Internet Application Traffic Monitoring and Analysis

2004

김 민 섭

박 사 학 위 논 문

Internet Application Traffic Monitoring and Analysis

김 명 섭 (金 命 燮)

전자 컴퓨터 공학과 (네트워크 전공)

포항공과대학교 대학원

2004

인터넷 응용 트래픽 수집 및 분석

Internet Application Traffic Monitoring and Analysis

Internet Application Traffic Monitoring and Analysis

by

Myung-Sup Kim
Division of Electrical and Computer Engineering
(Computer Science and Engineering)
Pohang University of Science and Technology

A thesis submitted to the faculty of Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering.

Pohang, Korea
June 1, 2004

Approved by

Major Advisor: James Won-Ki Hong

인터넷 응용 트래픽 수집 및 분석

김 명 섭

위 논문은 포항공과대학교 대학원 박사 학위논문으로 학위논문 심사위원회를 통과하였음을 인정합니다.

2004년 6월 1일

학위논문심사위원회 위원장 홍 원 기 (인)

위원 김 치 하 (인)

위원 김 종 (인)

위원 서 영 주 (인)

위원 홍 충 선 (인)

DECE 20003197 김명섭, Myung-Sup Kim, Internet Application Traffic Monitoring and Analysis, 인터넷 응용 트래픽 수집 및 분석, Division of Electrical and Computer Engineering (Computer Science and Engineering), 2004, 114P, Advisor: James Won-Ki Hong, Text in English.

ABSTRACT

Two critical problems exist in traffic monitoring and analysis of today's Internet traffic compared to the past network environment. The first is how to capture and handle the huge amount of traffic data in a real-time manner generated from high-speed network links, such as 2.5 Gbps and higher. The second is how to analyze diverse and complex types of traffic generated by many different types of network-based applications, such as streaming media, peer-to-peer, and game applications.

Considering the first problem, this thesis presents the design of a network traffic monitoring and analysis system, called NG-MON, for high-speed networks such as 10 Gbps and above. Using distributed, pipelining and parallel processing techniques, we have designed a flexible and scalable monitoring and analysis system, which can run on off-the-shelf, cost-effective computers. The monitoring and analysis task in NG-MON is divided into five phases; packet capture, flow generation, flow store, traffic analysis, and presentation. Each phase can be executed on separate computer system and cooperates with adjacent phases using pipeline processing. Each phase can be composed of a cluster of computers wherever the system load of the phase is higher than the performance of a single computer system. In addition, we have defined efficient communication methods and message formats between phases.

Another critical problem with recent Internet traffic monitoring and analysis concerns the large number of newly emerging network-based applications which possess more complicated communication structures and traffic patterns than

traditional applications. The amount of traffic generated by these newly emerging applications is reported to be well over a half of total traffic. Characterizing these new types of traffic is a challenging area of current traffic monitoring and analysis. Moreover, the dynamic use of port numbers, the use of multiple sessions, and other features of these applications complicate the characterization of Internet traffic. Application-level traffic identification is a preliminary but essential step toward traffic characterization, which this thesis mainly addresses. Traditional traffic identification methods based on well-known port numbers is not appropriate for the identification of the peer-to-peer, streaming, and other new types of applications.

This thesis proposes a new method to identify current Internet traffic. We categorized current network-based applications into several classes according to their traffic patterns. Using this categorization, we developed a flow grouping method that determines the application name of traffic flows. We have incorporated our method into NG-MON to analyze Internet traffic between our enterprise network and the Internet, and characterized all the traffic according to their application types. This thesis presents the problems with the existing application-level traffic identification methods and difficulties in the characterization of recent Internet traffic. The methodologies to solve these problems are given. Moreover, this thesis describes application-level traffic characteristics of recent Internet traffic generated from POSTECH campus network. Our experience with developing application traffic identification method and analyzing Internet traffic from POSTECH provides guidelines for in-depth understanding of Internet traffic.

Contents

List of Figures

List of Tables

1. Introduction

This section gives an overview of IP traffic monitoring and analysis of current network environment. The problems in current IP traffic monitoring and analysis are listed and the approaches to solve these problems are mentioned in this thesis.

1.1 Backgrounds

Today, as the dependency of the individual users and enterprises on the Internet is growing, the Internet service providers (ISPs) are trying to provide stable, reliable, and better network services in many ways. The main concern of ISPs has been to improve service quantity, such as the increase of link speed and the number of available links. Consequently, the multi-gigabit networks are becoming common within or between ISP networks. The bandwidth of ISP's backbone networks is evolving from OC-48 (2.5Gbps) to OC-192 (10Gbps) to support rapidly rising Internet traffic [1, 2]. In addition, the importance of the network service quality, such as availability and reliability, is growing. They give a lot of interest in the Service Level Agreement (SLA) [3, 4, 5], Quality of Service (QoS) [6, 7, 8], Customer Relationship Management (CRM) [9], and so on.

The network traffic monitoring and analysis provides basic and critical information for improving network services, in terms of quality and quantity. In addition to the fundamental and traditional purpose of traffic monitoring, such as network planning, network problem detection, and network usage reporting, the traffic monitoring and analysis is required in many new areas to improve network service qualities, as illustrated in Figure 1. It is necessary to detect the abnormal traffic, such as traffic generated by Internet worm (ex. SQL Slammer, MS Blaster, Nimda) [10, 11] and DoS/DDoS attack (ex. Smurf, Flooding, Fraggle) [12, 13, 14]. The Internet billing system, shifting from flat-rate model to usage-based billing model [15], critically depends on the result of the network traffic monitoring and

analysis.

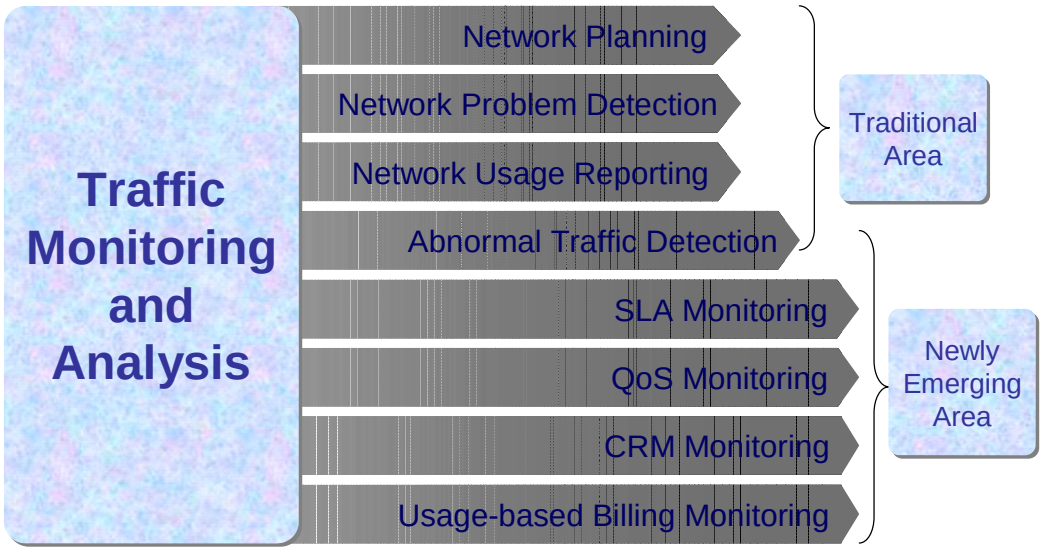


Figure 1. Areas of Traffic Monitoring and Analysis

In addition, the SLA monitoring and QoS monitoring are highly dependent on the IP traffic monitoring and analysis, which are essential parts of the SLA enabled and QoS enabled network services. Moreover, the CRM requires the traffic monitoring and analysis to comprehend the customers' behaviors of network usage. Beyond the areas which mentioned in this section, the network monitoring and analysis is required in many other areas.

1.2 Problem Statements

To come up with the evolution of the Internet in terms of underlying technologies and user services, the network traffic monitoring and analysis techniques should be improved in terms of system architecture and analysis methodology. Two critical problems exist in today's Internet traffic monitoring and analysis. The first is how to handle an increased and massive amount of traffic data generated from high-speed network links, such as 2.5 Gbps and higher, in

real-time manner [1, 2, 16, 17, 18]. The other is how to analyze sophisticated traffic data generated from various newly emerging network-based applications, such as streaming media, peer-to-peer (P2P), and game applications [19, 20, 21].

Regarding the first issue, some excellent research results are reported. To improve capturing performance, research institutions and companies developed specialized network cards for traffic monitoring (for example, DAG card [22, 80]). They also introduced various flow formats, such as Cisco NetFlow [23] and InMon sFlow [24]. The majority of commercial routers are now fully equipped with the functionality to export flow information in widely known formats (i.e. Cisco NetFlow). The IETF working group IPFIX (IP Flow Information Export) [53] is launched to define the notion of standard IP flow in 2001. In the aspect of real-time traffic monitoring architecture, RTFM [25] influenced the development of many traffic monitoring and analysis systems [1, 2, 26, 27]. However, most of the previous and current research focused on a single dedicated part of monitoring system rather than the whole system architecture for high-speed network link [22, 23, 24, 53]. Otherwise, they provided general system architecture [25], which should be redesigned to handle high-volume of traffic data. Therefore, we have to develop new system architecture suitable for the high-speed network links.

The other issue is caused by numerous numbers of network-based applications. The types and patterns of current network traffic are more complex than the past. In the past network environment, most Internet traffic was occupied by HTTP, FTP, TELNET, SMTP, and NNTP. Today, the proportion of these well-known ports based traffic is decreasing. Instead, P2P, streaming media, and game traffic are increasing. Internet2 administrators report that about 16% of the traffic carried by their network is P2P traffic while a further 54% of unidentified traffic is most likely belonging to P2P applications [28]. The amount of P2P application traffic in many ISPs is reported to be over 50% of total traffic [29, 30, 31]. The difficulty with traffic analysis is that this newly emerging traffic is not as straightforward as the well-known port based traffic. Therefore, we need a new technique to analyze P2P application traffic.

The features of current Internet-based applications are as follows:

- A large number of different Internet-based applications are developed and widely used. P2P, streaming media, and game applications are the best examples of these newly emerging Internet-base applications. Moreover, the number of these applications will increase continuously in future. It is difficult to identify the origin application of captured individual packets.
- Many new applications use proprietary application-layer protocols. These proprietary protocols are complex and difficult to understand in terms of format and operation. Some applications, such as KaZaA [32, 33] and VPN, encrypt the data flowing into the network to hide their behavior and protect their systems from potential security attacks.
- The port numbers used by these applications are irregular. Most use ephemeral port numbers as default application port, which are greater than 1024. The proportion of TCP traffic using ephemeral ports is more than 40% of total TCP traffic at a large ISP backbone network as of January 2003 [30]. In our investigation, the number of TCP sessions using ephemeral ports as a server port is more than 50% of total TCP sessions in the POSTECH campus network.
- The default port numbers for many applications are not registered at the IANA port list [34]. The developers of applications for regional users usually do not register their port numbers to IANA. It is impossible to regulate this type of complex and unfettered use of port numbers, because no authorized organization can regulate the use of port numbers in the current Internet environment.
- The topological application architecture is shifting from the traditional client/server model to a P2P communication model. Most P2P applications can directly transfer data to other peers. The overlay networks constructed by these applications are complex. A single client program simultaneously communicate with multiple peers, while in traditional client/server program a client program usually communicates with a single server at a time. For

example, a network game application communicates with a central server to exchange information and at the same time transfer data to the other peers playing together.

- Most newly emerging applications use multiple sessions to communicate with each other. For example, streaming media applications such as Microsoft windows media client/server [36] and QuickTime client/server [35, 49] establish more than two sessions to transfer control data and multimedia data like FTP application. Sometimes they use TCP and UDP simultaneously. Most P2P applications also use multiple sessions.
- The traditional Internet-based applications use statically predefined port numbers to communicate with each other. However, the current P2P and streaming media applications use dynamically determined port numbers by the negotiation between two peers. In case of streaming media traffic using RTSP [35] and MMS [36], the port number and protocol for the delivery of multimedia data is decided by the negotiation between client and server. The port number for file transfer in MSN [50] instant messaging application is also determined dynamically. The P2P file sharing applications use dynamically generated port numbers to evade detection and control [43, 44].

All of the above features of current applications make it difficult to analyze Internet traffic at the application level. The application-level traffic identification is a fundamental and significant step towards the proper analysis of application-layer traffic. The application-level traffic characterization is another challenging area of traffic analysis. Thus, this paper focuses on application-level traffic identification and characterization. We concentrated on the following two crucial questions about the second problems in current traffic monitoring and analysis. How can we identify individual applications from IP traffic? What are the characteristics of the current IP traffic at the application level?

1.3 Research Approaches

Considering the two critical problems of current traffic monitoring and analysis on IP network, which is mentioned in previous sections, we have developed noble and efficient solutions. In this section, we present the research approaches taken to solve these problems.

For the first problem, we have developed a network traffic monitoring and analysis system, called NG-MON, suitable for high-speed networks such as 10 Gbps and above. Using distributed, pipelining and parallel processing techniques, we have designed a flexible and scalable monitoring and analysis system, which can run on off-the-shelf, cost-effective computers. The monitoring and analysis task in NG-MON is divided into five phases; *packet capture*, *flow generation*, *flow store*, *traffic analysis*, and *presentation*. Each phase can be executed on separate computer systems and cooperates with adjacent phases using pipeline processing. Each phase can be composed of a cluster of computers wherever the system load of the phase is higher than the performance of a single computer system. In addition, we have defined efficient communication methods and message formats between phases.

Regarding the second problem which we mentioned before, the traditional traffic identification methods, based on pre-defined port numbers and IANA port list, cannot be used to analyze the P2P, streaming, and other new applications. We provide a new method to identify the current Internet traffic. First, we categorize Internet traffic from the traffic analysis point of view. We categorized most of the current network-based applications into several classes according to the traffic pattern generated by these applications. Using this categorization, we developed a new method called a flow-grouping method, which determines the application name of individual packets. The main idea of this flow grouping method comes from the fact that there exist some dependencies among flows belonging to certain applications.

This flow grouping method is composed of four crucial steps. The first step is

to make the Application Port Table (APT) by an exhaustive off-line search of existing Internet-based applications. The second step is the Important Port Number Selections (ISP) from each flow record, which determines the important port among source and destination port numbers in each flow record in the decision of individual application. The third step is the Flow Relationship Map (FRM) which aggregate flows according to the dependency among them. The final step is the decision of application name for each flow. To validate the proposed algorithm, we have designed and implemented an Internet traffic analysis system and presented the characteristic of current Internet traffic by deploying this system in the Internet junction of our campus network. We have applied our method to analyze Internet traffic, generated by our campus network, and characterized them according to their application type. Our research result can give basic and important information to characterize and control current and future Internet traffic.

1.4 Organization of Thesis

The remainder of this thesis is organized as follows. Chapter 2 describes some previous approaches on the current traffic monitoring and analysis. Especially, we present several research results and industrial initiatives for traffic monitoring on high-speed networks. We also summarize some related work on the identification and characterization of Internet traffic and their good and bad features. In Chapter 3, we present the proposed system architecture, called NG-MON, which is a passive traffic monitoring and analysis system suitable for high-speed network links. Chapter 4 describes our proposed traffic identification method. The design and implementation issues are described in Chapter 5. In Chapter 6, we describe the validation of the proposed method and the Internet traffic characteristics found using our method. Finally, Chapter 7 summarizes our work and contributions. Possible future research directions for further extension of this research are also discussed.

2. Related Work

In this Chapter, we introduce related research work on traffic monitoring and analysis in the current network environment. First, we present several research results and industrial initiatives for traffic monitoring on high-speed network links. Next, we mention some previous work on application-level traffic identification. We also introduce some research results about recent Internet traffic characteristics.

2.1 Traffic Monitoring on High-Speed Network Links

As the link speed of ISP and enterprise network is growing higher than Gbps, the performance of existing traffic monitoring system became problematic. Using a single off-the-shelf general-purpose computer system, it has become impossible to handle the huge amount of packets generated from high-speed network links. In this section, we summarize the previous research results to overcome the difficulties in the monitoring of high-speed network links.

2.1.1 The Drawbacks of Exiting Traffic Monitoring Systems

Ntop [75] is a well-known on-line traffic monitoring system in passive mode that provides in-depth protocol and throughput analysis, and reports the analysis results using a web-based graphical user interface. Ntop is developed as an open source by Luca Deri, et al. However, Ntop is not suitable for monitoring high-speed network links, such as enterprise back-bone network. It cannot scale beyond monitoring a fully-utilized 100 Mbps link, because all the monitoring functions are implemented in a single system and use standard libpcap library [41] to capture packets. The standard libpcap module over a general-purpose NIC cannot capture more than 50,000 packets per second.

FlowScan [52], another open-source traffic monitoring system, analyzes and reports on NetFlow format data [23] collected using CAIDA's cflowd flow tool [76]. FlowScan examines flow data and maintains counters reflecting what was found. Counter values are stored using RRDtool [77] that is a database system for time-series data. Finally, FlowScan uses visualization capabilities of RRDtool, arts++ [78], and other front-ends to report the processed flow data. However it analyzes only NetFlow [23] data format and is not suitable for monitoring high-speed networks either, because its performance is absolutely dependent on the performance of connected network equipments. It is reported that the Cisco NetFlow is often implemented very badly on modern network equipment. For instance, popular network appliances produced by companies such as Juniper Networks and Extreme Networks that can switch millions of packet/second, are featured with NetFlow that can handle less than 10,000 packet/sec. For this reason, network administrators cannot base their standard NetFlow-based monitoring tools on the equipment they use for operating the network unless they want to experience severe packet loss [17]. Ntop and FlowScan may be appropriate for analyzing relatively low speed networks such as an underutilized backbone link or a LAN segment.

The RTFM (Real-time Traffic Measurement) architecture was proposed by the homonymous IETF Working Group and it is a logical monitoring structure for different types of networks. It is composed by four entities that communicate by means of a suitable protocol which is not explicitly specified in the definition: Meter, Meter Reader, Manager, and Analysis Application.

- The meter is the component which has the task of accounting packets according to some attributes such as their source and destination addresses. For the traffic measurement, the meter follows a set of rules which specify the attributes of the traffic flows to be observed; a packet is counted if all its attributes values match.
- The manager is an application that configures and controls the activity of one or more meters and meter readers; it works according to the requirements of

the applications that make use of accounting data.

- The meter reader collects and transfers the registered data in a reliable way between the other three applications.
- The analysis application is an application that handles accounting data for different purposes such as the determination of a network performance or its management.

NetraMet [26, 79] is an open-source implementation of the RTFM architecture [25] for network traffic flow measurement, which is developed and supported by Nevil Brownlee at the University of Auckland. In the reference implementation of RTFM architecture, NeTraMet is the meter and NeMaC is both the manager and meter reader; these two elements exchange information through the SNMP protocol. A Rule Set, created by the user, contains the specifications of the tests on packet attributes, in particular IP address and TCP/UDP port number of source and destination. However, original NetraMet collects raw packets form network links without consideration of the link speed. Moreover, the RTFM architecture does not designed with the reflection of high-speed network links. Therefore, many traffic monitoring systems influenced by the RTFM architecture should be changed to get along with the high-speed network links.

2.1.2 Researches to handle high-speed networks links

To cope with the huge amount of traffic data form high-speed network links, much research has been performed. As a result, many excellent outcomes are reported.

First, specialized network interface cards with NPU (network process unit) have been developed. The DAG card from Endace Inc. [22] is a good example of this category. The performance degradation of general-purpose computer equipment in traffic monitoring is mainly caused by the limitation of NIC performance and PCI bus speed [73]. The DAG NIC transfers data at the full speed of the network into the memory of the host PC, with few packet loss

guaranteed in even worst-case conditions, by performing lots of CPU-consuming jobs in the embedded NPUs. Recently, Endace Inc. has developed DAG6.1 NIC supporting one direction of 10Gbps Ethernet Link, equipped in a system with PCI-X 64-bit 133MHz Interface and Linux 2.4.x OS.

Many recent research projects and traffic monitoring systems are shifting from normal NIC to these kinds of specialized NICs. For example, various versions of NetraMet have been developed to support various kinds of traffic sources (Cisco NetFlow and CoralReef data source) and NIC cards (DAG and Apptel ATM interface cards) to support high-speed network links. The sprint IPMON project uses Endace DAG cards to collect traffic data form Sprint backbone network [80]. They perform an off-line analysis of the traffic data collected from multiple sites, after putting them together.

CoralReef [27] is a comprehensive software suite developed by CAIDA to collect and analyze data from passive Internet traffic monitors, in real time or from trace files. Real-time monitoring can support various traffic gathering devices or software to handle large range of network link speed. They support general system network interfaces (via libpcap), FreeBSD drivers for Apptel POINT (OC12 and OC3 ATM) and FORE ATM (OC3 ATM) cards, and support for Linux drivers for WAND DAG (OC3 and OC12, POS and ATM) cards. The package also includes programming APIs for C and perl, and applications for capture, analysis, and web report generation. CoralReef is reported to be able to monitor up to OC-48 network links. With respect to the load distribution, only CoralReef suggests a separation of flow generation and traffic analysis, but without consideration of clustering of processing systems in each stage.

Second, many studies investigate the performance limitations of existing software components for traffic capturing and tried to improve their performance. For example, Loris Degioanni, et al. [72] analyzed the widely used library for traffic analysis (Winpcap), highlights its bottlenecks. In addition, they proposed their own solution that almost doubles the overall speed, thus enabling the deployment of software-based tool on high-speed networks.

Third, Luca Deri [17] proposed a new traffic monitoring system architecture to successfully monitor high-speed networks using commodity hardware and open source software (Ntop), which is a two-layer architecture composed of a preprocessor and the existing monitoring application (Ntop). This solution has the advantage that the Ntop code does not need to be changed nor traffic analysis facilities be removed in order to increase the application performance. A preprocessor is responsible for reducing the amount of work that the monitoring application has to carry out. The simplest preprocessor is a traffic sampler that discards packets according to a specific rate similar to what sFlow-based probes [24] do.

Another traffic monitoring system architecture for high-speed network links are proposed by Ahang Shiyong, et al. [73]. The proposed system includes the Flow Distribute module which generates flows from captured traffic data and then distributes the flow data into multiple traffic analysis handler. Because the Flow Distribute module can distribute the traffic data into as many as traffic handler, the proposed architecture supports lossless packet capture in the high-speed network. However, they proposed only system architecture with no system implementation.

Fourth, the support from network devices for traffic monitoring is very interesting. SNMP agent is a de-facto standard providing the in/out traffic count from various protocol layer, which is embedded into most network equipments. To support more comprehensive and detailed analysis, many network devices have a function to export flow data. The Cisco NetFlow [23] and InMon sFlow [24] are good examples of these flow data format. Among them, the Cisco NetFlow exporting module is embedded in most backbone switches or routers. The embedded flow exporting module makes the architecture of a traffic monitoring system simple, and reduces the implementation overhead of a traffic monitoring system. They also provide sampling and filtering rules in the generation of flow data. This reduces the performance degradation in highly utilized network links and provides user-selected traffic information. The IETF working group IPFIX

[53] is trying to define standard notion of flow data on the basis of Cisco NetFlow V9.

Fifth, the sampling and filtering scheme is another way to cope with high-speed network link using low-capacity traffic monitoring systems. The InMon sFlow is working based on sampling mechanism. The sFlow monitoring system consists of an sFlow Agent and a central sFlow Collector. The architecture and sampling techniques used in the sFlow monitoring system were designed for providing continuous site-wide (and enterprise-wide) traffic monitoring of high speed switched and routed networks. The sFlow Agent uses sampling technology to capture traffic statistics from the device it is monitoring. sFlow datagrams are used to immediately forward the sampled traffic statistics to an sFlow Collector for analysis. The IETF working group psamp [59] is chartered to define a standard set of capabilities for network elements to sample subsets of packets by statistical and other methods, which is launched in 2003. Many sampling and filtering schemes were proposed form the psamp WG. C. Estan, et al. [74] also proposed a sampling mechanism to select heavy flows from high-speed network links using small amount of memory and low processing power.

All above efforts in various fields help to handle effectively the high volume of traffic data generated by high-speed network links. However, we have lot of challenging areas in the monitoring of high-speed network links.

2.2 Application-Level Traffic Identification

It is highly suspected that the peer-to-peer applications are accountable for a large portion of Internet traffic in the ISP networks [29, 30]. (What distinguishes the peer-to-peer applications from the traditional ones is that they acquire a combination of ephemeral and dynamically allocated ports.) For the confirmation of this assertion, we collected Internet traffic generated by our campus network over a week using NG-MON [1, 2]. The NG-MON aggregates individual packets into flows in the flow generation module. In the NG-MON system, the flow is

defined as an aggregation of packets traveling from one host to another with the same 5-tuple packet header values: source IP address, destination IP address, source port number, destination port number, and transport protocol number. The definition of flow in NG-MON is similar to the CISCO NetFlow [23] version 5 format. From the traffic trace collected, we discarded non-IP flows and flows with spoofed IPs. The portion of traffic in this category was 0.5% in bytes, 3.3% in packets, and 28% in flows of total traffic, respectively. Among the total captured 14TBytes of traffic data, the ratio of traffic data that use port numbers greater than 1024 was 77.9% in bytes, 73.8% in packets, and 89.5% in flows of total traffic, respectively. Considering TCP traffic, which occupies 97% of total traffic, the proportion of traffic with port numbers above 1024 was 78.0% of total traffic in bytes. Recent studies [29, 30, 31, 32.] imply that most of this traffic is peer-to-peer file sharing traffic. To precisely identify and characterize this large amount of traffic is indispensable in the traffic monitoring and analysis of current network.

We consider two steps to determine the origin application name of individual packets. The first step is to decide the important port number between source port and destination port. Most traditional Internet-based applications, such as WWW, FTP, Telnet, etc. use a well-known port that is below 1024. This simplifies determining the important port of the packets with a port number below 1024. However, most newly emerging Internet-based applications use ports greater than 1024 as a default port for communication. It is not straightforward to decide the important port of the packets that have both ports above 1024 and belong to these new applications. The second step is the actual decision of the application name from the pool of selected important port numbers. In this step, the dynamically negotiated port number of newly emerging applications and the large number of Internet-based applications are two critical problems. The following is some related work on the identification of application level traffic.

2.2.1 Traditional Application-level Traffic Identification Method

The traditional method is based on the well-known ports registered to the IANA port list [34]. For example, the Web traffic caused by web client/server applications uses port numbers 80, 8080, or 443. In the traditional method, the important port is the port less than 1024 or the port that appears in the IANA port list. By this well-known port, we can easily determine the corresponding application name. Just a few years ago, this technique was sufficient enough to identify most Internet traffic. However, we cannot rely on this method any more because of new features in recent Internet traffic. For example, this method cannot detect the dynamic port numbers generated by streaming media applications, such as Microsoft Window Media Server/Player [36]. Also, the traditional method cannot distinguish the traffic between two different applications simultaneously using the same port numbers. If the target port number is not present in the IANA port number list, then this method is simply not applicable.

2.2.2 Payload Examination based Method

To detect the dynamically determined ports of streaming media applications and P2P applications, the payload examination method is one possibility. Mmdump [37] and SM-MON [20, 21] used this method to differentiate streaming media traffic from other Internet traffic. In the streaming media service, two types of session are established between the client and server: a control session and a data session, as illustrated in Figure 2.

The control session is used to transfer control data for streaming media service, such as the data session setup, media data profile, and start/stop/pause of media data transfer. RTSP [35] and MMS [36] are the best examples of control session protocols. These control session protocols usually use TCP with well-known and fixed ports, like 554 for RTSP and 1755 for MMS, respectively. The data session is responsible for the delivery of multimedia data from media server to media player. For the data session protocol, many different protocols are

introduced, including RDT (Real Networks Data Transfer) [38], RTP (Real time Transfer Protocol) [39], and MMST/MMSU (MMS over TCP/UDP) [40]. Both TCP and UDP can be used as data session protocols. More than two different data sessions can be established to deliver audio data and video data, respectively. The port number of the data session is determined dynamically by the negotiation between client and server using the pre-established control session. In addition, it is also useful to detect a data session of passive FTP traffic which shares a similar fashion with streaming media.

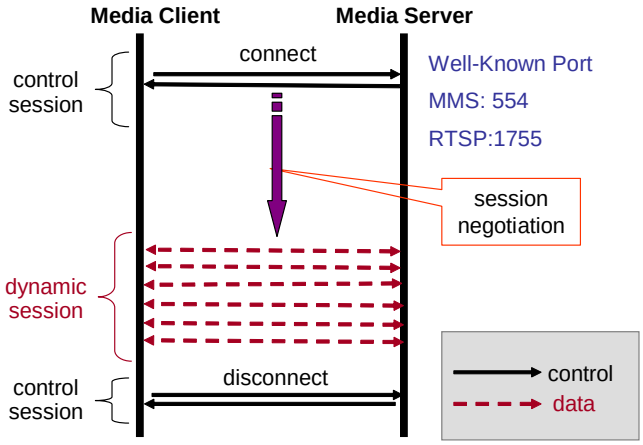


Figure 2. Data Communication in Streaming Service

This method inspects all control session data to discover the dynamic port numbers for the data session. Using the port numbers and IP address information gathered from control session data, it is possible to determine streaming media traffic from a considering amount of unknown traffic. This method provides high accuracy for the identification of streaming traffic. However, it causes additional system overhead for the inspection of payload data because the load of a capture system is proportional to the number of captured packets and the snapshot size copied into the user-level. When the port number of control session is changed, this method is no longer effective. When the control session data is encrypted as with the KaZaA P2P application, payload inspection is also not possible. To

identity all Internet traffic using this method, we should know every single detail of all application-layer protocols; nevertheless, we believe this is impossible.

2.2.3 Signature Mapping based Method

The port number based traffic identification method cannot absolutely confirm that a packet using source port number 80 is WWW traffic, because any other application can also use the same port number 80 to transfer data among them. To increase identification accuracy, the signature mapping based method [15] was introduced. In this method, a portion of payload data that is static, unique, and distinguishable from other packets is examined for all applications, regardless of the protocol they are using. These are decided as signatures of those applications. By comparing every packet payload with pre-determined signatures, this method can identify application traffic more accurately than the traditional method.

However, it requires a large amount of offline work to discover the signatures of individual applications. It may be easy to determine the signature of the applications using standard and open protocols such as HTTP and FTP. Today, the number of network-based applications is large and increasing rapidly, and many use their own proprietary protocols. It is more difficult to examine the signature of these proprietary applications. This method causes greater system overhead in the process of packet capture and payload comparison. To apply this method to the on-line and real-time traffic analysis system, it is necessary to refine and fine-tune the basic algorithm for the target environment.

2.2.4 Methods used for P2P traffic identification.

A current outstanding tendency of Internet traffic is to shift from web traffic to P2P traffic. Many studies focus on the characterization of P2P traffic [29, 30, 31, 32, 33]. The first step of all these studies is to determine the P2P application traffic from the entire range of network traffic. The technique used in this first

step was the traditional method using corresponding default port numbers for P2P applications; such as TCP port 6346/6347 (Gnutella), 1214 (FastTrack), and 411/412 (DirectConnect), respectively [29]. A research on Internet content delivery systems, including WWW, P2P, and CDN, also distinguished traffic type by default port numbers of each application and server IPs providing corresponding services [31]. However, new versions of the KaZaA system do not use default port number (1214) any longer. They use dynamically assigned port number to transfer data between peers. Furthermore, the port numbers used for file transfer between peers in MSN messenger application changed from a fixed port (6981) to a dynamic port. Recent research [30] finally gave up identifying the traffic according to application. Instead, they proposed a new traffic type, called TCP-Big, which is the aggregation of unknown flows that transmit more than 100KB in less than 30 minutes. They showed that the TCP-Big traffic had almost the same properties as P2P traffic. The identification of P2P traffic is the most challenging area in the traffic monitoring and analysis of current Internet traffic.

2.3 Traffic characterization of recent Internet Traffic

In this section, we present some related studies on traffic characterization. Many recent studies addressed to the IP traffic characterization. They collected and analyzed the IP traffic from an enterprise network [63] or a large ISP backbone network [62]. They focused on traffic characteristics from various perspectives, such as user perspective characteristics, packet-level and flow-level features [61, 62], routing protocol-level behaviors [60], and so on. Concerning application-layer traffic characterization, a sizable amount of recent studies concentrates on P2P traffic. A number of recent studies [29, 42, 43, 44, 45, 46, 47, 48] contributed to ascertain the nature of P2P traffic, particularly the traffic generated by FastTrack (KaZaA, KaZaA Lite), Gnutella (Morpheus, LimeWire, etc), and Overnet (eDonkey) that generates a significant share of Internet traffic. Other research [30, 31] focused on the comparison of P2P traffic with other

traditional Internet traffic, such as WWW and FTP.

A study [29] used TCP flow-level data gathered from multiple routers across a large Tier-1 ISP to analyze three P2P applications (KaZaA, Gnutella and DirectConnect). While this data does not reveal application level details and insights explaining the observed behavior, it is an important step in characterizing these applications from a network engineering perspective. For example, the study found that, although the distribution of generated P2P traffic volume is highly skewed at the individual host level, the fraction of the traffic contributed by each network prefix remains relatively unchanged over long time intervals.

Two recent studies [31, 32] considered that, although KaZaA's protocol (FastTrack) is proprietary, KaZaA uses HTTP to transfer data files, enabling this traffic to be logged and cached. Both these studies monitor HTTP traffic on costly links: traffic from a large Israeli ISP to US and Europe [32], or from the University of Washington campus to ISP [31]. They reported that most Internet traffic constitutes KaZaA traffic and a tiny number of files are generated in most download activities. They also suggested the feasibility of traffic caching, and empirically demonstrated its benefits. Furthermore, they compared KaZaA traffic with traffic generated by traditional content distribution systems, such as Akamai and Web traffic [32]. They quantified the rapidly increasing importance of P2P traffic, characterized the behavior of these systems from the perspectives of clients, objects, and servers, and derived implications for caching.

The characterization of current Internet traffic in this thesis is an expansion of the results of the recent studies. While the basic characteristics of these previous studies remain valid for our study, we investigate new aspects of the current traffic in the application layer that is not previously explored.

3. NG-MON

In this chapter, we describe the architecture of NG-MON, which is a passive traffic monitoring and analysis system suitable for high-speed network. We have considered many requirements in the design of NG-MON. With the distributed, pipelining, and parallel processing techniques, we have designed a flexible and scalable monitoring and analysis system, which can run on off-the-shelf, cost-effective computers.

3.1 Requirements

The following are the requirements we have considered in designing NG-MON.

- **Distributed architecture**: With a single general purpose PC system, it is hard to monitor and analyze all the packets on a multi-gigabit network. So it is required to divide monitoring task into several functional units and distribute processing loads. With respect to the distribution method, we considered the pipelined and paralleled methods. And also considered the packet distribution by using the functions which are provided by network devices.

- **Lossless packet capture**: We need to capture all packets on the link without any loss to provide required information to various applications.

- **Flow-based Analysis**: While analyzing, it is better to aggregate packet information into flows for efficient processing. By doing this, packets can be compressed without any loss of information.

- **Consideration of limited storage**: The amount of captured packets in high-

speed networks is more than hundreds of megabytes per minute even though aggregated into flows [64]. An efficient method is needed for storing these large amounts of flows and analyzed data in the limited storage space.

- **Support for various applications**: It should be flexible enough to provide data to various applications in diverse forms. When a new application needs to use the system, it should be able to easily support the application without changing the structure of the system.

3.2 Design of NG-MON

In the design of NG-MON, the key features we have employed are pipelined distribution and load balancing techniques. In Figure 3, traffic monitoring and analysis tasks are divided into five phases: packet capture, flow generation, flow store, traffic analysis, and presentation of analyzed data.

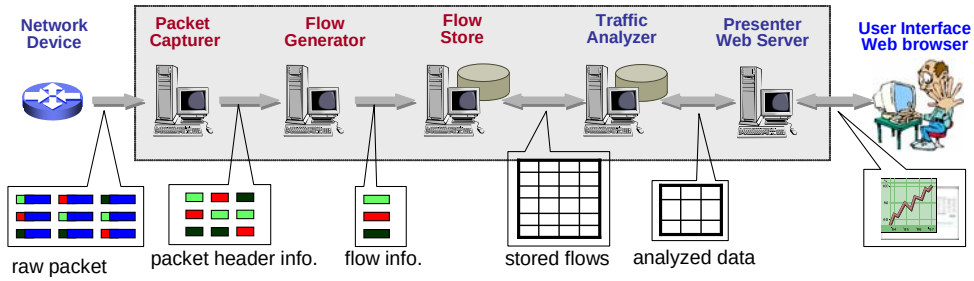


Figure 3. Pipelined Architecture of NG-MON

These five phases are serially interconnected using a pipelined architecture. One or more systems may be used in each phase to distribute and balance the processing load. Each phase performs its defined role in the manner of a pipeline system. This architecture can improve the overall performance. And each phase is configured with cluster architecture for load distribution. This provides good scalability. We have also defined a communication method between each phase. Each phase can be replaced with more optimized modules as long as they provide

and use the same defined interface. In the following sections, we describe each phase in detail. This gives flexibility. Rather than using expensive server-level computers, we use low-cast, off-the-shelf PC-level computers. Since our solution is all software-based, as more processing power is needed one can simply replace existing hardware or add more if needed. We believe this is a very cost-effective and scalable approach.

3.2.1 Packet Capture

In the packet capture phase, one or more probe machines (or packet capturer) collect the entire raw packets passing through the network link. By using the mirroring function provided in network devices such as switches and routers, all the packets on the link are directed toward probe systems as illustrated in Figure 4. Each probe processes incoming packets and keeps the minimum packet header information that we are interested in.

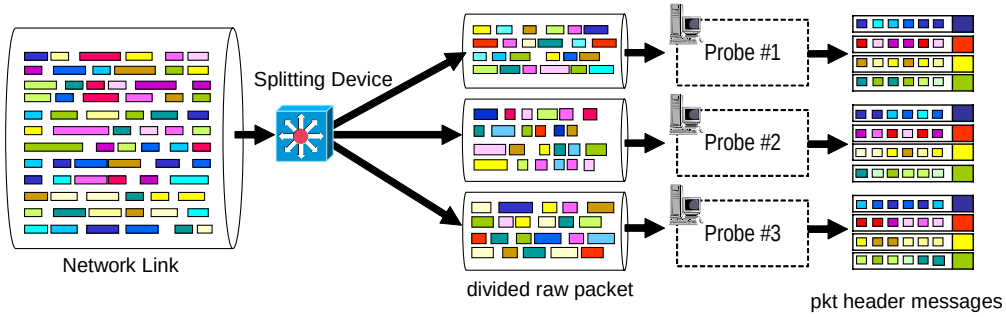


Figure 4. Packet Distribution and Packet Capture

A single off-the-shelf PC cannot process all the packets coming from the high-speed links such as 10 Gbps networks due to performance limitations [16, 17, 55]. It is essential to use multiple systems for capturing all the packets without loss. In the use of multiple such systems as probes, we can use an L4 switch to distribute packets over the probe systems. An L4 switch can distribute packets to probes using a round-robin distribution algorithm [65, 66]. Although processing loads are evenly distributed with the round-robin distribution, the packets in the

same flow can be scattered. Each probe has as many export buffer-queues as number of flow generators. One export buffer-queue is allocated for each flow generators. The probe fills the buffers with header information using the source IP based hashing over export buffer-queues. When this buffer-queue is full, the probe constructs a UDP message and then sends it to the next phase, the flow generator. The destination of the constructed UDP message is assigned among the flow generators as to the buffer-queues. Therefore, temporally scattered packets in the same flow would be gathered together into the same flow generator. One UDP message is composed of up to 50 packet header information. The format of raw packet header data is given in Figure 5.

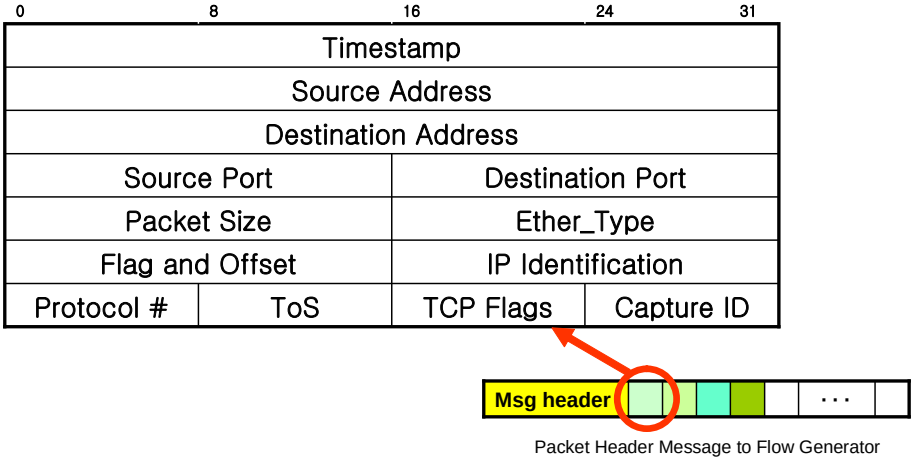


Figure 5. Packet Header Data Format

The size of packet header data kept is 24 bytes for each packet. All the fields except Timestamp and Capture ID are extracted from IP and TCP/UDP headers of each packet. The Timestamp indicates the time when a packet is captured by a probe. The Capture ID indicates the machine which captured that packet for later use.

3.2.2 Flow Generation

There are various definitions about the flow [64, 67, 68]. In this paper, we use

the traditional one, which defines the flow as a sequence of packets with the same 5-tuple header values: source IP address, destination IP address, protocol number, source port, and destination port.

In Figure 6, UDP messages from the packet capture phase are distributed over flow generators by assigning their destinations to the corresponding flow generators.

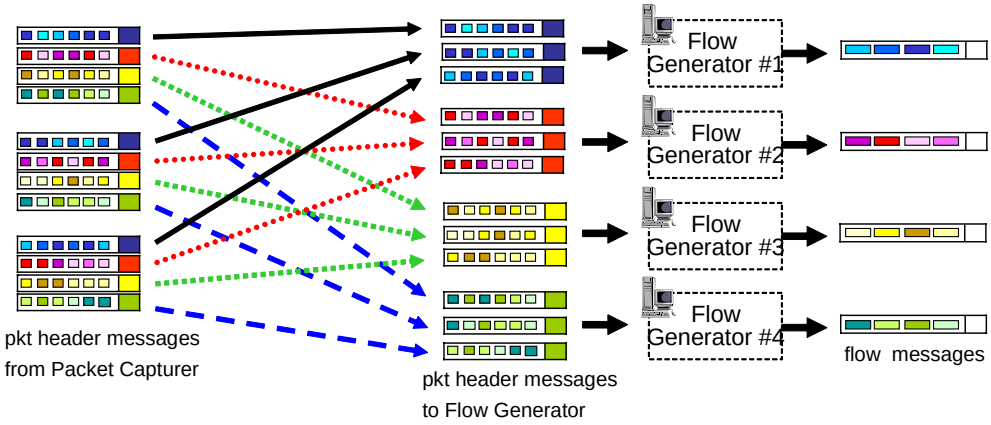


Figure 6. Load Distribution in the Flow Generation Phase

A flow generator stores the flow data in its memory area for processing. When a UDP message containing raw packet data arrives, the flow generator searches the corresponding flow data from its flow table and then updates it, or creates a new flow if one does not already exist. Packets in the same flow are aggregated into the same entry of the table by increasing the packet count and adding the length to the total packet size. The flow generator exports the flow data to the flow store when one of the following conditions is satisfied: when the flow is finished (if TCP, when a FIN packet received), the time has expired or the flow table is filled.

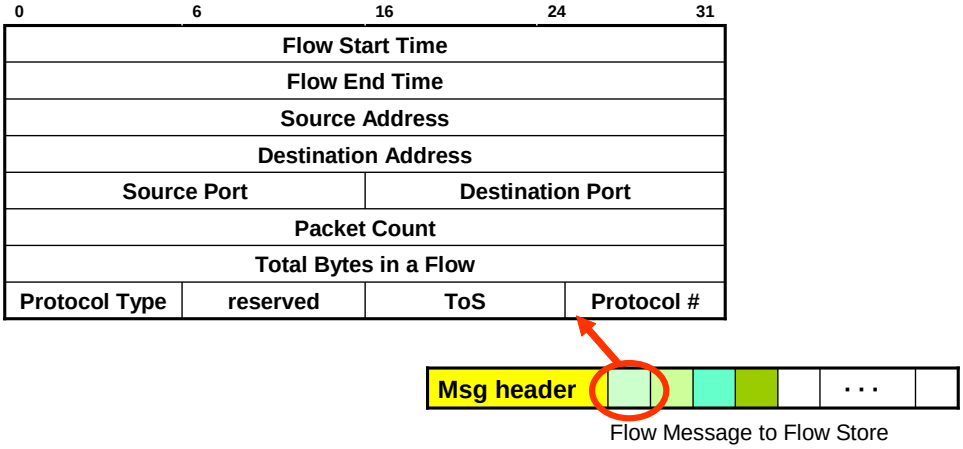


Figure 7. Flow Data Format

The flow data format is given in Figure 7. Besides the 5-tuple information, the flow data has several other fields such as flow start time and flow end time. The flow start time indicates the time when the first packet of a flow is captured by a probe, and the flow end time means the capture time of the last packet of the flow. The size of the flow data format is 32 bytes. For our flow generator, it can send up to 40 flows in a single UDP message of approximately 1350 bytes.

3.2.3 Flow Store

In our earlier work [56, 69], we realized that one of the bottlenecks of the monitoring process is a storing of flow data. Therefore, when the flow data is stored to the flow store, the load balancing should be considered. In Figure 8, the destination of the exported UDP messages is assigned among the flow stores in turn by a round-robin algorithm. The assigning period is determined by the transfer rate of export flow data, capabilities of the flow stores, and the number of flow stores. In this way, the processing load to store the flow data is distributed over the flow stores.

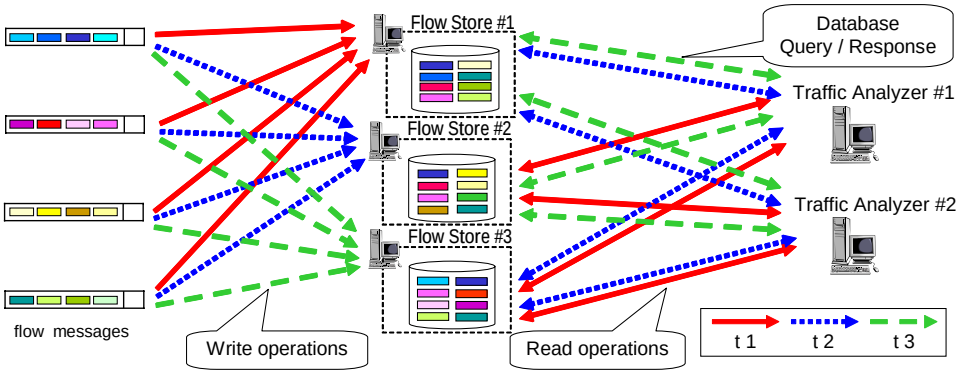


Figure 8. Load Distribution in the Flow Store Phase

When several flow generators assign a destination flow store of the UDP messages, there can be a time synchronization problem. But the components of NG-MON would tend to be deployed in a local area network, thus the required degree of time synchronization is not so high. Therefore, the time synchronization protocol like NTP [57] can be used to synchronize the system clocks of the components.

In our system, we separate write (i.e., insert) operations from database query operations performed by the analyzers. Insertion does not occur at the same time as other operations in a single flow store. Thus, traffic analyzers query databases of flow stores when they are not receiving flow data. An example is illustrated in Figure 8. At time t1, the flow store #1 receives flow data from flow generators and the flow stores #2 and #3 process the query from traffic analyzers. That is, the flow store concentrates on requests of one side at a time. Flow stores discard the flow data table when they are finished with analysis by traffic analyzers. Only the most recent flow data is stored in the flow store, so the flow store only requires a small, fixed amount of disk space.

There can be various traffic analyzers for supporting various applications after the flow store phase. This means that the flow store should provide an efficient analysis API to analyzers.

3.2.4 Traffic Analysis

In this phase, the traffic analyzer queries the flow data stored in the flow store according to the various analysis scopes. It sends query messages to the flow stores and makes various matrices and tables from the response. If all the scope of analysis is put into one table, the size of a single table will be too large to manage. Therefore, we place several reduced set of tables corresponding to each analysis scope. For example, NG-MON Analyzer in Figure 9, provides the details on network throughput with layered and temporal analysis. And in order to provide temporal analysis, the analyzer has a set of tables according to every time-unit of minute, hour, day, month, and year. It is impractical to store all the flow data into the time-series tables because of voluminous data, and limitation of storage space. To reduce the storage requirement, we preserve tables with only the most significant N entries. Thus, the total size of database will have some degree of boundary. The analyzer fills up the tables simultaneously in a pipelined manner. If the reception time period of flow stores is 5 minutes, there can be 20 tables for storing every 5 minutes' analyzed flow data. After updating the 5-minute table, the corresponding hour table gets updated. There should be these kinds of time-series tables for each scope of analysis in the analyzer.

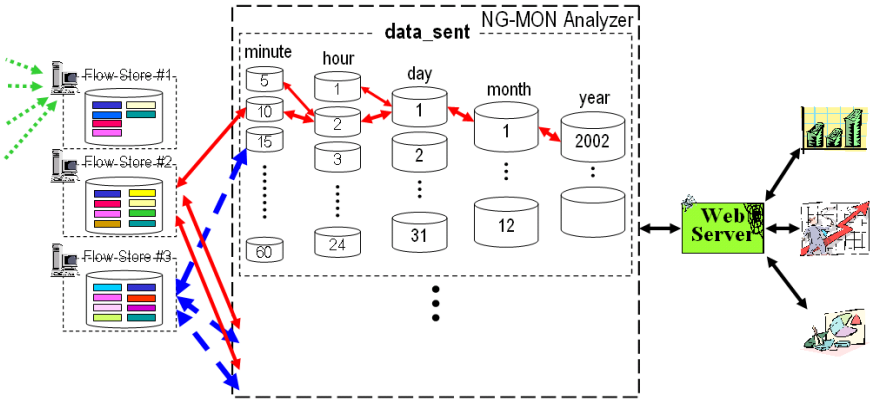


Figure 9. Traffic Analyzer and Various Applications

The presentation phase provides traffic analysis results to users in various

forms using Web interfaces. Before designing an analyzer, we have to determine analysis items to be shown in this phase. Then the traffic analyzer can generate the corresponding DB tables based on these items. That is, a different analyzer is required to support different purpose presenters. Because tables contain analyzed data which is ready to be shown, the time needed to create reports and HTML pages is very short, typically less than a second.

3.2.5 Presentation

This phase provides analyzed data to corresponding applications. Because the header information of all packets has been stored to the flow store being compressed into flows, it can provide any information to applications in a flexible and efficient way. NG-MON can provide necessary information to the billing applications on IP networks, IDS systems, and so on.

3.3 Design Analysis

We have validated our design of NG-MON analytically for monitoring high-speed networks, such as 10 Gbps. We already described the flexibility and the scalability of our design. At each phase, we can assign a number of systems for load distribution, or can merge some phases into one system. The appropriate number of systems will be determined from this analysis.

3.3.1 Assumptions

The monitored network link speed is 10 Gbps, and our system captures all inbound and outbound packets. The size of a single packet header data is 24 bytes, and that of a single flow data is 32 bytes.

Link speed: L=10 Gbps
Full duplex link (in&out): *multiplication factor* = 2

A single packet header data size: H_p = 24 bytes
A single flow data size: H_f = 32 bytes

Then we calculate the size of data to be processed in a second in a probe, flow generator, and flow store.

3.3.2 The amount of data to be processed

The total raw packet (T_p) in the packet capture phase, the total raw packet header information (T_h) in the flow generation, and the total flow data (T_f) in the flow store processed in one second are as follows:

Total raw packet (T_p) = L(10) × in&out(2) / 8 = 2.5 Gbytes
Total raw packet header data (T_h)

$$= \frac{T_p(2500)}{\text{average packet size}(400)} \times \left[H_p(24) + \frac{\text{MAC header}(14) + \text{IP header}(20) + \text{UDP header}(8)}{\text{the number of raw packet headers in an export UDP packet}(50)} \right]$$

$$= 363.5 \text{ Mbytes}$$

Total flow data (T_f)
= T_p(2500) / average packet size(400) × H_f(32) / average number of packets per flow(16)
= 12.5 Mbytes

The average number of packets per flow (P_{avg}) is 16-20 for TCP [58]. As the major portion of traffic is TCP (85-95 percent of total packets on the Internet), we assumed it as the average number of whole packets per flow. Though P_{avg} can be different as flow timeout interval varies, it converges approximately on 16 as the link speed and timeout interval increases.

It seems meaningless to calculate the size of total flow data in a second (T_f) because it is too short to be used as an assigning period in the flow generation phase. So we observe the size of flow data in a minute (1 min-T_f), five minutes (5 mins-T_f), and an hour (1 hour-T_f):

1 min-T_f = 12.5(T_f) × 60 = 750 Mbytes
5 mins-T_f = 3.75 Gbytes
1 hour-T_f = 45 Gbytes

If we choose one minute as the assigning period, each flow store requires only 750 Mbytes of disk space. In the same way, if we choose 5 minutes, 3.75 Gbytes are required per flow store.

3.3.3 Allocation of systems to each phase

The amount of data to be processed at each phase is typically beyond the processing capacity of a single off-the-shelf general-purpose PC system. For example, a single probe cannot process all the raw packets of 2.5 Gbytes in one second because of the limited network interface card (NIC) and PCI capacities in a PC.

	NIC (GE)	**PCI bus** (66MHz/64bit)	**Memory** (RDRAM, 800MHz/16bit)	**CPU** (P4, 2GHz)
Bandwidth (Mbytes/sec)	125	533	3,200	128,000

Table 1. Selected System Configuration

During processing in a single system, there are several subsystems that affect the monitoring capacity: NIC bandwidth, PCI bus bandwidth, memory, CPU processing power, storage and so on. Let us consider a computer system with a gigabit Ethernet NIC and 66MHz/64bit PCI bus and 1 Gbyte RDRAM (800MHz/16bit) and 2GHz Pentium 4, like in Table 1. Then the theoretical max transfer rate of a gigabit Ethernet NIC is 125 Mbytes/sec, PCI bus is 533

Mbytes/sec, and dual channel RDRAM (800MHz/16bit) is 3.2 Gbytes/sec [70, 71]. The probe can have multiple NICs, one for sending raw packet header information to flow generators, the others for capturing. Then, there can be 4 NICs within the bandwidth of a PCI bus so far as the number of PCI slots permits. Therefore, it requires 7 probe machines to receive total raw packet of 2.5 Gbytes in a second.

In the flow generator phase, it receives 363.5 Mbytes of raw packet header per second (T_h), so theoretically at least 3 flow generators are needed. In the flow store phase, though it is sufficient for processing the rate of flow data with one system, the execution time of queries affects required number of flow stores. Such an execution time varies as to the kind of database system, DB schema, query construction, and so on. Therefore, the number of flow stores is flexible regarding to those kinds of factors. In our previous work [56, 69], it took about 4 seconds to insert 20 Mbytes of raw packet headers into MySQL database running on an 800 MHz Pentium 3 with 256 Mbytes of memory. If we assume the runtime of an insert is O(N), it will take 150 seconds to insert 1 min-T_f data into the database. Here we assume the analysis system takes about 2 minutes for querying. Then it will take 4 minutes and 30 seconds to insert and analyze the 1 minute flow data. As we have to finish all these operations within 1 minute, it requires 3 systems for inserting, and 2 systems for analyzing in the flow store phase.

	Packet Capture	Flow Generation	Flow Store	Total
100 Mbps	1			1
1 Gbps	1		2	3
10 Gbps	7	1	5	13

Table 2. The required number of systems in each phase

Therefore, it requires approximately 15 systems (7 in the packet capture phase, 3 in the flow generation phase, 5 in the flow store phase) to provide flow data to analysis systems in a fully-utilized, full-duplex 10 Gbps network. Table 2 shows the other cases of 100 Mbps and 1 Gbps. In a 100 Mbps network, the amount of flow data in a minute is less than 10 Mbytes. Thus, three phases can

merge into one system. In a 1 Gbps network, packet capture and flow generation phase can merge into one system which has 3 NICs. And the flow store phase can be composed of 2 systems if the time to insert and query the 1min-T_f of 1Gbps network is less than a minute per each operation.

3.4 Comparison with Other Monitoring Systems

Table 3 compares NG-MON with other passive network monitoring systems. Ntop [75] is a monitoring software system that provides detailed protocol analysis and a graphical user interface. However, Ntop is not suitable for monitoring high-speed networks from our experience in deploying in a real network. It cannot scale beyond monitoring a fully-utilized 100 Mbps link.

	Ntop	FlowScan	CoralReef	Sprint IPMon	NG-MON
Input Data Format	raw packet NetFlow	NetFlow	raw packet	raw packet	raw packet
Analysis Purpose	Throughput Protocol	Throughput Protocol	Throughput Protocol	Packet Trace	Throughput Security Protocol
Maximum Link Speed	Less than 100Mbps	Less than 155Mbps	Less than 622Mbps	More than 10G	More than 10G
H/W or S/W	Software	Software	Hardware + Software	Hardware + Software	Hardware + Software
Sampling Method	No	Yes (Network Device)	Configurable	No	No but Configurable
Analysis Type	On-Line	On-Line	On-Line	Off-Line	On-Line

Table 3. Comparison of NG-MON with Other Monitoring Systems

FlowScan [52], a NetFlow analysis software system, can only process NetFlow data format and is not suitable for monitoring high-speed networks either. Ntop and FlowScan are appropriate for analyzing relatively low speed networks such as a WAN-LAN junction or a LAN segment. The distributed and load-balancing architecture of NG-MON may be applied to Ntop and FlowScan to improve their processing capabilities. CoralReef [27] can monitor up to OC-48 network links (622Mbps of maximum link utilization). Considering the load

distribution, CoralReef only suggests a separation of flow generation and traffic analysis, but without consideration of clustering of processing systems in each stage. Sprint's IPMon project [60] developed a probe system for collecting traffic traces, which is used for off-line analysis after transferring to a laboratory. Their approach uses specialized NIC (DAG card) to assist the packet capturing and processing.

Our NG-MON has been designed for monitoring high-speed IP networks. It takes raw packets as input, and analyzes captured data online and then generates various throughput related data. NG-MON is a software-based solution (i.e., does not depend on any specific hardware), which can be easily installed and run on a variety of UNIX and Linux platforms. NG-MON does not use sampling for capturing packets without any loss. However, the system configuration user interface allows the system to be configured to capture packets using sampling if needed.

4. Application-Level Traffic Identification

In this chapter, we propose a new method to identify Internet traffic type in the application level, which suits sophisticated current Internet traffic. First, we will investigate the communication behavior of current Internet applications and classify them accordingly. Second, we will present the proposed method for identifying Internet traffic based on this classification.

4.1 Communication Behavior of Internet Applications

The communication behavior of current Internet-based applications is very complex. Traditional client/server based applications such as web, telnet, nntp, and smtp applications, usually use a single TCP or UDP session with a fixed port number to communicate with each other. However, newly emerging Internet-based applications use multiple sessions with dynamic ports, which makes traffic identification more difficult. In this section, we examine the communication behavior of recent Internet-based applications and categorize them from this perspective.

Type	Applications / Application layer protocol
Traditional	http, https, ftp, telnet, ssh, nntp, dns, smtp, pop3, timed, etc.
Streaming Media	rtsp, sip, mms, rtp/rtcp, rdt, mmsu/mmst, q.931, h.245, etc.
P2P	gnutella, fasttrack, overnet, directconnect, msn messenger, etc.
Game	starcraft, warcraft, diablo, counter strike, etc.
Internet Disk	popdesk, internetdisk, webhard, woorihard, coolhard, etc.

Table 4. Current Internet-based Applications

Table 4 shows a list of current Internet-based applications that generate most

Internet traffic. We categorized these applications according to their service type and protocol. Traditional applications use standard and open protocols that simplify identification of the traffic generated by them. Other newly emerging applications that began to be developed from few years back use application layer protocols such as RTSP, SIP, Q.931, and H.245 to deliver control data between streaming server and client. RTP/RTCP and MMSU/MMST are used to transfer multimedia data from a server to client. The QuickTime streaming service and Real Networks streaming service use open protocols called RTSP and RTP/RTCP while Microsoft's Windows Media service [36] uses a proprietary protocol called MMS. Most popular peer-to-peer applications are file sharing applications, such as FastTrack (KaZaA, KaZaA Lite) [32], Gnutella (BearShare, Bnucleus, Morpheus, LimeWire) [48], and instant messaging applications (MSN messenger). Many game applications use networks to make multi-user games interactive. In addition, Internet disk service provides another type of file sharing method. A service provider may construct a large amount of file server at Internet Data Centers (IDC) and its users can use a portion of the provided disk space and share it with other users by paying fees.

Type	session	Port	Hosts	Example
Type S-F-2	Single	Fixed	Between two	Web, telnet, ssh, smtp, snmp, nntp
Type M-F-2	Multiple	Fixed	Between two	Active mode FTP
Type M-D-2	Multiple	Dynamic	Between two	Passive mode FTP Streaming Applications (Quicktime,
Type M-F-3	Multiple	Fixed	Three or more	Instant messaging p2p application (Soribada) File sharing p2p (Gnutella, DirectConnect) Game application (starcraft, Diablo)
Type M-D-3	Multiple	Dynamic	Three or more	Instant messaging p2p (MSN messenger) File sharing p2p (Kazaa, Kazaa Lite) Game application (Counter-Strike) Internet Disk (popdisk, webhard)

Table 5. Classification of Communication Behaviors

We investigated the communication behavior and the port numbers used by these widely deployed applications from the traffic monitoring and analysis perspective. We selected more than 100 popular applications on our campus network and installed them in several systems. We captured all the packets generated by these applications in each end systems using Ethereal [51] and Tcpdump [41], and examined their behaviors, especially the source and destination port numbers, peers' IP addresses, the direction of transferred data, and the number of established sessions. From the investigation of each application, we categorized the communication behavior into five types as shown in Table 5. We used three types of information to categorize the communication behaviors: the number of sessions among the involved systems, the way of selecting port number, and the number of involved systems to provide a service. We describe the details of each type below.

4.1.1 Type S-F-2

This type is the communication behavior of most traditional Internet-based applications such as web, telnet, ssh, smtp, and etc. The communication structure is client/server architecture and uses a well-known fixed port number. A server may simultaneously communicate to multiple clients. However, those communications are independent of each other from the clients' perspective; No communication occurs between clients, but data is transferred only between a client and the server. The traffic of this type can easily be determined. The port number used by each application is the key for application identification: for example, 80 for http, 23 for telnet, 22 for ssh, 25 for smtp.

In cases where the fixed port number of an application is above 1024 - for example, port number of 3306 used for MySQL applications - we need to take a preliminary step before applying this method. That is to decide which port number should be considered between source and destination port numbers because the randomly determined client port numbers by system are greater than

1024. The selection method of important port numbers will be described in Section 4.4.

4.1.2 Type M-F-2

The best example of this type of communication behavior is FTP communication with active mode like Figure 10-(a). This type uses two different sessions to perform FTP service; one is a control session and the other is a data session. In an active mode FTP, well-known fixed port number of 21 is used for control session. FTP servers open the port number 21 and wait for the connection request from clients. The port number 20 is used to transfer data between client and server as illustrated in Figure 10-(a).

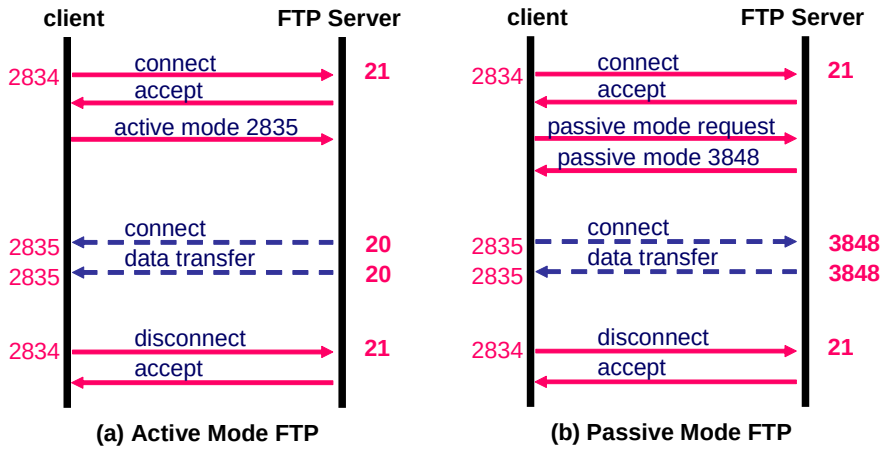


Figure 10. Two Types of FTP Communications

Identifying this type of application is straightforward like the previously mentioned Type S-F-2. Packets with port number 21 are FTP control packets. Packets with port number 20 are FTP data packets if and only if the traffic with port number 21 between the two same hosts appears. We can use the IANA port list to determine this type of applications. However, the same problem of the Type M-F-1 is occurred in this type of traffic. For example, RealVNC application use port number 5800 and 5900, MS-SQL applications use port number 1433 and

1434, respectively.

4.1.3 Type M-D-2

This type of communication behavior uses multiple sessions between two hosts like the Type M-F-2. However, the port numbers for one or more sessions are dynamically determined by the negotiation between the two involved hosts. The best examples of this type of communication are the FTP service with passive mode and the streaming media service.

As illustrated in Figure 10-(b), using the control session port number of 3848, which is decided by the FTP Server, is delivered to the client, and the client makes a TCP connection to the server using that port number. The dynamically generated session can use TCP or UDP according to the applications. Most streaming media applications use UDP sessions to deliver multimedia data from media server to client. The Figure 11 illustrates a detail communication of streaming media applications.

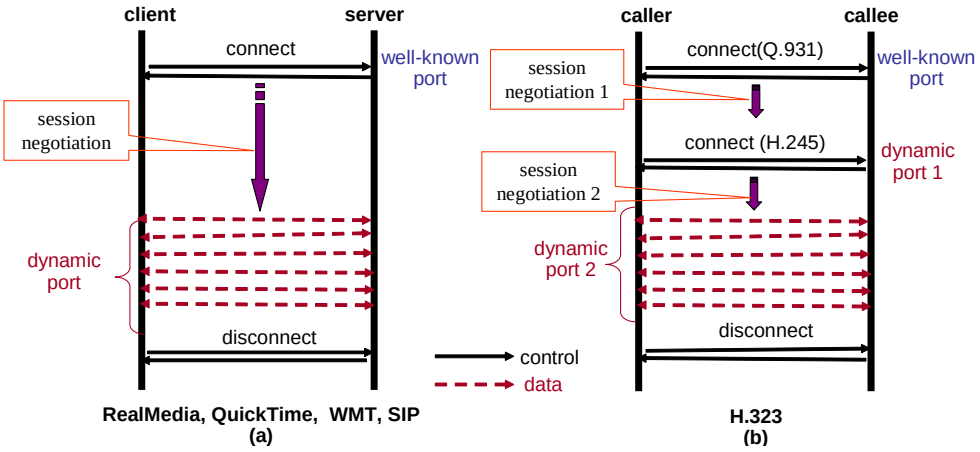


Figure 11. Multimedia Service Control and Data Session

Figure 11 illustrates a process in which control sessions and a data session are constructed and then multimedia data is transferred. To begin, the control session is constructed through a well-known default port number: 1755 for MMS, 554 for

RTSP, 5060 for SIP, 1720 for Q.931. As described in Figure 11-(a), streaming media services (e.g. Real Networks, QuickTime) or applications based on SIP have one control session. On the other hand, H.323 applications have two control sessions: Q.931 and H.245 sessions. The control sessions create a new data session by negotiating a transport protocol and port numbers. The data session then transfers multimedia data through the dynamically assigned transport protocol and port numbers. In this paper, we introduce a new term, ***dynamic session*****.** It makes a use of the transport protocol and port numbers that are dynamically negotiated by the control session for data session and H.245 control session in Figure 11, and data session in passive mode FTP service in Figure 10-(b). Moreover, Type M-F-2 could be considered as a specific case of this type.

Identifying the dynamic session of this type is not simple. We may use the payload examination method used in the mmdump [37] and SM-MON [20, 21]; when a control session negotiates a dynamic session, the packet payload of the control session contains negotiation results such as a transport protocol and port numbers used in the dynamic session. Otherwise, a heuristic method may be considered; when a flow occurs with an unknown port number between two hosts while an active control connection exists between them, it assumes that the flow corresponds to a dynamic session. This method is adopted by FlowScan [52], a flow-based traffic analysis system.

4.1.4 Type M-F-3

Many of new applications use multiple sessions and communicate with multiple peers simultaneously. P2P file sharing application, P2P instant messaging application, and game applications are good examples. Figure 12 illustrates two prevalent types of P2P communication architecture used by those applications: ***the central arbiter type*** and ***the pure distributed type***.

Most instant messaging P2P applications, game applications and Internet disk services are based on the central arbiter type. In this communication type, one or

more central servers exist and they contain all the information of each peer and send the information to any requesting peers. This architecture is easier to implement and maintain status information. However, the central server could be a single point of failure and much of the traffic may be concentrated on the central servers.

Some file sharing P2P applications such as Gnutella [48] uses the pure distributed communication type where there is no centralized server. Thus, possibility of service failure is less likely than in the central arbiter type and the search and file transfer mechanism is efficient.

A hybrid type of these two critical types is considered in many P2P applications: the concept of SuperNodes in KaZaA [32] and UltraPeers in Gnutella [48].

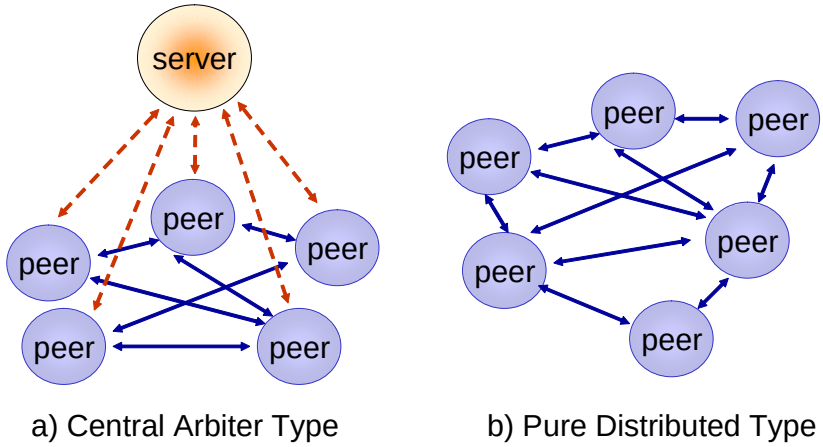


Figure 12. Peer-to-Peer Communication Architecture

As illustrated in Figure 12, a peer communicates with other peers or central servers simultaneously, which establishes multiple sessions. Many P2P applications and game applications use fixed port numbers for theses multiple sessions and use both TCP and UDP protocol without any prejudice. This type of communication type is derived from the behavior of these applications. The multiple sessions with fixed port numbers among more than three systems are the

features of this type of communications. Although multiple sessions are involved, the use of fixed port numbers makes the determination of type M-F-3 traffic straightforward

4.1.5 Type M-D-3

Early versions of the MSN messenger used a fixed port number to transfer files between peers: TCP port number 6891. However, the latest version of MSN messenger (version 6.1) uses a dynamically assigned port number for file transfer. Also earlier versions of KaZaA [32] used port number 1214 for data transfer, but the latest version of KaZaA can allocate the server port number dynamically as needed. Figure 13 illustrates the communication behaviors of current versions of KaZaA Lite K++ and MSN messenger applications.

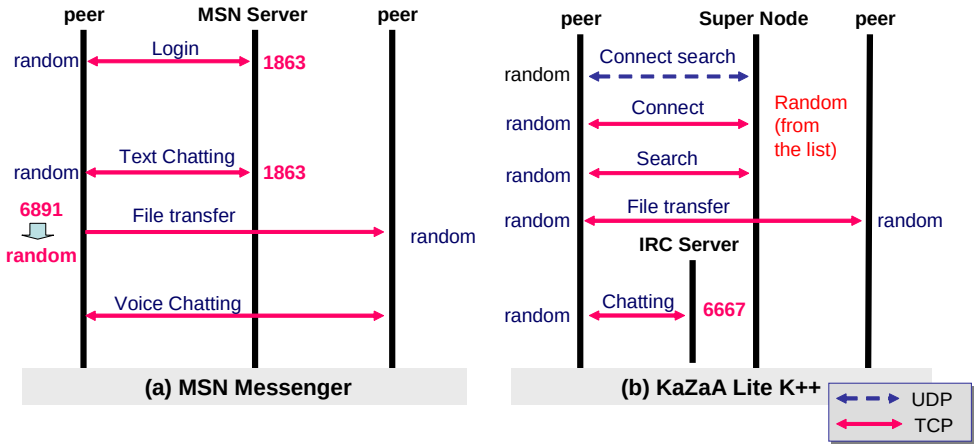


Figure 13. Communication behaviors of MSN Messenger and KaZaA Lite K++

As seen in Figure 13-(a), a MSN peer logs on to one of the MSN servers, using server port 1863, and talk with peers via another MSN server having the same port number 1863. To transfer a file to a peer, the earlier version used a fixed port number 6891, but the latest version uses a dynamically assigned port number that is negotiated between the two peers via MSN server. MSN messenger application also provides some other functions such as voice chatting, video

chatting, application sharing, and etc. MSN messenger also uses other dynamically allocated port numbers to perform these functions. Earlier versions of MSN messenger traffic are of M-F-3 type and the latest version of MSN messenger is of M-D-3 type. To determine the traffic of the latest version of MSN messenger, we should develop a new method.

The latest version of KaZaA uses completely random port numbers with no default port number like MSN messenger does. As illustrated in Figure 13-(b) a user can configure the default port of his KaZaA client to anything. In addition, the SuperNodes of KaZaA select the server port numbers randomly. A user may use an IRC chat service to find out the IP addresses and listening port numbers of SuperNodes. The port number for data transfer is also dynamically allocated by the negotiation between peers.

The traffic type M-D-3 is the most difficult to identify. Therefore, we need an efficient method to identify Internet traffic in application level. The communication behavior of many current Internet-based applications is shifting from the Type M-F-3 to the Type M-D-3. One of the reasons of this migration is to evade the detection and control.

In the next section, we propose a method to identify current Internet traffic in application level. The proposed method is based on the classification of communication behaviors described in this section. We should also consider some additional issues. First, two different applications may use same port numbers. Second, we may not examine the protocol format and its operation on all applications. Third, we cannot capture all packets flowing through the Internet. Moreover, the capturing system may not be able to capture all packets passing through a target link. Fourth, between source and destination port numbers in a packet, we must select the ports that are significant in the decision of application name.

4.2 Notations for Flow Grouping Method

We define some basic notations to describe our proposed method, which is illustrated in Table 6.

Notation	Description
$f_{(sip, dip, dip, dport, proto)}$	A flow with ***sip***, ***sport***, ***dip***, ***dport***, ***proto***; a long form of a flow
f_a	A short form of a flow notation
$f_a(ip)$	The ***important port*** of a flow f_a
$f_a(sip)$	The ***sip*** of a flow f_a
$f_a(sport)$	The ***sport*** of a flow f_a
$f_a(dip)$	The ***dip*** of a flow f_a
$f_a(dport)$	The ***dport*** of a flow f_a
$f_a(proto)$	The ***proto*** of a flow f_a
${}_rf_a$	The reverse flow of a flow f_a: $f_a(sip)= rf_a(dip)$, $f_a(sport)= rf_a(dport)$, $f_a(dip)= rf_a(sip)$, $f_a(dport)= rf_a(sport)$, $f_a(proto)= rf_a(proto)$
$f_a = f_b$	The complete equality: $f_a(sip)= f_b(sip)$, $f_a(sport)= f_b(sport)$, $f_a(dip)= f_b(dip)$, $f_a(dport)= f_b(dport)$, $f_a(proto)= f_b(proto)$
$f_a = f_b \mid (x,y,..)$	The conditional equality: $f_a(x)= f_b(x)$, $f_a(y)= f_b(y)$, ..
A_a	An application
$G(A_a)$	An application group, the elements of $G(A_a)$ are the flows generated by the application A_a.

Table 6. Notations for FRM

We use two notations to describe a single flow: a long form and a short form. The long form of flow notation is expressed by the flow definition; a unidirectional series of IP packets of a given protocol traveling between source and destination IP/port pair within a certain period of time, which is distinguished by the 5-tuple values: source IP address, destination IP address, source port, destination port, and protocol. The short form of flow notation makes it easy to read the equations with many flows.

The reverse flow of a flow f_a is denoted as ${}_r f_a$. The reverse flow, ${}_r f_a$, is a flow f_b that satisfies the following five conditions; the protocol of f_b is equal to that of f_a; the source port of f_b is equal to the destination port of f_a; the source IP address of f_b is equal to the destination IP address of f_a; the destination port of f_b is equal to the source port of f_a; and the destination IP

address of f_b is equal to the source IP address of f_a.

Two flows, f_a and f_b, have complete equality if and only if all 5-tuple values of f_a are equal to those of f_b. We denote the complete equality of the two flows f_a and f_b with $f_a = f_b$. f_a and f_b have conditional equality if and only if some of the 5-tuple values of f_a are equal to those of f_b, respectively. Generally, we use the following notation to describe the conditional equality: $f_a = f_b \mid (x, y, ...)$. ***x*** and ***y*** can be any of the 5-tuple notations (***sip***, ***dip***, ***sport***, ***dport***, ***proto***).

4.3 Traffic Identification Method using Flow Grouping

In this section, we present our proposed method for Internet traffic identification. The main idea of the proposed method is as follows. We determine dependencies among flows that are generated by same applications, and group the flows according to their corresponding applications. For example, Web traffic (Type S-F-2) typically uses port number 80 or 8080 for HTTP and 443 for HTTPS as the default port number. Type M-F-2 traffic can easily be grouped according to their default port numbers. However, in the case of Type M-D-3 traffic (e.g. P2P traffic), flow grouping is not as simple as in the case of Web traffic, because they use port numbers greater than 1024 and many port numbers are dynamically assigned. If all Internet traffic could be grouped according to their applications, Internet traffic analysis and characterization would be performed with higher accuracy.

Our proposed method consists of three steps, as illustrated in Figure 14. These steps are the Application Port Table (APT), Important Port Selection (IPS), and Flow Relationship Map (FRM). In our proposed method, we do not examine the payload of each packet. Instead, we only use the packet header information after aggregating them into flows.

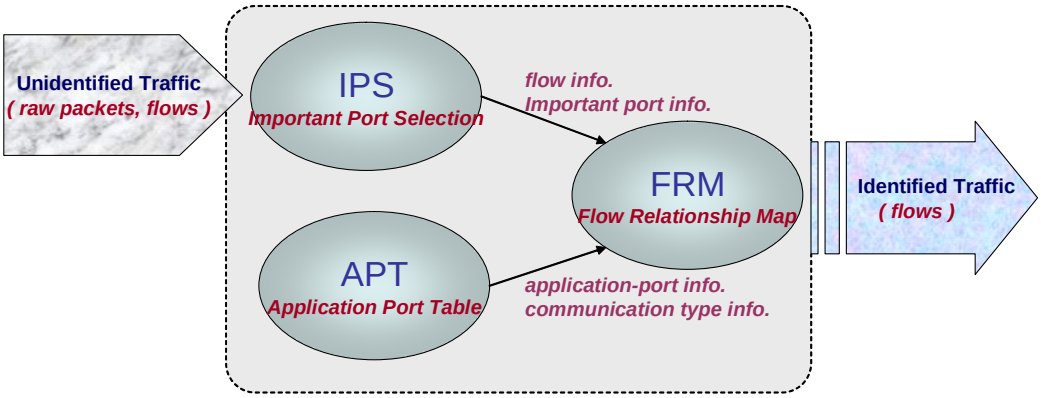


Figure 14. Overall Process of Traffic Identification Method

The first step of the proposed method is to construct an Application Port Table (APT). An APT is constructed by an exhaustive off-line search of each application using packet analysis tools. It contains the application name, frequently used port numbers excluding dynamically assigned port numbers, and transport-layer protocol numbers. This information is used to decide the application name of each flow in the FRM step.

The second step is to do Important Port Selection (IPS). The input of IPS may be either a raw packet or flow data. The outputs of IPS are flow data and important port information. In this step, the flow information is generated from the captured packets according to their 5-tuple information: source IP address, destination IP address, source port, destination port, and protocol. We then select the important port number from the flow information of TCP flows. Because both source and destination port numbers of most flows exceed 1024, it is necessary to distinguish the important port to decide the corresponding application name correctly.

The third step is to construct Flow Relation Map (FRM). The output of the IPS, which is flow data, is the input of the FRM. The FRM groups flow according to their relationships and marks flows with corresponding application name. Most of newly emerging applications use multiple connections with one or multiple

peers to perform various functions. Hence, it is possible to discover a relationship among flows that belong to the same application. Using this dependency information, we classify flows into a number of groups. For example, the flows caused by MSN messenger applications are grouped into a single group by the FRM. After this grouping process, the FRM determines the application name of each flow group using APT information.

4.4 Application Port Table

As mentioned above, the first step of the proposed method is to construct an Application Port Table (APT). To decide the application name from captured flow information, we perform preliminary examinations of the communication behaviors for widely-used applications. We determine the name, default port number, transport protocol, and communication type of these applications. For an exhaustive search of applications, we used packet analysis tools such as Tcpdump and Ethereal. Based on this investigation, we construct an APT that contains the information about each application. An APT contains the application name, frequently used TCP/UDP port numbers, one representative port number, and communication type. It is also an extension of the IANA port list, including some additional information.

We examined more than 100 popular applications and constructed APT. Table 7 is a small portion of our Application Port Table. As demonstrated in Table 7, most P2P applications use multiple port numbers. Most port numbers are not registered in IANA port list [34], especially the port numbers used by the applications targeting regional users; for example, only for Korean users. Some P2P applications such as Soribada [83] and ShareShare [54] simultaneously use both TCP and UDP. Nevertheless, most of applications use TCP only.

Application Name	Representative Port	TCP well-known ports	UDP well-known ports	Communication Type
WWW	80	80, 8080, 443		Type S-F-2
Telnet	23	23		Type S-F-2
FTP	21	20, 21		Type M-D-2
MSN Messenger	1863	1863, 6981-6990, 14594		Type M-D-3
Yahoo Messenger	5101	5101, 5050		Type M-F-3
Windows Media	1755	1755		Type M-D-2
Real Media	554	554		Type M-D-2
KaZaA	1214	1214		Type M-D-3
Soribada*	22322	22322, 7675, 7676, 7677	22321, 7674	Type M-F-3
eDonkey	4661	4661, 4662, 6667		Type M-D-3
Guruguru*	9292	9292, 9999		Type M-D-3
V-share*	8404	8403, 8404		Type M-D-3
Shareshare*	6399	6399	6388, 6733, 6777	Type M-D-3

* Applications targeting regional users (e.g., users in Korea)

Table 7. An Example of Application Port Table

We also record the communication type of each application according to the classification in Table 7. We use this communication type information in the FRM process to finalize the grouping of flows. We select one port number as the representative port of each application among the frequently used port numbers. Even though an application may use both TCP and UDP and many port numbers, only one representative port number is assigned to one application. This representative port number is used to indicate the application name of flows.

4.5 Important Port Selection

In this section, we describe the proposed algorithm to select an important port number among source and destination port numbers in each flow data. The important port number is the port number that is necessary for traffic identification. The proposed Important Port Selection (IPS) algorithm considers only TCP flows. Figure 15 demonstrates a normal TCP communication sequence.

To establish a connection between a client and a server, a 3-way handshaking mechanism is applied. To terminate the connection, FIN packets are sent to each other.

In TCP communication, the server opens a port and waits for a client connection request. The server listening port is an important port for identifying TCP traffic; then how can we select the server listening port from the captured flow information?

First, we utilize SYN and SYN-ACK packets taken from the 3-way handshaking operation. The destination port in a SYN packet and the source port in a SYN-ACK packet are the server listening port. Using this information, we can determine the important port number from all TCP flows. It is important to check both SYN packet and SYN-ACK packet because the SYN packets without SYN-ACK packets are captured frequently in a real Internet environment. Therefore, the preliminary step of IPS is to determine whether a flow has a corresponding reverse flow. In deciding the important port number, we exclude flows that do not have corresponding reverse flows.

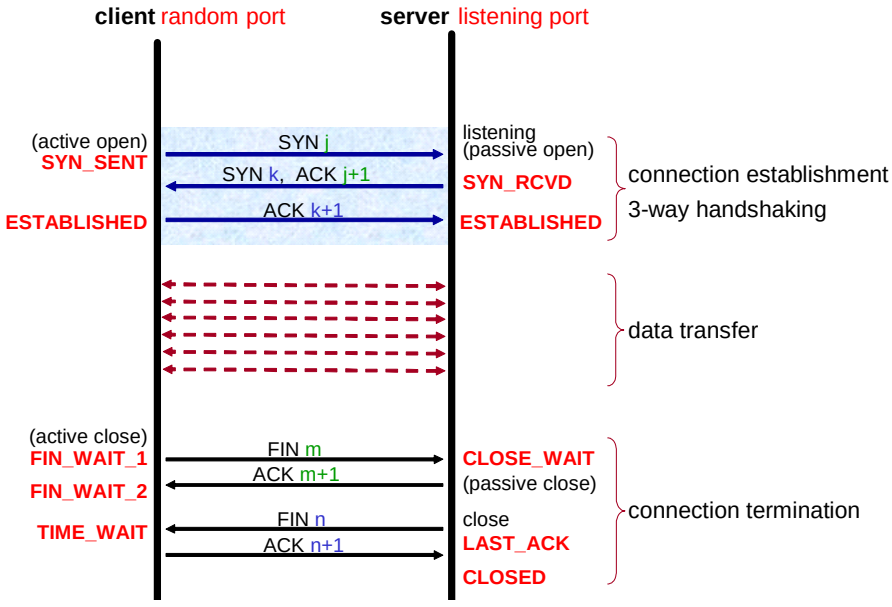


Figure 15. TCP Communication Sequence

To improve the performance of the IPS algorithm, we use the fact that no system uses port number less than 1024 as random client port number; randomly system generated port numbers are always greater than 1024. We choose an important port by checking port numbers less than 1024, before we apply the proposed SYN/SYN-ACK packet based method.

In case of UDP flows, we cannot apply this kind of method because it does not use 3-way handshaking mechanism like TCP does. Instead, we may use the flow dependencies between UDP packets to choose an important port number. We know by experiments that patterns of UDP flows are simple compared to those of TCP flows. Therefore, finding a relationship among UDP flows is not very difficult.

We should also consider various cases that could arise in a real-world environment. First, we may not be able to capture entire packets (SYN and SYN-ACK packets) needed for the IPS algorithm in a high-speed network link. Sometimes, intentional packet drops occur, especially when sampling is needed. Sometimes, unintentional packet drops may occur due to asymmetric routing and performance limitations of a capturing system. Secondly, we should consider long lasting TCP sessions; a streaming media service, VoIP service, or network game connections may last for a long period (e.g., more than 30 minutes). SYN and SYN-ACK packets appear only at the beginning of a TCP connection. In a case that an IPS module starts capturing packets from the middle of these connections, we cannot determine the important port for these flows. Thirdly, we should consider the flow data, such as Cisco NetFlow data, as an input to our IPS module to extend our method to various environments. In this case, the flow information may not have TCP flag information that is the key to identify an important port; for example, Cisco NetFlow V5 data does not have this information. To solve this problem in our IPS, we use the following five facts of Fact 1.

Fact 1:

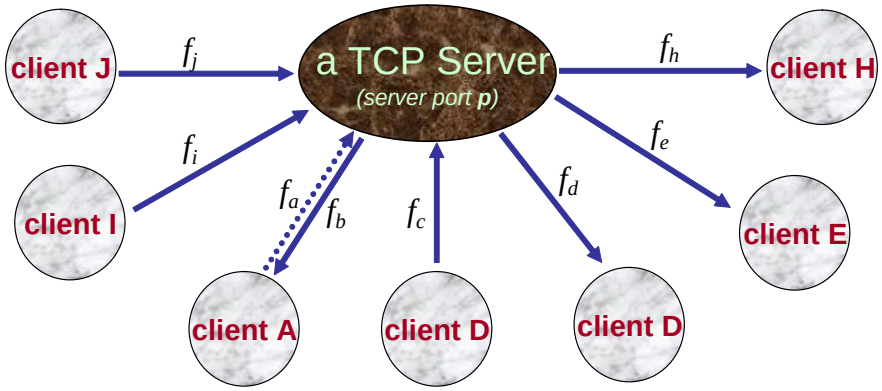
① If the important port of a TCP flow f_a is determined, then the important port of its reverse flow $_r f_a$ is also determined.

② If the important port of a TCP flow f_a is the source port, then the important port of a TCP flow f_b with the same source port and source IP address as those of f_a is the source port:
If $f_a(ip) = f_a(dport)$ and $f_a = f_b \mid (dip, dport, proto)$, then $f_b(ip) = f_b(dport)$

③ If the important port of a TCP flow f_a is the destination port, then the important port of a TCP flow f_b with same destination port and destination IP address as those of f_a is the destination port:
If $f_a(ip) = f_a(sport)$ and $f_a = f_b \mid (sip, sport, proto)$, then $f_b(ip) = f_b(sport)$

④ If a TCP flow f_a and a TCP flow f_b are different and both have the same source port and same source IP address, then the important ports of f_a and f_b are the source port:
If $f_a = f_b \mid (sip, sport, proto)$, then $f_a(ip) = f_a(sport)$ and $f_b(ip) = f_b(sport)$

⑤ If a TCP flow f_a and a TCP flow f_b are different and both have the same destination port and same destination IP address, then the important ports of f_a and f_b are the destination port:
If $f_a = f_b \mid (dip, dport, proto)$, then $f_a(ip) = f_a(dport)$ and $f_b(ip) = f_b(dport)$



the flow whose important port is determined is flow f_a only.
The import port number of f_a is p which is destination port of f_a.

Figure 16. An Example of Important Port Selection using Fact 1.

Figure 16 illustrates an example of the facts we found. A TCP server opens a server port number **p** and waits for client connection requests. Multiple clients can simultaneously connect to the server using server listening port **p**. Suppose that the important port number is determined only for the flow f_a and those of the other flows are not determined because of the three reasons mentioned above. Using the above five facts, we can determine the important port numbers of other flows. These facts will increase the success ratio of the IPS and we can effectively apply our method to real network environment.

By **Fact 1**-①, the important port of flow f_b can be determined because f_b is the reverse of f_a. By **Fact 1**-②, the important port of flow f_d can be determined, because f_d has the same source port and same source IP address as f_b does, and the important port of f_b is determined by the **Fact 1**-①. By **Fact 1**-③, the important port of flow f_c can be determined, because f_c has the same destination port and same destination IP address as f_a does. By **Fact 1**-④, the

important port of flow f_e and that of flow f_h can be determined, because f_e and f_h have the same source port and source IP address. By **Fact 1**-⑤, the important port of flow f_i and flow f_k can be determined, because f_i and f_k have the same destination port and destination IP address.

For the first problem stated above, we can use all of the five facts. For the second problem, we can use the **Fact 1**-④ and **Fact 1**-⑤. However, the **Fact 1**-④ and **Fact 1**-⑤ are not enough to determine the important port of all TCP flows. For example, in a case where a TCP server allows only a single connection from a specific client and closes the server port after finishing the communication, then we cannot apply the **Fact 1**-④ and **Fact 1**-⑤. Dynamically generated TCP sessions may be the main cause for this kind of situation; for example, a TCP session for file transfer of MSN messenger. However, it needs to be verified that how often this kind of single TCP session occurs in real network environment.

To realize the proposed IPS method, we construct a ***TCP Server Port Table***, which contains the information of server listening port. If a SYN or SYN-ACK packet is captured, the server listening port and server IP address is added to the TCP server port table. Using the ***TCP Server Port Table***, we can realize the **Fact 1**-①, **Fact 1**-②, and **Fact 1**-③. After applying the TCP server port table, we compare the remaining flows whose important port is not determined with the **Fact 1**-④ and the **Fact 1**-⑤.

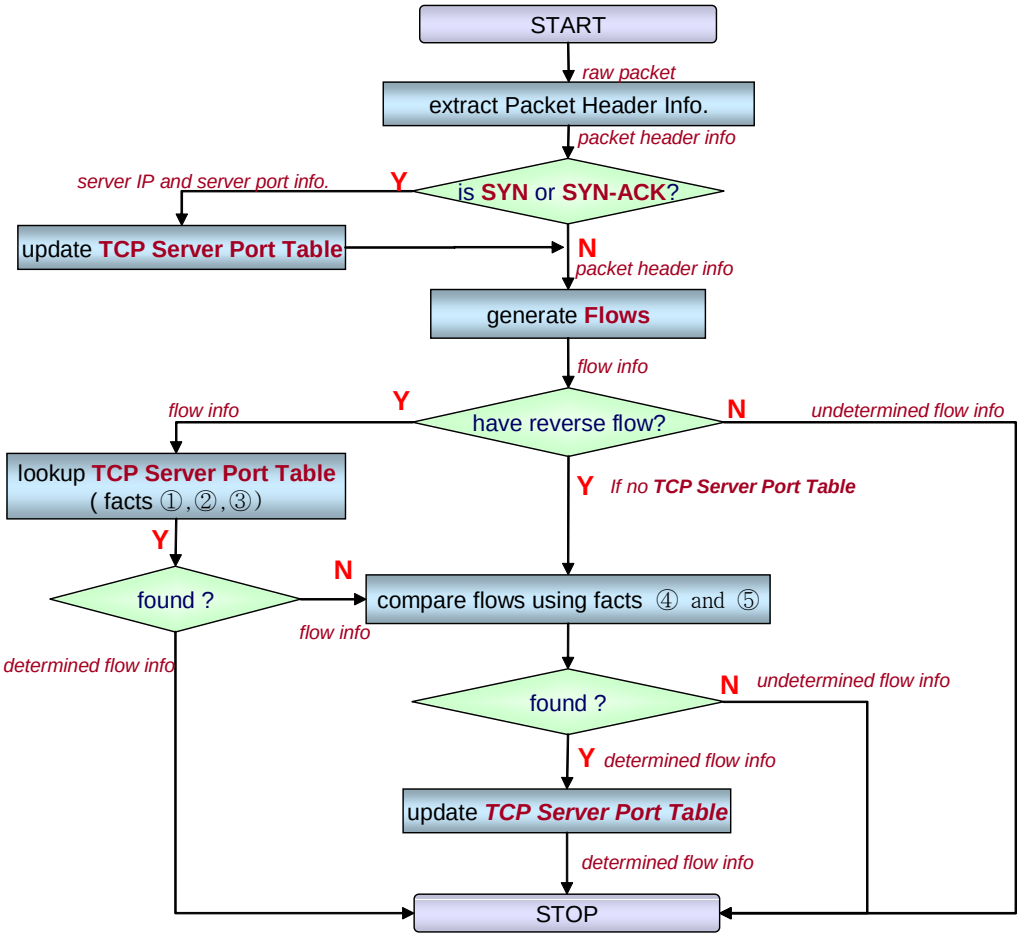


Figure 17. A Flow Chart Diagram for IPS

Figure 17 illustrates the overall algorithm for the important port selection method. The first step of the IPS algorithm is to find SYN and SYN-ACK packets. Moreover, the server port and server IP address are stored in the ***TCP Server Port Table***. The second step is to generate flow information from packet header information, which is extracted form each raw packet. The third step is to check whether the flow is a normal flow or not. If the flow is unidirectional (no reverse flow) with small number of packets and small flow size then we consider the flow as abnormal flow. For normal flows, we search the ***TCP Server Port Table*** to find

corresponding server port of each flow in the fourth step. In the fifth step, we apply the **Fact 1**-④ and the **Fact 1**-⑤ to any remaining flows.

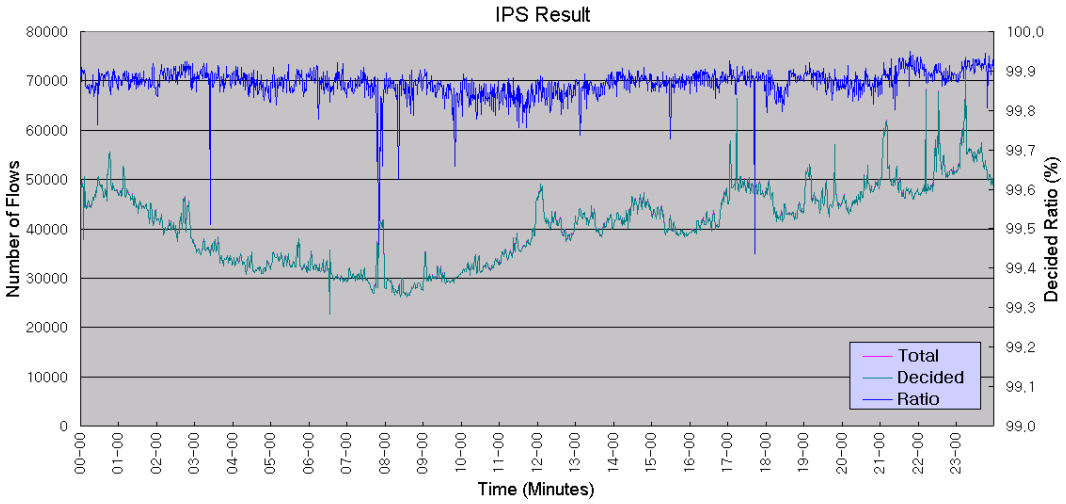


Figure 18. Important Port Determination Ratio (Feb. 1, 2004)

Using the combination of SYN-packet based IPS algorithm and the Fact 1, we could determine the important port of most TCP flows from POSTECH Internet Junction, as shown in Figure 18. The average number of TCP flows for one minute was 40,762. Among them, we could determine the important port of 99.9% of TCP flows. The remaining 0.1% of TCP flows was considered as abnormal traffic caused by DoS/DDoS attacks or Internet Worms. In the case of UDP flows, we could not apply this type of method because it does not use the 3-way handshaking mechanism like TCP does and both source port and destination port are important equally. Therefore, we used FRM algorithm directly to group UDP flows, as described next.

4.6 Flow Relationship Map

The Application Port Table (APT) and Important Port Selection (IPS) cannot identify Internet traffic with 100% accuracy. We may think of three reasons for

this. First, an enormous number of applications exist around the world to examine. In addition, complete investigation of all used port numbers for all existing applications may not be possible. Second, application traffic of dynamic sessions may not be identified, because the APT maintains only the fixed port numbers. Third, it may happen that two different applications use the same important port number. In this case, without additional information, we cannot correctly decide which application generates the flow. Therefore, we propose the third step to increase analysis accuracy: the Flow Relationship Map (FRM).

Using the Flow Relationship Map (FRM), we can group TCP and UDP flows according to the corresponding applications, and we can discover unidentified port numbers used by applications. In addition, the FRM can easily discover newly emerging applications in the future. Finally, we can increase the accuracy of application-layer traffic identification.

The main idea of the FRM reflects the fact that there exist dependencies among flows belonging to the same application. According to the dependencies, the FRM groups the individual flows into a number of sets. The flows belonging to a set are the flows generated from the same application. The FRM algorithm consists of two consecutive processes: Property Dependency Grouping (PDG) and Location Dependency Grouping (LDG).

4.6.1 Property Dependency Grouping

The Property Dependency Grouping (PDG) classifies individual flows according to the flow property dependencies. We define the property dependencies as the relationship between two individual flows belonging to the same applications in terms of protocols, port numbers, and IP addresses, which are described in Fact 2. The three facts in Fact 2 are the general property dependencies that are commonly applied to UDP and TCP flows. We use them to make the flows into groups in PDG.

Fact 2:

① If a flow f_a belongs to an application group $G(A_a)$, then the reverse flow, $_r f_a$, of the flow f_a also belongs to the same application group, $G(A_a)$:
If and only if $f_a \in G(A_a)$, then $_r f_a \in G(A_a)$

② If a flow f_a belongs to an application group $G(A_a)$ and a flow f_b has a conditional equality to f_a with ***sport***, ***sip***, and ***proto***, then the flow f_b also belongs to the same application group, $G(A_a)$:
If $f_a \in G(A_a)$ and $f_a = f_b \,|\, (sip, sport, proto)$, then $f_b \in G(A_a)$

③ If a flow f_a belongs to an application group $G(A_a)$ and a flow f_b has a conditional equality to f_a with ***dport***, ***dip***, and ***proto***, then f_b also belongs to the same application group, $G(A_a)$:
If $f_a \in G(A_a)$ and $f_a = f_b \,|\, (dip, dport, proto)$, then $f_b \in G(A_a)$

Figure 19 illustrates **Fact 2** as a diagram of flows. **Fact 2-**① is obvious like Figure 19-(a). Because a flow is unidirectional, the reverse flow of the original flow always exists and belongs to the same application. If two flows have the same source port number, source IP address, and protocol number, then two flows belong to the same application by **Fact 2-**②, as illustrated in Figure 19-(b). In addition, flows with the same destination port number, destination IP address, and protocol number belong to the same application by **Fact 1-**③ that is illustrated in Figure 19-(c).

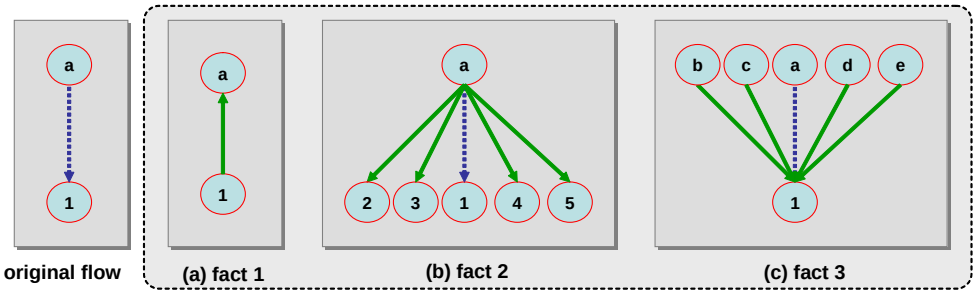


Figure 19. Flow Diagram for Fact 2

Fact 2 is obvious and provides accurate grouping of flows. The flows grouped by the PDG algorithm are generated by the same applications. The process of flow grouping by PDG is as follows. First, we select one flow among the captured flows. Second, we select the reverse flow of the selected flow. Third, we select the flows with conditional equality to the selected flow in ***sip***, ***sport***, and ***proto***. Fourth, we select the flows with conditional equality to the selected flow in ***dip***, ***dport***, and ***proto***. Fifth, we repeat the process from the second step to the fourth step for the flows selected in the third and fourth step. This recursive flow selecting process continues until no flow is selected any further. After finishing grouping flows from the first selected flow, we select another flow from the remaining flows and continue the same steps until no flow remains.

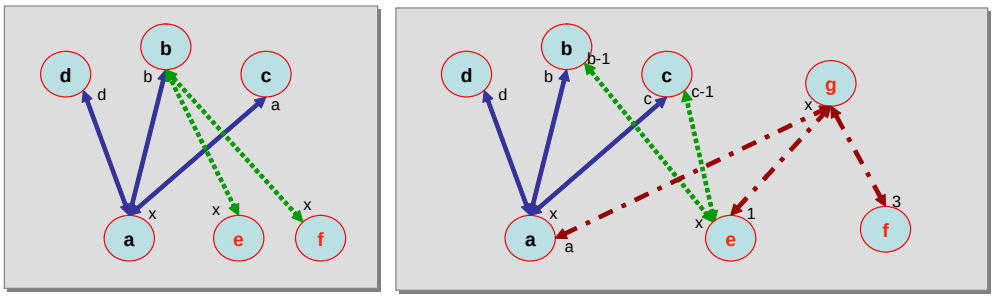
Pseudo algorithm for the proposed PDG is described in Figure 20. To remember the flows selected by the Fact 2 in the PDG process, we defined a stack in this pseudo algorithm. The stack temporarily stores the flows, which does not apply to the three steps of Fact 2 in the PDG process. We pop flows one by one from the stack and select flows by applying the Fact 2 for the popped flow from the captured flow pool. The selected flows are inserted to the stack for the next recursive step. This iterative algorithm using stack reduces the PDG grouping time comparing to any recursive algorithm.

```
1 Procedure FlowGroupingByPDG ( captured flow pool )
2 {
3    while ( any unselected flow exist in the captured flow pool )
4    {
5        prepare an empty flow group G for a new application;
6        select one flow f_a from the captured flow pool;
7        push the selected flow f_a in the stack;
8        while ( the stack is not empty )
9        {
10           pop one flow f_b form the stack;
11           insert the poped flow f_b into the group G;
12           select the reverse flow from the captured flow pool by the Fact 2-1 of the popped flow;
13           push the reverse flow in the stack;
14           select the flows from the captured flow pool by the Fact 2-2 of the popped flow;
15           push the selected flows in the stack;
16           select the flows from the captured flow pool by the Fact 2-3 of the popped flow;
17           push the selected flows in the stack;
18       }
19   }
20 }
```

Figure 20. Pseudo Algorithm for Property Dependency Grouping

We can apply this general PDG algorithm without any modifications to the UDP flows. Any applications using a UDP port ***p*** can send packets to multiple hosts and receive packets from multiple hosts using ***p*** simultaneously. Therefore, Fact 2-1, Fact 2-2, and Fact 2-3 can be used for any single UDP flow. The grouping of UDP flows is more extensive and effective. However, we cannot use all three facts of Fact 2 in TCP flows because of the difference between UDP flows and TCP flows that is illustrated in Figure 21. As illustrated in Figure 21-(b), the server application can communicate with multiple hosts with his opening TCP port ***x***. Client applications establish only two flows with corresponding TCP server host with a single TCP port; one is upward flow and another is corresponding reverse downward flow. Usually the client port is dynamically generated by the system where the client application is running. Even though many of the same application entities using TCP connections are widely running,

the PDG algorithm cannot group the corresponding TCP flows together since we cannot apply the three facts of Fact 2 for each TCP flow as it is. We can only use the two among the three facts of Fact 2: Fact 2-1 and one of Fact 2-2 and Fact 2-3.



(a) UDP flows **(b) TCP flows**

Figure 21. Difference between TCP and UDP flows

To overcome this problem of PDG algorithm, we use the result of the IPS algorithm, which is described in the previous Section. For all TCP flows, we can select the important port by IPS algorithm, which is the server listening port. To apply our proposed PDG algorithm, we ignore the client's dynamic port by setting it to NULL (0) from every TCP flow. Therefore, one side of all TCP flow has the port number 0 and the other side has non-zero port number, the server port. In addition, we modified the original PDG algorithm for these manipulated TCP flows as follows:

Fact 2':

① If a TCP flow f_a belongs to an application group $G(A_a)$, then the reverse TCP flow, ${}_r f_a$, of f_a also belongs to the same application group, $G(A_a)$: If and only if $f_a \in G(A_a)$, then ${}_r f_a \in G(A_a)$

② If a TCP flow f_a belongs to an application group $G(A_a)$, $f_a(sport) = 0$, and a TCP flow f_b has a conditional equality to f_a with ***sip***, ***sport***, ***dport***,

and ***proto***, then the TCP flow f_b also belongs to the same application group, $G(A_a)$:
If $f_a \in G(A_a)$, $f_a(sport) = 0$, and $f_a = f_b \mid (sip, sport, dport, proto)$,
then $f_b \in G(A_a)$

③ If a TCP flow f_a belongs to an application group $G(A_a)$, $f_a(sport) = 0$, and a TCP flow f_b has a conditional equality to f_a with ***sport***, ***dip***, ***dport***, and ***proto***, then f_b also belongs to the same application group, $G(A_a)$:
If $f_a \in G(A_a)$, $f_a(sport) = 0$, and $f_a = f_b \mid (sport, dip, dport, proto)$,
then $f_b \in G(A_a)$

④ If a TCP flow f_a belongs to an application group $G(A_a)$, $f_a(dport) = 0$, and a TCP flow f_b has a conditional equality to f_a with ***sip***, ***sport***, ***dport***, and ***proto***, then f_b also belongs to the same application group $G(A_a)$:
If $f_a \in G(A_a)$, $f_a(dport) = 0$, and $f_a = f_b \mid (sip, sport, dport, proto)$,
then $f_b \in G(A_a)$

⑤ If a TCP flow f_a belongs to an application group $G(A_a)$, $f_a(dport) = 0$, and a TCP flow f_b has a conditional equality to f_a with ***sport***, ***dip***, ***dport***, and ***proto***, then f_b also belongs to the same application group, $G(A_a)$:
If $f_a \in G(A_a)$, $f_a(dport) = 0$, and $f_a = f_b \mid (sport, dip, dport, proto)$,
then $f_b \in G(A_a)$

Fact 2'-1 is same as Fact 2-1. Fact 2'-2 and fact 2'-4 are the variation of the original Fact 2-1. In addition, Fact 2'-3 and Fact 2'-5 are the variations of the original Fact 2-2. Figure 22 illustrates the Fact 2' as a flow diagram.

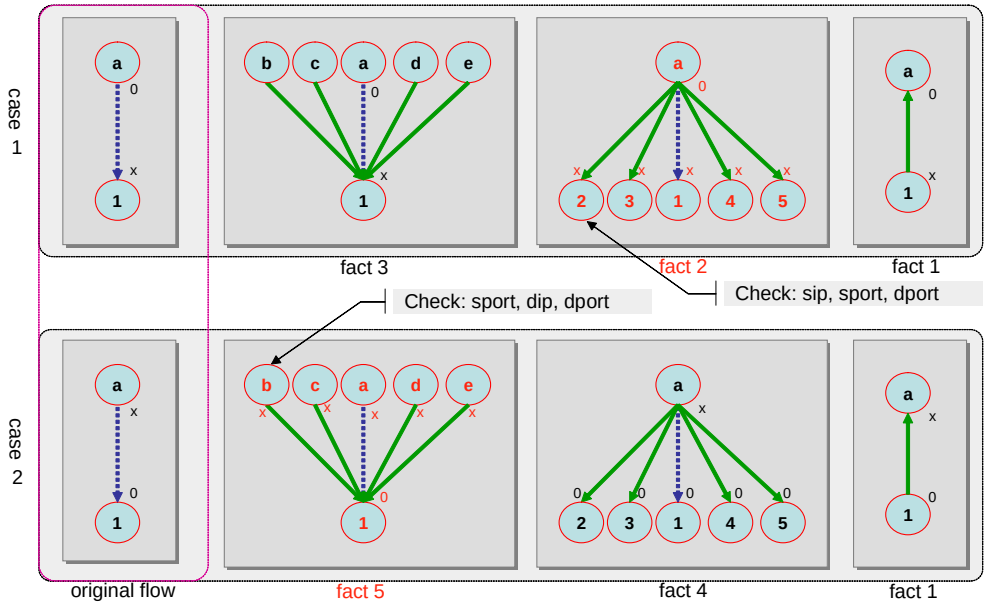


Figure 22. A Flow Diagram for the Fact 2'

Using the Fact 2', we can group the TCP flows more efficiently. By applying the IPS algorithm, the Fact 2'-2 and 2'-5 can be additional facts we can use to group TCP flows. For a TCP flow having its destination port same as the important port, a flow with the same destination port from the same host can be grouped by the Fact 2'-4. Moreover, for a TCP flow having its source port same as the important port, a flow with the same source port and the same destination IP address can be grouped by the Fact 2'-5. These two additional facts can group TCP flows more efficiently then the original Fact 2.

When we applied our proposed PDG algorithm to IP traffic in POSTECH Internet junction, we were able to reduce about 100,000 of concurrent flows into 500 of PDG groups.

4.6.2 Location Dependency Grouping

With the PDG algorithm, we can classify all TCP and UDP flows into a

number of PDG groups. The flows belonging to the same group have a higher probability that they have been originated from the same application. However, we cannot guarantee that the flows generated by a single application will be grouped into the same group by the PDG algorithm. Many applications, such as P2P applications, use multiple port numbers, and sometimes they use TCP and UDP protocols together. The problem occurs when the PDG algorithm is unable to group TCP flows and UDP flows into a single group. In addition, the PDG algorithm cannot group two TCP flows with different port numbers into a single group even though the same application generated them. This situation frequently occurs because many of current applications use multiple TCP port numbers or dynamically assigned TCP port numbers.

To solve this shortfall of the PDG algorithm, we have developed a method called Location Dependency Grouping (LDG). The main role of LDG is to connect the preliminary PDG groups according to their inter-dependencies, as illustrated in Figure 23. By the PDG algorithm, flows are classified into a number of groups. Then, by the LDG algorithm, the PDG groups generated by the same application are grouped into one single group.

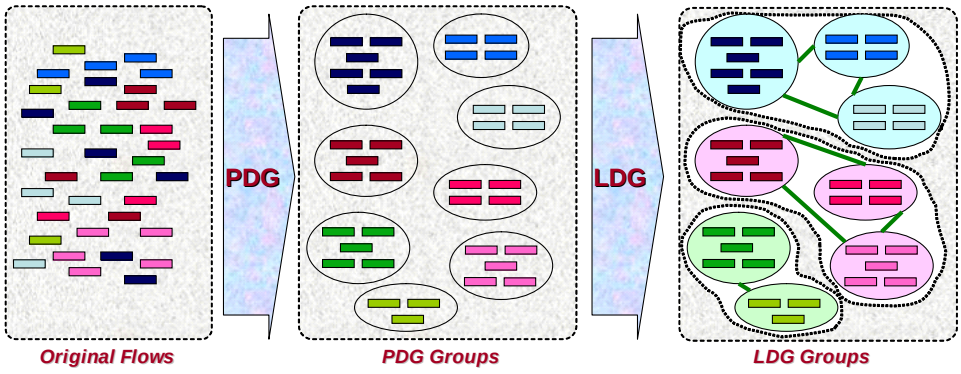


Figure 23. Flow Grouping with PDG and LDG

To combine inter-related PDG groups, we use the weight concept between PDG groups, as illustrated in Figure 24. We calculate the weight values of every

PDG group pair and we merge the PDG groups by their inter-relationship. If the weight value of two PDG groups is larger than a specified threshold value, then the two PDG groups are joined into a single group. For example, as illustrated in Figure 23, the eight different PDG groups are merged into the final three LDG groups by the LDG algorithm.

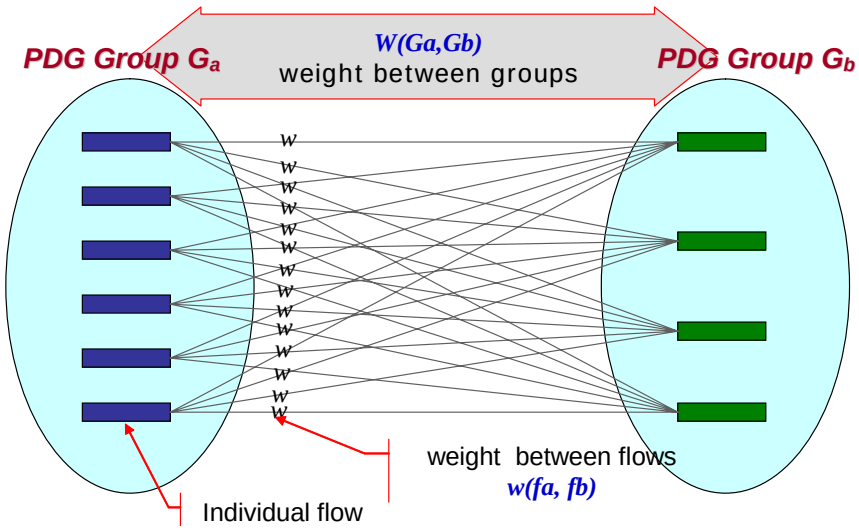


Figure 24. Weight between PDG Groups

To obtain the weight value $W(G_a, G_b)$ between every two PDG groups G_a and G_b, we define a weight value $w(f_a, f_b)$ between two flows f_a and f_b for a preliminary step. We do not consider the weight value between the two flows that belong to the same PDG group. The weight value $w(f_a, f_b)$ of two flows f_a and f_b is calculated by the following equation:

$$w(f_a, f_b) = \begin{cases} 100 & \text{if } (f_a(sip) = f_b(sip)) \text{ and } (f_a(dip) = f_b(dip)), \\ 100 & \text{if } (f_a(sip) = f_b(dip)) \text{ and } (f_a(dip) = f_b(sip)), \\ 10 & \text{if } (f_a(sip) = f_b(sip)) \text{ or } (f_a(dip) = f_b(dip)), \\ 10 & \text{if } (f_a(sip) = f_b(dip)) \text{ or } (f_a(dip) = f_b(sip)), \\ 0 & \text{otherwise,} \end{cases}$$

where $f_a \in G_a$ and $f_b \in G_b$ and $G_a \neq G_b$.

The weight value is 100 when the two flows are located between two different hosts; the two different flows between two hosts have a high possibility of being the same application traffic. The conditions for the value 100 cover the Type M-F-2 and Type M-D-2 cases of APT. The weight value is 10 when two flows are located among three different hosts. In this case, two flows have a small possibility of having been generated by a single application. The cases of value 10 cover the Type M-F-3 and Type M-D-3 of APT. Otherwise, the weight value between any two flows is 0. Using the weight value between two flows, we calculate the weight value between two different PDG groups as follows:

$$W(G_a, G_b) = \sum w(f_a, f_b) \text{ , where } f_a \in G_a \text{ and } f_b \in G_b \text{ and } G_a \neq G_b .$$

The weight value $W(G_a, G_b)$ of two PDG groups G_a and G_b is the sum of the weight values of each pair of flows that belong to two different PDG groups G_a and G_b. After calculating the weight value among the PDG groups, we merge them by the following rules:

$$\text{If } \max\left(\frac{W(G_a, G_b)}{n(G_a)}, \frac{W(G_a, G_b)}{n(G_b)}\right) \geq threshold, \text{ where } n(G_a) \text{ is the number of flows in } G_a,$$
$$\text{then } G_a \text{ and } G_b \text{ can be merged into a single group.}$$

Currently we are using 50 as a threshold value, which was determined by many experiments with IP traffic from the POSTECH Internet junction. To get suitable threshold value, we have setup a traffic source host where more than 10 Internet applications using dynamic ports are running and generating various traffic flows simultaneously. In addition, we have setup the proposed flow grouping system at the Internet junction. By applying various threshold values, we have found the best values which distinguished the flows from the traffic source host according to the corresponding applications. We may have to use a different

threshold value to apply the LDG algorithm to other network environments. However, this equation works efficiently in current network environment of POSTECH and may not be changed in different network environment. By the LDG algorithm, we can merge all PDG groups into LDG groups, as in Figure 23. When we applied our proposed algorithm to POSTECH Internet traffic, we could classify about 100,000 concurrent flows into 500 concurrent LDG groups on average, as illustrated in Figure 25.

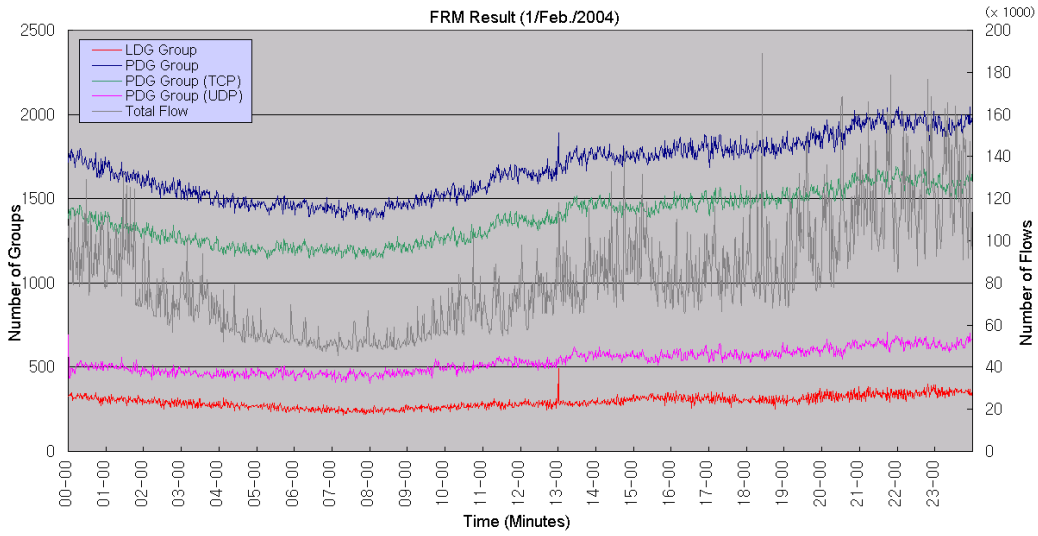


Figure 25. The Result of Flow Grouping Method

Considering the Figure 25, we calculated the number of Groups in each process of our proposed flow grouping method. The average number of flows for a minute during one day (Feb. 1, 2004) was 81,679. (Minimum was 45,344 and Maximum was 188,808). The average number of PDG groups after PDG grouping was 1,677. (UDP Flow Groups: 294, TCP Flow Groups: 1383). Finally, the average number of LDG groups after LDG grouping was 533. This mean that the 81,679 flows are generated less than 533 Internet applications.

During the LDG process, we use the information of APT to decide the application name of each flow. Before the LDG algorithm is applied to PDG

groups, we select PDG groups of Type S-F-2, Type M-F-2, and Type M-F-3 using APT information. The PDG groups of these types can easily be picked up, because they use fixed port numbers. For the remaining PDG groups, we apply our LDG algorithm. The flows in the remaining LDG groups are the flows of the Type M-D-2 and Type M-D-3. The flows in the selected LDG groups are tagged with the representative port numbers of corresponding applications. We can use the tagged port number to recognize the corresponding application name in the subsequent phases of traffic analysis.

The application name of all LDG groups may not be determined if the APT does not have a corresponding application name in the list. In this case, the flows in the undetermined LDG groups are marked with minus values. This makes it easy to find new applications because we have many hits in the undetermined LDG groups. The newly investigated application information is added to the APT, and from that time the undetermined LDG group can be determined and tagged with its corresponding representative port number. By the proposed flow-grouping algorithms, we can accurately identify Internet traffic at the application level and easily find new applications.

5. Design and Implementation of Application-Level Traffic Analysis System

In this chapter, we describe the design and implementation details of the application-level traffic identification system using the proposed method in the previous chapter. The application-level traffic identification system is implemented as embedded in the general NG-MON architecture. In addition, we deployed our NG-MON prototype system in our campus Internet junction and analyzed the Internet traffic generated from our campus network. All the details of this analysis are described in this chapter.

5.1 Design of Application-Level Traffic Identification System

Figure 26 illustrates the overall design of the application-level traffic identification system (ATIS), which consists of three main modules; Important Port Selector (IPS) module, Application Port Table (APT) module, and Flow Relations Mapper (FRM) module.

The IPS module consists of a Packet Capturer, a Flow Generator, and a TCP Server Port Table. The Packet Capturer receives raw packets from a network link and extracts packet header information from each raw packet and the packet header information is sent to the Flow Generator. If a packet is a SYN or SYN-ACK packet, it is stored in the TCP Server Port Table. The SYN Packet Table keeps the TCP listening port information. To select the important port number from each flow, the Flow Generator looks up the TCP Server Port Table. We used hash tables to store flow information and server listening port information in the Flow Generator to improve the search operation.

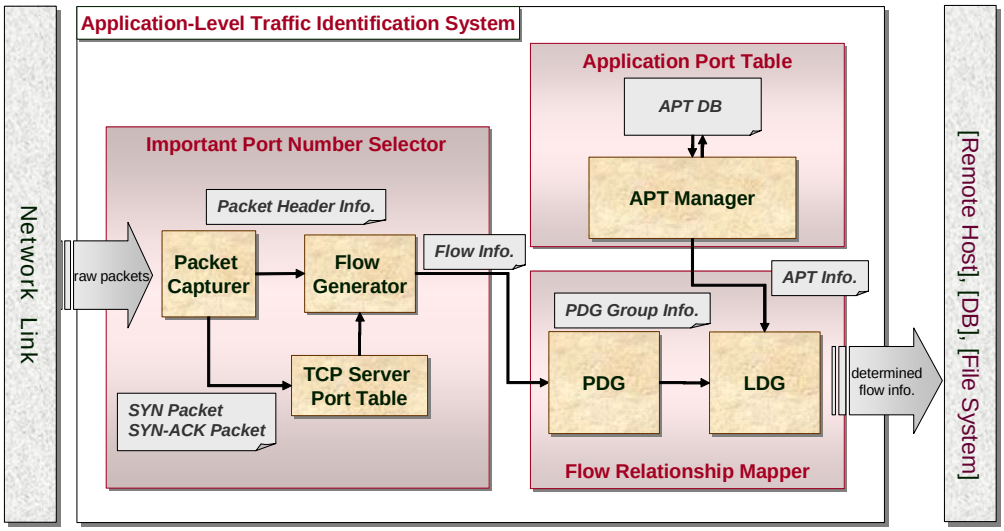


Figure 26. Overall Architecture of Application-Level Traffic Analysis System

The IPS module can be implemented in a single system or multiple systems depending on the network links we monitor. When the capacity of a single system is sufficient to handle the entire raw packets, the IPS module can be implemented in a single system. However, multiple capture systems are necessary occasionally; for example, when the target link speed is high or when we have a large number of target links to be captured. In that case, the IPS module should be separated into two levels: the first-level IPS and the second-level IPS, as illustrated in the Figure 27-(b). While the first-level IPS's are only responsible for their assigned links, the second-level IPS collects the results of first-level IPS's and merges them.

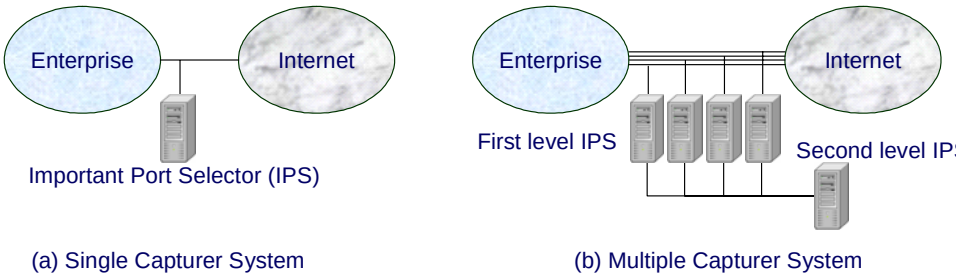


Figure 27. Variance of IPS Module depending on the Target Links

The important port determined that flows are sent from the IPS module to the Flow Relationship Mapper (FRM) module. The PDG module of FRM first receives the flows, and classifies them into a number of PDG groups using the proposed PDG algorithm. The next LDG module merges the incoming PDG groups using APT information and the proposed LDG algorithm. The result of the LDG module may be stored in a file system or a database, or sent to the other analysis system for subsequent analysis.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<apt-config>
        <app appName="MSN Messenger" repPort="1863" type="M-D-3" class="p2p">
                <session protocol="tcp" port="1863" />
        </app> <!-- MSN Messenger -->

        <app appName="WWW" repPort="80" type="S-F-2" class="traditional">
                <session protocol="tcp" port="80" />
                <session protocol="tcp" port="8080" />
                <session protocol="tcp" port="443" />
        </app> <!-- WWW -->

        <app appName="WMedia" repPort="1755" type="M-D-2" class="streaming">
                <session protocol="tcp" port="1755" />
        </app> <!-- Windows Media Streaming -->

        <app appName="FTP" repPort="21" type="M-D-2" class="traditional">
                <session protocol="tcp" port="21" />
        </app> <!-- FTP Service -->
</apt-config>
```

Figure 28. An Example of Application Port Configuration File

Through the off-line search of each application, we built an application port configuration file with XML. We used XML because it is easy to use and many XML-related libraries are provided in various languages such as C/C++ and Java. Figure 28 shows an example of an application port configuration file. The APT manager reads this configuration file and retains port group information. When a LDG module receives PDG groups from the PDG module, it looks up APT to decide the corresponding application name of each LDG group. Finally, the LDG module determines the application name of flows by tagging the flows with the corresponding representative port number.

5.2 Integration of ATIS with NG-MON

As mentioned previously, the application-level traffic analysis module is implemented as a plug-in module to NG-MON. Figure 29 illustrates the integration of the application-level traffic analysis module with the current NG-MON system.

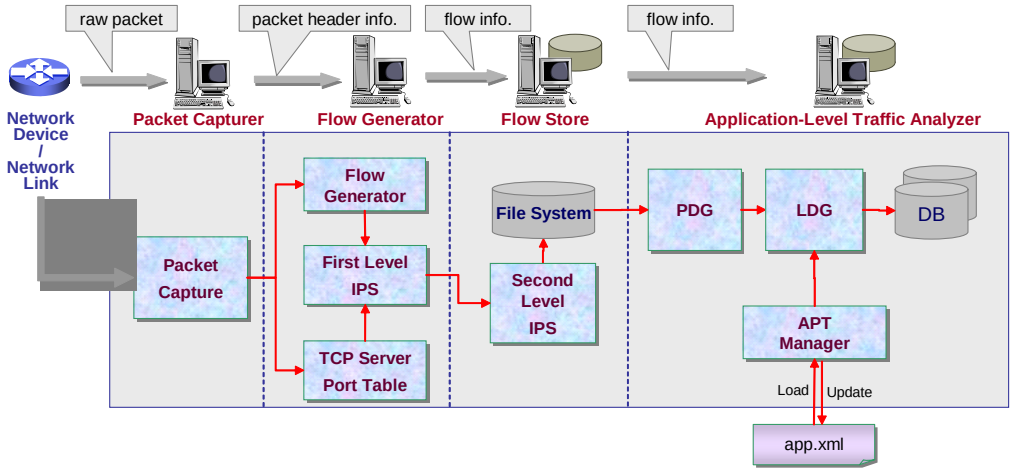


Figure 29. Integration of ATIS with NG-MON

The components of IPS module are separated into the Packet Capturer, Flow Generator, and Flow Store phases. We use the Packet Capturer module and Flow Generator module of NG-MON without any modification. The TCP Server Port Table and first-level IPS module are added to the Flow Generator, and the second-level IPS module is added to the Flow Store, as described in Figure 29. The important port number determined that flows are stored as a raw file format in the Flow Store. For the traffic analysis at the application-level, we developed a specialized analyzer where the APT manager, FRM module is running. The Analyzer receives the flow data from the Flow Store using the same approach as the other analyzers do. The result of the application-level traffic analyzer and the tagged flow information with the representative port number are stored in the Database, which may be used by the Presenter or other analyzers. The presenter and other analyzer determine the corresponding application name of each flow by

the representative port number. We integrated the additional IPS components to the existing NG-MON architecture without changing the philosophy of NG-MON: scalability and extendibility.

We implemented the application-level traffic analysis system using C language, libpcap library, and xml libraries in a Linux environment. To store the analysis result, we used MySQL Database because it is quick in data storing and retrieving among several freely available Database systems. We designed the packet capture system with multi-threaded architecture to handle multiple NICs in a single capture system. We used a semaphore for each capture thread to send the captured data to the exporting module without conflict. We designed the communication protocols between each phases over TCP. We used TCP rather than UDP in order to eliminate the possibility of data loss in the data delivery between phases. We used a number of hash tables to effectively perform the PDG and LDG algorithms and reduce their processing time.

5.3 Implementation of NG-MON with ATIS

We implemented the NG-MON on Linux Operating Systems. We used RedHat Linux 7.3 or higher, which use Linux Kernel version 2.4.18 or higher, as illustrated in Table 8.

As a packet capturing module, the standard pcap library with BPF (Berkeley Packet Filter) [41] was used. The performance of standard libpcap module was tested to capture as much as 50,000 packets per seconds in our experimentation. The CPU load of packet capture is determined by the byte size to be copied from Kernel to an analysis application and the number of packets to be captured. All the remaining processing modules of Packet Capture phase are implemented using C languages. We selected C languages because of its high performance and its ability to program low-level. For the communication between the Flow Generator phase and the Packet Capture phase, we used the standard socket functions, especially TCP sockets.

In regarding to the Flow Generator, we implemented it using C language. To reduce processing time of flow data generation, we used open hashing scheme with 0x100 numbers of buckets. The maximum number of inserted items in a bucket during the specified exporting interval (we are using 1 minute as static exporting interval) was less than 10 with more than 10,0000 numbers of total flow records. We use the 5-tuple packet header values (source IP, source port, destination IP, destination port, and protocol number) for the generation of hashing key. To export the flow data from the Flow Generator to the Flow Store, we defined our own protocols over TCP. We used TCP because of its reliable data delivery mechanism; an absolute reliability in data delivery is more important than fast delivery from the Flow Generator to the Flow Store in our case.

Phase	Packet Capture	Flow Generator	Flow Store	Analyzer	Presenter
Development Tool	xml library pcap library C language	xml library C language	xml library C language	xml library C language MySQL	PHP jpgraph library Apache Server
OS	Redhat Linux 7.3/8.0/9.0				

Table 8. Hardware and Software Specification for NG-MON

We implemented the Flow Store phase using C language. The flow data is stored in the local file system as a binary format. The flow data is stored with different file name every minute (e.g. flow_2004_03_15_09_00). Therefore, we can distinguish what time flow data file has been created. Each flow record is 48 bytes long. The average file size of flow data is about 6Mbytes in our network environments. The Flow Store keeps the binary flow data for a short period of time. Currently, we are using one hour as our store time period. After an hour from the time a flow data has been created, the Flow Store deletes the flow data. By this storing mechanism, the Flow Store requires small size of disk space (In our case, we needed less than 500Mbytes). Before a flow data file is deleted, the

analyzer should query the flow data to see if it is required.

We implemented the Traffic analyzer using C language. The analysis results are stored in MySQL DB. The analyzer queries the flow data file at the start of every minute and analyzes the file according to some predefined rules. The analyzer should finish each analysis task within a minute, because NG-MON is a real-time traffic analysis system. In our implementation of NG-MON, we developed three kinds of analyzer: host analyzer, security analyzer, and protocol analyzer. All these three analyzers are running in different systems and doing assigned analysis tasks. MySQL DB is running in the same analysis system.

The Presenter module was implemented using PHP and jpgraph php library to provide the analysis results in the form of Web pages. We used Apache server as the front-end web server. Therefore, the user can access the NG-MON system from any computer connected to the Internet and get the latest analysis results. Whenever a user browses a NG-MON webpage, the Presenter queries the MySQL DB and organizes the data into fine-looking reports. The jpgraph library generates various graph data, such as a pie-chart showing the distribution of transport layer protocols or a time-series graph showing the variation of total flows, total packets, and total bytes.

5.4 Deployment of NG-MON

We have deployed the NG-MON with the application-level traffic analysis module in the Internet junction of our campus network. Our campus Internet link is composed of two 100 Mbps Metro Ethernet. There are two core-switches and two Internet routers connected with six 1-Gbps Ethernet links in a full mash structure, as shown in Figure 30. The maximum utilization of a single Gbps Ethernet link is about 200Mbps, and the maximum utilization of the four Gbps Ethernet links is about 300Mbps. The average utilization of a single Gbps Ethernet link and four Gbps Ethernet link are 50Mbp and 200Mbps, respectively. The under-utilization of these Gbps Ethernet links are caused by the two

100Mbps Metro Ethernet links on the Internet side.

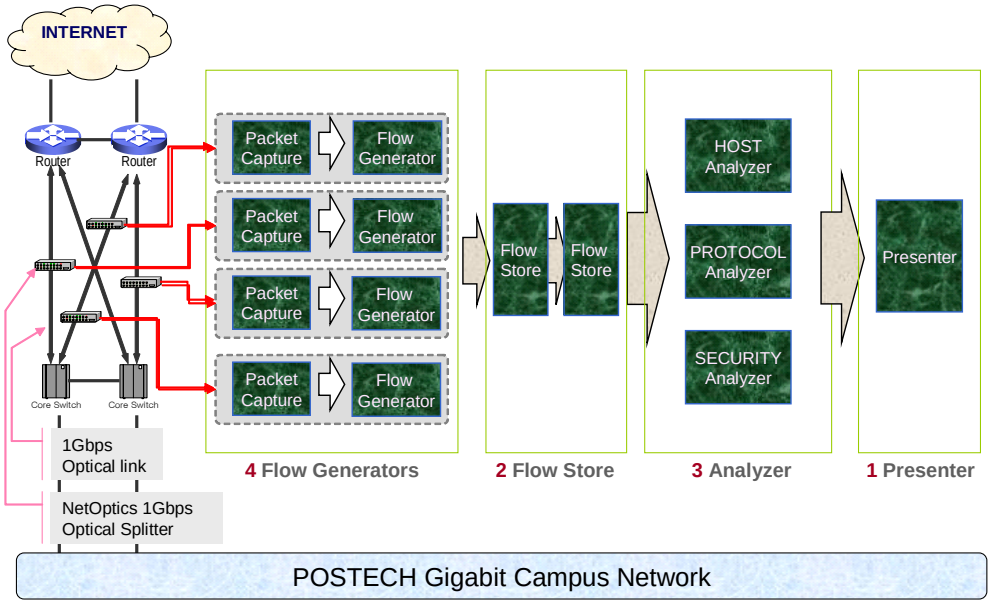


Figure 30. Internet Connection Structure of POSTECH

We set up four optical taps at the four 1-Gbps Ethernet links between core-switches and Internet routers. Through these four optical taps, we could copy all in/out Internet traffic flowing between our campus network and Internet.

Phase		Packet Capture	Flow Generator	Flow Store	Analyzer	Presenter
Hardware System	CPU	Xeon 2.4 GHz 2 CPUs		Pentium-III 800 MHz	Pentium-4 2.6 GHz	Pentium-III 800 MHz
	Memory	1 Gbytes		256 Mbytes	1 Gbytes	256 Mbytes
	NIC	2-1000 Mbps Ethernet		2-100 Mbps Ethernet	2-100 Mbps Ethernet	2-100 Mbps Ethernet
	Hard Disk	80 GBbytes		20 GBbytes	120 GBbytes	20 GBbytes
Number of Systems		4		2	3	1

Table 9. Hardware Specification of NG-MON

To monitor the Internet traffic generated from our campus network, we used

four Flow Generators, two Flow Stores, Three Analyzers, and one Presenter: total 10 Linux machines, as illustrated in Figure 30. The hardware specifications of these 10 Linux machines are described in Table 9.

Each Flow Generator systems have two 1-Gbps optical Ethernet NIC to capture both directions of traffic flows from each 1-Gbps Ethernet link between two core switches and two Internet routers. We put the Packet Capture module and the Flow Generator module into a single system with Xeon 2.4GHz 2 CPUs. Because the average utilization of each target link is low enough to be handled by the specified system. As the Flow Store system and the Presenter, we used two general-purpose computer systems with Pentium-III 800 MHz one CPU and 20 Gbytes of hard disk. We used the Pentium-4 2.6GHz single CPU machine with 120Gbytes hard disk as a Traffic Analyzer. Each Traffic Analyzer handles about 100,000 flows of traffic data every minute. With this hardware specification, the processing time of each Traffic Analyzer takes about 20 seconds. The hardware specification illustrated in Table 9 is enough to cope with the amount of traffic data generated from our campus network.

Three Analyzers (Host Analyzer, Security Analyzer, and Protocol Analyzer) are running on three different machines which are composed of the same hardware specifications. The Host Analyzer analyzes the bandwidth usage of each host. The Security Analyzer detects any abnormal flows in the network, which is generated from DoS/DDoS attack, Internet Worm, or Scanning. The Protocol Analyzer performs the task of application-level traffic identification. Moreover, we can easily add a new analyzer to this flexible NG-MON architecture.

5.5 NG-MON Presenter

The NG-MON Presenter provides the real-time analysis results of IP traffic from these three different analyzers to the network administrators via Web user interface. The Presenter also consists of three categories - Host view, Security view, and Protocol view - according to the three different analyzers.

NG-MON captures all the in/out Internet traffic and analyzes them from various points of view. The analysis granularity of current NG-MON is one minute, although it can be configured to other values. During each minute, each phase of NG-MON performs its assigned task. The application-level traffic analyzer also works with one-minute granularities; the analysis result is stored in the DB every minute.

We can notice the result of application-level traffic analysis in the protocol view pages of the NG-MON Presenter. Figure 31 shows an example of the application-level traffic analysis results.

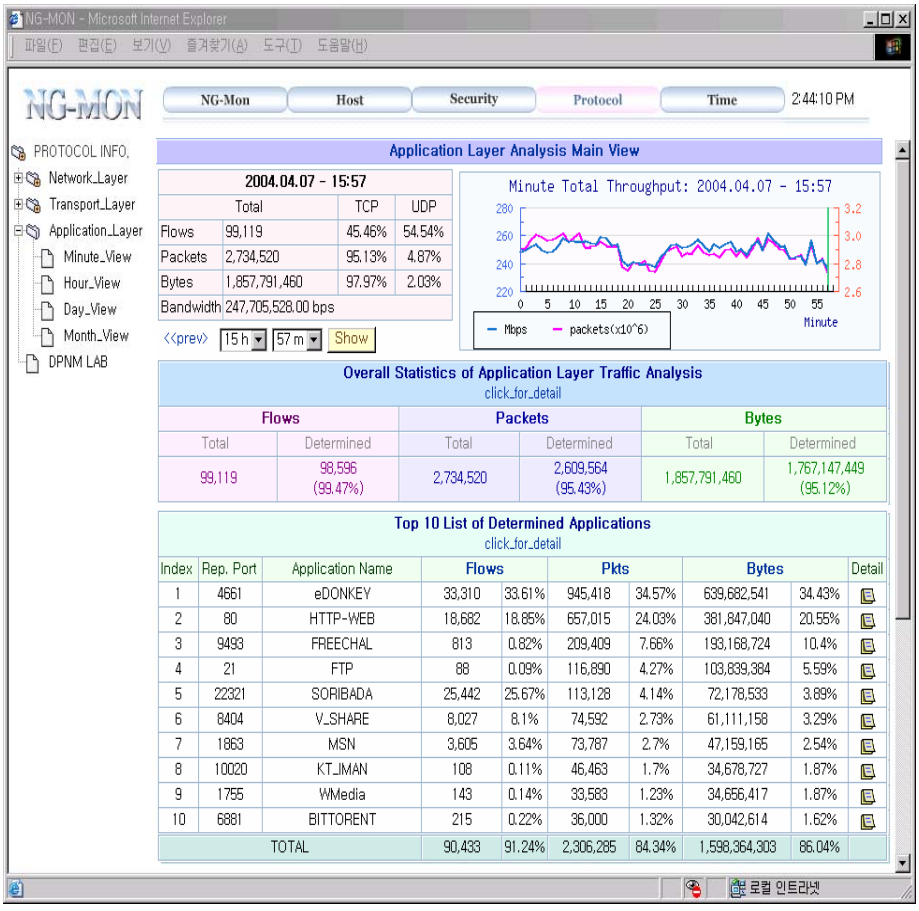
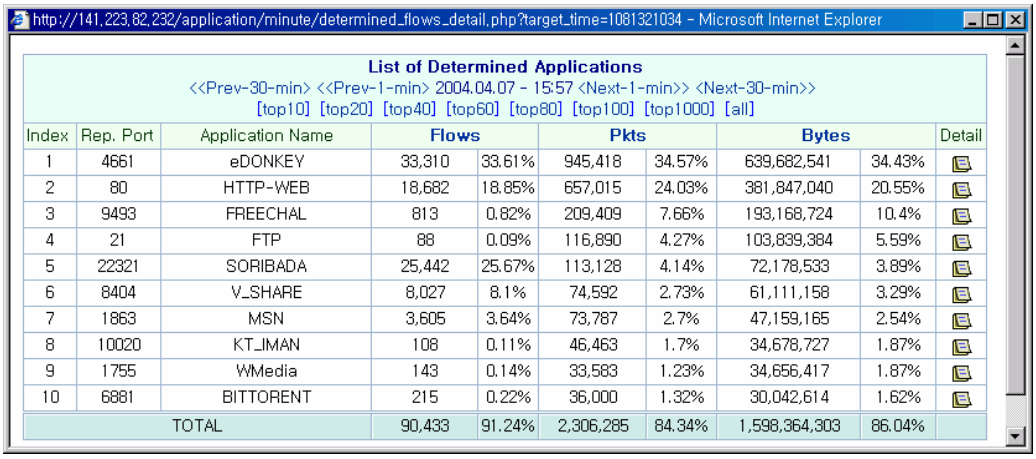


Figure 31. Application Protocol View Page of NG-MON

Figure 31 is the front page of the application-layer traffic analysis. From the left menu, we can select 4 different analysis time scales: minute, hour, day or month. The minimum analysis time granularity is minute, which can be changed. The hour data is a summary of minute analysis results, and the day data is a summary of hour analysis results. In the same manner, the month data is a summary of day analysis results.

Inside the main result page, the upper left table shows the summary of analyzed traffic data at the specified time scale. The upper-right time-series graph shows the variation of bandwidth and number of packets during a specified time interval. We can see the proportion of determined traffic at specified minutes in terms of bytes, packets, and flows in the middle table. We can see the top 10 list of application traffic and the top 10 list of undetermined flow groups. The bottom table shows the top-10 list of determined flows according to the application name. In addition, the undermined flows are listed on the front page of application-level traffic analysis.



List of Determined Applications

<<Prev-30-min> <<Prev-1-min> 2004.04.07 - 15:57 <Next-1-min>> <Next-30-min>>

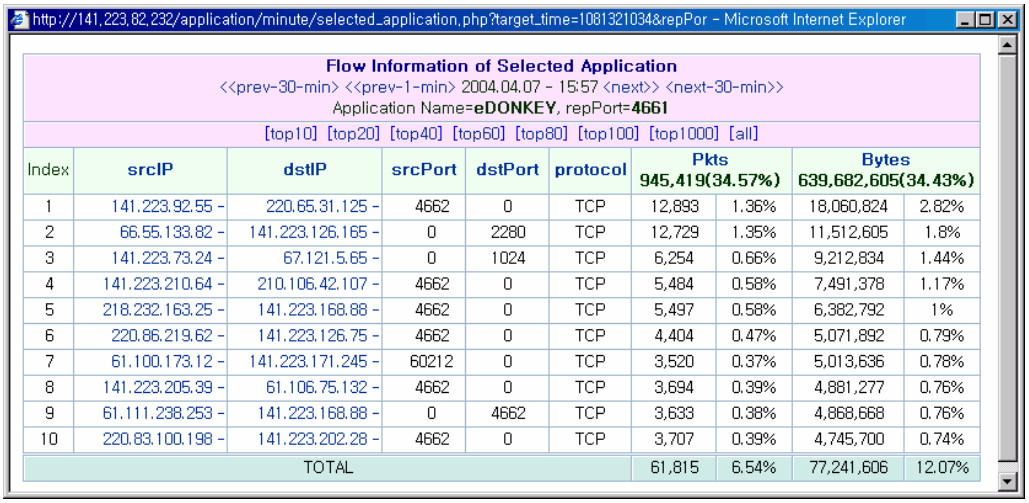
[top10] [top20] [top40] [top60] [top80] [top100] [top1000] [all]

Index	Rep. Port	Application Name	Flows		Pkts		Bytes		Detail
1	4661	eDONKEY	33,310	33.61%	945,418	34.57%	639,682,541	34.43%	
2	80	HTTP-WEB	18,682	18.85%	657,015	24.03%	381,847,040	20.55%	
3	9493	FREECHAL	813	0.82%	209,409	7.66%	193,168,724	10.4%	
4	21	FTP	88	0.09%	116,890	4.27%	103,839,384	5.59%	
5	22321	SORIBADA	25,442	25.67%	113,128	4.14%	72,178,533	3.89%	
6	8404	V_SHARE	8,027	8.1%	74,592	2.73%	61,111,158	3.29%	
7	1863	MSN	3,605	3.64%	73,787	2.7%	47,159,165	2.54%	
8	10020	KT_IMAN	108	0.11%	46,463	1.7%	34,678,727	1.87%	
9	1755	WMedia	143	0.14%	33,583	1.23%	34,656,417	1.87%	
10	6881	BITTORENT	215	0.22%	36,000	1.32%	30,042,614	1.62%	
TOTAL			90,433	91.24%	2,306,285	84.34%	1,598,364,303	86.04%	

Figure 32. Top 10 list of determined Applications

Figure 32 shows the sorted list of determined application traffic with the proportion of them in flows, packets, and bytes. The top-10 list based on flows, packets, and bytes could be different. In this page, we can understand how many

flows, packets, and bytes have been generated by a specified application and their proportions.



Flow Information of Selected Application

<<prev-30-min> <<prev-1-min> 2004.04.07 - 15:57 <next>> <next-30-min>>>

Application Name=**eDONKEY**, repPort=**4661**

[top10] [top20] [top40] [top60] [top80] [top100] [top1000] [all]

Index	srcIP	dstIP	srcPort	dstPort	protocol	Pkts 945,419(34.57%)		Bytes 639,682,605(34.43%)	
1	141.223.92.55 -	220.65.31.125 -	4662	0	TCP	12,893	1.36%	18,060,824	2.82%
2	66.55.133.82 -	141.223.126.165 -	0	2280	TCP	12,729	1.35%	11,512,605	1.8%
3	141.223.73.24 -	67.121.5.65 -	0	1024	TCP	6,254	0.66%	9,212,834	1.44%
4	141.223.210.64 -	210.106.42.107 -	4662	0	TCP	5,484	0.58%	7,491,378	1.17%
5	218.232.163.25 -	141.223.168.88 -	4662	0	TCP	5,497	0.58%	6,382,792	1%
6	220.86.219.62 -	141.223.126.75 -	4662	0	TCP	4,404	0.47%	5,071,892	0.79%
7	61.100.173.12 -	141.223.171.245 -	60212	0	TCP	3,520	0.37%	5,013,636	0.78%
8	141.223.205.39 -	61.106.75.132 -	4662	0	TCP	3,694	0.39%	4,881,277	0.76%
9	61.111.238.253 -	141.223.168.88 -	0	4662	TCP	3,633	0.38%	4,868,668	0.76%
10	220.83.100.198 -	141.223.202.28 -	4662	0	TCP	3,707	0.39%	4,745,700	0.74%
TOTAL						61,815	6.54%	77,241,606	12.07%

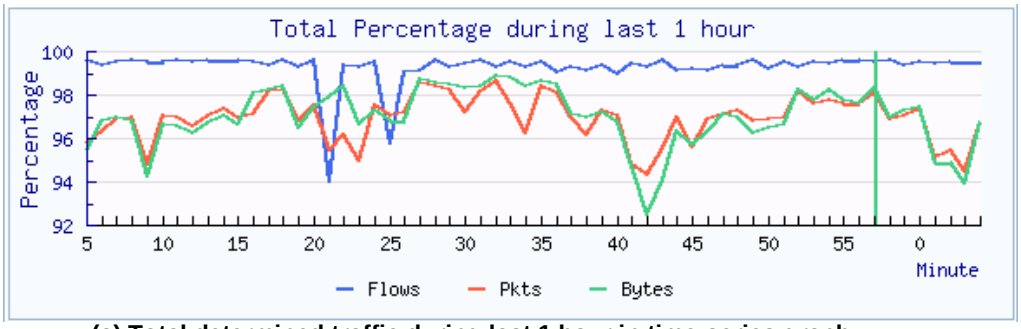
Figure 33. Flow-level details of a Selected Application

Figure 33 shows the flow-level details of specific application traffic. From this page, we can see the port numbers and transport protocol type used by a specific application. Furthermore, we can see which hosts are involved in the generation of specific application traffic.

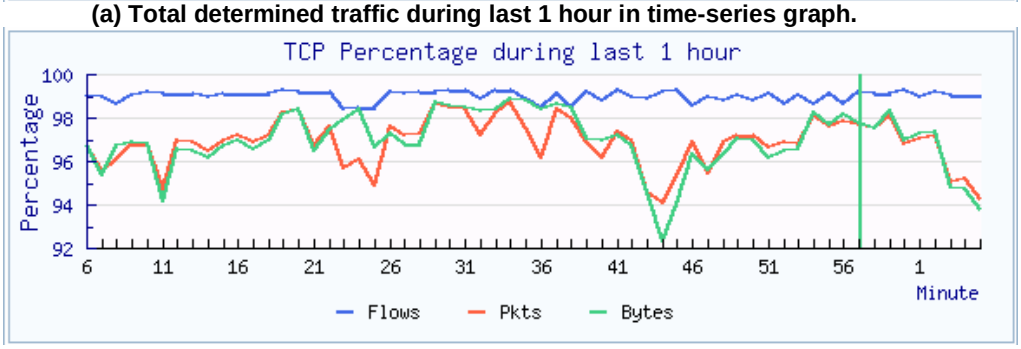
The graphs in Figure 34 are three time-series graphs showing the proportion of determined traffic in flows, packets, and bytes during a specified one-hour period. Figure 34-(a) shows the proportion of total determined flows, which indicates that more than 98% of flows are determined while 96% of packets and bytes, on average, are determined. The determined ratio of flows is always higher than that of packets and bytes. Figure 34-(b) and Figure 34-(c) shows the proportion of determined TCP and UDP traffic, respectively. As Figure 34-(b) and Figure 34-(c) illustrates, the determined ratio of UDP traffic is higher than TCP traffic.

The undetermined flow information is also presented in our application-layer protocol view pages, which helps us to investigate unknown applications in off-

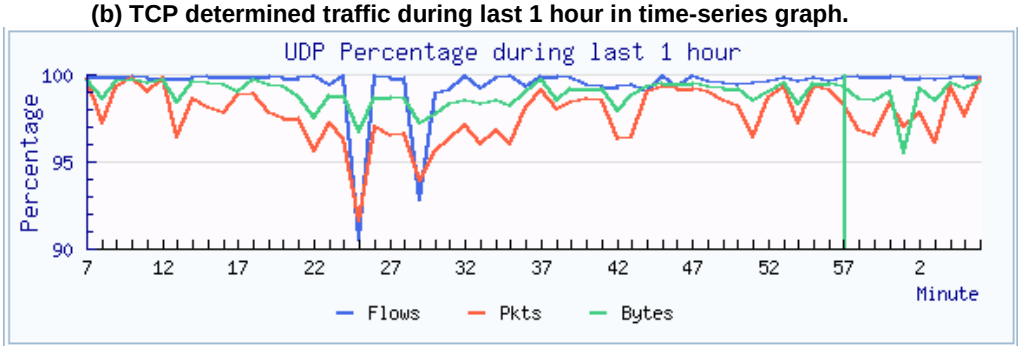
line. Our system aggregates the undetermined flows into several groups using the flow correlation information. Based on the flow data in a specified undetermined group, we can easily find newly emerging application. After adding the representative port of the newly detected application into APT table, the new application flows are analyzed.



(a) Total determined traffic during last 1 hour in time-series graph.



(b) TCP determined traffic during last 1 hour in time-series graph.



(c) UDP determined traffic during last 1 hour in time-series graph

Figure 34. The Proportion of Determined Traffic

6. Characteristic Analysis of IP traffic

In this chapter, we describe the analysis results of recent IP traffic. We collected the IP traffic using NG-MON from POSTECH Internet junction and determined the application name of each traffic flow using the proposed flow grouping method. We analyzed the captured traffic traces from the various perspectives and cauterized the recent IP traffic in application-layer.

6.1 Collection of IP Traffic Trace

In order to collect IP traffic data, we used the NG-MON Flow Store, which is deployed on the Internet junction of POSTECH, as illustrated in Figure 35. NG-MON is a real-time traffic monitoring and analysis system for high-speed network links, which is developed by our research group from 2002. The role of NG-MON Flow Store is to receive the flow data from Flow Generators, store them for some time, and provide them to any kinds of Traffic Analyzer.

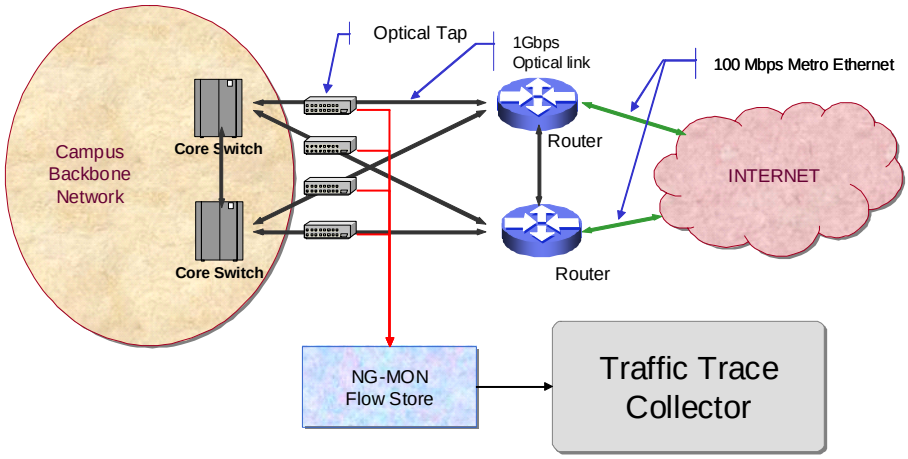


Figure 35. Traffic Trace Collection Method

To collect the IP traffic trace we developed a new traffic trace collector

system, whose function is to retrieve the flow data from the NG-MON Flow Store and keep the entire flow data for future off-line traffic analysis. For the traffic trace collector system, we used a Linux system with Pentium-II 400MHz single CPU, 640Mbyes memory, and four 120Gbytes hard disk (total 480G bytes).

The POSTECH campus Internet link is composed of two 100 Mbps Metro Ethernet links. The average link utilization is about 200Mbps, considering the two inbound and outbound traffic. We collected the flow data which is stored in the NG-MON Flow Store, not raw packet data or packet header data. The amount of flow data collected during a week was about 50 Gbytes in our campus network.

We have about 6,000 computers connected to the POSTECH campus network. Microsoft Windows Operating Systems (Windows 98/ME/XP/NT/2000/2003) are installed in 80% of computers, and remaining 20% of computers are running on the Unix-like operating systems, such as Linux, Sun Solaris, IBM AIX, HP-UX, etc. In addition, all of our 2,800 students live in campus dormitories.

Location		Internet Junction of POSTECH Campus Network								
Collection Period		2/1/'04 – 2/7/'04			2/17/04 – 2/23/04			3/6/'04 – 3/12/'04		
Total File Size		41 Gbytes			43 Gbytes			53 Gbytes		
		Flows (x 10^6)	Packets (x 10^6)	Bytes (GB)	Flows (x 10^6)	Packets (x 10^6)	Bytes (GB)	Flows (x 10^6)	Packets (x 10^6)	Bytes (GB)
Total	TCP	295 (34%)	18,345 (93%)	13,697 (98%)	325 (35%)	19,246 (92%)	13,619 (97%)	206 (24%)	21,015 (91%)	14,321 (97%)
	UPD	537 (62%)	1,089 (5%)	190 (1%)	543 (59%)	1,381 (6%)	327 (2%)	591 (70%)	1,564 (6%)	341 (2%)
	ICMP	33 (3%)	190 (0%)	16 (0%)	39 (4%)	177 (0%)	15 (0%)	38 (4%)	335 (1%)	33 (0%)
	Others	0.1 (0%)	1 (0%)	0.6 (0%)	0.1 (0%)	1 (0%)	0.2 (0%)	0.004 (0%)	2 (0%)	0.5 (0%)
	Total	866 (100%)	19,636 (100%)	13,905 (100%)	908 (100%)	20,806 (100%)	13,962 (100%)	836 (100%)	29,917 (100%)	14,697 (100%)

Table 10. Traffic Trace Summary

We collected IP traffic for three weeks during two months (Feb. and Mar.) in 2004. The overall summary of our traffic trace is illustrated in Table 10. The total number of flows captured during three weeks was $2{,}610 \times 10^6$, and the total bytes are over 40 TB. Among them, we considered only TCP and UPD traffic in the

analysis categories, which occupies more than 99% of total traffic in bytes. From the captured traffic data, we excluded the packets which had spoofed information in the packet header values. The portion in this category was 0.5% in bytes and 3.3% in packets of total traffic.

The average bytes per packet were calculated as 642 bytes from Table 10. The average bytes per TCP packet was 678 bytes, which was greater than that of UDP traffic (239 bytes). The average packet count of flow was 28 (average TCP and UDP packet counts of flow were 98 and 3, respectively). We assume that a large number of UDP flows are composed of less than 2 packets. The average bytes per flow were 18,239 bytes. (Average TCP and UDP bytes per flow were 67,043 and 756 bytes, respectively). This result well describes the usage of TCP and UDP. TCP is used to transfer an important and large amount of data between the client and server due to its reliable service mechanism, while UDP is usually used to send short messages, the drop of which could be tolerable.

The ratio of TCP and UDP traffic in bytes and packets are similar to each other; over 90% of total packets and total bytes are TCP traffic. Still, TCP is used by the majority of current Internet applications. However, the flow ratio of TCP and UDP traffic is opposite to the previous case. The number of total UDP flows is about two times greater than the number of total TCP flows, as illustrated in Table 10. A small number of UDP packets with less bytes than TCP packets cause a significant amount of flows. This implies that the UDP traffic in the current network environment highly influences the flow-based traffic analysis system negatively, because the performance of these systems depends on the number of generated flows rather than the number of packets and link utilization.

Another interesting fact about flow is illustrated in Table 11. The inbound and outbound traffic shows close one to one ratio in terms of flow count and packet count. The inbound traffic refers to the traffic transferred from the Internet to our campus network, and the outbound traffic is vice versa. Considering bytes, the total outbound traffic is 1.41 times greater than inbound traffic, which is commonly reported in many university networks [31]. This implies that the

inbound packet size is smaller than the outbound packet size, and the inbound byte size of a flow is also smaller than that of outbound flow. Moreover, the outbound TCP bytes are larger than inbound TCP bytes with the same ratio of total traffic, but the inbound UDP bytes are two times larger than outbound UDP bytes. We believe that the former case is mainly due to P2P traffic and popular FTP servers operated by students, and the latter case is due to multimedia service traffic from outside servers using UDP to transfer video/audio data.

Collection Period		2/1/04 – 2/7/04		
Ratio(%)		Flows (In:Out) (%)	Packets (In:Out) (%)	Bytes (In:Out) (%)
Total	TCP	(47:52)	(49:50)	(40:59)
	UPD	(51:48)	(52:47)	(69:30)
	ICMP	(46:53)	(78:21)	(76:23)
	others	(28:71)	(37:62)	(85:14)
	Total	(49:50)	(50:49)	(41:58)

Table 11. Inbound Traffic vs. Outbound Traffic

The total number of internal and external IP addresses appeared in the captured flows was about 3,600,000. Among them, the number of internal IP addresses was about 65,000 which was 2% of total IP addresses; the external IP addresses was 98%.

6.2 High-level Characteristics of IP Traffic Flows

Figure 36, Figure 37, and Figure 38 illustrate six time-series graphs of the traffic trace. Each graph shows variance of three-transport layer protocol (TCP, UDP, and ICMP) traffic and the sum of them in three analysis metrics (flow, packet, and byte). We also categorized the traffic into inbound and outbound traffic to compare the directional behavior of our campus traffic. The total flow distribution is mainly affected by the UDP flows, as illustrated in Figure 36-(a) and Figure 36-(b). The inbound and outbound flow distribution has a similar shape and the average number of outbound flows is slightly larger than that of the

inbound flow.

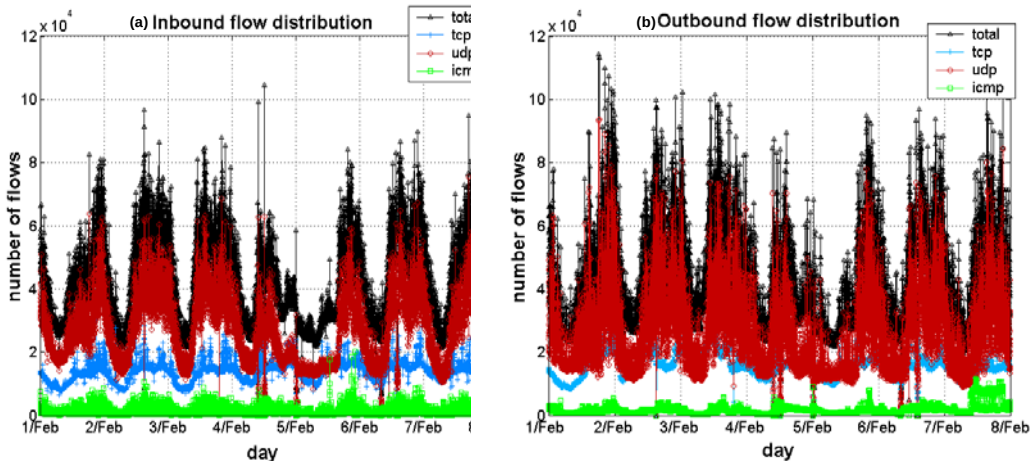


Figure 36. Flow Distribution in Time-series Graph

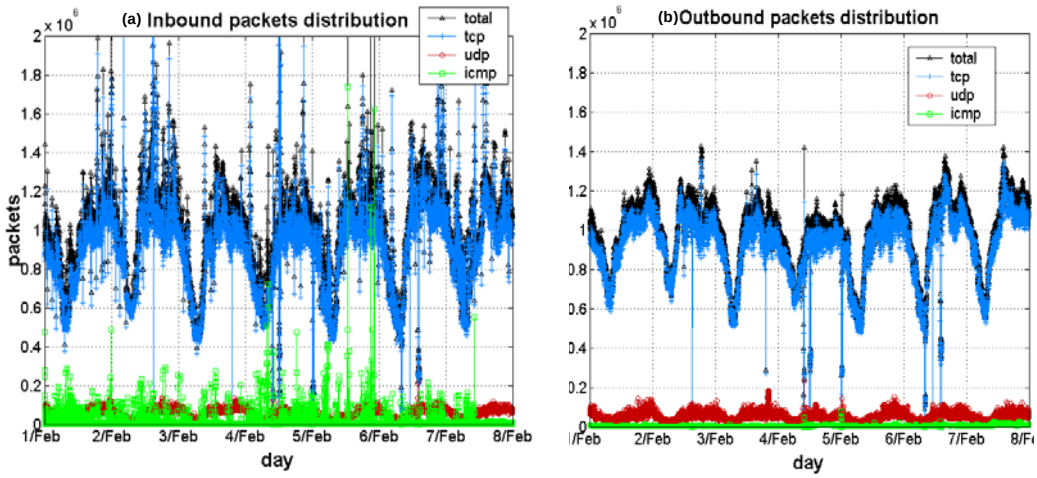


Figure 37. Packet Distribution in Time-series Graph

The shapes of packet distribution and byte distribution graphs are primarily affected by the amount of TCP traffic, which contradicts the shape of flow distribution. The time-of-day effect appears in all three kinds of graphs. The traffic increases from afternoon and marks to the peak between 10 p.m. and 1 a.m. of next day, and it goes down in the morning, which is typical of our university

Internet usage behavior since all of our students live in the campus dormitories.

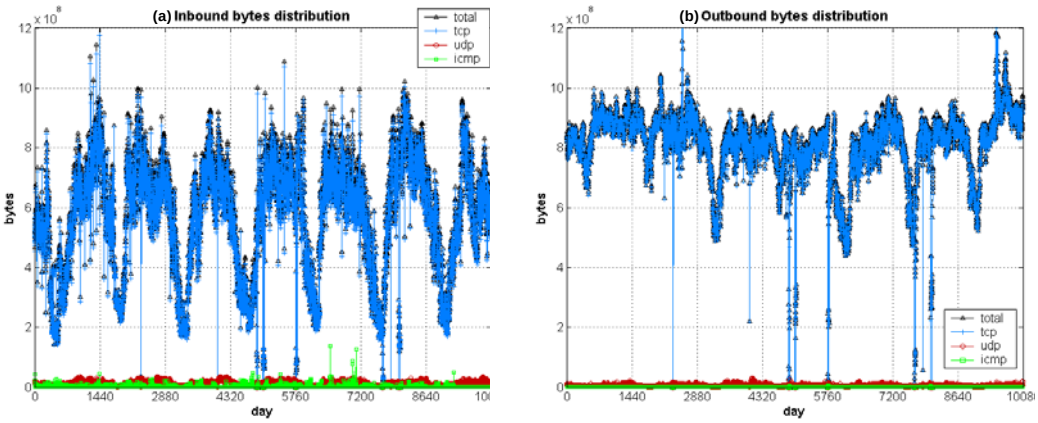


Figure 38. Byte Distribution in Time-series Graph

The incoming ICMP packets are much larger than the outgoing ICMP packets, as illustrated in Figure 37-(a) and Figure 37-(b). This implies that the outside IP addresses more frequently join and leave the network than the inside IP addresses and the number of outside IP addresses are more than that of the inside. The fluctuation of incoming bytes is higher than the outgoing bytes. That is because the number of outside users are much higher than the inside users. In other words, the more users access a network, the less fluctuation of download traffic appears.

6.2.1 Distribution of Flow Duration

Figure 39 illustrates the distribution of flow duration in the form of distribution function (DF) graph and cumulative probability distribution function (CPDF) graph of the traffic trace from Mar/06/2004 to Mar/12/2004, which are drawn in log scale. The total number of flows is 852×10^6. Among them, the number of UDP flows is 592×10^6, which is 2.7 times more than the number of TCP flows.

The average flow length of TCP flows is 57.32 seconds, which is 5.3 times greater than that of UDP flows - 10.72 seconds. As Figure 39-(a) illustrates, we

can observe some long-lasting flows over 10^5 seconds (about 1 day). The maximum value of flow duration is 392,217 seconds, which is 4 days 12 hours 56 minutes 57 seconds. The median of TCP and UDP flow duration is 1 second, which indicates that the flow duration of over 50% of total flows is less than 1 second. The standard deviations of TCP and UDP flows are 540.84 and 76.62, respectively. This indicates that the flow duration of most UDP flows belong to a small range of time intervals.

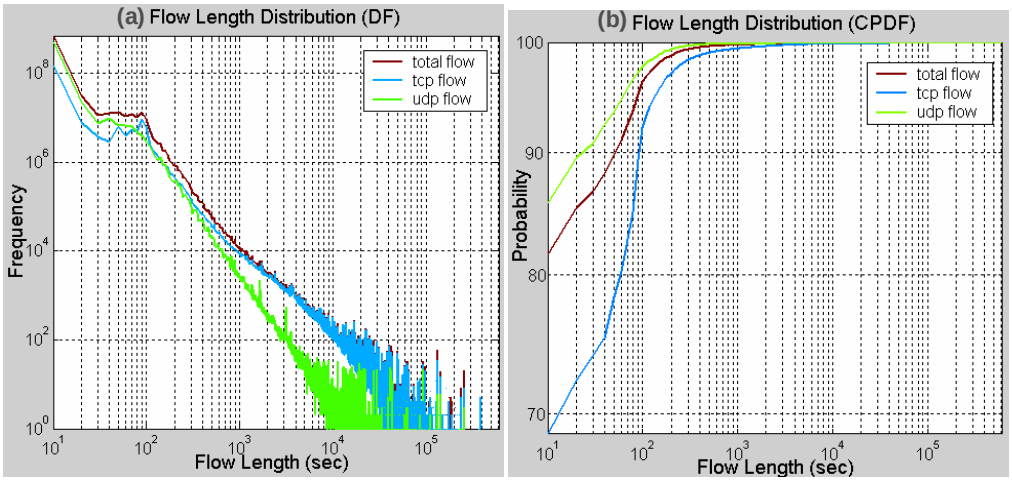


Figure 39. Distribution of Flow Duration

The number of UDP flows less than 80 seconds long is greater than the number of TCP flows, as illustrated in Figure 39-(a). By contrast, over 80 seconds long TCP flows are much more than the number of UDP flows. The number of UDP flows less than 10 seconds is 508,074,860, which is 3.4 times of TCP flows of the same duration. This number is 85.74% of total UDP flows, as illustrated in Figure 39-(b). Moreover, the UDP flows less than 100 seconds and 1000 seconds are 97.76% and 99.97% of total UDP flows, respectively. The duration of most UDP flows is less than 1000 seconds. In case of TCP flows, the TCP flows less than 10 seconds, 100 seconds, and 1000 seconds are 68.64%, 92.19%, and 99.47%, respectively. From this flow duration analysis we know that the duration

of TCP flows are more evenly distributed between 0 and 1000 seconds than UDP flows.

6.2.2 Distribution of Packets in Flows

Figure 40 illustrates the distribution of packets in flows with the form of DF graph and CPDF graph from the same traffic trace, which is also drawn in a log scale.

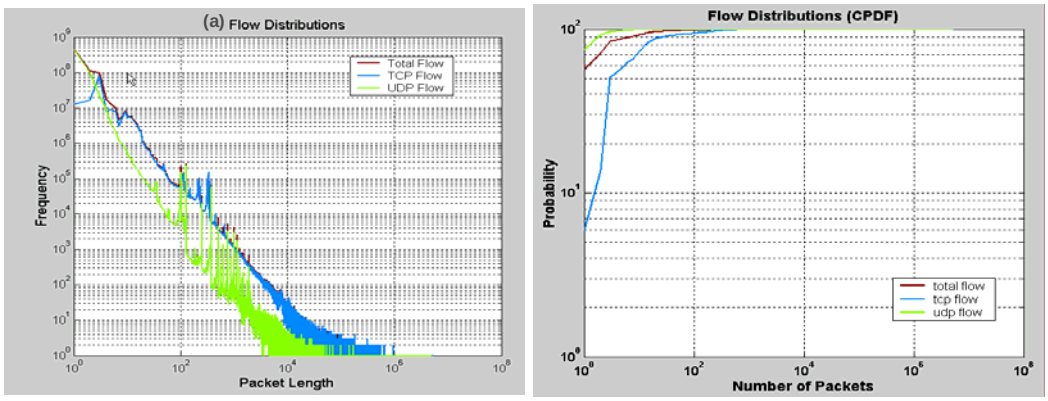


Figure 40. Distribution of number of packets in flows

Figure 40-(a) shows that the distribution of the total flows is similar to the distribution of TCP flows after the packet count exceeds 3. The number of UDP flows with less than 2 packets is 19 times more than that of TCP flows. After the packet count exceeds 3 on the X axis, the number of TCP flows is greater than the number of UDP flows. Figure 40-(b) shows that the ratio of TCP flows aggregated with only 1 packet is about 6% of the total TCP flows, compared to about 76% in the case of UDP flows. The number of TCP flows with less than 1000 packets occupies a large portion of the total TCP flows. By contrast, the number of UDP flows with less than 10 packets takes a large portion of the total UDP flows. Particularly, the number of UDP flows with a couple of packets takes about 92% of the total UDP flows. Consequently, the number of packets belonging to the TCP flows is greater than the number of packets of UDP flows.

6.2.3 Distribution of Bytes in Flows

Figure 41 illustrates the distribution of bytes in flows with the form of DF graph and CPDF graph from the same traffic trace, which is also drawn in a log scale.

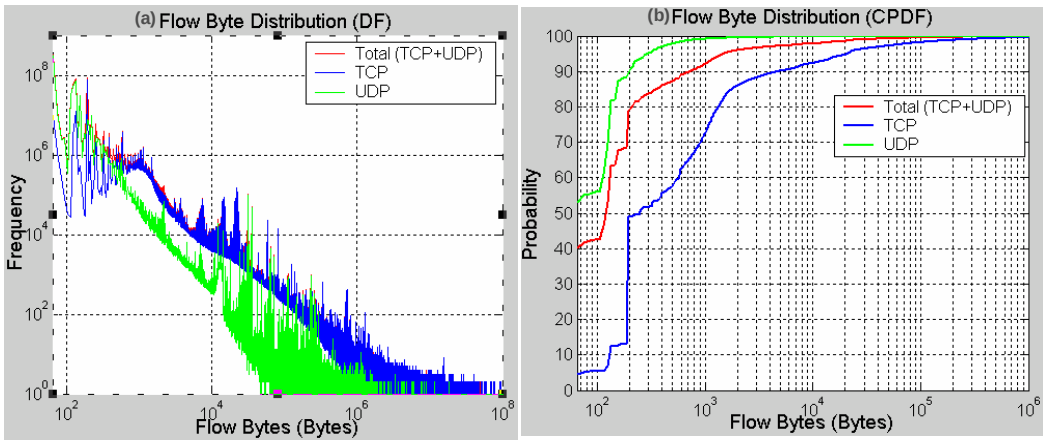


Figure 41. Distribution of byte size in flows

Considering TCP flows, the bytes of TCP flows are evenly distributed until 1000 bytes with some fluctuation. The number of TCP flows decreases exponentially after 1000 bytes. The number of UDP flows shows similar exponential decrease after 300 bytes. Considering the flows having less than 1000 bytes, the number of TCP flows is 154x106, which is 72% of total TCP flows. Figure 5(b) shows three vertical increases of TCP flows at 64, 130, and 200 bytes, respectively. About 90% of TCP flows are composed of less than 4000 bytes.

The number of UDP flows less than 1000 bytes is $587x10^6$, which is 99% of total UDP flows and 3.9 times larger than TCP flows. The 64 bytes UDP flows are 53%, which means that half of the total UDP flows are single packet flow. 90% of UDP flows are less than 200 bytes, as illustrated in Figure 41-(b).

6.2.4 Duration vs. Packets vs. Bytes

Figure 42 illustrates the relationship among three fields (duration, packets and bytes) in flows. We also compare the TCP and UDP flows using these three values together. We used a randomly selected 800,000 flows from our traffic trace to plot these graphs. Two-thirds of them are UDP flows and the remains are TCP flows.

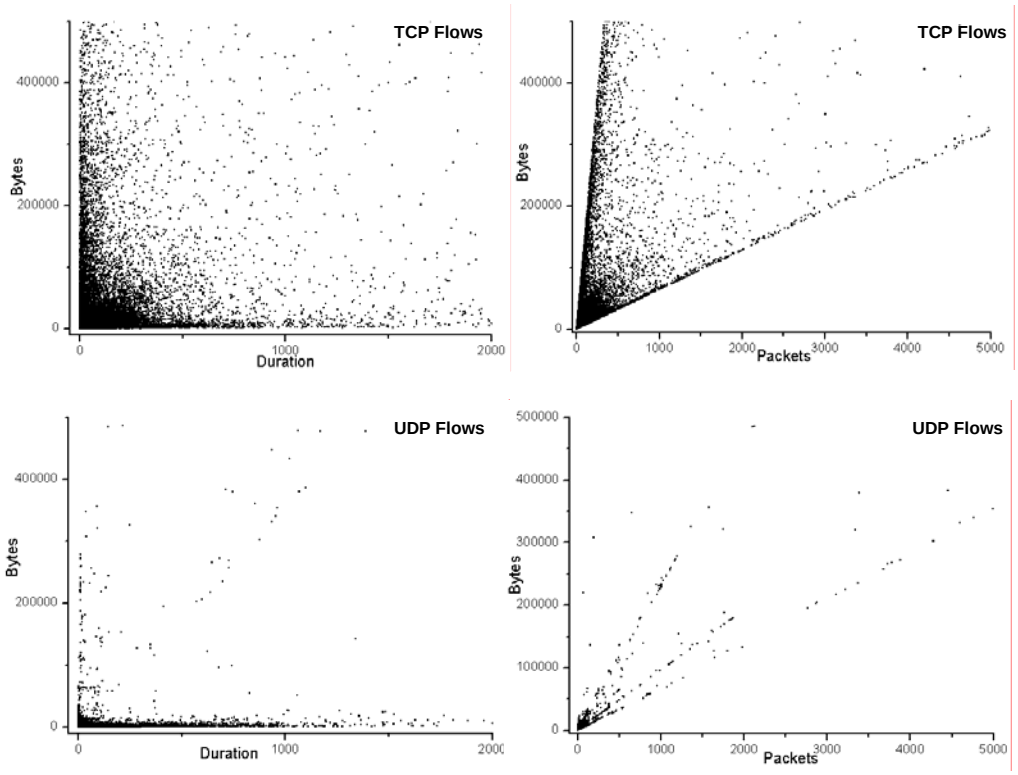


Figure 42. Relationship among duration, packets, and bytes

Figure 42-(a) and Figure 42-(c) show that the bytes in the flows are independent of the flow duration. The relationship between packets vs. flow duration is also independent of each other, which shows similar graphs to the bytes vs. duration graphs. The bytes in most UDP flows are less then 5000 bytes regardless of the flow duration. But the bytes of TCP flows are spread widely in the chart. The flows with large bytes and low duration and flows with small bytes

and high duration appear together in this graph.

The bytes in flows are proportional to the number of packets, as illustrated in Figure 42-(b) and Figure 42-(d). In the bytes vs. packets graph of TCP flows, two thick boundary lines appear, and all TCP flows belong between these two boundary lines. The bytes per packet around the lower boundary lines are 64 bytes, and the bytes per packet around upper boundary lines are 1500 bytes, which is the maximum packet size of the Ethernet frame. We have a considerable amount of TCP flows with 1500 bytes per packet, while no UDP flows has this amount of bytes per packet value. Most UDP flows have less than 500 packets and 50,000 bytes.

6.3 Application-level Characteristics of IP Traffic Flows

For application-level traffic identification, we constructed the APT information by investigating about 700 Internet-based applications. When we applied our proposed method to determine the application name of traffic flows, the proportion of determined traffic from the total traffic trace was 99.5% in terms of flows, 94% in terms of packets, and 92% in terms of bytes, respectively. The determined ratio of flows, packets, and bytes in each collection periods are illustrated in Table 12.

Collection Period	2/1/'04 – 2/7/'04			2/17/04 – 2/23/04			3/6/'04 – 3/12/'04		
	Flows ($x10^6$)	Packets ($x10^6$)	Bytes ($x10^9$)	Flows ($x10^6$)	Packets ($x10^6$)	Bytes ($x10^9$)	Flows ($x10^6$)	Packets ($x10^6$)	Bytes ($x10^9$)
Total	833	19,443	13,888	869	20,627	13,947	899	20,407	13,804
Determined	827	18,605	12,998	861	19,475	13,017	892	19,222	12,759
Ratio (%)	99.3	95.6	93.6	99.1	94.4	93.3	99.3	94.1	92.4

Table 12. Summary of Application Identification

The identification ratio of flows is greater than those of packets and bytes. The reason is the proposed method is based on flow correlations. A flow with

large amount of packets or bytes cannot be determined if we could not find any dependency information with other flows. In fact, we could easily find some applications which provide a way to set the peer's port number and IP address manually by user.

We found an interesting fact that most determined traffic were generated by less than 100 applications among the applications listed in the APT. Table 13 shows the 10 heaviest applications in three perspectives of traffic metrics: the number of flows, the number of packets, and the total byte size, respectively. As Table 13 shows, the flow distribution does not follow the packet and byte distribution, while the packet and byte distribution is almost in accordance with each other.

Flows		
Top 10 apps	ratio(%)	(In:Out) (%)
eDONKEY	48.5	(50.8 : 49.2)
SORIBADA	29.6	(49.9 : 50.1)
V_SHARE	4.5	(54.0 : 46.0)
HTTP-WEB	4.1	(49.1 : 50.9)
MSN	2.2	(50.2 : 49.8)
BATTLE_NET	2.0	(10.5 : 89.5)
AFS	1.8	(49.9 : 50.1)
DNS	1.4	(49.5 : 50.5)
SAYCLUB	0.9	(49.8 : 50.2)
FREECHAL	0.6	(49.9 : 50.1)
Total	95.6	(49.7: 50.3)

Packets		
Top 10 apps	ratio(%)	(In:Out) (%)
eDONKEY	27.3	(50.6 : 49.4)
HTTP-WEB	16.6	(56.0 : 44.0)
FREECHAL	7.5	(41.1 : 58.9)
FTP	7.2	(44.0 : 56.0)
V_SHARE	4.8	(48.9 : 51.1)
SORIBADA	3.2	(49.4 : 50.6)
AFS	2.9	(47.2 : 52.8)
MSN	2.7	(50.4 : 49.6)
mIRC	2.0	(44.9 : 55.1)
BITTORENT	1.8	(44.6 : 55.4)
Total	76.0	(49.6:50.4)

Bytes		
Top 10 apps	ratio(%)	(In:Out) (%)
eDONKEY	24.2	(47.6 : 52.4)
HTTP-WEB	18.1	(66.2 : 33.8)
FREECHAL	9.5	(14.6 : 85.4)
FTP	8.7	(21.2 : 78.8)
V_SHARE	5.8	(44.8 : 55.2)
MSN	2.8	(45.4 : 54.6)
mIRC	2.3	(5.5 : 94.5)
SORIBADA	2.0	(34.6 : 65.4)
BITTORENT	2.0	(25.8 : 74.2)
WMedia	1.3	(91.3 : 8.7)
Total	76.7	(43.2:56.8)

Table 13. Top 10 Most Popular Applications in Flows, Packets, and Bytes

The top 10 most popular applications occupy the 95.6% of total flows, the 76% of total packets, and the 76.7% of total bytes, respectively. This indicates that the flow distribution is more skewed than the other two distributions. Six applications in flow distribution and seven applications in packet and byte distributions in the above table are P2P applications, which belong to M-D-3. Our results are in accordance with the results of several previous results in P2P traffic analysis [29, 30, 32, 33]. The Web traffic is still one of the most traffic-consuming applications, while the FTP application is less than web application. The world-wide P2P applications such as eDonkey [82] and KaZaA [32] occupy a large part

of Internet traffic. In addition, the nation-wide P2P applications such as V_SHARE [84], FREECHAL [85], SAYCLUB [81], and SORIBADA [83] are located in the top 10 list of three different distributions and occupy a large part of Internet traffic.

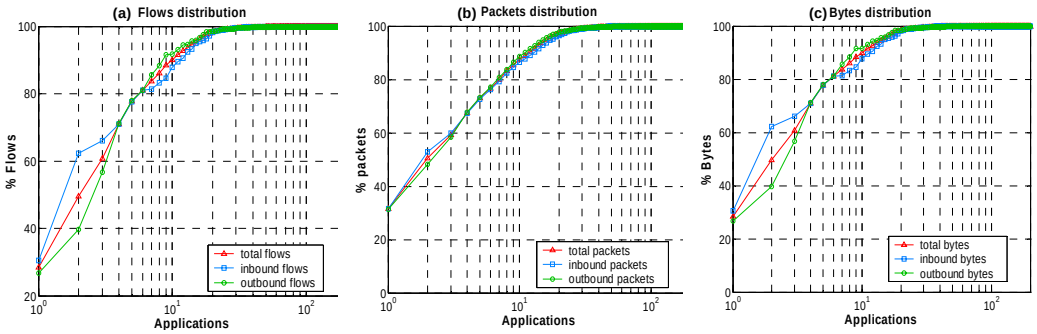


Figure 43. CDF of top-150 applications generating traffic

Figure 43 shows cumulative probability distributions in terms of flows, packets, and bytes of the top 150 traffic-generating applications in our traffic trace. Figure 43 indicates that the flow distribution is more skewed than the other two distributions. Several popular applications generate most IP traffic flows; the top six applications occupy about 80% of total traffic flows. In addition, the outbound traffic flows are more skewed than the inbound traffic flow distribution and byte distribution.

6.3.1 Traffic Statistics of Selected Applications

Among the top-10 applications, we have selected seven applications for the following categories: Traditional, P2P file sharing, instant messaging, and streaming applications. Web and FTP are representatives for the traditional applications which still take a significant portion of Internet traffic. SORIBADA (a Korean version of Napster), V_SHARE, and eDonkey are our choice of P2P file sharing applications widely used in Korea as well as in the rest of the world. MSN is the most popular instant messaging application without a question. Finally,

we investigate the streaming media traffic of Microsoft's Windows media application.

Packets	WWW	FTP	Window Media	Soribada	eDonkey	MSN Messenger	V-share
	80	21	1755	22321	4661	1755	8404
Min	1	1	1	1	1	1	1
Max	1,078E3	2.491E6	658364	357403	375090	894156	1.341E6
Mean	197.16	558.50	1696.58	2.768	21.49	239.26	33.31
std	4533.56	16492.50	10185.18	492.97	1787.54	7293.31	4125.44

Bytes	WWW	FTP	Window Media	Soribada	eDonkey	MSN Messenger	V-share
	80	21	1755	22321	4661	1755	8404
Min	64	64	64	64	64	64	64
Max	1.380E9	3.545E9	9.868E8	6.625E7	4.819E8	1.326E9	1.819E9
Mean	154603.15	464653.32	1.806E6	626.50	14313.59	173030.81	31271.88
std	5.14173E6	2.00648E7	1.249E7	106643.33	2.103E6	8.912E6	5.164E6

Flow Duration	WWW	FTP	Window Media	Soribada	eDonkey	MSN Messenger	V-share
	80	21	1755	22321	4661	1755	8404
Min	0	0	0	0	0	0	0
Max	120047	77409	106554	85970	42906	141228	26432
Mean	61.32	20.08	337.38	9.00	30.19	547.54	1.98
Std	606.28	421.74	1465.44	158.12	279.82	3761.41	100.70

Table 14. Statistics of Selected Applications

Table 14 illustrates packets, bytes, and flow duration summary of each application. One interesting behavior of SORIBADA is to generate excessive number of flows which contain only a couple of small size packets. This characteristic results in the relatively low average number of packets per flow - 2.768 packets per flow. The mean number of packets in eDonkey's flow is 21.49 packets which is slightly larger than SORIBADA's. The architectural differences, such as node structure and search mechanism, between the two applications are responsible for this phenomenon.

The maximum values of flow bytes of all the 7 applications are very high (over 10E7). This indicates that all these applications have a functionality to transfer large amount of data in a flow like FTP. Emerging new applications (P2P

file sharing, instant messaging, and streaming media applications) share the similar characteristic with FTP in the category of the maximum transferred bytes. Since P2P file sharing and instant messaging application support the data transferring functionality, they create full size packets within the data session, like the FTP's data session. Streaming media applications also require sending out the full packets. In addition, web also seems to share the similar property. This is because some of the applications use port 80 for download, and the downloadable content is embedded into web sites.

Flow duration is low for P2P files sharing applications. We believe that short query and search messages are responsible for this phenomenon; the average flow duration of SORIBADA and V_SHARE is 9 and 1.98 seconds, respectively. The mean duration of Web is greater due to the user behavior of the Web browsing applications where they involve the frequent user interaction. However, eDonkey does not fall into this category due to the large user population in the campus and relatively well optimized query and search mechanism.

Finally, MSN messenger, an instant messaging application, has the longest mean value of flow duration among the selected seven applications (547.54 seconds). There is a better chance for longer flow duration if the application is capable of producing packets constantly with the short inter-packet generation time. Consequently, the MSN messenger service acquires constant buddy list updates (periodic interaction with the central server) and has usual user behaviors of chatting tools.

6.3.2 Traffic Variation over Time

For each application, Figure 44, Figure 45, Figure 46 illustrate the time-series graph with a week long data in three different metrics: flows, packets, and bytes.

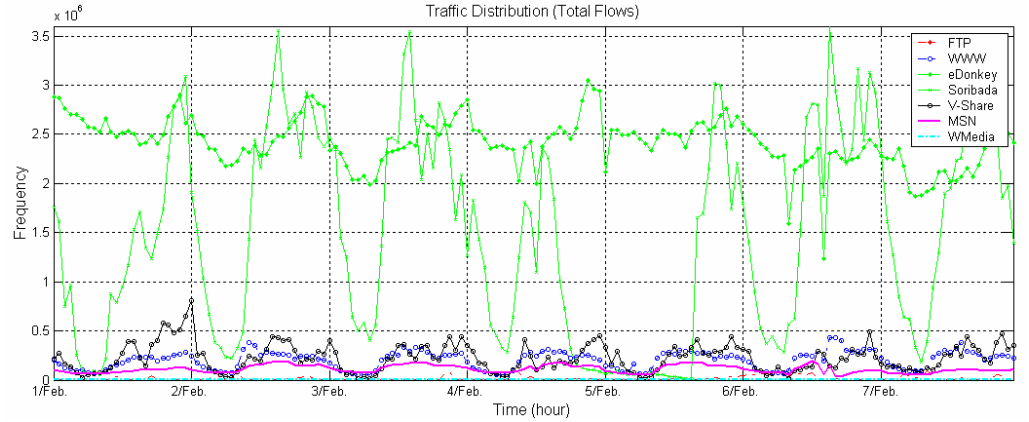


Figure 44. Flow Variations over Time

In Figure 44, eDonkey generates the top number of flows among the seven applications. Another P2P file sharing application, SORIBADA, produces the second most number of flows in the campus. However, SORIBADA's proportion in terms of packets and bytes are less significant. Due to its design architecture, most of SORIBADA's flows are search and query flows with small size packets and overwhelm the number of download sessions.

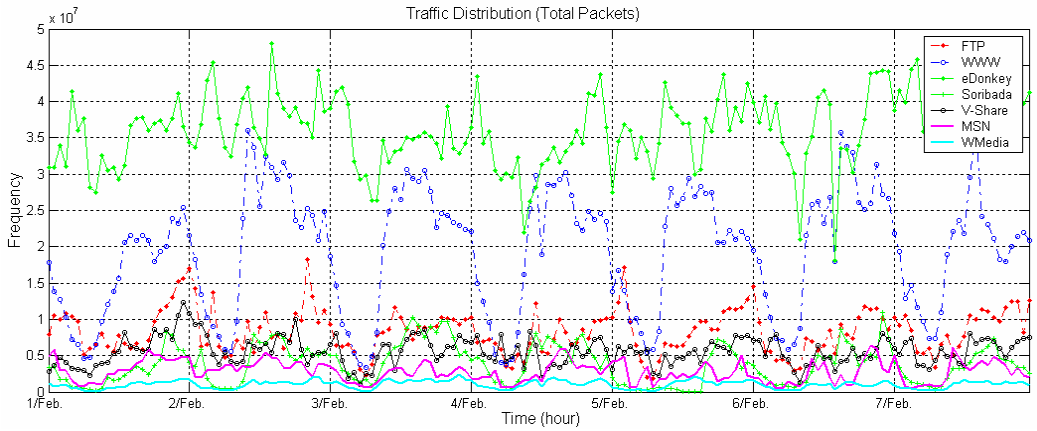


Figure 45. Variation of number of packets over Time

In Figure 45, we observe that the Web traffic emerges as the second heaviest traffic as well as in Figure 46. The average packets and bytes size per flow from the Web traffic must be greater than the V_SHARE traffic because there is less

number of flows with heavy packets. In addition, eDonkey again takes the top spot in packets and bytes consumption. It is no doubt that the P2P file sharing application traffic occupies a large volume of network traffic and reflects the recent traffic pattern.

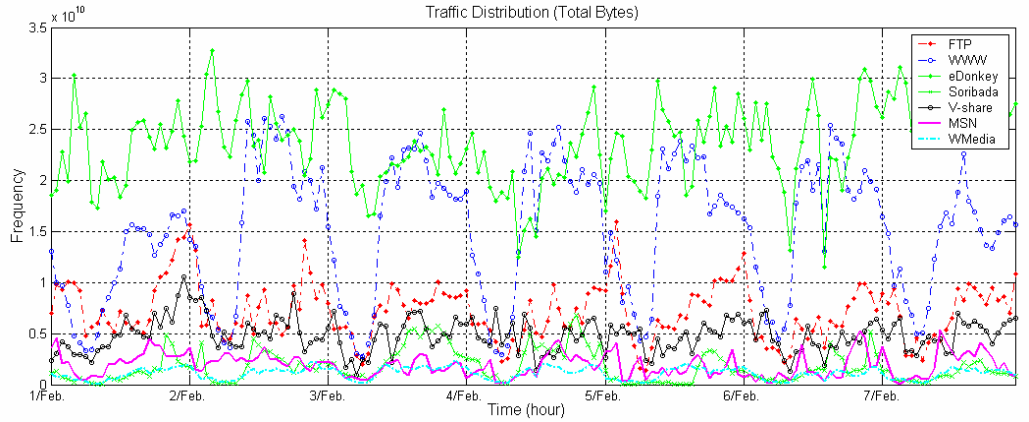


Figure 46. Variation of Bytes over Time

The traffic generated from each application appears with time of day feature. Especially, the applications which involve a strong user interaction, such as Web and MSN messenger, show the strong signs of this feature. On the other hand, it is less obvious for the eDonkey traffic because it is not only popular in Korea but also in the rest of the world.

6.3.3 Number of packets and bytes of application traffic flows

For each application, Figure 47, Figure 48, and Figure 49 illustrate the relationship between the two metrics: bytes and packets. One common characteristic of the figures is that they all have clear upper and lower boundaries. The upper and lower bounds indicates the range of Ethernet frame size – 64 ~ 1500 bytes.

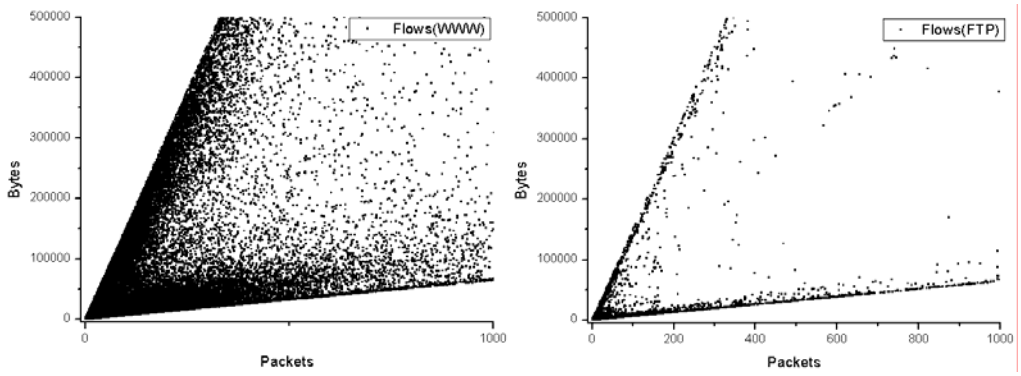


Figure 47. Traditional Applications

In traditional applications, the Web traffic consists of packets with wide range of byte sizes. Although it seems the density around the boundaries is quite high, it is simply because there are a large number of packets generated by Web (16% of all packets). On the contrary, FTP packets are concentrated on the two boundaries in Figure 48. This reflects the difference in bytes of the packets generated by two separate connection sessions of FTP: a control session and a download session. The control session usually contains simple command messages, such as start and stop, so the generated packets are small. The download session sends out the full packets (with maximum 1500 bytes). Thus most FTP related packets are either minimum or maximum of Ethernet frame unit.

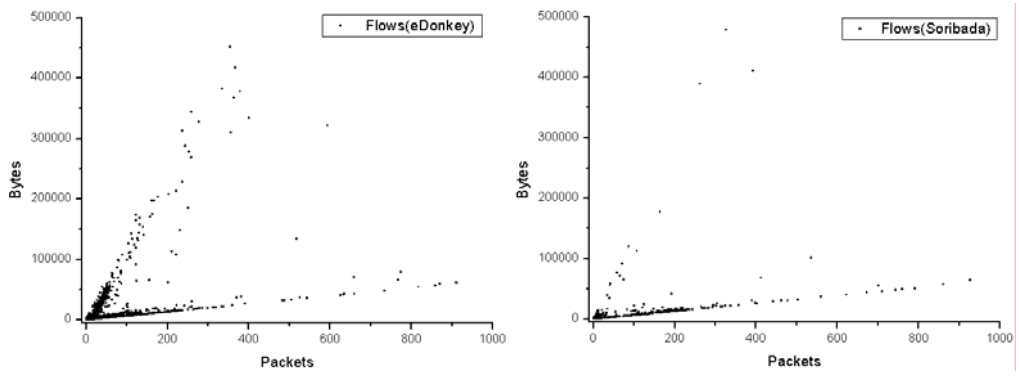


Figure 48. File Sharing Applications

In Figure 48, P2P file sharing traffic shows a somewhat similar shape to FTP

traffic. However, there is less number of full packets than FTP traffic has due to the following reasons: unstable connection and low successful download ratios. Unlike FTP's stable connection, P2P systems can not guarantee a reliable connection with the content provider. Also, the connection speed varies to the network condition, so users have the tendency to frequently cancel the established download session. In the case of SORIBADA, one additional factor causes the phenomenon described above. The content being exchanged is mp3 music files which are relatively small, usually less than 10 Mb. Furthermore, packets, which stay close to the lower boundary on the graph, consist of query, search, and ping-pong messages of typical P2P applications.

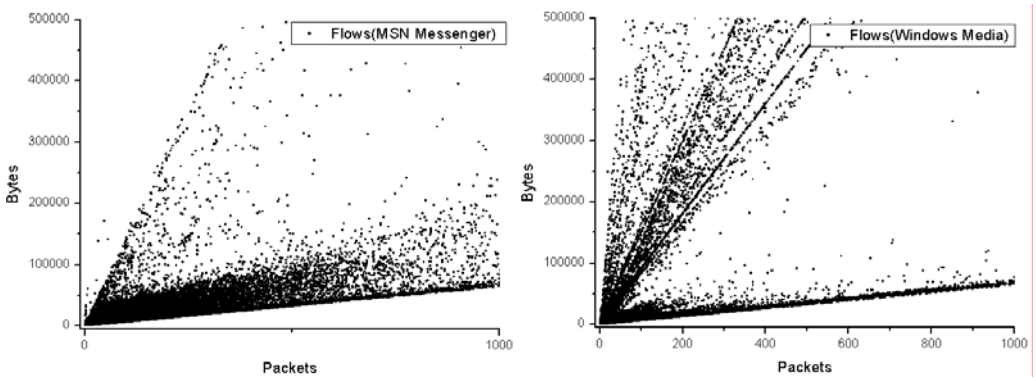


Figure 49. Instant Messaging and Streaming Applications

In Figure 49, we observe that there is high density of points nearby the lower boundary on the MSN messenger graph. This infers that a large portion of MSN messenger traffic consists of small size packets with simple text messages. We also monitor the full size packets that are used for P2P download sessions. In the Windows media example, the shape of the graph is again similar to FTP traffic. However, the wider distribution of packets around the upper boundary is present. We believe that rate control mechanism of streaming media applications is responsible for this distribution model. The data transmission ratio of the download session can be selected by the user or the provider (e.g. 100 Kbytes/sec, 300 Kbytes/sec, or higher) and has the influence on the size of data packets. In

addition, some of the points are placed beyond the maximum bound, 1500 bytes. These points appear in the data set because we reassemble the fragment packets in the flow generator phase.

7. Conclusion

This section summarizes the overall contents of the thesis and lists a set of contributions this thesis achieved. In addition, the future work is discussed.

7.1 Summary

Two critical problems exist in traffic monitoring and analysis of today's Internet traffic compared to the past network environment. The first is how to capture and handle the huge amount of traffic data generated from high-speed network links (such as 2.5 Gbps and higher) in a real-time manner, The second is how to analyze diverse and complex types of traffic generated by many different types of network-based applications, such as streaming media, P2P, and gaming applications.

Considering the first problem, this thesis presented a scalable and flexible design of a network traffic monitoring and analysis system, called NG-MON, for high-speed networks which can be applied any kind of network links. Using distributed, pipelining and parallel processing techniques, we have designed a flexible and scalable monitoring and analysis system, which can run on off-the-shelf, cost-effective computers. The monitoring and analysis task in NG-MON is divided into five phases; packet capture, flow generation, flow store, traffic analysis, and presentation. Each phase can be executed on separate computer system and cooperates with adjacent phases using pipeline processing. Each phase can be composed of a cluster of computers wherever the system load of the phase is higher than the performance of a single computer system. In addition, we have defined efficient communication methods and message formats between phases.

The other problem with traffic analysis is caused by the large number of applications and their use of dynamic sessions. The traditional traffic analysis mechanism which is based on the well-known port number is not suitable to

analyze newly emerging Internet traffic, such as P2P, streaming media, and game traffic. In this thesis, we have presented a method to identify the Internet traffic at the application layer, which is the preliminary but critical step for the characterization of Internet traffic as well as for other variety of uses. First, we categorized Internet traffic from perspective of traffic analysis. We categorized most current network-based applications into five classes according to their traffic usage pattern. Using this categorization, we developed a flow identification method, which determines the application name of individual traffic flows. The proposed method consists of three components: an Application Port Table (APT), an Important Port Selection (IPS), and a Flow Relationship Map (FRM).

The first step of the proposed method is to construct an Application Port Table (APT). The APT contains the application name, its frequently used port numbers excluding dynamically assigned port numbers, and transport-layer protocol numbers. The second step (IPS) generates flow information from the captured packets according to their 5-tuple information, and then selects the important port number from the flow information of TCP flows. The third step (FRM) classifies individual flows into a number of groups according to their relationships and marks flows with corresponding application name. The FRM consists of two consecutive flow grouping steps: PDG and LDG.

To validate our method, we designed and implemented an application-layer traffic analysis system as an essential part of the NG-MON system, and deployed this system on the Internet junction of our campus network. We collected IP traffic form our campus Internet junction and determined the application name of the IP traffic flows. We have identified more than 90% of Internet traffic from POSTECH using our proposed method. We characterized the flow-level features of current IP traffic based on the traffic traces. The analysis result shows that more than 50% of recent Internet traffic is caused by the newly emerging applications, especially P2P applications. In addition, most Internet traffic is generated by about 10 most popular applications.

7.2 Contributions

The thesis research was conducted on three critical areas of current IP traffic monitoring and analysis: the architecture of monitoring system for high-speed network links, the application-level traffic identification method, and the application-level traffic characterization on current IP traffics. The followings are core contributions that are expected from this thesis research according to these sub-research categories.

First, we have designed and implemented an architecture of a real-time traffic monitoring and analysis system (NG-MON) suitable for the high-speed network links from scratch. We have deployed the NG-MON on our campus Internet junction. From the first set up of NG-MON in 2002, NG-MON provides various analysis results about Internet traffic to the campus network administrators till now. Along with our experience in the development of NG-MON, a feasible guideline for developing a traffic monitoring system for high-speed backbone network is presented.

Second, we categorized current network-based applications into several classes according to their traffic patterns generated by them. This categorization must be helpful to understand the dynamics of IP traffic generated by various Internet-based applications, such as peer-to-peer, streaming, and gaming applications. Because the current networks are full of highly complicated and sophisticated applications like these, the monitoring and characterizing of IP traffic is highly challengeable.

Third, we developed a new method to identify IP traffic in the application layer. This method is highly significant as the preliminary step to the application-level traffic characterization. Our method is the first research result which considers the dynamically assigned port numbers in the entire range of applications. Through the knowledge achieved from the development of the proposed method, along with our experience in implementing and deploying it, it was possible to distinguish Internet traffic according to the corresponding

applications with high accuracy.

Finally, we have performed various analyses on the IP traffic from the perspective of flows. As a result, we found that the new types of Internet-based applications generate large amount of shot-time and short-size flows, which influences the performance of modern traffic monitoring and analysis systems negatively. In addition that, many new features of recent Internet traffic are presented from this thesis research, which will help to understand current IP traffic widely and deeply.

7.3 Future Work

Real-time traffic monitoring on high-speed network and application-level traffic characterization are critical research challenges on current traffic monitoring and analysis. In this thesis research, we handled these two problems and proposed solutions with good result. The followings are a list of future work we have in mind.

In this thesis, we characterized Internet traffic using traffic trace of a short time period and from limited network links, which is insufficient to discern the overall trends of recent Internet traffic. We are in the process of constructing a long-term traffic trace by collecting traffic for a week in each month, and making various traffic traces by collecting traffic in a number of ISP and enterprise networks.

The IP traffic analysis performed in the thesis research mainly focused on the flows in various perspectives. Though the analysis results provide a range of useful information about current application traffic, we have many other analysis items which should be taken. We are planning to do wide and in-deep analysis on the collected traffic traces to get more information about current IP traffic.

The exact traffic identification in application layer leads to the accuracy of high-level traffic analysis, such as P2P and streaming traffic characterization. Therefore, we intend to apply the proposed method to various traffic trace to

improve our proposed method, especially the FRM algorithm. Currently, we applied our traffic identification method only to the normal traffic trace. We should consider the traffic in case that abnormal traffic by DoS/DDoS and Internet Worm occurs. We are going to apply our identification method to this kind of traffic trace and validate and improve our method.

Further, the detail and accurate traffic analysis can provide useful information towards the control of current Internet traffic. We are also considering the QoS traffic provisioning based on the analysis result of Internet traffic as the next step of our research.

References

[1] Se-Hee Han, Myung-Sup Kim, Hong-Taek Ju and James W. Hong, "The Architecture of NG-MON: A Passive Network Monitoring System", LNCS 2506, DSOM 2002, October 2002, Montreal Canada, pp. 16-27.

[2] Se-Hee Han, Hong-Taek Ju, Myung-Sup Kim and James W. Hong, "Design of Next Generation High-Speed IP Network Traffic Monitoring and Analysis System", Proc. of 2002 Asia-Pacific Network Operations and Management Symposium (APNOMS 2002), Jeju, Korea, September 25-27, 2002, pp. 282-293.

[3] Hyo-Jin Lee, Myung-Sup Kim, James W. Hong and Gils-Haeng Lee, "Mapping between QoS Parameters and Network Performance Metrics for SLA monitoring", Proc. of 2002 Asia-Pacific Network Operations and Management Symposium (APNOMS 2002), Jeju, Korea, September 25-27, 2002, pp. 97-108.

[4] TM Forum, "Performance Reporting Concepts and Definitions," TMF701 v2.0, Nov., 2001.

[5] TM Forum, "Service Level Agreement Management Handbook," GB917 v1.5, Jun., 2001.

[6] T. Choi, S. Yoon, H. Chung, C. Kim, J. Park, B. Lee, T. Chung,"Wise: Traffic Engineering Server for A Large-scale MPLS-based IP Network," Proc. of NOMS 2002, Florence, Italy, Apr., 2002, pp. 251-264.

[7] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," IETF RFC2475, Dec., 1998.

[8] E. Rosen, A. Viswanathan, R. Callon, “Multiprotocol Label Switching Architecture “, RFC3031, IETF, Jan. 2001.

[9] Moon, Y.S., Leung, C.C., Yuen, K.N., Ho, H.C., and Yu, X, “A CRM model based on voice over IP,” 2000 Canadian Conference on Electrical and Computer Engineering, Volume: 1 , 7-10 March 2000, pp. 464 - 468.

[10] Kun-chan Lan, Alefiya Hussain, and Debojyoti Dutta, “Effect of Malicious Traffic on the Network,” Proc. of PAM 2003, San Diego, California, April 2003.

[11] Symantec Security Response, http://www.symantec.com/avcenter/.

[12] L. John Ioannidis and Steven M. Bellovin, “Implementing pushback: Router-based defense against DDoS attacks,” Proc. of Network and Distributed System Security Symposium, NDSS '02, San Diego, California. February 2002.

[13] Hun-Jeong Kang, Seung-Hwa Chung, Seong-Cheol Hong, Myung-Sup Kim and James W. Hong, "Towards Flow-based Abnormal Network Traffic Detection", Proc. of 2003 Asia-Pacific Network Operations and Management Symposium (APNOMS 2003), Fukuoka, Japan, October 1-3, 2003, pp. 369-380.

[14] Myung-Sup Kim, Hun-Jeong Kang, Seong-Cheol Hong, Seung-Hwa Chung, James W. Hong, "A Flow-based Method for Abnormal Network Traffic Detection", Accepted to appear in the Proc. of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2004), Seoul, Korea, April 2004.

[15] TS Choi, CH Kim, SH Yoon, JS Park, HS Chung, BJ Lee, HH Kim, and TS Jeong, “Rate-based Internet Accounting System Using Application-aware Traffic Measurement,” Proc. of 2003 Asia-Pacific Network Operations and Management Symposium (APNOMS 2003), Fukuoka, Japan, October 1-3, 2003, pp.404-415.

[16] Deb Agarwal, Jose Maria Gonzalez, Goujun Jin, and Brian Tierney , "An Infrastructure for Passive Network Monitoring of Application Data Streams", Passive and Active Measurement Workshop, La Jolla, California, April 2003.

[17] Luca Deri, "Passively Monitoring Networks at Gigabit Speeds Using Commodity Hardware and Open Source Software", Passive and Active Measurement Workshop, La Jolla, California, April 2003.

[18] Gianluca Iannaccone, Christophe Diot, Ian Graham, and Nick McKeown, "Monitoring Very High Speed Links", ACM SIGCOMM Internet Measurement Workshop, San Francisco, Nov. 2001.

[19] Myung-Sup Kim, Hun-Jeong Kang and James W. Hong, "Towards Peer-to-Peer Traffic Analysis Using Flows", Lecture Notes in Computer Science 2867, Edited by Marcus Brunner, Alexander Keller, 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2003), Heidelberg, Germany, October, 2003, pp. 55-67.

[20] Hun-Jeong Kang, Myung-Sup Kim and James Won-Ki Hong, "A Method on Multimedia Service Traffic Monitoring and Analysis", Lecture Notes in Computer Science 2867, Edited by Marcus Brunner, Alexander Keller, 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2003), Heidelberg, Germany, October, 2003, pp. 93-105.

[21] Hun-Jeong Kang, Hong-Taek Ju, Myung-Sup Kim and James W. Hong, "Towards Streaming Media Traffic Monitoring and Analysis", Proc. of 2002 Asia-Pacific Network Operations and Management Symposium (APNOMS 2002), Jeju, Korea, September 25-27, 2002, pp. 503-504.

[22] Ian D Graham and John G Cleary, "Cell level measurements of ATM traffic," Proceedings of the Australian Telecommunications Networks and Applications Conference, pp. 495-500, December 1996.

[23] Cisco Systems, White Papers, "NetFlow Services and Applications," http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm .

[24] P. Phaal, S. Panchen and N. McKee, "InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks", IETF RFC 3176, September 2001.

[25] N. Brownlee, C. Mills and G. Ruth, "Traffic Flow Measurement: Architecture", IETF RFC 2722, October 1999.

[26] N. Brownlee, "Traffic Flow Measurement: Experiences with NeTraMet", IETF RFC2123, March 1997.

[27] Ken Keys, David Moore, Ryan Koga, Edouard Lagache, Michael Tesch, and K. Claffy, "The Architecture of CoralReef: An Internet Traffic Monitoring Software Suite," PAM Workshop 2001, April 23-24, 2001

[28] Internet2, "http://netflow.internet2.edu/weekly/," 2003.

[29] Subhabrata Sen and Jia Wang, "Analyzing peer-to-peer traffic across large networks", in Proceedings of the second ACM SIGCOMM Workshop on Internet Measurement Workshop, Nov. 2002.

[30] Alexandre Gerber, Joseph Houle, Han Nguyen, Matthew Roughan, and Subhabrata Sen, "P2P The Gorilla in the Cable", National Cable & Telecommunications Association (NCTA) 2003 National Show, Chicago, IL, June 8-11, 2003.

[31] Stefan Saroiu, Krishna P. Gummadi, Richard J. Dunn, Steven D. Gribble, and Henry M. Levy, "An Analysis of Internet Content Delivery Systems", Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), Boston, MA, December 2002.

[32] Nathaniel Leibowitz, Matei Ripeanu, and Adam Wierzbicki, "Deconstructing the Kazaa Network", 3rd IEEE Workshop on Internet Applications (WIAPP'03), June 2003.

[33] Nathaniel Leibowitz, Aviv Bergman, Roy Ben-Shaul, and Aviv Shavit, "Are File Swapping Networks Cacheable?", 7th International Workshop on Web Content Caching and Distribution (WCW), Boulder, Colorado, August 14-16, 2002.

[34] IANA port number list, IANA, http://www.iana.org/assignments/port-numbers.

[35] H. Schulzrinne, A. Rao, and R. Lanphier, “Real Time Streaming Protocol (RTSP),” RFC 2336, April 1998.

[36] Microsoft, Windows Media Technology, http://www.microsoft.com/windows/windowsmedia/default.asp.

[37] Jacobus van der Merwe, Ramon Caceres, Yang-hua Chu, and Cormac Sreenan “mmdump- A Tool for Monitoring Internet Multimedia Traffic,” ACM Computer Communication Review, 30(4), October 2000.

[38] Real Networks, Real Media Technology, http://www.realnetworks.com/.

[39] H. Schulzrinne, S. Casner, R. Frederick, V, and Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” RFC1889, January 1996.

[40] J.Craig Lowery, “Using Dell PowerApp.cache for Caching and Splitting Media Streams,” http://www.dell.com/us/en/esg/topics/power_ps3q01-lowery.htm, May 2001.

[41] libpcap, http://www.tcpdump.org/.

[42] Ranjita Bhagwan, Stefan Savage, and Geoffrey Voelker, "Understanding Availability", Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), Berkeley, CA, Feb 2003.

[43] B. Krishnamurthy, J. Wang, and Y. Xie, "Early measurements of a cluster-based architecture for P2P systems," ACM SIGCOMM Internet Measurement Workshop, (San Francisco, CA), Nov. 2001.

[44] Krishna P. Gummadi, Richard J. Dunn, Stefan Saroiu, Steven D. Gribble, Henry M. Levy, and John Zahorjan, "Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload," Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP-19), October 2003.

[45] S. Saroiu, P. Gummadi, and S.D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," Proceedings of International Conference on Distributed Computing Systems, 2002.

[46] P. Krishna Gummadi, Stefan Saroiu, Steven Gribble, "A Measurement Study of Napster and Gnutella as Examples of Peer-to-Peer File Sharing Systems".

[47] J. Chu, K. Labonte, and B. Levine, "Availability and locality measurements of peer-to-peer file systems," Proceedings of ITCom: Scalability and Traffic Control in IP Networks, July 2002.

[48] E. P. Markatos, "Tracing a large-scale Peer-to-Peer System: an hour in the life of Gnutella," 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2002.

[49] Apple, QuickTime, http://www.apple.com/quicktime.

[50] MSN Messenger, http://messenger.msn.co.kr, Microsoft.

[51] Ethereal, http://www.ethereal.com.

[52] Dave Plonka, FlowScan, http://net.doit.wisc.edu/~plonka/FlowScan/.

[53] IETF Working Group IPFIX (IP Flow Information Export), http://www.ietf.org/html.charters/ipfix-charter.html.

[54] Shareshare, http://www.shareshare.com.

[55] J. Michael, H. Braun and I. Graham, "Storage and bandwidth requirements for passive Internet header traces," Proc. of the Workshop on Network-Related Data Management 2001, Santa Barbara, California, USA, May 2001.

[56] Soon-Hwa Hong, Jae-Young Kim, Bum-Rae Cho, James W. Hong, "Distributed Network Traffic Monitoring and Analysis using Load Balancing Technology," Proc. of 2001 Asia-Pacific Network Operations and Management Symposium, Sydney, Australia, September 2001, pp. 172-183.

[57] David L. Mills, Network Time Protocol, RFC 1305, IETF Nework Working Group (March 1992), http://www.ietf.org/rfc/rfc1305.txt.

[58] K. Thompson, G. Miller, and M. Wilder, "Wide-area internet traffic patterns and charateristics," IEEE Network, vol. 11, no. 6, November-December 1997, pp. 10-23.

[59] IETF Working Group PSAMP (Packet Sampling), http://www.ietf.org/html.charters/psamp-charter.html.

[60] Sharad Agarwal, Chen-Nee Chuah, Supratik Bhattacharyya, and Christophe Diot, "The Impact of BGP Dynamics on Intra-Domain Traffic", Sprint ATL Research Report Nr. RR03-ATL-111377. Sprint ATL. Nov. 2003.

[61] Juergen Quittek, Marcelo Pias, and Marcus Brunner, "Integrating IP Traffic Flow Measurement", Proc. of Workshop on Passive and Active Measurements (PAM2001), April 23-24 2001.

[62] Chuck Fraleigh, Sue Moon, Bryan Lyles, Chase Cotton, Mujahid Khan, Deb Moll, Rob Rockell, Ted Seely, and Christophe Diot, "Packet-Level Traffic Measurements from the Sprint IP Backbone," IEEE Network, 2003.

[63] Remco Poortinga, Remco van de Meent, and Aiko Pras, "Analysing campus traffic using the meter-MIB", Proc. of the Passive and Active Measurement workshop (PAM2002), March 25-27 2002.

[64] CAIDA, "Preliminary Measurement Spec for Internet Routers," http://www.caida.org/tools/measurement/measurementspec/.

[65] Cisco, "Release Notes for Catalyst 4840G SLB Switch for Cisco IOS Release 12.0(13), WT6(1)," http://www.cisco.com/univercd/cc/td/doc/product/l3sw/4840g/ios_12/120_10/rn104231.htm.

[66] Nortel Networks, "Alteon WebOS 9.0," http://www.nortelnetworks.com/products/library/collateral/intel_int/webos.pdf.

[67] Siegfried Lifler, "Using Flows for Analysis and Measurement of Internet Traffic," Diploma Thesis, Institute of Communication Networks and Computer Engineering, University of Stuttgart, 1997.

[68] J.Quittek, T. Zseby, B. Claise, K.C. Norsth, "IPFIX Requirements," Internet Draft, http://norseth.org/ietf/ipfix/draft-ietf-ipfix-architecture-00.txt.

[69] James W. Hong, Soon-Sun Kwon and Jae-Young Kim, "WebTrafMon: Web-based Internet/Intranet Network Traffic Monitoring and Analysis System," Computer Communications, Elsevier Science, Vol. 22, No. 14, September 1999, pp. 1333-1342.

[70] Rambus, RDRAM memory, http://www.rambus.com/technology/rdram_overview.html.

[71] J. Michael, H. Braun and I. Graham, "Storage and bandwidth requirements for passive Internet header traces," Proc. of the Workshop on Network-Related Data Management 2001, Santa Barbara, California, USA, May 2001.

[72] Loris Degioanni, Mario Baldi, Fulvio Risso, and Gianluca Varenni, "Profiling and Optimization of Software-Based Network-Analysis Applications", Proceedings of the 15th IEEE Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2003), Sao Paulo, Brasil, November 2003.

[73] Ahang Shiyong, Wu Chengrong, and Guw Wei, "Network Monitoring In Broadband Network", Web Information Systems Engineering, in Proceedings of the Second International Conference, Vol. 2, pp. 171 - 177, 3-6 Dec. 2001.

[74] C. Estan and G. Varghese, "New directions in traffic measurement and accounting," in Proceedings of the 2001 ACM SIGCOMM Internet Measurement Workshop, pp. 75-80, San Francisco, CA, Nov. 2001.

[75] Luca Deri, Ntop, http://www.ntop.org.

[76] Daniel W. McRobb, "cflowd design," CAIDA, September 1998.

[77] RRDtool, http://www.rrdtool.com. CAIDA.

[78] ARTS++, http://www.caida.org/tools/utilities/arts/, CAIDA.

[79] NetraMet, http://www.caida.org/tools/measurement/netramet/, CAIDA.

[80] DGA card, http://www.endace.com/, ENDACE, Inc.

[81] SAYCLUB, http://www.sayclub.com/.

[82] eDonkey, http://www.overnet.com.

[83] Soribada, http://www.soribada.com.

[84] V-share, http://www.v-tv.co.kr.

[85] FREECHAL, http://www.freechal.com/.

요 약 문

인터넷을 기반으로 하는 IP 네트워크는 현재 지속적인 사용자의 증가와 다양한 서비스 및 응용 프로그램의 출현에 따라 Gbps이상의 고속 네트워크 구축이 일반화 되고 있다. 또한 사용자나 기업의 네트워크에 대한 의존도가 높아지면서 네트워크 트래픽 모니터링은 과거의 네트워크 사용량 파악이나 확장 계획 수립의 전통적인 요구에서부터 종량제 과금, 보안 분석, SLA, CRM등 다양한 분야에서 트래픽 모니터링이 요구되고 있고, 분석 내용 또한 응용 계층의 상세 분석을 요구하고 있다.

현재 네트워크 트래픽 모니터링 및 분석에 있어 중요한 두 가지 문제는 고속 네트워크에서 발생하는 대용량 트래픽의 실시간 처리와 다양한 응용 프로그램 및 서비스들로부터 발생되는 다양한 종류의 트래픽에 대한 응용 계층 상세 분석에 관한 것이다. 본 논문에서는 이 두 가지 문제에 있어서 해결책을 제시하였다. 첫째, 고속 네트워크 실시간 처리에 대하여 많은 실시간 트래픽 모니터링 시스템의 기반 구조인 RTFM 구조의 한계를 극복할 수 있는 유연하고, 확장성 있는 실시간 모니터링 시스템 구조인 NG-MON 구조를 제시하였다. 둘째, 응용 계층의 트래픽 상세 분석을 위해 우선 전제되어야 하는 것이 수집된 트래픽을 응용 프로그램 별로 분류하는 Traffic Identification이다. 현재 많은 인터넷 기반의 응용 프로그램들은 dynamic port를 사용하여 상호 통신을 하고 있으며, 이는 Traffic Identification에 있어 받듯이 해결되어야 하는 일이다. 본 논문에서는 수집된 트래픽의 응용 프로그램을 알아내는 방법론을 제시한다. 이 방법론을 NG-MON 구조를 바탕으로 구현하여 현재 POSTECH에서 발생하는 인터넷 트래픽의 다양한 분석 결과를 제시한다.

먼저 본 논문에서 제시하는 고속 네트워크에 적합한 실시간 트래픽 모니터링 구조 (NG-MON 구조)는 트래픽 분석 과정을 Packet Capture, Flow Generator, Flow Store, Traffic Analysis, Presenter의 5 단계로 세분화하였다. 각 단계에서는 해당 네트워크 링크에서 발생하는 트래픽의 양에 따라 처리 시스템을 쉽게 추가/제거할 수 있도록 로드 분산 구조를 이

용하여 설계하였다. 그리고 각 단계에서 수행 내용을 명확히 구분하여 Pipeline 방식의 트래픽 처리는 단계별 유연성 및 확장성을 높이는데 효과적이다. 또한 패킷 단위 트래픽 처리가 아닌 플로우 단위 트래픽 처리는 고속 네트워크에서 발생하는 대용량 데이터의 압축을 가능케 하였다. NG-MON 구조에 따른 실시간 모니터링 시스템의 구현과 POSTECH 인터넷 망에의 적용은 고속 네트워크에 적합한 실시간 모니터링 시스템의 개발에 유용한 Guideline이 될 것이다.

본 논문에서 제시하는 Traffic Identification 방법은 크게 Application Port Table (APT), Important Port Selection (IPS), Flow Relationship Map (FRM)의 3단계로 이루어진다. APT는 현재 많이 사용하고 있는 응용 프로그램의 이름과 고정적으로 사용하는 port number를 찾아내는 과정이다. IPS는 수집된 트래픽에서 플로우 정보를 생성하고, 각 플로우의 src/dst port들 중에서 응용 프로그램의 이름을 정하는데 중요한 port를 가려내는 과정으로, 특히 TCP 플로우에서 서버의 listening port를 가려내는 방법이다. 여기에서는 TCP communication상에서 나타나는 SYN/SYN-ACK 패킷을 기본으로 이용한다. FRM은 IPS과정을 거친 플로우 정보를 플로우들 간의 의존성을 이용하여 응용 프로그램에 따라 분류하는 최종 단계이다. 이 단계는 플로우의 port와 ip를 기준으로 분류하는 Property Dependency Grouping (PDG)과 PDG group들 사이의 weight를 계산하여 PDG group들을 합하는 Location Dependency Grouping (LDG)의 순차적인 두 단계로 구성된다. 제시된 Flow Grouping Method를 기반으로 90%이상의 Traffic Identification을 달성할 수 있었다. 또한 이 방법론을 NG-MON 구조를 이용하여 구현하고, POSTECH 인터넷 망에 설치하여 학교 인터넷 트래픽의 상세 분석을 수행함으로써, 현재 인터넷 트래픽의 응용 계층 특성의 실시간 분석을 가능케 하였다.

감사의 글

91년 20살의 젊은 나이에 포항공대 전자계산학과에 입학한 이후로 14년이 지났습니다. 이제 저는 이곳에서 박사 졸업과 함께 제 인생의 가장 치열하고 화려했던 지난날을 추억하며, 박사 학위를 받을 수 있게 도와주신 모든 분들께 감사의 마음을 전합니다.

학부 과제 연구 때부터 시작하여 석사, 박사 과정의 8년 동안 저를 한 사람의 당당한 연구자로 만들기 위해 지도 편달을 아끼지 않으신 지도 교수님이신 홍원기 교수님께 가장 먼저 감사를 드립니다. 학부 4학년, 과제 연구 신청을 위해 교수님과 처음 면담을 하던 그때가 오늘의 저를 있게 한 운명의 시작이었습니다. 여러 가지로 많이 부족한 저를 기꺼이 받아들여 주셨고, 모자란 부분은 채워주셨고, 모난 부분은 다듬어 주셨기 때문에 오늘의 제가 있습니다. 지금까지의 교수님의 가르침을 바탕으로 앞으로 사회에 나가 훌륭한 연구자의 길을 갈 것을 약속드립니다. 그리고 저의 박사 논문 심사를 위해 바쁘신 와중에도 기꺼이 시간을 내주시고, 논문에 깊은 관심을 가져주시고 저의 앞날의 발전을 기원해 주신 김치하 교수님, 김종 교수님, 서영주 교수님, 그리고 멀리 경희 대학교에서 오신 홍충선 교수님께도 진심으로 감사를 드립니다. 심사 날 교수님들께서 해주신 많은 충고와 격려의 말씀을 가슴 속 깊이 간직하고, 포항공대 졸업생으로서 부끄럽지 않은 삶을 살도록 최선의 노력을 다 하겠습니다. 그리고, 검도를 통해 인생을 가르쳐 주신 강교철 교수님께 감사와 죄송한 마음을 전합니다.

돌이켜 보면 저의 인생의 1/3 이상을 포항공대에서 보냈고, 포항공대 생활의 2/3를 DPNM 연구실에서 보냈습니다. 그 기간 동안 만들어진 많은 인연들이 순간으로 지워지지 않고 영원히 기억되고 이어지길 간절히 기원합니다. 98학번 석사 동기인 윤규형, 태선형, 숙향, 미정. 처음 DPNM 연구실 생활을 시작하던 그 힘든 시기에 마음 든든한 친구이자 동지였습니다. 특히 같이 박사 과정까지 함께 해 준 미정에게 감사를 드립니다. 7년이란 시간을 보내는 동안 그대는 나에게 든든한 동지가 되

어 주었습니다. 앞으로 어디에서 무엇을 하든 서로에게 힘이 되고 자극이 되는 동지이길 기원합니다. 박사 선배이신 주홍택 박사님, 김재영 박사님께도 감사 드립니다. 박사 초년병일 때 선배님들의 말과 행동에서 많은 것을 배웠음을 선배님이 떠난 후에야 절실히 느꼈습니다. 아마도 제가 박사 과정과 랩장 시기를 무사히 마칠 수 있었던 것은 모두 선배님들의 가르침이 있었기 때문이라고 생각합니다. 그리고 저와 같이 지난날 NG-MON 팀에서 동고동락한 세희, 효진, 훈정, 영미에게 감사 드립니다. 그대들이 있었기에 NG-MON이 DPNM연구실 최고의 작품이 되었다고 믿습니다. 현재 NG-MON 팀을 이끌어가는 성철, 승화, 룡권, 영준, 형조, Deepali에게 감사 드립니다. 그대들이 있어 NG-MON 제 2의 도약은 성공적일 것입니다. 특히 박사 논문 작성에 있어 같이 일하고 밤새워 주었던 영준에게 고맙다는 말을 전하고 싶습니다. 그리고 연구실에서 같이 동고동락하는 은희, 동현, 소정, 선미에게도 앞날에 좋은 결과가 있기를 기원합니다.

믿음 하나로 오랜 시간 기다려주신 우리 가족들. 시골서 농사를 지으면서 자식들만 바라보고 고생만 하신 아버지, 어머니. 오늘의 결과가 부모님의 고생에 조금이나마 보상이 될 수 있을까요? 그렇지만 저를 슬픔에 빠지게 하는 것은 이 기쁨을 어머니와 함께 할 수 없는 것 입니다. 먼 하늘에서나마 이 아들의 성취를 대견스럽게 바라보고 계시리라 믿습니다. 뒤늦게 시골 생활을 시작하시면서 인내로 살아가시는 새 어머니께도 감사를 드립니다. 직장 생활도 그만두시고 저희 아들 재형이를 도맡아 아낌없는 사랑으로 키워주시는 장모님께도 이 자리를 빌어 감사의 마음을 전합니다. 박사 과정 학생을 남편으로 맞아 지금까지 3년 동안의 긴 기다림의 시간을 믿음과 사랑으로 힘든 내색 하나 없이 견뎌준 나의 아내, 지선. 그대는 나에게 어머니가 떠난 이 세상에서 잃어버린 희망을 찾아준 사람입니다. 내 젊은 날 인고와 열정의 결실을 그대와 우리의 사랑스런 아들 재형, 그리고 앞으로 태어날 우리의 희망에게 바칩니다.

이 력 서

성　　명 : 김 명 섭

생년월일 : 1972년 10월 29일 (음력)

출 생 지 : 경북 경주시 강동면

주　　소 : 경기도 안양시 동안구 비산동 한양샛별아파트 104-105

학 력

1991.3 ~ 1998.2 포항공과대학교 전자계산학과 (B.S.)

1998.3 ~ 2000.2 포항공과대학교 컴퓨터공학과 (M.S.)

2000.3 ~ 2004.8 포항공과대학교 컴퓨터공학과 (Ph.D.)

논문 실적

International Journal Papers

1. Hun-Jeong Kang, Myung-Sup Kim, James Won-Ki Hong, “Streaming Media and Multimedia Conferencing Traffic Analysis using Payload Examination, " ETRI Journal, Vol.26, No.3, June 2004, pp.203-217.
2. Myung-Sup Kim, Mi-Joung Choi and James W. Hong, "A Load Cluster Management System using SNMP and Web", International Journal of Network Management (IJNM), Vol. 12, No. 6, November 2002, pp. 367-378.
3. Myung-Sup Kim and J. Won-Ki Hong, "A CORBA-based Framework for the Management of Multimedia Services, Interoperable Communication Networks", Baltzer Science Publishers, i Vol. 2, No. 2-4, January 2000, pp. 189-198.

4. J. W. Hong, Y. M. Shin, M. S. Kim, J. Y. Kim and Y. H. Suh, "Design and Implementation of a Distributed Multimedia Collaborative Environment", Cluster Computing, Baltzer Science, Vol. 2, No. 1, January, 1999, pp. 45-49.

International Conference Papers

5. Myung-Sup Kim, Hun-Jeong Kang, Seong-Cheol Hong, Seung-Hwa Chung, James W. Hong, "A Flow-based Method for Abnormal Network Traffic Detection", Accepted to appear in the Proc. of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2004), Seoul, Korea, April 2004, pp. 599-612.

6. Myung-Sup Kim, Hun-Jeong Kang and James W. Hong, "Towards Peer-to-Peer Traffic Analysis Using Flows", Lecture Notes in Computer Science 2867, Edited by Marcus Brunner, Alexander Keller, 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2003), Heidelberg, Germany, October, 2003, pp. 55-67. (SCIE).

7. Hun-Jeong Kang, Myung-Sup Kim and James Won-Ki Hong, "A Method on Multimedia Service Traffic Monitoring and Analysis", Lecture Notes in Computer Science 2867, Edited by Marcus Brunner, Alexander Keller, 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2003), Heidelberg, Germany, October, 2003, pp. 93-105. (SCIE).

8. Hun-Jeong Kang, Seung-Hwa Chung, Seong-Cheol Hong, Myung-Sup Kim and James W. Hong, "Towards Flow-based Abnormal Network Traffic Detection", Proc. of 2003 Asia-Pacific Network Operations and Management Symposium (APNOMS 2003), Fukuoka, Japan, October 1-3, 2003, pp. 369-380.

9. Se-Hee Han, Myung-Sup Kim, Hong-Taek Ju and James W. Hong, "The Architecture of NG-MON: A Passive Network Monitoring System", Lecture Notes in Computer Science 2506, Edited by M. Feridun, P. Kropf and G. Babin, 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2002), Montreal, Canada, October, 2002, pp. 16-27. (SCIE).

10. Hyo-Jin Lee, Myung-Sup Kim, James W. Hong and Gils-Haeng Lee, "Mapping between QoS Parameters and Network Performance Metrics for

SLA monitoring", Proc. of 2002 Asia-Pacific Network Operations and Management Symposium (APNOMS 2002), Jeju, Korea, September 25-27, 2002, pp. 97-108.

11. Se-Hee Han, Hong-Taek Ju, Myung-Sup Kim and James W. Hong, "Design of Next Generation High-Speed IP Network Traffic Monitoring and Analysis System", Proc. of 2002 Asia-Pacific Network Operations and Management Symposium (APNOMS 2002), Jeju, Korea, September 25-27, 2002, pp. 282-293.

12. Hun-Jeong Kang, Hong-Taek Ju, Myung-Sup Kim and James W. Hong, "Towards Streaming Media Traffic Monitoring and Analysis", Proc. of 2002 Asia-Pacific Network Operations and Management Symposium (APNOMS 2002), Jeju, Korea, September 25-27, 2002, pp. 503-504.

13. Myung-Sup Kim, Mi-Jeong Choi and James W. Hong, "Highly Available and Efficient Load Cluster Management System using SNMP and Web", Proc. of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2002), Florence, Italy, April 2002, pp. 619-632.

14. Myung-Sup Kim, J Won-Ki Hong, "SNMP & Web-based Load Cluster Management System" 2001 Asia-Pacific Network Operations and Management Symposium (APNOMS 2001), Sydney, Australia, September 26-28, pp. 268-279.

15. Jae-Young Kim, Myung-Sup Kim and Won-Ki Hong, "Management of Differentiated Services Using the SNMP Framework," Proc. of the 2nd International Conference on Advanced Communication Technology (ICACT 2000), Muju Korea, February 16-18, 2000, pp.624-629.

Domestic Journal Papers

16. Hyo-Jin Lee, Myung-Sup Kim, James W. Hong and Gil-Haeng Lee "QoS Parameters to Network Performance Metrics Mapping for SLA Monitoring", KNOM Review, Vol. 5, No. 2, December 2002, pp. 42-53.

Domestic Conference Papers

17. 강훈정, 김명섭, 홍원기, "멀티미디어 서비스 트래픽 모니터링과 분석

", Proc. of KNOM 2003 Conference, Daejeon, Korea, May 22-23, 2003, pp. 201-209.

18. 김명섭, 강훈정, 홍원기, "Flow Grouping을 통한 P2P 트래픽 분석 방법에 관현 연구", Proc. of KNOM 2003 Conference, Daejeon, Korea, May 22-23, 2003, pp. 210-218.

19. 김명섭, 홍원기, " SNMP와 Web 기반의 Load Cluster 관리 시스템의 설계 및 구현", Proc. of KNOM 2001 Conference, Daejeon, Korea, May 24-25, 2001, pp. 163-169.

20. 신영미, 김명섭, 홍원기, 서영호, 김용, "W/S 기반의 원격 공동연구 플랫폼 설계 및 구현", 1998 한국감성과학회 춘계학술회의, 대전, May 1998, pp. 298-303.

21. Myoung-Sup Kim, Won-Ki Hong, "Development of DMI MIF Decoder and Browser", (in Korean), Proc. of the 24th KISS Fall Conference, Seoul, Korea, October 1997, pp. 349-352.