IEICE TRANSACTIONS

on Information and Systems

VOL. E97-D NO.
OCTOBER 2014

A PUBLICATION OF THE INFORMATION AND SYSTEMS SOCIETY
The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER

A Lightweight Software Model for Signature-Based Application-Level Traffic Classification System

Jun-Sang PARK[†], **Sung-Ho YOON**[†], **Youngjoon WON**[††], ***Nonmembers***, ***and*** **Myung-Sup KIM**[†a)], ***Member***

SUMMARY Internet traffic classification is an essential step for stable service provision. The payload signature classifier is considered a reliable method for Internet traffic classification but is prohibitively computationally expensive for real-time handling of large amounts of traffic on high-speed networks. In this paper, we describe several design techniques to minimize the search space of traffic classification and improve the processing speed of the payload signature classifier. Our suggestions are (1) selective matching algorithms based on signature type, (2) signature reorganization using hierarchical structure and traffic locality, and (3) early packet sampling in flow. Each can be applied individually, or in any combination in sequence. The feasibility of our selections is proved via experimental evaluation on traffic traces of our campus and a commercial ISP. We observe 2 to 5 times improvement in processing speed against the untuned classification system and Snort Engine, while maintaining the same level of accuracy.

key words: *Internet traffic classification, payload signature, processing speed, signature hierarchy*

1. Introduction

As individual and corporate users become progressively dependent on the Internet, network speeds are increasing and a variety of services and applications are being developed. Therefore, effective monitoring and analysis of Internet traffic from an application perspective is necessary for efficient network operation and management of various commercial services such as pay-for billing, CRM and SLA.

Internet application traffic identification involves a series of processes ranging from capturing packets (traffic) on a target network link to identifying and labeling packets for categorization purposes. In previous studies, categorization was not associated with an application name but rather an L7 protocol name [1]–[3]. However, criteria for application naming are needed because many applications can use an L7 protocol such as hypertext transfer protocol (HTTP) for various purposes. Distinguishing these types of HTTP traffic based on their applications rather than on a single type of HTTP traffic is reasonable. Therefore, the number of signatures necessary to identify applications has been increasing. As the number and complexity of signatures increase, the processing speed of a payload signature-based classification system has become a critical element in determining the performance of a traffic classification system (TCS).

However, processing speed has been a concern in handing real-time traffic on high-speed networks [4], [5]. Speed-downgrading issues exist such as complexity and memory usage of matching algorithms and the scalability of hardware approaches (e.g., ASIC, FPGA).

In this paper, we propose three software-based techniques to improve the processing speed of a payload signature traffic classifier. These include (1) the selective matching algorithms based on signature type, (2) signature reorganization using hierarchical structure and traffic locality, and (3) early packet sampling in flow. Each technique can be applied individually or in combination.

In particular, we address factors affecting the processing speed of a payload signature classifier by examining input data. Based on the various experimental analyses of these performance factors, our proposed design techniques attempt to minimize the search space of input traffic data as well as that of signatures in the classification system. Signature reorganization especially reduces the signature search space through a two-level signature hierarchy. Furthermore, signature reorganization determines not only L7 protocol naming but also application naming. Finally, we propose the most effective combination of signature-matching algorithms to several different forms of payload signatures.

We evaluate the feasibility of our methods by means of campus and commercial ISP traffic traces. We rely on our previous implementation of an untuned classification system and in this paper, refer to the baseline system, which we compare to the original and Snort systems [7]. In our results, we observed two to five times improvement in processing speed while maintaining the same level of accuracy and completeness.

The paper is organized as follows. Section 2 describes related work. Section 3 identifies factors affecting processing speed and proposes our methods. In Sect. 4, we apply our proposal to the classification system and prove its validity. Section 5 concludes the paper with suggestions for future research.

2. Related Work

Internet traffic classification techniques such as port number matching, payload signature matching, statistics, and application behavior-based methods have been the foundation for traffic classification engines for many years [1], [3], [8], [9]. However, these methods are not yet applicable to real

Manuscript received December 24, 2013.
Manuscript revised June 3, 2014.

[†]The authors are with the Dept. of Computer & Information Science, Korea Univ., Korea.
[††]The author is with the Dept. of Information System, Hanyang Univ., Korea.
a) E-mail: tmskim@korea.ac.kr (Corresponding author)

DOI: 10.1587/transinf.2013EDP7454

networks.

Port number matching is a simple but effective technique of identifying Internet traffic. The Internet Assigned Numbers Authority (IANA) maintains a list of assigned port numbers, but it does not guarantee sole usage by one application. Arbitrary port usage is possible for multiple applications or for bypassing firewalls. Payload signature is a unique string or binary pattern in the packet's payload and payload signature matching favors identification accuracy. Despite its wide use in commercial products, payload signature matching has problems that Baldi et al. summarize as follows. It (1) lacks robustness against packet loss, fragmentation, and segmentation, (2) cannot extract payload signatures, (3) is ineffective in encryption and tunneling, and (4) requires many computing resources. With respect to (4), the computation time is proportional to the traffic volume (e.g., session count) and the number of signatures [10]. Some alternatives have been investigated to overcome these issues, namely, statistic- [8] and behavior-based identification [9]. However, their applicability to real networks is still questionable because of the dynamic nature of Internet traffic.

In this paper, we examine the problem of computation time of a payload signature classifier. The processing speed of a payload signature classifier is insufficient for real-time management of high-speed networks. Snort is a popular payload-signature intrusion detection and application-level traffic classification system and has a processing speed of 100 Mbps on general-purpose processors [7]. Pattern-matching time is the chief concern because it consumes 40 to 70% of the total processing time [4], [5], [12]. For faster matching, several studies developed software-based solutions to take advantage of the spatial locality of a nondeterministic finite automaton (NFA) and the temporal locality of deterministic finite automaton (DFA) [11], [12]. However, the time complexity of the matching algorithms is entirely dependent on input signature configuration. In particular, multiple regular expression matching (e.g., ".*") requires intensive system resources and often results in a performance bottleneck. Its high storage requirements and computational cost to match all packets that traverse a link makes it impossible for online classification of traffic at high-speed (Gbps) links. Therefore, real-time traffic analysis on high-speed links might be insufficient. T. Liu et al. [5] and R. Kandhan et al. [13] proposed a fast pattern-matching method using a hierarchical signature structure, which assumes that most signatures include the meta-character "^" and a common substring. However, the signatures for identifying the application name do not satisfy this assumption. Park et al. [15] had suggested that the signature grouping method employ a wildcard and group an expression such as "*" or "|" to improve the processing speed of the classification system. However, this method dramatically increases the processing speed when applied to an NFA-partial algorithm. For example, five signatures can be combined in a single signature. This method, however, increases the matching speed by 20 times that of the five single-signature matching method.

Hardware-based systems such as ASIC and FPGA are faster than software-based systems, but they have insufficient scalability to support new applications. To update them, the chip must be redesigned, which usually requires high production cost. Mitra et al. [14], [17] implemented an NFA-based regular-expression engine on an SGI Altix 4700 workstation with FPGA support. The throughput of the NFA improved noticeably as a result of their study, whereas the compact memory requirement was maintained. However, these approaches require a specialized computer with high-processing power. Zhang et al. [6] proposed a multicore architecture for application traffic classification and their design improved the overall execution time. However, the lack of program parallelism in legacy network applications greatly limits the full use of multicore architecture.

3. Methodology

In this section, we propose methods to improve the processing speed of a payload signature classifier. In addition, we introduce the traffic trace and baseline classification system and conduct experiments to prove the validity of these solutions.

3.1 Data Description

We collected two traffic traces. The first comes from the Internet junction of the Korea University campus network, which has 3,000 active users. The second derives from a collection of 13,000–22,000 residential lines from a commercial ISP in Korea. Table 1 displays the details of the traffic trace, that is the full payload used to evaluate the performance of the proposed methods in the experiment.

Figure 1 is a diagram that shows the verification network for the location of traffic collection, configuration of the classification system, and verification of the classified

Table 1 Traffic trace.

Location	Date and duration	Flow	Packet
Campus	03/11/12 1 Day	48,477(K)	1,712,490(K)
ISP	06/09/09 16:30 1 h	3,803(K)	432,550(K)

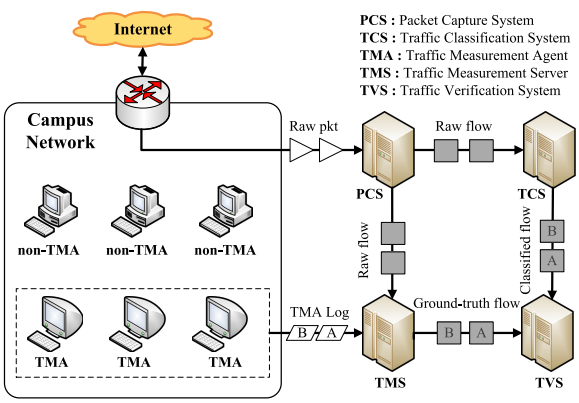

Fig. 1 Configuration of our baseline traffic classification system (TCS).

Table 2 Application breakdown.

Category	Applications	Campus	ISP
Web	HTTP (naver, daum, nate) HTTPS (google, dropbox)	31.55%	45.34%
P2P	torrent, donkey	26.32%	13.85%
Streaming	afreeca, alsong, melon	10.23%	8.45%
Internet Disk	clubbox, downs	12.96%	0.26%
Mail	smtp, pop3, imap	2.99%	4.23%
Management	dns, dhcp, ntp, sntp	2.35%	13.80%
etc	-	4.89%	5.33%
Unknown	-	5.46%	4.23%

traffic. All packets are captured from the link of the Internet border router by the packet capture system (PCS). The PCS reports the traffic flows to the TCS, which in turn determines the application name of the flow based on the payload signatures. A traffic measurement agent (TMA) is installed at the end host, which then collects the connection/process information of the host and periodically sends it to the traffic measurement server (TMS). The information collected by the TMAs includes socket connection information such as IP addresses, port numbers, and L4 protocols [15] from which we can obtain ground-truth information about the traffic flows. The ground-truth flows from the TMS are used to assess the accuracy of classification results from the TCS.

Table 2 indicates application usage in bytes using the baseline classification system [15], which can cover 245 applications comprised of 845 payload signatures and can achieve approximately 94% completeness on both traffic traces.

Bro [21] is widely used for application protocol-level analysis. We analyzed both traffic traces using Bro 2.1 to prove the effectiveness of the TMA-based verification system. Bro and our classification system employ similar methods regarding the classification units of flow and connection. The classification result of our system was considerably similar to that of Bro.

The campus and ISP traces contain mainly web service (naver, daum, nate) and P2P traffic generated by torrents and donkey. These web services are provided by Internet portal sites that are popular in Korea. The ISP HTTP traffic includes traffic generated by multimedia-file download applications. The campus network uses fixed IP addresses, whereas the ISP uses dynamic IP addresses. Therefore, the ISP trace includes some DHCP traffic for the assignment of dynamic IP addresses. Both the campus and ISP traces contain approximately 5% unknown traffic.

Table 3 displays the accuracy and completeness of the baseline classification system used in the campus network trace, which is evaluated by comparing the classification result of our baseline system and the ground-truth traffic data at the TVS, as shown in Fig. 1. The accuracy is defined by the rate of traffic that is correctly classified from the total, as shown in Eq. (1). Completeness refers to the results

Table 3 Performance of the baseline classification system.

Method \ Metric	Accuracy (%)			Completeness (%)		
	Flow	Pkt	Byte	Flow	Pkt	Byte
the baseline system	95.57	98.31	98.70	92.54	93.67	94.54

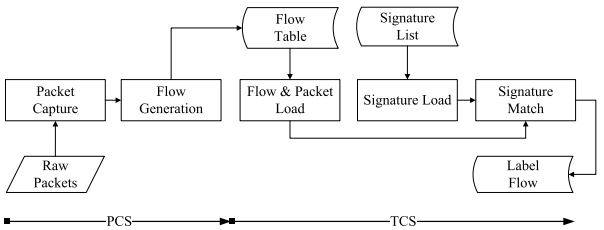

Fig. 2 Classification process of the baseline system.

classified by the classification system, which is expressed as a ratio of the total traffic, as shown in Eq. (2).

$$accuracy = \frac{Correctly\ Identified\ Traffic}{Identified\ Traffic} \quad (1)$$

$$completeness = \frac{Identified\ Traffic}{Total\ Traffic} \quad (2)$$

This guarantees greater than 95% accuracy and 92% completeness in terms of flow/packet/byte. The adequate performance of the baseline classification system allows us to evaluate the proposed methods with the goal of improving the processing speed of the baseline classification system while maintaining the same level of accuracy and completeness.

3.2 Baseline Classification System

In this section, we describe the baseline classification system used to evaluate the performance of the proposed methods. The overall classification process of the baseline system consists of two consecutive subsystems: the PCS and TCS.

The PCS collects all IP packets from a target network link and generates flow data. A flow is a set of packets containing the same 5-tuple packet header information (source IP, destination IP, source port, destination port, and L4 protocol) and their reverse packets. Flow data contains the minimal information necessary to reduce the amount of traffic data, such as the connection data; statistical data, which contains packet count, flow duration, byte count, etc.; and payload data of the first few packets. The TCS identifies the application name of the flows based on the payload signatures. We have developed the baseline system and deployed it in our campus network for real-time classification of campus Internet traffic. The system specifications of the TCS were Intel(R) Core(TM) i7 3.40 GHz CPU with 8 Gb RAM. The baseline system adopts several algorithms and methods that are commonly used in an application TCS. In this paper, we mainly focus on the modules that utilize the TCS.

Figure 3 shows the input-data memory structure and the inspection range of input data for the pattern-matching

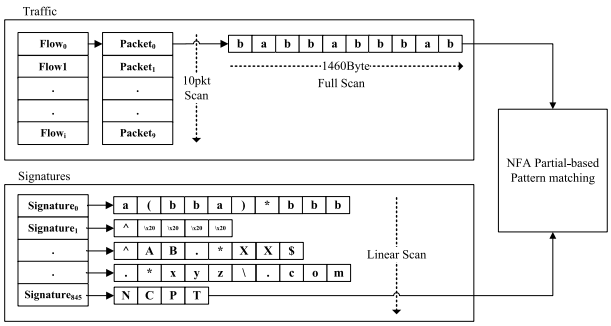

Fig. 3 Detail of the signature-matching module.

module of the baseline classification system. The baseline system performs matching for the first 10 packets in a flow and all bytes in a packet until a signature identifies a flow. The system then compares every signature in order to analyze each flow.

Regarding the matching unit, the matching approaches are divided into packet-based matching (PBM) and stream-based matching (SBM). PBM sequentially compares payload signatures and packets within a flow one by one, whereas SBM compares the same elements to a packet stream, which is the sum of two or more packet payloads. The SBM thus requires more packets than does the PBM. Moreover, the SBM requires additional overhead for recombining packets that incur packet loss and undergo asymmetrical routing [1]. Therefore, we use PBM rather than SBM in the baseline system.

We use an NFA-partial matching algorithm, which is commonly used in a TCS such as Snort [7], for the baseline system. In addition, an NFA-partial matching algorithm experimentally provides the highest average performance of a single-matching algorithm used with all types of signatures.

The baseline classification system inspects the first 10 packets in a flow. Park et al. [16] investigated the first 100 packets in each flow for four different applications. They discovered that the first few packets are used to send a signal before transmitting the content data, and most of the application signatures are located within the first few packets in a flow. We confirm that the first 10 packets are sufficient to ensure accuracy in the baseline system.

The classification taxonomy uses the application criterion and not the L7 protocol criterion. Many studies tend to confuse the concepts of L7 protocol and application classification. For instance, it is unclear how to classify HTTP traffic from a YouTube application for data transfer. The L7 protocol of the traffic is HTTP, whereas the application name is YouTube. If we do not use application criteria, we cannot determine the end-user application. By using application criteria, we can obtain more specific information about traffic and human behavior patterns, and classify the same traffic discovered by the L7 protocol.

3.3 Framework of the Proposed Method

Figure 4 shows the overall framework of the proposed and

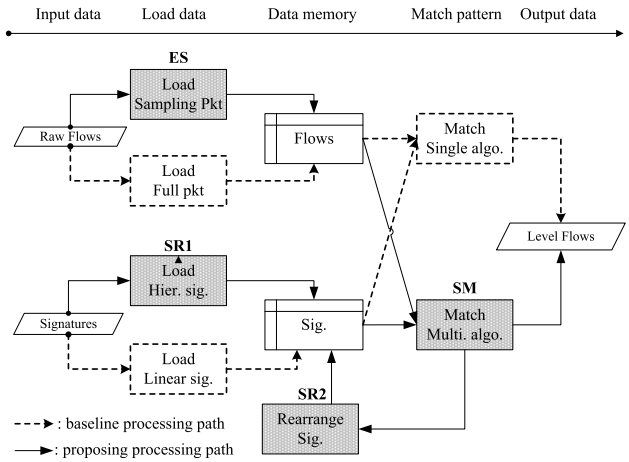

Fig. 4 Framework of the proposed and baseline methods.

Table 4 Composition of signature.

Signature Type		# of sig. (845)	Example	
Explicit String	Fixed offset	54	^\x13bittorrent protocol	(torrent)
	Variable offset	337	SS_SD_TEX	(afreeca)
Regular Expression	".*" <= 2	387	^GET.*Host: apsso\.inlive\.co\.kr.*	(alsong)
	".*" > 3	67	.*@.* @.*p$.*edonkey.*	(edonkey)

baseline method that represents the processing paths of the two methods regarding input and output data, the input-data loading module, input-data memory structure, and pattern matching.

We propose three main ideas: selective matching (SM), signature reorganization (SR), and early sampling (ES). SM uses different matching algorithms in accordance with the signature types. SR minimizes the signature search space by means of a two-level hierarchical signature structure (SR1) and by dynamically changing the signature-memory ordering, referred to as a signature cache (SR2). ES minimizes the number of packets in a flow and limits the byte size in each packet. These methods can be applied individually or in any sequence-free combination.

3.4 SM

Many algorithms exist for pattern matching, but no matching algorithm exists for all input types. The performance of the matching algorithm depends on how the signatures are represented [12]. We separate the signatures into four groups to evaluate the performance of the matching algorithm according to the type of signature representation as shown in Table 4. We initially divide the signature types into two categories: explicit string and regular expression. The explicit string is further divided into fixed- and variable-offset strings based on the signature location. The fixed-offset string defines the signature as a string or as hex values that appear on a specified offset of a packet payload. In most cases, the offset appears at the beginning of the

Table 5 Comparison of matching time based on signature type.

Signature Type \ Matching Algorithms		Rabin-Karp	DFA Full	NFA Partial	NFA Full
Explicit String	Fixed offset	0.03(sec)	0.05	0.08	0.08
	Variable offset	1.28	0.32	0.90	0.42
Regular Expression	".*" <= 2	3.45	0.19	0.08	0.16
	".*" > 3	1.35	0.06	0.05	0.55

payload (e.g., eDonkey "0xe3" at offset 0). The variable-offset string refers to the signature's appearance at any position. Regular expression is divided again into two types according to the frequency of the wildcard character ("*") in the signature. Table 4 shows that most signatures belong to either one of the two groups, that is, either to the variable offset of the explicit string or to the regular expression that uses wildcard characters of less than or equal to 2.

Many studies on payload signature classifiers use multiple wildcard meta-characters (e.g., ".", "*") in their signature representation [18]. Some patterns even contain more than 10 such wildcard fragments. As regular expressions are converted into state machines for pattern matching, many wildcards can cause the corresponding finite automaton to grow exponentially. In fact, this may frequently occur when the regular expression contains the Kleene closure operator, which means that an arbitrary number of characters may be present in the specified position in the data being analyzed. Therefore, we divide the signature types according to the frequency of appearance of the wildcard character "*" in a signature. We apply each signature type to the Rabin-Karp, DFA, and NFA matching algorithms. The Rabin-Karp string-searching algorithm is widely used for pattern matching [20], and it is easy to implement for hardware-based devices. The DFA and NFA are typically used to accept regular expressions.

We apply each signature type to the Rabin-Karp, DFA-full, NFA-partial, and NFA-full matching algorithms to determine the most reliable matching algorithm based on the type of signature representation. Table 5 shows the average processing time of the four matching algorithms for various signature types.

The Rabin-Karp string-matching algorithm reveals the most efficient performance for signatures of fixed offset. Without a priori knowledge of the starting and ending positions of the payload, the Rabin-Karp algorithm created for recognizing all substring matches can be extremely complex. The DFA-full algorithm is selected for an explicit string signature using a variable offset. Explicit strings generate DFA of length linear to the number of characters in the pattern. In the worst case, the time complexities of DFA and NFA are O(n) and $O(n^2)$, respectively. Finally, the NFA-partial matching algorithm is used for signatures in the form of a regular expression. The NFA-partial and NFA-full matching algorithms are different. For example, given a signature "^AUTH.*" and an input "AUTH abc," NFA-full matching can report four possible matches: "AUTH," "AUTH a," "AUTH ab," and "AUTH abc." In practice, it is unnecessary to report all matching substrings because most applications can be satisfied by a subset of those matches. The NFA-partial will report one match instead of the four and therefore, it is faster than the NFA-full matching algorithm. We implemented the Rabin-Karp algorithm and used the PCRE library [22] for the NFA algorithm, and the Boost library [23] for the DFA algorithms.

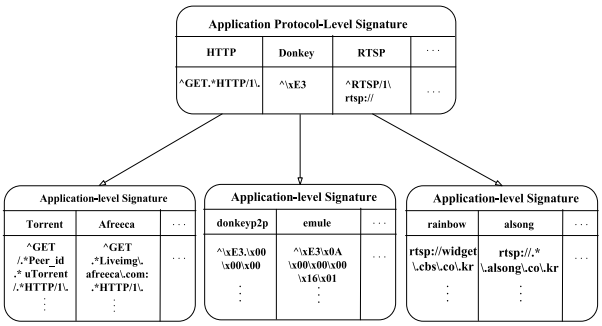

Fig. 5 Two-level hierarchical signature structure.

3.5 SR

The performance of a string matching algorithm depends on the search space of input data. This section describes how to minimize the search space of the payload signatures provided to the classification system. We propose two main ideas for reorganizing signatures: an SR1 and SR2.

3.5.1 SR1

In this section, we propose an SR1 to reduce the signature search space and to determine an application-protocol name as well as application name for each flow. The SR1 consists of application protocol-level signatures and application-level signatures at the first and second levels, respectively. An application protocol-level can be used by many applications for various purposes. In our signature hierarchy, the HTTP traffic is detected at the first level and the application name is determined at the second level. The signature hierarchy is defined by an inclusion relationship in which S_Y is an application protocol-level signature and S_X is an application-level signature. If all of the traffic identified using signature S_X can be classified using signature S_Y, then S_Y would include S_X.

We define the SR1 using the automata formula. Formally speaking, DFA is a quintuple $M = (Q, \Sigma, \delta, q0, F)$ where

1. Q is a finite set of states;
2. Σ is a finite set of input symbols;
3. δ: Q x $\Sigma \rightarrow$ Q is a transitional function that takes a state and an input symbol and returns a state;
4. q0 is an initial state that belongs to the Σ set; and
5. $F \subset Q$ is a set of final or accepting states.

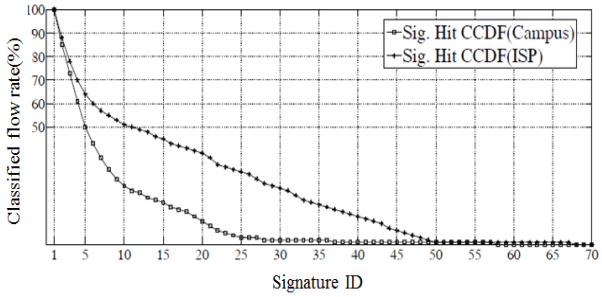

Fig. 6 Classified rate of traffic flows in CCDF.

For every signature, a corresponding finite automaton M exists that accepts the language L generated by the signature.

$$L(M) = \{x \mid \delta(q_0, x) \in F\} \tag{3}$$

If M_0 is an application protocol signature and M_1 is an application signature, they satisfy the inclusion relationship shown in Eq. (4).

$$L(M_0) \supseteq L(M_1) \tag{4}$$

In the SR1, the classification system first identifies the input flow by means of the application protocol-level signature. If a flow is classified by an application protocol-level signature containing application-level signatures, then the classification system can identify the flow by means of those signatures. This hierarchical analysis can reduce the signature search space of the classification system as well as the processing time. The baseline analysis method compares all of the signatures one by one to analyze a flow, while the proposed method can improve the processing speed of the classification system by reducing the search space.

3.5.2 SR2

The popularity of certain applications can be inconsistent because of the availability of well-known alternatives that offer similar services such as popular websites, e-mail, etc. [19]. These services motivate us to determine the signature-matching order in the classification system. Figure 6 displays a CCDF graph that represents the signature hit rate on the campus and ISP traffic traces. In our study, signature hits occurred in only 70 of the 845 signatures for each minute of traffic trace at a certain time of the day. In addition, 70% of the traffic flows of both campus and ISP traces were matched by only 35 or fewer signatures. The byte distribution of classified traffic according to the signatures is more uniform in the ISP than in the campus trace. Because the number of hosts running on the campus is limited relative to the ISP, only certain applications operate during a specified period. The majority of the traces at both the campus and ISP are web-browser traffic and the HTTP signature works well. The signature ID 1 is an HTTP signature for both campus and ISP traces.

Most traffic can be classified using a few signatures during a specific period. We can minimize the search space by first examining frequently occurring signatures and dynamically changing the ordering of signature memory according to the signature hit ratio. We call this mechanism signature caching. We use the following algorithm to rearrange signature-memory ordering.

Algorithm 1. Signature Memory Rearrangement

```
PS(L)   : Application Protocol-level Signature (List)
AS(L)   : Application-level Signature (List)
AHC     : Average Cache Hit Count
HC      : Cache Hit Count
α       : Constant Value
i : PS Index, j : AS Index, n : time index

for PS0 to PSi in PSL
    for AS0 to ASj in PSi
        AHCn= α×HCn+ (1-α)×AHCn-1
    end for
end for

for PS0 to PSi in PSL
    sortbyAHCn(ASL in PSi)
end for
```

Alg. 1 Signature memory rearrangement.

We use the exponential average (EA) value of the signature hit count to rearrange the position of signatures in the two-level signature structure. The EA value of each signature is calculated periodically with the hit count of the period and the previous value to reflect the recent signature hit count more than that of the past.

The system can deal with changes in application usage according to the time flow by updating the hit count constantly through the EA. The algorithm first calculates the exponential average of the signature hit count (HC_n) for the current signature, and then, it determines the average cache hit count (AHC_n) of the signature. The constant α is set to 0.6 to calculate the EA. Based on the AHC values of every signature, we rearrange the signatures by sorting them in the descending order. The coefficient α represents the degree of weight required to decrease the AHC_{n-1}. The coefficient α is such that the AHC_n is given more weight than the AHC_{n-1} and means that more recent HC are considered more important than older HC. This approach can achieve a reduction of more than 50% signature search space than that of the baseline classification system.

3.6 ES

Flow-based rather than packet-based analysis is commonly used in traffic analysis and is employed in this study to minimize both the number of packets in a flow and the byte size in the packet that is searched.

Figure 7 shows the distribution of the matched offset of signatures regarding packet sequence in a flow and byte position in a packet as determined through experimental evaluation on traffic traces at the campus and ISP. Most signatures were found in the first two packets of the flow and within the

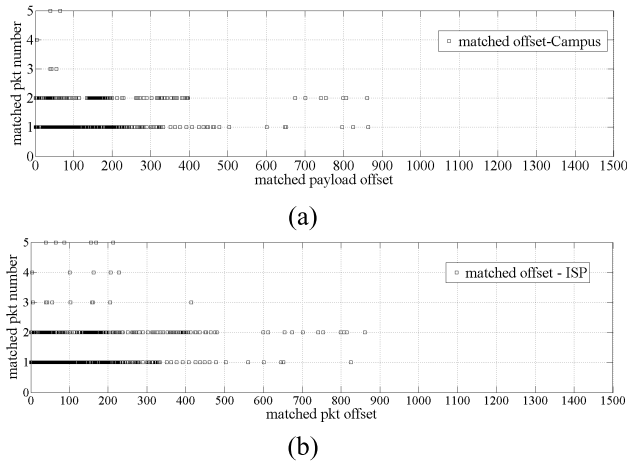

Fig. 7 Distribution of the matched offset of the signature in packets at the campus (a) and ISP (b).

first 500 bytes of those packets. Our baseline classification system performs matching for the first 10 packets in each flow and all bytes in a packet, thus dramatically degrading performance. We can utilize these experimental results to reduce the search space of the input data for the matching algorithm.

The packets inspected are defined as the first n packets having a payload after the TCP connection setup. According to analysis results, the classification accuracy and completeness increase as the number of packets inspected increases. However, packets that follow the fifth packet are almost identical. In other words, before sending the content packets, most connections transmit a few control packets that are common amongst all those having the same type of connection. Most of the payload signatures can be extracted from the first few packets in a flow. Therefore, the classification result can be sufficiently accurate and the classification time can be reduced by limiting the number of packets inspected to the first five packets in each flow.

In addition, the byte limit of a packet must be considered to reduce the search space of the input data for the classification system. We can reduce the processing time of our classification system by limiting the number of bytes inspected to the first 1,000 bytes in each packet payload. This allows our system to cope with greater bandwidth from a high-speed link.

4. Evaluation

We demonstrate a traffic classifier designed through the proposed methods to achieve the goal of throughput improvement. Then, we evaluate our proposed method by comparing it with the baseline classification system and Snort systems.

4.1 Performance Gain

Figure 8 displays the improvement in performance of the

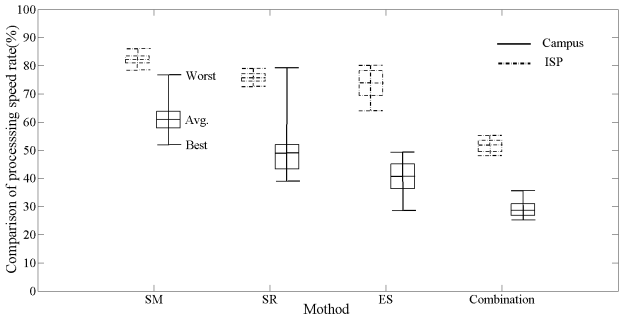

Fig. 8 Performance improvement of the methods when implemented individually and in combination as compared to the baseline system.

methods when employed individually and in combination. The figure uses box-plot graphs to show the improved rates of processing speed in the worst, average, and best cases, employing dotted and solid lines to represent the ISP and campus traces, respectively. The processing speed correlates to the consumption time necessary to perform major functions (e.g., activities related to load traffic and the pattern-matching module) at one-minute intervals.

When all three methods are combined, the processing speed improves by two times for the ISP and by five times for the campus traces. Fewer applications run on the campus network than on the ISP network, and therefore, the SR2 in the campus network outperforms that in the ISP network. It improves 1.25 times in the worst case when each method is independently applied to the ISP and campus network traces. The baseline classification system compares every signature in order to analyze a flow, whereas the suggested analysis method based on the hierarchical structure and signature caching can improve the processing speed of the classification system by reducing the search space. This method can achieve a reduction of more than 200 out of 845 signatures. Our baseline classification system performs matching for every packet in the flow and all bytes in a packet until a flow is identified by a signature, resulting in major performance degradation. However, the ES, which shows the best performance improvement, considerably reduces the traffic search space for pattern matching. As a result of our experiments, the ES can achieve a reduction of more than 60% payload as compared to that of the baseline classification system. The SM is faster than the NFA-partial method, yields the highest average performance for all types of signatures, and can manage an average of 10% more payload than can the NFA-partial method. The SR was obtained by checking signatures proactively to minimize the search space and is influenced by the number of applications and users on the target network. The number of applications increases as more users generate traffic and this, in turn, causes frequent cache exchange in the SR2.

Figure 9 shows the results of the payload signature-based analysis used to evaluate the performance of the proposed methods through experimental evaluation on our campus traffic trace. The graph shows the processing time spent in classifying a minute of traffic data during a single

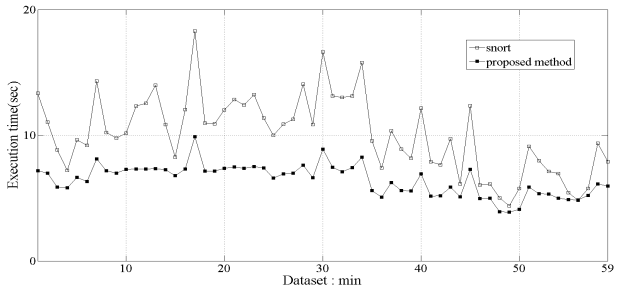

Fig. 9 Processing speed: Snort vs. the proposed method.

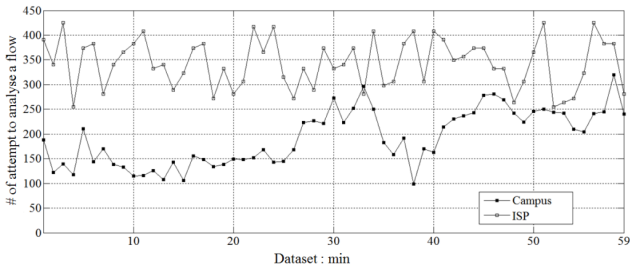

Fig. 10 Average number of matching attempts to analyze a flow.

hour. Approximately two times improvement in processing speeds can be observed compared to Snort.

The proposed method can improve the processing speed of the baseline classification system while maintaining the same level of accuracy and completeness. Our payload signature classifier achieves more than 95% accuracy and 92% completeness. Although we can reduce the search space of the classification system, the classification accuracy and completeness are the same as those of the original system.

4.2 Spatial Effect

Signatures used in this study are extracted from the campus trace and applied for the purpose of comparison with their performance on the ISP trace. The spatial effective study comprises two experiments: (1) determining the number of matching attempts to analyze a flow, and (2) analyzing the matched offset of signatures regarding byte position in packets.

Figure 10 compares the number of matching attempts for a flow in both the campus and ISP traffic traces. We draw the average matching attempts of the SM for one minute of traffic trace during a one-hour period. The number of matching attempts necessary to analyze a flow of ISP is more than that for the campus trace.

We can reduce the search space of a signature through the SR, which uses the exponential average value of the signature hit count to sort the SR2. Because the number of applications running on the campus network is fewer than that on the ISP network, only certain applications operate during a specified period in the campus network trace. Therefore, the signature sorting is more effective on the campus than on the ISP trace.

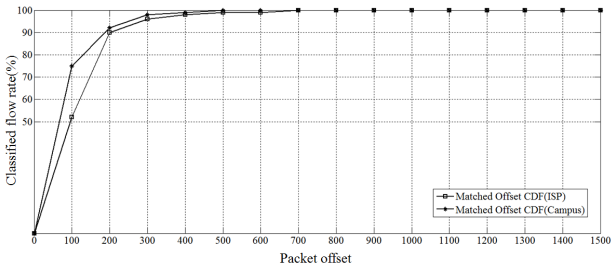

Fig. 11 Average matching offset of identified packets in CDF.

Figure 11 presents the CDF graphs that represent the matched offset of signatures regarding byte position in packets on campus and ISP traffic traces. More than 75% and 52% of the classified flows in campus and ISP traces, respectively, are matched within only 100 bytes. These flows occur mainly because of activities in P2P-file download applications such as torrent and donkey. The number of flows generated by torrent and donkey in the campus network are more than those in the ISP networks. Therefore, the performance of the ES on the campus network trace is better than on the ISP network.

5. Conclusion

Payload signature classifiers are widely used in network monitoring and analysis applications but have a major drawback in achieving real-time processing in a high-speed network field.

In this paper, we address the factors affecting the processing speed of the payload signature classifier. We experimentally evaluated each factor and proposed a method to create an efficient classification system. We proposed minimization methods for the search spaces of signatures and input traffic data. It is possible to design a high-speed Internet TCS according to the proposed methods. The suggested architecture improved processing speed by approximately two to five times compared to that achieved in both the baseline classification system and Snort, whereas maintaining the same level of accuracy and completeness.

This method provides a software-based means to improve the processing speed of general classification systems in a given computing environment. We also plan to design a multicore-based classification that will allow real-time analysis on a large-scale network.

Acknowledgements

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), Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2010-0020728) and Brain Korea 21 Plus (BK21+).

References

[1] F. Risso, M. Baldi, O. Morandi, A. Baldini, and P. Monclus, "Lightweight payload-based traffic classification An experimental evaluation," Proc. IEEE International Conference on Communications, pp.5869–5875, Beijing, China, May 2008.
[2] A. Chaudhary and A. Sardana, "Software based implementation methodologies for deep packet inspection," Proc. Information Science and Applications, pp.1–10, Jeju Korea, April 2011.
[3] A. Dainotti, A. Pescape, and K. Claffy, "Issues and future directions in traffic classification," IEEE Network, vol.26, no.1, pp.35–40, Jan. 2012.
[4] F. Fusco and L. Deri, "High speed network traffic analysis with commodity multi-core systems," Proc. SIGCOMM Internet Measurement, pp.218–224, Melbourne, Australia, Nov. 2010.
[5] T. Liu, Y. Sun, L. Guo, and B. Fang, "Improving matching performance of DPI traffic classifier," Proc. ACM Symposium on Applied Computing, pp.514–519, New York, USA, 2011.
[6] K. Zhang, J. Wang, B. Hua, and X. Tang, "Building high-performance application protocol parsers on multi-core architectures," Proc. IEEE International Conference on Parallel and Distributed Systems, pp.188–195, Tainan, Taiwan, Dec. 2011.
[7] S. Campbell and J. Lee, "Prototyping a 100G monitoring system," Proc. IEEE Parallel, Distributed and Network-Based Processing Euromicro, pp.293–297, Garching, Germany, Feb. 2012.
[8] A. Monemi, R. Zarei, and N. Marsono, "Online NetFPGA decision tree statistical traffic classifier," Comput. Commun., vol.36, no.12, pp.1329–1340, July 2013.
[9] K. Xu, M. Zhang, M. Ye, M. Chiu, and J. Wu, "Identify P2P traffic by inspecting data transfer behavior," Comput. Commun., vol.33, no.10, pp.1141–1150, June 2010.
[10] M. Baldi, A. Baldi, N. Cascarano, and F. Risso, "Service-based traffic classification: Principles and validation," Proc. IEEE Sarnoff Symposium, pp.1–6, Princeton, NJ, USA, March 2009.
[11] L. Christopher and Y. Luo, "DPICO: A high speed deep packet inspection engine using compact finite automata," Proc. IEEE Symposium on Architecture for Networking and Communications Systems, pp.195–203, Orlando, Florida, USA, Dec. 2007.
[12] R. Antonello, S. Fernandes, D. Sadok, and J. Kelner, "Characterizing signature sets for testing DPI systems," Proc. IEEE Management of Emerging Networks and Services Workshop – Globecom, pp.678–683, Houston, TX, USA, Dec. 2011.
[13] R. Kandhan, N. Teletia, and J.M. Patel, "SigMatch: fast and scalable multi-pattern matching," Proc. VLDM Endownment, vol.3, no.1, pp.1173–1184, Sept. 2010.
[14] A. Mitra, W. Najjar, and L. Bhuyan, "Compiling PCRE to FPGA for accelerating SNORT IDS," Proc. ACM/IEEE Symposium on Architecture for Networking and Communications Systems, pp.127–136, Florida, USA, Dec. 2007.
[15] J.S. Park, S.H. Yoon, and M.S. Kim, "Software architecture for a lightweight payload signature-based traffic classification system," Proc. Traffic Monitoring and Analysis Workshop, pp.136–149, Vienna, Austria, April 2011.
[16] B.C. Park, Y.J. Won, M.S. Kim, and J.W. Hong, "Towards automated application signature generation for traffic identification," Proc. Network Operations and Management Symposium, pp.160–167, Salvador, Bahia, Brazil, April 2008.
[17] P.C. Lin, Y.D. Lin, Y.c. Lai, and T.H. Lee, "Using string matching for deep packet inspection," Computer, vol.41, no.4, pp.23–28, April 2008.
[18] M. Becchi, C. Wiseman, and P. Crowley, "Evaluating regular expression matching engines on network and general purpose processors," Proc. ACM/IEEE Symposium on Architectures for Networking and Communications Systems, pp.30–39, Princeton, New Jersey, USA, Oct. 2009.
[19] Z. Zhou, T. Song, and F. Wenliang, "RocketTC: A high throughput traffic classification architecture," Proc. Computing Network and Communications, Maui, Hawaii, USA, pp.407–411, Feb. 2012.
[20] M. Zubair, F. Wahab, and M. Ikram, "Text scanning approach for exact string matching," Proc. Networking & Information Technology, Manila, Philippines, pp.118–122, June 2010.
[21] Bro, http://bro-ids.org/index.html, accessed Dec. 19. 2013.
[22] PCRE, http://www.pcre.org, accessed Dec. 19. 2013.
[23] Boost, http://www.boost.org, accessed Dec. 19. 2013.

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.

Sun-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 and network management.

Youngjoon Won is an assistant professor at Hanyang University, Seoul, Korea. He was a researcher at Internet Initiative Japan Inc., Tokyo, Japan. Prior to IIJ, he was a postdoctoral researcher at INRIA, France. He received his B.Math (2003) from the University of Waterloo and PhD (2010) from POSTECH.

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.