World Academy of Science, Engineering and Technology
International Journal of Information and Communication Engineering
Vol:3, No:12, 2009

Network Application Identification Based on Communication Characteristics of Application Messages

Yuji Waizumi, Yuya Tsukabe, Hiroshi Tsunoda†, and Yoshiaki Nemoto
Graduate School of Information Sciences Tohoku University
Sendai–shi, Miyagi, 980–8579 Japan
† Tohoku Institute of Technologies
Email: wai@ecei.tohoku.ac.jp,† tsuno@m.ieice.org

Abstract—A person-to-person information sharing is easily realized by P2P networks in which servers are not essential. Leakage of information, which are caused by malicious accesses for P2P networks, has become a new social issues. To prevent information leakage, it is necessary to detect and block traffics of P2P software. Since some P2P softwares can spoof port numbers, it is difficult to detect the traffics sent from P2P softwares by using port numbers. It is more difficult to devise effective countermeasures for detecting the software because their protocol are not public.

In this paper, a discriminating method of network applications based on communication characteristics of application messages without port numbers is proposed. The proposed method is based on an assumption that there can be some rules about time intervals to transmit messages in application layer and the number of necessary packets to send one message. By extracting the rule from network traffic, the proposed method can discriminate applications without port numbers.

Keywords—Network Application Identification, Message Transition Pattern

I. Introduction

A person-to-person information sharing is easily realized by P2P networks in which servers are not essential. Leakage of information, which are caused by malicious accesses for P2P networks, has become a new social issues.

Information leakage incidents are caused by misoperation of network applications or wrong set up of security parameters in many cases. Recently, the incidents caused by inappropriate usage of peer-to-peer(P2P) software are increasing. Damage of information leakage caused by P2P software can be more serious because users and administrators may not realize the leakage. The traffic of P2P software should be blocked to prevent the information leakage. But it is difficult to discriminate and block them by using port number because most P2P software can alter its port number to use communication. Because protocol of P2P system are unreleased and cipher communication are used, any effective countermeasures to prohibit P2P communication have not been established yet.

To tackle the information leakage problem, some studies [1], [2], [3], [4] have been conducted to discriminate network applications using transition pattern of packet size and packet type, and using statistical information of connection requests. These studies are practical in terms of the privacy communication because these methods observe the headers of packets only. However, to identify the application, these methods need a lot of packets, and their identification accuracy is not enough.

The remainder of this paper is organized as follows. In Section II, related works in network application identification technologies is described. In Section III, two kinds of vectors extracted from network packets are proposed. In Section IV, identification results with an identification system using the two kinds of vectors are demonstrated. Finally, our conclusions is in Section V.

II. Related Works

As a traffic classification technique not using port numbers, a payload based approach has been proposed [5], [6], [7]. This approach examines payload of each packet to determine whether the payload contains specific characteristic signatures. This approach works well to identify network applications including P2P file sharing applications. However, this method hardly works when payload is encrypted. In addition, the payload examination can cause privacy problems.

Traffic identification techniques which are based on the behavior of hosts are proposed [8], [9], [10]. These techniques focus on behavior of each host, such as how many hosts are connected and how many port numbers are used. Although they can detect specific running applications, they cannot identify flows which are transmitted by the applications. Consequently, these techniques cannot block flows transmitted from applications in violation of network management policies.

Other methods which use flow statistics are proposed [11], [12], [13], [1], [14]. These methods are based on statistical information of a flow such as the number of packets, total data size transported in a flow, mean packet size and mean packet arrival time interval for instance. Although these method can identify each application flow without using port numbers or payloads, they need a lot of packets to obtain significant information for the identification. It is difficult for these methods to identify applications in the early stage of their flow and to block these flows before they are finished.

To identify applications at the early stage of their connections, an approach using first few packets of each flow has been proposed [15], [2], [16], [17] These methods are based on an assumption that some typical interactions such as version of application, authentication messages, would be sent in the first few packets of each flow. A method which can evaluate the typical interactions using the character code histograms of the first few packets of a flow has been proposed [17]. This method can identify applications before flows are finished. However, since this method uses payload information of packets, it can hardly identify applications if payloads are encrypted. On the other hand, a method proposed in [15] is based on payload length and direction of the first five packets of a flow. This method does not use the payload information but focus on payload lengths of the typical interactions at the early stage of flows. However if there are two applications whose characteristics of the first five packets are similar, the methods can not identify the applications by the first five packets. Thus these methods need more packets to identify applications but the number of packets for accurate identification is not clear. And, considering many packets causes "the contents communication problem". The problem is the decline of identification accuracy because the payload lengths of contents communication which is the communication of application data begins after the typical interactions are tend to change around MTU. This is a severe problem.

Some identification methods have been proposed too[17], [18], [19]. These proposed methods adopt transition pattern of the size of packets and contents of TCP payloads. Although the proposed methods can achieve over 90% identification accuracy, malicious user can evade identification by padding data to packets in order to change the packets size. Thus identification methods using new observables are necessary in order to reduce the risk that malicious users evade identification by tampering with packet size and header. In this paper, a new network application identification method using time intervals to transmit messages in an application layer and the number of necessary packets to send one message are proposed.

III. Extraction of Message Communication Characteristics

To build a network application identification system, it is assumed that each application has its specific pattern among time intervals to transmit massages and the number of necessary packets for each message in application layer. Our proposed method in this paper extracts the pattern from a *flow* which is the aggregated packets in a TCP connection. The extract methods will be described in following sections.

A. Time Interval for Transmitting Application Messages

In this paper, the time interval for transmitting application messages is defined as the differential time from a messages has been received until its response message has been transmitted. This transmission interval can be regarded as a processing time to respond the received message and to make its response message. This time interval may represent characteristics of an application because the processing time to respond a received message, i.e. a request message, varies from an application to another. For instance, HTTP servers needs retrieval and transmit process of html file after a "GET" message of HTTP. This processing time can represent any characteristics of HTTP application. On the other hand, SMTP server sends "250 + host name" after receive a "HELO" message from another Message Transfer Agent (MTA). This process does not need file retrieval process. Thus, a processing time to respond a request message can differ from an application to another because different process can be done for each application. By evaluating this differences of processing time, it can be consider that network applications can be identified.

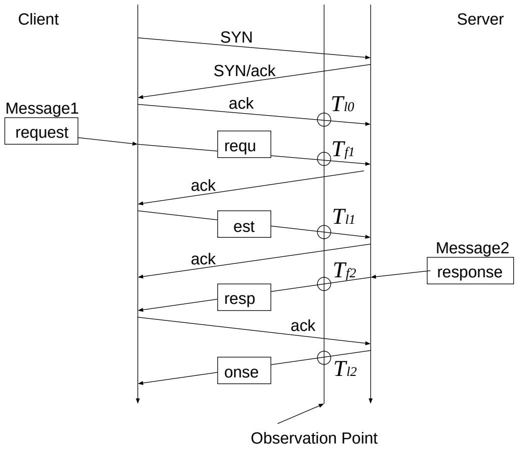

Fig. 1. Transmission Interval Extracting Method

Packets whose payload sizes of transport layer are larger than zero of a *flow* are used to extract the time interval because packets with zero size transport payload have no application layer information. The time interval can be calculated as time difference between the received time of the last packets of an application message and the sent time of the first packet of the response message by a host. Fig. 1 shows an example of the time interval calculation. T_{l0} in the figure is the finish time of 3 way-handshake. T_{f1} is the observed time of the first packet of the first message from the client and T_{l1} denotes the the observed time of the last packet of the first message. The time interval ΔT_i is defined as $\Delta T_i = T_{fi} - T_{l(i-1)}$. The time interval of Message1 is ΔT_1 is $T_{l0} - T_{f1}$ and it is considered as processing time to generate the first message after finished 3 way-handshake in the client. The processing time for the second message as response for the first message transmitted from the server is $\Delta T_2 = T_{f2} - T_{l2}$.

Multiple time intervals can be extracted from a *flow* because an application communicates multiple messages between a client and a server. These multiple time intervals from a *flow* is represented as a vector of which elements are ΔT in calculated order, it will be called *interval vector*. The vector is defined as follows:

$$\vec{v_i} \quad = \quad (\Delta T_1, \Delta T_2, \cdots, \Delta T_n)^T \qquad (1)$$

where i is the identifier of a flow and n denotes the number of the considered messages of the $flow_i$, and n is a parameter which shows how many messages are utilizes to identify network application.

Fig. 2 shows *interval vector* of Secure Shell (SSH). The horizontal axis is element number of *interval vector*, and the

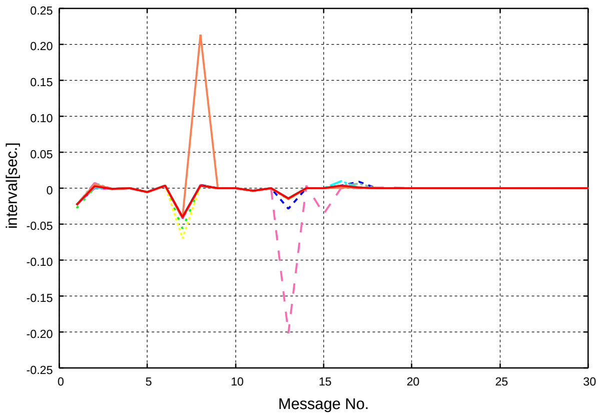

Fig. 2. Time Intervals of SSH

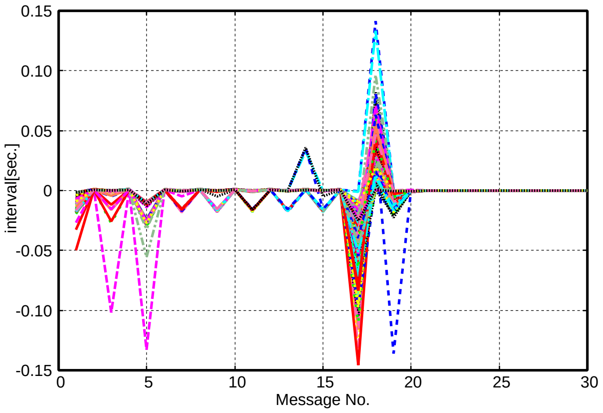

Fig. 3. Time Intervals of FTP

vertical axis shows the time interval. Here the client of a TCP connection is defined as a host which send a connection request, i.e. SYN packet, and denote the response time of the client by positive value. On the other hand, the response time of the server, which received SYN packet from the client, are represented by negative value.

There are 21 flows in Fig. 2. From this figure, it can be seen that the most of flows are overlapped with few exceptions, and that the response time of server at $n = 1, n = 7$ tends to be larger. Fig. 3 shows *interval vector* of File Transfer Protocol (FTP). The number of *interval vector* in this figure is 339. Although variance of each elements of the vector is large, the 5th element of all *interval vector* tends to be greater, and this increase of 5th element does not exist in SSH. From these results, it can be considered that *interval vector* can represent the characteristics of each application, and it can be used *interval vector* in order to identify network applications.

B. Number of Packets for Transmitting One Message

The size of the message of each application can be differ from the other applications.In this section, the number of necessary packets for transmitting one message of an application will be used to evaluate the difference of the message size. The sequence of the number of packets for transmitting multiple messages is represented as a vector, it is referred to as *number vector*.

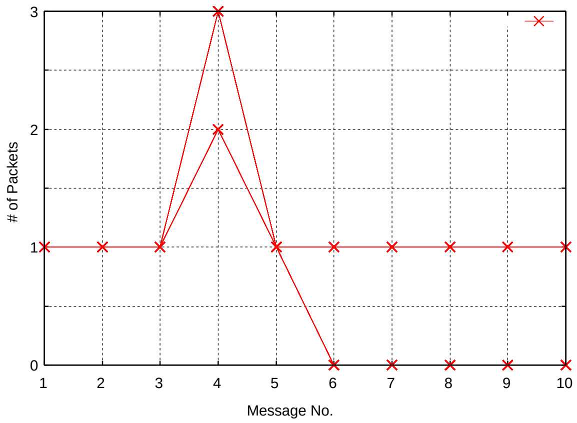

Fig. 4. # of packets per one message of POPS

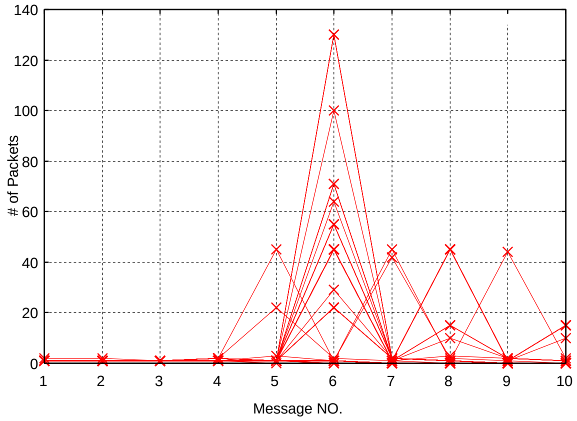

Fig. 5. # of packets per one message of Winny

Fig. 4 shows the *number vector* of Post Office Protocol over SSL (POPS). The *number vectors* in Fig. 4 have 10 elements, that is, 10 messages of application layer are used to make a *number vector*. The horizontal axis is element number of *number vector*, and the vertical axis shows the number of packets to transmit one message of POPS. The elements of the flows in which the number of messages is less than 10 are denoted as zero. From fig. 4, it can be confirmed that the *number vectors* have three patterns although there are 70 flows in the figure. That is, the *number vectors* of POPS are very similar each other.

Fig. 5 depicts *number vectors* of Winny, which is a P2P file sharing software and causes the most serious information leakage incidents in Japan. From fig. 5, it can be seen that the number of packets per message begins to increase at 5th or 6th message, the number of packets of 6th message increase significantly. This increment of packets does not exist that of POPS and consequently POPS and Winny can be discriminated by using *number vector*.

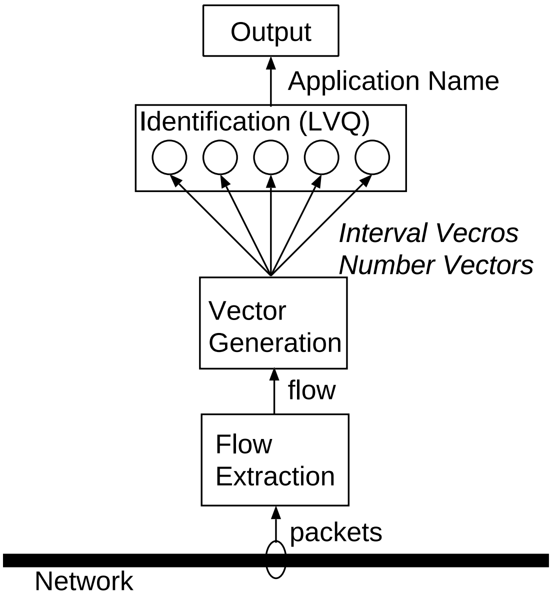

Fig. 6. Identification System

IV. Network Application Identification Method Using *Interval Vector* and *Number Vector*

From the analysis in the previous section, it is found that *interval vector* and *number vector* derived from different applications belong to different clusters. In this section,an application identification method using Learning Vector Quantization(LVQ)[20] is proposed.

A. Identification System

Our proposed identification system is shown in Fig. 6. The system is composed of flow extraction module, vector generation module and identification module. The flow extraction module extracts packets exchanged between client and server, and reconstructs flows from the extracted packets. The vector generation module makes *interval vectors* and *number vertors* from these flows. Optimized LVQ1 is applied to the identification module because Optimized LVQ1 can achieve higher classification accuracy among variations of LVQ. LVQ is a subtype of artificial neural networks, which adopts supervised learning. LVQ trains codebook vectors, which represent each class, using labeled sample flows which are identified their applications in advance. When application identifying phase, LVQ compares distances between an input vector which is extracted from a newly observed flow and codebook vectors, determines that the flow is transmitted by the application of the codebook vector nearest to the flow.

B. Data Set

Flows of following eight kinds of applications are captured: http, smtp, ssh, pop3, imap, pop3 over ssl (pops), http over ssl(https) and rtsp from a LAN, which is consisted of about 50 hosts. Flows of Winny and Share, which are P2P applications cause serious information leakage incidents in Japan, are also used. These two applications flow are captured from experimental network which is consisted of 10 hosts. 300 flows for each application are used to learn LVQ, and 300 flows for each application as test data except rtsp because less than 600 rtsp flows are collected. The number of rtsp flows of test sample is 100.

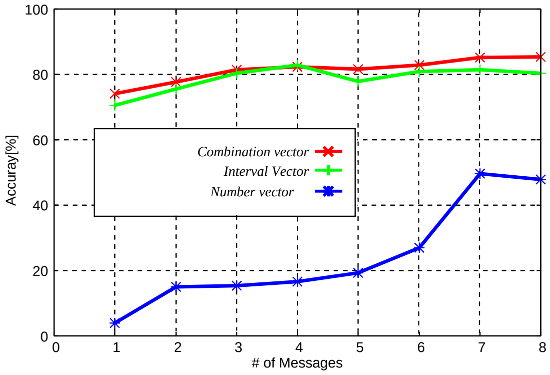

Fig. 7. Identification Accuracy for All Test Flows

In previous section, two types of feature vector are proposed, *interval vector* and *number vector*. In the identification experiment, *combination vector* of them are also used. The elements of *interval vector* and *number vector* are alternately placed in *combination vector*.

C. Indentification Results

1) Identification Accuracy for All Flows: Fig. 7 shows identification accuracy with *interval vector*, *number vecotr* and *combination vector*. The horizontal axis is the number of considered messages and the vertical axis is identification accuracy. For the *combination vector*, two elements are needed for one message because the *combination vector* consists of time interval and the number of packets for one message of an applicaiotn. Thus the length of the *combination vector* increases by two as one message are used for identification.

The identification accuracy for all flows $Accu_{all}$ is defined as follows:

$$Accu_{all} \quad = \quad \frac{\#\ of\ correctly\ identified\ flows}{\#\ of\ all\ test\ flows}. \qquad (2)$$

From fig. 7, it can be seen that the highest identification accuracy 85.12% can be obtained by using *combination vector* when the number of messages is seven.

2) Identification Accuracy of each Application: In this section, the identification accuracy of the proposed method for each application is analysed by comparing the identification accuracy by using the inverse number of packet size [18], [19]. The identification method using the inverse number of packet size identify network applications based on vectors consisting of the inverse number of packet size. The packet size inverse method used the same flow data in previous section and the same learning algorithm OLVQ1.

Table I shows the identification accuracy. The accuracy of the inverse number method is estimated with the number of packets by which can achieve the highest accuracy shown in

TABLE I
IDENTIFICATION ACCURACY OF EACH APPLICATION (# OF CONSIDERED MESSAGES IS 7)

	interval vector	*number vector*	*combination vector*	the inverse method
ftp	100.00	0.00	100.00	100.00
http	80.00	62.35	74.51	76.00
imap	18.67	1.33	21.00	97.00
pop3	70.00	0.00	99.33	99.33
pops	99.33	0.00	99.33	100.00
share	93.00	99.33	97.33	98.00
smtp	81.00	92.33	80.67	95.67
ssh	98.33	100.00	99.67	94.67
Winny	91.79	89.37	88.89	100.00
rtsp	92.00	100.00	100.00	92.91
Total	81.40	49.70	85.12	95.40

TABLE II
THE HIGHEST IDENTIFICATION ACCURACY FOR EACH APPLICATION

Application	Highest Accuracy	# of Considered Messages
ftp	100.00	5,6,7
http	88.00	3
imap	65.67	1
pop3	99.33	7
pops	100.00	2,3
share	99.00	2
smtp	81.67	5
ssh	99.67	7
Winny	96.00	6
rtsp	100.00	5,6,7

[18], [19]. The number of messages of the proposed method is seven which can obtain the highest total identification accuracy estimated by 2. Here the identification accuracy of each application is calculated by 3.

$$Accu_A = \frac{N_{TP}}{N_{all}} \times 100 \tag{3}$$

$$N_{TP} \; : \; \#\ of\ flows\ identified\ as\ application\ A$$

$$N_{all} \; : \; \#\ of\ all\ test\ flows\ of\ application\ A$$

Tabel II shows the highest identification accuracy for each application and the number of considered messages when the highest accuravy can be obtained.

From teble I, it can be confirmed that the degragation of identification accuracy for seven applications except for imap, smtp and Winny is less than two point comparing the inverse method.

The identification accuracy for Winny, which is a P2P software, causes information leakage incidents in Japan, can achieve 96.00% at the number of considered messages is six (table II) although the identification accuracy is decreased to 88.89%.

The accuracy for imap increase to 65.67% when the number of considered messages is 1. This result shows that

Although the identification accuracies of SSH with *nunmber vector* is 100%, this result does not mean that the identification method with *nunmber vector* can identify SSH flows correctly. Most of messages of SSH consist of one packet, conrrespondingly, the number of packets of the messages of FTP, POP3 and POPS is one. Therefore, flows of FTP, POP3 and POPS are identified as SSH flow, and the identification accuracies for FTP, POP3 and POPS are 0% with*nunmber vector*.

The identification accuracy of RTSP is relatively low with the inverse method because the inverse method cannot evaluate the differences between HTTP and RTSP because RTSP is designed based on HTTP. On the other hand, the proposed method with *number vector* can identify RTSP flows with 100% accuracy because *Number vector* can recognize the continuity of the packets transmission to deliver the streaming contents.

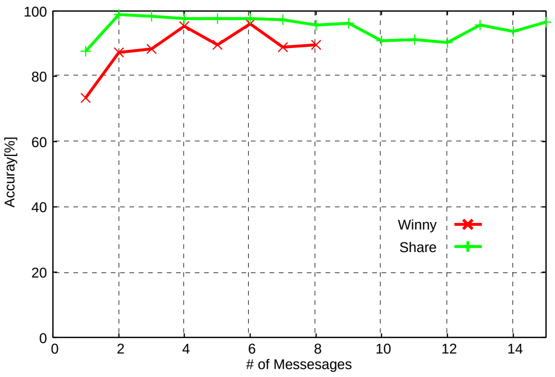

Fig. 8. Identification Accuracy for Winny and Share

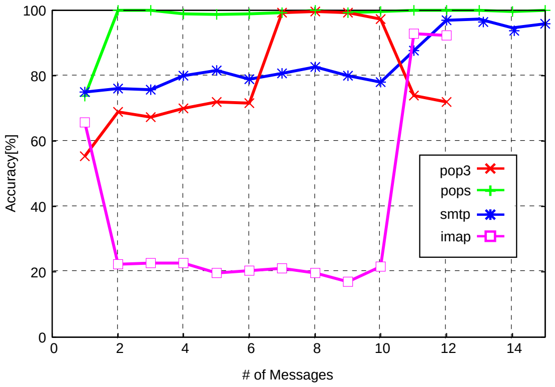

Fig. 9. Identification Accuracy for imap, pop3, pops and smtp

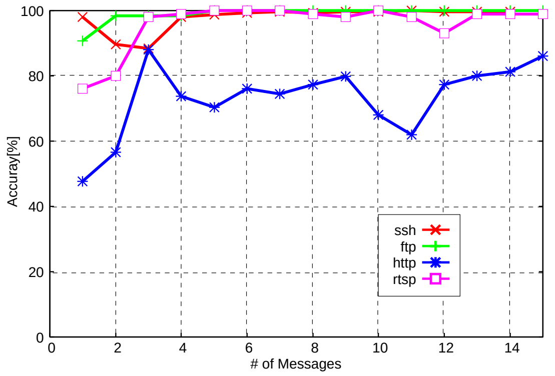

Fig. 10. Identification Accuracy for ftp, http, ssh and rtsp

V. Conclusions

In this paper, a new network application identification method using time intervals to trasmit messages in an application layer and the number of necessary packets to send one message are proposed in order to reduce the risk that malicious users evade identification by tampering with packet size and header. While the time intervals seemed to be unstable observations due to the fluctutaion of system load, the identification accuracy with the time interval did not drop significantly. On the other hand, the identification accuracy with the number of packets per one message is lower than with the time interval although the number of packets per one message seemd to be stable observations. It can be considered that the number of packets per one message does not have enough information to evaluate the differences between applications. However by using the combination vector the elements of which are the time intervals and the number of packets per one message, the highest identification accuracy can obtained. This is because that there is correlation between the time interval and the number of packets, and new discrimination boundaries are created by the correlation.

The total identificaiotn accuracy of the proposed method is lower than that of the conventional method which uses the inverse of packet size. However, the accuracies for some applications are higher than the conventional method. The proposed method can achieve higher accuracy for RTSP. Since RTSP is designed based on HTTP and the packet size of RTSP and HTTP are similar each other, the identification method using the packet size cannot identify them with high accuracy. On th other hand, our proposed method can discriminate the differences between the retrival process of html files of HTTP and the control of video streaming of RTSP. This is because that our proposed method can identify RTSP with high accuracy.

The number of considered messages to achieve the highest accuracy varied for each application. Constructing a method to set the suitable number of considered messages is remained work.

Acknowledgement

This work was supported by Grant-in-Aid for Young Scientists (B) (21700066).

References

[1] T.Masak, A.Shingoand, and O.Ikuo. A classification method for bulk/real-time traffic based on flow statistics. *IEICE technical report*, NS2006-28:29–32, May 2006.

[2] T.Kitamura, T.Shizuno, and T.Okabe. Traffic identification method with packet-type transition pattern analysis. *IEICE technical report*, NS2006-27:25–28, May 2006.

[3] N.Fumitaka, M.Takashi, W.Yasushi, and T.Yoshiaki. Traffic feature analysis and application discrimination. *IEICE technical report*, NS2007-80:57–62, Sep. 2006.

[4] T.MATSUDA, F.NAKAMURA, Y.WAKAHARA, and Y.TANAKA. P2p traffic discrimination technique based on tcp session statistics. *2005 IEICE General Conference*, B-6-121, May 2005.

[5] S. Sen, O. Spatscheck, and D. Wang. Accurate, scalable in-network identification of p2p traffic using application signatures. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 512–521, 2004.

[6] P. Haffner, S. Sen, O. Spatscheck, and D. Wang. Acas: Automated construction of application signatures. In *SIGCOMM '05 Workshops*, Augst 2005.

[7] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield. Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 135–148. ACM Press, 2004.

[8] T.Karagiannis, K.Papagiannaki, and Michalis Faloutsos. Blinc: Multilevel traffic classification in the dark. *ACM SIGCOMM*, pages 229–240, 2005.

[9] T. Karagiannis, A. Broido, M Faloutos, and K.C.Claffy. Transport layer identification of p2p traffic. In *IMC'04*, October 2004.

[10] F.Nakamura, T. Matuda, Y.Wakahara, and Y.Tanaka. Traffic feature analysis and application discrimination. In *IEICE technical report, NS2006-80:57-62*, Sep. 2006.

[11] J.Erman, M.Arlitt, and A.Mahaniti. Traffic classification using clustering algorithms. *MineNet '06: Proceedings of the 2006 ACM SIGCOMM workshop on Mining network data*, pages 281–286, 2006.

[12] A. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *SIGMETRICS'05*, 2005.

[13] N. Williams, S. Zander, and G. Armitage. A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. In *ACM SIGCOMM Computre Communication Review, Vol.36,Number 5*, 2006.

[14] T. Kitamura, T. Shizuno, and T. Okabe. Application classification method based on flow behavior analysis. In *IEICE technical report, NS2005-136:13-16*, Dec. 2005.

[15] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian. Traffic classification on the fly. In *ACM SIGCOMM Computre Communication Review*, 2006.

[16] L. Bernaille, R.Teixeira, and K. Salamatian. Early application identification. In *In Proc. of Confernce on Future NetworkingTechnologies*, Dec. 2006.

[17] Yuji Waizumi, Abbas Jamalipour, and Yoshiaki Nemoto. Network application identiffication based on transition pattern of packets. In *IEEE Wireless Rural and Emergency Communications Conference (WRECOM) 2007*, Oct 2007.

[18] Shinnosuke Yagi, Yuji Waizumi, Hiroshi Tsunoda, Abbas Jamalipour, Nei Kato, and Yoshiaki Nemoto. Network application identiffication using transition pattern of payload length. In *IEEE Wireless Commun. and Network Conference (WCNC) 2008*, Apr 2008.

[19] Shinnosuke Yagi, Yuji Waizumi, Hiroshi Tsunoda, and Yoshiaki Nemoto. A reliable network application identification based on transition pattern of payload length. In *IEEE Globecom 2008*, Dec 2008.

[20] T.Kohonen. *Self-organization and Associate Memory (2nd Edition)*. Spring-verlag, 1998.