JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 31, 1669-1692 (2015)

Traffic Identification Based on Applications using Statistical Signature Free from Abnormal TCP Behavior*

HYUN-MIN AN[1], SU-KANG LEE[1], JAE-HYUN HAM[1,2] AND MYUNG-SUP KIM[1,+]
[1]*Department of Computer and Information Science*
Korea University
Sejong, 30019 Korea
[2]*The 2nd R&D Institute-1*
Agency for Defense Development
Daejeon, 34188 Korea
E-mail: {queen26; sukanglee; jhham; tmskim}@korea.ac.kr

As network traffic becomes more complex and diverse from the existence of new applications and services, application-based traffic classification is becoming important for the effective use of network resources. To remedy the drawbacks of traditional methods, such as port-based or payload-based traffic classification, traffic classification methods based on the statistical information of a flow have recently been proposed. However, abnormal TCP behaviors, such as a packet retransmission or out-of-order packets, cause inconsistencies in the statistical information of a flow. Furthermore, the analysis results cannot be trusted without resolving the abnormal behaviors. In this paper, we analyze the limitations of traffic classification caused by abnormal TCP behavior, and propose a novel application-based traffic classification method using a statistical signature with resolving abnormal TCP behaviors. The proposed method resolves abnormal TCP behaviors and generates unique signatures for each application using the packet order, direction, and payload size of the first *N* packets in a flow, and uses them to classify the application traffic. The evaluation shows that this method can classify application traffic easily and quickly with high accuracy rates of over 99%. Furthermore, the method can classify traffic generated by applications that use the same application protocol or are encrypted.

Keywords: application-level traffic classification, application identification, statistical signature, signature-based classification, statistics-based classification

1. INTRODUCTION

The classification of network traffic flows based on the application or services that generated them is crucial for the effective management and operation of network resources and for assuring quality of service (QoS) and service-level agreements (SLAs). For these policies, fast and accurate traffic classification at the application layer is essential. Accurate real-time traffic classification is an important part of determining the reliability of monitoring and controlling the application traffic of individual applications [1, 2]. Traditionally, traffic classification was accomplished using TCP or UDP port numbers, or payload data inspection [3]. However, once applications started using random port numbers or port numbers assigned to other protocols, the reliability of port-based classification is diminishing. At the same time, the payload data inspection began

Received April 30, 2014; revised October 18, 2014; accepted December 2, 2014.
Communicated Meng Chang Chen.
* This research was supported by a Korea University Grant.
+ Corresponding author: tmskim@korea.ac.kr.

being limited in terms of complexity, privacy, and application encryption, which have made it difficult to effectively classify application traffic.

Several statistical-flow-information based methods have been introduced to remedy the drawbacks of traditional port-based and payload-based classification [4, 5]. Classification approaches that use statistical information have certain advantages. For example, they can be applied to encrypted traffic, the usage of which has been recently increasing. In addition, such approaches do not need to analyze the packet payload data, and thus can classify traffic quickly. One problem of a classification approach is that it must wait until the end of the flow to complete the statistical information. Therefore, the traffic cannot be classified in real time. To overcome this problem, methods utilizing the first N packets of a flow have been studied. However, these methods demand a computation cost for feature extraction when producing statistical information, and because their machine learning (ML) algorithms have an extremely high computation complexity, they cannot be applied to real-time classification in networks with a very large bandwidth. In addition, because these methods classify application traffic into application protocols, their results are also not categorized based on each application. When applications use the same protocol, these methods will classify several applications into one application protocol. Therefore, they cannot be applied to many different network management and operation policies that are adjusted to each application. Most of all, no studies have shown a process for resolving abnormal TCP behaviors, such as packet retransmissions and out-of-order packets.

TCP provides transparent data transfers to the upper layer (application layer) through a trustworthy data stream channel between two end hosts. If an error occurs during the middle of a session, the receiver throws the packet away and the sender retransfers the packet. Thus, when a packet retransmission occurs from such an error, the receiver resolves the problem by throwing the erroneous packet away. Out-of-order packets can be incurred when a previous packet takes longer than a following packet during the transfer process. The receiver can detect and resolve this type of problem by comparing the sequence number of the packets. However, there is no detection process or resolution to these problems available at the traffic collection point. Therefore, all packets are collected, and the packet order may differ at each collection point. These problems cause the packets to become disordered and affect the statistical flow information. Owing to this limitation, the features lose their consistency.

In this paper, we analyze abnormal TCP behaviors and propose an algorithm for resolving them. We also propose a traffic classification method based on applications using a statistical signature for resolving such behaviors. The signature represents the unique flow pattern of each application, which can be utilized to distinguish different applications. Our method generates statistical signatures for each application using statistical flow information from traffic traces of an application. Furthermore, our method classifies the application traffic easily and quickly during real network operations through a simple matching of new flows to the signatures of each application.

In the generation of signatures for an application, flows of the application's traffic traces are converted into flow vectors using the packet order, direction, and payload size of the first N packets of each flow. The flow vectors are then grouped according to the similarity between flow vectors using our flow grouping algorithm for identifying the unique flow patterns of an application. The groups of flow vectors are optimized and the

application signatures are extracted from each group using our group-optimization and signature-generation algorithm, respectively. For the classification of application traffic, the flow vector of a new flow in a real operation network is compared with each signature using our signature-matching algorithm to determine the application belonging to the flow.

The proposed method has four advantages. First, it can classify traffic based on the applications generating the traffic in real time, can be applied to networks with a large bandwidth, and is able to manage a large amount of traffic. It also uses only the packet order, direction, and payload size of the first N packets of a flow during traffic classification, and therefore does not need to analyze the packet payload data and can generate flow vectors without incurring a computation cost for the feature extraction from the flow. Our method also uses a simple comparison for signature matching with an extremely low computational complexity in comparison to ML algorithms. As a result, our method can operate effectively in real network operations. Second, it can obtain highly accurate classification results because it uses statistical signatures that reflect the unique flow patterns from each application. Our evaluation shows that our method can classify application traffic with high accuracy rates of more than 99.97%. Third, it can classify application traffic into each application, and not the application protocols, because it uses unique signatures for each application. Our evaluation also shows that our method can classify application traffic utilizing the same application protocol into the proper applications. Fourth, it resolves abnormal TCP behaviors that cause feature inconsistency, and is therefore more robust than other methods.

The remainder of this paper is organized as follows. Section 2 briefly reviews and summarizes previous work in this area. Section 3 analyzes different abnormal TCP behaviors and proposes algorithms for detecting and resolving these behaviors. Section 4 introduces our proposed classification method in detail. Section 5 describes our experimental method and analyzes the classification results. Some concluding remarks and areas for future work are finally provided in Section 6.

2. RELATED WORK

Several traffic classification methods utilizing statistical information of application traffic flows have recently been developed [4, 5]. These methods commonly use ML algorithms and the characteristic features (*e.g.*, port number, flow duration, inter-arrival time, and packet size) of the application traffic. Because they do not analyze the payload data, such methods can classify traffic faster than payload-based methods and no privacy problems are incurred. In addition, these methods can classify encrypted traffic. Furthermore, by using high-quality algorithms qualified for the ML field, these methods can classify traffic with highly accurate results.

Table 1 compares the recent ML-based traffic classification methods that use features as statistical information on the traffic flows. The Feature Extraction Range shows the range in a flow necessary for extracting the features for traffic classification. In other words, this column shows the amount of packets that must be investigated to complete the features. A Full flow indicates that the method requires flow completion to extract the flow features, which can be conducted at the end of the flow; therefore, it cannot classify application traffic in real time. A Partial flow means that the method uses part of a flow

for extracting the flow features. The Feature Computation Cost shows the computational overhead for the feature extraction. A value of Low means that the method does not require any computations for feature extraction. An Average value indicates that the method requires a simple computation for extracting features, and that the number of features used is less than ten. A value of High indicates that the method uses features requiring complex computations to be extracted, or uses more than ten features that require a simple computation. ML Algorithm indicates the machine-learning algorithm used in the proposed method, and it infers the computation complexity of the traffic classification. Classification Traffic Class shows the class of application traffic classified by the proposed method. Protocol indicates the application protocol, and Application is each individual application. For example, *N* Protocols, two applications means that the method can classify application traffic into *N* application protocols and two individual applications.

Table 1. Comparison of recent ML-based traffic classification using statistical flow information

Related Work	Feature Extraction Range	Feature Computation Cost	ML Algorithm	Classification Traffic Class
Bernaille *et al.* [6]	Partial flow	Low	*K*-Means, GMM, HMM	*N* Protocols, two applications
Tomaz Bujlow *et al.* [7]	Partial flow	High	C5.0	*N* Protocols, three applications
Y. Jin *et al.* [8]	Full flow	Medium	Modular architecture combines three linear ML Algorithms	*N* Protocols
J. Tan *et al.* [9]	Full flow	Medium	SVM optimized by Particle Swarm Optimization	*N* Protocols, one application
R. Yuan *et al.* [10]	Full flow	Medium	SVM	*N* Protocols
Runyuan Sun *et al.* [11]	Full flow	High	Probabilistic Neural Networks	*N* Applications
C. Yin *et al.* [12]	Partial flow	Low	HMM	*N* Protocols, five applications

Table 1 shows that some methods extract features at the end of a flow [8-11]; therefore, they cannot be applied to real-time traffic classification because they can determine the application of a new flow only after the flow finishes. To overcome this limitation, methods for extracting features in the first *N* packets of a flow have been studied [6, 7, 12], but the high feature computation cost or high computational complexity of the ML algorithm used in these methods makes them difficult to achieve real-time traffic classification in networks that have a very large bandwidth. In addition, most of these methods mainly classify application traffic into each application protocol, not each application, which is insufficient when applying the method to network management or operation policies that adjust or control individual application traffic. Finally, there have

been no studies resolving the limitations that occur from abnormal TCP behaviors or that can change the statistical information of TCP sessions.

To overcome the limitations of previous methods using statistical information, the proposed method resolves abnormal TCP behaviors and generates statistical signatures for each application in advance. During traffic classification, our method composes flow vectors from the first N packets of new flows without incurring any computation cost, and performs signature matching with linear computation complexity for application traffic classification. The proposed method classifies the application traffic into each individual application and has the advantage of traffic classification using statistical information.

3. ABNORMAL TCP BEHAVIOR

When a session is established using TCP, one of the transport layer protocols in TCP/IP, a stream that guarantees continuous data transfers, is provided. TCP provides transparent data transfers to the application layer through a trustworthy data-stream channel between two end hosts.

TCP uses the sequence (Seq) number for checking the transfer data sequence. The Seq number increases with the payload size of the transferred packet. TCP also uses an Acknowledge (ACK) number for checking whether the data transfer has been properly completed. When data are transferred with no errors, the receiver sends an ACK number indicating the transferred Seq number plus the transferred payload size to the sender. If the receiver detects an error in a packet, it throws the packet out and sends the Ack number for the discarded packet. The sender then retransmits the packet. If packets arrive at the receiver in a disordered manner for a certain reason (*e.g.*, the path of an earlier sent packet to the receiver is longer than that of a later packet), the receiver rearranges the packets in the proper order using each packet's Seq number. In this way, TCP can guarantee the correct order of a data stream.

However, TCP has no responsibility for correcting the packet order at a traffic collection point. If there are no such processes for guaranteeing a data stream, traces of collected traffic can contain several retransmitted packets and the sequence of packets is disordered. Because errors such as a packet retransmission and out-of-order packets occur irregularly, flows that are generated through the same behavior in an application can be collected in a different packet sequence and indicate different statistical information. Finally, the traffic traces collected with no way to handle such errors are hard to use for statistical-information based traffic classification. Therefore, we propose a method for eliminating packet retransmissions, and properly reordering the packet sequence at the traffic collection points.

3.1 Packet Retransmission

A packet retransmission influences the statistical information of a flow because a retransmitted packet is collected at the traffic collection point along with the original packet. This can change the information, such as the sequence, size, and the number of the packets.

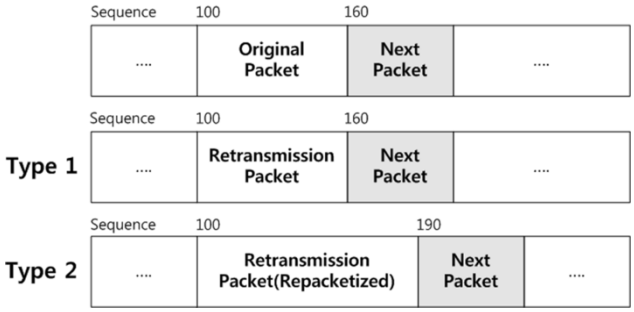


Fig. 1. Two types of packet retransmission.

There are two types of packet retransmission, each with its own process of elimination. In the first type of packet retransmission, the original packet and a retransmitted packet have the same payload size and Seq number. To eliminate this, we store the original packet and ignore the retransmitted packet for the original application behavior. In the second type of packet retransmission, the payload size of the retransmitted packet is bigger than that of the original packet because, when a packet retransmission is required, TCP repacketizes the packet by adding more data to the payload of the original packet to improve the performance. Fig. 1 shows the two types of packet retransmission. For Type 2, the payload size of the retransmitted packet is changed at the transport layer. In other words, with repacketized packets, the statistical flow information cannot reflect the original application behavior.

To correct the packet sequence for Type 2, we store the original packet and ignore the retransmitted packet. Furthermore, we discard packets that are related with the sequence of the retransmitted packet. To do so, we use the Seq number of each packet.

Remove all non-payload packets from the packet sequence

1: **procedure** Resolving Packet Retransmission
2: Input: $P(n)$ in a TCP flow
3: **find** $P(k)$ which $P(k)$.dir == $P(n)$.dir && biggest k in $0 < k < n$
4: **if**($P(k)$.seq == $P(n)$.seq)
5: **then** Delete $P(n)$
6: **if**($P(k)$.seq + $P(k)$.len < $P(n)$.seq)
7: **then** Delete $P(n)$
8: **end procedure**
9: Payload packet: a packet with payload data
10: Non-Payload packet: a packet without payload data
11: $P(n)$: nth payload packet in a TCP flow
12: $P(n)$.seq: nth payload packet's sequence number
13: $P(n)$.ack: nth payload packet's acknowledge number
14: $P(n)$.dir: nth payload packet's transmission direction
15: $P(n)$.len: nth payload packet's payload length

Fig. 2. Algorithm for resolving packet retransmission.

Fig. 2 shows the algorithm used for resolving a packet retransmission. When $P(n)$ is inputted, it is compared with the pre-stored packets that are transmitted in the same direction. If a packet retransmission is detected, $P(n)$ is deleted. The find $P(k)$ function checks only whether two packets have same direction or not, and the value of N is small enough to make the proposed method classifies the traffic in real-time. Therefore, find $P(k)$ function is light-weight function and its overhead can be ignored.

3.2 Out-of-order Packets

When a connection is established between two end hosts, packets can be transferred through different paths depending on the network status. If an earlier-sent packet is transferred through more routers than a later-sent packet, the later-sent packet may arrive at the receiver before the earlier-sent packet. In this situation, the packets are said to be out-of-order. Fig. 3 shows an example of out-of-order packets.

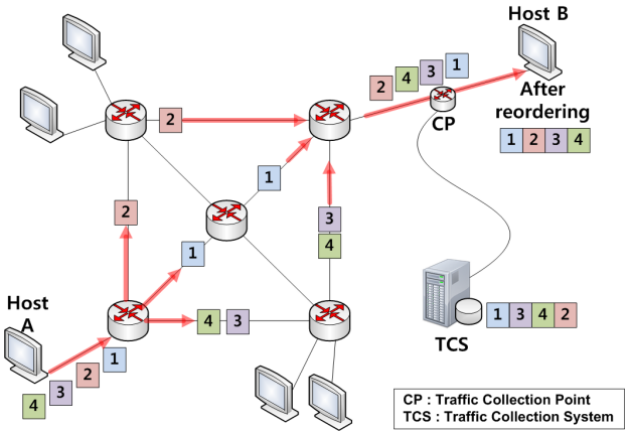


Fig. 3. Out-of-order packets.

The packet sequence at the receiver (Host B) is different from that of the sender (Host A). The packet sequence at the receiver is corrected automatically based on the TCP behavior. However, for traffic collection system (TCS), when traffic is collected at the collection point (CP), a reordering process is basically not provided. Therefore, the traffic is collected in a different sequence from the application behavior, which causes a lack of consistency in the features. Fig. 4 shows such inconsistency caused by out-of-order packets at CPs 1, 2, and 3. In this way, out-of-order packets influence the statistical flow information. For a trustworthy traffic classification, the problem of out-of-order packets must be resolved.

Fig. 5 shows the algorithm used for resolving out-of-order packets. When a packet is captured, our method checks whether a packet is retransmitted or not, and removes the packet when it is retransmitted. After that, it checks whether a packet is an out-of-order packet or not. Therefore, the process to resolve out-of-order packets can adjust the out-of-order packets regardless of retransmission packets.

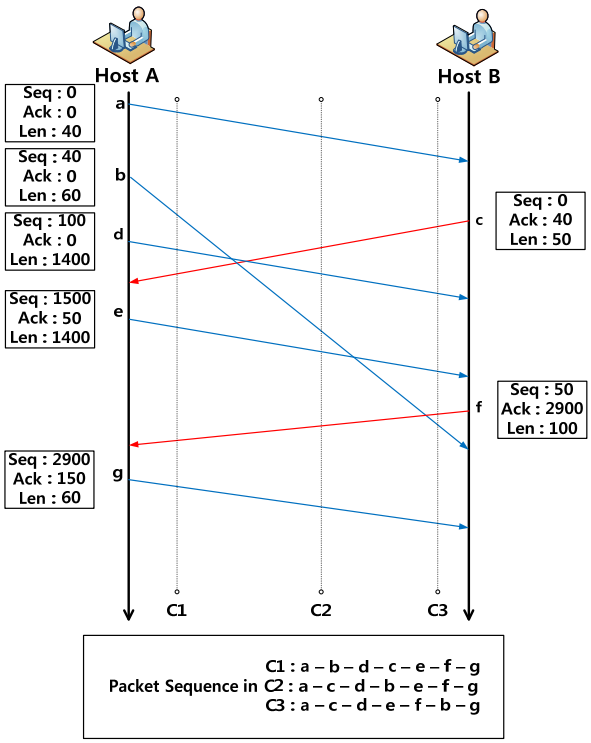


Fig. 4. The feature inconsistency caused by out-of-order packets.

Remove all non-payload packets from the packet sequence

```
1: procedure Resolve Packet Out-of-order
2:    Input : P(n) in a TCP flow
3:    find P(k) which P(k).dir == P(n).dir && biggest k in 0 < k < n
4:    if(P(k).seq > P(n).seq)
5:       find P(i) which P(i).dir == P(n).dir && P(i).seq < P(n).seq
6:                                          && biggest i in 0 < i < k
7:       find P(j) which P(j).dir == P(i).dir && smallest j in i < j < k
8:       for P(m) from P(j–1) to P(i+1)
9:          if(P(m).dir != P(n).dir && P(m).ack == P(n).seq + P(n).len)
10:            then put P(n) before P(m)
11:            end procedure
12:         if(P(m).dir != P(n).dir && P(m).ack < P(n).seq + P(n).len
13:            then put P(n) after P(m)
14:            end procedure
15:      end for
16:   put P(n) after P(i);
17:   end if
18:end procedure
```

Fig. 5. Algorithm for resolving out-of-order packets.

When $P(n)$ is inputted, it is compared with the pre-stored packets transmitted in the same direction. If there is a pre-stored packet that has a larger Seq number than $P(n)$, the out-of-order packet is detected and the situation is resolved. The latest transferred packet that has the same direction and a smaller Seq number than $P(n)$ is $P(i)$. The earliest transferred packet that has same direction and a larger Seq number than $P(n)$ is $P(j)$. $P(n)$ is located between $P(i)$ and $P(j)$ when there are no packets moving in the opposite direction. If there are packets between $P(i)$ and $P(j)$ that are moving in the opposite direction, the location of $P(n)$ is determined based on these packets. If we find a packet being transferred in the opposite direction that has the same Ack number as the Seq number of $P(n)$ plus payload size of $P(n)$, we put $P(n)$ in front of the packet. If we find a packet transferred in the opposite direction that has a smaller Ack number than the Seq number of $P(n)$ plus payload size of $P(n)$, we place $P(n)$ behind this packet.

4. TRAFFIC CLASSIFICATION METHODOLOGY USING STATISTICAL SIGNATURE

In this section, we describe the proposed traffic classification method after defining a flow vector and describing the characteristics required to distinguish application traffic. To classify application traffic, the signatures that are unique to each application are necessary because we classify traffic into individual application classes, instead of into application protocols. Compared with previous methods that classify traffic into application protocols, the proposed method can be utilized in a broad range of fields.

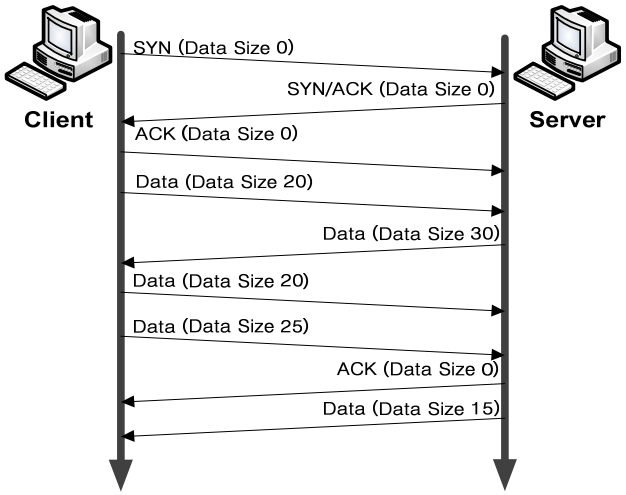
4.1 Flow Vector

In general, the first few packets of a flow communicate based on pre-defined rules of the application. The first N packets of a flow can be used as a distinguishable feature to identify the application because they communicate according to pre-defined rules and are extremely different in each application [6].

In this study, we define a flow as a set of packets with sequence that are transmitted in both directions based on a 5-tuple (source IP, destination IP, source port, destination port, and L4 protocol). The packet order is formed according to a standard order, which we defined previously, using the collection time.

The flow vector is presented with a sequence composed of the payload size and the direction of the first internal N packets based on the packet order. The payload size and transmission direction of each packet are expressed as an integer and "+/−," respectively. In the case of TCP, the transmission direction from the client to the server is defined as "+," and the opposite direction is defined as "−." In the case of UDP, because the distinction between the client and the server is not clear, the direction of the first packet is expressed as "+," and the opposite direction is determined as "−." The flow vector is composed only of packets that have a payload. The control packets, such as SYN or ACK, in the TCP sessions are excluded. This prevents irregular control packets from affecting the flow vector.

For example, if a flow communicates in both directions, as shown in Fig. 6, the flow vector has values of +20, −30, +20, +25, and −15, excluding the control packets, such as SYN, SYN/ACK, and ACK.



Flow Vector = (+20, -30, +20, +25, -15)

Fig. 6. The flow vector.

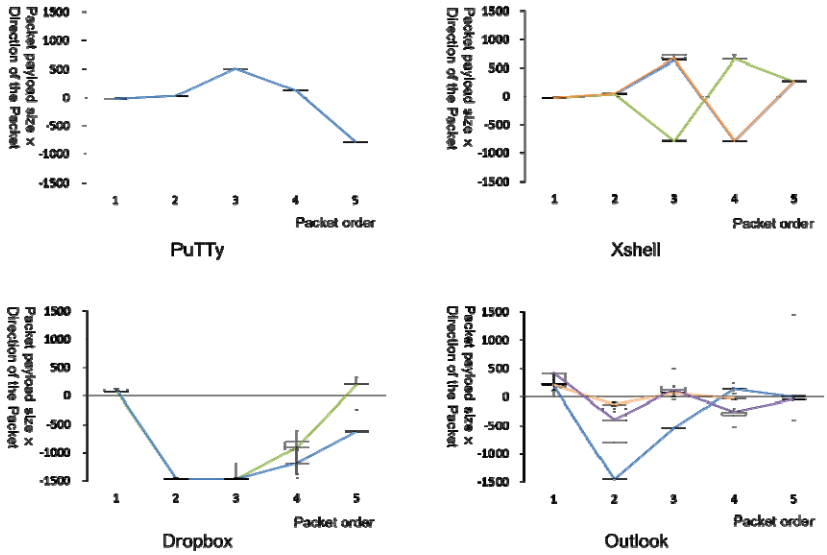


Fig. 7. The flow vectors of four different applications.

From the observations on popular applications used in the Korea University campus network, we found that applications can be distinguished based on their flow vectors. Fig. 7 shows the flow vectors from four different applications, Dropbox, Microsoft Outlook, PuTTY, and Xshell. All flow vectors that have the same packet order and direction are expressed as a group, which is plotted using a polygonal line. The vertical axis shows a multiplication of the payload size and direction of the packets. The horizontal axis shows the packet order. The multiplication of the payload size and direction of the packet can have a value ranging from −1,460 to 1,460.

Unlike PuTTY, most of the applications in Fig. 7 do not have flows that communicate using one identical flow vector pattern. The flows can be divided into either two or three specific patterns, such as Xshell and Dropbox, or divided into a number of specific patterns, such as Outlook. However, the flow vectors of each application show its own unique characteristics when compared with the flow vectors of the other applications.

Fig. 8 shows each flow vector from six different applications, *i.e.*, Xshell, PuTTY, Dropbox, Nateon, Skype, and Naverlive. These are projected onto a two-dimensional space to show the possibility of classification using a flow vector. We expressed only the flows of each application after removing the outlier flows through our flow-grouping and group-optimization algorithms. The flow grouping indicates the binding process of flows that have similar flow vectors to identify the specific flow patterns of each application. The outlier flows indicate the flows that cannot be regarded as a signature because the number of similar flow vectors is extremely small. The horizontal axis shows the first value of the flow vector, which indicates the multiplication of the payload size and the direction of the first packet of the flow. The vertical axis shows the fourth value of the flow vector, which is a multiplication of the payload size and the direction of the fourth packet of the flow. Each point denotes a pair (first and fourth values) of flow vectors of each application.

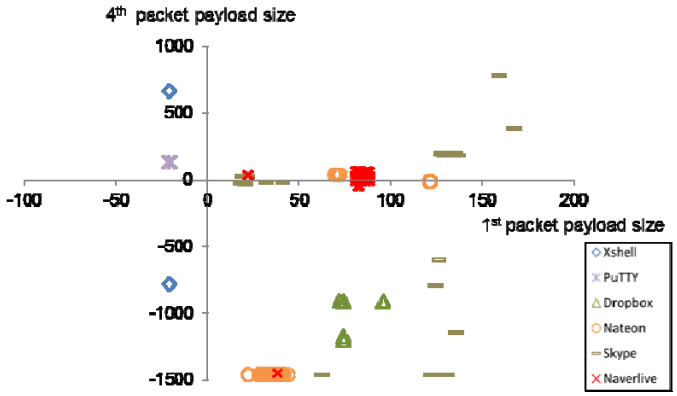


Fig. 8. Projection of the flow vectors of six applications onto two-dimensional Euclid space.

Originally, four flow groups are generated at Xshell, but three groups appear to be a single group on a two-dimensional space because they have the same value for the first and fourth packets. One group is generated for PuTTY, and four groups are generated for Dropbox, but two groups overlap because they have similar values for the first and fourth packets. For a similar reason, five groups are generated for Nateon, but three groups are marked as a single group. For Naverlive, nine groups are generated, but seven groups overlap. For Skype, 16 groups are generated, and some appear to be overlapped.

A flow group indicates a specific flow pattern or application behavior. As shown in Fig. 8, the flows of each application have regular patterns, even if the flow vectors are projected onto a two-dimensional space. Regular flow patterns can be used to distinguish between each application. Fig. 8 shows that, with the exception of Skype, the other five applications have less than ten repeated flow patterns. Because the application behavior

varies, a number of flow groups are generated for Skype. Nateon and Naverlive overlap, as shown in Fig. 8, but they have distinguishable values for the second, third, and fifth packets. The flow groups of Naverlive and Skype appear to be overlapping, but they are in fact slightly different in the other packets. Therefore, we know that the proposed flow vector can be used to classify application traffic. Furthermore, distinguishable flow vectors between PuTTY and Xshell, which use the same secure shell (SSH) protocol, indicate that the proposed flow vector can be used to classify application traffic that uses the same application protocol or is encrypted.

4.2 Statistical Signature

A signature represents a unique flow pattern of each application that can be utilized to classify application traffic. The proposed method vectorizes flows from traces of application traffic into flow vectors per application, and groups flow vectors based on their similarity to identify flow patterns of each application. In sequence, a signature is generated using a combination of a representative vector, which represents the flow vectors of each group, and a distance threshold vector, which includes the flow vectors of each group. If all of the individual flow vectors are used as a signature, the number of signatures increases. As a result, managing the signatures becomes difficult, and the system becomes overloaded when identifying the application traffic. Therefore, an optimal signature combines the representative vector and distance threshold vector that represent each group by grouping the application flow vectors.

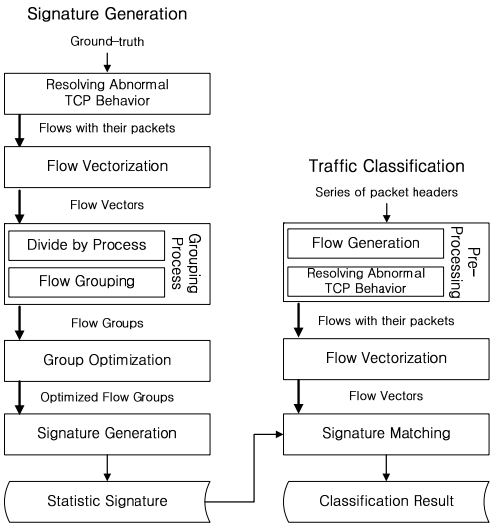


Fig. 9. Flowchart of application traffic classification using statistical signature.

Fig. 9 shows a flowchart of the proposed application traffic classification method using statistical signatures. Our method is divided into two stages: signature generation and traffic identification. In the signature generation stage, the first step is to remove abnormal TCP behaviors from the traces of traffic. Next, application flows of traces of

ground truth traffic are vectorized into flow vectors, and the flows are grouped based on the similarity between two flow vectors. Finally, signatures are extracted after optimizing the groups. In the traffic identification stage, a new flow is generated from a series of packets during a real network operation. Next, abnormal TCP behaviors in the flow are removed, and the flow is vectorized into a flow vector using the first N packets. The flow vector is classified into each application through signature matching with each signature. The proposed method needs first N packets of each flow which don't have abnormal TCP behavior for feature. Therefore, when the packet retransmission occurs in a flow, it collects packets more than the number of N for making up the N packets excluding the retransmission packet.

4.2.1 Statistical signature generation

Statistical signature generation starts by removing abnormal TCP behaviors from the traces of traffic. In the second step, flows are vectorized into a flow vector. The flow vector v_k of flow f_k can be expressed as Eq. (1), and each element $v_{k,i}$ of v_k can be written as Eq. (2). Here, $d_{k,i}$ is the transmission direction of the ith packet of f_k, and its value is either +1 or −1. In addition, $s_{k,i}$ is the payload size of the ith packet of f_k.

$$v_k = (v_{k1}, v_{k2}, \ldots, v_{kn}) \tag{1}$$

$$v_{k,i} = d_{k,i} \times s_{k,i} \tag{2}$$

In the third step, flow vectors are grouped based on their similarity. The similarity between two vectors is expressed as the distance vector that has elements as distance between two vectors by each dimension. Similarity is used in the flow grouping and signature matching algorithms. The distance vector $d_{x,y} = d(v_x, v_y)$ between v_x and v_y can be written as (3).

$$d\left(v_x, v_y\right) = \left(\left|v_{x,1} - v_{y,1}\right|, \left|v_{x,2} - v_{y,2}\right|, \ldots, \left|v_{x,n} - v_{y,n}\right|\right) \tag{3}$$

Statistical signature s is represented using a representative vector and distance threshold vector, as in Eq. (4), where **c** is the centroid vector of the flow vector group; t is the distance threshold vector, which can include every flow vector of the flow vector group; and s is the combination of c and t of the flow vector group.

$$s = (c, t) \tag{4}$$

Each flow vector v of flow vector group $V(s)$, which is represented by the signature $s = (c, t)$, should satisfy Eq. (5). The similarity between all v that belong to $V(s)$ and the representative vector c should be less than or equal to distance threshold vector t.

$$d(v, c) \le t \text{ for } \forall v \in V(s) \tag{5}$$

Multiple signatures of each application can exist, and Eq. (6) shows signature set S.

$$S = \{s_1, s_{,2}, \ldots, s_n\} \tag{6}$$

The flow vectors of each application are extracted from flows of ground truth traffic traces that are collected in advance. The statistical signature set is then generated based on the flow vectors. Application traffic can be classified according to the signature set.

$$\hat{S}\ such that \begin{cases} \text{minimize} |S| \\ \text{minimize} \sum t(s) \\ \text{maximize} \sum |V(s)| \end{cases} \tag{7}$$

To obtain an optimal signature set $\hat{S}$ from the given ground truth flows of each application, the conditions in Eq. (7) should be satisfied. Here, $|S|$ indicates the number of signatures s of S, $t(s)$ is the distance threshold vector of signature s, and $|V(s)|$ is the number of flow vectors of set $V(s)$ represented by signature s. Thus, Eq. (7) indicates the minimization of the number of signatures, the minimization of the sum of distance threshold vectors, and the maximization of the number of flow vectors that belong to each signature for obtaining an optimal signature set.

The fourth step is the group optimization step. To generate the optimal signature set $\hat{S}$, the group optimization step removes the inappropriate outlier flow vectors, along with the outlier groups after completing the flow grouping. The outlier flow vectors that are inside the group and do not meet the requirement of Eq. (8) are removed. Because c is recalculated every time a flow is grouped, the flows that are contained in a group can be changed. Removing outlier flows is performed once at a flow and it just calculates distance between the flow vector and the centroid vector of the flow vector group. Therefore, the time complexity of removing outlier flows is $O(n)$. In addition, the outlier groups that satisfy the condition of Eq. (9) are removed because the groups that contain a significantly small number of flows cannot represent the behavior of a specific application. Removing outlier groups is performed once at a group and it just compares the number of flows in the group and the threshold value. Therefore, the time complexity of removing outlier groups is $O(n)$.

$$d(v_j\, c_i) \le t_i \text{ for } \forall\ v_i \in V_i\ c_i t_i \text{ of } G_i \tag{8}$$

$$|V_i| = \sum_{j=1}^{m} \tau_{i,j} < \text{minimal flow count} \tag{9}$$

In the final step, signature set $\hat{S}$ is generated, and has five attributes similar to those in Table 2. In Table 2, ID is the identity of the application for the signature. Proto is an $L4$ protocol for the signature, and has a value of either TCP or UDP. Flow grouping is achieved by flows that have the same $L4$ protocol, even if the flows are generated from the same application. Dim is the dimension of the flow vectors that belong to a signature. All flows that belong to the same signature have the same dimension. Because the dimension indicates the number of internal packets of the flow that are vectorized into the flow vector, the value of the dimension can range from one to N. However, the minimum value is defined as three in this study because it is difficult to generate a sensible signature using one or two-dimensional flow vectors. In addition, N is empirically defined as five when considering real-time and accurate traffic classification.

Bernaille *et al.* [13] also showed that the first five packets of a flow are sufficient to distinguish each application. C-vector is the centroid vector of the signature and is calculated using all flow vectors that belong to the signature. T-vector is the condition of flow grouping and indicates the range of the group. The initial value of T-vector must be able to include the largest number of flows that indicate the same flow pattern, and can exclude flows that indicate different patterns. C-vector is used along with T-vector to identify the application of a new flow during traffic classification.

Table 2. Attributes of a flow group.

Attribute	Description	Example
ID	Application name	bittorrent
Proto	L4 protocol: TCP/UDP	UDP
Dim	Dimension of vector	5
C-vector	Centroid vector	(+20,+30,-50,+20,-30)
T-vector	Distance threshold vector	(2, 5, 8, 4, 5)

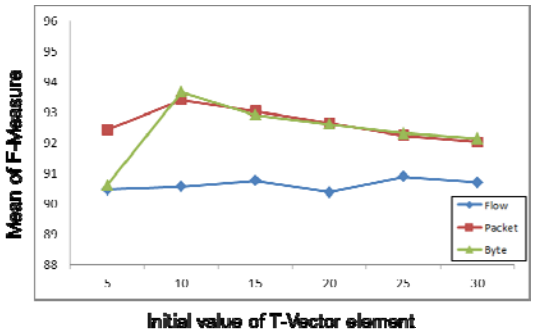


Fig. 10. Optimal initial value of T-vector element according to the F-measure.

Fig. 10 shows the classification performance of the proposed method using different initial value of T-vector element. Performance is evaluated by mean of ten applications f-measure. In the packet and byte perspective, using 10 for the initial value of T-Vector element shows best performance. It is because, if the initial value of T-vector element increases, the flows of different applications can be grouped together. However, there are some variations in the flow perspective. In the network management, the amount of the packet and byte are more important than the number of the flow because the packet and byte occupy the bandwidth indeed. Therefore, in this study, the initial value of every T-vector element is set 10 to identify unique flow patterns of each application properly in the flow grouping.

4.2.2 Traffic identification

In traffic identification, the traffic of a real operation network is captured and classified based on the signature of each application. A new flow is generated with a series of

packets captured from a real network operation. Abnormal TCP behaviors in a flow are removed, and the flow is vectorized into a flow vector using the packet order, direction, and payload size of its first N packets. The flow vector is compared with each signature to determine its identity. If the flow vector of a new flow is included in a signature by the C- and T-vectors, its application is determined as the ID of the signature.

Thus, traffic identification finds signature s_i shown in Eq. (10) for the flow vector v of a new flow. At this time, s_i should have the same L4 protocol and dimension as v.

$$s_i \text{ such that } d(c_i, v) \le t_i \text{ for } s_i = (c_i, t_i) \tag{10}$$

In this study, a flow vector uses only the packet order, direction, and payload size of the first five packets of a flow; therefore, flow vectorization for converting a flow into a flow vector using these features does not need any computational costs. In addition, signature matching is a simple similarity comparison operation with a linear computation complexity, as shown in Eq. (10). Therefore, traffic classification using these simple flow vectorization and signature-matching algorithms does not require high computational costs in comparison to previous methods, which enables real-time traffic classification in networks that have a very large bandwidth.

Table 3. Ground truth traffic.

Traffic Class	Applications	Flow (10^3)	Packet (10^3)	Byte (10^6)
Skype	P2P communications	2.9	43.9	17.6
Naverlive	Video streaming	2.6	50,807.5	41,404.3
GomTV	Internet TV service	15.2	2,637.0	2,515.6
Xshell	Telnet/SSH client	1.2	177.1	22.8
Teamviewer	Remote control	1.8	722.6	291.7
Nateon	Instant messaging	0.9	337.1	62.5
Dropbox	Cloud file sharing	11.1	294.8	158.5
PuTTY	Telnet/SSH client	0.7	80.4	11.5
Outlook	MS mail service	12.7	1,150.4	692.3
uTorrent	P2P download	1,116.4	62,151.0	49,494.5

5. EVALUATION

In this section, we describe the results of a traffic classification test on a campus network to verify the proposed traffic classification using statistical signatures.

For our evaluation, we collected bi-directional packet traces from the Korea University campus network, which was configured with one router at the Internet junction, and we collected the traces of traffic from the router using port mirroring.

To evaluate the traffic classification, it is crucial to obtain a firm ground truth. We deployed traffic measurement agents (TMAs) on selected hosts in the campus network and created ground truth traffic [14]. Using the ground truth traffic through these agents is more reliable for evaluating the proposed classification method than using the results of another particular classification method [15].

Table 3 shows a summary of the ground truth traffic we obtained for our evaluation. The traffic was arranged by each application, verifying that our method can classify the traffic into each application. Because the traffic was collected over a period of time, the traffic volume varies according to the frequency of use of each application and the amount of traffic generated.

We divided the ground truth traffic into two different sets obtained on different dates. One traffic set was used only for signature generation. The other traffic set was used only for the traffic classification test. Table 4 shows the traffic flows for signature generation, and the number of signatures that were generated for each application.

We use the completeness and accuracy as evaluation metrics in accordance with most of the traditional literature [3, 4, 16]. Completeness is a metric of how much traffic was classified. Accuracy is a metric indicating the rate of traffic that was correctly classified. Accuracy is determined by comparing the classification results with the ground truth. Further, accuracy is divided into the overall accuracy and the accuracy per application, which represents the precision and recall for each application. These evaluation metrics are expressed by the flow, packet, and byte to provide more detailed information.

The proposed method classifies a large amount of ground truth traffic in just a few minutes, which indicates its real-time traffic classification capability. Table 5 shows the overall accuracy and completeness of the traffic classification test in terms of flows, packets, and bytes. The Unresolved case indicates the results of a test without the removal process for abnormal TCP behaviors, and the Resolved case indicates the results of the test after all TCP abnormal behaviors were removed.

Table 4. Traffic flows for signature generation and the number of signatures for each application.

Traffic Class	Flow	Signature
Skype	1,000	16
Naverlive	1,138	9
GomTV	2,229	9
Xshell	730	4
Teamviewer	842	9
Nateon	456	5
Dropbox	6,232	4
PuTTY	382	1
Outlook	5,774	49
uTorrent	64,773	192

Table 5. Overall accuracy and completeness using proposed method.

	Overall Accuracy		Completeness	
	Unresolved case	Resolved case	Unresolved case	Resolved case
Flow	99.83%	99.95%	68.42%	68.43%
Packet	99.89%	99.97%	85.01%	87.71%
Byte	99.90%	99.97%	84.84%	87.98%

For the unresolved case, the results show that the proposed method can achieve high accuracy rates of greater than 99.83% for all units for application traffic that were classified. From the perspective of completeness, the flow completeness was 68.42%, but the packet and byte completeness were 85.01% and 84.84%, respectively. The reason for the low flow completeness compared to the packet and byte completeness is the existence of uTorrent traffic. The classification results of uTorrent, which occupies the largest ground truth, have a significant impact on the flow completeness. uTorrent traffic occupies a maximum of 95% of ground truth flows, but only 5.8% of the uTorrent flows are used to generate signatures. It is difficult to analyze the remaining 94.2% of flows using the signatures from 5.8% of flows. However, the uTorrent classification results show a maximum packet classification of 81.25% and a byte recall of 81.13%, which affect the packet and byte completeness. This shows that the signatures of uTorrent correctly classify heavy flows, such as file downloads with several packets and bytes, which are more crucial for traffic monitoring and network management [17].

For the resolved case, the overall accuracy rates for all units of application traffic are higher than 99.95%. This result is higher than for the unresolved case. Furthermore, the flow, packet, and byte completeness are 68.43%, 87.71%, and 87.98%, respectively. These results are also higher than for the unresolved case. These results show that resolving abnormal TCP behaviors can improve the performance, even when using the same method.

Tables 6 and 7 show the precision and recall for each application, respectively, which are responsible for the overall accuracy and completeness. The results of the unresolved case show that the proposed method can achieve high precision rates of more than 99.79% for every application for all units, even when a few recall rates are relatively low and affect the overall completeness.

Table 6. Precision of each application using proposed method.

Traffic Class	Flow		Packet		Byte	
	Unresolved case	Resolved case	Unresolved case	Resolved case	Unresolved case	Resolved case
Skype	99.85%	99.96%	99.85%	99.97%	99.85%	99.97%
Naverlive	99.85%	99.88%	99..90%	99.91%	99..91%	99.91%
GomTV	99.81%	99.93%	99.85%	99.93%	99.87%	99.93%
Xshell	100%	100%	100%	100%	100%	100%
Teamviewer	99.85%	99.94%	99.85%	99.96%	99.85%	99.96%
Nateon	99.80%	99.91%	99.83%	99.92%	99.83%	99.92%
Dropbox	99.85%	99.90%	99.87%	99.90%	99.87%	99.90%
PuTTY	100%	100%	100%	100%	100%	100%
Outlook	99.79%	99.93%	99.80%	99.94%	99.81%	99.94%
uTorrent	99.83%	99.95%	99.89%	99.97%	99.89%	99.97%

For the unresolved case, each precision rate for every application reaches almost 100%. The flow precision rates are greater than 99.88% for all applications for all units, and the packet and byte precision rates are over 99.90%. Furthermore, all precision rates

Table 7. Recall of each application using proposed method.

Traffic Class	Flow		Packet		Byte	
	Unresolved case	Resolved case	Unresolved case	Resolved case	Unresolved case	Resolved case
Skype	65.70%	69.59%	68.90%	72.70%	81.18%	86.12%
GomTV	70.10%	71.56%	90.26%	90.35%	90.44%	90.51%
Naverlive	83.26%	84.24%	89.42%	90.96%	89.47%	91.07%
Nateon	90.67%	91.30%	98.60%	98.64%	96.38%	96.41%
Outlook	98.12%	98.50%	99.41%	99.80%	99.16%	99.55%
Xshell	96.53%	96.28%	97.58%	97.29%	97.57%	97.29%
PuTTY	72.15%	70.20%	65.45%	64.81%	59.08%	58.97%
Teamviewer	99.05%	99.35%	98.72%	99.06%	98.31%	98.71%
Dropbox	84.69%	88.87%	84.98%	85.15%	84.48%	84.56%
uTorrent	66.50%	66.43%	81.25%	85.15%	81.13%	85.69%

for PuTTY and Xshell reach 100%. The PuTTY and Xshell use same protocol, SSH. Even if the applications use the same application protocol, still they can generate different payload data in the same procedure of pre-defined rules because there can be different information of application, and the applications can customize the protocol in the permissible range from protocol manual for their convenience. There are distinct differences between Xshell and PuTTY in the direction and payload size of first five packets based on packet order, although they use the same application protocol (SSH). First, the sizes of the second packet are different from each other. The second packet contains the SSH version and client program information. The difference in the payload size occurs because the client program information of the two applications is different. The payload size of second packet of Xshell is 49 bytes, and that of PuTTY is 28 bytes. Second, the Xshell client transmits only the third packet for "key exchange init", but the PuTTY client transmits the third and fourth packets for "key exchange init". Therefore, the payload sizes of the third and fourth packets are different, additionally fourth packet has different transmission direction for the two applications, which affects payload size and transmission direction of successive packets such as the fifth packet. The proposed method generates the signatures by application unit, so it can extract the characteristics of individual application and can classifies the applications that use same protocol into individual applications. For the resolved case, the performance indicates higher rates than for the unresolved case. However, misclassifications still exist. A misclassification is incurred by different abnormal behavior in each application. When an end host sends its data stream regardless of the transferring data of opponent end host, the packet sequence is disordered. There is no correct answer for this problem because there is no intension of it. It occurs in the full-duplex TCP session or UDP session. Traffic classification robust to this abnormal behavior of applications is an important topic for our future research.

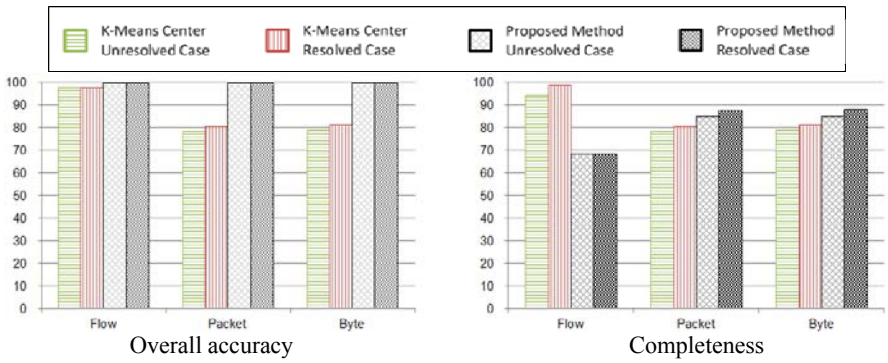
A low flow completeness problem is shown in the flow recall rates of a few of the applications, such as Skype, Naverlive, Xshell, and uTorrent. The flow recall rate is also

less than the packet and byte recall rates for most of the applications. This is because of flows that have only one or two packets. We defined the minimum value for the dimension of the flow vector as three, and generated the signatures. However, we used only real traces of traffic of the applications studied in this test, and the signatures cannot classify flows with only one or two packets. In addition, flows of TCP sessions that terminated abnormally in traffic traces contributed to the low recall rates. Naverlive has the largest gap between the flow and packet (or byte) recall. This means that the recall of small flows that have only a few of packets (or bytes) and are unrelated to streaming is low, but the recall of large flows that have several packets (or bytes) and are related to streaming is high.

The recall rate of each application is also affected by the signature conflict policy. For this study, our method classifies a new flow that is matched to several signatures with different IDs as an unknown flow to provide high accuracy rates. If other policies are applied, the recall rate can be improved. For example, our method can classify an unknown flow into the application of the signature that has the highest similarity [18], use consecutive port numbers [19] or server-specific port numbers [20], or report applications with signatures that are matched to the unknown flow for the network manager to determine its application. In the future, we will study the signature conflict to improve the recall rate of each application. Performance improvement was achieved in the recall of each application. Some applications, but not all, that use more full-duplex sessions than others showed specific improvements in performance.

We compare the performance of the proposed method against that of the method that is introduced by Bernaille *et al.* [6]. Bernaille *et al.* [6] proposed the use of clustering techniques to achieve fine-grained classification based on size and direction of packets. They used partial flow feature and their feature computation cost is low enough to classify traffic in real-time. We implemented and evaluated *K*-Means Center from [6] according to the paper [6] faithfully. The *K* is decided as 170 for the unresolved case and 160 for the resolved case by experimental basis likewise introduced in paper [6].

Fig. 11 shows the results of performance comparison of the proposed method and *K*-Means Center. The performances of *K*-Means Center are increased after resolving the abnormal TCP behaviors like proposed method. It evidently shows that if there are the abnormal TCP behaviors, *K*-Means Center cannot classify the traffic accurately because of the feature inconsistency occurred by abnormal TCP behaviors.



Overall accuracy Completeness

Fig. 11. Comparing performance of two methods.

Shown as left side of Fig. 11, the proposed method achieves higher accuracy than *K*-Means Center in the flow, packet, and byte perspectives. However, in the completeness perspective, shown as the right side of the Fig. 11, the *K*-Means Center achieves higher flow completeness than the proposed method. This is because the *K*-Means Center can classify the almost uTorrent traffic while the proposed method classifies only heavy uTorrent traffic. However, the proposed method achieves higher completeness in packet, and byte perspective because the proposed method can classify the heavy flows better than *K*-Means Center. In the network management, the amount of the packet and byte are much more important than the number of the flow because the packet and byte occupy the bandwidth indeed. Except the flow completeness, the proposed method shows higher performance than *K*-Means Center. This is because the proposed method uses N-dimensional similarity for measuring similarity between flow vectors while the *K*-Means Center uses Euclidean distance that is only 1-dimensional similarity regardless of the number of dimension of feature. Using *N* integers and *N* thresholds are much stricter than using one real number and one threshold, obviously. In addition, the proposed method groups the flow vectors which are generated from the same process, while *K*-means Center clusters the total flow vector at once. Therefore, the proposed method can extract the characteristic of individual application more accurately.

6. CONCLUSIONS

In this paper, we analyzed abnormal TCP behaviors and proposed an application-based traffic classification method that uses statistical signatures by resolving such behaviors. Abnormal TCP behaviors are the problems that should be resolved for robust traffic classification because they result in feature inconsistency. A statistical signature represents the unique flow pattern of each application and can be used to distinguish applications. Statistical signatures for each application are generated by our flow grouping, group optimization, and signature-generation algorithms. Our method then classifies new flows into individual applications through signature matching in real network operations.

Our method can be effectively applied to real-time traffic classification in networks that have a very large bandwidth because it does not require any computational cost for feature extraction, and because signature matching is a simple similarity comparison. Our evaluation shows that the proposed method can classify application traffic easily and quickly with high accuracy rates of more than 99% for every application for all units when all abnormal TCP behaviors are resolved. We also evaluated the performances by comparing the proposed method and *K*-Means Center that is introduced by Bernaille *et al.* [6]. The proposed method achieves higher performance in the all perspectives except the flow completeness. However, in the network management, the amount of the packet and byte are much more important than the number of the flow because the packet and byte occupy the bandwidth indeed. The evaluation also showed that our method can classify traffic into each application that uses the same application protocol or encrypts its payload. Therefore, our method can be applied to various types of network management and operations that must control individual applications with high accuracy. In addition, the resolution of abnormal TCP behaviors was verified through performance improve-

ments for the resolved cases.

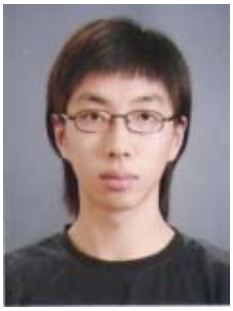
Our future studies will focus on three areas. First, we will conduct a study on signature conflicts to improve the completeness and recall ability of our method. Second, we intend to rearrange the UDP packets according to the behaviors of each application to increase the completeness and accuracy. Third, we will study an accurate and detailed classification method for HTTP traffic by extending our proposed method.

REFERENCES

1. M. S. Kim, Y. J. Won, and J. W. K. Hong, "Application-level traffic monitoring and an analysis on IP networks," *ETRI Journal*, Vol. 27, 2005, pp. 22-42.
2. T. T. T. Nguyen, G. Armitage, P. Branch, and S. Zander, "Timely and continuous machine-learning-based classification for interactive IP traffic," *IEEE/ACM Transactions on Networking*, Vol. 20, 2012, pp. 1880-1894.
3. A. Callado, C. Kamienski, G. Szabo, B. Gero, J. Kelner, S. Fernandes, *et al.*, "A survey on internet traffic identification," *IEEE Communications Surveys and Tutorials*, Vol. 11, 2009, pp. 37-52.
4. T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys and Tutorials*, Vol. 10, 2008, pp. 56-76.
5. A. Dainotti, A. Pescape, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE Network*, Vol. 26, 2012, pp. 35-40.
6. L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in *Proceedings of ACM CoNEXT Conference*, 2006, Article No. 11.
7. T. Bujlow, T. Riaz, and J. M. Pedersen, "A method for classification of network traffic based on C5.0 machine learning algorithm," in in *Proceedings of International Conference on Computing, Networking and Communications*, 2012, pp. 237-241.
8. Y. Jin, N. Duffield, J. Erman, P. Haffner, S. Sen, and Z. L. Zhang, "A modular machine learning system for flow-level traffic classification in large networks," *ACM Transactions on Knowledge Discovery from Data*, Vol. 6, 2012, pp. 1-34.
9. J. Tan, X. Chen, and M. Du, "An internet traffic identification approach based on GA and PSO-SVM," *Journal of Computers*, Vol. 7, 2012, pp. 19-29.
10. R. Yuan, Z. Li, X. Guan, and L. Xu, "An SVM-based machine learning method for accurate internet traffic classification," *Information Systems Frontiers*, Vol. 12, 2010, pp. 149-156.
11. S. Runyuan, Y. Bo, P. Lizhi, C. Yuehui, Z. Lei, and J. Shan, "Traffic classification using probabilistic neural networks," in *Proceedings of International Conference on Natural Computation*, 2010, pp. 1914-1919.
12. C. Yin, S. Li, and Q. Li, "Network traffic classification via HMM under the guidance of syntactic structure," *Computer Networks*, Vol. 56, 2012, pp. 1814-1825.
13. L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *ACM SIGCOMM Computer Communication Review*, Vol. 36, 2006, pp. 23-26.
14. B. C. Park, Y. J. Won, M. S. Kim, and J. W. Hong, "Towards automated application

signature generation for traffic identification," in *Proceedings of IEEE Network Operations and Management Symposium*, 2008, pp. 160-167.
15. F. Gringoli, L. Salgarelli, M. Dusi, N. Cascarano, F. Risso, and K. C. Claffy, "GT: picking up the truth from the ground for internet traffic," *ACM SIGCOMM Computer Communication Review*, Vol. 39, 2009. pp. 12-18.
16. H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: myths, caveats, and the best practices," in *Proceedings of ACM CoNEXT Conference*, 2008, Article No. 11.
17. K. C. Lan and J. Heidemann, "A measurement study of correlations of internet flow characteristics," *Computer Networks*, Vol. 50, 2006, pp. 46-62.
18. R. T. Gu, H. X. Wang, Y. M. Sun, and Y. F. Ji, "Fast traffic classification using joint distribution of packet size and estimated protocol processing time," *IEICE Transactions on Information and Systems*, Vol. E93D, 2010, pp. 2944-2952.
19. C. N. Lu, C. Y. Huang, Y. D. Lin, and Y. C. Lai, "Session level flow classification by packet size distribution and session grouping," *Computer Networks*, Vol. 56, 2012, pp. 260-272.
20. J. S. Park, S. H. Yoon and M. S. Kim, "Performance improvement of the payload signature based traffic classification system using application traffic temporal locality," in *Proceedings of Asia-Pacific Network Operations and Management Symposium*, 2013, pp. 1-6.

Hyun-Min An received his B.S. degree in Computer Science from Korea University, Korea, in 2012, and his M.S. degree in Computer Science from Korea University, Korea, in 2014. He is currently a researcher of Research Institute for Advanced Industrial Technology, Korea University, Korea. His research interests include Internet traffic classification and network management.



Su-Kang Lee received his B.S degree in Computer Science from Korea University, Korea, in 2014. He is currently a master's student of Korea University, Korea. His research interests include Internet traffic classification and network management.

Jae-Hyun Ham received his B.S. degree in Computer Science and Engineering from Dongguk University, Korea, in 1999, and his M.S. degree in Computer Science and Engineering from Postech, Korea, in 2001. He joined the Agency for Defense Development, Korea, in 2001, where he is working currently as a Senior Researcher in the Department of the 2nd R&D Institute-1. He is also currently a Ph.D. student of Korea University, Korea. His research interests include tactical network management, and traffic monitoring and analysis.

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.