Application-Level Traffic Monitoring and an Analysis on IP Networks

Myung-Sup Kim, Young J. Won, and James Won-Ki Hong

Traditional traffic identification methods based on well-known port numbers are not appropriate for the identification of new types of Internet applications. This paper proposes a new method to identify current Internet traffic, which is a preliminary but essential step toward traffic characterization. We categorized most current network-based applications into several classes according to their traffic patterns. Then, 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, a traffic analysis system, to analyze Internet traffic between our enterprise network and the Internet, and characterized all the traffic according to their application types.

Keywords: Passive traffic monitoring and analysis, application-level traffic identification, application-level traffic characterization, Internet traffic, streaming traffic, peer-to-peer traffic.

Manuscript received Apr. 8, 2004; revised Aug. 10, 2004.

This work was in part supported by the Electrical and Computer Engineering Division at POSTECH under the BK21 program of Ministry of Education, and the Program for the Training of Graduate Students in Regional Innovation of Ministry of Commerce, Industry and Energy of the Korean Government.

Myung-Sup Kim (phone: +82 54 279 5654, email: mount@postech.ac.kr), Young J. Won (email: yjwon@postech.ac.kr) and James Won-Ki Hong (email: jwkhong@postech.ac.kr) are with the DPNM Laboratory, POSTECH, Pohang, Korea.

I. Introduction

In addition to the fundamental and traditional purposes of traffic analysis such as network planning, network problem detection, and network usage reporting, traffic monitoring and analysis is required in many areas to improve network service quality such as in abnormal traffic detection and usage-based accounting. However, to come up with an evolution of the Internet in terms of underlying technologies and user services, 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 problem 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 a real-time manner [1]-[5]. The other problem 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 [6]-[8].

Regarding the second problem, the types and patterns of current network traffic are more complex than they were in 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 port-based traffic types is decreasing. Instead, P2P, streaming media, and game traffic are increasing. Internet2 administrators report that about 4% of the traffic carried by their network is P2P traffic, while a further 54% of unidentified traffic most likely belongs to P2P applications [9]. The amount of P2P application traffic in many ISPs is reported to be greater than 50% of total traffic [10]-[12]. The difficulty with current traffic analysis is that the traditional method is inadequate to analyze this newly emerging traffic. We need a new method to analyze application traffic. The

features of the newly emerging network-based applications are as follows:

- A large number of different Internet-based applications have been developed and widely used. The number of these applications will increase continuously in the future.
- 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 [13], [14] 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 internet applications use ephemeral port numbers greater than 1024 as a default application port. As of January 2003, the proportion of TCP traffic using ephemeral ports is more than 40% of the total TCP traffic at a large ISP backbone network [11]. In our investigation, the number of TCP sessions using ephemeral ports as a server port is more than 50% of the total TCP sessions in the POSTECH campus network.
- The default port numbers for many applications are not registered on the IANA port list [15]. However, the developers of applications for regional users usually do not register their port numbers to IANA.
- 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.
- Most newly emerging applications use multiple sessions to communicate with each other. For example, streaming media applications establish two or more sessions to transfer control data and multimedia data. Sometimes they use both the transmission control protocol (TCP) and user datagram protocol (UDP) simultaneously.
- Many P2P and streaming media applications use dynamically determined ports to communicate between peers. In the case of streaming media traffic, the port number and protocol for the delivery of multimedia data are decided by the negotiation between the client and server. The port number for a file transfer in the MSN instant messaging application is also determined dynamically. Some of them use dynamic port numbers to evade detection and control [7], [8].

All of the above features of the 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. How can we identify individual applications from IP traffic? And, what are the characteristics of the current IP traffic at the application level?

Traditional traffic identification methods, based on well-known port numbers and the IANA port list, are not suitable for determining the traffic from P2P, streaming, and other new applications. This paper proposes a noble method to identify the recent Internet traffic. First, we categorize most of the current network-based applications into several classes according to their traffic patterns. Using this categorization, we developed a new method called flow-grouping, which determines the application name of individual packets. The flow-grouping method consists of three steps. The first step is to build the application port table (APT) by an exhaustive off-line search of applications. The second step is the important port selection (IPS) from each flow record. It determines the important port number between source and destination port numbers in each flow record. The third step is the flow relationship map (FRM), which aggregates flows according to their inter-dependency and decides a corresponding application name to each flow. To validate the proposed algorithm, we have designed and implemented a traffic analysis system, called NG-MON [1]. NG-MON is currently deployed at the Internet junction of POSTECH and provides us with important characteristics of Internet traffic captured between our campus and the Internet.

The organization of this paper is as follows. Section II describes previous approaches on the identification and characterization of Internet traffic and explains their advantages and disadvantages. In section III, we present our traffic identification method. Section IV addresses the design and implementation issues of our prototype system. In section V, we describe the deployment of the proposed method and summarize the Internet traffic characteristics as an analysis result. Finally, section VI concludes the paper with possible future work.

II. Related Work

In this section, we describe related research work on application-level traffic identification. We also introduce some research results about recent Internet traffic characteristics.

1. Application-Level Traffic Identification

We consider two steps to determine the original 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, and Telnet use a well-known port that is below 1024. This simplifies determining the important port for 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 easy to decide an important port for packets that have both ports over 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.

A. Traditional Application-Level Traffic Identification Method

The traditional method is based on the well-known ports registered to the IANA port list [15]. 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 the many new features in recent Internet traffic. For example, this method cannot detect the dynamic port numbers generated by streaming media applications such as Microsoft Windows Media Server/Player [16]. The traditional method also cannot distinguish the traffic between two different applications using the same port numbers simultaneously. If the target port number is not present in the IANA port number list, then this method is simply not applicable.

B. Payload-Examination-Based Method

To detect the dynamically determined ports of streaming media applications and P2P applications, the payload examination method is one possibility. The tools mmdump [17] and SM-MON [7], [8] used this method to differentiate streaming media traffic from other Internet traffic. In a streaming media service, two types of sessions are established between the client and server: a control session and a data session. The port number of the data session is determined dynamically by the negotiation between the client and server using the pre-established control session. In addition, it is also useful to detect a data session of passive FTP traffic that shares a similar fashion with streaming media.

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 the 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.

C. Signature-Mapping-Based Method

To increase identification accuracy, the signature-mapping-based method [18], [19] 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. This portion of payload data is 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 might also cause greater system overhead in the process of packet capture and payload comparison. However, some researches [18] have high hope to overcome such limitations, and they are actively exploring more advanced techniques in signature mapping.

D. Methods Used for P2P Traffic Identification

The recent tendency in Internet traffic is a shift from web traffic to P2P traffic. Many studies focus on the characterization of P2P traffic [10]-[14]. 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) [10]. A research on Internet content delivery systems also distinguished traffic type by the default port numbers of each application and the server IPs providing the corresponding services [19]. However, new versions of the KaZaA system do not use default port number (1214) any longer. They use dynamically assigned port numbers to transfer data between peers. Furthermore, the port number used for a file transfer between peers in the MSN messenger application changed from a fixed port (6981) to a dynamic port. Recent research [11] 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 100 KB in less than 30 minutes. They showed that the TCP-Big traffic had almost the same properties as P2P traffic.

2. Traffic Characterization of Recent Internet Traffic

In this section, we present some related studies on traffic characterization. Many recent studies addressed the IP traffic characterization. They collected and analyzed the IP traffic from an enterprise network [20] or a large ISP backbone network [21]. They focused on traffic characteristics from various perspectives, such as user perspective characteristics, packet-level and flow-level features [21], [22], routing protocol-level behaviors [23], and so on. Concerning application-layer traffic characterization, a sizable amount of recent studies concentrates on P2P traffic. A number of recent studies [10], [24]-[30] 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 [11], [12] focused on the comparison of P2P traffic with other traditional Internet traffic, such as WWW and FTP.

A study [10] 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 [12], [13] 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 the US and Europe [13], or from the University of Washington campus to its ISP [12]. They reported that KaZaA traffic constitutes the most Internet traffic and a tiny number of files generate most of the 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 [13]. 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 paper 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.

III. Application-Level Traffic Identification

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

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.

Table 1 shows a list of current Internet-based applications that generate the most Internet traffic. We categorized these applications according to their service type and protocol. The traditional applications use standard and open protocols that simplify identification of their traffic. The others are newly emerging applications that began a few years ago. The application layer protocols RTSP, SIP, Q.931, and H.245 are

Table 1. Current Internet-based applications.

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 2. Classification of communication behaviors.

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)

used to deliver control data between a streaming server and client. Protocols RTP/RTCP and MMSU/MMST are used to transfer multimedia data from a server to a client. The Quicktime and Real Networks streaming services use open protocols called RTSP and RTP/RTCP. However, the Microsoft windows media service [16] uses a proprietary protocol, MMS. The most popular peer-to-peer applications are file sharing applications, such as Fasttrack (KaZaA, KaZaA Lite) [13], Gnutella (BearShare, Bnucleus, Morpheus, LimeWire) [28], and instant messaging applications (MSN messenger, etc.). Many game applications use a network to make multi-user games interactive. In addition, Internet disk service provides another type of file sharing method. The service provider constructs a large file server at Internet data centers (IDC). A user can use a portion of the disk space and share it with other users by paying a fee.

We investigated the communication behavior and 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 system using ethereal and tcpdump and examined their behaviors, especially the source and destination port numbers, peer 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 2. We used three types of information to categorize the communication behaviors: the number of sessions among the involved systems, the way of selecting a port number, and the number of involved systems to provide a service. We describe the details of each type below.

A. Type S-F-2

Type S-F-2 is the communication behavior of most traditional Internet-based applications, such as web, telnet, ssh, smtp, etc. The communication structure is a client/server architecture and uses a well-known single fixed port number. In Type S-F-2, a server can simultaneously communicate to multiple clients. However, these communications are independent of each other from the client's perspective. No communication occurs between clients; instead, data are transferred only between a client and server. Traffic of this type is easily determined. The port number used by each application is a clue for application identification: 80 for http, 23 for telnet, 22 for ssh, 25 for smtp, and so forth.

B. Type M-F-2

The FTP communication in active mode is a good example of type M-F-2. The FTP service use two different sessions: a control session and a data session. In active mode FTP, the fixed well-known port number 21 is used for the control session. The FTP service uses port number 20 to transfer data. The identification of this type is the same as the previous 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 a TCP session with port number 21 simultaneously appears between the same two hosts. We can use the IANA port list to determine this type of traffic. For example, the RealVNC application uses port numbers 5800 and 5900, while MS-SQL applications use port numbers 1433 and 1434.

C. Type M-D-2

This type of communication behavior uses multiple sessions between two hosts like type M-F-2. However, the port numbers for one or more sessions are dynamically determined by the negotiation between two involved hosts. The best examples of this type of communication are FTP traffic in passive mode and streaming media traffic. The dynamically generated session can use TCP or UDP according to the

applications. Typically, streaming media applications use UDP sessions to deliver multimedia data from a media server to a client. The identification of the dynamic session of this type is complex. We can use the payload examination method used in mmdump [17] and SM-MON [7], [8]. Otherwise, we can consider a heuristic method such as Flowscan [31], a flow-based traffic analysis system.

D. Type M-F-3

Many new applications use multiple sessions and communicate with multiple peers simultaneously. P2P file sharing applications, P2P instant messaging applications, and game applications are the best examples. Figure 1 illustrates two prevalent types of P2P communication architectures used by these applications: the *central arbiter type* and the *pure distributed type*. As illustrated in Figure 1, a peer communicates with other peers or central servers simultaneously, which establishes multiple sessions. Some P2P applications and game applications use fixed port numbers for these multiple sessions and both TCP and UDP.

This type of communication is derived from the behavior of these applications. The multiple sessions with fixed port numbers among many peers are features of this type of communication. Although multiple sessions are involved, the use of fixed port numbers makes the determination of this type M-F-3 traffic straightforward.

E. Type M-D-3

The early versions of the MSN messenger application 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 transfers. In addition, the earlier version of KaZaA [13] used port number 1214 for data transfers, but the latest version of KaZaA can allocate the server port number dynamically as needed. In this type of communication behavior, a single host establishes multiple sessions with more than three peers. Some of those sessions are established using dynamically determined port numbers. Many P2P applications and game applications are developed using this communication type. The traffic type M-D-3 is the most difficult to identify. Therefore, we need an intelligent and efficient method to identify Internet traffic at the application level. The communication behavior of many Internet-based applications is shifting from type M-F-3 to type M-D-3. One main reason for this migration is to evade detection and control.

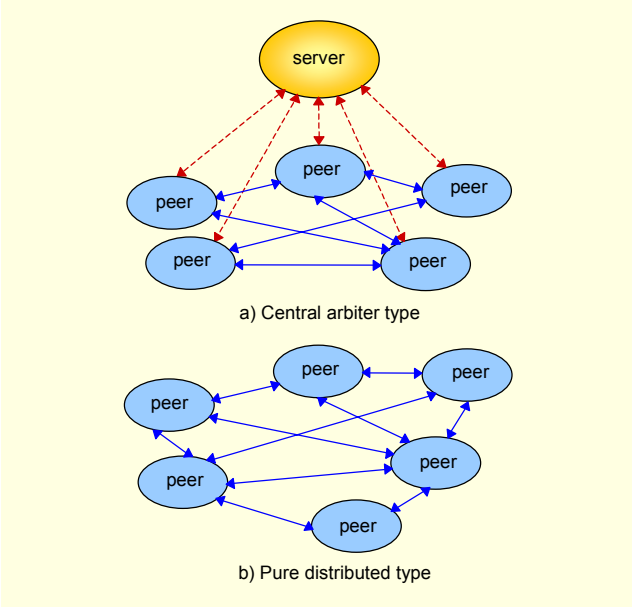


Fig. 1. Peer-to-peer communication architecture.

2. 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 the 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 be easily grouped according to their default port numbers. However, in the case of type M-D-3 traffic (ex., P2P traffic), flow grouping is not as simple as web traffic because they use port numbers greater than 1024 and many port numbers are dynamically assigned. If all Internet traffic can be grouped according to their applications, then Internet traffic analysis and characterization can be performed with high accuracy.

Our proposed method consists of three steps, as illustrated in Fig. 2. These three steps implement the application port table (APT), important port selection (IPS), and flow relationship map (FRM), respectively In our proposed method, we do not examine the payload of each packet. Instead, we use only the packet header information after aggregating them into flows, as described in section II.

The first step of the proposed method is to construct an APT. The APT is constructed by an exhaustive off-line search of each application using packet analysis tools. The APT contains the application name, its frequently used port numbers excluding the 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 IPS. The input of IPS may either be a raw packet or flow data. The outputs of IPS are flow data and

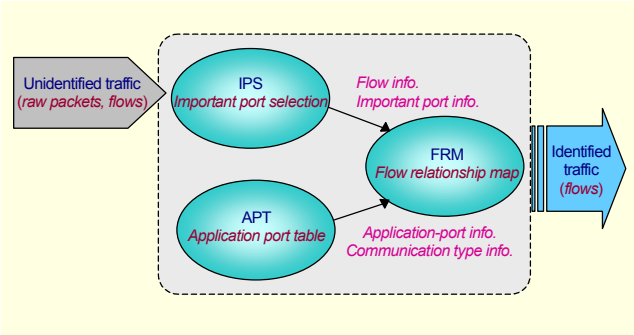


Fig. 2. Overall process of the traffic identification method.

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. Then, we select the important port number from the flow information for 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 the FRM. The flow data, the output from the IPS, is the input into the FRM. The FRM groups flows according to their relationships and marks flows with the corresponding application name. Most of the newly emerging applications use multiple connections with a single or multiple peer(s) to perform various functions; therefore, it is possible to discover the 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 destined to belong to a single group by the FRM. After this grouping process, the FRM determines the application name of each flow group using APT information.

3. Application Port Table

The first step of the proposed method is to construct an APT. To decide the application name from captured flow information, we perform the preliminary examination 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. Using this investigation, we construct an APT that contains the information about each application. The APT contains the application name, frequently used TCP/UDP port numbers, one representative port number, and its communication type. The APT is an extension of the IANA port list containing additional information. We examined more than 100 popular applications and constructed an APT, a small portion of which is shown in Table 3.

We also record the communication type of each application according to the classification in Table 2. 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 along with many different port numbers, only one representative port number is assigned to one application. This representative port number is used to indicate the group of flows.

4. Important Port Selection

In this section, we describe the proposed algorithm to select the important port number among source and destination port numbers in each flow data. The important port number is the

Table 3. An example of an application port table.

Application name	Representative port	TCP well-known ports	UDP well-known ports	Communication type
WWW	80	80, 8080, 443		Type S-F-2
FTP	21	20, 21		Type M-D-2
MSN Messenger	1863	1863, 6981-6990, 14594		Type M-D-3
Windows Media	1755	1755		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
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)

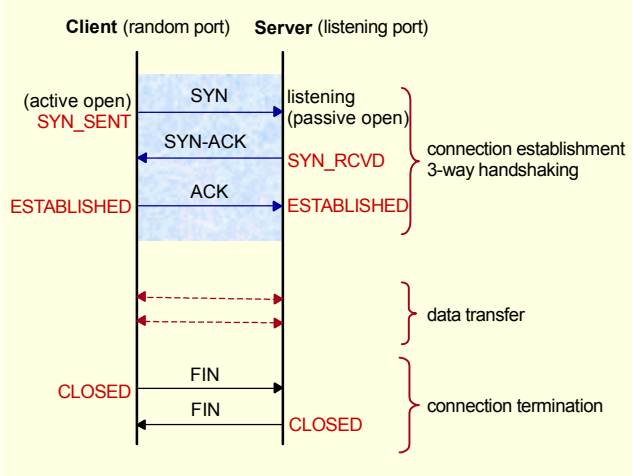


Fig. 3. TCP communication sequence.

port number that is necessary for traffic identification. The proposed IPS algorithm considers only TCP flows. Figure 3 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 the important port for identifying TCP traffic. 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, we can determine the important port number from all TCP flows. It is important to check both the SYN packet and SYN-ACK packet because in a real Internet environment the SYN packets without SYN-ACK packets are captured frequently. 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.

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

We should consider various cases that can arise in a real-world environment. First, we might not 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 required. Sometimes, unintentional packet drops may occur due to asymmetric routing and the performance limitations of a capturing system. Further, we should consider that long lasting TCP sessions, a streaming media service, VoIP service, or network game connections may continue for a long period (e.g., more than 30 minutes). The SYN and SYN-ACK packets appear only at the beginning of a TCP connection. In case the IPS module starts to capture packets from the middle of these connections, we cannot determine the important port for these flows. For the third case, we should consider the flow data, such as Cisco NetFlow data, as an input to our IPS module in order to extend our method to various environments. In this case, the flow information might not have TCP flag information as a clue to identify the important port; for example, Cisco NetFlow V5 data does not have this information. To solve these problems in our IPS, we use the following five points of Assumption 1.

Assumption 1.

a. If the important port of TCP flow f_a is determined, then the important port of its reverse flow ${}_r f_a$ is also determined.
b. If the important port of TCP flow f_a is the source port, then the important port of TCP flow f_b with the same source port and same source IP address as flow f_a is the source port.
c. If the important port of TCP flow f_a is the destination port, then the important port of TCP flow f_b with the same destination port and same destination IP address as flow f_a is the destination port.
d. If TCP flow f_a and TCP flow f_b are different and both have the same source port and source IP address, then the important ports of flows f_a and f_b are the source port.
e. If TCP flow f_a and TCP flow f_b are different and both have the same destination port and destination IP address, then the important ports of flows f_a and f_b are the destination port.

The first three assumptions are obvious. The last two also hold in general networks because they are typical outcomes from TCP connections. Using the combination of the IPS algorithm and Assumption 1, we could determine the important port of 99.99% of the total TCP flows at the POSTECH Internet Junction. The remaining 0.01% of TCP flows is considered abnormal traffic caused by DoS/DDoS attacks or Internet worms. In the case of UDP flows, we cannot apply this type of method because it does not use a 3-way handshaking mechanism like TCP. Instead, we use the FRM algorithm directly to group UDP flows, as described next.

5. Flow Relationship Map

Using the FRM, we can group the TCP and UDP flows according to the corresponding applications, and we can

Table 4. Notation for FRM.

Notation	Description
$f(sip, dip, dip, dport, proto)$	A flow with *sip, sport, dip, dport, proto*; a **long form** of a flow notation
f_a	A **short form** of a flow notation
$f_a\,(sip), f_a\,(sport), f_a\,(dip), f_a\,(dport), f_a\,(proto)$	The *sip, sport, dip, dport, proto* of a flow f_a
${}_r f_a$	The **reverse flow** of a flow f_a : $f_a\,(sip) = {}_r f_a\,(dip), f_a\,(sport) = {}_r f_a\,(dport), f_a\,(dip) = {}_r f_a\,(sip), f_a\,(dport) = {}_r f_a\,(sport), f_a\,(proto) = {}_r f_a\,(proto)$
$f_a = f_b$	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.

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).

We define some basic notations to describe PDG and LDG, which are illustrated in Table 4. 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. The short form of flow notation makes it easy to read equations that include many flows. The reverse flow of flow f_a is denoted as ${}_r f_a$. We denote the complete equality of the two flows f_a and f_b with $f_a = f_b$. The two flows f_a and f_b have a conditional equality, if and only if some of the 5-tuple values of f_a are equal to those of f_b. Generally, we use the following notation to describe the conditional equality: $f_a = f_b \mid (x, y, \ldots)$. The values of x and y can be any of the 5-tuple notations (*sip*, *dip*, *sport*, *dport*, *proto*).

A. 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 is described in Assumption 2. The three points in Assumption 2 are the general property dependencies that are commonly applied to UDP and TCP flows. We use them to assign the flows to groups in PDG.

Assumption 2.

a. If flow f_a belongs to application group $G(A_a)$, then the reverse flow ${}_r f_a$ of flow f_a also belongs to the same application group $G(A_a)$:

$$\text{If and only if } f_a \in G(A_a), \text{ then } {}_r f_a \in G(A_a).$$

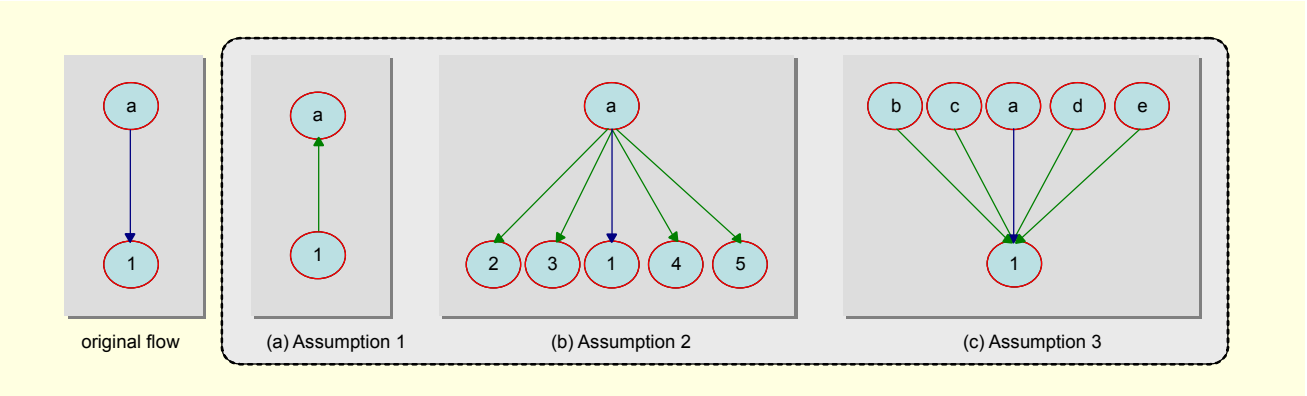
b. If flow f_a belongs to application group $G(A_a)$ and flow f_b has a conditional equality to flow f_a with *sport*, *sip*, and *proto*, then flow f_b also belongs to the same application group $G(A_a)$:

$$\text{If } f_a \in G(A_a) \text{ and } f_a = f_b \mid (sip, sport, proto), \text{ then } f_b \in G(A_a).$$

c. If flow f_a belongs to application group $G(A_a)$ and flow f_b has a conditional equality to flow f_a with *dport*, *dip*, and *proto*, then flow f_b also belongs to the same application group $G(A_a)$:

$$\text{If } f_a \in G(A_a) \text{ and } f_a = f_b \mid (dip, dport, proto), \text{ then } f_b \in G(A_a).$$

Figure 4 illustrates Assumption 2 as a diagram of flows. Assumption 2(a) is obvious, as shown in Fig. 4(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, the same source IP address, and the same protocol number, then the two flows belong to the same application by Assumption 2(b), as illustrated in Fig. 4(b). In addition, flows with the same destination port number, the same destination IP address, and the same protocol number belong to the same application by Assumption 1(c), as illustrated in Fig. 4(c).



Filg. 4. Flow diagram for Assumption 2.

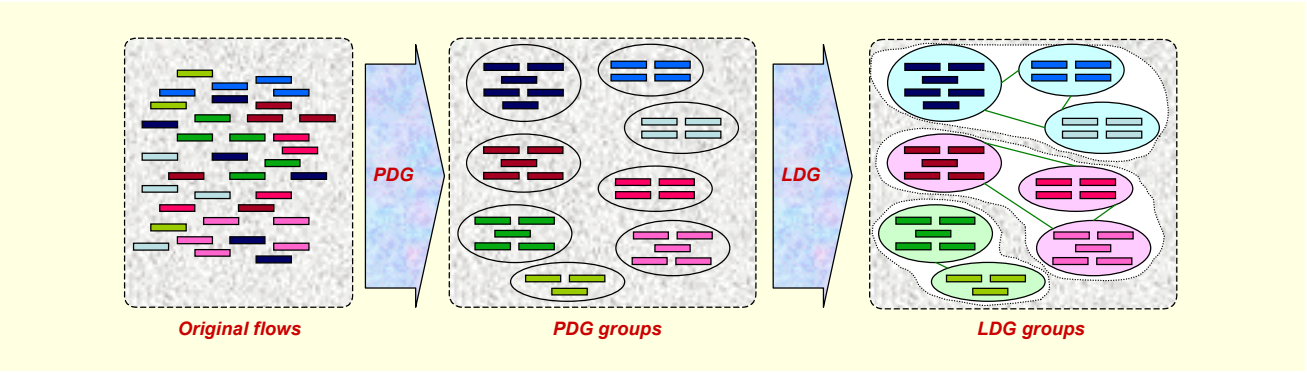


Fig. 5. Flow grouping with PDG and LDG.

Assumption 2 is apparent and provides an accurate grouping of flows. The flows grouped by the PDG algorithm are from the same applications. The process of flow grouping by PDG is as follows. First, we select one flow among the captured flows. Second, we choose 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 steps. This recursive flow selecting process continues until no flow is selected any further. After finishing the grouping of flows from the first selected flow, we select another flow from the remaining flows and continue the same steps until no flow remains.

We can apply this general PDG algorithm without modification to the UDP flows. However, considering the TCP flows, we need a little modification to the original PDG algorithm because the client side program in a TCP connection cannot establish multiple connections using a single client port, while the server program can establish multiple connections using a server port. To improve the capacity of the original PDG algorithm, we applied it to the result of the IPS algorithm that is equal to the flows whose important port numbers are determined. Therefore, we modified the original PDG algorithm. When we applied our proposed PDG algorithm to IP traffic in a POSTECH Internet junction, we were able to reduce about 100,000 of the concurrent flows into 500 PDG groups.

B. 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 high probability that they originated from the same application. However, we cannot guarantee that the flows generated by a single application are grouped into a single group by the PDG algorithm. Many applications, such as P2P applications, use multiple port numbers, and sometimes 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 applications currently use multiple TCP port numbers or dynamically assigned TCP port numbers.

To solve this incomplete solution of the PDG algorithm, we

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 Fig. 5. The PDG algorithm classifies flows into a number of groups. Then, using the LDG algorithm, the PDG groups generated by the same application are grouped into one single group.

To combine inter-related PDG groups, we use the weight concept between them. We calculate the weight values of every PDG group pair. After we obtain the values, we merge the PDG groups by their inter-relationship. If the weight value of two PDG groups is greater than a specified threshold value, then the two PDG groups are joined into a single group. As illustrated in Fig. 5, the eight different PDG groups are merged into the final three LDG groups by the LDG algorithm.

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 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 because the two different flows between two hosts have a high possibility to be the same application traffic. The conditions when the value is 100 cover the type M-F-2 and type M-D-2 cases of the APT. The weight value is 10 when two flows are located among three different hosts. In this case, two flows have a slight possibility to be generated by a single application. The cases when the value is 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:

$$W(G_a,G_b)=\sum w(f_a,f_b),$$

where $f_a \in G_a$ and $f_b \in G_b$ 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 rule:

$$\text{If } \max\left(\frac{W(G_a,G_b)}{n(G_a)},\frac{W(G_a,G_b)}{n(G_b)}\right) \geq threshold,$$

where $n(G_a)$ is the number of flows in G_a, then G_a and G_b can be merged into a single group.

Currently we are using 50 as a threshold, which was determined from many experiments with IP traffic from the POSTECH Internet junction. We may have to use a different threshold value to apply the LDG algorithm to other network environments. However, this equation works efficiently in the current POSTECH network environment. By the LDG algorithm, we can merge all PDG groups into LDG groups, as in Fig. 5. When we applied our proposed algorithm to POSTECH Internet traffic, we could classify about 100,000 concurrent flows into 150 concurrent LDG groups on average. The details of our analysis results are described in section V.

During the LDG process, we use the information of the APT to decide the application name of each flow. Before the above 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 be easily 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 type M-D-2 and type M-D-3. The flows in the selected PDG and LDG groups are tagged with the corresponding representative port numbers. 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 investigate the new application because we have many clues in the undetermined LDG groups. The newly investigated application information is added to the APT, and from that time on, 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 popular applications.

IV. Design and Implementation of the Application-Level Traffic Analysis System

In this section, we describe the design and implementation of the application-level traffic analysis system using the proposed method.

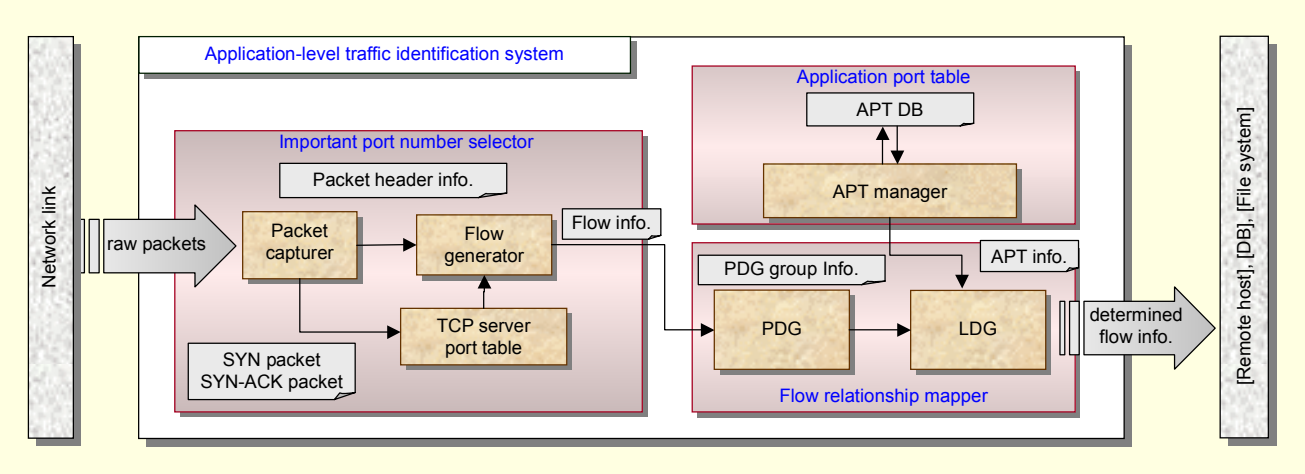


Fig. 6. Overall architecture of the application-level traffic analysis system.

1. Design of the Application-Level Traffic Analysis System

Figure 6 illustrates the overall design of the application-level traffic analysis system, which consists of three main modules. They are the APT module, the IPS module, and the 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. 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 and the TCP server port table to improve the search operation.

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 all of the raw packets, the IPS module can be implemented in a single system. However, multiple capture systems are necessary occasionally, such as when the target link speed is high or when we have a number of target links to be captured. In that case, the IPS module should be separated into two levels: the first-level IPS and second-level IPS. While the first-level IPSs are only responsible for their assigned links, the second-level IPSs collect the results of first-level IPSs and merge them.

The important port-determined flows are sent from the IPS module to the FRM module. The PDG module of the 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 can be stored in a file system or DB, or be sent to the other analysis system for subsequent analysis.

Through the off-line search of each application, we built an application port configuration file using 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 7 shows an example of an application port configuration file. The APT manager reads this configuration file and retains port group information. When the LDG module receives PDG groups from the PDG module, it looks up the 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.

```
<?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 –>
</apt-config>
```

Fig. 7. An example of an application port configuration file.

2. NG-MON

NG-MON [1] is a real-time Internet traffic monitoring and analysis system for high-speed networks developed at POSTECH. We have implemented our application-level traffic analysis system as an essential part of NG-MON. Figure 8 illustrates

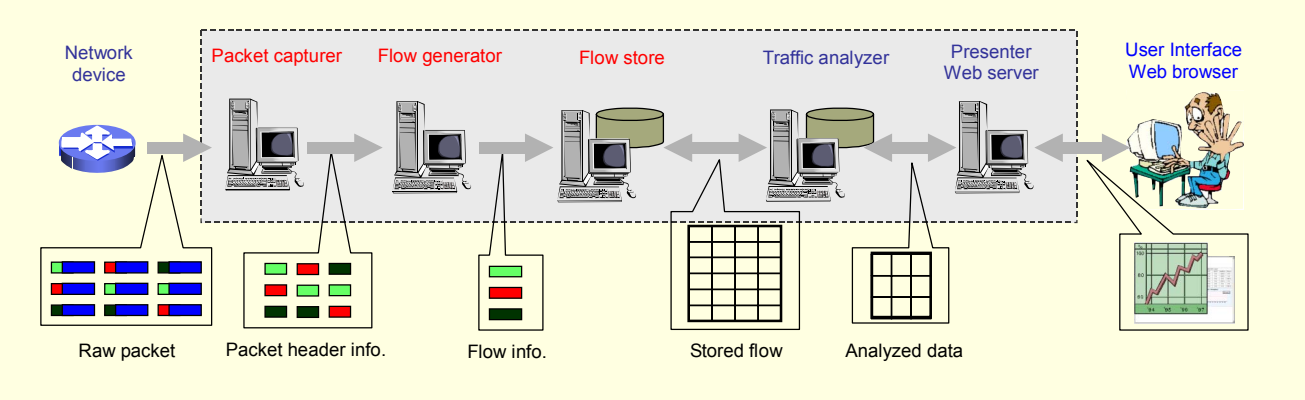


Fig. 8. Overall architecture of NG-MON.

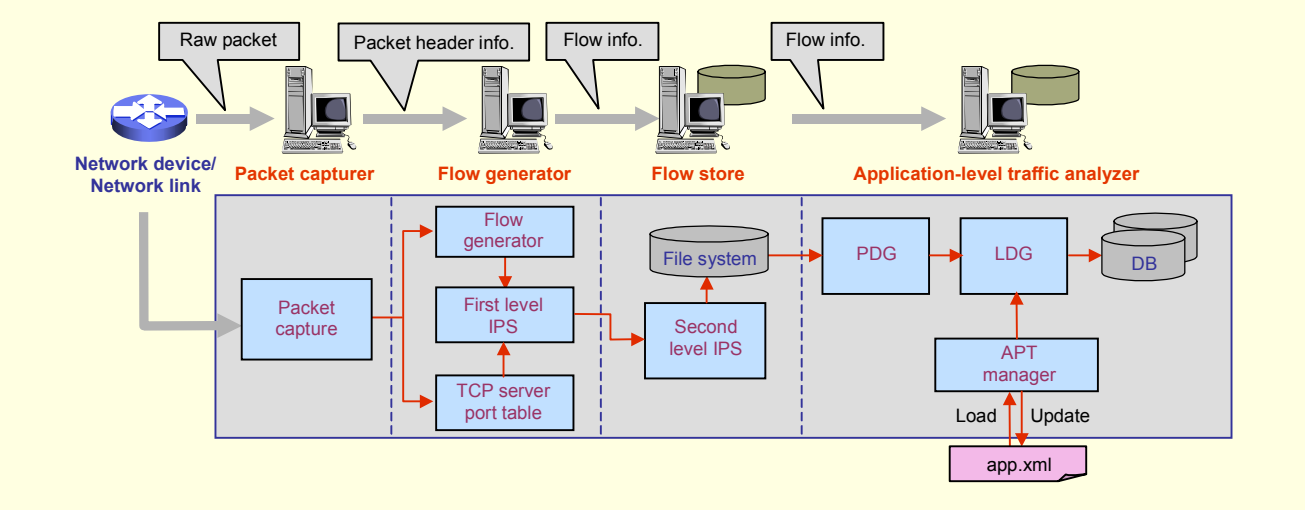


Fig. 9. Integration of the application-level traffic analysis system with NG-MON.

the clustering and pipeline architecture of NG-MON. The analysis task of NG-MON is divided into five phases: packet capture, flow generation, flow store, traffic analysis, and presenter.

The raw packets are captured in the packet capture phase. The packet header information extracted from each raw packet is delivered to the second phase, the flow generation phase. The flow information is generated in the flow generation phase. The fragmented IP packets are reassembled into a single packet in this phase before being added to the corresponding flow record. The flow information is stored in the flow store phase. The traffic analyzer queries the flow store, analyzes the fetched flow data for their analysis purpose, and stores analyzed data into a DB or file system. The presenter provides the analysis results to users using a Web interface.

3. Integration of Application-Level Traffic Analysis System with NG-MON

As mentioned previously, the application-level traffic analysis module is implemented as a plug-in module to NG-MON. Figure 9 illustrates the integration of the application-level traffic analysis module with the current NG-MON system.

The components of the IPS module are broken 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 Fig. 9. The important port-number-determined 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 and FRM module are running. The analyzer receives the flow data from the flow store using the same approach as the other analyzers. The result of the application-level traffic analyzer, the tagged flow information with the representative port number, is stored in the database, which may be used by the presenter or other analyzers. The presenter and other analyzers 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 destroying 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 results, we used MySQL Database because it is the fastest in data storing and retrieving among several freely available database systems. We designed the packet capture system with a multi-threaded architecture to handle multiple network interface cards 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 to eliminate the possibility of data loss in the delivery of data between phases and made use of a number of hash tables to effectively perform the PDG and LDG algorithms and reduce their processing time.

Some common issues for traffic monitoring and analysis systems are already taken care of by the NG-MON's existing modules. For example, one might be questioning how we deal with fragmented packets in the analysis modules. Fragmented packets are reassembled before being aggregated into flows, so they would not be passed to the analyzer module for simplicity reasons. Another issue, which relates to NG-MON's design, is how to deal with the time dependency between flows. We have taken care of this issue in the actual implementation by handling the flows with the granularity of a minute. When identifying the current minute's flows, we search for the identical flows and look up their application names in the previous minute data. According to a few additional conditions we define, our algorithm will decide whether the previous minute's decision on application names is still valid in the current minute. We refer to this procedure as a history lookup mechanism.

V. Experience and Results

In this section, we describe the deployment of the application-level traffic analysis system in a POSTECH campus network. We also present the analysis results.

1. System Deployment

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 networks. There are two core-switches and two Internet routers connected with four 1-Gbps Ethernet links in a mash structure, as shown in Fig. 10. We used four optical taps to capture all in/out Internet traffic from the four 1-Gbps Ethernet links between core-switches and Internet routers.

We can see the results of the application-level traffic analysis in the protocol view page of the NG-MON presenter. Figure 11 shows an example of the application-level traffic analysis results.

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, though it is configurable. 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. Figure 11(a) is a front page of the application-layer traffic analysis, where we can see the proportion of determined traffic at specified minutes in terms of bytes, packets, and flows. We can see the top 10 list of application traffic and the top 10 list of undetermined flow groups. The undetermined flow group information helps us to investigate unknown applications off-line. Figure 11(b) shows the sorted list of determined application traffic with the proportion of them in flows, packets, and bytes. Figure 11(c) shows the flow-level details of specific application traffic. The

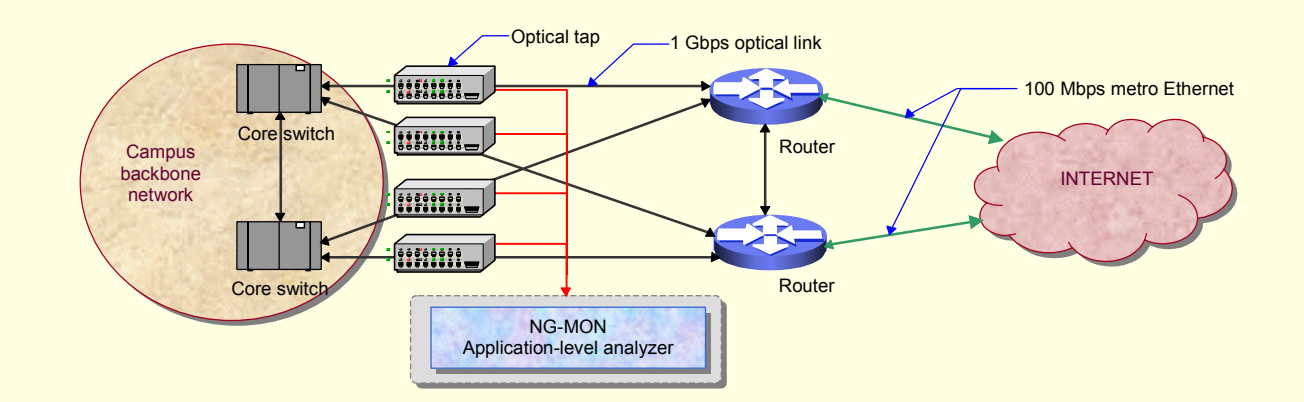
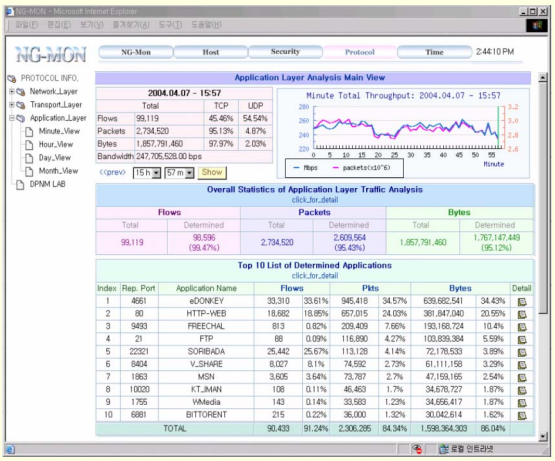
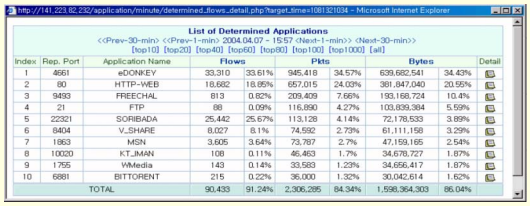


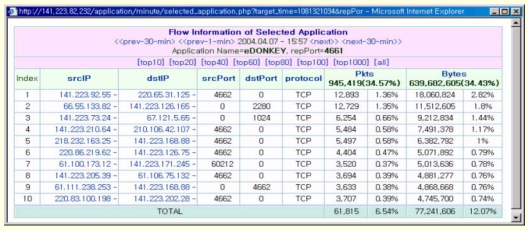
Fig. 10. Internet connection structure of POSTECH.



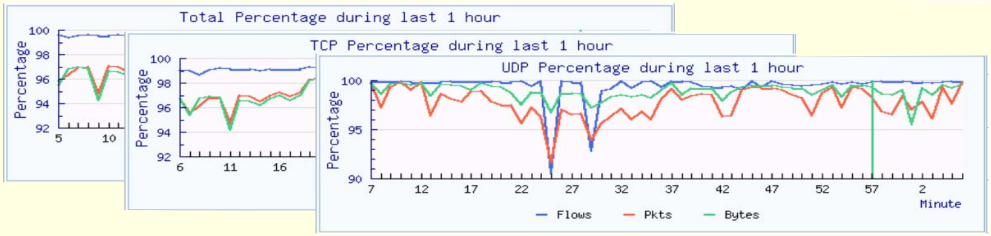
(a) Application protocol view of NG-MON presenter



(b) Top 10 list of application layer traffic



(c) Detail in flow level of eDonkey traffic



(d) The analyzed traffic proportion during last 1 hour in time-series graph

Fig. 11. Application protocol view page.

Table 5. Traffic trace summary.

Collection period		Feb. 1, 2004 to Feb. 7, 2004								
Collection location		Internet junction of POSTECH campus network								
		Flows			Packets			Bytes		
		(× 10^6)	Ratio (%) (In:Out)		(× 10^6)	Ratio (%) (In:Out)		GB	Ratio (%) (In:Out)	
Total	TCP	295	34%	(47:52)	18,345	93%	(49:50)	13,697	98%	(40:59)
	UPD	537	62%	(51:48)	1,089	5%	(52:47)	190	1%	(69:30)
	ICMP	33	3%	(46:53)	190	0%	(78:21)	16	0%	(76:23)
	Others	0.1	0%	(28:71)	1	0%	(37:62)	0.6	0%	(85:14)
	Total	866	100%	(49:50)	19,636	100%	(50:49)	13,905	100%	(41:58)

graphs in Fig. 11(d) are three time-series graphs showing the proportion of determined traffic in flows, packets, and bytes during a specified one-hour period.

2. Application Layer Traffic Characteristics

We collected Internet traffic at the POSTECH campus network for one week in Feb. 2004. The overall statistics of the traffic trace are described in Table 5.

Among the captured packets, we excluded the non-IP packets and IP packets with spoofed IP addresses. We considered the spoofed packet as packets whose source and destination IP address does not belong to our campus network address range. The portion of the excluded data was 0.5% of the total traffic in bytes. As Table 5 illustrates, the outbound traffic is 1.4 times larger than the inbound traffic in bytes. The

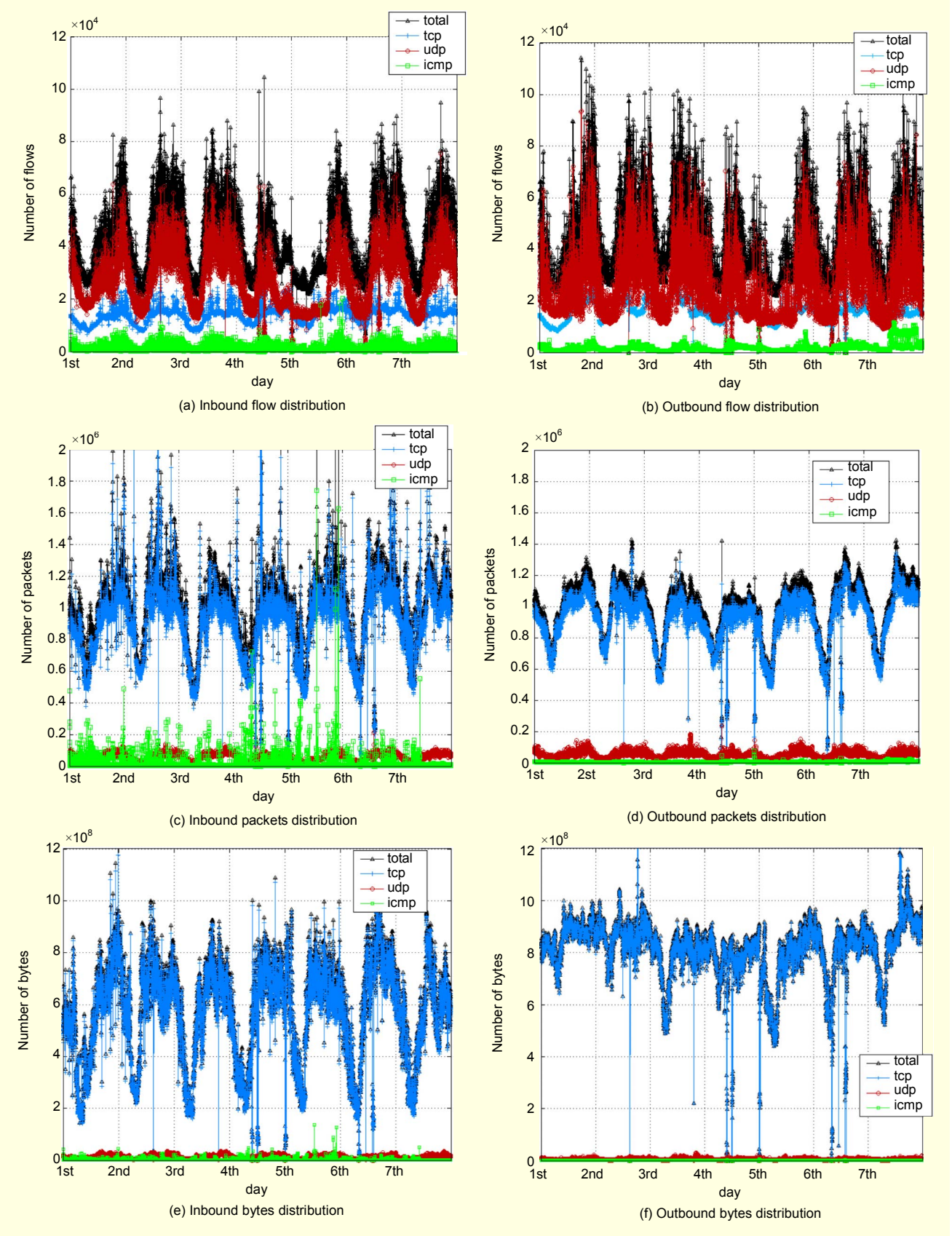


Fig. 12. Time-series graph in three analysis metrics (flow, packet, and byte) for our traffic trace (measured from Feb.1 to Feb 7, 2004).

inbound traffic refers to the traffic transferred from the Internet to our campus network, and the outbound traffic is vice versa. In addition, the number of UDP flows is much greater than the number of TCP flows, while the TCP traffic overwhelms the UDP traffic in the packet count and byte size. This fact infers that the small size UDP packets are very widely used by network-based applications. This makes the flow-based traffic analysis systems overloaded. Moreover, the inbound packet count is 10% larger than the outbound packet count, while the total inbound byte count is about 15% smaller than the total outbound byte count. This means that the average packet size of inbound traffic is much smaller than the outbound traffic. We believe this is mainly due to P2P traffic and popular FTP servers operated by students. The total number of internal and external IP addresses appeared in the captured flows was about 3,600,000.

Figure 12 illustrates six time-series graphs of the traffic trace. Each graph shows a 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 Figs. 12(a) and 12(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.

The shapes of packet distribution and byte distribution graphs are primarily affected by the amount of TCP traffic, which contradicts the shape of the flow distribution. The time-of-day effect appears in all three kinds of graphs. The traffic increases from the afternoon and peaks between 10 p.m. and 1 a.m. of the next day, and then goes down in the morning, which is typical of our university Internet usage behavior since all of our students live in the campus dormitories. The incoming ICMP packets are much larger than the outgoing ICMP packets, as illustrated in Figs. 12(c) and 12(d). 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 is more than that of the inside. The fluctuation of incoming bytes is higher than the fluctuation of outgoing bytes. This is because the number of outside users is much higher than inside users. In other words, the more users access a network, the less the fluctuation of download traffic appears.

For application-level traffic identification, we constructed the APT information by investigating about 800 Internet-based applications. The determined proportion of the traffic traced by our proposed method is 99.5% of total flows, 94% of total packets, and 92% of total bytes. Most determined traffic were generated from less than 100 applications among the applications listed in the APT. Table 6 shows the ten heaviest

Table 6. Top 10 most popular applications in flows, packets, and bytes.

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

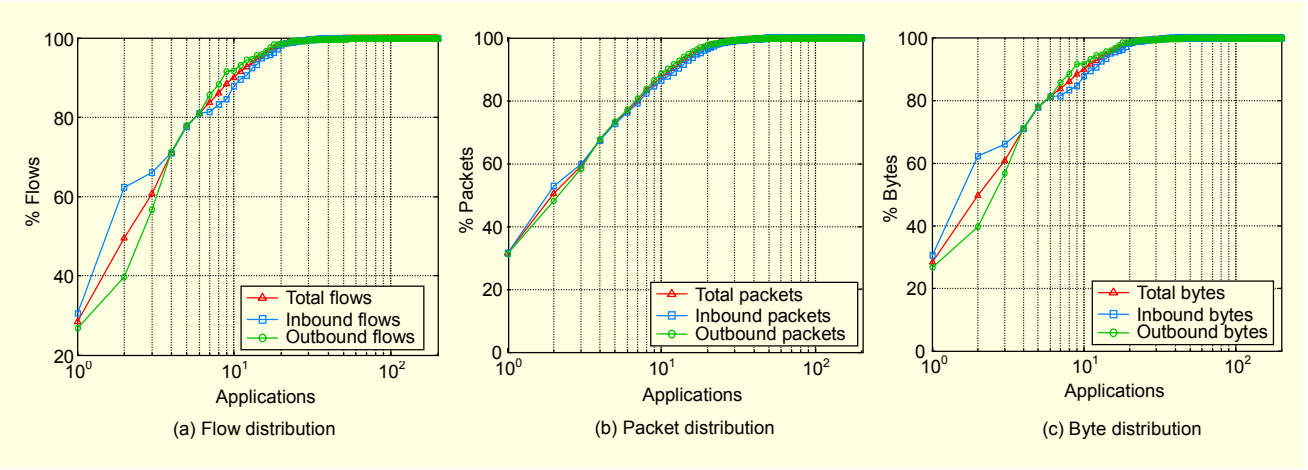


Fig. 13. Cumulative probability distribution of flows, packets, and bytes of the top 150 traffic generating applications.

applications in three perspectives of traffic metrics: the number of flows, the number of packets, and the total byte size. As Table 6 shows, the flow distribution does not follow the packet and byte distribution, while the packet and byte distribution is almost in accordance with the other.

The top 10 most popular applications occupy 95.6% of the total flows, 76% of the total packets, and 76.7% of the total bytes. This indicates that the flow distribution is more skewed than the other two distributions. Six of the applications in the flow distribution and seven of the applications in the 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 [10]-[14]. Web traffic is still one of the most traffic-consuming applications, while the FTP application is less than the web application. World-wide P2P applications such as eDonkey and KaZaA occupy a large part of Internet traffic. In addition, nation-wide P2P applications such as V_SHARE, FREECHAL, SAYCLUB, and SORIBADA are located in the top 10 list of the three different distributions and occupy a large part of Internet traffic.

Figure 13 shows a cumulative probability distribution of flows, packets, and bytes from the determined traffic in our traffic trace of the top 150 traffic-generating applications. Figure 13 indicates that the flow distribution is more skewed than the other two distributions. Several popular applications generate most of the IP traffic; the top six applications occupy about 80% of the total traffic. In addition, the outbound traffic is more skewed than the inbound traffic in flow distribution and byte distribution.

3. Performance and Limitations

The performance of our FRM algorithm solely depends on the flow counts rather than the physical link speed or real bandwidth utilization. Although there is a greater chance to have an increase in the number of flows in the multi-gigabit links, we believe that the number of flows generated from those links is tolerable by the current version of our implementation. From the recent deployment experiences at various sites (e.g. POSTECH, Korea Internet eXchange), we were able to verify the scalability of the system and provide precise figures for a system performance measurement.

The current version of our system can handle about 100,000 flows within or a little over 5 seconds, and 1,000,000 flows in around 30 seconds. The average number of flows generated from the POSTECH Internet junctions during 1 minute was 100,000 where the link utilization was 250 to 300 Mbps. Using the FRM algorithm, we could classify the 100,000 flows into about 500 LDG groups and decide the corresponding application name of 99% of the flows, 95% of the packets, and 92% of the bytes. In addition to that, we had access to the government-owned facility called Korea Internet eXchange (KIX), which was the country's major PoP and consisted of gigabit backbone links with various utilization ratios. Our deployment took place in four different one-gigabit links, simultaneously, with the maximum of 2.4 Gbps and 1,200,000 flows per minute in total. For the record, these figures are obtained from both real-world deployments and testing environments using SmartBits 600, a packet generating hardware tool.

We define an application (in the context of application-level traffic analysis) as a collection of flows that are directly or indirectly originated from the same application-layer protocol, while real-world applications imply actual programs that make use of the application-layer protocols. For example, there are hundreds of FTP applications in the Internet, such as CuteFtp, WsFtp, and so on. Obviously, they all share one thing in

common: they use the FTP protocol. In our system, the traffic generated from these applications is shown as FTP traffic as a whole, not CuteFtp or WsFtp traffic. In this case, we actually grouped flows according to the type of service they provide (file transfer) as well as the protocol in use (FTP) for this service. Here is a somewhat confusing scenario of using a session initiation protocol (SIP). SIP signaling traffic should be grouped as SIP-based traffic regardless of the origin real-world applications. What is different from the previous example is that the type of service is not one but many (e.g., VoIP, video conferencing); but the application protocol in use is the same at the initial phase for all these services. The objective of our algorithm is to reduce the unknown traffic as much as possible; however, this does not necessarily imply the complete investigation over all existing vendor applications. Therefore, in this case we simply narrow the problem down to SIP-based traffic and clear some of the ambiguity in traffic. For a more detailed analysis on specific services, we provide a basis (the sample data contains the particular service traffic) for an offline analysis.

Throughout the recent experience of system deployments into various networks, we strongly believe that our methods are generally useful for the purposes of an in-depth monitoring of IP networks, application usage, and user behaviors. However, there are a few limitations. Even though we analyze the application-level information of the networks, it is not entirely correct to call ours a layer 7 analysis system because we only work with the packet header information. The good news is that the payload examination method becomes limited because of the privacy issue and the encryption of contents. The second constraint is that the APT is not universal. The slight modification to the APT might be necessary at the deployment site. For example, popular applications in Europe might not appear among the user groups in Korea. We are considering these limitations in our future research.

VI. Conclusion

The large number of applications and their use of dynamic sessions cause one of the main problems with traffic analysis in the recent high-speed and high-volume Internet. The traffic types and patterns of the recent Internet are complex and sophisticated compared to traditional traffic, such as HTTP, FTP, and TELNET. The traditional well-known port number-based traffic analysis mechanism is not suitable to analyze newly emerging Internet traffic, such as P2P, streaming media, and game traffic. Therefore, we must develop a new method able to handle the various trends of current traffic.

In this paper, we have presented a method to identify 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 the traffic analysis point of view. We categorized most current network-based applications into five classes according to their traffic pattern. Using this categorization, we developed a flow grouping method, which determines the application name of individual packets. The proposed method consists of three components: an APT, an IPS, and a FRM. To validate our method, we designed and implemented an application-layer traffic analysis system as an essential part of the NG-MON system. We have identified more than 90% of Internet traffic from POSTECH using our proposed method. The analysis results show that more than 50% of recent Internet traffic is caused by newly emerging applications, especially P2P applications. Previous studies presented similar statistical results. In addition, most Internet traffic is generated by about ten of the most popular applications.

In this paper, we also characterized Internet traffic using a short time period of traffic trace, 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 one week in each month and making various traffic traces by collecting traffic in a number of ISP and enterprise networks. The exact traffic identification in an 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 traces to improve our proposed method, especially the FRM algorithm. Further, the detailed and accurate traffic analysis can provide useful information towards the control of current Internet traffic. We are also considering monitoring and analysis needs for the IPv6 environment and the QoS traffic provisioning based on the analysis results 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*, Montreal, Canada, Oct. 2002, pp. 4-27.

[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 Symp. (APNOMS 2002)*, Jeju, Korea, Sept. 25-27, 2002, pp. 282-293.

[3] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," *RFC3031*, *IETF*, Jan. 2001.

[4] 4 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, Apr. 2003.

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

[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 Int'l Workshop on Distributed Systems: Operations and Management (DSOM 2003)*, Heidelberg, Germany, Oct. 2003, pp. 55-67.

[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 Int'l Workshop on Distributed Systems: Operations and Management (DSOM 2003)*, Heidelberg, Germany, Oct. 2003, pp. 93-105.

[8] 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 Symp. (APNOMS 2002)*, Jeju, Korea, Sept. 25-27, 2002, pp. 503-504.

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

[10] Subhabrata Sen and Jia Wang, "Analyzing Peer-to-Peer Traffic across Large Networks," *Proc. of the second ACM SIGCOMM Workshop on Internet Measurement Workshop*, Nov. 2002.

[11] 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.

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

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

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

[15] IANA, http://www.iana.org/assignments/port-numbers.

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

[17] Jacobus van der Merwe, Ramon Caceres, Yang-hua Chu, and Cormac Sreenan "mmdump- A Tool for Monitoring Internet Multimedia Traffic," *ACM Computer Communication Review*, vol. 30, no. 4, Oct. 2000.

[18] 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 Symp. (APNOMS 2003)*, Fukuoka, Japan, Oct. 1-3, 2003, pp.404-415.

[19] Argus, http://www.qosient.com/argus.

[20] 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)*, Mar. 25-27, 2002.

[21] 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.

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

[23] 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.

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

[25] 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.

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

[27] S. Saroiu, P. Gummadi, and S.D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Proc. of Int'l Conf. on Distributed Computing Systems*, 2002.

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

[29] J. Chu, K. Labonte, and B. Levine, "Availability and Locality Measurements of Peer-to-Peer File Systems," *Proc. of ITCom: Scalability and Traffic Control in IP Networks*, July 2002.

[30] E.P. Markatos, "Tracing a Large-Scale Peer-to-Peer System: an Hour in the Life of Gnutella," *2nd IEEE/ACM Int'l Symp. on Cluster Computing and the Grid*, 2002.

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

Myung-Sup Kim received the BS, MS and PhD degrees in computer science and engineering from Pohang University of Science and Technology (POSTECH) in 1998, 2000 and 2004. Currently, he is a Post-Doctoral Fellow in the Dept. of Electrical and Computer Engineering, University of Toronto, Canada. His research interests include Internet traffic monitoring and analysis, application-level traffic analysis, and network-security attack detection and prevention. He is a member of IEEE and KNOM.

Young J. Won received his BMath degree in computer science from the University of Waterloo, Canada in 2003. Currently, he is a graduate student in the Dept. of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Korea. His research interests include Internet traffic monitoring and analysis, application-level traffic analysis, and network-security attack detection and prevention.

James Won-Ki Hong is an Associate Professor in the Dept. of Computer Science and Engineering, POSTECH, Pohang, Korea. He has been with POSTECH since May 1995. Prior to joining POSTECH, he was a Research Professor in the Dept. of Computer Science, University of Western Ontario, London, Canada. Dr. Hong received the BSc and MSc degrees from the University of Western Ontario in 1983 and 1985 and the PhD degree from the University of Waterloo, Canada in 1991. He has been very active as a participant, program committee member, and organizing committee member for IEEE CNOM sponsored symposiums such as NOMS, IM, DSOM and APNOMS. For the last few years, he has been working on various research projects on network and systems management, which utilize Web, Java, CORBA and XML technologies. He is IEEE Comsoc Director of On-Line Content, CNOM Vice-Chair and KICS KNOM Chair. His research interests include network and systems management, traffic monitoring and analysis, and security management. He is a member of IEEE, KICS, KNOM and KISS.