

# Application Management to support Network Management and Traffic Engineering

Kyoungwon Suh and Brett Vickers  
kwsuh,bvickers@cs.rutgers.edu

Dept. of Computer Science  
Rutgers University  
110 Frelinghuysen Road  
Piscataway, NJ 08854-8019

DCS-TR-425 \*

May 2, 2001

---

\*This work was supported by grants from Telcordia Technologies (formerly Bellcore) Inc.

## **Abstract**

This paper is motivated by the observation that gathering and analyzing the behavior and performance of applications leads to more informed decisions by network managers. In particular, application management can be used to improve network traffic engineering tasks. In this paper, we investigated the existing/potential network monitoring frameworks and showed how they can be effectively used to extract application-level information from network applications, running environments and the communication links. We also identified important application-level information for network management and traffic engineering, which couldn't be easily supported by the simple network-level monitoring information.

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Classification of application monitoring mechanisms</b>	<b>6</b>
2.1	Packet Sniffers . . . . .	7
2.2	Application Instrumentation . . . . .	9
2.3	Instrumentation of Service Object . . . . .	12
2.4	Intelligent Agent . . . . .	13
2.5	Active Measurement . . . . .	15
<b>3</b>	<b>Types of Information</b>	<b>17</b>
3.1	Identification of Application Flows . . . . .	17
3.2	Performance Data of running Applications . . . . .	18
3.3	The Behavioral Characteristics and Status of Applications . .	20
3.4	Future Expectation . . . . .	21
3.5	Historical Data . . . . .	22
<b>4</b>	<b>Potential usage of application-specific information</b>	<b>23</b>
4.1	Fault Management . . . . .	23
4.2	Performance Management . . . . .	24
4.3	Configuration Management . . . . .	27
4.4	Security Management . . . . .	29
4.5	Accounting Management . . . . .	30
<b>5</b>	<b>Conclusion and future work</b>	<b>32</b>

# 1 Introduction

The basic function of network management is guaranteeing that network applications are able to utilize available or assigned network resources efficiently and consistently. Since meeting this goal demands careful attention be paid to a large number of complex issues, the International Standards Organization has classified network management tasks into five categories in order to make the problem of network management more tractable. These categories are fault management, configuration management, accounting management, performance management, and security management.

More recently, however, researchers and network operators have begun paying increased attention to a sixth category of management tasks: *application management*. Sturm [29] defines application management as the sum of the processes used to monitor and control the software elements that make up application systems. The motivation behind application management originates from the observation that network users are primarily concerned with end-to-end services. Using existing network management approaches, enterprises may know whether their networks are up or down, but they cannot know whether their users are experiencing adequate application performance or whether a new application will perform acceptably once deployed in the network [8].

There are several reasons for the trend toward application management. First, much of the information of interest to network managers resides with the applications or the users themselves, not in the network or in the packets generated by network applications. For instance, only an application's users know if the end-to-end performance they are experiencing is tolerable; such information cannot be gleaned by monitoring network equipment or packets on network links. Second, new software technologies are rendering many network-oriented management tasks ineffective. Secure IP, for example, encrypts the contents and headers of IP packet, rendering network managers blind when attempting to determine, say, the port number to which a packet is destined. Third, the introduction of new transmission/switching technologies such as all-optical switching (Lambda switching) make the monitoring and control of each flow very difficult. Also, the increasing speed of transmission and switching technologies make the online analysis of each flow a real challenge job.

The concept of application management generally covers such broad topics as inventory management; software distribution, configuration and update; application performance management; and application fault management. Among the many aspects of application management, we are partic-

ularly interested in application performance and fault management.

This paper is motivated by the observation that gathering and analyzing the behavior and performance of applications leads to more informed decisions by network managers. In particular, application management can be used to improve network traffic engineering tasks.

The remainder of the paper is organized as follows. Section 2 overviews the mechanisms for application performance monitoring and discusses related work in this area. Section 3 presents a taxonomy of the different types of monitoring information that can be gathered using application management. Section 4 discusses some of the potential uses for the management information introduced in section 3, with heavy emphasis on the use of application management information to improve traffic engineering. And section 5 provides some concluding remarks.

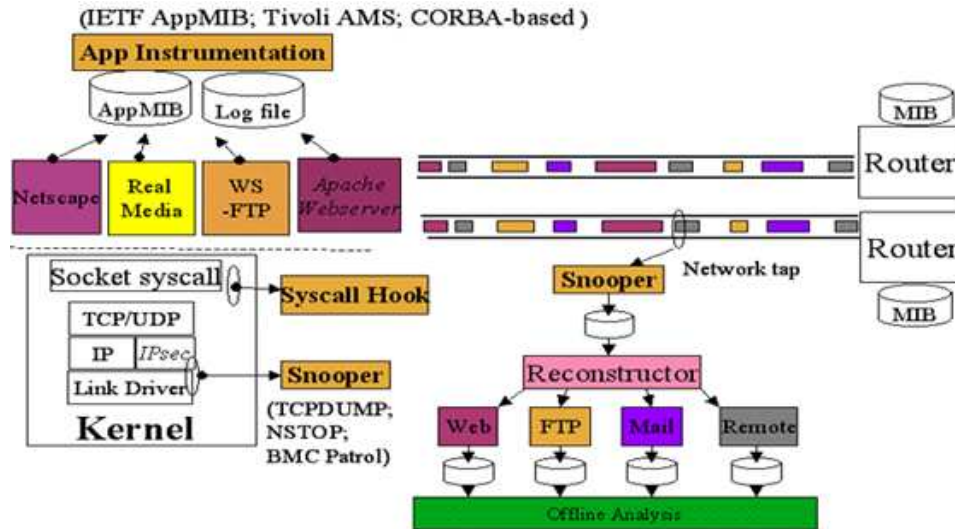


Figure 1: Application Monitoring Mechanisms

## 2 Classification of application monitoring mechanisms

The characterization of application performance and behavior can be performed from several vantage points. As a first approximation, application performance may be measured at one or both of the end-points of network communication. That is, it may be measured at the server or at the client. It may also be measured at a location close to the server or the client, perhaps on a host with promiscuous monitoring capability. In fact, any point between the client and the server can be used to monitor application performance [30]. However, the types of information available to the performance monitor will vary depending on the choice of monitoring location. In this chapter, we will survey several proposed frameworks for monitoring and controlling the performance of applications. These frameworks are classified according to the way in which performance information is gathered and analyzed [31]. Figure 1. shows the various monitoring mechanisms which are covered in this chapter.

## 2.1 Packet Sniffers

A packet sniffer is any host capable of monitoring packets being exchanged by a client and a server. Perhaps the most well-known form of packet sniffer is a host with an Ethernet interface configured to operate in promiscuous mode; all packets transmitted on the Ethernet are monitored by the promiscuous Ethernet host. Packet sniffing mechanisms have been widely used to monitor network flows, and many of them are based on the well-known “tcpdump” tool. When the Internet was smaller, it was not difficult for programs like tcpdump to identify individual application flows. Standard applications such as email, ftp and telnet use well-known port numbers, so the network application corresponding to each packet was easily determined by monitoring packet headers. However, with the growth of the Internet and the number of the Internet applications, the meaning of port numbers is becoming less obvious. The increasing proliferation of web browsers and Microsoft Winsock make the misclassification problem more serious. For example, BackWeb push-technology and Streamworks or VDOLive multimedia clients use UDP ports that are either assigned to or used as defacto-standards by other network applications such as Citrix, H.323 Gatekeeper and RealAudio [14, 15]. Moreover, the restrictions placed on users by security firewalls has led a number of software developers to write their applications to tunnel data through well-known ports, leading packet sniffers to misidentify the source of the data. For example, RealAudio server and client can be configured to tunnel audio data over the well-known HTTP port 80. These examples illustrate the problem with mapping a packet’s port number onto an application. Without correct identification of application flows, a packet sniffer cannot accurately monitor the activity and performance of network applications. Thus, instead of simple mapping, many packet sniffing tools are made more complex in order to understand what application protocols may be doing. In other words, they attempt to reconstruct the behavior of the application protocol and trace the protocol’s performance with each sniffed packet.

Due to the large volume of traffic in most networks, many packet sniffers analyze application performance off-line after the data has been collected. However, there are several packet sniffing approaches that provide real-time identification of application flows. Nevertheless, the identification of application flows is only a first step in the monitoring of network applications; obtained detailed performance information about application flows is also often necessary.

Since application specific information is not directly available to packet sniffers, they must be analyzed based on the specification of application

protocols and heuristics. Some of important application-specific data can be measured and analyzed from the analysis of sniffed packets. Application response time is one of the most common application performance metrics obtained by these heuristic, primarily because it is so general and because it applies to so many different applications. Application response time is typically measured by observing the request-response structure of an application's communication. For instance, by matching up a transport protocol's data and acknowledgement sequence numbers, a general response time can be determined for many applications [1]. In order to obtain other measures of application performance, more detailed knowledge of each packet's application protocol is usually required. The challenges of online protocol reconstruction is described by Feldmann [10].

The primary advantage of the packet sniffing approach is that it obtains general application performance measurements without needing to modify existing network applications. Moreover, the measurements provided by packet sniffers reflect what is actually taking place within the communication network. Packet sniffing's weakness, however, is that it does not always provide network managers with an accurate view of the application's performance. For example, response time is not simply a function of packet transmission delay; it is also a function of host processing. In order to obtain truly accurate application response times, a network manager should really probe the clients. Also, only applications which use well-known protocols and port numbers can be easily analyzed by packet sniffers. Another problem that complicates the use of packet sniffers is the problem of asymmetric packet routes. For example, an application's request and response packets may take different paths, making it impossible for a packet sniffer on one of the paths to observe both requests and responses. Finally, not all network locations are amenable to the placement of packet sniffers; many points in an application's end-to-end path are inaccessible to network managers (e.g., links in another administrative domain, links within a hardware switch, etc.).

Despite their weaknesses, a number of packet sniffing products have been developed. AT&T Laboratory's PacketScope is one example. It is a high-performance packet sniffing system for the monitoring and analysis of IP headers in a large network [4]. Unix workstations are used as base systems for packet capture, and tcpdump stores captured packet headers to disk. In addition to simple monitoring of several standard application protocols, PacketScope is capable of monitoring multimedia traffic and tracing HTTP protocol.

NetScout is another well-known packet sniffer [1, 18]. It is actually a

family of packet sniffers that includes probes for the monitoring of shared LANs, switched LANs, WANs, and frame relay links. Using the NetScout Manager software, it is possible to decode 11 protocol stacks through all seven layers. The NetScout family of probes supports real-time analysis of application response time and network utilization per application.

EcoScope is a packet sniffer that uses software probes called “super monitors”, which run on appropriately placed packet sniffers equipped with promiscuous-mode interfaces [17]. These probes automatically discover the network’s topology and recognize the protocols of hundreds of applications. Like NetScout, EcoScope examines typical request and response semantics in order to discern actual application performance metrics including session-level response times, application response times, and network latency. Using this information, the EcoScope Interactive View provides a high-level view of the applications, servers, workstations and protocols running in a network.

Application Vantage is a packet sniffer that relies on “capture agents” to collect data for use in measurement and troubleshooting [19]. The capture agent, a PC running the Microsoft Windows operating system, observes a network’s data through its promiscuous interface and filters packets associated with selected criteria such as specific IP addresses, MAC addresses, or IP sockets. When the capture is completed, all filtered data is compressed and transmitted to the performance monitor, where application protocols of several major business applications are identified and their performance metrics, such as response time, are measured.

NTOP is a packet sniffing software tool developed under the GNU public license, so its source code is freely available [7]. Like many of the other packet sniffers, NTOP observes and captures packets passing through a network. However, it only identifies and monitors packets identified as using standard protocols such as FTP, HTTP and NFS.

## 2.2 Application Instrumentation

Instrumentation of application source codes is one of the most natural ways to manage network applications. Source codes are modified to report performance data or to export API to allow external management applications to get the performance data. The exported API may also provide control functions for the application. That is, application developers voluntarily provide management interfaces or information report to network manager in this approach. In the simplest case, applications are not required to provide any management interface for management servers and they are just asked to generate “status” or “performance statistics” report [26]. Depending on the

types of communication mechanisms that the application agent adopts, the report may be written to Management Information Base (MIB), the interface of application agent can be called by management applications, or the report can be directly delivered to a management station. As a most complicated form, the applications can be instrumented to export management interface which allows network managers to get statistics and to control the behaviors of applications. Therefore, the instrumentation is most straightforward way to provide the capabilities to manage application from the initial implementation of applications.

One of the benefits to use the application instrumentation is that the appropriate granularity of monitoring and control can be easily adjusted. As an example, Tivoli's Application Response Measurement API (ARM) shows how the subtransactions can be traced and how the subtractions can be related to the parent transaction. By providing more detailed view of subtractions and relationships between them, Tivoli ARM architecture makes it easier to pinpoint the location where the performance degradation occurs [33]. Also, the correlation of subtractions with the parent transactions allows to observe the overall performance, especially in Web applications. The Desktop Management Task Force(DMTF) in 1998 formed a new subgroup called "Distributed Application Performance", which is another good example of application instrumentation. The DMTF is developing a model for application performance that which is consistent with other Common Information Model (CIM) models and with the ARM API. Based on these on-going efforts, it is highly expected that the ARM is accepted as standard for application response time measurement in near future.

A strong point of application instrumentation is that it can provide virtually every kind of information which is related to an application under the control of application developers and network administrators. Application MIB and WWW service MIB of IETF are good examples, which illustrate the powerful features. The Application MIB makes it possible to accomplish the following features: capturing measurement of application throughput, allowing managed applications to identify their units of work, enabling application response time monitoring capabilities, monitoring resource utilization, enabling the control of applications such as start, stop, suspend, resume, and configure [29, 20]. The WWW Service MIB consists of three component groups such as service information, protocol statistics, and document statistics. The service information group has a single table that identifies the WWW services that are to be managed. It also contains some administrative network management information. The protocol statistics group holds information about traffic activities for a Web service, as

well as information about certain DTP protocol operations. The document statistics group collects information about the documents being accessed [29].

Among the various performance metrics of applications, application response time has been the most popular metric for existing monitoring mechanisms to measure application-level performance. Initially, Tivoli ARM version 1.0 introduced a way to measure service level-transaction status and response time. The following standard, ARM Version 2.0, enhanced this adaptability by permitting applications to provide three types of information. The first one is the parent/child relationship between transactions and their subtransactions within one system or across systems. The second one is the measurements about the transaction (such as the number of bytes in a file transfer) or the state of the application (such as the number of threads being used). The last one is the diagnostic information (such as an error code of the text of a failing SQL Query). We can simply imagine how it would be difficult to measure/analyze the detailed metrics and correlate them from the packet sniffers. The application response time have been accepted as the most important metric for the application performance management in enterprise network and the architecture of ARM gives the much flexibility to adjust the level of monitoring granularity. Henceforth, the broad acceptance of this mechanism is quite promising.

Another interesting architecture based on application instrumentation is Berkely SPAND [28]. The SPAND shows the importance of shared application specific metrics, which can be effectively gathered only from application instrumentation. In the SPAND architecture, client applications are instrumented to measure the state of the network while communicating with distant network hosts. When communication is finished, clients create performance reports that summarize the observed performance to distant hosts and send these reports to performance servers. These reports are application-specific and SPAND introduced metrics of several typical applications. SPAND measures the following characteristics of the data transfer for bulk transfer application: the available bandwidth for the connection, the round trip time for the connection, and the time to completion for a specific transfer size of B bytes. For HTTP statistics, it measures and reports web object download time as a primary metric for HTTP applications. For real-time applications such as RealMedia, it measures the rate of missing frames and the receiving rate considering the effect of retransmission in TCP-level.

With Tivoli's Applications management Specification (AMS), Tivoli introduces the idea of Tivoli-ready applications and the tools that manage them. The applicaitons are programmed based on AMS specification to pro-

duce necessary standard information about applications themselves and also export some control interfaces. AMS covers the broad areas of application management such as structure and topology, deployment and distribution, installation, dependency checking, monitoring events, verification, and operational controls.

Though the application instrumentation has so much architectural benefits, it has not been widely accepted because of several drawbacks. Since instrumentation entails costs and extra development time, the developers have been reluctant to do it. The lack of standards also deterred the deployment of this mechanism. However, the situation is being changed. The information model for applications is being standardized and Tivoli's AMS is one of the candidates in that direction. Nowadays, especially the enterprise users definitely want the manageability in their business applications and they started to understand the benefits of instrumenting application from the beginning stage of application development. Application designers and developers know better than anyone the appropriate way how their new applications are monitored and managed.

### **2.3 Instrumentation of Service Object**

The main concept residing in the instrumentation of service objects is not quite different from the instrumentation of application. Nonetheless, this has to be classified into an independent category because it does not require application developers' much efforts and also it is based on a separate component model. One of the benefits in this approach is that the application developers don't have to worry about the details of manageability and the monitoring capability.

Parnes [23] proposed a new framework to manage distributed applications. The basic assumption of the framework is that applications are built following an agent-based architecture, where applications consist of a number of different agents which encapsulate separate functionalities. For instance, a video-agent would be responsible for capturing and displaying video data and a database-agent would act as an interface to a database engine. Numerous agents can be deployed within an application based on his proposal. To give the manageability to applications which use the various agents, the agents are augmented with the functionality of realtime reporting and control.

The disadvantage of the instrumentation of service objects is that it is not always possible to provide general components, especially common modules for a given application's requirement. Since programming libraries are

scarcely shared between the products of different companies, there seems to be little chances that common component modules enhanced with manageability functions are introduced into the different kinds of application products. The several drawbacks of the applications instrumentation are also true for this approach similarly.

## 2.4 Intelligent Agent

It is obvious that the server machines or the client machines where the applications run are the best place to see the users' view of the activities and the performance of the applications. The main concept that intelligent agents are loaded on users' machines such that they can monitor exactly what users see without instrumentation of application or service objects. The intelligent agents recognize network request/response activities of end clients and determine the appropriate application transaction requests associated with this activity. Generally, the agents are composed of several monitors such as transport-layer monitor and process monitor. Unlike packet sniffers, intelligent agents only monitors the activities within the host where the agent is located. It is also different from instrumentation mechanisms because it does not require the modification of existing applications. That is, the tools which belong to this category may directly interact with all the available information sources on application activities in the host without any modification of applications or libraries.

It can be easily imagine that that the environment where the application is running might be a good source. it is true that the operating system can provide realtime information on the application because an application runs as a process created by operating system. The capabilities to monitor and control processes in operating system reveal the status of a process and provide the control mechanisms such as suspend and terminate, though the control mechanisms are not graceful. Moreover, by monitoring the TCP/IP stack which is implemented inside OS kernel, the the current status of the protocol layers can be traced with the detailed view of incoming/outgoing packets. A newly emerging candidate would be the hooking of library API or of OS system call API, which can provide more detailed view of application activities.

One of the real technical issue is the cost of deployment. Since an management agent has to be installed on user's machine in this mechanism, the following principles are considered important: minimal change in the client machine, minimal cost of deployment, and minimal resource consumption. According to a Lucent whitepaper [30] generally, there is a fair amount of

computing resources in client/server machines, which can be set aside for analysis during the time, when the agent is collecting TCP and application-level data buffers. Henceforth, simple analysis does not produce big performance problem [22]. Nonetheless, the agent should minimize its measurement locally as far as possible and the amount of the transmitted information to management server has to be as small as possible [11].

It is true that the accuracy of the measurement can be improved if we are able to observe and correlate measurements from multiple vantage points such as client machine and a server machine. However, there are technical challenges in correlating raw data from multiple sources in real time [30]. Therefore, generally only the locally monitored data at a single point in client/server machine are analyzed without taking risk to try to correlate the two datasets on real time. Compared to network-centric packet sniffers, these mechanisms make it easier to get the detailed analysis of application performance.

More recently, system call (syscall) hooking and library hooking in operating systems are proposed as alternative methods to capture and analyze the application performance and its behavior without requiring the actual instrumentation of every application. Based on the authors' knowledge, there is few real implementation or products which use the hooking methods for network monitoring. Since the main goal of application management has been focused on the monitoring service level agreements (SLA) which is based on application response time, the demands to monitor the detailed behavior of each application have not been large. However, the demands to monitor the detailed behavior and performance metrics of applications are increasing and the hooking mechanisms start to be considered as one of the potential monitoring methods. Especially in these schemes the behavior data of applications can be gathered even in the case that the monitored application is using secure protocols such as IPsec. Currently, we are working to develop a monitoring mechanism based on this concept to monitor IPsec applications. The log file which is generated by applications for the report of errors and performance may be a good starting place to analyze the problem and the performance of applications. However, since this method needs the cooperation of applications to generate the log file, its usage is limited.

The application-specific information which can be gathered from this category is generally richer than the ones which can be gleaned in the network-centric packet sniffers. The software modules used by applications such as TCP/IP stack and OS components are priorly known to the intelligent agent such that more accurate application-level activities can be analyzed. The cost of operation is also very small because the costly high-performance

sniffers and analyzers are not necessary. The obvious advantage is that it provides true end-user view, delivering measurements that are actual. The disadvantage is that it requires resource such as CPU, memory, and disk on every client, and it adds codes to desktops. Another difficulty is that it needs end-users' agreement on the installation of agents on their local machines.

Several products and proposals could be found which belong to this approach. FirstSense Enterprise collects information from Win-32 or Solaris agents and sends it both to a console application (for real-time monitoring) and to a SQL database. By using baselining mechanism, it can automatically detect when desired service levels of a particular user are not being met. Usually application response time is used as an important performance metric to be monitored and it is targeted primarily at database applications [1].

Like Firstsense, BMC Patrol requires that its own agent (PATROL agent) be resident on every end user's system. BMC's agent is designed to function independently of the manager. An interesting component of Patrol is the PATROL Knowledge Modules. The Patrol Knowledge Module is comprised of one or more files from which the PATROL Agent obtains the information that it needs to monitor and manage an object of interest called an application class. An application class can be any physical or logical system entity that is possible to monitor, such as major component of an OS such as memory or a commercial software application. The agent potentially may monitor any parameters related to a specific managed application. The data collected is stored locally in history files. The data in the history files can be queried by the manager or even transferred to the remote manager for further analysis [29].

The Lucent Technologies' VitalAgent adopts similar architecture as the BMC Patrol and FirstSense. A key principle employed by VitalAgent is to interpret the current performance measurement against previously observed measurements where the agent is installed. Measurements are then used to establish historical baselines in the form of performance distributions. Using these distributions, performance indices are computed and presented to the vitalAgent user in a more useful form [30, 22].

## 2.5 Active Measurement

Active measurement is usually used as an alternative method to measure application performance when the collection of performance data by passive mechanisms are insufficient. They deliberately add load to a network and then record how well, or how poorly, it responses [1]. Active measurement do not rely on any monitoring capabilities built into the networks, servers,

and service applications. The measurement can be initiated from multiple locations where it is possible to provide closer approximation of customer's perceived performance [5].

Under the Hewlett Packard's Firehunter system, several active measurement tools are developed and deployed. In web service measurement, like a typical web client retrieving a web page, this web service test resolves the IP address of a target web server, establishes a TCP connection with the web server, and then issues a GET request to a specified web page. By interpreting the HTTP response header returned by the server, the overall response time to retrieve the web page can be determined. The individual response time components such as DNS resolution time, TCP connection time, server response time, and data transfer time can reveal different potential bottlenecks. The E-mail delivery and retrieval test can be used to measure the performance E-mail service. Like the response time components in Web service, the response time component in email service can also be analyzed. Besides above two services, synthetic transaction can be constructed to test the performance of each application.

In AT&T Labs, the WorldNet In-Process Metric (WIPM) system provides an infrastructure for measuring edge-to-edge performance of the IP backbone using a suite of active measurement. The system currently measures and reports round-trip packet delay, round-trip packet loss, throughput, and HTTP transfer delay, including domain name service (DNS) resolution time, across the fully connected mesh of the IP backbone.

The advantages of active measurement are obvious. The measurement results are consistent. Since scripts are always played back the same way, the administrator can be sure that any variations in performance are due to network performance [1]. However, it is known that there are considerable weaknesses in this mechanism. The biggest concern is that the results of active measurements are artificial. If the artificially mimicked application flows do not exactly represent the real ones, the measurement data may be misleading. The size of packets and the intervals of each packet of a flow drastically affect the measured performance so they should be carefully selected. The other disadvantage is that it generate additional traffic over the network. Also, it cannot measure the current status of behavior of applications running in the network.

## 3 Types of Information

In this chapter, we investigate the types of information which is valuable for network management, traffic engineering and quality of service control. The types of useful information which can be gleaned by monitoring application activities are classified into five categories. The gathered information is not limited to the metrics which are currently used in network management frameworks, but also covers the potential information for application and network management.

### 3.1 Identification of Application Flows

The correct identification of an application flow is the most important process before we decide to utilize any other kind of application-specific information for network management and traffic control. The basic concept of the identification means the discovery of every protocol layer related to a flow, especially including application-layer protocols. While the identification of a flow is relatively easy for the mechanisms based on the instrumentation, it's very difficult for network-centric packet sniffer to do the same job effectively. It is well known fact that many application protocols, especially peer-to-peer multimedia application protocols, dynamically choose non-standard TCP or UDP port numbers cheat firewalls. Thus, we cannot simply rely on the well known port numbers for flow classification as we did for traditional traffic types such as ftp, telnet, and http [4].

Furthermore, when Secure IP such as IPsec is engaged especially, it is virtually impossible for packet sniffers to get any kind of information, including application protocol engaged. It is true that several proposal/attempts are made to overcome the limitation of packet sniffing architecture, and AT&T PacketScope is one of the good examples. In PacketScope, the packets of session control protocols are analyzed to identify the dynamically chosen port numbers. However, their solutions are still limited to a few multimedia applications which use standard session control protocols [21].

Flow aggregation can be another useful application-specific information in terms of flow identification. Aggregation refers the addressing level at which traffic is combined to form a flow. For example, a flow may be defined to contain all traffic flowing between two IP hosts such as host-to-host flows. Alternatively, a flow may contain only a port-to-port flow [9]. In this paper, a flow means a port-to-port flow and the flow aggregation means identifying and grouping of all the flows in an application. The aggregated flows of an application usually have a same source IP address but they may have dif-

ferent destination IP addresses. Generally multiple flows need to be applied with same routing and admission policies, so the correct aggregation of flows is considered important.

### 3.2 Performance Data of running Applications

Based on our observation, the true performance of network application is sometimes quite different from the measured performance of network/transport layer because of application-specific handling of data frames. This fact creates the need to define appropriate application performance metrics and the right methods to measure the performances correctly.

As stressed throughout this paper, application response time is considered one of the most important performance metrics in terms of application performance. Many of the network applications require appropriate responsiveness from the servers or the peer applications they are communicating with. Especially in corporate environments, the response time of each critical network applications such as SAP, email, and PeopleSoft directly affect the productivity of the company. One of the good things in response time metric is that it can be applied to most of the applications generally. It is also relatively easy for network provider and users to make a Service Level Agreement (SLA) based on the application response time. Conceptually, when the currently measured application response time grows larger than the tolerable level, that event can be reported to the network administrator such that the investigation of the problem can be initiated by that input. Response time is generally composed of multiple parameters such as network delay, routing instability, throughput, client delay and server delay. Therefore, pinpointing the sources of the delay in the network application is not a easy job and the it usually requires at least two monitoring points, one is near the client application and the other is near the peer/server side. Barford [3] recently proposed methods that aid in locating the root causes of delays in TCP transactions by passively collecting/analyzing network traffic traces at the end of points during TCP transactions. It enables the accurate assignment of delays to either the network, the client, or the server. However, generally the different administration domain and the difficulties in correlating the monitored data at multiple points limit the practical usage of these approaches to single domains.

The loss rate of application-level frames is sometimes quite different from the one of network packets. This phenomenon especially manifests in real-time application protocols such as RealAudio, which resort to hard realtime requirement of packet arrival and the retransmission mechanisms. In these

realtime applications, the requested packet or retransmitted packet should arrive before its playback time. If a packet arrives in time, it is considered a "non-dropped" part of a frame. Otherwise, it belongs to a "dropped" part of a frame [28]. Since there is a strong correlation between consecutive packets, even the packets which arrived in time might be considered useless in application level. Therefore, it is very difficult to calculate the loss rate of application frame simply by counting the number of dropped packets from a network-centric sniffing host. Only the instrumentation of applications can give the accurate view of application level loss rate. Since the application-level loss rate directly affects the playback performance in realtime applications, getting the information is quite useful for performance management.

The connection failure of applications is one of the most undesirable events in the networked environment. The connection failures can be defined in several different forms. The most obvious one is the timeout of an application request. As an implicit example, in Web applications, the number of pressed reload buttons within a small amount of time indirectly shows an indication of a connection failure, even in the case that the timeout does not occur since the end user cancels the previous requests by pressing reload button continuously. These events should be recognized as critical problems in the network or in the opposite peer with which the end user is communicating so that they can be reported to an administrator.

The quality indicator of running applications inside network can be a metric for performance/fault management. Usually, network administrators cannot figure out exactly how the end-users feel about the performance of applications. Although the administrator could not find out any problem after monitoring routers and sniffing links for long time, the monitored data might miss the performance problems of applications. It is known that sometimes network administrators try to find out problems in networks only after they receive complains from end users. Therefore, it would be helpful for critical applications to be instrumented (or other mechanism can be used) to report the quality indication from time to time. The values of the indicator is per-application basis so it might be calculated by specific rules designed by application developers. It might be simple as response time, playback performance, and throughput, or it may be some linear combination of the individual metrics. Whatever the calculation rule is, the value of quality indicator is always combined with the requirements of the applications and its current status.

### 3.3 The Behavioral Characteristics and Status of Applications

Besides application performance, the current behavior and status of each network application are interesting information for network management. The start time and the finish time of each application can be measured and reported to the network administrator. They can be very useful information for usage charging and dynamic routing. Since the exact start/termination time of an application is not available, an arbitrary timeout value which is manually set by network manager is used to guess the finish time of each flow. ( Timeout refers to how long a flow is idle before it is considered closed.) For example, a flow may be defined as closed if no traffic at the chosen aggregation level has been detected in the previous 60 seconds [9]. However, the mechanism based on timeout value does not guarantee the correct information of the finish time of applications. Frequent reuse of port number may mislead such a decision based on heuristics.

It is desirable that all the internet traffic are TCP-friendly. Even though TCP protocol is not used as transport protocol in some application protocols, similar adaptation mechanisms are expected to be used to share limited network resources fairly and efficiently. One obvious way to encourage to adopt TCP-friendly mechanisms in applications is to apply different charging schemes and alternative would be rerouting/penalizing noncooperating (non-adaptive) flows to a separate path. In spite of its usefulness, it is very difficult and time-consuming to measure the adaptability of flows of an application by packet sniffing hosts. Thus, it would be reasonable that the adaptability of an application is measured at each server-side or client-side. By matching the loss rate of acknowledgement or the feedback information from a receiver with the change pattern of the transmission rate, it would be possible to analyze the adaptability of flows of an application.

The sharing of the information on hacking attacks will be helpful for protecting all the hosts inside network from hackers. Although several PC-based detecting tools for hacking attacks are available, there's no sharing mechanisms available now. The sharing of information on the virus infection will be also helpful for protecting all the hosts inside network from virus. Furthermore, the reporting of the information can be useful to track down the source and dispersion of the virus infection and to identify the location of the internal/external hackers.

The webservers under a Web-hosting farm of an ISP are maintained by the ISP personnel so that the usage logs are easily accessible. However, the usage logs of privately owned webservers and FTP servers are not usually

open to network administrators. One possible solution could be to require every webserver in local network to equip an interface for network administrators to check some limited information on user's access patterns such as hot service spots (URL) and the number of possible connections. IETF's WWW service MIB started to provide similar kinds of information for web servers [13].

Still a large number of Internet users use dial-up modems to connect to their ISP. The duration of each dial-up connection, the negotiated baud rate for the connection, and a web page retrieval time can be a good source for fault and performance management of the host and its applications which use dial-up connection. Also, the number of call failures until the dial-up connection is eventually established is an important metric. These information can be more easily obtainable from application agents if they are installed on users' machine.

### 3.4 Future Expectation

The application metrics which belong to future expectation of application's behavior have not been well covered until recently. However, the difficulty of management decision which originates from the uncertainty of the future behavior of applications can be minimized by gathering the known future behavior from applications. The following examples show the possibility to guess the future end time of an application. When a big file is being transferred in FTP or HTTP connection, the application knows the size of the file and the expected time for the activity. As a similar example, when a user starts to see a movie or listen to a music over the Internet, the size is known to the application. Also, from the known semantics of the contents which is delivered in IP packets, the approximate expected time of the activities can be calculated. For example, if we comes to know that a flow is normal voice calls, it can be assumed that the flow would occupy at most 8Kbps bandwidth on average. In some cases, the duration for each call may not be available from the application semantics. However, if the history of previous calls is maintained, we can statistically (or simply choose average value or maximum duration) guess the expected duration of the traffic.

The expected bandwidth of clients can be implicitly gleaned from applications in some situations. Several websites which serve audio/video web-casting provide multiple versions of URLs for a service with different transmission rates. In those websites, the clients are required to select one of the transmission rates at which they want to receive IP packets. By monitoring the user's selection, it is possible to gather the expected bandwidth of users

for the realtime audio/video services over the local network. Also, as an alternative simple approach, the average, minimum and maximum bandwidth requirements of a source can be specified directly from application as we did for the specification of QoS requirement in several QoS proposals. The type of media used in multimedia application protocol can implicitly show the upper bound of the required bandwidth.

Some routine or scheduled services which entail large and long-lived traffic such as remote disk backup and multicasting of distance learning programs are usually designed to keep its scheduled time. Since these kinds of services generate large traffic, if we know the starting/ending time of each service beforehand we can setup an alternative path before the service traffic is generated. The scheduler of users' system can be instrumented to report network-related scheduled jobs which may generate huge traffic.

### 3.5 Historical Data

Historical data of the metrics which were covered in previous subsections are sometimes valuable information for network management. Application MIB of IETF already adopted the idea and included the historical data for management purposes [20]. In Application MIB, the data for formerly open channels that were connections and the one for connection-specific historical information is kept in the former connection table. In the connection table, the following information is available for each connection: identification of the transport protocol in use, near-end address and port, far-end address and port, and identification of the application layer protocol in use. Also, the two transaction history tables provide per-transaction-kind breakdowns for channels carrying transaction-structured flows. Historical data can be averaged over the long time and provide baselining values which can be used to detect anomaly in application and network performance easily. Also, the historical behavior data of application can be valuable for mid-term and long term network management such as capacity planning.

## 4 Potential usage of application-specific information

In this chapter, we investigate the possible ways to use the monitoring information which can be gathered from application itself, running environment, and sniffing hosts to assist the network management. The usages are classified into five subdomains of network management. Especially, the potential use of application-specific information for traffic engineering is stressed in addition to the usage of the information in the traditional network management. Traffic engineering may include rehomeing, rerouting, load balancing, and congestion control [24]. Traffic engineering refers to the process of selecting the paths chosen by data traffic in order to balance the traffic load of the links on the network. Some degree of load balancing can be obtained by adjusting the metrics associated with network links, but a greater degree of control over network load balancing can be accomplished by explicitly routing traffic flows [32]. Traffic engineering is also an essential input for capacity expansion, network dimensioning, and network planning.

### 4.1 Fault Management

A fault can be defined as a deviation from the set operating goals, system functions, or services [12]. The fault management is usually concerning about the link failure and the malfunction of the routers or switches such that it resorts to the notification events which are generated from the local routers and switches. This kind of notification mechanism reasonably works well in many situations but it does not show how the fault affects the local applications [25]. Also, the notification of the fault is only available from the local network such that it is impossible to know the fault outside of the local network even though it is affecting the applications.

By gathering and analyzing application-specific information, the detection of faults can be easier and more complete. Also, the identification of the application affected by the faults can be made possible. Several application-specific information such as connection failure and overall quality indication fits to this purpose. As mentioned in previous chapter, connection failure means both of the implicit and explicit connection anomaly. The increasing number of reports on connection failures might mean that something went wrong in the paths the local applications are following. By using the traceroute mechanism, the network administrator can find out the links of the problematic path. In addition, by comparing the common links of the multiple paths with connection failures, it would be possible to pinpoint the

location of the fault. There may be several reasons why the users are experiencing connection failure. The simplest reason can be the link down or too much traffic. Also, there may be too much reserved bandwidth for higher service classes such that the flows which belong to low classes experience high packet dropping which cannot be easily detected without help of the application itself.

The basis to measure quality levels of an running application has been considered too subjective because they are supposed to be defined and calculated by each application and client arbitrarily. However, quality level indicators from the application can be effectively used to diagnose the network faults. The true quality-related information can be measured only at a highest protocol layer such as application-layer. That is, the quality level indicator is the real representation of the users' perception on the network performance. It is known that sometimes network administrators try to find out problems in networks only after they receive complains from end users. To overcome the problem, every application can be instrumented (or other monitoring mechanism can be deployed) to report the quality level indication from time to time. The values of the indicator is per-application basis so it is calculated by some rules designed by application developers. Whatever the calculation rule is, the values of quality indicator are always compared against the requirements of the applications and its acceptable status.

## 4.2 Performance Management

Performance management encompasses all the measures required for ensuring that the quality of service conforms to the service level agreements [12]. Performance metrics are measurable indicators that impact application performance. Usually, a technique called baselining or the QoS requirement which is explicitly defined by each application is used to track application performance subsequent to deployment. Performance baselining includes taking measurement over a long enough period of time to encompass the normal and busy times when a specific baseline values are not specified directly by applications. Later by comparing the baselined values or the QoS specification with the current measurement data, the status of application performance can be determined as normal/abnormal. It is known that the optimal way to assess quality of service is to apply measurement methods based on visible quantities at the service interface rather than to use an analysis of the technology supplied by the provider [12]. However, these days most of the applications are not implemented with that appropriate performance reporting capability such that the provider-centric monitoring and

analysis procedure is mainly used to gather quality data for performance management.

There is a consensus that the Application Response Time (ARM) is the most important metric for performance management. Especially in corporate environment, the reasonable response time of business applications is considered the most important metric which guarantee the performance of the company. By measuring the application response time and comparing it with the baselined value, it is possible to detect the potential problems which affect the performance of network, clients, and servers. The monitoring of response times can find out some network problems, even in case that the network administrator may not find any anomaly in the network-level statistics which is gathered from the routers. Therefore, the monitoring of application response time gives supplementary information to the network administrators who have been resorting to the network-level statistics for performance management. In addition to the measurement of ARM, there are many potential metrics of application which may help performance management, especially in terms of admission control and traffic engineering.

It is known that long-lived flows occupies big portion of network traffic. Therefore, in terms of load balancing if we can distribute long-lived flows appropriately or if we can make the long-lived flows to use alternative paths, it is possible to utilize multiple routing paths more efficiently. Fortunately, we may know the expected duration (or termination time) of flows in some cases so that the known information can provide another flexibility for routing decision. One of the main difficulties in deploying dynamic routing is its flapping effect. If we apply dynamic routing to short-lived flows, it would be impossible to avoid the flapping effect. However, if we can selectively apply dynamic routing mechanism only to long-lived traffic, we can get better utilization of the links without experiencing flapping and large overhead to set up a new path. Shaikh [27] showed a possibility to use the dynamic routing of long-lived traffic for the purpose of short-term load balancing. However, since we cannot simply get the expected duration from monitored packets, the work to find out effective threshold becomes a very difficult job. The application hints for the expected duration of a flow can solve the drawback. The potential hints were covered in the previous chapter. In addition, the call admission control near/at the edge routers can utilize the hints such that it can allow more calls which would have been dropped otherwise. As an example, it would be inefficient to block all the new calls when the current connections are about to finish. Likewise, the reporting of starting and ending time of each application can be good hints for routing decision.

The expected bandwidth can be used as a threshold value to report

performance problems to network manager. If the effective bandwidth is less than expected bandwidth, an event message is generated and sent to a network administrator. When a large number of users experience the performance problem, a network manager can set up an additional or alternative path. Also, if the expected bandwidth requirement of an application is over some threshold, the flow of the application can be selectively rerouted to an alternative path. In terms of bandwidth expectation of clients, media/encoding types chosen by clients can reveal the glimpse preference and expectation of users for the current network bandwidth. Based on the expected bandwidth requirement which is gathered from applications, better long-term capacity planning will be made possible.

There has been much research efforts on guaranteeing fair-sharing of limited network resources. TCP congestion control is a good example based on end-to-end mechanism. RED and weighted fair queueing are examples of router-based approaches. The behavior of TCP flows has been well understood and the TCP flows are shown to work cooperatively when network congestion occurs thanks to the TCP congestion avoidance mechanism. However, the behavior of application protocols based on UDP flows are difficult to know because all the congestion control mechanism is expected to be implemented at the application layer. Though router-based approaches are expected to be clean solutions for fair-share, most of them have not been deployed in the Internet because of many reasons, such as the unproved effectiveness of the new mechanisms and the implementation overhead etc. As a simple and practical solution to guarantee fairness in terms of UDP flows, the network administrator may set up two different path, one for adaptive application, protocols and the other for non-adaptive applications. To protect the flows of adaptive applications, rerouting the nonadaptive application flows to different paths can also be taken into account.

The easy discovery of the hot service spots of application servers such as web servers and ftp servers are useful basis to determine the sources of a transient congestion and the expected duration of the congestion. One of the difficulties for the network administrators in handling congestion problems is that they cannot measure how long a congestion will last. For example, when a congestion occurs due to a large number of connection request to a webserver, a network administrator needs an easy way to find out the congestion and to deduce the expected duration of the congestion. By requiring service providers such as web servers and ftp servers to export the interface to show the hot spots of their services, network administrator can make an informed decision to treat the sudden network congestion. The possible treatment would be rerouting, admission control at the border router, and

bandwidth provisioning. By enforcing the new requirements to equip the interface, network administrators can easily find out the list of culprits and its popularity in terms of URLs and hit counts whenever a congestion occurs. If the access pattern of a webserver shows that a congestion occurred due to the popularity of some URL, the network administrator can inspect the contents. In case that the congestion will not be a long-lived one such as interesting spot news, a temporary multiple path such as Label Switching Paths (LSPs) LSP in MPLS-enabled network can be setup for the server. If it turns out that the access pattern is not concentrated on small number of URLs but distributed over the whole webpages, then we can check whether the traffic pattern is repeated over the days. In case that we find out the repeated patterns, we can provision more bandwidth by setting up permanent multiple paths.

### 4.3 Configuration Management

Configuration can be defined as an adaptation process of systems to operating environments and it includes installing new software, expanding old software, attaching devices, or making changes to network topology or to traffic load. Configuration management usually encompasses setting parameters, defining threshold values, setting filters, allocating names to managed objects - loading configuration data to make changes to network topology or to traffic load [12]. Recently these kinds of configuration operations are made under the name of policy-based management. The main goal of the policy-based management is to provide simple mechanisms to map business rules to the behavior of the network infrastructure. A policy-based manager understands the network as a system, and can implement logical rules throughout the network without an administrator needing to interact directly with each element network device within the system [16]. This policy-based management can be implemented as various forms.

Recently application-specific routing, or sometimes so-called L5 routing is being introduced to use application-specific information for routing decision. Application specific routing can be realized by identifying the application protocol and the application-specific information such as URL. Recent measurements on the MCI backbone show that about 95% of the bytes transmitted across the network are carried by TCP and these are 50-70% are HTTP messages among them [6]. Therefore, the attempt to route web traffic based on the identification of web traffic is being accepted as quite a main stream in application-specific routing. The web-routing system considers session level information, such as URL when routing a connection

to a server node. Consequently, it makes it possible to partition the URL space among the server nodes thus improving the performance of the server cluster. In addition to the web routing, it is possible to set up different paths for several important applications flows such that the network administrator can loosely guarantee the quality of service for specific applications [2]. The introduction of MPLS routers makes the application-specific routing as more practical one.

Sometimes the configuration parameters of routers and switches are sensitive to the type of an application and its behavioral characteristics. Some static values are usually assigned to these configuration parameters such that the optimality of the values are not usually guaranteed. The configuration parameters of Random Early Detection (RED) is one of the good examples. The RED mechanism is widely used nowadays for routers and its effectiveness for TCP applications has been proven by many researchers. The default values of the RED parameters such as min and max thresholds in routers have been chosen for general cases which are based on the network-centric measure such as network link utilization or aggregate TCP throughput. The optimal settings of RED parameters are dependent on the types of application protocols which will use the router. Therefore, the application-specific performance metrics came to be considered important to measure the efficiency. However, the evaluations of the RED in real working environment have largely been network-centric measure such as network link utilization or aggregate TCP throughput [6]. In Christiansen's paper, the effect of RED on the response time for an HTTP request is measured as a first attempt to use application-specific performance as a way to tune the RED parameters. In addition to the work to tune the parameters for web traffic statically, the characteristics and performance measure of any application protocol can be used to adjust the RED parameters dynamically. For example, the voice calls can be considered On/Off traffic so the normal value of min threshold may be too pessimistic for the traffic flow. Under the normal min value, the router may cause too much dropping blindly. Thus, it would be a better idea to raise the min threshold value according to the types of application flows which use the router.

Identifying multiple flows which belong to an same application or to a same group of applications will be useful for decision to reroute flows and enforce admission controls. When an application maintains multiple flows, it is usually expected that the performances of each flow are similar. If parallel LSPs in a MPLS-enabled network are configured for load balancing, two flows which belong to a same application might be assigned different labels such that they are delivered via different paths. Sometimes child flows

of an application are correlated such that the application would experience better performance if the delay and throughput of child flows are similar. One good example would be the Web client applications. Therefore, when applications use already established multiple paths, it would be necessary that they can provide hints on their child flows to edge routers such that the routers can aggregate child flows into a single path. As another example of usage of flow aggregation, when admission control is forced, all the flows which belong to a same application instance should be admitted or rejected together. Also, when a dynamic routing is deployed, it would be better to route all the flows of an application instance to a same path. Especially it is true for realtime applications because each routing path might have different delay characteristics and the flows of realtime applications need inter-synchronization. Recently several proposals call for grouping sequences of related packets into flows and sending these packets through fast switching paths, or shortcuts [4]. This fact shows that the importance of correct flow aggregation grows. Also, for long term period, restricting the number of active flows per each application at edge routers may be necessary. In this case, the information on the flow aggregation takes a important role.

Knowing the scheduled traffic beforehand can be helpful for the optimization of network operation and load balancing. Before the scheduled traffic is generated, an alternative path can be setup for the duration. Since the alternative path is setup before the big traffic is poured into a shared path, this can prevent the possible congestion which may affect the other flows otherwise. Also, when it is expected that the scheduled flow will make too much congestion, admission control can be enforced to the scheduled flow. In case that the admission scheduled flow cannot be accepted, the admission manager can provide alternative time slots or the suggested requirement of flow-spec to the application instead.

#### 4.4 Security Management

Security requirements and goals are established on the basis of threat analyses and the values (resources and services) needing protection [12]. One of the important principles to handle security management is that the information related to security hazard should be shared and analyzed instead of simply solving the problem individually. Recently the external hackers' attacks and virus infection are major threats to security in corporate environment. Especially Denial-of-Service (DoS) attack is quite problematic to major web server administrators. For Windows 9x/NT PCs, virus detection and treatment tools such as Symantec's Antivirus are available. Also, de-

tection of hackers' attack and prevention tools such as BlackICE are being installed on end users' PC. However, the detection and treatment is done individually and the information on the virus and the attack are not shared. Several security tools available such as Symantec's AntiVirus and Network ICE's BlackICE to detect virus and hacker's attack. If they are instrumented to log or report the infection history and attack history, the tracking down the source of the problem and the infection/attack path can be more easily determined. Also, if the attack is made from the outside, by using the tracking information, the firewall can be adjusted to block the attack to protect all the clients inside local network.

#### 4.5 Accounting Management

Accounting management includes compiling usage data (resource usage or service usage accounting based on monitoring and metering), defining accountable units, keeping settlement accounts and accounting logs, allocating costs to these accounts, assigning and monitoring quotas, maintaining statistics on usage, and lastly, defining accounting policies and tariffs, which leads to billing and charging [12].

The newly introduced mechanisms such as Diffserv, RSVP, and MPLS makes it possible to provide different Quality of Services to clients such that the importance to apply the right charging scheme grows. The appropriate charging scheme can prevent the misbehavior of network users such as generating too much traffic and can return the invested revenue to network providers. Based on the behavior of applications in terms of adaptability to network status, we can apply different charging schemes for each packet. As a highly probable example, we can apply high-rate service charge if a user consistently use non-adaptive applications. As an alternative charging scheme, it is possible to consider ignoring the retransmitted frames in TCP protocol for charging purpose. The tracking of retransmission of frames is not so easy in sniffing approach because the retransmission can be made by differently fragmented packets and there is no signature to represent the retransmission in TCP packets. Therefore, the client-side/server-side monitoring of retransmission behavior in transport layer is considered better approach. Starting and ending time of applications can be signals to trigger the charging counter and terminate the charging counter. In addition to considering the number of packets(or bytes transferred), the online time of an network application may be used as a basis since the network should at least set aside a minimum bandwidth for the long-lived network application. In WindowsNT and Unix systems the login procedure is mandatory

such that the information on real owner of a flow can be known. In terms of accounting management, the network administrator can get the userid information from each end-host/PC, such that an appropriate charging can be done.

## 5 Conclusion and future work

In this paper we investigated the possible frameworks for monitoring application performance and activities, and introduced several related works. Among the frameworks, the link-level packet sniffing has been most widely used for its nonintrusiveness. Although the mechanism to instrument application is not so popular at this time, it is expected that application instrumentation for supporting management will be quite common in the near future because of the standardization efforts of many groups such as IETF and Tivoli. However, the sniffing approach is expected to be used in several areas. Especially, the application-specific routing for short-lived flows seems to be handled only by the information which is revealed by sniffing mechanisms or signalling methods. On the contrary, the increasing deployment of secure communication protocols such as IPsec will necessitate peer-based transport/application layer monitoring tools.

Until recently, most of the research efforts on application management have been concentrated on the design of the effective monitoring and analysis methods and the standardization of information models. Though many aspects of application management such as software distribution/update, licensing management, and application performance monitoring have been focused on, the modest integration of application management with the recently evolving technologies has not been well discussed. In viewpoint of network providers, the semantics of application have the power to help to enhance the network operation. Recently dynamic routing, admission control, and traffic engineering are being investigated by several researchers and they are considered promising for effective use of limited network resources. We identified the importance of application-specific information for these new areas which cannot be supported by the simple network-level monitoring information.

In the later part of this paper, we suggested several potential information that might be interesting and helpful for network administrators to adopt admission control, dynamic routing, and traffic engineering. Also, we suggested several possible way to use these information. As part of our on going work, we are investigating the effective use of application information for traffic engineering. We're interested in developing a novel framework to drive MPLS rerouting decision based on the application semantics. To realize the framework, it is necessary that appropriate requirement of monitoring mechanisms are identified and the corresponding monitoring system should be built. Also, we need a simulation model to utilize the application information.

## References

- [1] Jonathan Angel. Application Performance Management Software. *Network Magazine Online*, May 1999.
- [2] G. Apostolopoulos, D. Aubespin, V. Peris, P. Pradhan, and D. Saha. Design, Implementation and Performance of a Content-Based Switch. In *Proc. of IEEE Infocom*, March 2000.
- [3] Paul Barford and Mark Crovella. Critical Path Analysis of TCP Transactions. In *Proc. of ACM SIGCOMM*, Sweden, August 2000.
- [4] R. Caeres, T. Johnson, and J. Rexford et al. Measurement and Analysis of IP Network Usage and Behavior. *IEEE Communications Magazine*, 38(5):144–151, May 1999.
- [5] C.Darst and S.Ramanathan. Measurement and Management of Internet Services. In *Proc. of the Sixth IFIP/IEEE International Symposium on Integrated Network Management*, May 1999.
- [6] Mikkel Christiansen, Kevin Jeffay, David Ott, and F. Donelson Smith. Tuning RED for Web Traffic. In *Proc. of ACM SIGCOMM*, Stockholm, Sweden, August/September 2000.
- [7] Luca Deri, Finsiel S.p.A., and Stefano Suin. Effective Traffic Measurement using NTOP. *IEEE Communications Magazine*, 38(5):138–143, May 1999.
- [8] Lenny Bonsall et al. The Via Architecture: A fresh perspective on Application Performance Management. Whitepaper, Lucent Technologies Inc., January 1998.
- [9] Anja Feldmann, Albert Greenberg, Carsten Lund, Nick Reingold, Jennifer Rexford, and Fred True. Continuous online extraction of HTTP traces from packet traces. In *W3C Web Characterization Group Wrokshop*, November 1998.
- [10] Anja Feldmann, Jennifer Rexford, and Ramon Caceres. Efficient policies for carrying Web traffic over flow-switched networks. *IEEE/ACM Transactions*, 6(6):673–685, December 1998.
- [11] Rainer Hauck and Helmut Reiser. Monitoring of Service Level Agreements with flexible and extensible agents. In *Workshop of the OpenView University Association(OVUA '99)*, June 1999.

- [12] Heinz-Gerd Hegering, Sebastian Abeck, and Bernhard Neumair. *Integrated Management of Networked Systems: Concepts, Architectures, and Their Operational Application*. Morgan Kaufmann Publishers, San Francisco, California, 1998.
- [13] H.Hazewinkel, C.W. Kalbfleisch, and T. Braunschweig. Definitions of managed Objects for WWW Services. Internet Draft, Internet Engineering Task Force, February 1999. Expires August 1999.
- [14] Apptitude Inc. Application responsiveness through traffic flow analysis. Whitepaper, Apptitude Inc., October 1999.
- [15] Apptitude Inc. MeterFlow Network DecisionDataEngine. Whitepaper, Apptitude Inc., January 2000.
- [16] Cabletron Systems Inc. Application-Based Networking: Implementing utility-like, application-aware network infrastructures. Whitepaper, Cabletron Systems Inc., February 1999.
- [17] Compuware Inc. How to analyze performance with EcoScope. Whitepaper, Compuware Inc., September 1999.
- [18] Netscout Inc. Application Flow Management Brochure. Whitepaper, Netscout Inc, February 1999.
- [19] Optimal Inc. Optimal Smart Agents. Whitepaper, Optimal Inc., November 1999.
- [20] C. Kalbfleisch, C.Krupczak, R.Presuhn, and J.Saperia. Application Management MIB. Internet Draft, Internet Engineering Task Force, May 1999. Work in progress.
- [21] Art Mena and John Heidemann. An empirical study of real audio traffic. In *Proc. of IEEE Infocom*, March 2000.
- [22] Tom Pagan. Enterprise applications and performance engineering. Whitepaper, Lucent Technologies Inc., February 1999.
- [23] Peter Parnes, Kare Synnese, and Dick Schefstrom. Real-Time control and management of distributed applications using IP-multicast. In *Integrated Management*, May 1999.
- [24] Ammar Rayes and Karen Sage. Integrated management architecture for IP-based networks. *IEEE Communications Magazine*, pages 48–53, April 2000.

- [25] Anoop Reddy, Deborah Estrin, and Ramesh Govindan. Fault isolation in multicast trees. In *Proc. of ACM SIGCOMM*, August 2000.
- [26] A. Schade, P.Trommler, and M.Kaiserswerth. *Object Instrumentation for Distributed Applications Management*. Chapman & Hall, London, 1996.
- [27] Anees Shaikh, Jennifer Rexford, and Kang G. Shin. Load-sensitive routing of long-lived IP flows. In *Proc. of ACM SIGCOMM*, August 1999.
- [28] Mark Stemm. *An network measurement architecture for adaptive applications*. Ph.D Thesis, University of California, Berkeley, 1999.
- [29] Rick Sturm and Winston bumpus. *Foundations of Application Management*. John Wiley & sons, Inc., New York, NY, 1999.
- [30] Lucent Technologies. Characterizing End-to-End Performance. Whitepaper, Lucent Technologies Inc., 1999.
- [31] Elizabeth Suet H. Tse-Au and Patricia A. Morreale. End-to-End QoS Measurement: analytic methodology of application response time vs. tunable latency in IP networks. In *IEEE/IFIP Network Operations and Management Symposium*, April 2000.
- [32] A. Viswanathan, N. Feldman, Z. Wang, and R. Callon. Evolution of Multiprotocol Label Switching. *IEEE Communications Magazine*, pages 165–173, May 1998.
- [33] Mark W.Johnson and Steve Smead. Beyond ARM 2.0-API extensions that enable pervasive Service Level instrumentation. In *Computer Measurement Group Conference*, December 1998.