Journal of Network and Computer Applications 32 (2009) 1023–1030

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Review

Application classification using packet size distribution and port association

Ying-Dar Lin [a], Chun-Nan Lu [a,*], Yuan-Cheng Lai [b], Wei-Hao Peng [a], Po-Ching Lin [a]

[a] Department of Computer Science, National Chiao Tung University, HsinChu, Taiwan
[b] Department of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan

ARTICLE INFO

Article history:
Received 7 November 2007
Received in revised form
3 March 2009
Accepted 3 March 2009

Keywords:
Traffic classification
Transport layer behaviors
Packet size distribution
Ports association

ABSTRACT

Traffic classification is an essential part in common network management applications such as intrusion detection and network monitoring. Identifying traffic by looking at port numbers is only suitable to well-known applications, while signature-based classification is not applicable to encrypted messages. Our preliminary observation shows that each application has distinct *packet size distribution* (PSD) of the connections. Therefore, it is feasible to classify traffic by analyzing the variances of packet sizes of the connections without analyzing packet payload. In this work, each connection is first transformed into a point in a multi-dimensional space according to its PSD. Then it is compared with the representative points of pre-defined applications and recognized as the application having a minimum distance. Once a connection is identified as a specific application, *port association* is used to accelerate the classification by combining it with the other connections of the same session because applications usually use consecutive ports during a session. Using the proposed techniques, *packet size distribution* and *port association*, a high accuracy rate, 96% on average, and low false positive and false negative rates, 4–5%, are achieved. Our proposed method not only works well for encrypted traffic but also can be easily incorporated with a signature-based method to provide better accuracy.

Contents

* Corresponding author.
E-mail addresses: ydlin@cs.nctu.edu.tw (Y.-D. Lin), cnlu@cs.nctu.edu.tw (C.-N. Lu), laiyc@cs.ntust.edu.tw (Y.-C. Lai), whpeng@cs.nctu.edu.tw (W.-H. Peng), pclin@cs.nctu.edu.tw (P.-C. Lin).

1084-8045/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2009.03.001

1. Introduction

Traffic classification plays an important role in common network management applications, such as intrusion detection and network monitoring. An enterprise or a service provider can apply various rules to protect network resources or enforce organization policies according to the classification results. Accurate traffic classification is therefore the keystone in network management and monitoring. However, it is challenging to classify the applications associated with network connections according to their diverse characteristics and behaviors.

A number of methods have been proposed to identify and classify the applications associated with the traffic. Although traditional classification methods based on port numbers (Moore et al., 2001; Fraleigh et al., 2003; Karagiannis et al., 2004a, b; Moore and Papagiannaki, 2005) and signatures in the packet payload (Roesch, 1999; Paxson, 1999) are usable, they do not work well for growing peer-to-peer (P2P) applications, which intend to disguise themselves. Some specific protocols, like HTTP, are frequently used to relay other kinds of traffic and emerging applications tend to avoid the use of well-known ports, such as peer-to-peer services. Nowadays, P2P applications have the ability to use arbitrary ports in an attempt to go undetected. Furthermore, they intentionally try to camouflage their traffic to go undetected. Since many existing non-standard P2P protocols use payload encryption and port randomization to hide from firewalls and network security systems, the difficulty in classification gets even higher. Therefore, it is important to find new characteristics to help traffic classification.

In this paper, a technique that utilizes packet size distribution (PSD) per connection to associate with its application is proposed. Connections are defined by the 5-tuple source IP, destination IP, protocol, source port and destination port. Each distinct application has its own specific PSD. When the PSD of a certain connection is determined, it is compared with all representatives of the pre-defined applications to decide which application it belongs to. Since the approach does not perceive the information of payloads, this method works whether the packet payloads are encrypted or not. In addition, it is observed that applications tend to use consecutive ports to communicate. Therefore, we can use port locality to associate connections of the same session of an application. If the source and destination IP addresses of two connections are the same and their port numbers are consecutive, the two connections are inferred to belong to the same session of an application. From simulations and online tests, our method obtains accurate results with fast speed.

This paper is organized as follows. Section 2 describes the related work. In Section 3, two observations of our work would be introduced. In Section 4, our classification methodology is formally stated. We evaluate the accuracy and performance of our proposed methods in Section 5. Finally, the conclusion and some future work are given.

2. Related work

The idea of using observed statistical properties of network traffic to classify flows or to describe their behavior is not new. Those work (Paxson, 1994; Paxson and Floyd, 1995), proposed by Paxson et al. on Internet traffic characterization, concentrate on the relationship between the statistical properties of flows and the application that generated them. The analytical models based on random variables like packet length, flow duration and inter-arrival times are shown to be applicable to express the behavior of a few application protocols. But it does not make any further attempt to classify network traffic according to application layer protocols. Later, Hernandez-Campos et al. (2003) showed that the similarity of traffic patterns between different application layer protocols can be used to group flows into hierarchical clusters. The technique presented by Roughan et al. (2004), using statistical signature-based classification, classifies traffic into different classes of services for QoS implementations based on five categories (simple packet, flow, connection, intra-flow/connection and multi-flow).

Some works classify network traffic based on summarized flow information such as packet length, flow duration, number of packets, packet train length, packet train size and mean packet inter-arrival time (Saifulla et al., 2002; Divakaran et al., 2006; Crotti et al., 2006, 2007). The work of Bernaille et al. (2006) also belongs to this kind of class, proposes identifying the application based on packet sizes and direction of packets. After observing the first packets of a TCP connection and characterizing the behavior of the pre-selected applications by a set of clusters in a multi-dimensional space, a flow is then classified to an application by means of the minimum Euclidean distance. Compared with our approach, it has however two limitations for network management activities. First, it could only do basic statistical counts, like the times a specific application session appears, but could not provide detailed characteristics of the session, like the numbers of flows and the individual volume count of flows. Second, it could not track down suspicious activities to enforce proactive prevention. Like eMule, which described one kind of peer-to-peer applications, its execution could be conceptually separated into three stages: server lookup, peers lookup and data exchanging. The study by Bernaille et al. (2006) could be only aware of the emergence times of eMule by observing the first packets in server-lookup stage. It cannot even block, filter or record the related peers with whom considerable peers build connections in the following peers-lookup stage.

BLINC, proposed by Karagiannis et al., introduces a new approach for traffic classification based on the analysis of host behavior. It associates Internet host behavior patterns with one or more applications, and refines the association by heuristics and behavior stratification. It is able to accurately associate hosts with the service they offer or use by inspecting all the flows generated by specific hosts. However, it cannot classify a single TCP connection because it has to gather information from multiple flows for each individual host before it can decide on the role of the host.

Apart from aforementioned schemes, there are a number of proposals devoted to the application of machine learning (ML) techniques to traffic classification (Moore and Zuev, 2005; Nguyen and Armitage, 2006; McGregor et al., 2004; Zander et al., 2005). The work of Moore and Zuev (2005) adopts the Naïve Bayesian technique to classify traffic and uses the Fast Correlation-Based Filter method to select suitable features. Nguyen and Armitage (2006) also used the Naïve Bayesian ML algorithm and the classifier is trained with multiple sub-flows generated by sliding windows like inter-packet length variation. Generally speaking, ML researches involve mainly two steps. First, extensive *features* are defined based on statistical characteristics of application protocols such as flow duration, inter-arrival times, packet length etc. An ML classifier is then trained to associate sets of features with known traffic classes, and apply the well-trained ML classifier to classify unknown traffic using previously learned rules (Nguyen and Armitage (2008)). Table 1 lists a summary of related works.

3. Features in classification

This section describes two important observations: (1) the same kind of applications has similar PSD, and different kinds of

applications have distinct PSDs; (2) the port numbers of related connections among a session are adjacent. In the following context, an application session is defined as follows: (1) for P2P applications, one instance executes continuously until stopping intentionally even if there are multiple connections in it; (2) a client's request is served by servers until completed even if there are multiple requests and responses contained in it; and (3) a voice instance runs without breaking off from beginning to end.

3.1. Packet size distribution

The PSD of an application can be obtained from all of its own connections. We manually generate traces of each application to get pure application traffic. The main advantage of manual generation is that we know exactly what application generated within each connection. Each pre-selected application is executed in turn, and the traffic generated is recorded while passing through the network interface. Other traffic, except only pre-selected application traffic, is then filtered out. The applications used and their corresponding information are listed in Table 2. Pre-selected applications' traffic is produced in different scenarios.

(1) *Time duration*: 30 s, 5 min, and the whole session.
(2) *Data sources*: for the file-sharing applications, different file sources are used to generate traffic. Otherwise, the whole session is normally recorded.
(3) *Host loads*: different host loads are selected while the application traffic is generating.
(4) *Application configurations*: different configurations of bandwidth, quality-of-services, or encrypted protocol are selected.

We found that different applications have distinct PSDs. Each application has certain frequent packet sizes. Fig. 1 shows the PSD of each pre-selected application excluding the common full-payload packets and the shortest packets such as ACK/SYN/FIN ones. ShoutCast and WoW have packet sizes whose proportions are too small to show up on the tops. We also observed that the same kind of applications has similar PSD. Fig. 2(a) and (b) shows the PSDs of two instances from Bittorrent version 0.81 under different host loads. The observations can validate that the distribution of packet sizes is a good characteristic to classify applications.

3.2. Port locality

In addition, we also observed that applications tend to have the ability of port locality, which means applications get used to assign ports to communicate in a consecutive manner. During communication, if an application session needs to build a new connection, the application will assign continuous ports for each new connection, even for the applications using randomized destination port numbers first. Once one certain port is used for the first connection, the remaining connections would very likely use continuous port numbers from the first port assigned. This feature is very useful because if a connection is recognized as a specific application, the port information could be used to associate with other connections of the same session. Although port locality is helpful, it is likely that every application executed uses non-consecutive ports. In this case, each connection is individually regarded as single connection.

4. Methodology

Based on the aforementioned observations, our proposed classification methodology runs in two phases: an offline center training phase and an online traffic classification phase. Fig. 3 shows the overview of our classification procedure.

The left side represents the steps of the training phase and the right side shows the components of the online classifier. First, the offline center training phase uses a set of traffic data to decide the representative centers of the pre-selected applications in a multi-dimensional space and outputs the representative centers to the online classification phase. The online traffic classification phase first parses all connections in real-world traffic and extracts the 5-tuple (source IP, destination IP, protocol, source port, destination port) and the packet size distribution of the connection. Next, the online classifier compares the connections with the entries in the table of port locality. If they are similar except for port used, the new connection is inferred to belong to the same application session with the entry in the table. Otherwise, the new connection would be judged according to classification criteria and determines which applications they belong to.

4.1. Dominating sizes (DS) and DS proportion (DSP)

To develop the representative centers of pre-selected applications from a set of traffic data, we need a representation of each connection. Excluding the common full-payload packets and the shortest control packets such as ACK/SYN ones, we pay attention to those packet sizes with larger proportions in a connection. Assume that there are multiple packets collected in a connection P, and the packets of the same sizes are put together. Then the set of different packet sizes are sorted in a decreasing sequence, $SEQ_P = \langle ps_1, ps_2, ps_3, \ldots \rangle$, where $pro(ps_{i-1}) \geqslant pro(ps_i)$. Here $pro(ps_i)$ denotes the proportion of the ith larger packet size in P. Based on SEQ_P, the *Dominating Size Proportion* of a connection P is defined as a subsequence of SEQ_P, $DSP_P = \langle pro(ps_1), pro(ps_2), pro(ps_3), \ldots, pro(ps_t) \rangle$, and the sum of all proportions in DSP_P, t distinct kinds of packet sizes, is larger than a user-defined threshold, 90% here, which is called *Size bound*. In the meanwhile, the corresponding subsequence $\langle ps_1, ps_2, ps_3, \ldots, ps_t \rangle$ is defined as *Dominating Sizes*.

4.2. Change cycle (CC)

Some applications have higher size variations than others. It is helpful to decide which application a connection belongs to by quantifying the size variation. Let s_i be the size of the ith packet in a connection. If s_i and s_{i-1} are not the same, we call it as *size change*. The average number of packets between two consequent size changes is defined as *Change Cycle*.

4.3. Port association table (PAT)

To assist and speed up identification, we propose an auxiliary structure called *Port Association Table*, which is a table used to store the information of port locality. Once a connection is recognized as the session of a specific application, 4-tuple ⟨SrcIP, SrcPort, DstIP, DstPort⟩ is extracted from the connection. This information will be recorded in PAT with the form ⟨SrcIP, SrcPort, AppName⟩ and ⟨DstIP, DstPort, AppName⟩. For any other connection, if one of its hosts is already in PAT and the port number in the pair ⟨SrcIP, SrcPort⟩ or ⟨DstIP, DstPort⟩ is continuous compared with one entry of the PAT, this connection is inferred to belong to the same session.

Table 1
A summary of related works.

Proposals	Features	Algorithms	Computation overhead	Year
McGregor et al. (2004)	• Packet size statistics (min, max, quartiles, etc) • Inter-arrival statistics • Byte counts • Connection duration • Number of transitions between transaction mode and bulk transfer mode • Time spent (idle, bulk transfer, transaction mode)	Expectation maximization	Medium	2004
Roughan et al. (2004)	• Packet-level (mean packet size, variance, root mean square size) • Flow level (mean flow duration, mean data volume, mean number of packets, variance) • Connection level (the symmetry of a connection, advertised window size, throughput distribution) • Intra-flow/connection (inter-arrival times between packets, loss rates, latencies) • Multi-flow	Nearest neighbor and linear discriminant analysis	Medium	2004
Moore and Zuev (2005)	248 per-flow discriminators for only TCP connections like • Flow duration • TCP Port • Packet inter-arrival time (mean, variance,…) • Payload size (mean, variance,…) • Effective bandwidth based upon entropy • Fourier transform of the packet inter-arrival time	Naïve Bayes techniques (Naïve Bayes Kernel estimation plus fast correlation-based filter)	Medium	2005
Zander et al. (2005)	• Packet inter-arrival time • Packet length mean and variance • Flow size (bytes) • Flow duration	Autoclass	Medium	2005
Karagiannis et al. (2005)	• Host negotiation and ports relationship • Combining signature based with host relationships • Use Social level, network level, and application level to describe the behavior of applications	Supervised clustering	High	2005
Bernaille et al. (2006)	Size of the first five data packets of each TCP flow	K-means (unsupervised clustering)	Low	2006
Nguyen and Armitage (2006)	• Inter-packet arrival interval • Inter-packet length variation and IP packet length (all with min, max, mean and standard deviation values)	Naïve Bayes technique (supervised)	Medium	2006
Divakaran et al. (2006)	• Packet train length (number of packets within a flow) • Packet train size (sum of the sizes of all packets within a flow)	Vector quantization and Bayesian technique (gaussian mixtures)	Medium	2006
(Our work)	Packet size distribution within per flow (the variances of packet sizes of the flows)	Supervised clustering	Low	2007
Crotti et al. (2007)	• Packet size • Inter-arrival times • Packet arrival order	Protocol fingerprint	Medium	2007
Huang et al. (2008)	• Statistical information of each flow at first k rounds • Elapsed time • Transmitted size • Throughput • Response time • Inter-arrival time	Supervised clustering	Medium	2008

4.4. *Offline center training*

This phase runs in an offline manner and finds out the representatives of pre-defined applications. It takes packet traces of pre-defined applications as input. Each new TCP connection is associated with a spatial representation based on the PSD. Assume there are at most n different packet sizes of DS. Each connection is represented as a point of $\langle$DS, DSP, CC$\rangle$ in a $(2n+1)$-dimensional space. The dimension of Moore and Keys and Koga and Lagache and Claffy KC, 2001, n represents the packet size, the dimension of $[n+1, 2n]$ is the packet size proportion, and the last one is the average number of packets in a change cycle. In order to balance the impact of the three factors as much as possible, some normalization steps taken to let these values ranging between 0

and 1 are as follows:

$$\text{normalized\_DS} = \text{DS}/\text{Maximum Transmission Unit (MTU)} \quad (1)$$

$$\text{normalized\_}DSP = DSP \quad (2)$$

$$\text{normalized\_}CC = 1/CC \quad (3)$$

Besides, we also noticed that well-known protocol applications, such as Apache and zFTP server, have less-frequent size changes than P2P and streaming applications such as Bittorrent and eMule. Different connections may have different dimensions because of a distinct number of DS. To compare two connections of different dimensions, the lower-dimension connection would be expanded to the same dimension as the higher-dimension one and the newly expanded dimensions are filled up with zeros.

Then the center of an application is derived from all connection vectors of this application. Assume that k connection vectors exist in an application A, and then the center of A is defined as $C = \sum v_i/k$ for $1 \leqslant i \leqslant k$, where v_i stands for the ith connection vector. Using different application centers causes totally different results. Note that our proposed method will generate different centers for various implementations of the same application, since an application having various implementations on different platforms actually has different behaviors.

Table 2
Pre-selected application categories used.

Application name	Class	Version	Protocol
BitTorrent	P2P	0.81	TCP
eMule	P2P	0.47b	TCP/UDP
Skype	P2P	3.0	TCP/UDP
MSN	P2P	8.1	TCP
Apache	Http	2.2.4	TCP
zFTP server	FTP	7.4.4	TCP
ShoutCast	Streaming	1.9.7	TCP
World of warcraft (WoW)	Gaming	2.0	TCP

4.5. Online traffic classification

This phase takes online traffic as input, transforms all connections into the vectors defined in Section 4.2, and compares their IP addresses and port numbers with entries in PAT. If a connection is found in PAT, it is inferred to belong to a certain application, and its ⟨SrcIP, SrcPort⟩ and ⟨DstIP, DstPort⟩ would also be added into PAT. If the corresponding pairs of the connection do not appear in PAT, a measurement of similarity between the representations of two connections is needed. Here, the minimum *Euclidean distance* criterion is adopted. Our method then computes the *Euclidean distance* between the new connection and all the centers and decides to which application the connection belongs to. The *Euclidean distance* is defined as follows:

$$E\_Dist = \sqrt{Dist(DS_A, DS_B) + Dist(DSP_A, DSP_B) + (CC_A - CC_B)^2} \quad (4)$$

where $Dist(A, B)$ is the *Euclidean distance* between two-dimensional connection A and B, and DS, DSP, and CC represent Dominating Sizes, DS Proportion, and Change Cycle, respectively. Then the end hosts of the newly recognized connection would be added into PAT. If the Euclidean distance is larger than a threshold, the new connection cannot be regarded as belonging to an application even if its distance among all is the smallest.

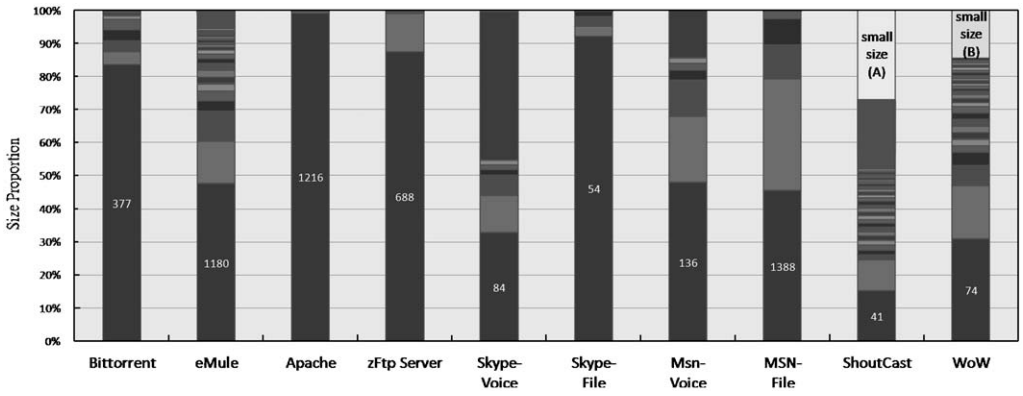

Fig. 1. PSD of each pre-selected application. Section A, B: too many small proportions of sizes to show. The number in each application is the most frequent packet size.

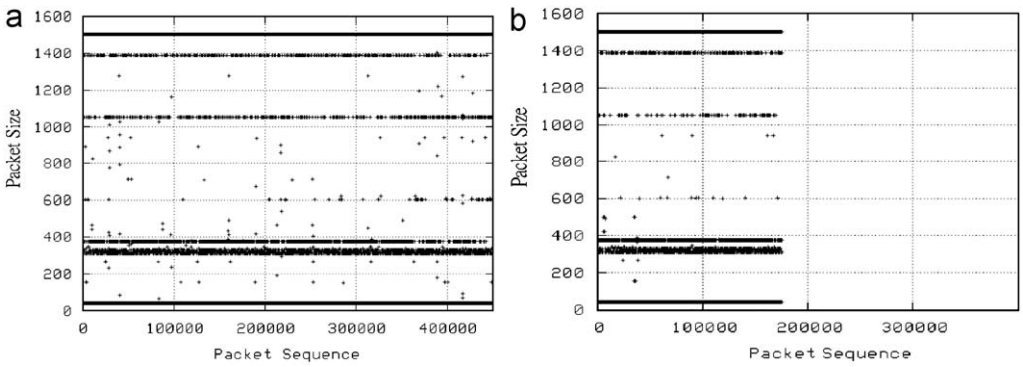

Fig. 2. Bittorrent application in two different host loads have similar PSD. (a) Bittorrent instance 1 (b) Bittorrent instance 2.

5. Evaluation

We obtain the traffic traces of pre-selected applications in a controlled fashion by running the instances of these applications and collecting the traffic. We classify the real-world traces recorded from High Speed Lab and Department of Computer Science in National Chiao-Tung University to evaluate our method because there are no standard testing procedures and benchmarking metrics. Most proposals build their classification models based on the sample data collected from different networks and scenarios, which makes it difficult to compare the distinct approaches.

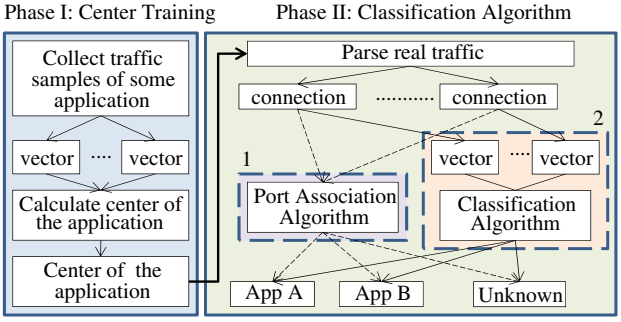

Fig. 3. Classification methodology.

Table 3
Connections in the collected traffic.

Application name	A	B	C	D	Sum
eMule	1865	0	744	1381	2125
BitTorrent	1675	0	785	890	335
MSN-File	0	0	250	250	500
Skype-File	0	0	250	250	500
MSN-Voice	0	0	250	250	500
Skype-Voice	0	0	250	250	500
Apache	500	0	250	250	500
zFTP server	740	0	360	380	740
Shoutcast	500	0	250	250	500
Worldofwarcraft	0	250	0	0	250
Background traffic					2210
Sum of all connections					10000

5.1. Evaluation environment

In the traffic-recording environment, we select four PCs, say hosts A, B, C, and D, to run specified applications on them. The host "A" plays the role of the server; it supplies the seed of P2P file-sharing applications, and runs as Apache servers, zFTP servers, and ShoutCast servers. Other interactive applications without our intervention are executed on host "B", like World of WarCraft, MSN chatting, and Skype chatting. The rest "C" and "D" play the role of receivers. For the purpose of tracing every connection that comes from any application, we run Netlimiter inside the PCs to log all information of connections passed through. Table 3 shows the connections collected in the traffic.

5.2. Connection recognition (CR) rate

Fig. 4 presents Connection Recognition rate of our method. Connection Recognition rate means the percentage of connections of an application that is correctly classified. From this figure, it is helpful for recognition to identify applications with PAT. For BitTorrent, eMule, and zFTP server, the recognition rates with and without PAT are increased. For the remainder applications, the recognition rates with and without PAT are equal. For the equal results of comparisons with and without PAT, there are two reasons. First, the applications use only a single connection at running, like MSN-file transferring, MSN-voice chatting, Skype-file transferring, Skype-voice chatting, ShoutCast streaming, and WoW. Second, the individual port numbers assigned to connections of an application may not be continuous or even dynamically assigned. In this case, the whole connections of the application may be regarded as independent connections, and stored, respectively, in PAT.

5.3. Session recognition (SR) rate

A metric, session recognition rate, which represents the percentage of sessions of an application that is correctly classified, is used to measure the accuracy of session recognitions. Fig. 5 shows the session recognition rate for every application using our method. It is promising that P2P applications and well-known protocol implementations, like Apache and zFTP server, have very high accuracy, say 98% on average. But for Skype-voice and MSN-voice chatting, 74% and 80%, respectively, it is difficult to be recognized because their constituent connections are not always

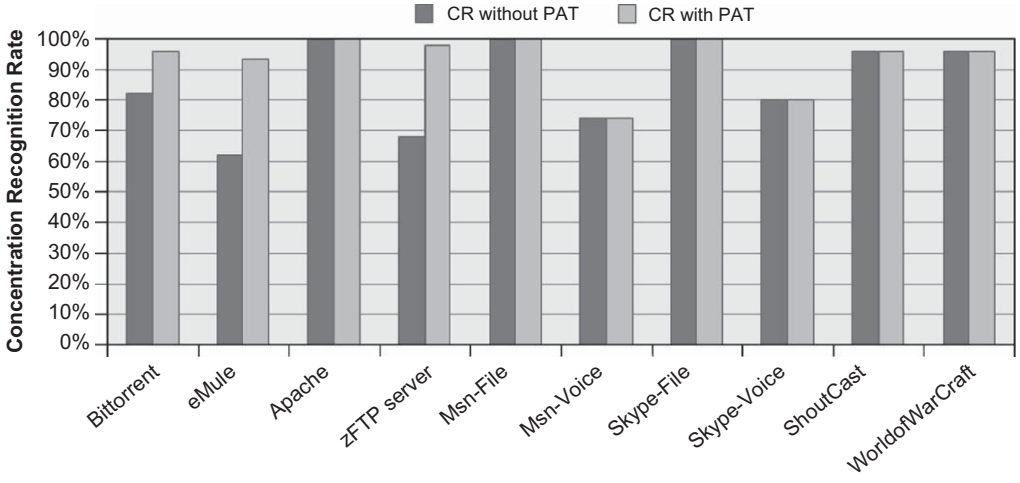

Fig. 4. Connection recognition (CR) rate.

similar. Generally speaking, an application session is identified only if all of its constituent connections are recognized. However, by utilizing the characteristic of port locality, an application session can be quickly identified by recognizing any one of its constituent connections first and using port association later. Observing from Figs. 4 and 5, it is not unexpected for the results to agree with our method. Therefore, it is concluded that using port association can help session recognition.

5.4. False-positive rate and false-negative rate

A false positive means that a connection is classified into an incorrect application, while a false negative means that it cannot be identified. The possibilities of incorrect classification are caused by two main reasons. First, when the centers of two different applications are too close, an ambiguous decision is made. Second, when the numbers of packets in one connection are too small to obtain enough information to classify it correctly, a wrong decision is also made. For the first, we could choose appropriate centers that are not too close to recognize that different implementations of the same application have distinct application characteristics of PSD and the alternative center may be more suitable for that case. For the second, we can enlarge the dominating size threshold in our method to try to obtain more packets to be used.

Table 4 presents false-positive and false-negative rates of each application. False negative happens due to two reasons. First, some applications may have many diverse implementations that are not included in our training traffic. Second, the application has other links used to query and control that are not considered. Our method mainly focuses on data transferring and if other links have changing volumes of data and the proportions are relatively low, it may not be identified. For zFTP server, the false positive occurs when we evaluate its command connection, and for streaming services, like MSN or Skype, it happens when the PSD varies more widely than usual.

6. Conclusions and future work

In this paper we have proposed two observations and a statistical classification technique based on the analysis of simple properties of network traffic. One of the two observations is that the same kind of applications has similar PSD, and different kinds of applications have distinct PSDs. From the viewpoint, PSD is a good characteristic to classify applications. The other one is that the port numbers used in the related connections are adjacent. Once one certain port is used for the first connection, the remaining connections would use continuous port numbers from the first port assigned. This feature is also useful because if a connection is recognized as a specific application, the port information could be used to associate with other connections of the same application session.

By utilizing these two observations, we develop a classification algorithm without accessing packet payload information and achieve high accuracy at the expense of low false-positive rate for applications. The classification mechanism is composed of two phases, offline training phase and online classification phase. Based on a set of training traffic, the representative centers of pre-selected applications could be decided in the offline training phase. With the representative centers of pre-selected applications, the online classifier can classify network traffic easily. As long as one certain connection is parsed and recognized with PSD, a port association is used to help in recognizing other connections of the same session. Using the proposed techniques, a high accuracy rate, 96% on average, and low false-positive and false-negative rates, 4–5%, are achieved. Additionally, our method also works well for encrypted applications, like Skype-file sharing and voice chatting, because our method does not need to access the packet payload information and can be easily incorporated with a signature-based method to provide higher accuracy.

Table 4
False-positive rate and false-negative rate of each application.

Application name	False positive (%)	False negative (%)
Bittorrent	0	4
eMule	0	7
Apache	0	0
zFTP	1	1
MSN-file	0	0
MSN-voice	9	1
Skype-file	0	0
Skype-voice	2	18
Shoutcast	1	3
World of warcraft	0	4
Unknown	7	0

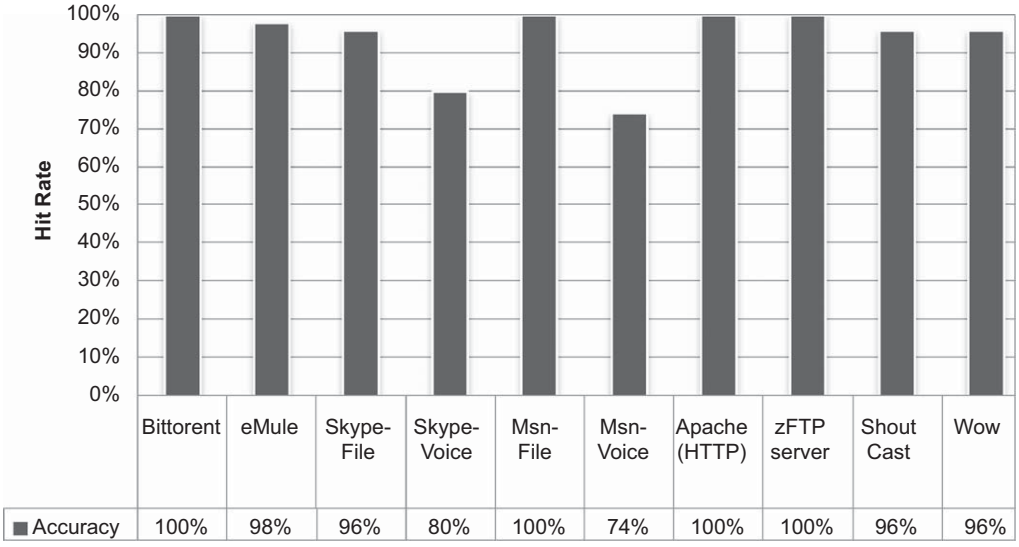

Fig. 5. Accuracy for session recognition.

In the near future, we will study further towards three directions. First, we will evaluate other factors that may affect PSD, like network link type. It is obvious that the size distribution of the same connection would be affected by the network link type and surely different. For example, the MTU of Ethernet is 1500 bytes, and the one of PPP is 1492 bytes. Second, we hope to explore the effect of different implementations of the same kind of application, especially for non-standard application protocols. Different implementations might produce distinct application centers. How to raise the accuracy and reduce the sensitivity due to different implementations is undoubtedly significant. Third, we want to make more detailed analysis about the evolving applications. We try to use alternative viewpoint of dividing an application into multiple execution stages based on its diverse PSD information. If we could collect more accurate training traffic about each execution stage to refine the stage representative sub-centers respectively, it would be more easy to administer the kind of applications or track down suspicious activities by monitoring the execution progress.

References

Bernaille L, Teixeira R, Akodjenou I, Soule A, Salamatian K. Traffic classification on the fly. In: Proceedings of the ACM SIGCOMM computer communication review, 2006.

BitTorrent. 〈http://bitconjurer.org/BitTorrent〉.

Crotti M, Gringoli F, Pelosato P, Salgarelli L. A statistical approach to IP-level classification of network traffic. In: Proceedings of the 41th IEEE international conference on communications, June 2006.

Crotti M, Dusi M, Gringoli F, Salgarelli L. Traffic Classification through Simple Statistical Fingerprinting. In: ACM SIGCOMM computer communication review on vol. 37, no. 1, January 2007.

Divakaran DM, Murthy HA, Gonsalves TA. Traffic modeling and classification using packet train length and packet train size. In: IPOM 2006.

eMule. 〈http://www.emule-project.net/〉.

Fraleigh C, Moon S, Lyles B, Cotton C, Khan M, Moll D, et al. Packet-level traffic measurements from the sprint IP backbone. In: Proceedings of the IEEE Network, 2003.

Hernandez-Campos F, Donelson F, Jeffay Smith K., Nobel AB. Statistical clustering of internet communications Patterns. In: computing science and statistics, vol. 35, July 2003.

Huang Nen-Fu, Jai Gin-Yuan, Chao Han-Chieh. A high accurate machine-learning algorithm for identifying application traffic in early stage. In: Proceedings of the IEEE ICC, 2008.

Karagiannis T, Papagiannaki K, Faloutsos M. BLINC: multilevel traffic classification in the dark. In: Proceedings of the conference on applications, technologies, architectures, and protocols for computer communications SIGCOMM, 2005.

Karagiannis T, Broido A, Brownlee N, Claffy K, Faloutsos M. Is P2P dying or just hiding? In: Proceedings of the IEEE GLOBECOM, 2004a.

Karagiannis T, Broido A, Faloutsos M, Claffy K. Transport layer identification of P2P traffic. In: Proceedings of the fourth ACM SIGCOMM conference on internet measurement, 2004b.

McGregor A, Hall M, Lorier P, Brunskill J. Flow clustering using machine learning techniques. In: Proceedings of the fifth passive and active measurement workshop (PAM 2004), March 2004.

Moore AW, Papagiannaki K. Toward the Accurate Identification of Network Applications. In: Proceedings of the sixth passive and active measurement workshop (PAM 2005), October 2005.

Moore A, Zuev D. Internet traffic classification using Bayesian analysis techniques. In: Proceedings of ACM international conference on measurement and modeling of computer systems (SIGMETERICS), 2005.

Moore D, Keys K, Koga R, Lagache E, Claffy KC. The CoralReef software suite as a tool for system and network Administrators. In: LISA '01: Proceedings of the 15th USENIX conference on systems administration, December 2001.

Netlimiter. 〈http://www.netlimiter.com〉.

Nguyen T, Armitage G. Training on multiple sub-flows to optimize the use of machine learning classifiers in real-world IP networks. In: Proceedings of IEEE 31st conference on local computer network (LCN 2006), 2006.

Nguyen T, Armitage G. A survey of techniques for internet traffic classification using machine learning. In: Proceedings of the IEEE communications on surveys and tutorials, 2008.

Paxson V. Empirically derived analytic models of wide-area TCP connections. In: Proceedings of the IEEE/ACM transactions on networking, August 1994.

Paxson V. Bro: a system for detecting network intruders in real-time. In: Computer networks, 1999.

Paxson V, Floyd S. Wide area traffic: the failure of Poisson modeling. In: Proceedings of the IEEE/ACM transactions on networking, June 1995.

Roesch M. SNORT: Lightweight intrusion detection for networks. In: LISA '99: Proceedings of the 13th USENIX conference on systems administration, November 1999.

Roughan M, Sen S, Spatscheck O, Duffield N. Class-of-service mapping for qos: A statistical signature-based approach to IP traffic classification. In: Proceeding of the IMC, 2004.

Saifulla MA, Murthy HA, Gonsalves TA. Identifying patterns in internet traffic. In: Proceedings of the 15th international conference on computer communication, 2002.

Zander S, Nguyen T, Armitage G. Automated traffic classification and application identification using machine learning. In: Proceedings of IEEE conference on local computer networks (LCN 2005), 2005.