JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 32, 1241-1259 (2016)

Performance Improvement of Traffic Classification Based on Application Traffic Locality

JUN-SANG PARK[1], SUNG-HO YOON[1], SU-KANG LEE[1],
YOUNGJOON WON[2] AND MYUNG-SUP KIM[1,+]
[1]*Department of Computer and Information Science*
Korea University
Sejong, 30019 Korea
E-mail: {junsang_park; sungho_yoon; sukanglee; tmskim}@korea.ac.kr
[2]*Department of Information System*
Hanyang University
Seoul, 04763 Korea
E-mail: youngjoon@hanyang.ac.kr

Application-level traffic classification is an essential requirement for stable network operation and resource management. The payload signature-based classifier is considered a reliable method for Internet traffic classification. However, with this system, processing speeds are slower when high volumes of traffic are being classified in high-speed networks in real time. In this paper, we propose a method for server IP-port pair cache-based traffic classification, with the aim of increasing the processing speed and completeness of payload signature-based traffic classification. This approach takes application traffic locality into consideration. Moreover, we propose a cache data management method that has the purpose of minimizing the utilization of cache memory and processing speed and maximizing level of completeness. When our proposed method was applied to a campus network, we observe 10 times improvement in processing speed and 10% increasing in completeness against the payload signature-based classifier without a server IP-Port pair cache.

Keywords: network traffic monitoring and analysis, internet traffic classification, payload signature, processing speed, application-level traffic

1. INTRODUCTION

Traffic classification is an essential preliminary step in achieving efficient network resource management and the provision of a stable network service. While a number of classification methods have been proposed in the literature in recent years, the payload signature-based classification method has been shown to exhibit the highest levels of performance in terms of accuracy, completeness, and practicality [1-4]. However, the processing speed of this system is difficult to meet the requirement for real-time handling of the large volume of traffic data passing through high-speed networks [5-8]. Taking into consideration the increasing number of Internet-based applications being utilized, as well as the expanding use of applications that generate high volumes of traffic, the inadequate processing speeds of payload-based analysis systems is an important issue that needs to be addressed.

Most of the previous studies have aimed at improving the performance of payload

Received June 28, 2015; revised November 17, 2015; accepted January 8, 2016.
Communicated by Meng Chang Chen.
[+] Corresponding author: tmskim@korea.ac.kr.

signature-based classifiers by making use of the pattern-matching algorithm [9-11]. However, there is a limit to the improvement in performance that can be provided by this algorithm, and it is practically impossible for this method to achieve real-time processing of the large volumes of traffic in high-speed networks.

In this paper, we propose a server IP-port pair cache-based traffic classification method aimed at increasing the processing speed of the payload signature-based traffic classification method that is currently being utilized. We furthermore propose an efficient system for managing cache data that is adaptively based on the popularity bias and the temporal locality of application traffic. The results of our experiments show that the processing speed was more than 10 times faster than that of the payload signature-based classifier without a cache and the level of completeness was improved by more than 10% when the proposed method was applied.

The applications that are used in a given network at any time may vary. However, we found that most of the network traffic is generated by only a small number of applications by understanding from our experiments on our campus network. Therefore, the number of server IP-port that clients in a network are connected to is limited, with only a few of these addresses receiving heavy traffic. When we classified the traffic in a campus network in our study, we found that the average ratio of the TCP flow to the server end-point was approximately 20 during the entire observation period. In other words, averagely five clients were connected to one server end-point, which is an example of popularity bias. The users in a network typically utilize the Internet for similar purposes. As a result, if a certain client is connected to a particular server, it is highly likely that other client will be connected to the same server in the near future. This phenomenon is referred to as temporal locality.

Several classification methods that make use of server IP port pair information have been proposed to aid in increasing the processing speed of the payload signature-based method. Baldi *et al.* [12] described an approach whereby the payload signature is used in a service that classifies the flows of a server's IP-port, and then performs classification based on the service table. However, with this method, as the cache search space becomes larger, the hit rates decreases, because it is not possible to replace and delete entries in the service table when it is enlarged indefinitely. Furthermore, this approach is limited because it is possible that the IP-port of the server will dynamically change, such as p2p application. Park *et al.* [13, 14] proposed a time-out based cache deleting method in order to avoid explosion of the cache memory. However, the time-out based approach can only apply in unlimited cache memory environment and keeps the unused server IP-port address during the certain period time in the cache. In addition, they did not mention how to set the time-out threshold value. Therefore, a more advanced cache management method is required for efficient use of limited cache memory. In this paper, we propose a more advanced cache deletion and replacement method, in a limited and an unlimited cache memory environment, respectively.

This paper is organized as follows: In Section 2 we discuss the motivations for this study, namely, the popularity bias and temporal locality of application traffic. An overview of the methodology applied in the study is provided in Section. In Section 4, the validity of the proposed method is evaluated by means of applying it in a classification system. Finally, we provide our conclusions and outline future work in Section 5.

2. PROPERTIES OF TRAFFIC LOCALITY

In this section, we define the popularity bias and temporal locality of application traffic that the motivation for our proposed method. In the previous study, we have only defined the popularity bias for the server IP-Port cache-based classification [13, 14]. Since it only reflects the popularity of applications, we additionally define the temporal locality to reflect that of time dimension. And we provide experimental verification of our method based on the actual traffic of a campus network.

2.1 Data Description

We now describe the types of traffic that were generated, as well as how the traffic that we used was collected for the purpose of verifying the validity of our proposed method.

Table 1 provides the details of the traffic trace making up the full payload that was used to analyse the performance of the proposed method in the experiment. The trace was obtained from the Internet junction of our campus network, which has 3,000 active users. All packets were captured from the link of the Internet border router. After analysis of long-term traffic, we conformed that our campus network has a time-series periodicity based on a day. So, we choose only one-day traffic to analyse traffic characteristic and to verify out method.

Table 1. Traffic trace.

Location	Date and duration	Flow	Packet
Campus	12/09/12 (1 Day)	51,477(K)	2,012(M)

Table 2. Application breakdown.

Top 10	App. Type	Flows	Packets	Bytes
1	P2P file-sharing	25,395K	561,709K	356GB
2	Web browser	17,043K	1,037,368K	864GB
3	IM	1,086K	3,0381K	16GB
4	Utility	973K	4,1905K	32GB
5	Multimedia	759K	20,5534K	172GB
6	Game	691K	31,088K	18GB
7	SNS	449K	9820K	6GB
8	Security	248K	6,283K	4GB
9	Vaccine	230K	28,970K	28GB
10	Commercial	126K	22,358K	20GB

Table 2 shows the 10 highest traffic volumes according to application type, based on the payload signature-based traffic classification system that we developed in previous study [1]. Since we used our previously developed system to classify all of the Internet traffic in the campus network, we found that the accuracy for flows was above 99%, and the level of completeness was greater than 85% for bytes and packets [1].

The results that are provided in Table 2 are similar to those obtained from the Bro [15] system, which is a system that is widely used for application-level traffic analysis. It

can be seen that the P2P file-sharing applications occupied the highest number of flows, at more than 50%, followed by web browser traffic. These results reveal that most of the traffic passing through the network was generated by only a small number of applications. If the assessment time these applications can be reduced, the processing speed of the entire classification system can be increased significantly.

2.2 Popularity Bias

In this section, we discuss the distribution of the server IP-port pair among the collected traffic and define the popularity bias for the application traffic in the study.

The number of server IP-port in the campus network was limited. Fig. 1 illustrates the hourly TCP flows as well as the quantities of server IP-Ports, in order to determine their distribution. We refer to a group of flows that share the same server IP, port, and L4 protocol as an SSIP (Same Server IP Port). An average of 29.59% of the flows qualified as SSIPs. The number of SSIPs in the network remained similar during the peak hours and the off-peak hours in terms of the number of users and the volumes of traffic. This is due to the fact that the application traffic in the pertinent network was limited to a certain number of SSIPs, which was independent of the number of users and the traffic volumes. We now describe how the traffic that we used was collected, and which application traffic was generated, in order to verify the validity of our proposed method.

In the study, only a certain number of SSIPs received flows. In order to determine the frequency of the appearances of SSIPs in the application traffic, Fig. 2 shows a CDF

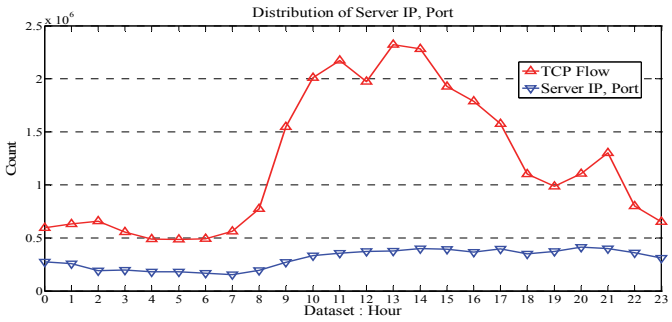

Fig. 1. SSIP vs. flow.

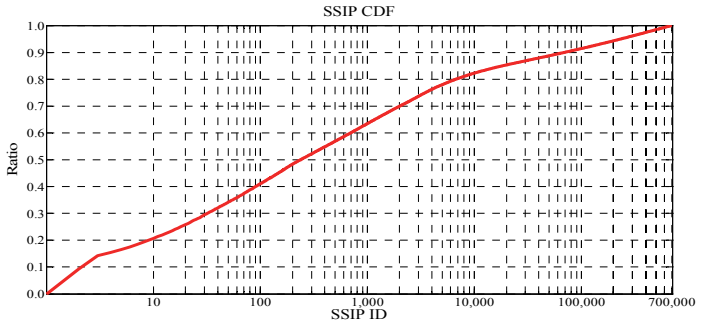

Fig. 2. SSIP CDF.

graph of the TCP flows that were occupied by the SSIPs, as illustrated in Fig. 1. The X-axis makes use of a log scale to represent the number of SSIPs.

A total of 80% of the TCP flows were accessed by making use of fewer than 10,000 SSIPs, which means that only a small number of SSIPs received most of the applications traffic. This is an example of popularity bias. Server IP-port pairs are the addresses that are used by clients to connect to certain application services. If it is possible for such applications to be classified based on fewer than 10,000 SSIPs, and this information remains in the classification system, the flows that are accessed through one of the these server IPs/Ports may be determined without payload signature classification being performed.

2.3 Temporal Locality

This section explains the concept of the temporal locality of application traffic. Temporal locality implies that, if flows are accessed through an SSIP for a certain application in a network, it is highly probable that this SSIP will receive another flow in the near future.

Fig. 3 shows a staircase graph that displays the temporal localities of the 10 most utilized SSIPs based on time. SSIP ID 2 and SSIP ID 6 appeared continually throughout the entire assessment, while the remainder of the SSIPs appeared continually at specific times. Based on these characteristics, we can conclude that, if the SSIPs of the flows of recognized applications remain in the classification system, it is possible to classify other flows that make use of the same SSIPs without performing payload signature classification.

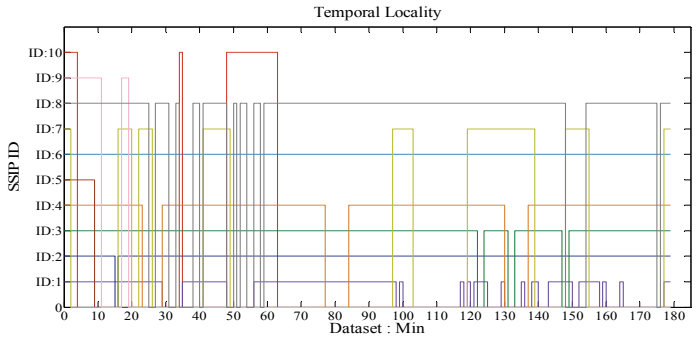

Fig. 3. Temporal locality.

In this paper, we propose an adaptive cache-based traffic classifier that takes the popularity bias and the temporal locality of application traffic in a network. This method is capable of increasing the processing speed of the payload signature-based classifier.

3. ADAPTIVE SSIP CACHE-BASED CLASSIFICATION METHOD

In this section, we describe the processes implemented in the proposed SSIP cache-based method for the classification and management of the adaptive cache.

3.1 SSIP Cache-based Classification

The proposed method is capable of increasing the processing speed and completeness of the traffic classification system. The SSIPs of the flows that are classified using the payload signature are stored in the cache, and are thereafter classified in order for the load for pattern matching to be reduced and the processing speed to be improved. Although the traffic classification system fails to maintain the latest payload signature due to the frequent renewals of applications that occur, if any flows are connected through the same SSIP that is stored in the cache, it is possible for these to also be classified, which increases the level of completeness.

Fig. 4 is a conceptual diagram outlining the proposed traffic classification method. In the diagram, clients A and B each create two connections through the same IP-port pair of an application server. If flow #1 that is initialized by Client A is classified based on the payload signature, flow #2 from this client, as well as flows #3 and #4 from Client B, can be classified based on the SSIP cache, without a payload signature-based classification being performed. Furthermore, it is possible for the flows to be classified by means of the SSIP cache even if the classification system does not possess the signatures for flows #2 and #4 as a result of changes in the applications that are being used.

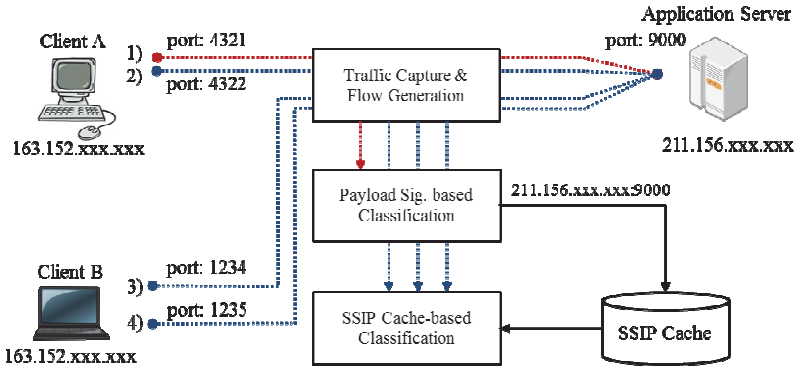

Fig. 4. Conceptual diagram of proposed method.

Fig. 5 shows a flow chart summarizing the proposed method. The classification system ultimately generates flows that are labelled with the corresponding application's name after receiving the payload signature and unlabelled flows as input. The classification system is made up of two modules: the signature-matching module and the SSIP cache management module.

The purpose of the signature-matching module is to match the flows with the SSIP cache, and then to classify the flows based on payload signatures. The module first matches flows with the SSIP cache. If a flow is classified successfully, the module then extracts the SSIP from the flow and sends it to the cache manager. Any flows that are not classified by the SSIP cache are matched by means of the payload signature-based classification method. Thereafter, the results are sent to the cache manager. The purpose of the SSIP cache management module is to insert and renew classification results in the cache, while replacing and deleting existing cache data for effective memory management.

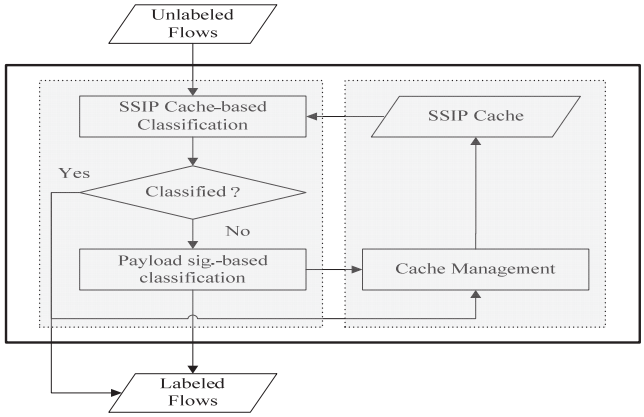

Fig. 5. Flow chart of the proposed method.

3.2 Cache Management

In this section, we describe the cache data management process used in our proposed method, which minimizes usage of the limited memory and guarantees a maximum level of completeness.

The replacement and deletion of cache data are processes that are critical to determining the processing speed and completeness of a classification system. If it were possible for traffic classifications to be stored in the cache permanently, the completeness of the cache would continually increase. However, in actual applications, the volume of available memory is limited, and the processing speed of the classification system decreases when the volume of cache data is greatly increased due to the permanent storage of data. Furthermore, when a small cache is used, only a small number of flow classifications may be stored, which also hinders the improvement of processing speed.

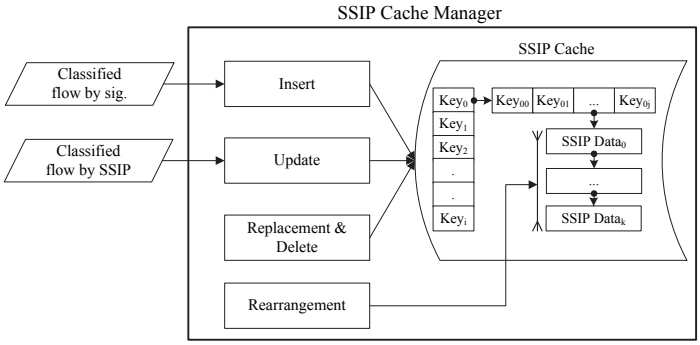

Fig. 6. Composition of cache management.

A diagram of the functions of the cache manager as well as the structure of the memory is provided in Fig. 6. The cache manager is used to insert, update, replace, delete, and rearrange the data in the cache. The cache data is stored by means of a double hashing structure, in order to increase search speeds.

The purpose of the insert module is to use the payload signature to extract the SSIPs from the classified flows, and store them in the cache. The update module updates the hit counts for the SSIPs that are associated with the flows classified according to the SSIP cache that have been stored in the cache. These hit counts then become a criterion for determining whether or not cache data should be deleted.

The replace and delete module serves to replace or delete cache data. The SSIP cache is used to improve the processing time and completeness. If cache data are permanently stored in the cache and repeatedly classified, the search volume increases, which results in processing delays. Furthermore, if the server host information for P2P applications remains in the cache, the accuracy of classification is decreased. As a result, the replacement and deletion of cache data is required.

The rearrangement module makes use of hit counts to arrange the entries that possess the same hashing keys. Frequent occurrences of high-volume traffic may decrease the processing speed even though the double-hashing memory structure reduces the volume of data. Cache data that have the same key values are rearranged according to the frequency of their use during classifications, and are matched in order of the most frequently used data. This process may contribute to increasing the speed of searches.

The method that should be applied for cache data management is determined according to whether the amount of memory is limited or unlimited. In this study, we propose a cache management method that takes both memory conditions into consideration: the replacement module is applied when the memory is limited, while the delete module is used in the case of unlimited memory.

3.2.1 SSIP cache replacement

Cache replacement is applied in environments where there is limited memory. When the cache memory is full and the insertion of a new SSIP that has been classified based on the payload signature is requested, an existing SSIP entry in the cache will be replaced with this new entry. Formula 1 is applied to determine which data can be deleted.

The parameter represents an entry that is stored in the SSIP cache, while $C(t)$ denotes the set of all entries that are stored in the cache during a period of time t. $F_{e,t}$ represents the hit count for the pertinent entry that is expected for t, while $H_{e,t}$ is the hit count that is actually measured for t. Initially, $F_{e,t}$ is set to 0. The α parameter is a constant that ranges from 0 to 1. If α is set to a larger value, the previously estimated hit count ($F_{e,t}$) is given a greater weighted value, and the next hit count is then estimated. If α is set to a smaller value, however, the current hit count that is measured is given a greater weighted value, and the next hit count is then estimated.

$$\forall e \in C(t), F_{e,t+1} = \alpha F_{e,t} + (1-\alpha)H_{e,t}, 0 \le \alpha \le 1 \tag{1}$$

Formula 1 reflects the concepts of popularity bias and temporal locality simultaneously. The SSIPs with high cache hit rates are available as a result of the popularity bias. Such SSIPs need to remain in the cache since they increase the cache hit rates, while being used frequently for only a certain period of time due to the temporal locality. As a result of this, there is a need for some SSIPs to be deleted when they have not been used recently. If the current hit count is given a greater weighted value based on the α value,

the SSIPs that were being used frequently a substantial amount of time ago are deleted, while those that are being used frequently at the current time remain in the cache.

A section of pseudo code for cache data management in a limited memory environment is provided in Algorithm 1. Insertions, deletions, and updates are performed at the time of flow classification, and the value of $F_{e,t+1}$ is calculated once the analysis of t has been completed.

Algorithm 1: Cache management with limited cache memory.

```
1:  e: Entry in cache      α: Constant value
2:  F_{e,t}: Forecast cache hit count at t
3:  H_{e,t}: Hit count at t
4:  // replace an old e with a new one
5:  for each SSIP of classified flow do
6:      if cache is full
7:          searchEntryMinExpectHitCount()
8:          deleteEntryMinExpectHitCount()
9:          insertEntry(H_{e,t} = 1)
10:     else
11:         if e exists in cache
12:             updateEntry(H_{e,t}++)
13:         else
14:             insertEntry(H_{e,t} = 1)
15: done
16: // update F value of all e in cache
17: for each e in cache at t do
18:     F_{e,t+1} = αF_{e,t} + (1−α)H_{e,t}
19: done
```

3.2.2 SSIP cache deletion

The cache replacement takes place when the cache memory is full, whereas the cache deletion is applied in cases where unlimited memory exists such as time-out based deletion method [13]. Typically a time-out-based deletion method is used to delete data if it the data has remained in cache memory for longer than the threshold time regardless of memory usage. However, this approach keeps the unused server IP-Port address during the certain period of time in the cache. Therefore, we propose a more advanced cache deletion method for efficient use of limited memory.

The cache deletion is performed when $F_{e,t+1}$ decreases to below the threshold that is independent of the cache memory.

When a flow is classified, deletions and updates occur within the cache. When a flow is classified for t, $F_{e,t+1}$ is calculated and entries that fall below the threshold are deleted.

Algorithm 2: Cache management with unlimited cache memory.

```
1:  e: Entry in cache    α: Constant value
2:  F_{e,t}: Forecast cache hit count at t
3:  H_{e,t}: Hit count at t
4:  TH_F: Threshold for forecast cache hit count
5:  // insert a new e into cache
6:  for each SSIP of classified flow do
7:      if e in cache
8:          updateEntry(H_{e,t}++)
9:      else
10:         insertEntry(H_{e,t} = 1)
11: done
12: for each e in cache at t do
13:     F_{e,t+1} = αF_{e,t} + (1−α)H_{e,t}
14: Done
15: // delete an old e in cache
16: for each e in cache at t do
17:     if F_{e,t+1} less than TH_F
18:         deleteEntry(e)
19: done
```

3.2.3 SSIP cache rearrangement

Classification systems that are applied to high-volume traffic operate in real time and store large volumes of data in the SSIP cache. There is therefore a need for a memory structure and a matching mechanism that are capable of managing the cache memory effectively.

In this study, we propose a cache management method that makes use of a double-hashing memory structure, The entries in the cache are rearranged by probability of occurrence, which are computed from the history of successful matches.

Single hashing increases the number of nodes possessing the same keys when searches are being performed using a high volume of data. As a result, single hashing can lead to unwanted loads in traffic classification processes. Enlarging the size of the hash table is a potential solution, but is ineffective because the memory spaces are limited in classification systems. Double hashing minimizes the number of nodes possessing the same keys and reduces the size of the search space by means of a two-phased hashing key generation method. Based on the traffic volumes in the campus network used in our study, we set the size of the hash table to 2^3*2^{16}. For the first-phase hash keys, we used only three bits, which were formed from the sum of the last eight bits of the server IP, port, and L4 protocol. The second-phase hash keys were formed by adding two bytes to the beginning and end of the server IP. Although a double-hashing memory structure reduces the volume of data, frequent crashes may still decrease the processing speed. The cache data with the same keys are rearranged for classification based on the frequency of their use, and are matched in order of the most frequently used data. These processes are capable of enhancing search speeds.

4. EVALUATION

For the evaluation of our study, we used the actual traffic obtained from a campus network to verify the method that was proposed in Section 3, and to assess the processing speed and completeness of the proposed SSIP cache-based traffic classifier.

In the evaluation, formula 1 was applied to estimate a weighted hit count. A reasonable α value was required because α has a significant effect on the estimated hit count. Formula 2 determines the MSE (Mean Square Error) of an estimator in a time series analysis.

$$MSE = \sum (H_{e,t} - F_{e,t})^2 / T \tag{2}$$

The MSE of our proposed system was estimated hit counts were closest to the actual hit counts when α was set at 0.3. Table 3 shows the MSE values for the estimated hit counts as α increases from 0.1 to 0.9. When the value of α was small, the current hit count was assigned a greater weighted value in order for the MSE to be reduced. When α was larger, the estimated hit count was more affected by the past usage frequencies, which hindered the estimation of the current values. Since the usage frequencies of the SSIPs were greatly varied, it was appropriate to assign higher weighted values to the current values.

Table 3. MSE value based on α.

α	0.1	0.3	0.5	0.7	0.9
MSE	2,879	2,741	2,901	3,757	10,099

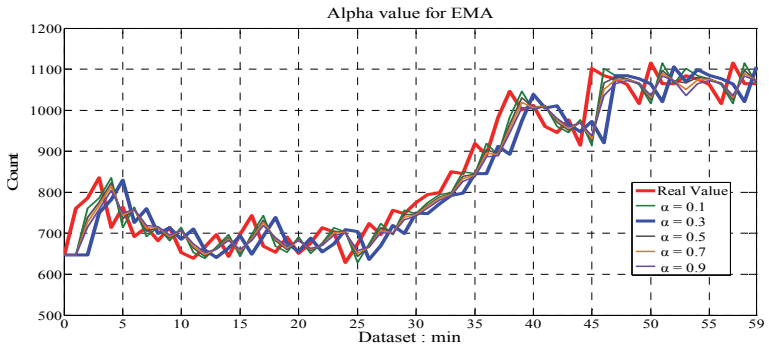

Fig. 7. Comparison of the estimated and the actual hit counts according to α.

Fig. 7 illustrates the estimated hit counts and the actual hit counts as the value of α was changed. It can be seen that the estimated hit counts were closest in value to the actual hit counts when α was equal to 0.3.

4.1 Limited Cache Memory

In this section, we assess the performances of the LFU and LRU methods that are used to determine the optimal size of the cache in a limited memory environment. These

methods are generally applied to cache replacement algorithms. We furthermore assess the performance of our method.

In this study, we assessed the completeness levels achieved when using the LRU, LFU, and proposed methods, in order to determine the optimal size of the cache, while changing the size of the cache based on all of the SSIPs (709,200) that were generated during a particular day. Fig. 8 illustrates the changes in levels of completeness according to the different sizes of the cache. The three algorithms exhibited a similar level of completeness when the SSIP cache was at 60% of its maximum size.

In order to assess the performances of the three algorithms, we calculated the number of insertions that were made into the cache, and found that the proposed method required the lowest number of insert operations.

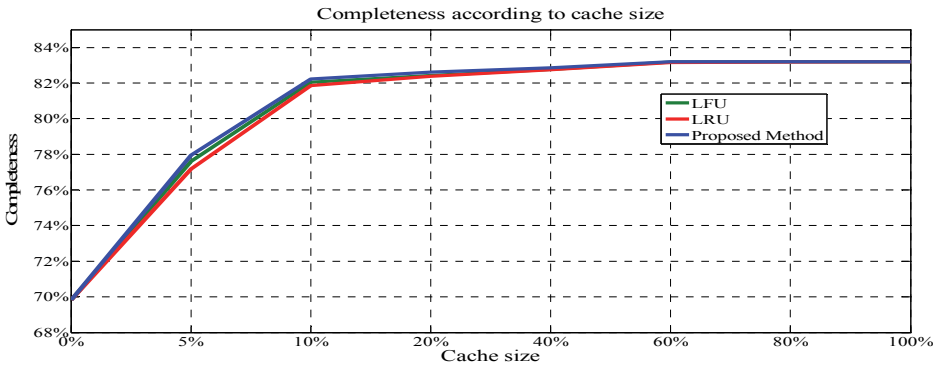

Fig. 8. Comparison of completeness according to cache size.

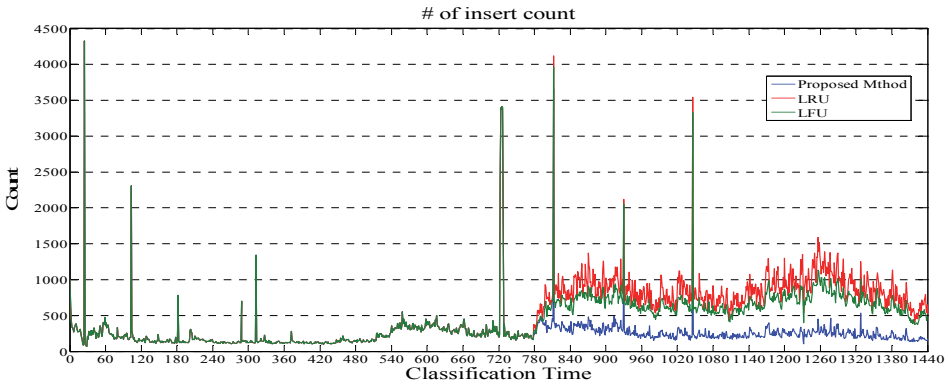

Fig. 9. Comparison of insertion operation counts.

Fig. 9 shows the counts of the insert operations for the LRU, LFU, and proposed methods when the cache size was at 60% of its maximum size. All three algorithms exhibited very similar counts for insert operations until the cache reached capacity, but thereafter, the proposed method was found to have the lowest counts. This means that the entries that were likely to be requested from the cache continued to be stored in it. In

addition, we conduct a random replacement strategy which deletes random entry when the number of entry reaches maximum cache capacity. However, the result of random strategy showed non-steady performance result in more than 10 repetitive tests but the average performance of the random strategy was much lower compared with other three methods. So, we ignore the random strategy in this paper.

4.2 Unlimited Cache Memory

In this section, we compare the performance of the cache deletion method [13, 14] to the time-out based deletion method in an unlimited cache memory environment. The time-out-based method is used to delete data if it the data has remained in cache memory for longer than the threshold time. The performance of the time-out-based method varies based on the threshold applied; therefore, it is important that the correct threshold be determined. In order to determine the time-out threshold, we measured the flow inter-arrival time for each SSIP, as shown in the CDF graph provided. In our study, the majority of the SSIPs received flows within two hours. Therefore, the time-out threshold was set to two hours.

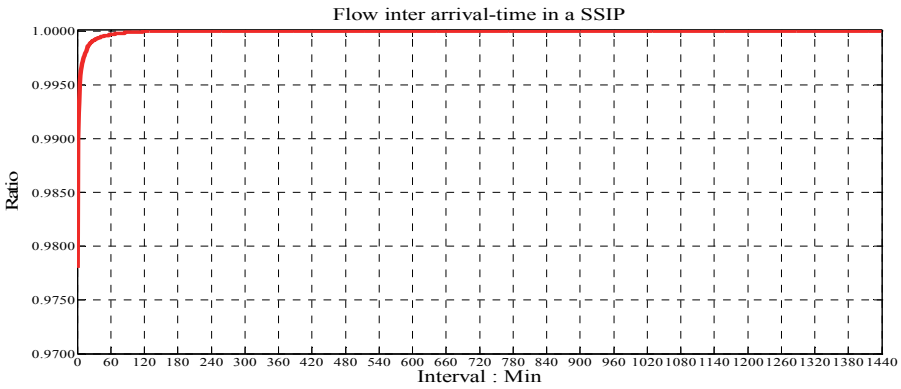

Fig. 10. Flow inter-arrival time in an SSIP.

The method applied in the unlimited memory environment makes use of a threshold value for the deletion of cache data. When the threshold value for the proposed method was set to 10^{-5}, the completeness was shown to be similar to that of the time-out-based method.

Table 4 shows a comparison of the levels of completeness of the proposed method when different thresholds are used, and the time-out-based method. In the time-out-based method, the unused data from the past two hours is also stored in the cache. However, the proposed method makes use of weighted values to guarantee a similar completeness within a shorter time.

When the threshold value was set to 10^{-5} and the cache sizes of the time-out-based method and our method were compared, it was found that the cache size of the our method was 3%-10% smaller than that of the other method, as shown in Figure 11. The reason for this result is that the proposed method was capable of deleting unused data at a faster rate than the other method.

Table 2. Completeness according to threshold.

Method	Threshold	Completeness
Proposed	10^{-1}	75.21%
	10^{-2}	79.63%
	10^{-3}	80.01%
	10^{-4}	82.21%
	10^{-5}	83.53%
Time-out	2 hours	83.54%

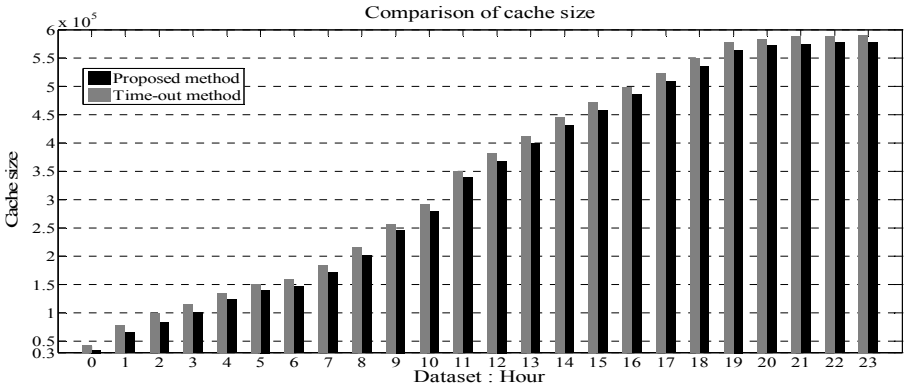

Fig. 11. Evaluation of cache size.

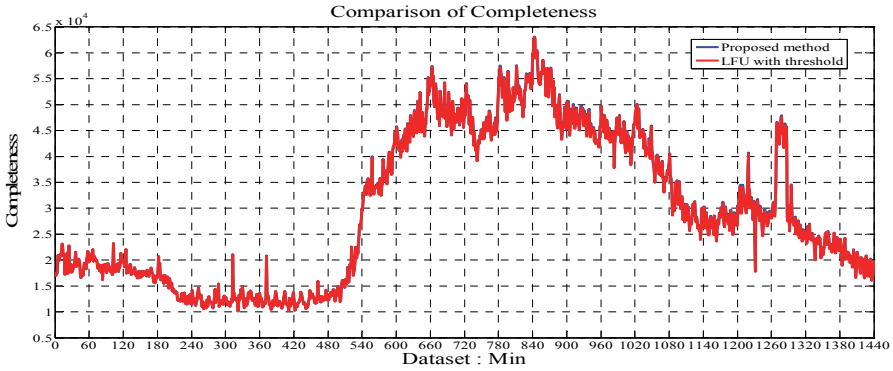

Fig. 12. LFU with threshold vs. proposed method.

Fig. 12 shows the comparison between the levels of completeness of the proposed method and the method that makes use of unlimited cache memory in conjunction with the threshold from the LFU, which exhibited solid performance in limited cache memory settings. As can be seen from the figure, the level of completeness of the proposed method was slightly higher.

4.3 SSIP Cache Rearrangement

In this section, we discuss the results of a performance assessment of a method where

whereby cache entries are rearranged using the same hash keys and frequently used SSIPs are prioritized during the matching process.

When the cache data were rearranged, the number of cache matching attempts made was found to be 10% lower than the number of attempts that were made in the random storage method. Fig. 13 provides a comparison of the matching attempts of the proposed method, in which the cache entries were rearranged using an estimated hit count, and a method in which the cache entries were stored randomly. When the number of entries in the cache was low, the effect of the rearrangements was minor, because fewer hash key conflicts occurred. However, when the number of entries exceeded a certain threshold, the rearrangements resulted in a 10% reduction in the number of matching attempts when compared to the random matching attempts.

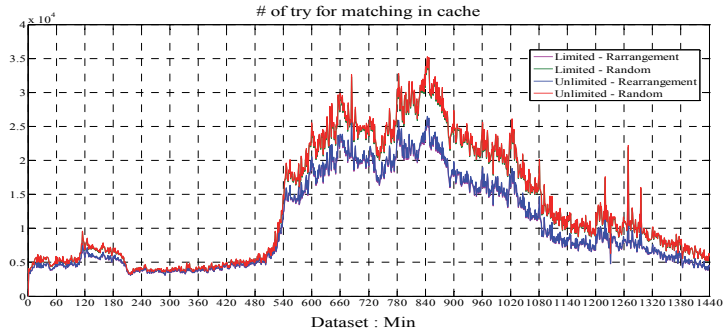

Fig. 13. Number of attempt for matching in cache.

4.4 Performance of SSIP Cache-based Classification

This section provides an assessment of completeness and processing speeds of the proposed method and that of a payload signature-based classifier without a cache. The processing speed of the proposed method was averagely 5 times faster than that of the payload signature-based classifier without a cache. Fig. 14 shows the classification times when using the SSIP cache-based classification method, with the hours for the payload signature-based classifier used as the baseline.

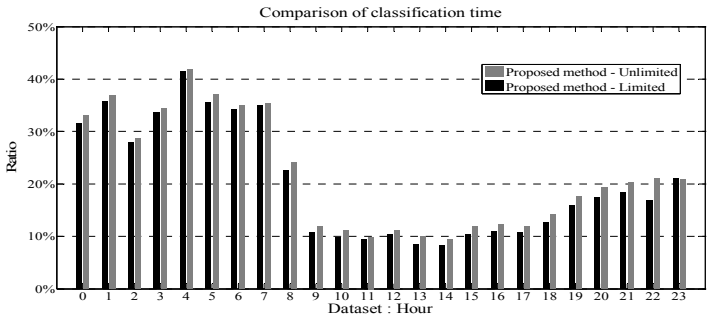

Fig. 14. Comparison of classification times.

The processing time of the payload signature-based classifier without a cache increased drastically between 09:00 and 21:00 when there was a peak in traffic. In contrast, the processing time of the proposed method did not exhibit a significant increase during those hours, because the cache-based classification method was capable of processing higher volumes of traffic.

In comparison to the payload signature-based classifier without a cache, the proposed method classified an additional 10% of the traffic according to flows. Fig. 15 shows the traffic rates for the payload signature classifier and the SSIP cache classifier. Area A represents the percentage of traffic that was classified by means of the payload signature-based classifier without a cache, while Area B represents the percentage of traffic that was classified using the SSIP cache-based classifier.

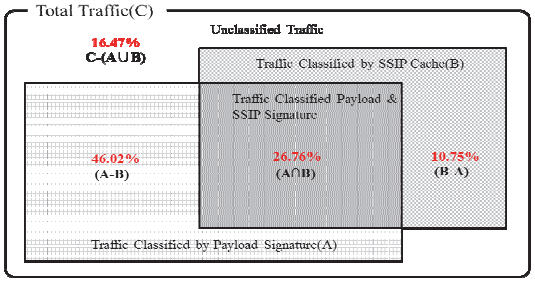

Fig. 15. Level of completeness based on each classifier.

Area A∩B represents the percentage of traffic that could be used by the proposed classification system to increase the processing speed. Although payload signature matching could also be applied in this area, the SSIP cache was capable of increasing the processing speed by using matching that was based only on server IP-Port pair. That is, 26.76% of the flows in the university network could be classified by using only the server IP-Port pair.

Area B-A shows the percentage of traffic that was classified by means of the proposed classification method but failed to be classified by the payload signature-based classification method. When a single server IP-Port pair extracted a portion of the payload signature, as opposed to the entire payload signature, for an application with multiple functions, the SSIP cache classified the remaining traffic flows. This result makes it clear that the level of completeness was increased by applying the proposed method. In addition, the proposed method was shown to be capable of using information regarding the server IP-Port pairs to classify the flows from which payload signatures were not extracted due to protocol obfuscation or encryption.

5. CONCLUSION AND FUTURE WORKS

In this paper, we proposed an SSIP cache-based classification method for increasing the processing speed of the payload signature-based traffic classification system. We furthermore proposed a cache management method that is appropriate for environments

with either limited or unlimited memory caches and maximizes the performance of the SSIP cache classifier. Our approach takes both the popularity bias and the temporal locality of application traffic into consideration. In our study, we compared our proposed classification method to the payload signature-based classification method without cache, and found that the processing speed was more than 10 times faster when our method was applied. Furthermore, the level of completeness was increased by 10% as a result of the classification of the traffic flows that could not be classified by the payload signature-based classifier without a cache.

We proposed a cache management method that applies the same weighted values to hit counts for the server IP-port caches for all of the applications. We plan to study a cache management method that considers the traffic patterns generated by different types of applications and applies different cache replacement policies.

ACKNOWLEDGEMENTS

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2015R1D1A3A01018057) and Institute for Information and Communications Technology Promotion (IITP) grant funded by Korea government (MSIP) (No. B0101-15-0300).

REFERENCES

1. J. S. Park, S. H. Yoon, and M. S. Kim, "Software architecture for a lightweight payload signature-based traffic classification system," in *Proceedings of International Conference on Traffic Monitoring and Analysis Workshop*, 2011, pp. 136-149.
2. A. Dainotti, A. Pescape, and K. Claffy, "Issues and future directions in traffic classification," *IEEE Network: The Magazine of Global Internetworking*, Vol. 26, 2012, pp. 35-40.
3. R. Antonello, S. Fernandes, D. Sadok, and J. Kelner, "Characterizing signature sets for testing DPI systems," in *Proceedings of International Conference on IEEE Globecom Management of Emerging Networks and Services Workshop*, 2011, pp. 678-683.
4. G. Aceto, A. Dainotti, W. de Donato, and A. Pescape, "PortLoad: taking the best of two worlds in traffic classification," in *Proceedings of International Conference on IEEE Infocom Computer Communications Workshops*, 2010, pp. 1-5.
5. N. F. Huang, G. Y. Jai, H. C. Chao, Y. J. Tzang, and H. Y. Chang, "Application traffic classification at the early stage by characterizing application rounds," *Information Sciences*, Vol. 232, 2013, pp. 130-142.
6. T. Ban, S. Guo, M. Eto, D. Inoue, and K. Nakao, "Towards cost-effective P2P traffic classification in cloud environment," *IEICE Transactions on Information and Systems*, Vol. E95-D, 2012, pp. 2888-2897.
7. J. M. Khalife, A. Hajjar, and J. Díaz-Verdejo, "Performance of OpenDPI in identifying sampled network traffic," *Journal of Networks*, Vol. 8, 2013, pp. 71-81.

8. G. Xie, M. Iliofotou, R. Keralapura, M. Faloutsos, and A. Nucci, "SubFlow: towards practical flow-level traffic classification," in *Proceedings of International Conference on IEEE INFOCOM*, 2012, pp. 2541-2545.
9. R. Antonello, S. Fernandes, C. Kamienski, D. Sadok, J. Kelner, I. Gódor, and T. Westholm, "Deep packet inspection tools and techniques in commodity platforms: Challenges and trends," *Journal of Network and Computer Applications*, Vol. 35, 2012, pp. 1863-1878.
10. T. Liu, Y. Sun, L. Guo, and B. Fang, "Improving matching performance of DPI traffic classifier," in *Proceedings of International Conference on ACM Symposium on Applied Computing*, 2011, pp. 514-519.
11. Z. Zhou, T. Song, and W. Fu, "RocketTC: a high throughput traffic classification architecture," in *Proceedings of International Conference on Computing, Network and Communications*, 2012, pp. 407-411.
12. M. Baldi, A. Baldini, N. Cascarano, and F. Risso, "Service-based traffic classification: Principles and validation," in *Proceedings of International Conference on IEEE Sarnoff Symposium*, 2009, pp. 1-6.
13. 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 the Asia-Pacific Network Operations and Management Symposium*, 2013, pp.1-6.
14. F. He, F. Xiang, Y. Xue, and J. Li, "Towards high-performance network application identification with aggregate-flow cache," *International Journal of Computer Networks and Communications*, http://arxiv.org/abs/1105.5684, 2014.
15. Bro, http://bro-ids.org/index.html, 2014.

Jun-Sang Park received his B.S. degree in Computer Science from Korea University, Korea, in 2008, his M.S. degree in Computer Science from Korea University, Korea, in 2010, and his Ph.D. degree in Computer Science from Korea University, Korea, in 2014. He is currently a researcher of Vehicle Components Company, LG Electronics, Korea. His research interests include Internet traffic classification and network management.

Sung-Ho Yoon received his B.S. degree in Computer Science from Korea University, Korea, in 2009, his M.S. degree in Computer Science from Korea University, Korea, in 2011, and his Ph.D. degree in Computer Science from Korea University, Korea, in 2015. He is currently a researcher of Vehicle Components Company, LG Electronics, 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.

Young-Joon Won is an Assistant Professor at Hanyang University, Seoul, Korea. He was a researcher at Internet Initiative Javan 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 Ph.D. (2010) from POSTECH.

Myung-Sup Kim received his B.S., M.S., and Ph.D. degrees 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.