic software layer that isolates monitoring and management components from the type of network element (i.e., commercial or Linux-based routers) actually used.

5.2.1. Phases of Monitoring *System Operation*

Various parts of the *SLS Management*, *Traffic Engineering*, and *Policy Management* subsystems that require monitoring information must request it 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 some monitoring information from parts of *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 configure the monitoring components (Node, Network, SLS), to perform the required actions. Clients request monitoring actions by supplying configuration information, including the metric to be monitored, sampling and summarization periods, and threshold values. Clients can request as many monitoring actions as they require. Clients have the option of requesting one or more aggregation functions applied to the monitoring data, from a set of available statistical functions. Clients specify their requirements to the *Monitoring system* using a defined XML schema. A more detailed example of how a client would request one-way-delay monitoring of an LSP is described later.

Initially the client gets a reference to the `MonitoringFactory` from the CORBA naming service. The `MonitoringFactory` is registered with the well-known name *MonitoringFactory*. Next the client uses the `MonitoringFactory` to get a reference to the `NodeMonitorSourceFactory` by invoking `MonitoringFactory.getNodeMonitorSourceFactory()`. The client then gets a reference to the `NodeMonitorSource` for the required node by invoking `NodeMonitorSourceFactory.getGetInstance(NodeId)`. Where the `NodeId` parameter is equivalent to the router ID, and is defined in the network repository. A new monitor is added to the `NodeMonitorSource` by invoking `NodeMonitorSource.addMonitor(MonitorSpecXML)`. The `MonitorSpecXML` parameter is an XML encoded string that must conform to the monitor specification XML schema. An example monitor specification is given here:

```
<OneWayDelayNodeMonSpec>
  <LSP>
    <InterfaceIPAddr>195.166.11.202</InterfaceIPAddr>
    <LSPId>2</LSPId>
  </LSP>
</OneWayDelayNetworkMonSpec>
```

After the monitor has been added, the client can proceed to add monitor jobs, and cause the monitor to generate events. A monitor job is added to the monitor by invoking `Monitor.addMonitorJob(MonitorJobSpecXML)`. The `Monitor-JobXML` parameter is an XML encoded string that must conform to the monitor job specification XML schema. An example monitor job specification if shown here:

```
<ThresholdMonJobSpec>
  <NormalLevel>5.0</NormalLevel>
  <UpperLevel>30.0</UpperLevel>
</ThresholdMonJobSpec>
```

If a client wants to receive the events generated by a monitor job, it must connect to the CORBA Notification Service event channel associated with the monitor of interest. The client gets a reference to the relevant event channel by invoking `NodeMonitorSource.getEventChannel()`.

Finally, the client must initialize the monitor by invoking:
`Monitor.init(MonitorInitParamatersXML)`.

Execution: Node Monitors perform the measurements based on the received configuration. Passive measurements can be performed by using either the Simple Network Management Protocol (SNMP) [12], or the emerging Common Open Policy service (COPS) [13] feedback reports, or by proprietary polling mechanisms such as CLI (Command Line Interfaces). Active measurements (one-way delay and loss) can be configured and measured using One Way Delay Protocol (OWDP) [14] defined by the IPPM working group. In our implementation, we hide the variability of the underlying network node capabilities by the Generic Adaptation Layer (GAL – see Fig. 2).

Data Storing: The NodeMon, NetMon, and SLSMon components process monitoring data and store the processed results in the MonRep for later analysis. NodeMon, NetMon, and SLSMon perform basic processing of raw data in order to reduce the quantity of data that must be persistently stored in the MonRep. This minimizes network traffic in line with the third principle of Table I. The MonRep schema is shown in Fig. 3. The MonRep is composed of four tables: MON_SOURCE, MON, MON_JOB, and MON_EVENT. The MON_SOURCE table contains one record for each NodeMon deployed in the network and an additional record for the centralized NetMon. Monitoring clients request monitoring from a particular NodeMon or the NetMon, and for each metric that is monitored a record is added to the MON table. Records in the MON table contain a reference to their parent NodeMon or NetMon record in the MON_SOURCE table, and an XML specification that defines the object and the metric. Objects that can be monitored are IP routes, LSPs and PHBs. Clients attach Monitor Jobs to monitors that define when events will be generated (e.g., threshold crossing, Moving Average statistics, etc.). Every Monitor Job has an entry in the MON_JOB table. Each record in the MON_EVENT table contains a reference to its parent record in the MON_JOB table that identifies the Monitor Job that created the event.

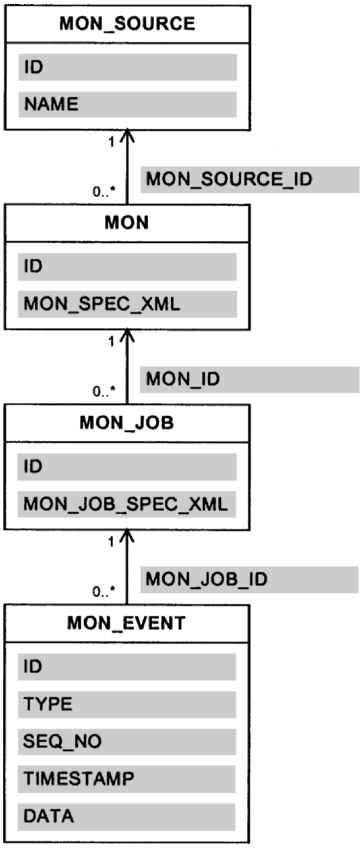

Fig. 3. Monitoring Repository Schema.

Reporting and Exception: NodeMons send back the analyzed measured data and 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, or the system administrator. The administrator may request the graphical display of any measurement data at the node, network, and SLS levels.

Overall, the *Monitoring* system is able to provide the measurements required for monitoring SLSs and other traffic engineering functions. The passive metrics measured by the *Monitoring* system are as follows:

- LSP offered load at the ingress and LSP throughput at egress points in bits per second
- PHB throughput in bits per second

- PHB packet discards in packets per second
- Offered load at ingress and throughput at egress points per SLS or flow.

The active metrics that are measured by the Monitoring system are the PHB, LSP, and IP route one-way packet delay and packet loss. The Monitoring system performs IP route monitoring using either the edge-to-edge technique, or the per-hop technique. PHB monitoring is performed using the per-hop technique. LSP monitoring is performed using either the edge-to-edge technique, or the hop-by-hop technique. Monitoring LSPs using the hop-by-hop technique involves performing per-hop PHB monitoring along the path of the LSP and aggregating the results. This is only possible for LSPs where the path is explicitly known.

6. ASSESSMENT OF THE MONITORING SYSTEM

When building a scalable monitoring system one very important requirement is to have a systematic methodology for assessing its value, performance and relative merits. In this section, we provide the guidelines and criteria for evaluating the measurement accuracy, benefit/cost, and scalability, of a monitoring system, and we subsequently use them in order to assess the proposed *Monitoring* system. We identify four major aspects of a monitoring system that need to be assessed.

Accuracy: Accuracy of the measurement techniques is important since the network operation and dynamic reaction of traffic engineering functions rely on monitoring information. Traffic and performance-related metrics must be measured reliably with great accuracy.

Benefit/Cost: Benefit/cost assessment must show what improvement to the dynamic operation of network is attributable to the monitoring system, along with a measure of the cost incurred in providing that benefit. The primary benefit of the monitoring system is to detect both congestion and under-utilization in the network. By providing real-time QoS-related measurements, traffic-engineered networks must be able to optimize the use of network resources and must offer superior QoS to conventional best effort networks.

Scalability: The monitoring system should scale with extending the network *topological scope*, increasing *load*, increasing *sampling rates* of measurements, etc. To quantify the scalability of the monitoring system, the following assessment tests are required:

- *The time taken by each monitoring system's component to perform its operation.* Each component must operate within a reasonable timescale in preparing the measurement results and in providing the results to the clients. The first comprises the measurement read-out periods, the time to calculate and process the measurement results, and the second includes the delay to transfer the results to the monitoring clients. As the network topological scope grows, the monitoring system's components must still be capable of informing clients in near real-time for tackling performance degradation and for managing congestion in the network. Monitoring components response times should not increase disproportionately with the extension of network topological scope.

- *The additional load on the network introduced by the synthetic traffic.* For a defined sampling rate, it is expected that the volume of synthetic traffic injected to the network for carrying out performance measurements will grow linearly when the network is subject to extending its topological scope but it must have minimal effect on network bandwidth.
- *The ability of SLS Monitor to cope with large number of SLSs.* SLS Monitor must provide the QoS level of individual customer SLSs in a timely fashion. The processing time of SLS Monitor should not grow disproportionately to the number of SLSs being monitored.

6.1. Experimental Set-up

The assessment of the proposed *Monitoring* system is based on the results obtained from an experimental testbed network, shown in Fig. 4, consisting of four commercial routers connected with three 2Mbps serial links in a serialized fashion; i.e., edge router ER1 connected to core router R1 via link 1; core router R1 connected to core router R2 via link 2; and core router R2 connected to edge router ER2 via link 3. A 1.5 MHz PC is attached to each router and hosts the Node Monitor, which ideally should have been part of the router. The PC attached

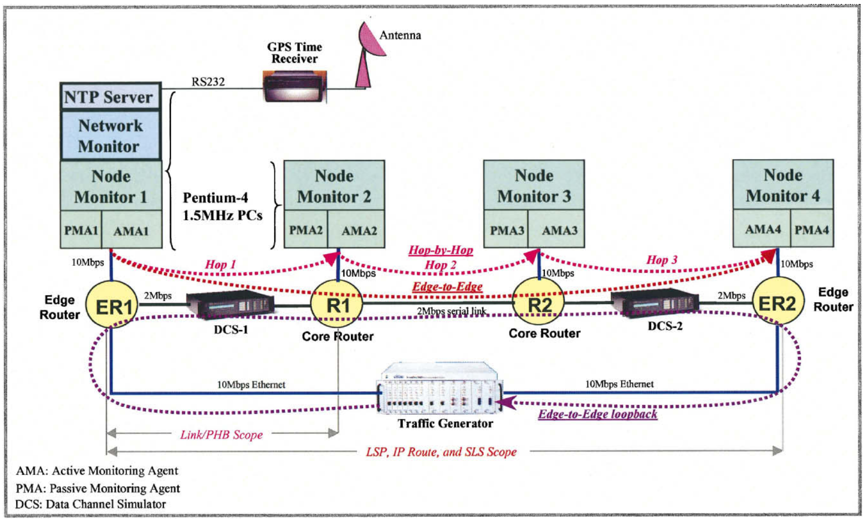

Fig. 4. Testbed configuration.

to edge router ER1 hosts the Network Monitor; note that NetMon should not be necessarily hosted there but it should be located in the Network Operation Center. This PC also acts as the NTP server. 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 ER1 and ER2 and is used to generate synthetic traffic in a loop-back form. The delay results measured by the traffic generator and the packet loss results programmed by the DCSs are used to verify the results measured by the *Monitoring* system.

In order to have an accurate stable reference clock source, the NTP server is time-synchronized via a GPS time receiver and NTP is used to synchronize the Node Monitors. We should mention here the problems we encountered in time-synchronizing the routers for getting accurate one-way-delay results. PC clocks synchronized via NTP were inaccurate even for a lightly loaded network, yielding jittery and periodic 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. In a real world environment a practical solution for the NTP traffic is to have a dedicated NTP server synchronized with a GPS time receiver and a dedicated link (either a NTP Ethernet segment, or a high priority queue in multi-class scenario) at every location that hosts a number of routers.

The assessment tests were conducted using the test scenarios specified in Table II. This table shows mean rate of synthetic traffic in Packet Per Second (PPS) injected by active monitoring agents, data summarization period[4], and the link delay in milliseconds (ms) or packet loss programmed in DCSs for each test scenario. The size of synthetic IP packets was set to 128 bytes in all scenarios. There was a small amount of background traffic in the network and a single PHB along the paths are used in scenarios 1 to 4.

6.2. Accuracy Tests

With this set of tests we are trying to assess the accuracy of the various measurement techniques employed by the proposed *Monitoring* system. These techniques include the active delay and packet loss measurements on a link and the edge-to-edge vs. concatenated hop-by-hop delay and loss measurements. The latter are of great importance since they are the basis for enhancing the scalability of *Monitoring* system, and help to considerably reduce the test traffic load.

6.2.1. One-way Delay Accuracy (Test Scenario 1)

The first test addresses one-way delay from ER1 to ER2 (edge-to-edge) using test scenario 1 specified in Table II. The AMA1 (Active monitoring Agent)

[4]NodeMon averages individual raw data measured by AMA during the summarization period in order to comply with principle set in Section 4.3.

Table II. Experimental Scenarios

Test scenario	Synthetic traffic mean injection rate	Summarization period	Programmed delay in DCS-1	Programmed delay in DCS-2	% Mean packet loss Programmed in DCS-1	% Mean packet loss Programmed in DCS-2
1	2 PPS	—	3 ms	7 ms	0.0	0.0
2	4 PPS	5 minutes	0.0	0.0	0.0	3.1
3	4 PPS	5 seconds	3 ms	7 ms	0.0	0.0
4	4 PPS	5 minutes	0.0	0.0	2.0	3.1
5	4 PPS	2 seconds	0.0	0.0	0.0	0.0

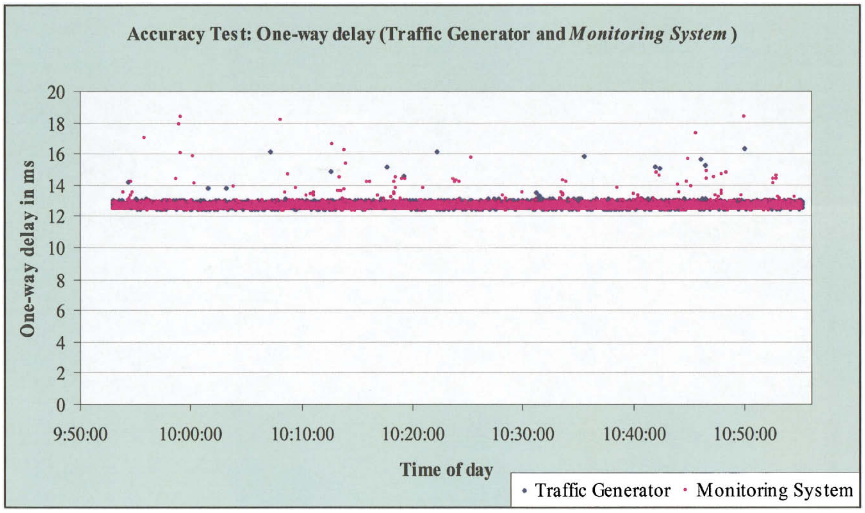

Fig. 5. One-way delay accuracy result.

timestamps and transmits the synthetic packets. AMA4 timestamps the received synthetic packets. At the same time, the traffic generator is used in a loopback form to generate synthetic traffic which is directed to ER1 and to receive the synthetic traffic from ER2. The traffic generator is configured to inject synthetic traffic with a similar profile to traffic injected by AMA1. The traffic generator timestamps the packets it transmits and receives using its internal clock. Figure 5 shows the raw measurement results for one-way delay as reported by proposed *Monitoring* system and by the traffic generator. The delay values measured by the *Monitoring* system are very close to the ones measured by the traffic generator. This verifies very good accuracy of monitoring results even in configurations like the one used here where the monitoring agents are located outside the routers.

We observed from the obtained results that the one-way delay measurement results are very accurate even in very small timescales. Therefore we can conclude that one-way delay resulted from the active measurements of our system can be used to assist both short and long term dynamic traffic engineering operations of the network as well as for SLS Monitoring.

6.2.2. One-Way Packet Loss Accuracy (Test Scenario 2)

The second test was conducted to measure one-way packet loss from ER1 to ER2 (edge-to-edge) using test scenario 2 specified in Table II. A summarization period for calculating the sampled packet loss ratio was defined to be 5 min. Figure 6 shows the result of one-way packet loss as measured by the *Monitoring* system, and as it was programmed in the DCS-2. The oscillation of measured values in different

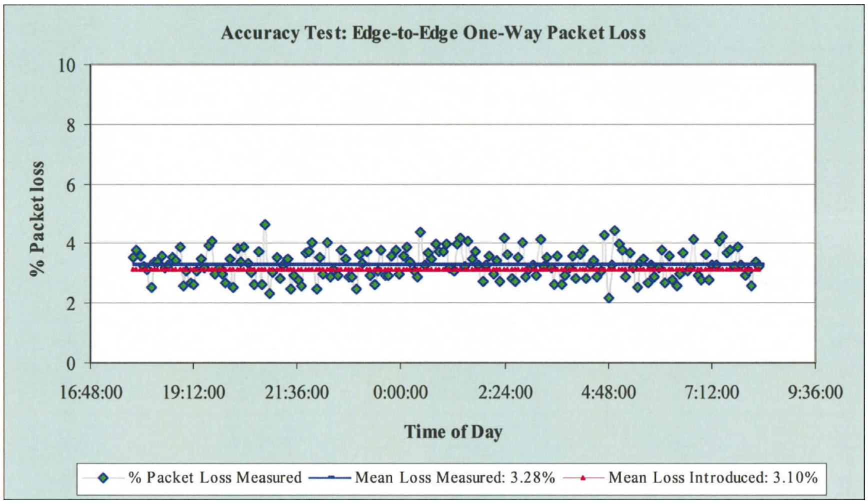

Fig. 6. One-way packet loss accuracy result.

intervals is due to the fact that synthetic packets are randomly generated and losses are introduced randomly resulting in different packet loss values in different intervals. Figure 6 shows that when we programmed 3.1% packet loss in DCS-2 at link 3, the packet loss measured by the *Monitoring* system was 3.28% on average.

The result shows oscillations of the measured packet loss even though the measurements were averaged over relatively large timescales. Based on these observations we conclude that one-way packet loss resulted from active measurements is suitable to be used only for longer-term route monitoring and SLS monitoring. If it is required to use the packet loss behavior for making short-term dynamic traffic engineering decisions, or for dynamically provisioning the various physical queues and schedulers (PHBs), we recommend to use passive monitoring of packet discards instead of relying on the sampled packet loss measurements.

6.2.3. Edge-to-Edge vs. Hop-by-Hop One-way Delay and Packet Loss Accuracy

We tested the accuracy of inferring the edge-to-edge delay in the network from aggregating per hop measurements. These tests are also important in terms of scalability and reduced measurement traffic overhead in the network. If the accuracy of the aggregated one-way delay and packet loss over the forwarding path is relatively accurate, then active per hop monitoring for each PHB, that scales linearly with the network size is beneficial compared with edge-to-edge active monitoring for every possible path. In the following, we summarize the main findings, while more details of these results are reported [15].

The configuration was similar to the one in Fig. 4. The mean difference between edge-to-edge and aggregated hop-by-hop one-way delay results (Test scenario 3) was in the order of 1 millisecond, which was mainly due to the fact that synthetic traffic had to cross the Ethernet segments from the supporting PCs to the routers and *vice versa*, and more measurement processing was required by the hop-by-hop approach. If the active monitoring agents were embedded in the routers, as should have been if the routers were capable of performing one-way active monitoring, the delay difference would have been considerably reduced. This eliminates the Ethernet packet transmission times. The average measured one-way packet loss (Test scenario 4) experienced edge-to-edge, and aggregated hop-by-hop was *5.2%* for edge-to-edge, and *5.1%* for aggregated hop-by-hop. The difference of 0.1% is considered negligible and can be attributed to rounding errors. Overall, we can conclude that hop-by-hop method gave comparable results with the edge-to-edge one, making the proposed hop-by-hop method more attractive because of enhanced monitoring scalability and reduced monitoring overhead (synthetic traffic loads).

6.3. Benefit/Cost Assessment

The benefit of a monitoring system to the operation of traffic-engineered networks is to provide measurement information at the required granularity in order to aid the various management and control functions to have sufficiently accurate and in time information to be able to make decisions and take actions. The latter will target for the improvement to the network resource utilization.

One cost associated with employing our *Monitoring* system is the introduction of synthetic traffic to the network and the communication overhead to transfer measurements information to the related management entities. The other cost is the need for deployment of reliable and accurate clock synchronization technology for PCs hosting Node Monitors and their associated active monitoring agents for performing one-way delay measurements. In the edge-to-edge method, only edge the NodeMons need to be synchronized, whereas in the hop-by-hop method, all pairs of NodeMons that perform per hop measurements need to be synchronized.

This experiment was conducted in order to measure one-way delay and throughput of a number of LSPs with different characteristics carrying traffic from different classes. In the experiment, we have configured three LSP tunnels from ER1 to ER2 for carrying traffic belonging to three DiffServ classes, i.e., Expedited Forwarding (EF), Assured Forwarding (AF1), and Best effort (BE). We used the Low Latency Queuing (LLQ) on the output interfaces of the routers that tunnel paths transit in order to provide strict priority queuing to the Class-Based Weighted Fair Queuing (CBWFQ) scheduling discipline supported by the commercial routers. With strict priority queuing, preferential treatment is given to the traffic that belongs to EF class over other classes. CBWFQ extends Weighted Fair Queuing (WFQ) functionality in order to support the configuration of user-defined traffic classes. A physical queue is reserved for each class, and the traffic belonging to a class is directed only to the physical queue for that class. Each queue was configured by setting the bandwidth to a certain amount and maximum queue size of 64 packets. The queues for the EF, AF1 and BE classes were assigned 600, 600, and 700 Kbps of bandwidth respectively. EF traffic was directed to the priority queue while AF1 and BE traffic used normal CBWFQ.

Three user traffic generation sources were used to generate EF and AF1 marked traffic at 600 Kbps rates and BE traffic at 800 Kbps (i.e., BE traffic is subjected to loss as more BE traffic is injected to the network than bandwidth reserved for it in order to bring the link to congestion). The user traffic pattern did not change during the course of experiment as the tests only aimed at showing *Monitoring* System is capable of measuring different metrics and provide feedback information. The packets from all three traffic generation sources were set to be 128 bytes long including headers. The user traffic was not shaped at the ingress point. The size of synthetic traffic packets was configured to be of similar size to the user traffic as created by the traffic generators. This was done in order to ensure that the CBWFQ would give the same treatment to both the synthetic traffic and user traffic. Therefore, the delay measured for synthetic traffic was of similar order to the delay experienced by user traffic.

Based on Test scenario 5 specified in Table II, three active monitor jobs were set-up to monitor one-way delay on the LSP tunnels. As shown in Fig. 7, on average EF, AF1, and BE traffic experienced 6.0, 42.7, and 144.9 ms of delay respectively, as was expected from the setting of configuration parameters. This experiment showed that the *Monitoring* system is capable of monitoring the configured classes

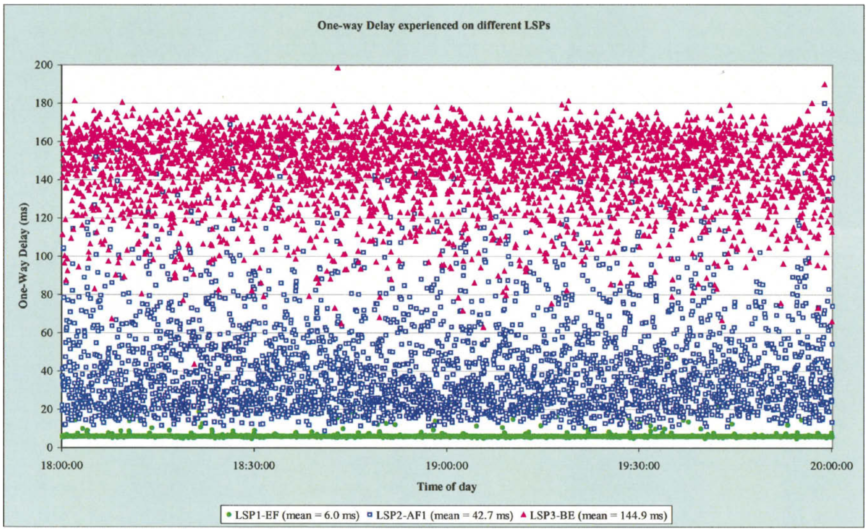

Fig. 7. One-way delay measured on three different LSPs.

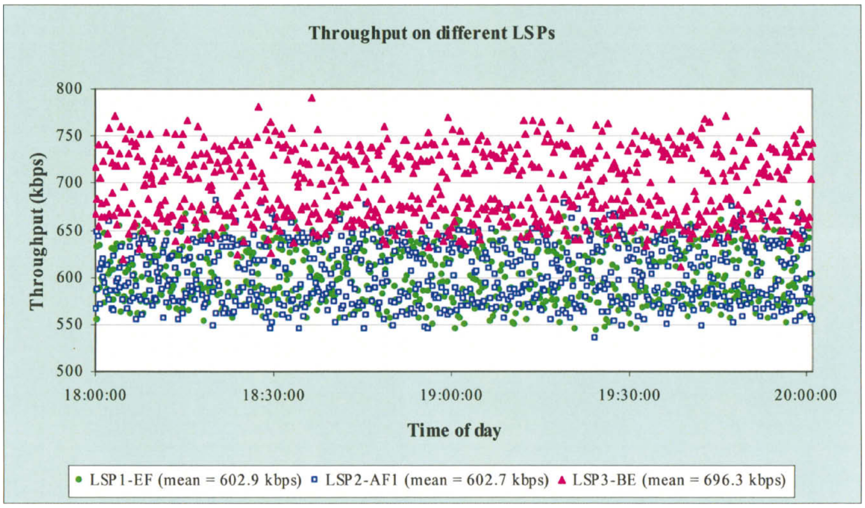

Fig. 8. Throughput measured on three different LSPs.

of services at the LSP level (principle defined in section 4.1) and providing LSP-related measurements. The information retrieved by the *Monitoring* system can be used by a traffic engineering system, to dynamically manage and control the various classes and LSPs by tuning their respective parameters, as described [10].

Three passive monitor jobs were set-up to monitor throughput on these LSPs. The results are shown in Fig. 8. It can be seen that the average measured throughput for the EF, AF1, and BE LSP tunnels is 602.9, 602.7, and 696.3 Kbps respectively, as it was expected from the configuration parameters set in traffic generator sources and queues. It should be noted that the summarization period was 2 seconds for one-way delay results (i.e., one-way delay measured values are averaged in 2-second intervals representing a data point in Fig. 7) and the read-out period for retrieving the throughput values from router was 10 seconds (representing a data point in Fig. 8). These timescales are quite small and can definitely aid short-term traffic engineering processes, therefore showing the benefits of using the *Monitoring* system.

6.4. Scalability Considerations

The *Monitoring* system has been deployed as part of a control and management system [10]. It has been operational in two large testbed and reference networks in the industry providing monitoring information to the functional entities of that control and management system. This *Monitoring* system is designed in a distributed fashion; hence it is able to cope with extending the network topological scope. Active and passive measurements and data aggregation are performed at the NodeMon level making the system independent of the network physical size. The CORBA Notification Service is a centralized service hosted on a single node/PC. For very large networks, the performance of the PC 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 a notification server per domain of nodes or even per node. The typical number of monitors running on a single edge NodeMon PC will be of the order of tens as there are a finite number of LSPs originating from an edge node and there are a limited number of PHBs configured for each node. Experiments that carried out indicate that this number of monitors would not place an unmanageable computational load and would not increase the response time. Therefore, the performance of individual NodeMons would not be a limiting factor in the scalability of the *Monitoring* system.

A scalability assessment was conducted in terms of load (the packet rate) introduced by hop-by-hop method compared with edge-to-edge method. The network link interface used supported a maximum number of eight PHBs. A considerable number of LSPs can traverse the link. The injection rate was set to four packets per-second in both the edge-to-edge and the hop-hop methods. Figure 9 shows that the number of synthetic packets crossing the link is equal in both methods if the number of LSPs and PHBs are less than or equal to eight. Increasing the number of LSPs traversing the link increases the number of synthetic packets in edge-to-edge method while the number of synthetic packets depends on the limited number of PHBs in of hop-by-hop method. This proves

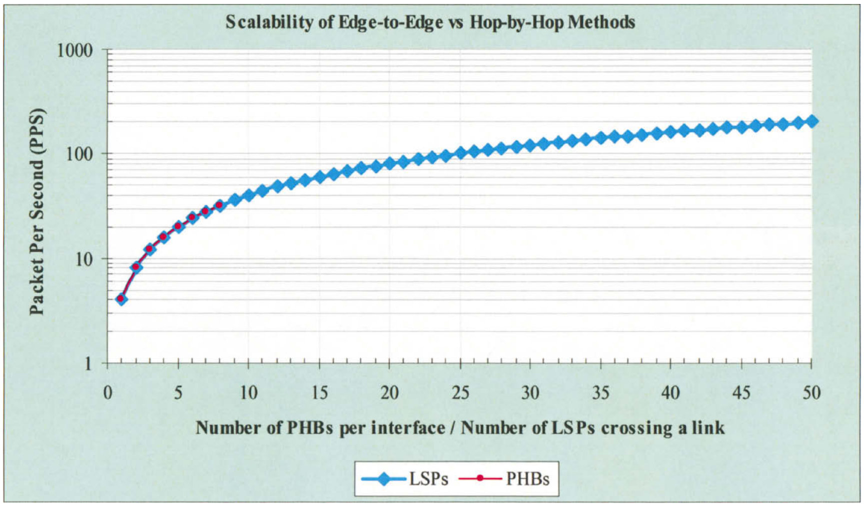

Fig. 9. Scalability of hop-by-hop vs. edge-to-edge methods.

that the scalability claim described in Section 4.5 for the hop-by-hop method is justified.

For scalability reasons, SLSMon is hosted on a different platform to Net-Mon as a separate component to avoid excessive load. Experiments indicate that SLSMon can cope with a potential large number of SLSs, since it retrieves active aggregated performance measurements at path/LSP level and only traffic measurements at the SLS level for its computations.

7. RELATED WORKS

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) working groups such as IP Performance Metrics (IPPM) (see [9] for more details). The development on monitoring systems is mainly focused on network performance monitoring especially path performance analysis and network-wide measurement infrastructure. This includes activities in RIPE Network Coordination Center [16], NIMI (National Internet Measurement Infrastructure) [17], CAIDA (Cooperative Association for Internet Data Analysis), IPMA (Internet Performance Measurement and Analysis Project) [18], and NLANR (National Laboratory for Applied Network Research) [19]. RIPE has implemented a number of the measurement protocols defined by IPPM. RIPE Test Traffic measurement project measures one-way delay and packet-loss, bandwidth, etc., between installed measurement probes (test boxes) in the Internet. NIMI is a software system for building network measurement infrastructures and managing measurement tools. NIMI relies on the servers to be deployed at different points of the Internet. 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 performance analysis and modeling are addressed by macroscopic topology analysis research at CAIDA. CAIDA measurement environment uses the Skitter tool for gathering QoS related data and evaluation of network path in terms of reachability and connectivity. All these systems interpret and analyze the collected data by post-processing and analysis.

There has been some work at the intra-domain level to use measurement information for tackling network performance degradation and managing congestion in operational networks in real-time as well as addressing real-time service level monitoring. NetScope [20] provides a unified set of software tools for traffic engineering of IP backbone networks by using network measurements to update the network configuration set-up in nonreal time fashion. Rondo [21] is designed as an automated control system using a monitoring system to react to and manage congestion in MPLS (Multi-Protocol Label Switching) Traffic-Engineered (MPLS-TE) networks in near real time. Regarding the SLA monitoring, a generic system for monitoring of VPNs is presented (see de Turch *et al.* [22]). Also, a system for monitoring the performance of SLAs based on aggregation is described by Lin and Chan [8], but a single class best effort is only considered without any considerations for multiple differentiated classes or for traffic engineering required monitoring. Dilman and Raz [7] studied a number of algorithms for minimizing the communication overhead of monitoring systems. This work is complementary to ours, and we actually use the notion of event-driven reporting based on threshold crossings.

There is also ongoing work on monitoring and measurements for traffic engineering at inter-domain level. The inter-domain QoS Monitoring aspects are addressed by different projects and activities in Europe and elsewhere. European projects include: NGNI [23], INTERMON [24], MoMe [25], SCAMPI [26], and so on. NGNI has a project among others on research and development activities in the areas of traffic measurement, monitoring and QoS in IP networks. The objective of the INTERMON project is to develop an integrated inter-domain QoS monitoring, analysis and modeling system to be used in multi-domain Internet infrastructure for the purpose of planning, operational control and optimization. The focus of MoMe is the enhancement of Inter-domain real-time QoS architectures with integrated monitoring and measurement. The objective of the SCAMPI project is to develop an open and extensible network monitoring architecture including a passive monitoring adapter at 10 Gbps speeds, and other measurement tools to be used for denial-of-service detection, SLS auditing, quality-of-service, traffic engineering, traffic analysis, billing and accounting.

8. 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 performance targets. Traffic engineering relies on measured data for off-line proactive and dynamic reactive operations. In this paper, we identified first the measurement requirements for traffic-engineered IP networks offering QoS-based value-added services. These include node, network, and service monitoring; aiming to facilitate traffic forecasting, route calculation according to expected traffic, dynamic resource allocation to deal with traffic fluctuations and user service auditing. 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 novel principles for designing monitoring systems and scalable methodologies for event monitoring and measurement statistics used for network operation and in-service performance verification of offered services. We presented a scalable *Monitoring* system architecture designed and built based on the outlined principles and its assessment. We highlighted guidelines about how to assess the measurement accuracy, benefit/cost, and scalability of such a monitoring system. The proposed

Monitoring system is distributed in order to guarantee quick response times and to minimize necessary management traffic. This ensures small reaction times and helps maintain stability as the network size increases. Based on the assessment results, we showed that the proposed *Monitoring* system provides good accuracy for both one-way delay and one-way packet loss while it also provides highly comparable edge-to-edge and hop-by-hop results. We also demonstrated the ability of the *Monitoring* system in providing measurements in relatively short timescales at the path and PHB granularities in order to assist various management and control functions. Finally, we discussed the issues concerning the scalability of the *Monitoring* system. In summary, we believe that the presented principles result in scalable monitoring systems that can contribute towards operationally optimized traffic-engineered networks that can support large number of customers.

ACKNOWLEDGMENTS

This work was undertaken in the Information Society Technologies (IST) TEQUILA project and was partially funded by the Commission of the European Union. The authors would like to thank their colleagues who participated in the TEQUILA project for the fruitful discussions, which have also contributed to the ideas presented in this article.

REFERENCES

1. D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and 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, December 1998.
3. D. Goderis (ed.), Service level specification semantics, parameters, and negotiation requirements, IETF Internet Draft: draft-tequila-sls-01.txt, work in progress, February 2002. Available at: *www.ist-tequila.org/sls*.
4. D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, Overview and principles of Internet traffic engineering, IETF Informational RFC-3272, May 2002.
5. D. L. Mills, Network time protocol (version 3) specification, implementation, IETF Draft Standard RFC-1305, March 1992.
6. E. Rosen, A. Viswanathan, and R. Callon, Multiprotocol label switching architecture, IETF Standards Track RFC-3031, January 2001.
7. M. Dilman and D. Raz, Efficient reactive monitoring, *IEEE Journal on Selected Areas in Communications (JSAC)*, Vol. 20, No. 4, May 2002.
8. Y.-J. Lin and M. C. Chan, A scalable monitoring approach based on aggregation and refinement, *IEEE Journal on Selected Areas in Communications (JSAC)*, Vol. 20, No. 4, May 2002.
9. V. Paxson, G. Almes, J. Mahdavi, and M. Mathis, Framework for IP performance metrics, IETF Informational RFC-2330, May 1998.
10. P. Trimintzios, I. Andrikopoulos, G. Pavlou, P. Flegkas, D. Griffin, P. Georgatsos, D. Goderis, Y. T'Joens, L. Georgiadis, C. Jacquenet, and R. Egan, A management and control architecture for providing IP differentiated services in MPLS-based networks, *IEEE Communications Magazine*, Vol. 39, No. 5, May 2001.

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. J. Case, M. Fedor, M. Schoffstall, and J. Davin, A Simple Network Management Protocol (SNMP), IETF Standard RFC-1157, May 1990.
13. K. Chan, J. Seligson, D. Durham, S. Gai, K. McCloghrie, S. Herzog, F. Reichmeyer, R. Yavatkar, and A. Smith, COPS usage for policy provisioning (COPS-PR), IETF Standards Track RFC-3084, March 2001.
14. S. Shalunov, B. Teitelbaum, and M. Zekauskas, A one-way delay measurement protocol, Internet Draft, *draft-ietf-ippm-owdp-03.txt*, work in progress, April 2002.
15. A. Asgari, P. Trimintzios, M. Irons, G. Pavlou, S. Van den Berghe, and R. Egan, A scalable real-time monitoring system for supporting traffic engineering, Proceedings of the IEEE Workshop on IP Operations and Management (IPOM 2002), Dallas, Texas, pp. 202–297, October 2002.
16. RIPE NCCs Test Traffic Measurements, *http://www.ripe.net/test-traffic/index.html*.
17. 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.
18. The Internet Performance and Analysis Project (IPMA), *www.merit.edu/ipma/*.
19. NLARs Active Measurement Program (AMP), *http://www.nlanr.net/*.
20. A. Feldman *et al.*, NetScope: Traffic engineering for IP networks, *IEEE Network Magazine*, Vol. 14, No. 2, pp. 11–19, March/April 2000
21. J. L. Alberi, Ta Chen, *et al.*, Using real-time measurements in support of real-time network management, *Proc. of the Second Workshop on Active and Passive Measurements (PAM2001)*, Amsterdam, April 2001.
22. F. de Turck, S. Vanhastel, F. Vandermeulen, and P. Demeester, Design and implementation of a generic connection management and service level agreement monitoring platform supporting the VPN service, *Proceeding of the Seventh IFIP/IEEE Integrated Management Symposium (IM'01)*, Seattle, USA, pp.153–166, May 2001.
23. Next Generation Network Initiative (NGNI): *http://www.ngni.org/default.htm*.
24. Advanced architecture for INTER-domain quality-of-service MONitoring, modeling, and visualization (INTERMON): *http://www.ist-intermon.org/*
25. Cluster of European Projects aimed at Monitoring and Measurement (MoMe): *http://www.ist-mome.org/*.
26. A Scaleable Monitoring Platform for the Internet (SCAMPI): *http://www.ist-scampi.org/*.

Abolghasem (Hamid) Asgari 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 University of Wales, Swansea, in 1997. He was with Iran Telecom. Research Center (ITRC) from 1986–1990 as a research engineer. He joined Thales Research and Technology (UK) in 1996, where he is an Assistant Chief Engineer specializing in the data communication systems/networks. He has been involved in simulation, design and analysis of integrated IP services and network architectures. He was active in the IST TEQUILA project mainly focusing on Intra-domain QoS monitoring at network and service levels and system level performance evaluation. He is currently working on IST MESCAL project. His research interests are in IP QoS technologies, QoS-aware monitoring and network performance evaluation.

Panos Trimintzios holds a B.Sc. in Computer Science and an M.Sc. in Computer Networks, both from the Computer Science Dept., University of Crete, Greece. Until 1998 he was a research associate at ICS-FORTH, Greece, working on research projects involving high-speed network management and charging. Currently he is a Research Fellow at the Center for Communication Systems Research (CCSR), University of Surrey, UK, where he is also at the final stage of his Ph.D. studies. At CCSR he has worked in numerous networking-related research projects funded by the European Commission and the UK research founding bodies. His main research interests include traffic engineering, architectures for resource management, policy-based networking, network monitoring, and service management.

Mark Irons received a M.Eng. in Electronic Engineering from the University of Wales, Bangor, in 1997. He joined Thales Research and Technology in 1996, where he works as a software engineer in the Networks Group. He specializes in distributed, object-orientated software design and implementation. He was involved with the IST TEQUILA project, where he contributed to the design and implementation of a distributed monitoring system. He is currently involved in research in the area of policy-based network management.

Richard Egan 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 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, that 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 Project TEQUILA and MESCAL and is Chairman of the Interworking of Networks Steering Group in the UK Mobile Virtual Centre of Excellence. His main interests are in routing, QoS, and self-organization in next generation wireless networks.

George Pavlou is a Professor of Communication and Information Systems at the Center for Communication Systems Research, School of Electronics and Computing, University of Surrey, UK, where he leds the activities of the Networks Research Group. He holds 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, UK. At Surrey he is responsible for a number of research projects and industrial collaborations. His research interests include network planning and dimensioning, traffic engineering, policy-based management, novel architectures and technologies for network and service management, multimedia service control, programmable/active networks, mobile ad-hoc networks and communications middleware. He was technical program chair of IEEE/IFIP Integrated Management 2001 and has contributed to standardization activities in ISO, ITU-T, TMF, OMG, and IETF.