Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/092L27

Behavior Signature for Fine-grained Traffic Identification

Sung-Ho Yoon, Jun-Sang Park and Myung-Sup Kim[*]

Dept. of Computer and Information Science, Korea University, Korea

Received: 6 Jun. 2014, Revised: 5 Aug. 2014, Accepted: 7 Aug. 2014
Published online: 1 Apr. 2015

Abstract: With the rapid development of the Internet and a vigorous emergence of new applications, traffic identification has become a key issue for efficient network management. Although various methods have been proposed, there are still several limitations to achieving fine-grained and application-level traffic identification. In this paper, we propose a new signature model called a behavior signature for Internet traffic identification that utilizes the inter-flow relation of application traffic. The proposed behavior signature is a unique traffic behavior pattern appearing in the first few packets of plural traffic flows when a specific function is conducted by an application with a combination of various optional traffic features. This is in contrast to other existing signature models that usually focus on a singular packet or flow for feature extraction and traffic identification. We proved the feasibility and applicability of the proposed behavior signature by developing an extraction and identification algorithm and by conducting experiments on several popular applications.

Keywords: behavior signature, traffic identification, traffic classification, network management

1 Introduction

The aims of network management are fully utilizing network resources and securely protecting network equipment from malicious attacks. To accomplish such goals, a network operator should set up its network policy properly in a timely manner. To make an efficient network policy, traffic identification should be preemptively achieved because the policy that blocks or adjusts the target traffic is conducted based on the identification results [1,2]. Traffic identification can be defined as the act of ascertaining which application is contributing to a given network traffic using distinguishable and unique characteristics, and then naming each of network traffic according to the corresponding application or service. The results of such identification are also utilized in the area of capacity planning, network provisioning, traffic engineering, and fault diagnosis.

Although several previous identification methods, most of which focus on completeness and accuracy, have been proposed, certain limitations still remain. A port-based method is used to check the port number of each packet and to identify the application according to the Internet Assigned Number Authority (IANA), which maintains a list of well-known and registered ports [3,4], but is less accurate due to violations of port assignment [5,6,7]. To overcome these limitations, a payload-based identification method performs a deep packet inspection to find the unique substring that represents the target application [8,9,10]. However, this method has limited usage due to payload encryption, privacy issues, and tunneling transfers [11,12]. A statistic-based method has recently emerged to solve the above problems [13,14,15,16,17,18]. This method applies a machine learning algorithm using flow features to distinguish statistical characteristics generated by the application protocol. This technique is suitable for course-grained protocol-level identification, but it is difficult to apply to fine-grained application-level identification because applications using the same protocol have similar characteristics.

A new signature model is required continuously in order to overcome the limitations of existing identification methods. A signature is the unique traffic pattern of an application that distinguished it from other applications. There are several considerations for developing a suitable signature model. First, a signature is made available through the combination and choice of optional existing features. Because each feature, such as the IP address, port number, and substring, has a reliable performance in a complementary relationship, the new signature should combine various traffic features to achieve maximum traffic identification performance. In addition, the new signature model should provide an

[*] Corresponding author e-mail: tmskim@korea.ac.kr

option to choose various features based on the application type and network situation. For example, a server-client model application has a unique pattern in terms of the IP address and port number, but a peer-to-peer (P2P) model application does not. In addition, certain networks might be unable to capture a payload of packets because of privacy issues. Thus, a combination and choice of optional features are required in a new signature model. Second, the signature should be based on the traffic unit that reflects the application behavior. Previously, most methods for traffic identification used a packet or flow as the traffic unit. A packet is the fundamental unit of the Internet, and a flow is a set of packets belonging to a session. Increases in capacity and multimedia applications have created complex traffic behaviors. The traffic of a target network appears to consist of independent packets. However, these packets are the result of the request/response process conducted by a specific function of the application. To complete this process reliably and accurately, the signature should reflect the behavior of the application traffic.

In this paper, we propose a new signature model called a behavior signature, and its extraction algorithm to overcome the limitations of earlier methods in accordance with the considerations indicated in the previous paragraphs. Most Internet applications generate multiple traffic flows when a user performs a specific function such as a login, chat, file transfer, and so on. For example, several flows that are involved in authorization, update, and encryption are generated in the login phase of most applications. Moreover, there is a unique pattern in the sequence and interval of these flows. Thus, we devise the behavior signature centered on the idea that the unique patterns of several flows can represent the specific function of a given application. The behavior signature uses an inter-flow unit to extract a signature and identify traffic, which is contradictory to the packet or flow units used in the previous signature model. An inter-flow unit is a set of the first request packets of more than one flow. This new traffic unit has certain advantages. Using this unit, it is possible to identify several flows at once and to combine various optionally chosen traffic features by reflecting the situation of the target network. This allows us to extract signatures easily because the new traffic unit reflects the behavior of an application, and multiple features can expand the extraction range compared to a single feature.

The remainder of this paper is organized as follows. We survey several existing traffic identification methods in Section 2. In Section 3, we describe the concept of the proposed behavior signature in detail. In Section 4, we propose an automatic extraction algorithm for a behavior signature. In Section 5, we discuss the experimental results proving the feasibility of the behavior signature. Lastly, we provide some concluding remarks and areas for future work in Section 6.

2 Related work

Due to the dramatic growth in Internet traffic and the fast development of various types of applications, several studies on traffic identification have been proposed. In this section, we categorize methods based on the traffic unit used for feature extraction and traffic identification, and describe pros and cons of each method.

2.1 Packet-level identification

The most primitive method for identifying traffic is packet-level identification. A packet is the minimum unit of network traffic. Thus, packet-level identification is conducted by checking the existence of a rule defined as an invariant pattern of the target application for each packet traveling across the network. Owing to its simple implementation, this method is the most widely used method in the industry. In addition, it is possible to be conducted and reflected on the traffic control in real-time because each packet is inspected. Typical examples of this method are port-based and payload-based methods.

A port-based method checks the port number of each packet and identifies the application according to the IANA list of well-known and registered ports [3,4]. In the early days of the Internet, this method performed well; currently, however, its accuracy has become less reliable because of violations to port assignments. An example of a port violation is using an arbitrary or dynamic port used by P2P application, or a well-known port used by other applications.

A payload-based method performs a deep packet inspection to find the substring representing the target application from each packet [8,9,10]. The result of this method is robust in terms of completeness and accuracy, and is therefore widely used in the industry at present. However, various limitations restrict the usage of a payload-based method. Typical limitations are payload encryption, a dataset without a payload for dealing with privacy issues, computational overhead, and a lack of publicly available documentations for a proprietary protocol. In addition, the requirement, that the extracted payload substring be only observable in the target application, depends on the professional background of the researchers and a repetitive time-consuming verification process. Although several automatic extraction methods have been proposed [19,20,21], their feasibility has yet to be assured. While this type of method can be categorized as a flow-level based on the use of sequential payloads in a flow, such methods are generally implemented using a packet unit.

2.2 Flow-level identification

A flow is a set of packets that contain the same 5-tuple packet header information, such as the source and destination address IP, source and destination port number, and transport layer protocol, which contains all of the forward and backward packets in a session established between two hosts. Flow-level identification uses statistical flow information, such as the number of packets and bytes in a flow, the flow bitrate, flow size, flow duration, and inter-arrival time between packets. Flow-based methods can be categorized into two types: learning-based and distribution-based methods.

A learning-based method uses an existing machine-learning technique, which can again be divided into supervised learning [13,14] and unsupervised learning [15,16]. In supervised learning, a statistical model based on a machine learning algorithm is constructed using ground-truth traffic (training data) labeled based on the nature of the traffic, and the target traffic (test data) is classified. This method guarantees highly accurate results in trained applications. However, the results of this method depend on the quality of the ground-truth traffic, and unknown applications excluded from the training data cannot be classified. Unsupervised learning classifies traffic through clustering techniques using similar statistical information from traffic generated by the same application protocol. It does not require a training phase and can classify unknown applications. However, the results of this method are affected by a tunable parameter, and an additional process is required to label the application name of each cluster.

A distribution-based method identifies traffic using the first K packet size and the direction distribution of a given flow [17,18]. Although this method solves the problem of privacy invasion from using the packet size and direction without inspecting the payload for real-time identification, it has limited application because it has difficulties identifying the traffic between applications implemented by the same protocol due to their similar statistical characteristics.

2.3 Host-level identification

Host-level identification method identifies a host based on a profile characterization of the traffic behavior of the host. This method aims to find the hosts of specific applications for traffic control, to identify malicious or victim hosts for security purposes, and to understand the trend of network usage for management. Example profiles for a host behavior are the number of peers and the number of sources and destination ports. Typical examples of this method are connection pattern-based methods [22,23].

A connection-pattern-based method checks the traffic behavior of the target host with a pre-defined connection pattern. This method is very simple and easy to use in diverse networks and it can identify traffic without using port numbers or payload data. One limitation, however, is that this method has difficulty distinguishing applications generated in a single host. This is caused by the assumption that the traffic on a single host is generated by a particular application.

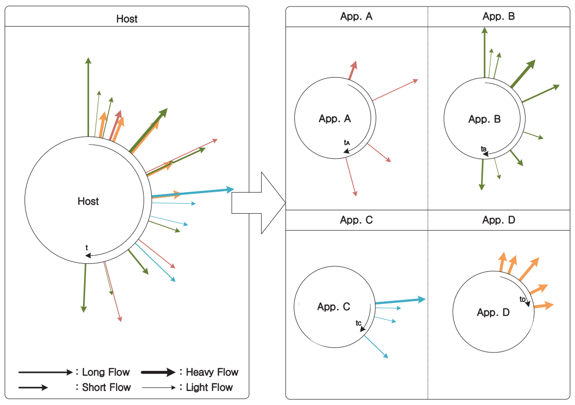

Fig. 1: Concurrent Internet application traffic on a single host

Although certain related works have shown a reliable performance, some limitations still remain. These limitations are caused by a difficulty in the flexible feature choice and usage of a microscopic traffic unit. To overcome these limitations, we propose the behavior signature providing various features, even in different combinations and optional choice, and a new traffic unit reflecting the application behavior.

3 Behavior signature

Advanced multitasking technology, i.e., sharing of common processing resources, has become possible on various applications through the rapid growth of the Internet and computer technologies. For example, we can enjoy music broadcasts over the Internet while Web shopping or play games with a voice-over-IP service. In this way, using multiple applications, Internet users may intentionally or unintentionally generate various types of concurrent application traffic simultaneously.

Figure 1 shows concurrent Internet application (A-D) traffic on a single host during a particular time, t. The circle indicates the host and application, both of which are the traffic source, and the arrows indicate the traffic (flow). To represent the traffic properties, the length and thickness of an arrow signify the duration (long and short flows) and statistical information such as the packet count and byte amount (heavy and light flows), respectively. Figure 1 shows that single-host traffic is generated by concurrent applications. Because several applications generate traffic simultaneously, traffic identification by means of a flow or packet unit causes a significant amount of overhead in the identification system, and cannot guarantee a high accuracy of the identification result. Most applications generate multiple flows when conducting a particular function, and these flows have clear patterns, such as their sequence and interval, from the perspective of application behavior. We can therefore

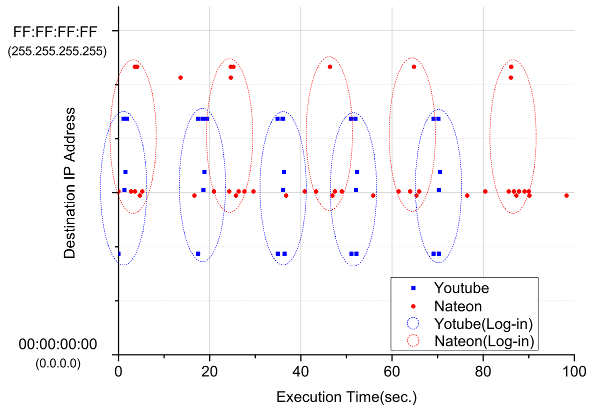

Fig. 2: Unique pattern of the log-in phase in terms of destination IP address

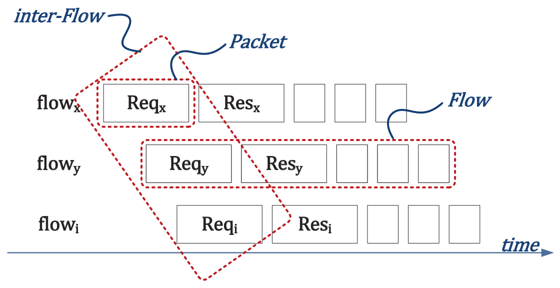

Fig. 3: Various traffic units for traffic identification

identify these concurrent flows as the application-level at once using their significant patterns. The proposed behavior signature based identification method extracts a signature combining traffic features and a unique pattern from the target application, and identifies their concurrent traffic (flow) simultaneously.

To grasp the feasibility of a behavior signature, we observe the unique pattern of the log-in phase in terms of the destination IP address when using Nateon (an instant messaging client) and Youtube (a video-sharing Website). As shown in Figure 2, the x-axis indicates the execution time of the test, and the y-axis shows the host byte order address converted from the destination IP address described as a TCP/IP network order. The value ranges from 0x00000000 (0.0.0.0) to 0xFFFFFFFF (255.255.255.255). In this test, we conduct a log-in phase five times repeatedly, as marked with the dots on the graph according to the destination IP address of the first request packet. Figure 2 shows the unique traffic pattern generated by a specific server farm during every log-in phase. Similarly, we observe that other traffic features such as the destination port number, L4 protocol, and payload also have unique patterns.

Figure 3 shows the various traffic units for traffic identification, i.e., packet, flow, and inter-flow units. The packet unit uses the packet header information (IP address, port number, and L4 protocol) and payload in a single packet. Using the identification results, the packet unit is a suitable means for real-time control due to its ability to control unwanted traffic immediately after the packet inspection. However, it is difficult to extract signature because the range of extraction is relatively small. In addition, a significant overhead is created by inspecting all packets for identifying traffic. The flow unit, on the other hand, uses not only the packet attributes but also additional information, such as the inter-arrival time, packet size distribution, and total byte size. The flow unit is suitable for extracting signatures because of its relatively large range of extraction. Moreover, it can be applied to encrypted traffic using only statistical features. In other words, more attributes can be utilized for signature extraction in the flow unit compared to the packet unit. However, limitations exist, such as low accuracy and real-time control. The flow unit can be used to identify traffic after a flow has ended.

The behavior signature is applied using an inter-flow unit, as shown in Figure 3, and seeks to minimize the limitations of both the packet and flow units, and to maximize their advantages. An inter-flow is a set of first request packets of several flows. Therefore, it uses not only packet unit attributes but also unique patterns such as sequence and interval information. It is easy to create a signature because the range of extraction is large owing to the use of plural flows. In other words, a signature is extracted using various traffic features. In addition, an inter-flow has the ability to control traffic in real-time because the traffic is identified by the signature in the first request packet located at the beginning of the flow.

The traffic features used in the behavior signature are the destination IP address, destination port, L4 protocol, and fixed offset N-bytes string in the first payload packet of a flow. The signature consists of a combination (called an entry) of these features. Because of this characteristic, it is convenient to extract a signature in this way as compared to using a single feature. Header information such as the IP address, port, and protocol has significant meaning in the server-client model and using fixed port traffic. Payload information has been used as a salient key for identifying traffic; however, because of the computational overhead, its usage has declined in traditional methods. The payload signature, usually described as a complex regular expression, causes a large amount of overhead owing to its characteristic in that the occurrence of a match can be located anywhere in the packet payload. To solve these problems, rather than using a random offset payload, we use a fixed offset N-bytes string only. It is easy to resolve the computational complexity problem because the signature uses a very small and simple bit string with a fixed offset and length at the beginning packet of the flow.

Figure 4 shows the proposed behavior signature model. The circle indicates the particular application; the arrows represent entries extracted from the first request packet of the flow generated by the application during a specific interval. As shown in Figure 4, four entries are

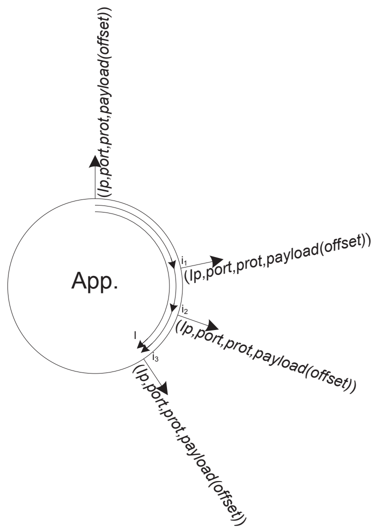
App.
(Ip,port,prot,payload(offset))
(Ip,port,prot,payload(offset))
(Ip,port,prot,payload(offset))
(Ip,port,prot,payload(offset))
i1
i2
i3
I

Fig. 4: Behavior signature model

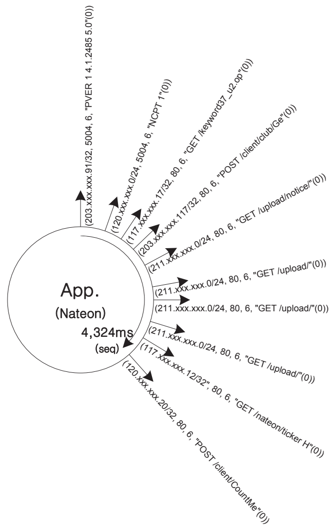
App.
(Nateon)
4,324ms
(seq)
(203.xxx.xxx.91/32, 5004, 6, "PVER 1 4.1.2485 5.0"(0))
(120.xxx.xxx.0/24, 5004, 6, "NCPT 1"(0))
(117.xxx.xxx.17/32, 80, 6, "GET /keyword37_u2.op"(0))
(203.xxx.xxx.117/32, 80, 6, "POST /client/club/Ge"(0))
(211.xxx.xxx.0/24, 80, 6, "GET /upload/notice/"(0))
(211.xxx.xxx.0/24, 80, 6, "GET /upload/"(0))
(211.xxx.xxx.0/24, 80, 6, "GET /upload/"(0))
(211.xxx.xxx.0/24, 80, 6, "GET /upload/"(0))
(117.xxx.xxx.12/32", 80, 6, "GET /nateon/ticker H"(0))
(120.xxx.xxx.20/32, 80, 6, "POST /client/CountMe"(0))

Fig. 5: Example of a behavior signature

generated by an application within time interval I, which consists of sub-interval i. This unique pattern can represent the behavior of application traffic. We therefore generate entries and combine them with the sequence type and time interval. Finally, we create a behavior signature.

A behavior signature consists of several entries having the aforementioned traffic attributes. The following equations can be used to define a behavior signature.

$$BS = \{A, T, I, i_1, i_2, ..., i_{n-1}, E_1, E_2, ..., E_n | n \geq 2, \\ Src(E_1) = Src(E_2) = ... = Src(E_n)\} \qquad (1)$$

$$E = \{2^F | F = \{ip, port, prot, payload(offset)\}, \\ E \neq \varnothing\} \qquad (2)$$

A behavior signature (BS) consists of the application name (A), type (T), interval (I), sub-intervals (i_{n-1}), and entries (E_n), where $n \geq 2$. Entry (E) is a power set of the destination IP address (ip), destination port ($port$), L4 layer protocol ($prot$), and N-bytes payload with an offset ($payload$), where a null set is excluded. In other words, we selectively use the features of entry in the traffic identification if the selected feature is meaningful. For example, we exclude the destination IP address and destination port from the entry when the application uses random ports under a P2P connection. In this case, we write *any* as ip or $port$ attribute. The source host ($Src(E_x)$) of all entries is the same because the behavior signature represents the behavior of single application traffic from a single host.

Table 1 describes the attributes of a behavior signature. The application name (A) is used for naming the identified traffic. The entry applying method type (T) is either a sequence (*Seq*) or set (*Set*), where *Seq* indicates that the identification is conducted in serial order, whereas *Set* means that the identification is conducted randomly within a particular time interval. Interval (I) is the period of time in milliseconds from the first entry to the last entry, i.e., the time during which all entries of the behavior signature are applied. The sub-Interval (i) is the period of time in milliseconds between each entry. Entry (E) consists of the destination IP address (ip), destination port number ($port$), L4 layer protocol ($prot$), and N-bytes payload with an offset ($payload$). The destination IP and port are the destination of traffic that will be identified by the entry. The IP address is represented in the Classless Inter-Domain Routing (CIDR) notation. The L4 layer protocol is either a Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) when transmitting traffic on the Internet. We use a fixed offset N-bytes payload rather than a random offset payload.

Figure 5 shows an example of a behavior signature. Nateon is a popular instant messaging client in South Korea. The example signature is extracted when a Nateon user log in to the client. Ten request packets based on HTTP are sequentially generated during the log-in phase within a 4,324ms time interval. We can therefore extract

Table 1: Behavior based signature attributes and their description

Attribute		Explanation
A		Application name
T		Entry applying method Sequence (*Seq*), Set (*Set*)
I		Interval applying all entries (*ms*)
i_n		Sub-interval between E_n and E_{n+1} (*ms*)
E	ip	Destination IP address in CIDR notation
	$port$	Destination port number
	$prot$	L4 protocol (TCP, UDP)
	$payload$	First N-bytes payload with offset
$Src(E_x)$		Source IP address of Entry x

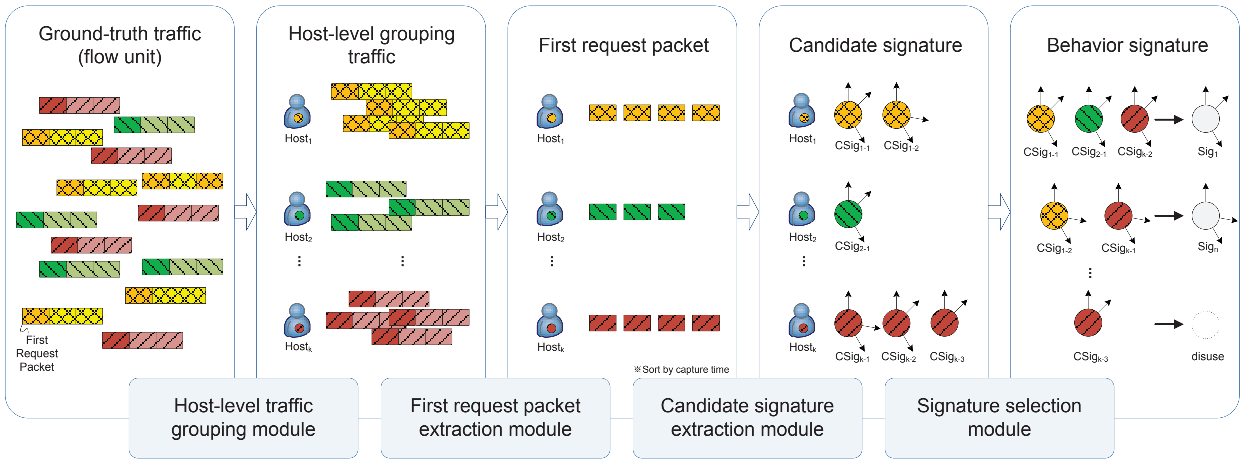
Ground-truth traffic (flow unit)
First Request Packet
Host-level grouping traffic
Host1
Host2
Hostk
First request packet
※Sort by capture time
Candidate signature
CSig1-1
CSig1-2
CSig2-1
CSigk-1
CSigk-2
CSigk-3
Behavior signature
Sig1
Sign
disuse
Host-level traffic grouping module
First request packet extraction module
Candidate signature extraction module
Signature selection module

Fig. 6: Overview of the proposed extraction algorithm

ten entries from each request packet feature, and combine them into a behavior signature based on the time interval and entry applying method.

4 Extraction algorithm

This section describes the algorithm used for extracting a behavior signature. The algorithm consists of a host-level traffic grouping module, a first request packet extraction module, a candidate signature extraction module, and a signature selection module. Figure 6 shows a detailed flow diagram of the extraction algorithm. Based on the input traffic, we group the traffic by host-level, extract the first request packet, and then extract all candidate signatures from every conceivable combination of entries. Finally, we select a behavior signature from the candidate signatures.

4.1 Host-level traffic grouping module

The host-level traffic grouping module receives ground-truth traffic as the input data, and groups the data based on the host-level. The ground-truth traffic indicates traffic labeled based on their source, such as an application or service. The source is the target of the extraction algorithm. The reason for grouping the traffic is to allow the signature to be extracted easily, and to reduce the overhead in the extraction system when grouping the traffic by host, because the behavior signature is based on a single host behavior.

There are two methods for ground-truth traffic collection. The first method is manual collection in which the target application is manually and intentionally run on several machines, and the corresponding generated traffic is labeled [24]. This method has an advantage of collecting ground-truth traffic in a simple manner, but cannot guarantee high-quality results since certain daemon applications such as operating systems and anti-virus programs unintentionally generate their own traffic. A more improved method is using a socket monitoring agent [20] on several machines and collecting the socket log, including the process name, IP address, port number, L4 protocol, and process path. The log data are composed of raw traffic from the machines; thus, we can obtain the ground-truth traffic. This method can easily classify the target application traffic from the total traffic of the machines; however, this requires the additional installation of an agent and a comparison process. In this paper, we collect the ground-truth traffic using the second method.

4.2 First request packet extraction module

The first request packet extraction module extracts the first request packet from the host-level grouped traffic. The extracted first request packet is sorted based on the time the packet occurs. As mentioned in Section 3, a behavior signature is based on the inter-flow unit. This means that the signature is extracted from the first request packet of several flows when a specific behavior is conducted on Internet application. By applying this module, the total amount of input data is reduced and time information such as the sequence and interval is provided.

4.3 Candidate signature extraction module

The candidate signature extraction module extracts the entry from the first request packet of each host and creates every conceivable combination as a candidate signature. Internet traffic is not the same even when the exact same function is conducted due to differences in application version, operating system (OS), and network congestion. To find the common patterns observed in all hosts conducting the same function, it is necessary to ensure

Table 2: Results of behavior signature extraction and examples

Application	Num. of signature	Examples
Nateon	48	{Nateon, Seq, 4324, (203.xxx.xxx.91/32, 5004, 6, ”PVER 1 4.1.2485 5.0”(0)), (120.xxx.xxx.0/24, 5004, 6, ”NCPT 1”(0)), (117.xxx.xxx.17/32, 80, 6, ”GET /keyword37_u2.op”(0)), (203.xxx.xxx.117/32, 80, 6, ”POST /client/club/Ge”(0)), (211.xxx.xxx.0/24, 80, 6, ”GET /upload/notice/”(0)), (211.xxx.xxx.0/24, 80, 6, ”GET /upload/”(0)), (211.xxx.xxx.0/24, 80, 6, ”GET /upload/”(0)), (211.xxx.xxx.0/24, 80, 6, ”GET /upload/”(0)), (117.xxx.xxx.12/32”, 80, 6, ”GET /nateon/ticker H”(0)), (120.xxx.xxx.20/32, 80, 6, ”POST /client/CountMe”(0))}
DropBox	1	{DropBox, Seq, 3258, (any, 443, 6, ”0x16 0x03 0x01 0x00 0x5B 0x01 0x00 0x00 0x57 0x03 0x01 0x50”(0)), (any, 80, 6, ”GET /subcribe?host_”(0))}
UTorrent	7	{UTorrent, Set, 5000, (any, any, 17, ”d1:ad2:id20:” (0)), (any, any, 17, ”A.”(0)), (any, any, 17, ”d1:ad2:id20:” (0))}
Skype	3	{Skype, Seq, 5000, (any, any, 6, ”GET /ui/0/5.10.” (0)), (any, any, 6, ”0x16 0x03 0x01 0x00”(0))}
Teamviewer	1	{Teamviewer, Seq, 4991, (any, 5938, 6, ”.$”(0)), (any, 5938, 17, ”0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00”(0))}
Youtube	18	{Youtube, Set, 1879, (74.xxx.xxx.0/24, 80, 6, ” GET /generate_204 HT”(0)), (74.xxx.xxx.0/24, 80, 6, ”GET /videoplayback?”(0)), (74.xxx.xxx.0/24, 443, 6, ”0x16 0x03 0x01 0x02 0x00 0x01 0x00 0x01 0x FC 0x03 0x03 0x53 0x0C”(0))}
Google	4	{Google, Seq, 4734, (74.xxx.xxx.0/24, 443, 6, ”0x16 0x03 0x01 0x02 0x00 0x01 0x00 0x01 0xFC 0x03 0x03 0x53 0x0C”(0)), (74.xxx.xxx.0/24, 80, 6, ”GET /accounts/Logout”(0))}
Facebook	24	{Facebook, Seq, 177, (96.xxx.xxx.0/24, 443, 6, any), (173.xxx.xxx.0/24, 443, 6, any), (31.xxx.xxx.0/24, 443, 6, any)}
Yahoo	11	{Yahoo, Seq, 144, (203.xxx.xxx.0/24, 443, 6, any), (23.xxx.xxx.120/32, 443, 6, any), (203.xxx.xxx.0/24, 443, 6, any), (74.xxx.xxx.0/24, 443, 6, any)}
Wikipedia	14	{Wikipedia, Seq, 460, (74.xxx.xxx.0/24, 443, 6, ”0x16 0x03 0x01 0x02 0x00 0x01 0x00 0x01 0xFC 0x03 0x03 0x53 0x0C”(0)), (198.xxx.xxx.96/32, 80, 6, ”GET /wiki/Main_Page”(0)), (198.xxx.xxx.112/32, 80, 6, ”GET /wikipedia/en/th”(0)), (74.xxx.xxx.0/24, 443, 6, ”0x16 0x03 0x01 0x02 0x00 0x01 0x00 0x01 0xFC 0x03 0x03 0x53 0x0C”(0))}

that the candidate signature is able to identify the traffic of all the hosts generating the target application traffic.

Additionally, this module conducts feature selection of an entry when extracting the entry from the first request packet. In the case of a P2P application, the destination IP address and port number attributes are excluded (represented as *any*). In the case of an encrypted application, the payload attribute is excluded. Depending on the amount of input traffic, this module has a high computational complexity because it considers every conceivable candidate signature. We therefore set thresholds, such as the maximum interval from the first entry to the last entry (`MAX_INTERVAL`), and the maximum entry size (`MAX_SIZE`); that is, candidate signatures are extracted by limiting `MAX_INTERVAL` and `MAX_SIZE`.

4.4 Signature selection module

The signature selection module chooses the behavior signature from the candidate with host counts exceeding the minimum peer count using the signature (`MIN_PEER`) from every possible candidate signature. The behavior signature is the most commonly used pattern when all hosts use a particular application.

5 Experiment and results

This section details the experiment results and the feasibility of a behavior signature based on ten popular applications.

We select the following ten popular applications and services as the target applications: Nateon, an instant messenger; DropBox, a file hosting; UTorrent, a P2P file transfer; Skype, an instant messenger; Teamviewer, a remote desktop; Youtube, a video-sharing; Google, search engine; Facebook, social network; Yahoo, a portal; Wikipedia, an online dictionary, as the target applications. To test the performance accurately, we collected the traffic data of these applications by conducting various functions on four different hosts at two particular times.

5.1 Signature extraction

Table 2 lists the results of signatures selected using the proposed algorithm. For this test, we set `MAX_INTERVAL` to 5,000ms, `MAX_SIZE` to 10, and `MIN_PEER` to 4.

Table 3: Accuracy of behavior signature

Application	Unit	Precision	Recall
Nateon	flow	1.00 (447/447)	0.60 (447/741)
	byte(K)	1.00 (5,064/5,064)	0.02 (5,064/254,110)
DropBox	flow	1.00 (193/193)	0.78 (193/247)
	byte(K)	1.00 (5,303/5,303)	0.15 (5,303/35,708)
UTorrent	flow	1.00 (2,999/2,999)	0.17 (2,999/18,106)
	byte(M)	1.00 (2,741/2,741)	0.66 (2,741/4,182)
Skype	flow	1.00 (127/127)	0.06 (127/2,088)
	byte(K)	1.00 (1,589/1,589)	0.02 (1,589/103,342)
Teamviewer	flow	1.00 (239/239)	0.63 (239/385)
	byte(K)	1.00 (8,237/8,237)	0.04 (8,237/215,845)
Youtube	flow	1.00 (59/59)	0.21 (59/278)
	byte(M)	1.00 (107/107)	0.76 (107/141)
Google	flow	1.00 (173/173)	0.12 (173/1,370)
	byte(K)	1.00 (4,223/4,223)	0.09 (4,223/42,573)
Facebook	flow	1.00 (1,866/1,866)	0.51 (1,866/3,620)
	byte(M)	1.00 (22/22)	0.57 (22/38)
Yahoo	flow	1.00 (254/254)	0.14 (254/1,797)
	byte(K)	1.00 (3,075/3,075)	0.15 (3,075/19,851)
Wikipedia	flow	1.00 (565/565)	0.20 (565/2,738)
	byte(K)	1.00 (5,193/5,193)	0.36 (5,193/14,422)
Total	flow	1.00 (6,922/6,922)	0.22 (6,922/31,370)
	byte(M)	1.00 (2,904/2,904)	0.57 (2,904/5,047)

In the case of Nateon, 48 different signatures were extracted because this service has complicated traffic patterns. In particular, Nateon communicates with the authorization, update, pop-up, and main server during the log-in phase. Because Nateon is operated under a server-client model and uses a fixed port, the signatures extracted from this application have all the attributes listed in Table 1. The example for Nateon shown in the right column of Table 2 indicates that if the ten entries are matched in serial order during an interval of 4,324ms to the ten first request packets of flows, the flows can be identified as being from Nateon. In the case of UTorrent, seven signatures were extracted. Thus, we mark the destination IP address and port number as *any* because this application operates under P2P and uses a random port. As indicated in Table 2, if two entries are matched to in the given interval in random order to the two first request packets of flows, the flows can be identified as being from UTorrent. In the case of Facebook, all traffic is encrypted during log-in phase. Thus, we mark its payload as *any*. The destination IP addresses and port number still have unique patterns to identify the traffic. This case shows the advantage of using a behavior signature in terms of its ability to choose optional features.

5.2 Performance evaluation

We measure the accuracy (precision and recall) of the proposed signature method using a mixture of traffic from the ten applications considered. The following equations are used to measure the precision and recall, respectively.

$$Precision = \frac{TP}{TP+FP} \tag{3}$$

$$Recall = \frac{TP}{TP+FN} \tag{4}$$

A true positive (TP) of application X indicates the proportion of X traffic identified correctly as X. Otherwise, a false positive (FP) of application X indicates the proportion of non-X traffic identified incorrectly as X. A false negative (FN) of application X indicates the proportion of X traffic identified incorrectly as not being X. Thus, the precision is the ratio of clearly identified traffic to the total amount of identified traffic, and the recall is the ratio of clearly identified traffic to the amount of application traffic.

Table 4: Payload signature for comparison test

Application	Num. of signature	Examples
Nateon	42	.*naeon\.nate\.nate\.com.* ^PVER.*
DropBox	3	^GET \subscribe.host_int=.* .*Dropbox,Inc.*dropbox\.com.*
UTorrent	13	*BitTorrent protocol.* .*d1:ad2:id20.*
Skype	1	.*User-Agent:.*Skype.*
Teamviewer	1	^..\x00\x17\x24\x6A.\x00.*
Youtube	16	^GET.*videoplayback.* ^GET./play204?.*
Google	10	.*accounts\.google\.com.* ^GET.*Host: www\.google\..*
Facebook	6	^GET.*Host: www.facebook.com.*
Yahoo	3	^GET.* Host:.*yahoo\.com.*
Wikipedia	3	^GET.*Host:.*wikipedia.org.* ^GET.*Host:.*wikimedia.org.*

Table 5: Comparison completeness between payload and behavior signature

App.	Unit	Completeness					
		PS	*BS*	*PS*∪*BS*	*PS*∩*BS*	PS^c∩*BS*	*PS*∩BS^c
Nateon	flow	0.73 (543/741)	0.60 (447/741)	0.73 (543/741)	0.60 (447/741)	0.00 (0/741)	0.13 (96/741)
	byte(M)	0.93 (235/254)	0.02 (5/254)	0.93 (235/254)	0.02 (5/254)	0.00 (0/254)	0.91 (230/254)
DropBox	flow	0.26 (64/247)	0.78 (193/247)	0.78 (193/247)	0.26 (64/247)	0.52 (129/247)	0.00 (0/247)
	byte(M)	0.01 (1/35)	0.15 (5/35)	0.15 (5/35)	0.01 (1/35)	0.15 (5/35)	0.00 (0/35)
UTorrent	flow(K)	0.79 (14/18)	0.17 (2/18)	0.80 (14/18)	0.15 (2/18)	0.01 (1/18)	0.63 (11/18)
	byte(G)	0.96 (4/4)	0.66 (2/4)	0.99 (4/4)	0.62 (2/4)	0.04 (1/4)	0.34 (1/4)
Skype	flow	0.02 (44/2,088)	0.06 (127/2,088)	0.06 (127/2,088)	0.02 (44/2,088)	0.04 (83/2,088)	0.00 (0/2,088)
	byte(K)	0.01 (51/103,342)	0.02 (1,589/103,342)	0.02 (1,589/103,342)	0.01 (51/103,342)	0.01 (1,538/103,342)	0.00 (0/103,342)
Team viewer	flow	0.01 (1/385)	0.62 (239/385)	0.62 (204/385)	0.00 (0/385)	0.62 (239/385)	0.01 (1/385)
	byte(K)	0.01 (1/215,845)	0.04 (8,237/215,845)	0.04 (8,239/215,845)	0.00 (0/215,845)	0.04 (8,237/215,845)	0.01 (1/215,845)
Youtube	flow	0.54 (151/278)	0.21 (59/278)	0.61 (172/278)	0.13 (38/278)	0.07 (21/278)	0.40 (113/278)
	byte(M)	0.98 (139/141)	0.76 (107/141)	0.99 (140/141)	0.76 (107/141)	0.01 (1/141)	0.23 (32/141)
Google	flow	0.21 (301/1,370)	0.12 (173/1,370)	0.32 (441/1,370)	0.02 (33/1,370)	0.10 (140/1,370)	0.19 (268/1,370)
	byte(M)	0.11 (4/42)	0.09 (4/42)	0.19 (8/42)	0.01 (1/42)	0.08 (3/42)	0.10 (4/42)
Facebook	flow	0.62 (2,255/3,620)	0.51 (1,866/3,620)	0.63 (2,300/3,620)	0.50 (1,821/3,620)	0.01 (45/3,620)	0.11 (434/3,620)
	byte(M)	0.77 (29/38)	0.57 (22/38)	0.78 (29/38)	0.56 (21/38)	0.01 (1/38)	0.20 (7/38)
Yahoo	flow	0.03 (48/1,797)	0.14 (254/1,797)	0.16 (302/1,797)	0.00 (0/1,797)	0.14 (254/1,797)	0.03 (48/1,797)
	byte(K)	0.01 (120/19,851)	0.15 (3,075/19,851)	0.16 (3,195/19,851)	0.00 (0/19,851)	0.15 (3,075/19,851)	0.01 (120/19,851)
Wikipedia	flow	0.23 (646/2,738)	0.20 (565/2,738)	0.28 (781/2,738)	0.15 (430/2,738)	0.05 (135/2,738)	0.08 (216/2,738)
	byte(M)	0.27 (3/14)	0.36 (5/14)	0.45 (6/14)	0.17 (2/14)	0.18 (2/14)	0.09 (1/14)

Table 3 shows the accuracy (precision and recall) of the behavior signature for the ten applications. All signatures identify the traffic precisely, i.e., the precision is 1.00 for all applications, and which is achieved because the signatures are extracted from several hosts. For the recall, the results depend on the application. The average recall is 0.22 in terms of flow units and 0.57 in terms of byte units. This is caused by the statistical characteristics of each application as having heavy or light flow. Thus, a behavior signature is more useful in the detection and control of an application than for traffic monitoring.

5.3 Comparison with payload Signature

We conduct a comparison test between the payload signature method and the proposed behavior signature method. We use a payload signature based on the Longest Common Subsequence (LCS) algorithm [20]. Examples of the payload signatures used in this test are listed in Table 4.

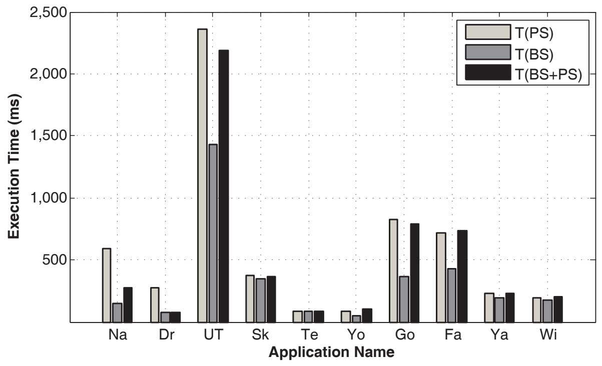

Fig. 7: Comparison of execution time between payload and behavior signature

Table 5 lists the results of the behavior and payload signatures. The following equation is the metric used to measure the performance.

$$Completeness = \frac{IdentifiedTraffic}{TotalTraffic} \quad (5)$$

PS is the ratio of identified traffic using a payload signature. BS is the ratio of identified traffic using a behavior signature. $PS \cup BS$ is the ratio of total identified traffic using either a payload or behavior signature. $PS \cap BS$ is the ratio of traffic overlap identified using a payload and behavior signature. $PS^c \cap BS$ is the ratio of traffic identified using a behavior signature, but not identified using a payload signature. $PS \cap BS^c$ indicates the reverse case.

The value of $PS^c \cap BS$ for Nateon is zero because the traffic identified using the payload method includes all traffic determined based on the behavior method, which is due to the characteristic of Nateon application using an open protocol instead of the traffic encryption. On the other hand, the behavior signatures of Dropbox, Teamviewer, and Skype include a payload signature. In the case of Dropbox, HTTPS traffic is used for data encryption. Therefore, it is difficult to extract the payload signatures using the LCS algorithm. The behavior signature method, however, can extract the signature using a combination of several entries to identify the traffic precisely.

According to this comparison test, we can find that the payload and behavior signatures have a complementary relationship in terms of traffic identification. When using an open protocol, the payload signature for Nateon and UTtorrent, shows a good performance. When using encryption and a proprietary protocol, such as in DropBox, Skype, and Teamviewer, a behavior signature is a suitable for it.

Figure 7 shows a comparison test of the execution time. $T(PS)$ is the execution time a payload signature is applied to the given test traffic. $T(BS)$ is the execution a behavior signature is applied. $T(BS+PS)$ indicates the execution time when the behavior signature is first applied, and the payload signature is then applied when any unidentified traffic exists. Although there are distinct differences in the execution times for each application caused by the difference in the amount of traffic and the number of signatures, $T(PS)$ is generally longer than $T(BS)$. $T(BS+PS)$ is longer than $T(BS)$ and shorter than $T(PS)$, while retaining the completeness of $PS \cup BS$. According to this test, we can find that a behavior signature is superior to a payload signature in terms of execution time. In addition, when we use a behavior signature as a supplementary method for a payload signature, both the execution time and completeness are improved.

6 Conclusion and future work

In this paper, we proposed the behavior signature and an automatic extraction method using the first request packets of multiple traffic flows when a single function is executed to identify big data traffic. This signature overcomes the limitation of previous methods using packet and flow units. We use ten popular applications to prove the feasibility of the proposed signature method. Although our method shows a low recall, the precision is 100% for all applications, which means that all extracted signatures correctly identified the traffic. A comparison test on the payload signature method proved that a behavior signature can be utilized as a supplementary method to identify encrypted traffic flows. The proposed method shows an improved performance in terms of execution time and completeness.

As future research, we plan to improve the extraction algorithm by applying it to various networks and applications. Moreover, we plan to develop an identification system based on the proposed signature for operation in real networks. Although the use of an inter-flow unit has many advantages, it also has certain disadvantages. An inter-flow unit operates under the assumption that plural flows occur when a single function is performed. If a single function makes a single flow, the behavior signature does not apply. We will address this limitation as future work.

Acknowledgement

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012R1A1A2007483), and Brain Korea 21 Plus (BK21+).

References

[1] Y. Wang, Y. Xiang, W. L. Zhou, and S. Z. Yu, Generating regular expression signatures for network traffic classification

in trusted network management, Journal of Network and Computer Applications **35**, 992-1000 (2012).
[2] B. Park, Y. Won, J. Chung, M. S. Kim, and J. W. K. Hong, Fine-grained traffic classification based on functional separation, International Journal of Network Management **23**, 350-381 (2013).
[3] IANA. (2014, Feb.). Service Name and Transport Protocol Port Number Registry [Online]. Available: http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml
[4] CAIDA. (1999, Mar.). CoralFeef Software Suite [Online]. Available: http://www.caida.org/tools/measurement/coralreef/
[5] T. Karagiannis, A. Broido, N. Brownlee, K. C. Claffy, and M. Faloutsos, Is p2p dying or just hiding?[p2p traffic measurement], in Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE, 1532-1538 (2004).
[6] G. Nam, P. Patankar, G. Kesidis, C. R. Das, and C. Seren, Mass purging of stale tcp flows in per-flow monitoring systems, in Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on, 1-6 (2009)
[7] S. Sen, O. Spatscheck, and D. Wang, Accurate, scalable in-network identification of p2p traffic using application signatures, in Proceedings of the 13th international conference on World Wide Web, 512-521 (2004).
[8] T. Choi, C. Kim, S. Yoon, J. Park, B. Lee, H. Kim, et al., Content-aware internet application traffic measurement and analysis, in Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP, 511-524 (2004).
[9] A. W. Moore and K. Papagiannaki, Toward the accurate identification of network applications, in Passive and Active Network Measurement, Springer, 41-54 (2005).
[10] M. Roesch, Snort: Lightweight Intrusion Detection for Networks, in LISA, 229-238 (1999).
[11] D. C. Sicker, P. Ohm, and D. Grunwald, Legal issues surrounding monitoring during network research, in Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, 141-148 (2007).
[12] K. Xu, M. Zhang, M. J. Ye, D. M. Chiu, and J. P. Wu, Identify P2P traffic by inspecting data transfer behavior, Computer Communications **33**, 1141-1150 (2010).
[13] R. X. Yuan, Z. Li, X. H. Guan, and L. Xu, An SVM-based machine learning method for accurate internet traffic classification, Information Systems Frontiers **12**, 149-156 (2010).
[14] M. Soysal and E. G. Schmidt, Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison, Performance Evaluation **67**, 451-467 (2010).
[15] C. C. Hu, N. A. X. Luo, X. H. Yan, and W. Z. Shi, Traffic Flow Data Mining and Evaluation Based on Fuzzy Clustering Techniques, International Journal of Fuzzy Systems **13**, 344-349 (2011).
[16] S. Dong, D. D. Zhou, W. Ding, and J. Gong, Flow cluster algorithm based on improved K-means method, Iete Journal of Research **59**, 326-333 (2013).
[17] N. F. Huang, G. Y. Jai, H. C. Chao, Y. J. Tzang, and H. Y. Chang, Application traffic classification at the early stage by characterizing application rounds, Information Sciences **232**, 130-142 (2013).
[18] H.-M. An, M.-S. Kim, and J.-H. Ham, Application traffic classification using statistic signature, in Network Operations and Management Symposium (APNOMS), 2013 15th Asia-Pacific, 1-6 (2013).
[19] Y. Wang, Y. Xiang, W. L. Zhou, and S. Z. Yu, Generating regular expression signatures for network traffic classification in trusted network management, Journal of Network and Computer Applications **35**, 992-1000 (2012).
[20] B.-C. Park, Y. J. Won, M.-S. Kim, and J. W. Hong, Towards automated application signature generation for traffic identification, in Network Operations and Management Symposium, 2008. NOMS 2008. IEEE, 160-167 (2008).
[21] M. Ye, K. Xu, J. Wu, and H. Po, Autosig-automatically generating signatures for applications, in Computer and Information Technology, 2009. CIT'09. Ninth IEEE International Conference on, 104-109 (2009).
[22] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, BLINC: multilevel traffic classification in the dark, in ACM SIGCOMM Computer Communication Review, 229-240 (2005).
[23] M. Iliofotou, H. C. Kim, M. Faloutsos, M. Mitzenmacher, P. Pappu, and G. Varghese, Graption: A graph-based P2P traffic classification framework for the internet backbone, Computer Networks **55**, 1909-1920 (2011).
[24] M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli, Tunnel hunter: Detecting application-layer tunnels with statistical fingerprinting, Computer Networks **53**, 81-97 (2009).

Sung-Ho Yoon received the B.S. and M.S. degree in computer science from Korea University, Korea, in 2009 and 2011, respectively. He is currently a Ph.D. candidate student of Korea University, Korea. His research interests include Internet traffic classification, Internet security and network management.

Jun-Sang Park received the B.S. and M.S. degree in computer science from Korea University, Korea, in 2008 and 2010, respectively. He is currently a Ph.D. candidate student of Korea University, Korea. His research interests include Internet traffic classification and network management.

Myung-Sup Kim received his B.S., M.S., and Ph.D. degree in Computer Science and Engineering from POSTECH, Korea, in 1998, 2000, and 2004, respectively. From September 2004 to August 2006 he was a postdoctoral fellow in the Department of Electrical and Computer Engineering, University of Toronto, Canada. He joined Korea University, Korea, in 2006, where he is working currently as an associate professor in the Department of Computer and Information Science. His research interests include Internet traffic monitoring and analysis, service and network management, and Internet security.