Journal of Network and Computer Applications 76 (2016) 60–74

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

High performance traffic classification based on message size sequence and distribution

Chun-Nan Lu[a,*], Chun-Ying Huang[a], Ying-Dar Lin[a], Yuan-Cheng Lai[b]

[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

Keywords:
Traffic classification
Packet size
Message size
Distribution
Sequence

ABSTRACT

Classifying network flows into applications is a fundamental requirement for network administrators. Administrators used to classify network applications by examining transport layer port numbers or application level signatures. However, emerging network applications often send encrypted traffic with randomized port numbers. This makes it challenging to detect and manage network applications. In this paper, we propose two statistics-based solutions, the message size distribution classifier (MSDC) and the message size sequence classifier (MSSC) depending on classification accuracy and real timeliness. The former aims to identify network flows in an accurate manner, while the latter aims to provide a lightweight and real-time solution. The proposed classifiers can be further combined to build a hybrid solution that achieves both good detection accuracy and short response latency. Our numerical results show that the MSDC can make a decision by inspecting less than 300 packets and achieve a high detection accuracy of 99.98%. In contrast, the MSSC classifier can respond by only looking at the very first 15 packets and have a slightly lower accuracy of 94.99%. Our implementations on a commodity personal computer show that running the MSDC, the MSSC, and the hybrid classifier in-line achieves a throughput of 400 Mbps, 800 Mbps, and 723 Mbps, respectively.

1. Introduction

Classifying a network flow into its source application is essential for application-aware network management. By associating network flows with source applications, network administrators can enforce various access control policies to better utilize the available network resources. However, it is not an easy task to correctly classify network flows into the corresponding applications because the use of obfuscation techniques such as port number randomization, payload encryption, and network tunneling. As a result, characterization of Internet traffic has become one of the major challenging issues in communication networks over the past few years (Azzouna and Guillemin, 2003).

A number of approaches have been proposed to classify network flows. The most primitive solution is port-based classification, which builds mappings from transport layer port numbers to applications. For example, map port 53 to DNS flows, port 20 and 21 to FTP flows, and port 25 to SMTP flows. The advantage of this solution is simple. However, it has an obvious flaw because an application is able to bypass the detection by using an unmapped port number or even masquerading an irrelevant well-known port number. One common case is the HTTP-tunneling, which is used to carry non-HTTP network flows over regular HTTP network flows using port 80. Therefore, port-based classification often fails to provide an accurate and reliable solution.

To overcome the drawback of port-based classification, researchers have proposed to detect network flows by finding specific signatures in payloads (Sen et al., 2004). Signature-based classification is considered to be more reliable. However, it did not solve all the issues. First, an application can employ encryption or encapsulation techniques to intentionally obfuscate packet contents; second, this solution requires precise and up-to-date signatures, which might not be applicable for proprietary applications; third, it is computation-intensive to compare characters in each payload against all the available signatures. These unresolved issues pushed research communities to seek for better solutions without inspection payloads.

Many recent approaches classify network flows based on statistical features. These solutions assume that an application would have certain unique statistical properties that can be obtained from empirical data and then used to classify flows into corresponding applications. Common statistical features include the volume, the duration, the burstiness, the payload size, and the jitter of network flows. Statistical-based traffic classification becomes a good alternative because it is

* Corresponding author.
E-mail addresses: cnlu.cs95g@nctu.edu.tw (C.-N. Lu), chuang@cs.nctu.edu.tw (C.-Y. Huang), ydlin@cs.nctu.edu.tw (Y.-D. Lin), laiyc@cs.ntust.edu.tw (Y.-C. Lai).

http://dx.doi.org/10.1016/j.jnca.2016.09.013
Received 29 September 2015; Received in revised form 21 April 2016; Accepted 29 September 2016
Available online 15 October 2016

possible to classify encrypted or obfuscated network flows.

Roughan et al. (2004) statistically abstracted application features based on application layer protocol attributes and used the features to classify network flows into a specific class-of-service, while Moore and Zuev, 2005 combined statistical analysis with the Bayes theorem to classify network flows. Selected features for the classifiers include the transport layer port number, the flow duration, the packet inter-arrival time, the payload size, and the effective bandwidth. Bernaille et al. (2006) adopted unsupervised clustering techniques to identify an application by using the sizes of the first five data packets of each TCP flow. The solution can make a decision in a pretty short time. However, the solution is sensitive to packet loss and out-of-order delivery.

Other researchers attempt to classify network flows based on observed application behaviors. They monitored and modeled application behavior profiles and then used the profiles to classify flows. Karagiannis et al. (2005) presented BLINC, which analyzed the communication patterns of transport layer host behavior at three levels of details: social, functional, and application, and then used these application features to classify network flows into groups.

However, the classification accuracy directly based on statistical features or observed behaviors are not satisfactory because of sophisticated application behaviors. Network behavior of one application may be similar to that of another application. For example, the behavior of an HTTP file transfer could be similar to that of an FTP transfer. In contrast, not all flows generated by an application behave similar. A BitTorrent client may simultaneously establish flows to retrieve the list of servers, look up resources, check peer status, and exchange files. Making good use of the scattered information can also help classification. Thus, to have a better classification accuracy, an approach, namely message size distribution classifier (MSDC) (Lu et al., 2012), was proposed to classify network flows into sessions and further obtain a complete picture of application behaviors.

MSDC contains two phases, i.e., flow classification and flow grouping. The former classifies network flows into applications by packet size distribution (PSD) and the latter groups related flows as a session by port locality. A flow is identified by the five-tuple information, which includes source IP, destination IP, source port, destination port, and transport layer protocol. When the PSD of one flow is determined, it is compared against the representative of each pre-selected application to decide which application the flow belongs to. Besides, flows are grouped as a session by checking port locality because underlying operation systems often allocate consecutive port numbers for flows of an application. If flows of a session are classified into different applications, an arbitration algorithm based on majority votes is then invoked to make corrections. Evaluations and online benchmarks show that MSDC can obtain accurate results and make a decision by inspecting at most 300 packets and the overall throughput exceeds 400 Mbps on a mainstream computer. Although MSDC can classify network flows accurately, it works in a not-so-fast manner. Therefore, we propose another lightweight and real-time solution called message size sequence classifier (MSSC).

MSSC classifies network flows into applications by message sequences observed during the activities between a pair of two endpoints. The packets exchanged between the two endpoints can be used to derive a sequence based on packet directions and packet sizes. Data exchanged between two endpoints must follow the protocol state machine and the protocol messages defined by involved network applications. MSSC compares the message size sequences (MSSes) of a flow among the representatives of all pre-selected applications to decide which application it belongs to. We also attempted to build a hybrid classifier by combing MSDC and MSSC to provide a balanced solution in terms of classification accuracy and response latency. Based on our analysis and evaluation, MSSC is able to respond by looking only at the very first 15 packets and have a better throughput of 800 Mbps and the hybrid classifier can achieve 723 Mbps.

The rest of this paper is organized as follows. In Section 2, we survey and review relevant researches on network flow classification. Section 3 describes the features that the proposed solutions used to classify network flows. The proposed MSDC and MSSC algorithms are then presented in Section 4. Section 5 provides an analysis for the proposed algorithms. Performance of the proposed solutions is discussed in Section 6. Finally, a conclusion is given in Section 7.

2. Related work

Various statistical-based network flow classification approaches have been proposed in recent years (Gomes et al., 2013). The advantage of these methods is the ability to classify an application without the need to inspect packet payloads. We classify all the approaches into two classes, i.e., the flow-level classification and the session-level classification. The former classifies each flow independently while the latter attempts to group network flows as sessions and then classifies network flows in a session-based manner.

2.1. Flow-level classification

Classifying network flows based on application behaviors is not new. Researchers assume that application behaviors are differentiable and the behaviors can be used to distinguish one application from another. Paxson (1994) modeled and analyzed individual connection characteristics, such as the number of bytes and packets transferred, connection duration, and packet inter-arrival time for different applications. The authors (Este et al., 2009) showed that the amount of information carried by the main packet-level features of Internet traffic flows, such as packet size and inter-arrival time, tends to remain rather constant irrespective of the point of observation and to the capture time.

Hereafter, more works endeavor to classify exclusively network traffic using statistics. They generally consist of two phases: training and classification. A representative model is first built using extracted statistical attributes of flows by learning the inherent structural patterns of datasets and the model is then used to classify network flows. Dewes et al (2003) analyzed and classified different Internet chat traffic using multiple flow characteristics such as flow duration, packet inter-arrival time, packet size, and bytes transferred. Roughan et al. (2004) used nearest neighbor and linear discriminant analysis to map applications to different Quality of Service classes using features such as average packet size, flow duration, bytes per flow, packets per flow, and root mean square packet size. Although Lin et al. (2009) also used the feature of packet size to classify network flows, they paid more attention on those packet sizes with larger proportions in a flow. When the packet size distribution and packet size change cycle of a flow is determined, it is compared against the representatives of all pre-selected applications and the flow is classified as the application having a minimum distance.

Some proposals utilized Machine Learning techniques to classify network traffic. The idea of applying Machine Learning techniques for traffic classification was introduced in (Frank, 1994). Machine Learning techniques are often divided into two phases, i.e., the training phase and the classification phase. Different Machine Learning techniques may perform different and often require distinct parameter configurations.

A number of works adopted probability models to identify and classify network traffic. With training data, probability models are derived for pre-selected applications, and flows are classified as the application having the maximum likelihood. These works assume that the application protocols exhibit consistent and observable structure and patterns in the series of packets they send. Wright et al. (2004) uses the left-right Hidden Markov Models (HMMs) with a large number of states and discrete emission probability distributions to identify TCP connections. Packet sizes and inter-arrival time are

defined as the states of the HMMs. Diainotti et al. (2008) built ergodic HMMs and Gamma-distributed emission probabilities for unidirectional TCP and UDP traffic. Munz et al. (2010) classified TCP connections with help of observable Markov models. Each state reflects certain packet attributes, such as the payload length, the packet direction, and the position within the TCP connection. Peng et al. 2015 applied mutual information and correlation analysis to analyze and find out the feature redundancies, and used machine learning classifiers to verify the classification accuracies. They showed that 5–7 packets are the best packet numbers for early stage traffic identification.

2.2. Session-level classification

A few works analyzed traffic at the session-level. Kannan et al. (2006) used a flow-level trace to derive abstract descriptions of the session-structure for different applications present in the trace. Based on flows' statistical information, Kannan's approach can discover and characterize flow/session causality relationship and further infer applications' internal session structures. Zhang et al. (2013a, 2013b, 2013c, 2015) incorporated flow correlation into the Naïve Bayes (NB) based classifiers with feature discretization, including number of transferred packets, volume of transferred bytes, packet size, and inter-packet time. Flow correlation can be discovered by the 3-tuple heuristic: destination IP, destination port and transport layer protocol. If the flows share the same 3-tuple information, they would be regarded as correlated flows. Besides, the authors employed comprehensively theoretical and empirical study to demonstrate the performance and extend the power to the classification of unknown zero-day application traffic. Besides, Lu et al. (2012) proposed a session-level flow classification algorithm. It measures the packet size distribution to classify traffic, and it can achieve higher accuracy rates and better online speedup effect without human intervention.

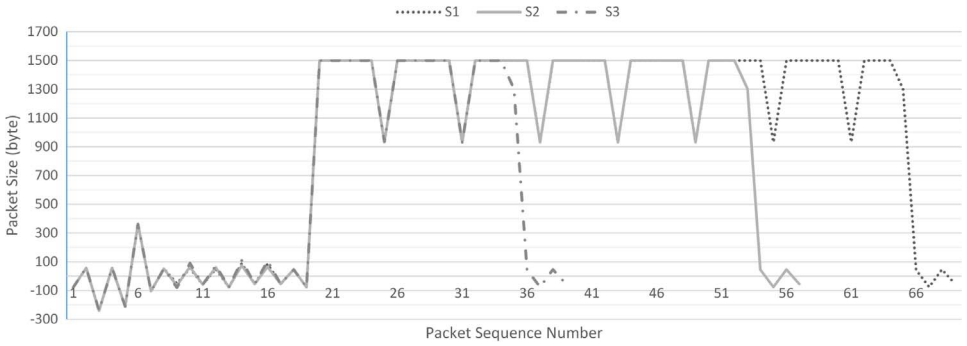
Karagiannis et al. (2005) introduced a traffic classification approach based on the analysis of host behavior. It associates Internet host behavior patterns with one or more applications, and refines the association by heuristics and behavior stratification. However, it cannot classify a single TCP connection because it has to collect aggregated information from multiple flows for each host before it can make a decision.

3. Features

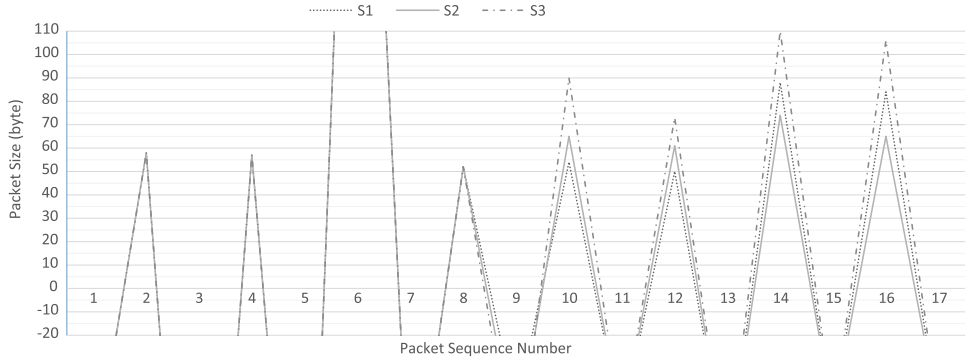
The proposed solution classifies network flows based on message size features of network flows. We assume that network application protocols can be classified as control protocols, data protocols, and control-data mixed protocols. Each application protocol would have several types of protocol messages. The messages carried in a protocol message would have fixed, limited, or similar formats and sizes. In addition, the order of delivering protocol messages would follow the state machine defined by the corresponding application protocol. Based on the assumption, we made several observations to inspect whether protocol messages have the aforementioned characteristics.

3.1. Packet size variation (PSV) and Packet size distribution (PSD)

We first observe packet sizes sent from different network applications. The packet sizes throughout this work means the payload of a packet. We manually capture traces of a single application in a crafted environment to collect the traffic of a specific application. The major advantage of manual collection is that all collected traffic belongs to the same application. Each application is executed in turn, and the generated traffic is recorded when it passes through a network interface. Flows generated by a network application often contain numerous packets of different sizes. Fig. 1(a) shows three instances of SMTP protocol, namely S1, S2, and S3, with different email addresses and mail bodies; Fig. 1(b) shows the first 17 packets of the three instances of Fig. 1(a). The horizontal axis is the packet sequence number and the



(a) Three instances of SMTP protocol



(b) The first 17 packets of the three instances of (a)

Fig. 1. The same type of packet has different sizes of parameters. (a) Three instances of SMTP protocol. (b) The first 17 packets of the three instances of (a).

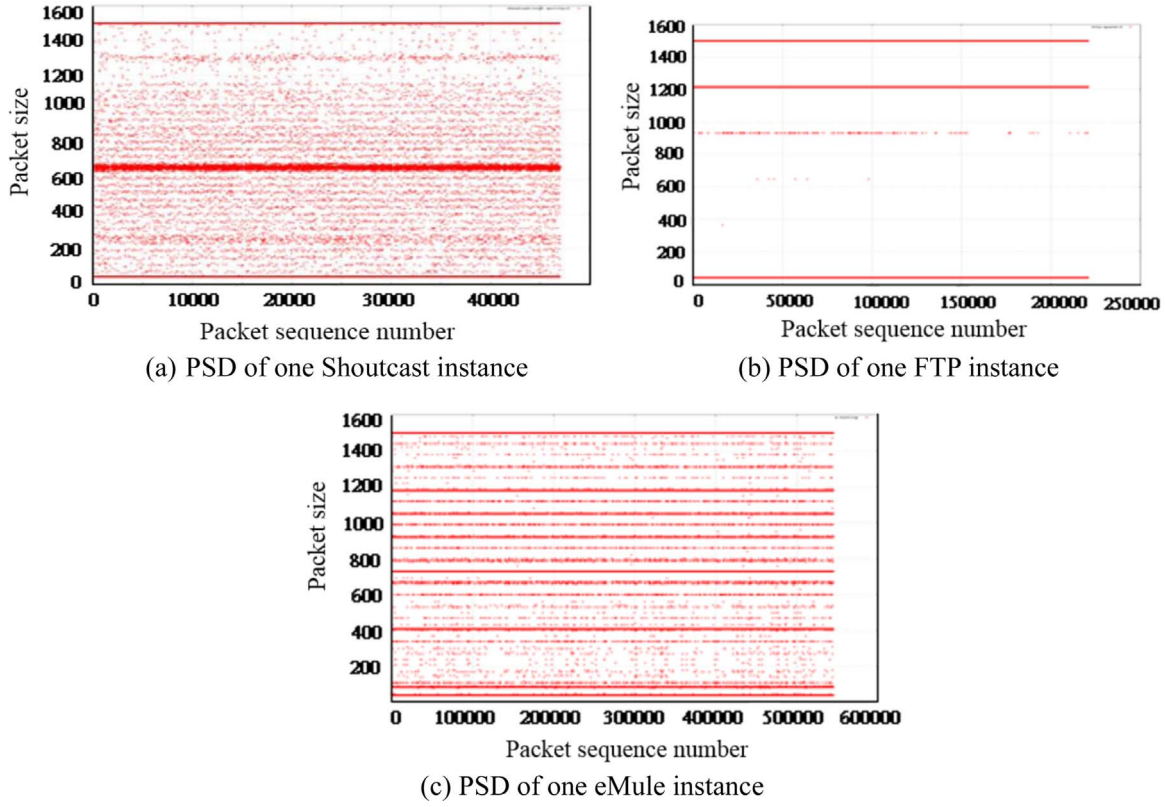


Fig. 2. Different types of applications have distinct PSD. (a) PSD of one Shoutcast instance. (b) PSD of one FTP instance. (c) PSD of one eMule instance.

vertical axis is the size of packets. Packets belong to the same application protocol have similar profiles, as shown in Fig. 1(a). From Fig. 1(b), although packets were generated from different instances, most of them have similar sizes. Those packets with different sizes are commands with different parameters. For example, the "MAIL FROM" command with different senders and recipients provided.

The steps to initiate an SMTP mail transaction are regulated, but the length of parameters carried by commands can be various. If the variation of parameter lengths, which affect packet sizes, can be bounded, packets of the same application would have similar size distribution.

Fig. 2(a), (b), and (c) show the sequences of packet sizes of three different applications, Shoutcast, FTP, and eMule, respectively. The horizontal axis is the packet sequence number and the vertical axis is the corresponding packet size. It shows that different applications have different PSDs. In addition, PSV and PSD can be gathered to provide quantitative information about packet sizes shown within flows generated in a session.

3.2. Message sequence (MS)

We then attempted to identify the relationships between packet sizes and the network application state machine. A network application usually involves with two end points. The two end points can be a client and a server or two peers. Data exchanged between the two ends must follow the protocol state machine and the protocol message format defined by involved network applications. Suppose that we are able to monitor network flows between two end points. Packets of a flow can be used to generate a size sequence based on packet directions and packet sizes. Based on our assumption, a protocol often has limited number of message formats and sizes. Therefore, sequences retrieved from different flows of an application would be similar. Hence, it is possible to differentiate and classify network applications based on the identified packet size sequences.

To verify the assumption, we started from analyzing SMTP. SMTP, initially defined by RFC 821, is a standard for delivering emails on Internet. Suppose that we have a client *C* and a SMTP server S. The client sends e-mails via the server *S* and Fig. 3 shows activities of a sample SMTP session for sending an email.

A lot of messages are involved in the sample session. In the client-server model, messages sent from a client to a server are often called request messages and messages responded from the server are called response messages. In the sample SMTP session, request messages often consist of a command and optionally followed by one or more arguments. A command is often composed of four printable ASCII characters, and the total length of the arguments could be up to 40 printable characters. In contrast, a response message usually consists of a three-digit status indicator, a keyword, and optionally followed by a brief description. We further examined the sizes of request and response messages. Although some message sizes vary greatly, there are still several messages having quite stable sizes. We believe that messages with stable sizes can be used to classify a network application.

Fig. 4 shows the observed packet size sequences for six different applications. The horizontal axis shows the packet sequence numbers and the vertical axis shows the packet sizes. Negative packet sizes indicate that the packets are transmitted in a reverse direction. It is obviously that different applications generate dissimilar message sequences. Readers should notice that many applications adopt a mixed control-data protocol design. This means that the resulted packet size sequences would contain sizes for both control and data messages. Size variations for control messages are usually limited, but variations for data messages are unpredictable.

Fig. 4(a) shows the packet sizes of an SMTP session. At the beginning, several control messages and responses are exchanged and then followed by the mail body if the server allows the client to send an email. We can clearly observe that the packet size variations at the beginning are limited but a great gap is shown when the mail body is being sent.

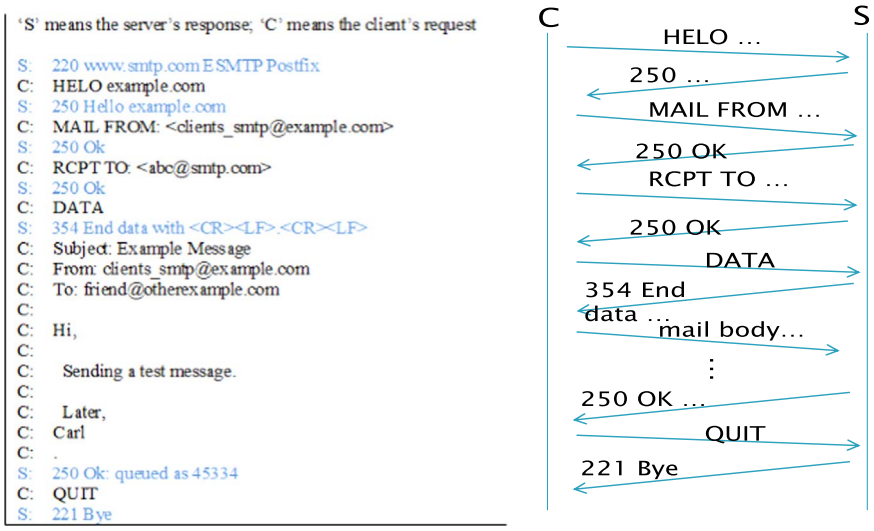


Fig. 3. Messages generated within a sample SMTP session.

3.3. Summary

The above observations show that (1) packet sizes of the same application may be various because of different lengths of parameters, (2) flows of the same application have similar PSDs, but flows of different applications have diverse PSDs, and (3) message sequences of flows sent from an application can be used to identify flows of the application because protocol messages are generated following the state machine of the application.

4. Classification approaches

With the selected features, we propose the MSDC and MSSC solution. The design objectives for the two classifiers are different. The former aims to provide an accurate output while the latter aims to provide a lightweight and real-time solution. In addition, we also combined the two classifiers to build a hybrid solution. Both MSDC and MSSC need to collect application network flows to develop application representatives. There are two ways that can be used: (1) capture all network flows generated while an application is running, and manually filter out irrelevant traces. We describe details of MSDC and MSSC in Sections 4.1 and 4.2, respectively. A brief comparison of MSDC and MSSC is presented in Section 4.3. Section 4.4 illustrates the design of the hybrid solution composed of the two classifiers.

4.1. Message size distribution classifier (MSDC)

MSDC runs in two phases: an offline training phase to obtain application representatives and an online session classification phase. Fig. 5 shows the overview of MSDC. The left block shows the steps of training phase and the right block shows the classification phase, which includes three stages: the flow classification stage, the session grouping stage, and the application arbitration stage.

The goal of the offline training phase is to find out application representatives, which are unique to an application. Hence, the training phase collects a set of network flows and extracts the representatives from the five-tuple information (source IP, source port, destination IP, destination port, transport layer protocol) as well as the PSDs of all captured flows. The more application network flows are collected, the better application representatives can be obtained. After extracting the PSDs of flows, the online flow classification stage compares the flows against obtained application representatives and classifies them into the application having a minimum similarity distance. Meanwhile, the session grouping stage attempts to group flows as a session based on port locality. After the above two stages, each flow is classified as an application and flows having adjacent ports are grouped into the same session. If flows belong to the same session are classified into different applications, the application arbitration stage is invoked to resolve conflicts and all flows in the session are then classified into the application having the largest number of flows in that session.

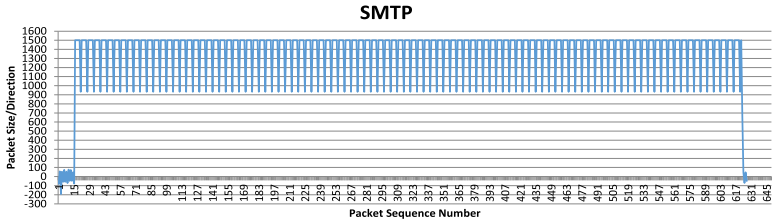
The details of MSDC are introduced in the following sections. Section 4.1.1 describes how packet size sequences of flows are transformed into a representative. The similarity distance metric and the approach used to develop the application representatives are described in Section 4.1.2. Sections 4.1.3 and 4.1.4 introduce the stage of flow classification and session grouping, respectively. The application arbitration stage is finally described in Section 4.1.5.

4.1.1. Flow representation – Dominating sizes (DS) and dominating sizes' proportion (DSP)

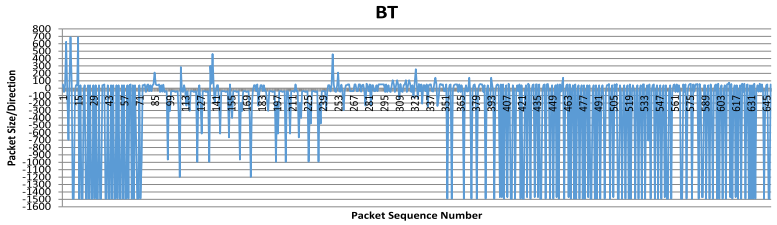
MSDC groups IP packets having the same five-tuple information into a flow. To determine whether two flows are similar, MSDC counts a packet size distribution for packets of each flow. If two flows are similar, their packet size distributions would be similar. To compute a distribution, it is impossible to store all packet sizes of each flow due to limited storage spaces. Thus, only sizes of dominating packets are kept as the feature of the flow.

Assume a number of packets are examined for a flow f. First, packets with payload sizes equal to zero or maximum segment size of the links are discarded. The max segment size can be detected on setting up a connection. Second, the number of valid packets for each distinct packet size is counted and stored as a pair of $(ps_f(i), \text{pro}(ps_f(i)))$, where $ps_f(i)$ is the ith distinct packet size sent in flow f and $\text{pro}(ps_f(i))$ is the proportion of $ps_f(i)$ over the total number of valid packets of flow f. All pairs $(ps_f(i), \text{pro}(ps_f(i)))$ are sorted in descending order by $\text{pro}(ps_f(i))$ and split into two vectors, DS_f and DSP_f, which present the dominating size vector and the corresponding dominating sizes' proportion vector for flow f, respectively. For example, if flow f contains packets of h distinct packet sizes, we would have a set of h pairs $\{(ps_f(1), \text{pro}(ps_f(1))), (ps_f(2), \text{pro}(ps_f(2))), \ldots, (ps_f(h), \text{pro}(ps_f(h)))\}$, and the DS and DSP vectors for the flow f are denoted as $DS_f = < ps_f(1), ps_f(2), \ldots, ps_f(h) >$ and $DSP_f = < \text{pro}(ps_f(1)), \text{pro}(ps_f(2)), \ldots, \text{pro}(ps_f(h)) >$. For the ease of discussion, we use $DS_f(g)$ and $DSP_f(g)$ to indicate the g-th entry in DS_f and DSP_f vector, respectively. The obtained DS and DSP vectors are also called the PSD feature of a flow. Table 1 summarized the notations of MSDC.

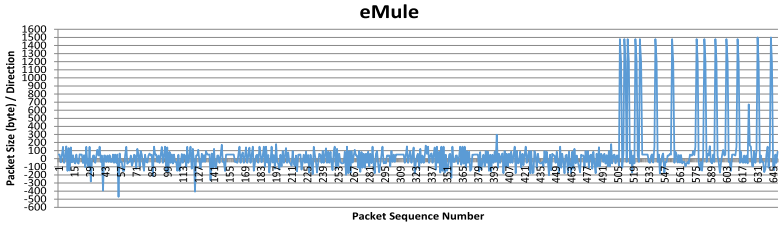
MSDC is designed to extract the unique characteristics of the packet size distribution of an application and uses them to classify network



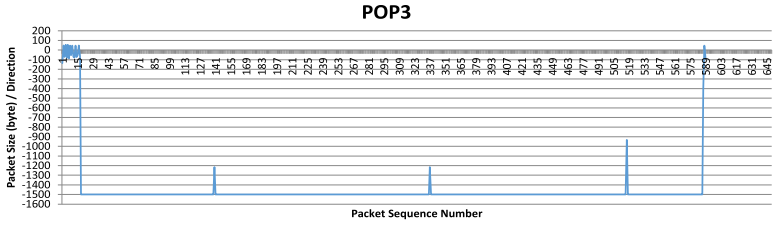
(a) An instance of SMTP protocol



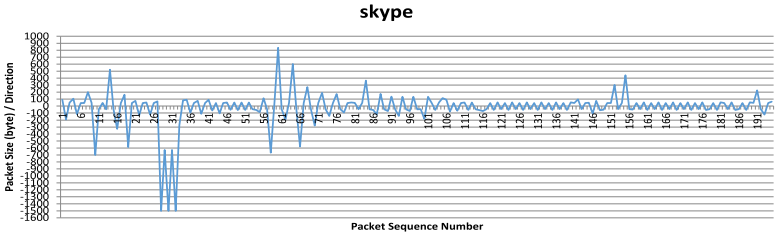
(b) An instance of BitTorrent protocol



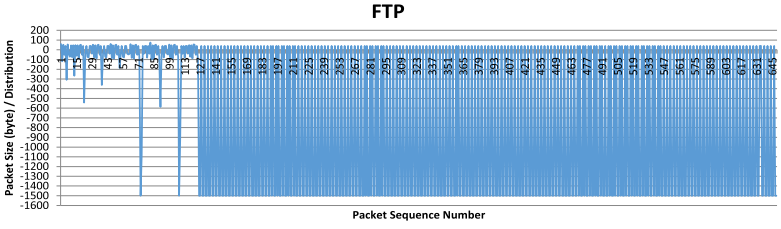
(c) An instance of eMule application



(d) An instance of POP3 protocol



(e) An instance of Skype application



(f) An instance of FTP protocol

Fig. 4. Different message sequences of six applications.

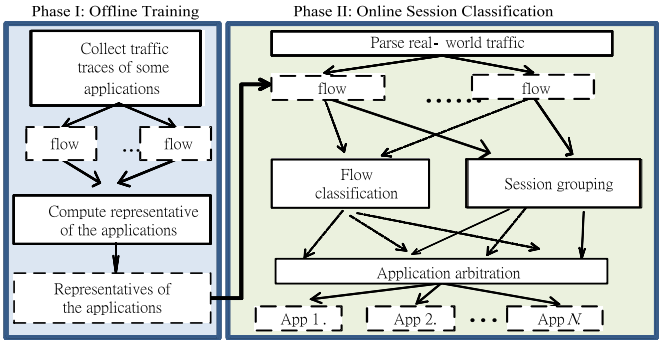


Fig. 5. Components and operation flows of the MSDC solution.

Table 1
Notations used in MSDC.

Notations	Description
$ps_f(i)$, where $0 \leqq ps_f(i) \leqq$ Max segment size	The ith distinct packet size of flow f
$pro(ps_f(i))$, where $0 < pro(ps_f(i)) \leqq 100\%$	The proportion of $ps_f(i)$ over the total number of packets of flow f
$DS_f = \{ps_f(1), ps_f(2), \ldots, ps_f(h)\}$	The corresponding dominating size vector of flow f based on DSP_f
$DSP_f = \{pro(ps_f(1)), pro(ps_f(2)), \ldots, pro(ps_f(h))\}$, where (1) $pro(ps_f(i-1)) \geqq pro(ps_f(i))$, and (2) $\sum_{i=1}^{h} pro(ps_f(i)) \geqq 90\%$	The dominating sizes' proportion vector of flow f
$DS_f(g)$	The gth entry in the DS_f
$DSP_f(g)$	The gth entry in the DSP_f
$f = <DS_f, DSP_f>$	Flow f

flows into applications. The premise of MSDC is that we have no access to the payload of the captured packets. The more common size payload packets we employ, the worst accuracy we have because not only an application can be the final decision. Therefore, the common size payload packets would be removed first to avoid confusion, which is why MSDC discards zero and maximum segment size payload packets.

4.1.2. Application representatives

In order to measure the *similarity* for two different flows, a distance metric is defined. However, special handling must be taken when the lengths of *DS* vectors are not equal. Suppose that there are two distinct flows f_1, f_2, and the number of entries in *DS* vectors of f_1 and f_2 are n and m respectively ($n \geqq m$). The similarity distance between f_1 and f_2 is defined as

$$\text{similarity} = \sum_{g=1}^{m} \sqrt{(DS_{f_1(g)} - DS_{f_2(g)})^2 + (DSP_{f_1(g)} - DSP_{f_2(g)})^2} + \sum_{g=m+1}^{n} \sqrt{DS_{f_1(g)}^2 + DSP_{f_1(g)}^2}. \quad (1)$$

The method used to group related flows is called *automatic clustering*. Based on previous observations, we assume that flows of the same behavior would have similar PSD features. Hence, a *deviation* is defined to tell whether two flows should be grouped together or not. If the PSD distance between two flows is equal to or less than one predefined *tolerant threshold* (*TT*), they are grouped together. Otherwise, they are classified into two different groups.

After classifying all flows into groups, a *representative averaging* (RA) algorithm is used to derive the representative feature from a group of flows. The RA algorithm is applied to each group and averages PSD features of all flows in group G, i.e. $Rep_G = \{\sum (DS_{f_i}/K), \sum (DSP_{f_i}/K)\}$ for $1 \leqq i \leqq K$, where DS_{fi} and DSP_{fi} are the *DS* and *DSP* vectors of flow f_i which contributes to the corresponding feature values within group G, and K is the total number of flows in the group. The final representative for the application is composed of all group-representatives.

4.1.3. Flow classification

To classify a flow, the PSD of the flow is computed first. The PSD is then compared against each application representative derived in the offline training phase using Eq. (1). The flow is finally classified into the application that owns a representative having the shortest distance to the PSD of the flow.

4.1.4. Session grouping

We also attempt to group flows into sessions. We assume flows having the same source IP address and adjacent source port numbers are in the same session. Similarly, flows have the same destination IP addresses and adjacent destination ports are also grouped into the same session. To reduce the possibility of incorrect session grouping, we limit two parameters, port range and flow inter-arrival time, to prevent irrelevant sessions from being grouped together. Port range is used to limit the maximum difference between the lowest port number and the highest port number in a group. Flow inter-arrival time ensures that flows in a grouped have temporal locality.

4.1.5. Application arbitration

When multiple flows are grouped as a session, it is possible that flows within a session are classified into different applications. This is because the flow classification and session grouping is done independently. A classification conflict happens if a grouped session contains flows of two or more different applications. Although an application may show several communication behaviors in a session, flows within this session should be classified into the same application. MSDC currently uses a majority voting strategy to resolve conflicts. If flows of two or more different applications are grouped together, all flows are treated as the application having the largest number of flows in the session.

4.2. Message size sequence classifier (MSSC)

MSSC also runs in two phases: an offline training phase to obtain application representatives and an online flow classification phase. Fig. 6 shows the overview of the MSSC. The left and the right blocks represent the stages of the offline training phase and the online classification phase, respectively.

The offline training phase uses a set of network flows and extracts applications' representatives from the five-tuple information, the size and the direction of each packet, and the message sequences (MSes) of all captured flows. It is common that a protocol message can be sent in a single packet, and hence packet sequences are another form of MSes. After extracting the message size sequences (MSSes) of incoming flows, the online flow classification mechanism compares the flows with preselected application representatives and classifies them into the application having a maximal likelihood.

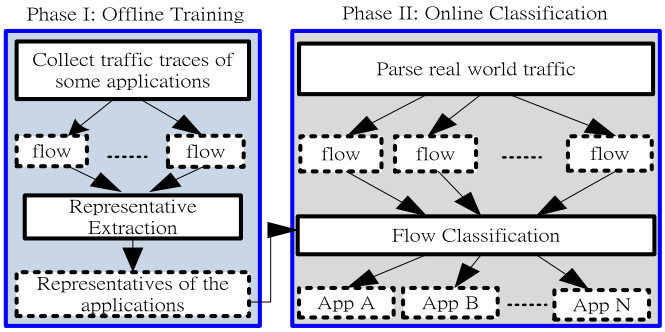


Fig. 6. Components and operation flows of the MSSC solution.

The details of MSSC are introduced in the following sections. Section 4.2.1 describes how a series of messages of flows are transformed into message size sequences (MSSes). The method used to measure the similarity of two different sequences is depicted in Section 4.2.2. Section 4.2.3 explains how to derive application representatives from flows. The flow classification is explained in Section 4.2.4.

4.2.1. Flow representation – Message size sequence (MSS)

We first define how a flow is represented in MSSC. We define a flow as the collections of packets having the same five-tuple information. Since a network flow is bi-direction, packets sent in a reverse direction are considered as the same flow. Assume a number of packets have been collected for a flow. First, packets with full-sized payloads or zero-length payloads are skipped. We pay attention to those packet payload sizes varied between zero-length and max segment size. All applications generate packets of full-sized and zero-sized payloads, and these packets do not benefit classification tasks.

Second, the preserved packets are converted into a form of MSS by extracting the packet sizes and packet directions. To simplify the notation, packet sizes for packets sent in forward directions and reverse directions are denoted as positive and negative numbers, respectively. For example, if a flow f_i contains packets of h distinct packet sizes, we can derive the MSS of flow f_i as $MSS^{f_i} = < \pm ps_1^{f_i}, \pm ps_2^{f_i}, \ldots, \pm ps_h^{f_i} >$, where $ps_h^{f_i}$ is the payload size of the h-th packet of flow f_i and "$\pm$" denotes the packet direction. To ensure that the positive and negative sign would not flip-flop for network flows of the same application, we always use positive numbers for packets having the same direction to the very first packet in a flow. Table 2 summarized the notations used in MSSC.

4.2.2. Similarity computation – Loose Longest Common Subsequence (LLCS)

The similarity measurement used in MSSC is based on the longest common subsequence (LCS) algorithm. It is common that MSSes generated by the same application have partial equal values in the sequences, but these equaled values may be not observed at the same position. It is then straightforward to compare these sequences using the LCS algorithm. In general, the longer the common subsequence we find, the more similar the two sequences are. In other words, given two sequences $X = < x_1, x_2, \ldots, x_m >$ and $Y = < y_1, y_2, \ldots, y_n >$, we expect to find the longest common subsequence $Z = < z_1, z_2, \ldots, z_k >$ for X and Y and the length of Z is used as the degree of similarity for X and Y. The LCS problem is already solved efficiently in the literature (Cormen et al., 2003).

Although there are general LCS solutions available, it may not fit perfectly in the proposed scenario. Based on the packet size variation observed in Section 3.1, packet sizes for the same protocol command message could be diverse because of variable-length parameters. The diversity may affect the performance of classical LCS algorithms because a payload size generated by the same command could have a slight difference. Therefore, we propose a modified version called *Loose Longest Common Subsequence* (LLCS) algorithm to better measure the degree of similarity between two MSSes.

LLCS takes three inputs: two subsequences and one tolerant threshold (*TT*). Suppose the LLCS algorithm compares two flows f_i and f_j. The first two inputs are the MSSes of the two flows, MSS^{f_i} and MSS^{f_j}, where MSS^{f_j}, where $MSS^{f_i} = < \pm ps_1^{f_i}, \pm ps_2^{f_i}, \ldots, \pm ps_m^{f_i} >$ and $MSS^{f_j} = < \pm ps_1^{f_j}, \pm ps_2^{f_j}, \ldots, \pm ps_n^{f_j} >$. If the numerical difference between two different packet sizes $ps_g^{f_i}$ and $ps_h^{f_j}$ from two MSSes, MSS^{f_i} and MSS^{f_j}, is equal to or smaller than *TT*, i.e., $| ps_g^{f_i} - ps_h^{f_j} | \leq TT$, the two payload sizes are considered to be equal. The LLCS algorithm returns: (1) the *longest common size subsequences* (LSS), and (2) the length of LSS to represent the degree of similarity. During the computation process, the LLCS algorithm not only tracks the occurrence order of specific messages but also tolerate limited variation on comparing two messages. Let $X_m = < x_1, x_2, \ldots, x_m >$ and $Y_n = < y_1, y_2, \ldots, y_n >$ be two sequences of length m and n respectively. X_O and Y_O are empty sequences. Let $Z_k = < z_1, z_2, \ldots, z_k >$ be a LSS of length k of X and Y. The LLCS algorithm has the following recursive structure:

1. If $|x_m - y_n| \leqq TT$, $z_k = x_m = y_n$ and Z_{k-1} is a LSS of X_{m-1} and Y_{n-1}.
2. If $|x_m - y_n| > TT$, $z_k \neq x_m$ implies Z is a LSS of X_{m-1} and Y.
3. If $|x_m - y_n| < TT$, $z_k \neq y_n$ implies Z is a LSS of X and Y_{n-1}.

Fig. 7 shows the detailed procedure of the proposed LLCS algorithm. If there is only one LSS Z found by the algorithm, Z is the answer and the length of Z is the similarity for X and Y. If multiple equal-length LSSes are found and the LSSes are *pairwise similar*, i.e., $|x_i - y_i| \leqq TT$ for all the ith value in two sequences X and Y, we further generate a new LSS by setting the i-th value to the *median* of the i-th values from all the sequences. A precise tolerant threshold is then computed for each value in the new LSS and stored in a vector TT'. Last, either the maximum-length LSS Z or the resulted vector TT' are returned.

The concept of LLCS can be illustrated more intuitively with the example of SMTP protocol. Given two independent SMTP protocol handshakes, there are two message sequences both send the following commands in the order of "Mail From" request, "Mail From" response, "RCPT To" request, "RCPT To" response, "DATA" request, and "End DATA" request. The resulted MSSes are {75, −51, 72, −51, 46, −77} and {75, −61, 46, −77} respectively. The longer one with six values can generate up to 2^6 subsequences and Fig. 8 shows one possible diagram of converting subsequences into a state machine. In the state machine, each state is denoted as a request/response message, and MS_i means the observed i-th distinct message size. The LLCS algorithm attempts to find out one state machine that is able to generate both the two MSSes without violating the tolerant threshold *TT*. If the LLCS algorithm succeeds, the two MSSes can be considered as the same. In this example, if *TT* is set to 5, the LLCS algorithm can find the state machine, which is composed of the commands "MAIL From" request, "DATA" request, and "End DATA" request.

4.2.3. Application representatives

In order to collect application-specific traffic, we capture training traces for a single application in a dedicated network. Collecting in a dedicated network guarantees that all collected traces belong to the

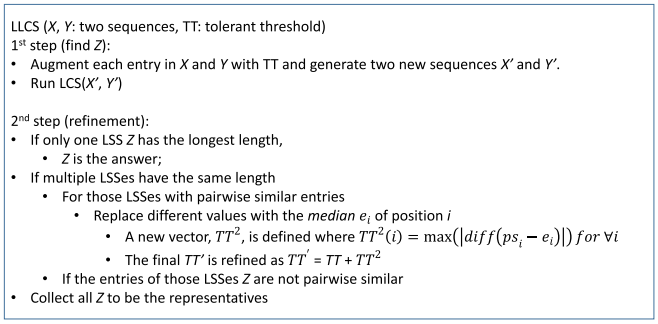


Fig. 7. The LLCS algorithm.

Table 2
Notations used in MSSC.

Notations	Descriptions
$\pm ps_h^{f_i}$, where '+' : $c \rightarrow s$ and '-' : $c \leftarrow s$	The hth distinct packet size of flow f_i
$MSS^{f_i} = < \pm ps_1^{f_i}, \pm ps_2^{f_i}, \ldots, \pm ps_h^{f_i} >$	The message size sequence (MSS) of flow f_i

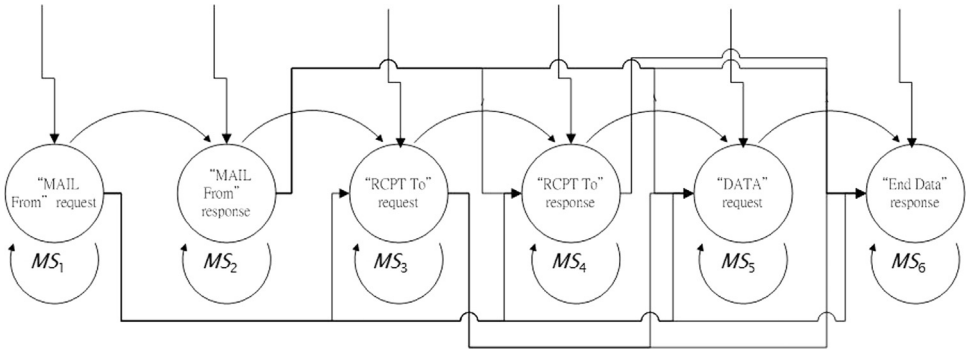


Fig. 8. The example state machine of SMTP protocol converted from the MSS.

same application. Each selected application is executed in turn, and the traces generated are captured when it passes through network interfaces. Likewise, packets with payload sizes equal to zero or maximum segment size are treated as invalid packets and hence omitted. The rest packets are grouped by five-tuple information and then used for generating the MSSes based on the size and the direction of each packet. The MSSes of all flows belong to an application are fed to the LLCS algorithm in pair to find the LSSes. The longest LSS is the representative for the application.

4.2.4. Flow classification

With the obtained representative for each application, a network flow is then classified by comparing its MSS against all application representatives to measure the similarities. If the tolerant threshold for a representative is in the form of a vector, comparing against the ith value in the representative requires checking the ith value in the tolerant threshold vector. For example, suppose we have to compare a MSS for a flow MSS^{f_i}against a representative having a LSS named $MSS^{f_j}$and a vector-based tolerant threshold TT^{f_j}. Given two different packet sizes $ps_g^{f_i}$(gth packet size in MSS^{f_i}) and $ps_h^{f_j}$(hth packet size in MSS^{f_j}), $ps_g^{f_i}$and $ps_h^{f_j}$are considered equal if and only if $|ps_g^{f_i} - ps_h^{f_j}| \le TT_h^{f_j}$. In contrast, if tolerant threshold is a singular value, the threshold value is used to determine whether two packet sizes are equal or not. A flow is always classified into the application that has the longest LLCS.

4.3. Summary

MSSC and MSDC have three major differences: (1) Similarity measurement: MSDC uses Euclidean-like distances while MSSC uses the length of the loosely longest common subsequence. (2) Selection of application representative: MSDC uses the RA algorithm to average all PSD features of flows while MSSC keeps all refined MSSes. (3) Strategy to make a decision: MSDC chooses the application having the minimal distance while MSSC chooses the application having the longest common subsequence. Table 3 summarized the differences.

Table 3
Comparisons between MSDC and MSSC.

	MSDC	MSSC
Similarity	Sum of Euclidean-like distance	The length of the longest common subsequence
Application representatives	$\{\sum_{i=1}^{K}\frac{DS_{f_i}}{K}, \sum_{i=1}^{K}\frac{DSP_{f_i}}{K}\}$	$\{\{LSS_i\}_{i<n}, TT'\}$
Decision	Minimum distance	Maximum length

4.4. The hybrid solution

MSDC provides a good accuracy, but it has a lower throughput because of its statistical computation overheads. In contrast, MSSC attempts to track the application states of flows to make the classifications. As long as the states can be clearly identified, MSSC can quickly make a decision. As a result, MSSC has better throughput. However, MSSC may be not that accurate because there could be short representatives that lead to false positives. Incorrect classifications may happen due to incomplete packet captured or similar protocol states between different applications. Therefore, we attempted to combine MSSC and MSDC and seek for a balanced solution in terms of classification accuracy and performance.

Fig. 9 shows the overview of the hybrid solution. At the beginning, MSDC and MSSC run in parallel and maintain all the required information including the five tuples, the PSD, the DS vector, the DSP vector, and the MSS. MSSC compares the MSS against all trained application representatives and MSDC also computes the similarity distance against all application representatives. A flow that matches more than 90% MSSC representatives of an application is immediately classified into that application. MSSC usually can make a decision in a very short time, but if it failed to make a classification, the decision is later made by MSDC.

5. Modeling and analysis

We discuss the estimated complexity and accuracy of the proposed MSSC and MSDC solutions in this section.

5.1. The worst-case complexity

The design of MSDC and MSSC both contain two phases: the offline training phase and the online classification phase. MSSC attempts to develop application representatives from the MSS within an application session while MSDC attempts to derive representatives from all PSD feature values within the session. The complexities for the two phases are discussed separately.

5.1.1. Offline Training Phase – MSDC vs. MSSC

In the offline training phase, MSSC uses LLCS to find out the application representatives from all trained MSSes. For any two flows, f_i and f_j, suppose that the corresponding MSSes, $X_m = <x_1, x_2, ..., x_m>$, $Y_n = <y_1, y_2, ..., y_n>$, and $Z = <z_1, z_2, ..., z_k>$ is a common subsequence of X and Y. If there exists a strictly increasing sequence $<t_1, t_2, ..., t_k>$ of indices of X such that for all r=1, 2, ..., k, we have $x_{t_r} = z_r$. Z can be derived from LLCS(X, Y) and can be classified into the following four cases,

1. ∅, if i=0 or j=0.
2. LLCS(X_{i-1}, Y_{j-1})∪{x_i}, if $|x_i - y_i| \leqq TT$.
3. LLCS(X_i, Y_{j-1}), if $|x_i - y_i| > TT$ and |LLCS(X_i, Y_{j-1})| > |

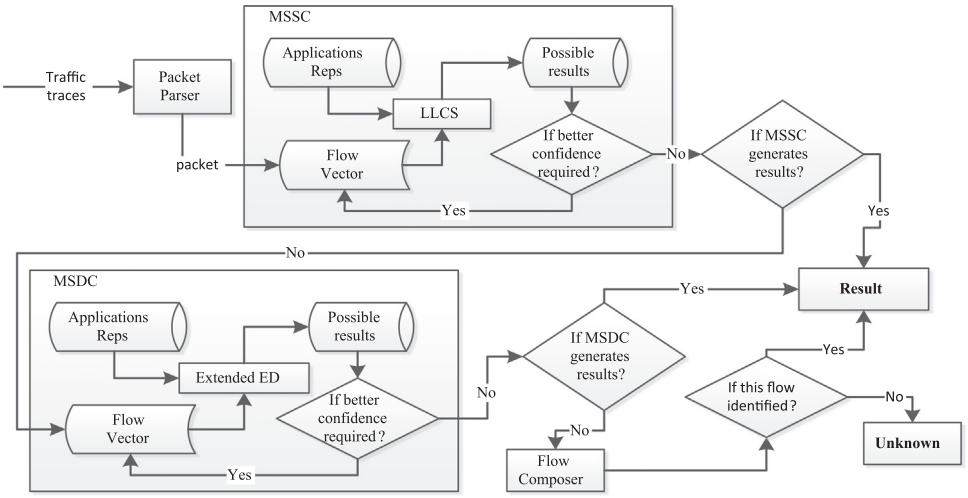


Fig. 9. The hybrid solution comprised of MSSC and MSDC.

$LLCS(X_{i-1}, Y_j)|$.

4. $LLCS(X_{i-1}, Y_j)$, if $|x_i\text{-}y_i| > TT$ and $|LLCS(X_i, Y_{j-1})| < |LLCS(X_{i-1}, Y_j)|$.

The complexity of $LLCS(X_i, Y_j)$ is $O(|f_i|*|f_j|)$ for any two flows f_i and f_j, where $|f_i|$ is the total number of packets in flow f_i. Assume the flow f_i is the flow having the longest length in an application session. The overall worst complexity of the offline training phase of MSSC can be obtained by $O(T)*O(|f_i|)+O(T)*O(T)*O(|f_i|*|f_j|)=O(T^2*|f_i|*|f_j|)$, where T is the total number of flows in the application session, $O(T)*O(|f_i|)$ is the complexity of filtering packets and converting all flows into the MSS format and $O(T)*O(T)*O(|f_i|*|f_j|)$ denotes the total number of combination of any two flows contained.

MSDC uses automatic clustering method to group relevant flows into a group and measure the similarity distance using Euclidean-like distance. For any two distinct flows f_i and f_j with different length in DS and DSP vectors. Suppose that $|f_i|=n$, $|f_j|=m$, and $n \geqq m$, the similarity distance of f_i and f_j can be obtained by Eq. (1). The complexity of Eq. (1) is $O(|f_i|*|f_j|)$ for any two distinct flows f_i and f_j, and the complexity of the RA algorithm is $O(T)*O(|f_i|)$ if flow f_i has the longest length for DS and DSP vectors among all flows in the application session. Finally, the overall worst-case complexity of the offline training phase is $O(T)*O(|f_i|)+O(T)*O(|f_i|log|f_i|)+O(T)*O(T)*O(|f_i|*|f_j|)=O(T^2*|f_i|*|f_j|)$, where T is the total number of flows in an application session, $O(T)*O(|f_i|)$ is the complexity of filtering packets, $O(T)*O(|f_i|log|f_i|)$ is the complexity of sorting packets, and $O(T)*O(T)*O(|f_i|*|f_j|)$ is the complexity of calculating similarity distance. Although the complexities for MSDC and MSSC are equal, MSSC has smaller overhead because MSSC computes similarity distances by using simpler arithmetic operations while MSDC computes Euclidean-like equation with high-dimension vectors.

5.1.2. Online Classification Phase – MSDC vs. MSSC

In the online classification phase, MSSC transforms an incoming flow into a MSS first and then compare it against all application representatives to find out the application sharing the loosely longest common subsequence with the flows. Assume there are total T application representative sequences. For a new flow f_t, the complexity would be $O(|f_t|+|f_i|)$ to verify if the shorter one is the subsequence of the longer one. Therefore, the complexity to verify if flow f_t matches one of the T sequences would be $O(T)*O(|f_t|+|f_i|)$, where $|f_t|$ is the number of entries of flow f_t and f_i is the application representative sequence having the longest length among the T representatives. For c flows waited to be classified, the overall worst-case complexity of online classification of MSSC is $O(c)*O(T)*O(|f_t|+|f_i|)$.

For the online classification phase of MSDC, a new flow is transformed into a set of points in a multi-dimension space and then compared against representatives. The complexity of filtering and sorting packets is $O(|f_i|+|f_i|log|f_i|)$. The flow is recognized as the application that is closest to the flow in terms of similarity distance. Therefore, for a new flow f_t having the number of message sizes, $|f_t|$, the complexity would be $O(|f_t|*|f_i|)$ to measure their similarity using Eq. (1). Therefore, the complexity would be $O(T)*O(|f_t|*|f_i|)$, where f_i is an application representative with the longest length of DS and DSP vectors among all T representatives. For c flows waited to be classified, the overall complexity of online classification of MSDC is $O(c)*O(T)*O(|f_t|*|f_i|)$.

For classification complexities, MSSC has much less overhead because the single comparison overhead of MSSC is less than that of MSDC and hence the cumulated overheads are also smaller than that of MSDC.

5.2. Accuracy simulation

We then estimate the classification accuracy of the proposed solutions. Before a classification is made, both MSSC and MSDC have to be trained with selected applications in a dedicated network. To work with the proposed solution, we assume that (1) the protocol headers of network layer and transport layer protocols are not encrypted, and (2) the packet sizes are not obfuscated. Although it is not necessary to parse the packet payload, we still need to obtain the information of the initiator, the responder, and the payload length. These two assumptions are reasonable because intermediate routers or hosts also need the information to relay packets.

In order to comprehensively evaluate the classification accuracy of MSDC and MSSC, we need application-specific flows to build MSDC and MSSC models respectively. We used hidden Markov model to generate individual application flows. We used the implementation from the General Hidden Markov Model (GHMM) library (GHMM) to generate the models. To work with the GHMM library, application-specific information must be provided first. The information includes a set S of n *states* over time $S=\{s_1, s_2, ..., s_n\}$, a vector I of *initial state probabilities* $I=<i_1, i_2, ..., i_n>$, a *transition probability matrix* $T=\{t_{s_1,s_2}\}$, where t_{s_1,s_2} denotes the transition probability from state s_i to state s_j, and a *emission probability matrix* $E=\{e_{s_i,o_k}\}$, where e_{s_i,o_k} denotes the probability of emitting observation (symbol sequence) o_k given that the model is in state s_i. In our case, each distinct state only has one specific symbol. With the help of the official documents of applications, related RFCs, and traffic analyzers such as Wireshark, we manually captured unencrypted application traces generated from each

selected application in a dedicated environment, identify individual flows, derive the states during the entire application activities, and compute the transition probabilities and emission probabilities from all states.

The initial state and transition probabilities can be estimated from our captured data by the equations

$$i_{s_j}=\frac{f(s_j)}{\sum_{k=1}^{n} f(s_k)} and t_{s_i,s_j}=\frac{f(s_i,\ s_j)}{\sum_{k=1}^{n} f(s_i,\ s_k)}, \quad (2)$$

where $f(s_j)$ is the number of appearances of state i and $f(s_i, s_j)$ denotes the number of transitions from state s_i to state s_j. For each selected application, we develop its HMM application profile and generate application-specific flows, which can be further divided into two different data sets. One is for training phase, and another is for classification. In the following analysis, each application model generated 240 flows, in which 150 flows are used for training and the other 90 flows are used for testing.

5.2.1. The MSDC model

With the MSDC model, each flow is transformed into a spatial representation based on the PSD and regarded as a point of < DS, DSP > in a high-dimensional space. To limit the length of DS and DSP, both DS and DSP are cut to of length k so that argmax $(\sum_{i=1}^{k} pro(ps_i))$ is larger than a user-defined threshold, e.g., 90% in our case. Each flow is regarded as a point in a multi-dimensional space, and the distance between every two flows is measured by Eq. (1).

For developing application representatives, MSDC uses automatic clustering approach, which computes the similarity distance between every pair of flows and groups the flows whose distances are equal to or smaller than the pre-defined tolerant threshold. We assume that flows having similar behavior have smaller similarity distance. The final application representatives are obtained by averaging PSDs of flows within the same group.

To classify a flow, we compute the individual similarity distance between the flow and each application representative obtained in the offline training phase. If an application has more than one representative, the final distance between the flow and the application is the sum of all similarity distances between the flow and each representative in a group. After all similarity distances are obtained, the flow is classified as the application having the minimum similarity distance.

MSDC pays more attention to the *domination* of related packets generated from flows of a session. The more dominant related packets or the more number of packets would disclose more information and help MSDC to make a decision.

5.2.2. The MSSC model

With the MSSC model, each flow is transformed into a message size sequence (MSS). The measurement of similarity between two MSSes is to find out the longest common size subsequences (LSS) between a pair of two MSSes belong to the same application. For developing application representatives, MSSC uses the LLCS algorithm for every two MSSes in a set of manually captured application flows. We assume that the more number of similar packet sizes contained in the compared two MSSes, the more similar the two MSSes are. The MSSes of all flows of an application are fed to LLCS in pair to find the LSSes and remove shorter size sequences if they can be derived from any longer one. The final LSSes are the representative for the application.

To classify a flow, we compute the individual similarity distance between the flow and each application representative obtained in the offline training phase. The flow is classified as the application that shares the longest common subsequence. MSSC pays more attention to the *occurrence order* of packets of flows within a session. Flows are generated according to behavior of an application, and packets of a flow are generated based on the state machine of the application behavior. A longer common subsequence would reveal more information and help MSSC to make a decision.

5.2.3. Summary

Fig. 10 lists the comparisons of classification accuracy between MSSC and MSDC based on simulated network flows. The average accuracy rates are 94.99% and 96.72% respectively.

The results show that MSSC and MSDC have higher accuracy for traditional protocols like FTP, SMTP, and POP3, but lower for P2P applications like BT, eMule, and Skype. This is because traditional network application protocols such are simpler and well documented. In contrast, P2P application protocols are often much more complex and are proprietary. In general, MSDC performs better than MSSC in terms of detection accuracy, especially when there are packet losses or out-of-order deliveries.

6. Experimental study

We further evaluated the performance of the proposed solutions using real traces. Two different data sets were used. Both were captured from the operational instances running in campus networks, not from a traffic generator or a lab. Data sets were split into two parts. One was for training and another was for testing. The training data contained all selected application traffic and it was only used to develop application representatives. The testing data was only used for the purpose of application classification. Table 4 shows the profile of the two data sets of each application.

The rest of this section is organized as follows. The parameters used for evaluating the classifiers are introduced in Section 6.1. Section 6.2 shows the classification accuracy. Section 6.3 shows the detection errors in terms of false positive rates and false negative rates. The comparisons against relevant works are shown in Section 6.4. Section 6.5 shows the throughput and Section 6.6 compares the experimental results and the simulation results.

6.1. Parameters

The parameter tolerant threshold (*TT*) required by LLCS affects the length of common subsequences and the accuracy of application classifications. Fig. 7 in Section 4 shows the detailed procedure of

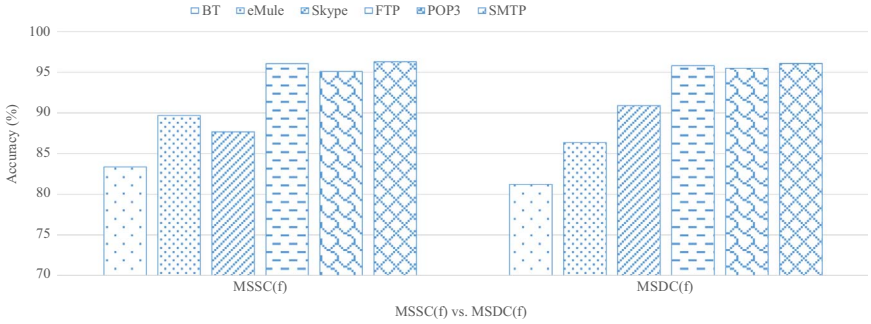


Fig. 10. Classification accuracy: MSDC vs. MSSC.

Table 4
Summarized profile of pre-selected application traces.

Application Name	Application-Layer Protocol	TCP flows (training)	TCP packets (training)	TCP flows (testing)	TCP packets (testing)
BitTorrent	P2P	4172	194,036	2241	104,481
eMule	P2P	18,569	920,951	9994	453,607
Skype	P2P	941	11,943	508	5889
FTP	FTP	1965	361,302	1308	230,997
POP3	POP3	210	24,158	140	15,479
SMTP	SMTP	210	24,407	140	14,335
HTTP	HTTP	150	129,866	100	93,267

LLCS algorithm. A precise tolerant threshold vector of TT′ can be trained and returned. An alternative tolerant threshold vector of different application flow groups can be obtained in the same way. Considering the computation complexity in on-line classification phase, only the effect of a fixed TT was shown in this work. Fig. 11 shows the classification accuracy with various *TT* values.

The horizontal axis is the chosen *TT* value and the vertical axis is the accuracy of traffic classification. From the figure, MSSC has the best classification accuracy when *TT* is 2. Compared to other *TT* values, *TT* of 2 is a moderate choice between being too strict or too loose. Therefore, the value of *TT* is set to 2 for the rest of experiments. Two other parameters of MSDC, *port range* and *flow inter-arrival time*, are set to 4 and 500 s according to suggestions recommended by (Zhang et al., 2013) and (Lu et al., 2012).

6.2. Classification accuracy

This subsection shows the classification accuracy of MSDC and MSSC. In addition to the flow-level classification, we also evaluated the performance for the session-level classification. The accuracy for the flow-level classification indicates the percentage of correctly recognized application flows. For the session-level classification, we further classify an unknown flow into a classified network flow by using the rules introduced in (Lu et al., 2012). Fig. 12 shows the classification accuracy for the six classification configurations: MSSC(f), MSDC(f), Hybrid(f), MSSC(s), MSDC(s), and Hybrid(s). A configuration ending with a "(f)" suffix indicates the use of flow-level classification and a configuration ending with "(s)" suffix indicates the use of session-level classification. We found that some applications have similar accuracies regardless of the use of session grouping and application arbitration. It might be caused by two reasons. First, those applications usually use only a single flow to communicate. Second, the correlations between the flows of those applications are very low.

For traditional protocols like FTP, POP3, and SMTP, MSSC(f) and MSDC(f) have similar classification accuracy. These protocols often have fixed-length requests and responses. For P2P applications and the HTTP protocol, MSDC and MSSC perform differently. For P2P applications, each P2P participant is not only a client but also a server. The operations of a P2P participant can be divided into three phases: 1) server registration; 2) resource indices exchanging; and 3) file exchanging. Different behaviors and peer responses generate distinct message sequences. MSSC is not good at classifying the HTTP protocol. This is because the HTTP protocol does not have fixed-length requests and responses. A lot of variable-length parameters including URL, the headers, the GET or POST parameters, and the response content affect the request and response lengths. Therefore, it is difficult to extract sufficient common subsequences for MSSC to classify the HTTP flows.

Classification accuracy for session-level classifiers is evaluated by the percentage of sessions that are correctly classified. In most cases, the classification accuracy can reach 100% except for the HTTP protocol. For MSSC and MSDC, the session-level classification accuracy for HTTP sessions is 96.4% and 99.98%, respectively. This is because connections of an HTTP session can have scattered port numbers due to redirection requests and incorrect decisions would propagate if most flows within a session were not classified correctly. The accuracy of Hybrid(f) and Hybrid(s) sits in-between MSSC and MSDC because MSDC is used to make the final decision if MSSC is unable to make a decision.

6.3. False positive rates and false negative rates

We discuss the classification error rates in terms of false positive rates and false negative rates. A false positive indicates that a flow is classified into an incorrect application, while a false negative indicates a flow belong to a known application cannot be identified. Classification errors are caused by two major reasons. First, if the representatives of two different applications were very similar, an incorrect classification could be made. Second, if the packets collected for a flow were incomplete, an erroneous classification could be made. For the former one, appropriate representatives that are not similar should be picked up to increase the accuracy of the classification. For the latter one, the classifier is able to collect more packets for a flow to avoid incompleteness situations and hence improve the accuracy. False negatives may happen because an application has very diverse behavior, which are far from those used to train the application representatives.

Table 5 presents false positive rates and false negative rates of each trained applications using MSSC(f), MSDC(f), MSSC(s), and MSDC(s) configurations. From Table 5, it is obvious that when flows are grouped into sessions, the false positive rates can be dramatically reduced because incorrect classification can be fixed by application arbitration.

6.4. Compared with relevant solutions

We also compared the proposed solutions against relevant flow classification techniques, including (GHMM; Bernaille et al., 2006), and (L7-filter). They use similar features including packet train lengths and packet sizes. L7-filter (L7-filter) is a traffic classifier extension to work with Linux Netfilter. It classifies packets based on packet payloads. The version of L7-filter we used is v2.22. By default, L7-

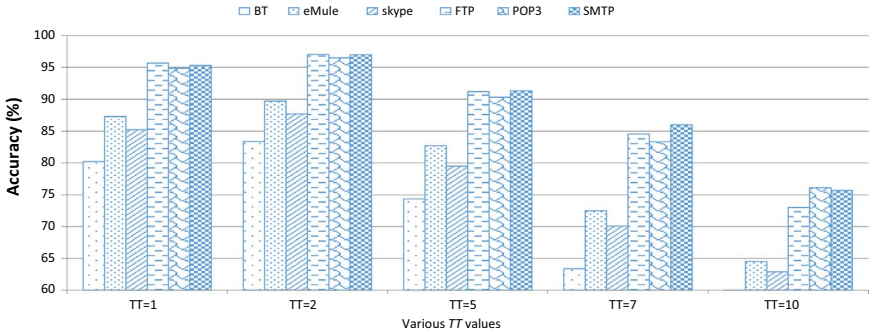


Fig. 11. Accuracy rates for different *TT*.

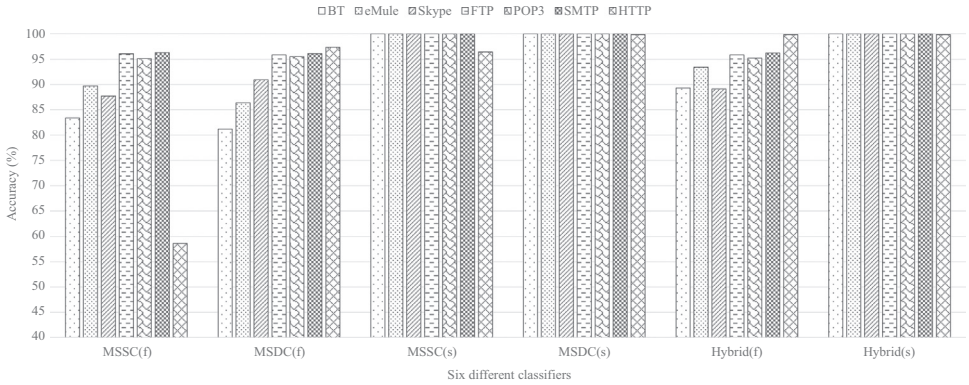


Fig. 12. Accuracy rates for six classification methods.

Table 5
False positive and false negative rates.

	MSSC (f)		MSDC (f)		MSSC (s)		MSDC (s)	
Application	FP (%)	FN (%)	FP (%)	FN (%)	FP (%)	FN (%)	FP (%)	FN (%)
BitTorrent	16.62	0	20	0	0	0	0	0
eMule	10.3	0	13.64	0	0	0	0	0
Skype	12.31	0	9.09	0	0	0	0	0
FTP	2.95	0	5	0	0	0	0	0
POP3	3.5	0	4.8	0	0	0	0	0
SMTP	3	0	3.9	0	0	0	0	0
HTTP	41.4	0	2.67	0	3.6	0	0.16	0

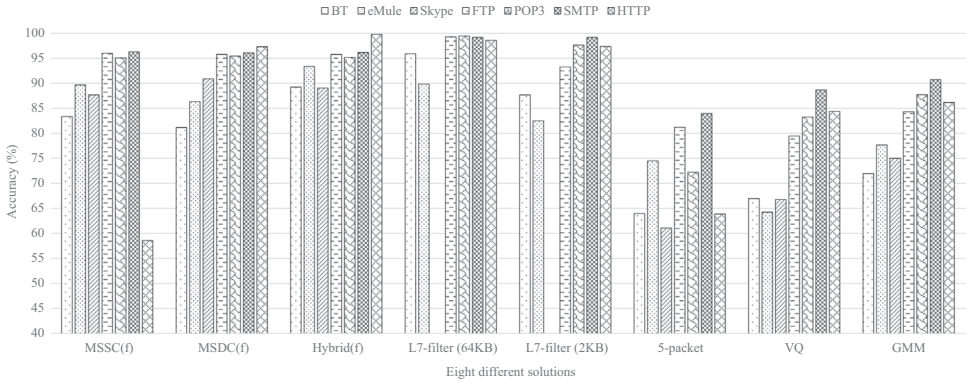


Fig. 13. Accuracy rates for different classification methods.

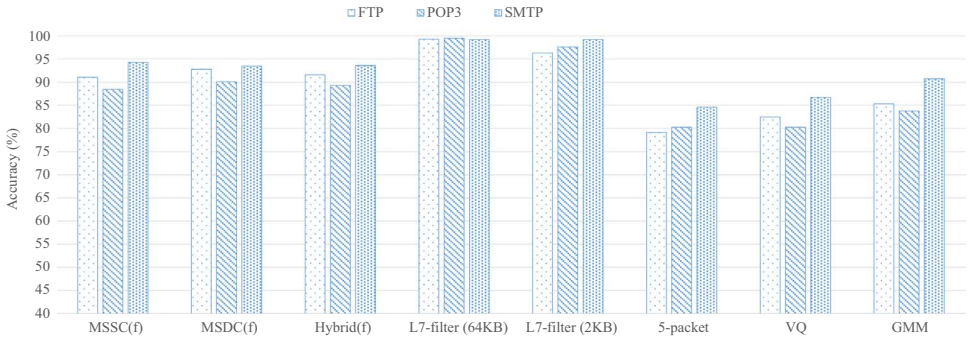


Fig. 14. Accuracy rates for all methods using DARPA data sets.

filter looks at the very first 2KB payloads of each flow. Users can modify the "maxdatalen" configuration on loading the module. However, a longer inspection length could degrade the overall performance. Bernaille et al. (2006) used the size of the first five data packets of each TCP flows to identify the application associated with a TCP flow. Divakaran et al., (2006) used clustering techniques based on vector quantization (VQ) and Gaussian mixture models (GMM) to identify network flows. The two techniques are labeled as VQ and GMM respectively in the figure. Except our hybrid solution, all classifiers are run in flow-level classification mode. Fig. 13 shows the results for all the compared solutions.

Note that we used two different configurations for L7-filter. One is

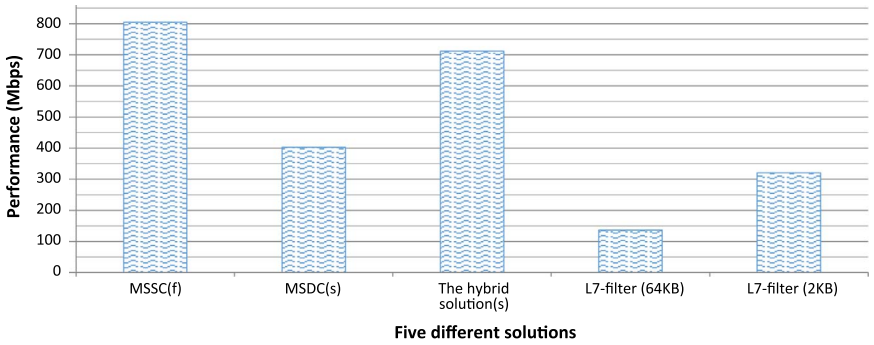


Fig. 15. Overall throughputs for five classification methods.

Table 6
Latency results for three methods.

Application	Latency (in packets)		
	MSSC (f)	MSDC (s)	L7-filter
BitTorrent	≦8	≦ 300	≦10 (except Skype)
eMule	≦ 8		
Skype	≦ 8		
FTP	≦15		
POP3	≦ 8		
SMTP	≦ 8		

configured to use a longer inspection length up to 64KB and another is configured to use a shorter inspection length up to only 2KB. It is obviously that a longer inspection length would get better classification accuracy. Readers should also note that in Fig. 13, L7-filter could not classify Skype flows because our evaluated Skype flows were transported by TCP protocol, which is not supported by L7-filter. Among unencrypted applications, L7-filter with a 64KB inspection length has the best results. All the solutions work better with traditional protocols than with P2P protocols. Except L7-filter with 64 KB data length, MSSC(f), MSDC(f), and the hybrid solutions have better accuracy rates than the others.

6.5. DARPA trace

We also use public data sets from DARPA (DARPA) to benchmark the performances of our solutions. The application traffic captured between the first and the third weeks were used as training data, and the application traffic captured between the fourth and the fifth weeks were used as testing data. Three applications were chosen in this experiment, including FTP, POP3, and SMTP protocols. Fig. 14 shows the results for all the compared solutions using DARPA data sets. The results showed that MSSC(f), MSDC(f) and Hybrid(f) all get better accuracies.

6.6. Throughput

Fig. 15 shows the overall throughput for five different classification solutions. MSSC has the best performance than the other solutions and the overall throughput exceeds 800 Mbps. The hybrid solution performed slightly slower than MSSC because the performance of the hybrid solution is counted in two parts: one is the classification results both agreed by MSDC(s) and MSSC(f), and the other is the classification results given by MSDC(s) if MSSC(f) is unable to classify or makes a decision different from MSDC(s). L7-filter with a 2 KB inspection length performs faster than that with a 64 KB inspection length because the lengths of inspected data are much shorter for the 2 KB case.

Table 6 shows the latency comparisons for MSSC(f), MSDC(s), and L7-filter. The latency is measured by using the number of inspected packets before a correct decision can be made. All packet payloads of the evaluated network flows are not encrypted because L7-filter is not able to work with encrypted payloads. Both MSSC(f) and L7-filter can make a decision within 10 packets. However, MSDC(s) did not perform well because it has to collect sufficient numbers of packets to compute the statistical features before it can make a decision.

6.7. Experimental results vs. simulation results

Fig. 16 shows the comparisons of experimental results and simulation results. Experimental results are labeled with an “E-” prefix and simulation results are labeled with an “S-” prefix. From the comparisons, we can conclude that: (1) classifying traditional protocols has higher accuracy than classifying P2P applications for both experimental and simulation results; (2) the simulation results are consistent with the experimental results, which means we can have an easier way to evaluate the classification accuracy of the proposed approaches without conducting real experiments.

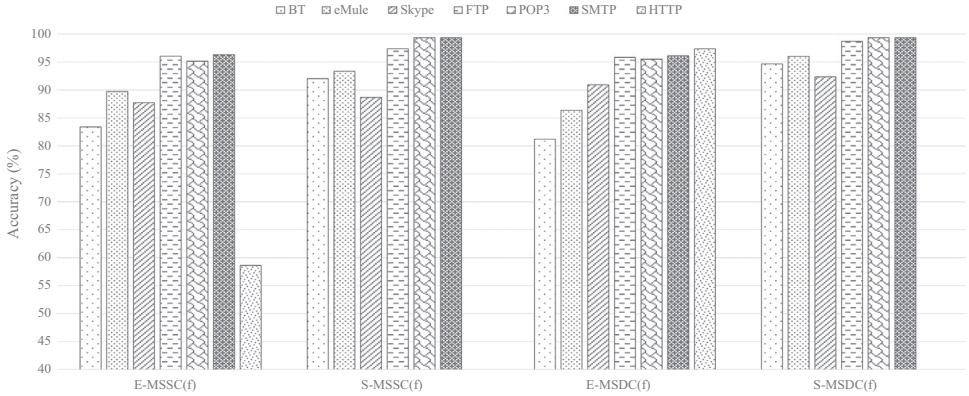


Fig. 16. Comparisons among experiments and simulations.

7. Conclusion

We proposed MSDC, MSSC, and a hybrid solution to classify network flows into their corresponding applications. The solutions are built based on features retrieved from packet sizes and port locality. MSDC solution is able to make a decision by inspecting less than 300 packets and achieve a high session-level classification accuracy of 99.98%. However, it has a relatively lower throughput of 400 Mbps on a commodity PC. To shorten the classification latency, we also explored the possibility of classifying based on application protocol states, which is inferred from the message size sequences. Although the revised MSSC solution has 5% loss on classification accuracy, it is able to make a decision by inspecting less than 15 packets and achieve a doubled throughput of 800 Mbps.

The hybrid solution combines MSSC and MSDC solutions and provide a balanced solution for flow classification. A flow classification is made by MSSC solution at first. If MSSC solution is not able to make a decision, the classification is postponed until MSDC is able to make a decision. The hybrid solution therefore achieves a classification accuracy of 99.97% and an overall system throughput of 723 Mbps. We believe that the proposed three solutions are able to help a network administrator to well manage a complex network in a timely and accurate manner.

References

Azzouna, N.B., Guillemin, F., 2003. Analysis of ADSL traffic on an IP backbone link. In: Proceedings of the IEEE global Telecommunication Conference (GLOBECOM'03), pp. 3742–3746.

Bernaille, L., Teixeira, R., Akodjenou, I., Soule, A., Salamatian, K., 2006. Traffic classification on the fly. Proc. ACM SIGCOMM Comput. Commun. Rev. 36 (2), 23–26.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C., 2003. Introd. Algorithms, 350–355.

DARPA Intrusion Detection Evaluation Data Set, Available at: ⟨http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/1999data.html⟩.

Dewes, C., Wichmann, A., Feldmann, A., 2003. An analysis of internet chat systems. In: Proceedings Third ACM SIGCOMM Conference Internet Measurement (IMC'03), pp. 51–64.

Diainotti, A., de Donato, W., Pescape, A., Salvo Rossi , P., 2008. Classification of network traffic via packet-level hidden markov models. In: Proceedings of IEEE Global Telecommunication Conference (GLOBECOM'08), pp. 1–5.

Divakaran, D.M., Murthy, H.A., Gonsalves, T.A., 2006. Traffic modeling and classification using packet train length and packet train size. In: Proceedings Sixth IEEE Conference IP Operartional Management (IPOM'06), pp. 1–12.

Este, A., Gringoli, F., Salgarelli, L., 2009. On the stability of the information carried by traffic flow features at the packet level. In: Proceedings of the ACM SIGCOMM Computer Communication Reviews, 13–18.

Frank, J., 1994. Machine learning and intrusion detection: current and future. In: Proceedings of the 17th National Computer Security Conference.

GHMM General Hidden Markov Model Library, Available at ⟨http://bioinformatics.rutgers.edu/Software/GHMM/⟩

Gomes, Joao V., Inacio, Pedro R.M., Pereira, Manuela, Freire, Mario M., 2013. Detection and classification of peer-to-peer traffic: a survey. ACM Comput. Surv. (CSUR) 45 (3).

Kannan, J., Jung J., Paxson, V., Koksal C.E., 2006. Semi-automated discovery of application session structure. In: Proceedings of the Sixth ACM SIGCOMM Conference on Internet Measurement (IMC'06), pp. 119–132.

Karagiannis, T., Papagiannaki, K., Faloutsos, M., 2005. BLINC: multilevel traffic classification in the dark. In: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'05), pp. 229–240.

L7-filter, Available at ⟨http://l7-filter.clearfoundation.com⟩.

Lin, Y.-D., Lu, C.-N., Lai, Y.-C., Peng, W.-H., Lin, P.-C., 2009. Application classification using packet size distribution and port association. J. Netw. Comput. Appl. 32 (5), 1023–1030.

Lu, C.-N., Huang, C.-Y., Lin, Y.-D., Lai, Y.-C., 2012. Session level flow classification by packet size distribution and session grouping. Comput. Netw. 56 (1), 260–272.

Moore, A., Zuev, D., 2005. Internet traffic classification using bayesian analysis techniques. In: Proceedings of the ACM SIGMETRICS Conference on Measurement Modeling and Computer System (SIGMETRICS'05), 55–60.

Munz, G., Dai, H., Braun, L., Carle, G., 2010. TCP traffic classification using markov models. [April]Proceedings Second Conference Traffic Monitoring and Analaysis (TMA'10), 127–140.

Paxson, V., 1994. Empirically derived analytic models of wide-area TCP connections. IEEE/ACM Trans. Netwon (TON) 2 (4), 316–336.

Peng, L., Yang, B., Chen, Y., 2015. Effective packet number for early stage internet traffic identification. Neurocomputing 156, 252–267.

Roughan, M., Sen, S., Spatscheck, O., Duffield, N., 2004. Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification. In: Proceedings of the Fourth ACM SIGCOMM Conference on Internet Measurement (IMC'04), 135–148.

Sen, S., Spatscheck, O., Wang, D., 2004. Accurate, scalable in-network identification of P2P traffic using application signatures. In: Proceedings of the 13th Conference on World Wide Web (WWW'04), 512–521.

Wright, C., Monrose, F., Masson, G.M., 2004. HMM profiles for network traffic classification. In: Proceedings of the ACM Workshop Visual Data Mining Computer Security (VizSEC/DMSEC'04), 9–15.

Zhang, J., Xiang, Y., Wang, Y., Zhou, W., Xiang, Y., Guan, Y., 2013. Network traffic classification using correlation information. IEEE Trans. Parallel Distrib. Syst. 24 (1).

Zhang, J., Chen, C., Xiang, Y., Zhou, W., Xiang, Y., 2013. Internet traffic classification by aggregating correlated naïve bayes predictions. IEEE Trans. Inf. Forensics Secur. 8 (1), 5–15.

Zhang, J., Chen, C., Xiang, Y., Zhou, W., Vasilakos, A.V., 2013. An effective network traffic classification method with unknown flow detection. IEEE Trans. Netw. Serv. Manag. 10 (2).

Zhang, J., Chen, X., Xiang, Y., Zhou, W., Wu, J., 2015. Robust network traffic classification. IEEE/ACM Trans. Netw. 23 (4), 1257–1270.