Sequential Message Characterization for Early Classification of Encrypted Internet Traffic

Wenxiong Chen, *Member, IEEE*, Feng Lyu, *Member, IEEE*, Fan Wu, *Member, IEEE*, Peng Yang, *Member, IEEE*, Guangtao Xue, *Member, IEEE*, and Minglu Li, *Senior Member, IEEE*

***Abstract*—Classifying Internet traffic is critical to many network management tasks, including malicious attack detection, usage monitoring, load balancing, etc. As current traffic packets are often transmitted with encryption, at randomized port numbers, and under highly dynamic network conditions, traditional approaches such as port mapping, deep packet inspection, and statistical analysis are no longer effective. In this paper, we first collect extensive traffic flows at the exit router of a university and label them into various source applications. After extracting the *message* (consisting of multiple consecutive TCP packets) sequence for all collected traffic flows, we find that each application type has *distinct sequential message features*. By leveraging the message sequential feature, we develop a system, named *SMC* (Sequential Message Characterization), which can perform online traffic classification with the sequential size information of a few message segments. In *SMC*, after confirming the long-term dependency among message segments, we create a Long Short-Term Memory (LSTM) neural network to conduct deep learning on message size sequence, and then build a multi-classifier to classify traffic types based on the probability profiles output by deep LSTM models. Extensive experiments are conducted and results demonstrate that the proposed *SMC* can achieve 97% of classification accuracy on average. Meanwhile, with as few as 6 pieces of message size information as input, *SMC* enables early online traffic classification especially for heavy-traffic flows with over 35 message segments in median.**

***Index Terms*—Deep LSTM, early online classification, internet traffic, sequential message characterization, traffic analytics.**

I. Introduction

ACCURATE identification and categorization of Internet traffic is a source of intelligence for many network management tasks such as diagnostic monitoring, flow prioritization, and traffic shaping/policing [1]–[6]. For instance, with the traffic classification, the network administrator is able to identify the problem of internal network, e.g., whether a certain type of multimedia traffics occupy the most network bandwidth and block the channel. The network service provider can analyze the traffic composition by categorizing them into classes, to benefit load balance strategies. Besides, with the aid of traffic classification, the firewall can detect malicious attacks proactively [7].

However, the current challenges to accurately classify real-time Internet traffic are as follows: First, detection time and classification accuracy are competing objectives. To finish the classification rapidly, only limited input features of the traffic flow can be extracted, which can hardly guarantee the classification accuracy. In addition, to improve accurate traffic classification, extracting multi-dimensional features including flow duration, arriving times of packets, etc., is essential, inevitably prolonging the finish time for detection. Second, Internet traffic flows can be very complex and are normally untraceable as there are so many source applications and network conditions are dramatically dynamic and uncertain [8]–[12]. Different source applications show interlaced transmission features while dynamic network conditions further render these features labile. Hence, the robustness of classification algorithm is as important as the accurate performance, which has to be efficacious under all kinds of network conditions. Third, to verify the efficiency of classification algorithm, adequate labeled Internet traffic flows which cover different source applications with sufficiently enough samples, are essential. Existing accessible data sets were collected many years ago (more then ten years), which may be obsolete and hard to represent hodiernal network behaviors.

Many traffic classification approaches have been proposed, which however may be no longer effective due the revolution of modern Internet applications. Particularly, Karagiannis *et al.* [13] proposed a port matching-based approach, which was efficient in early Internet days when there is a one-to-one match between the transport layer port number and source application. However, with the emergence of P2P technology, there is a rapid increase in the Internet application number, and then port numbers are misused by various applications to avoid detection, which significantly degrades the performance of port-based methods. Therefore, some researchers then proposed methods that rely on key characteristics of traffic flows as alternatives. Such methods are termed as deep packet inspection (DPI) [14]–[16], in which the first few TCP packets of the flow are investigated by searching the keywords or specific pattern in the packet

Manuscript received November 9, 2020; revised January 20, 2021 and February 25, 2021; accepted February 28, 2021. Date of publication March 5, 2021; date of current version May 5, 2021. This work was supported by the National Natural Science Foundation of China under Grants 62002389 and 62001180. The review of this article was coordinated by Prof. Igor Bisio. *(Corresponding author: Feng Lyu.)*

Wenxiong Chen is with the Business School, Central South University, Changsha 410083, China, and also with the Business School, Hunan Normal University, Changsha 410081, China (e-mail: chenwx@csu.edu.cn).

Feng Lyu is with the School of Computer Science and Engineering, Central South University, Changsha 410083, China (e-mail: fenglyu@csu.edu.cn).

Fan Wu is with the Department of Computer Science and Technology, Tsinghua University, Beijing 10084, China (e-mail: wufancs@tsinghua.edu.cn).

Peng Yang is with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: yangpeng@hust.edu.cn).

Guangtao Xue and Minglu Li are with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: xue-gt@cs.sjtu.edu.cn; li-ml@cs.sjtu.edu.cn).

Digital Object Identifier 10.1109/TVT.2021.3063738

payload. Mostly, the DPI-based techniques can be accurate and fine-grained. However, as data privacy becomes an increasing concern for Internet users, the transmitted Internet contents are usually encrypted by Transport Layer Security (TLS) or Secure Sockets Layer (SSL) protocol, restraining the applicability of DPI-techniques. On the one hand, the huge overhead and the cumbersome packet processing for keyword extraction degrade its prevalence. To this end, statistics-based algorithms [17]–[19] are then proposed, which try to extract multi-dimensional application features with the objective of generating a classifier. Those features include the flow duration, packet inter-arrival time, throughput, etc. However, the above time-related metrics may not be consistent and stable enough to serve as classifiers when the network dynamics, which has been verified in many previous works [20]. Another disadvantage of statistics-based methods is that it cannot support early classification, because features such as the mean packet size can only be obtained after the end of the process. Therefore, design a new effective traffic classification method that can accurately and timely classify various source applications even under dynamic Internet conditions, which is essential for the network management of perception applications.

In this paper, from the exit router of a university, we first collect extensive traffics (TCP-level) flows, and then label them into several source applications, i.e., Video, WeChat, SSH, HTTP, SMTP, etc., where most of network application types are covered. In addition, we discover that each traffic flow can be divided into consecutive *message* segments (i.e., message sequence), each representing a relatively independent piece of content transmitted between the server and client. For instance, for a large-size content, the server would deliver it by multiple message segments (containing multiple TCP packets) due to the limitation of Maximum Transmission Unit (MTU), and at the end of each message segment, the server could push advertisements expediently. After receiving a message segment, the receiver sends an ACK message segment, which acts as a heartbeat in response to the received message segment. For all traffic flows, after extracting the message sequence from underlying TCP packets, we then conduct extensive data analytics on message sequences and surprisingly find that there are very distinct "sequential features" from different source applications. For instance, large-size messages from the server interacting with small-size messages from the client (and vice versa) are frequently observed in video or P2P traffic flows whereas rarely appear in HTTP and other types of traffic flows.

Based on the extracted sequential features, we develop an online traffic classification system, named as *SMC* (i.e., Sequential Message Characterization), which aims at classifying traffic flows in real time by distinguishing massage sequence features. Specifically, in *SMC*, at offline training stage, we construct a Long Short Term Memory (LSTM) neural network [21], [22] to conduct deep learning on the message size sequence for all types of traffic flow, based on which unique LSTM-traffic models corresponding to distinct traffic types are generated, forming a *model forest*. Due to the advantages of the LSTM model in processing and predicting important events, even after a long time interval, we have disclosed the *long-term dependence* among message segments. Compared with the Hidden Markov Model (HMM), which can only 'remember' the states within fixed time-intervals, the LSTM neural network is more suitable for training message size sequence as it can 'remember' correlations over arbitrary time-intervals. At online traffic classification stage, given the input of message size sequence, the model forest can output the probability profiles, indicating the corresponding likelihood of the traffic belonging to different source applications. It can help filter out the classification result. Based on the real-world Internet traffic flows that we have collected and labeled, we conduct extensive experiments to evaluate the performance of *SMC* under the cross validation scheme with varying the training-data size, and experiment results demonstrate the efficacy of *SMC*. Particularly, *SMC* can achieve up to 98.9% accuracy (97% on average) on per-flow classification, outperforming the HMM-based approach significantly (with accuracy no more than 60%). In addition, *SMC* is able to classify the traffic flow at an early stage when the flow stars, as it requires as few as 6 successfully extracted message-size information, which is especially valuable for long-sequence multimedia traffic classification, such as Video and P2P whose median message sequence length normally reaches over 35 message segments.

We highlight the main contributions as follows.

- We collect extensive Internet traffic flows at a public router, and have labeled them into source applications via recruiting volunteers which provides a solid ground truth for traffic analytics and performance evaluation.
- After extracting message segments for each traffic flow, we conduct data analytics on message sizes, and disclose the unique message sequential feature for each type of traffic flow, which can be utilized as a "fingerprint" to distinguish them from each other.
- After confirming the long-term dependence among message segments, we create a LSTM neural network to conduct deep learning on the message size sequence. Based on the model, we develop and implement an online traffic classification system, named *SMC*, to classify encrypted Internet traffic flow at an early stage (long-sequence multimedia traffic) after the traffic flow generates. Extensive experiments are conducted and results demonstrate the superior performance of *SMC*.

The remainder of this paper is organized as follows. In Section II, we collect and label Internet traffic flows. Then, we extract message and conduct empirical studies on message sequences in Section III. Section IV elaborates on the design of *SMC*. In Section V, we present the performance evaluation for the proposed scheme. We review some related works in Section VI. Finally, we conclude and direct our future work in Section VII.

II. Collecting and Labeling Traffic Flows

A. Collecting Various Types of Traffic Flow

To our best knowledge, most available data sets are collected more than ten years ago [15], [23]. However, as the Internet traffic increases rapidly, previous works that work well on such data sets, cannot guarantee satisfied classification accuracy for

current traffic flows. To cope with the limitation, it is essential to collect sufficient Internet traffic flows before delving into the classification algorithm design. In this paper, we first collect traffic flows from an exit router of Shanghai Maritime University, where the traffic rate at peak time can reach over 7 GB/s. As packets frequently drop, the often-used traffic monitor (i.e., Tcpdump and Wireshark), fail to deal with such high-throughput data traffic. To this end, we have developed *Noff* [24], which is an extendible parallel tool for high-performance network traffic monitoring, to capture all packets from the router. The details of *Noff* are beyond the scope of this paper. With the collected packets, we then recompose them into traffic flows.

The traffic collection lasts for three days, based on which we extract complete traffic flows of SSH, SMTP, Remote Desktop, HTTP, and WeChat. Note that, in the university network, there is a load balancing scheme to balance the traffic load at each router. For large-size traffic, the chance is enlarged for these packets to be routed over multiple routers since the unbalanced load condition can be more likely to be triggered. Note that, for our data collection, incomplete traffic flows are dropped by checking the packet serial number in TCP/IP protocol. For multimedia streaming, as they are usually quite large and their packets are likely to be routed via different router nodes for satisfied performance, it can hardly achieve complete streaming from one router node. Therefore, three volunteers are recruited to collect P2P and Video traffic flows. Particularly, to collect P2P traffic flows, each volunteer downloads 10 movies via the BitTorrent tool, and totally 200 G P2P files are downloaded, where all packets are catched by Tcpdump. To collect Video traffic, each volunteer is required to watch videos on popular video websites, e.g., Iqiyi and Youku, for five hours every day (lasting five days), where network packets are collected by Tcpdump in the meantime. For each video, the duration ranges from 5 to 10 minutes, and the total duration adds up to 4000 minutes.

B. Labeling Traffic Flow Samples

With the collected traffic flows, we then ought to label them according to their source applications. The following two steps are executed to label each traffic flow type. Particularly, we first filter traffic flows by well-known ports (e.g., 22, 80, and 443), the goal of which is to significantly reduce the number of candidate flows. Then, we compare the first few packets of each flow with the proper format of source applications, to label each traffic flow type. For instance, to label the SSH flows, we first extract and reassemble TCP flows of port 22 from the raw packet trace. To identify the SSH flow, we investigate whether the message contains the SSH version following 'CR LF'. Therefore, we filter out those TCP flows beginning with 'SSH-* CR LF,' and label them into SSH traffic flows. Notice that, the checking of packet payload and port information is only used when labeling the traffic flows. In fact, the payload information can be encrypted, and customized ports can be used instead of well-known ports. For these traffic flows, our labeling methods cannot be applied. However, such omissions in labeling set would not impact the proposed algorithm evaluation since the pattern and message composition of those traffic flows will not change. This is the reason that we do not use the payload and port information in our model building (detailed in following sections).

By dealing with the three-day traffic flows, we have extracted the HTTP, SSH, SMTP and WeChat traffic flows, with the total number of 2649, 4792, 2041 and 2400, respectively. The detail of the collected samples are shown in *Table* I, the number of which are sufficient for in-depth statistic analysis and performance evaluation. For multimedia traffic flows of P2P and video, the traffic flows cannot be recognized directly by using the port. For example, video websites (e.g., Youku and Iqiyi) deliver video services via the port 443, but the application data forwarded also via the port 443 is usually encrypted. It means that it is difficult to identify whether such TCP flows contain video streaming or not. To overcome the limitation, we consider generate P2P and video streaming manually by recruiting volunteers. If the volunteer watches videos or downloads P2P contents, other network services are disabled. In the end, more than 2335 and 1630 multimedia streaming is labeled for P2P and video applications, respectively, the detailed traffic flow set is shown in Table I.

III. Empirical Studies on Message Segments

To capture the underlying features of each traffic flow, in this section, we extract the message sequence for all traffic flows and conduct extensive data analytics on message sequences.

A. Extracting Message Segments From TCP Packets

In order to mine the deep knowledge of traffic flows, we divide the traffic flow into multiple *message* segments, each of which implicitly indicates sender's instantaneous transmission intention. For example, a large-size video content will be transmitted through multiple message segments for performance satisfaction and convenient advertisements promotion. A HTTP flow can divide into multiple message segments to package the components of HTML, JSON, JavaScript, etc. Thence, it is reasonable to use the message segment information as input features to classify traffic flows, as they are capable of indicating the underlying characteristics of applications. As the message segments are normally large-size which have to be separated into many TCP packets to deliver in accordance with the limitation of maximum segment size (MSS),[1] we learn the negotiation process between the sender and receiver to detect the generation and end flag of each message, via which message segments are split out from the underlying TCP packets.

Particularly, in the TCP protocol, there are two major designs to enhance the transmission reliability. One is the ACK scheme that after the receiver successfully accepts a packet from the sender, the receiver will feed back a ACK packet to indicate a reception of the packet. Another one is the retransmission scheme that once the sender does not receive the ACK for the transmitted packet, it will transmit the packet again. Therefore, to extract message segment from raw TCP packets, the following rules are utilized: 1) retransmission packets and pure acknowledgment (ACK) are removed; 2) a packet with a MSS size data is accumulated to the current message segment; 3) a packet with a data

[1]MSS is determined by the MTU of the underlying network.

TABLE I
LABELED TRAFFIC FLOW SET

Category	HTTP	SSH	SMTP	WeChat	Remote Desktop	P2P	Video
Flow Numbers	2649	4792	2041	2400	2907	2335	1630
Total bytes	1.33 GB	15.3 MB	79.78 MB	61.55 MB	90.56 MB	200 GB	39 GB

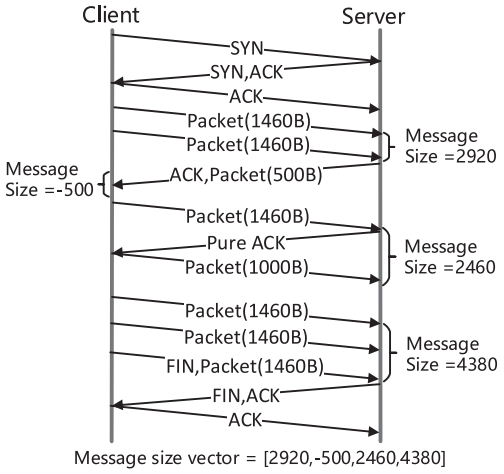

Fig. 1. An example of message segment extracting.

size smaller than MSS[2] is accumulated to the current message segment, which also indicates the end of current segment; 4) the beginning of message transmission from one side indicates the end of transmission of the other side. For instance, as shown in Fig. 1, the client is defined as the side that actives the connection, while the other side is defined as the server, and in this case, the network MSS is 1460 bytes. For the first two packets transmitted by the client, as their data sizes reach the MSS limitation, they are accumulated to the current message segment. Then, the first ACK sent by the receiver ends this message segment since it carries data. Therefore, the size of the first message segment is 2920 bytes. Likewise, for the second message, as the transmission from the client ends the segment initiated by the server, the segment size becomes -500 bytes. Notice that, we represent the message size transmitted from the client with a positive value (from the server with a negative value), the goal of which is to indicate the message direction, and the directions are not taken into account when conduct model training. Particularly, we implement the extracting tool by *Python*, and Algorithm 1 shows the pseudocode of the extracting algorithm. In Algorithm 1, the code line 7 and 15 are with respect to the rules 1) and 2), and the code line 10 corresponds to rules 3) and 4).

Algorithm 1: Message Segment Extracting Algorithm.

Input: $TCP\_packets$: a TCP packet list of the traffic flow
Output: $size\_vector$: the message segment sequence
1: **function** MESSAGE_EXTRACTING($TCP\_packets$)
2: $size\_vector = \{\}$
3: $message\_size = 0$
4: $last\_direc = 1$
5: **for** $packet$ in $TCP\_packets$ **do**
6: $payload\_size, direc \leftarrow packet.info$
7: **if** $payload\_size == 0$ **then**
8: Continue
9: **end if**
10: If$last_direc$!= $direc$ **or** $payload\_size < MSS$
11: $size\_vector.append(message\_size)$
12: $message\_size = 0$
13: $last\_direc = direc$
14: **end if**
15: $message\_size = message\_size + payload\_size$
16: **end for**
17: **return** $size\_vector$
18: **end function**

B. Sequential Message Characterization

After extracting message segments for all source applications, we show empirical studies on such message segments. As shown in Fig. 2, the message sequence of different types of traffic flow, in each of which 100 traffic flows are randomly chosen and shown. The color denotes the size of each message segment. Therefore, we have the three major observations. First, by checking each row in all figures, we can see that traffic flows from the same source application present *highly similar sequential features*, i.e., the colors of different rows in the same figure vary almost with the same pattern. Specifically, a few large-size message segments appear after some small-size message segments (possibly negotiation messages), and after that only small-size message segments exist for all SSH traffic, as show in Fig. 2(a). For all video streaming in 2(c), large-size message segments always interact with small-size message segments. Second, by comparing columns in different figures, we observe that for different types of traffic, their message sequential features are quite different from each other, i.e., columns in different figures show differential color variations. In Fig. 2(b), large-size message segments appear at the latter stage in Remote Desktop message sequences while they appear at the early stage in SSH message sequences. However, in Video traffic flows, large-size and small-size message segments appear alternately. Moreover, compare to SSH and Remote Desktop traffic flows, most of whose message sizes are smaller than 1500 bytes, in Video traffic flows, large-size message segments can generally reach up to 1 500 000 bytes. Other types of traffic also have different sequential message characteristics. Due to limited space, this paper will not elaborate on them.

[2]Note that, the input for our developed model, is the size of message segment, each containing multiple TCP packets. Although the value of MSS affects the size of TCP packet, it will not affect the size of message segment, which is determined by the upper-layer application behaviors.

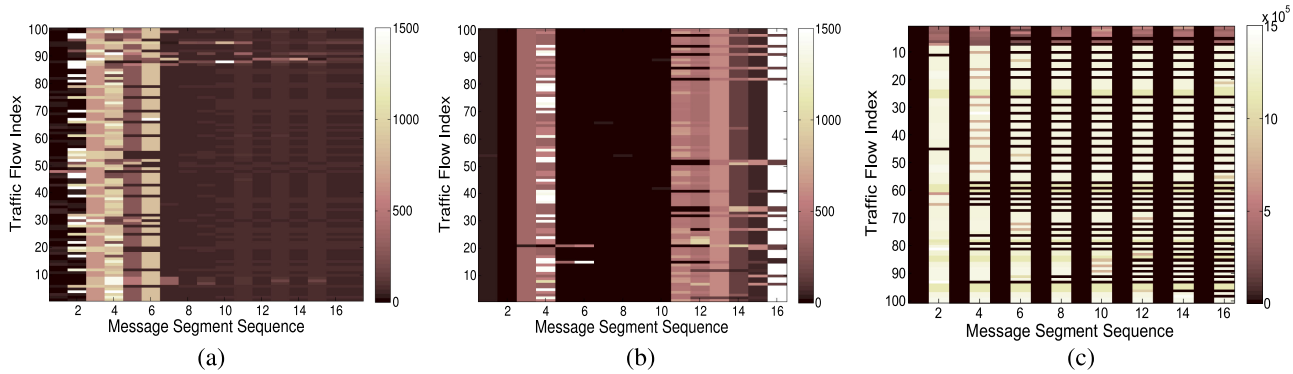

Fig. 2. Message sequences of different types of traffic flows. (a) SSH traffic flows. (b) Remote Desktop traffic flows. (c) Video streaming.

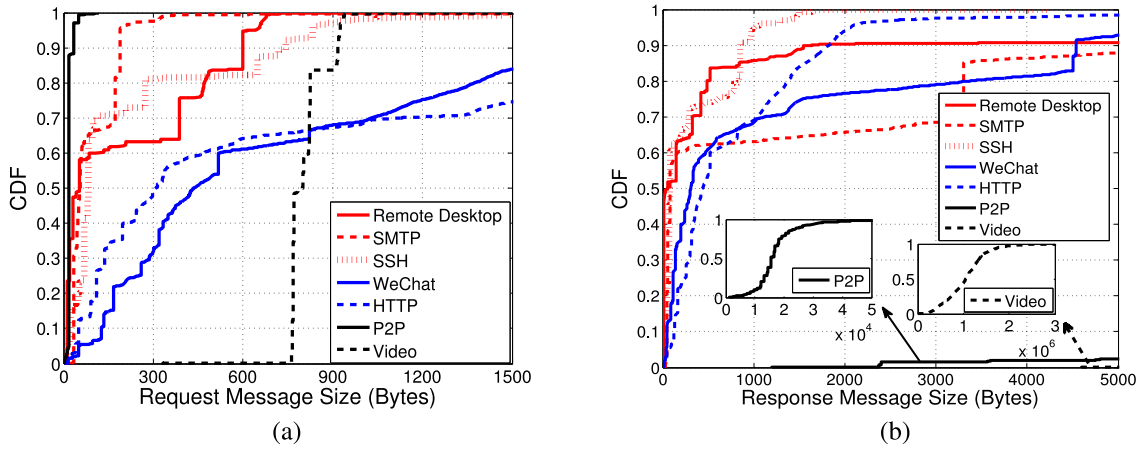

Fig. 3. CDFs of message sizes. (a) Request message sizes. (b) Response message sizes.

To further investigate the features of different traffic flows, we divide message segments from the same source application into two groups, i.e., request messages and response messages. We then plot Cumulative Distribution Functions (CDFs) of their message sizes for different source applications, which are shown in Fig. 3 (a) and (b), respectively. We can obtain two major observations. First, there is an obvious gap among all types of traffic flows in terms of both request and response message size. For instance, in Fig. 3(a), given a CDF threshold of 80%, the respective size of request message is about 100, 180, 290, 480, 850, 1380, and 1600 bytes for P2P, SMTP, SSH, Remote Desktop, Video, WeChat, and HTTP traffic flows. Second, compared with request messages, most (more than 80%) of whose sizes are smaller than 1500 bytes for all types of traffic flows, the differences in response message size are more significant (shown in Fig. 3 (b)). Specifically, with the CDF value of 80%, the message size for Remote Desktop, SSH, HTTP, WeChat, and SMTP traffic flows is about 500, 900, 1400, 3200, and 3200 bytes, respectively. Further, for P2P and Video traffic flows, their response message size can increase dramatically to 25, 000 and 1 500 000 bytes, respectively.

To this end, we can conclude that for different source applications, their message characteristics are significantly different from each other in terms of size and sequential feature. Therefore, it is possible to classify traffic flows according to their sequential message size information.

IV. Design of *SMC*

In order to classify traffic flows accurately, we develop an online classification system (also applicable to offline classification), named as *SMC* (Sequential Message Characterization), which can perform traffic classification in real time and output the classification result at the early stage of traffic generation.

A. System Overview

SMC classifies traffic flows by learning their sequential message size feature. Fig. 4 illustrates the work flow of *SMC*, which can be divided into offline component-*Modeling Traffic Flows*, and online component-*Classifying Traffic Flows*. In the offline part, for each type of traffic flows, we extract their message sequences and apply LSTM neural network to conduct *deep learning* on the sequential features, which can generate according LSTM models. In the online part, for a sample traffic flow, we first extract its sequential message size information (not necessary to be complete or from the beginning) and then input it to the model forest for testing. In the model forest, each LSTM model can output a probability profile that logs an accumulated

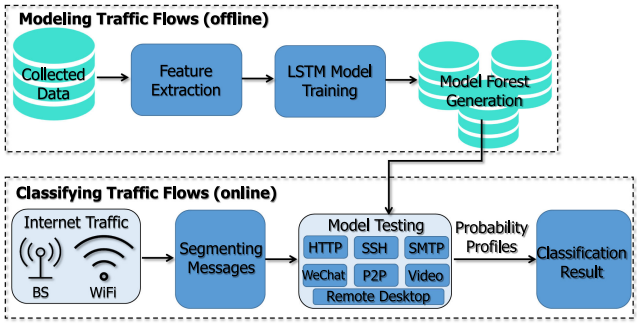

Fig. 4. System architecture and work flows.

prediction result for the sequential message size information. Based on the probability profiles, the classification result can be filtered out. In the following subsections, we will detail the design of each part.

B. Model Training at Offline Stage

1) Normalizing Message Size: Inspired by the preceding analytics, we adopt the sequential message size vector of each type of traffic flow as input, to train the classification model. We first normalize the value of message size so as to maintain their application semantics. For instance, for two messages with size of 35 and 135 bytes, for a certain application, they may have different semantic meanings in the training model, while for another application, other two messages with size of 20 and 25 KB may have the same semantic meaning. In other words, the larger the message size is, the less significance in the absolute size difference. To make messages with the similar semantic meaning have the same impact on the model, we normalize the original size of message segments as follows

$$f(x) = \begin{cases} 100 \times (\lfloor x/100 \rfloor + 1) & 0 \le x < 1000; \\ 500 \times (\lfloor x/500 \rfloor + 1) & 1000 \le x < 10\,000; \\ 10\,000 & x \ge 10\,000, \end{cases} \quad (1)$$

where the symbol $\lfloor . \rfloor$ is the floor function. After normalization, the diversity of sample values is significantly narrowed. As more clustered samples are achieved, it benefits the model learning and training with computation reduction and accuracy enhancement. To verify the efficiency of normalization, we examine the *entropy* of message sizes under original and normalizing conditions [25]. Specifically, let X be the random variable representing the message size measures of traffic flows. If there are totally N observed measures, denoted by a vector $U = \{u_1, u_2, \ldots, u_N\}$, with u_i $(1 \le i \le N)$ representing the i-th size measure, the probability of the measure being j can be calculated as x_j/N, where x_j denotes the number of measures being j. Therefore, the entropy for X is

$$H(X) = \sum_{j \in U} (x_j/N) \log_2 \frac{1}{x_j/N}. \quad (2)$$

The CDFs of entropy (each sample calculated by using 10 message sequences) of original and normalized message sizes are shown in Fig. 5, where solid lines are results of original message size while dashed lines are results of normalized message size. It can be easily seen that, after normalizing the message size, the entropy gets much smaller in all types of traffic flows.[3] Specifically, the median (with CDF value of 0.5) entropy of original message size is about 5.2, 6, and 7.6 bits in WeChat, Video and HTTP traffic flows, respectively, while the median entropy of normalized message size becomes 3.2, 1.1 and 3.8 bits, respectively.

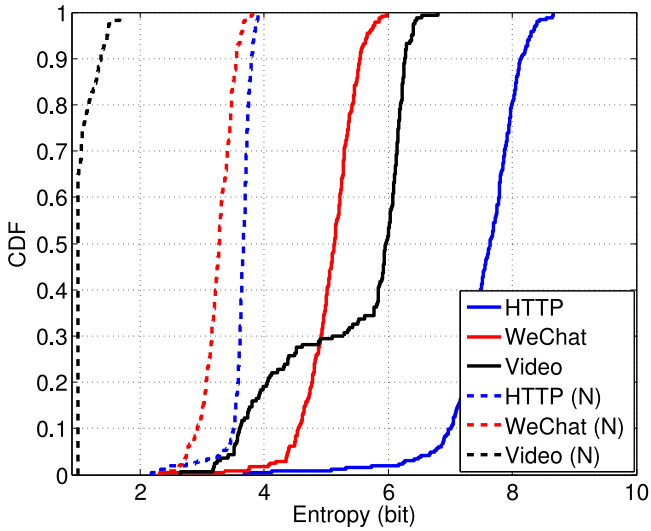

Fig. 5. CDFs of entropy of the original and normalized message size.

2) Long-Term Correlations Among Message Segments: To figure out an appropriate model for training message sequence features, we then investigate the correlations of message segments via examining the conditional entropy of message size measures. Particularly, when $k = 1$ (k means knowing the number of orders of previous states), let X' be the random variable representing the distribution of the previous measure given the measure X. In $U = \{u_1, u_2, \ldots, u_N\}$, when N is large enough, the distribution of X and X' would be the same, and the vector U can be rewritten as $U' = \{(u_i, u_{i+1}) : 1 \le i \le N-1\}$. Therefore, the joint entropy [26] of X and X' can be calculated as

$$H(X,'X) = \sum_{(X,'X) \in U'} P(X,'X) \log_2 \frac{1}{P(X,'X)}, \quad (3)$$

where $P(X,'X)$ is the frequency of $(X,'X)$ appearing in the vector U' divided by the total number of elements in U'. According to the *chain rule* of conditional entropy, with $H(X)$ (marginal entropy) and $H(X,'X)$, the conditional entropy of X when given X' is

$$\begin{aligned} H(X|X') &= H(X,'X) - H(X') \\ &= H(X,'X) - H(X). \end{aligned} \quad (4)$$

Similarly, when $k = 2$, let X'' be the random variable representing the distribution of the previous two measures given the measure X. The conditional entropy of $H(X|X'')$ can be

[3]Note that, the entropy results of other types of traffic flows are omitted due to the similar observation.

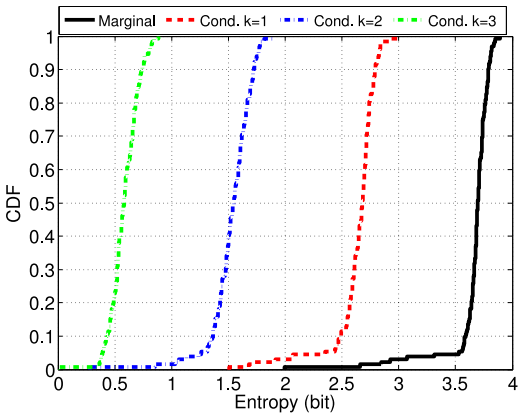

Fig. 6. Marginal and conditional entropy for HTTP.

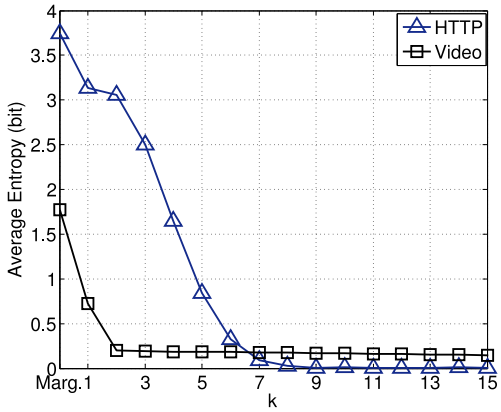

Fig. 7. Conditional entropy vs. k.

calculated as

$$H(X|X'') = H(X,'' X) - H(X'') = H(X,'' X) - H(X,' X). \quad (5)$$

Fig. 6 shows CDFs of the marginal entropy and conditional entropy of $k = 1, 2$ *and* 3 for HTTP traffic flows (other types of traffic flow have the similar observation), in which each sample is calculated by adopting 20 message sequences. We can easily observe that when $k = 1$, the conditional entropy is much smaller than the marginal entropy, and the conditional entropy of $k = 2$ and $k = 3$ are also much smaller than that of $k = 1$ and $k = 2$, respectively. For instance, the median marginal entropy is about 3.75 bits, while the median conditional entropy decreases below to 2.75, 1.6 and 0.6 bits when k equals to 1, 2 and 3, respectively. We can conclude that the message sequence *has strong correlation*, which means that with knowing previous message sizes, the uncertainty about current message size decreases. Generally, the more history message information we learn, the less uncertainty about the current measure has.

To understand how long the correlation lasts among message segments, we plot the average entropy (using all traffic flows) under the different value of k, which is shown in Fig. 7. Note that, for better illustration, only HTTP and Video results are shown, which is sufficient towards the following statement. It can be seen that, the correlation among message segments can last for

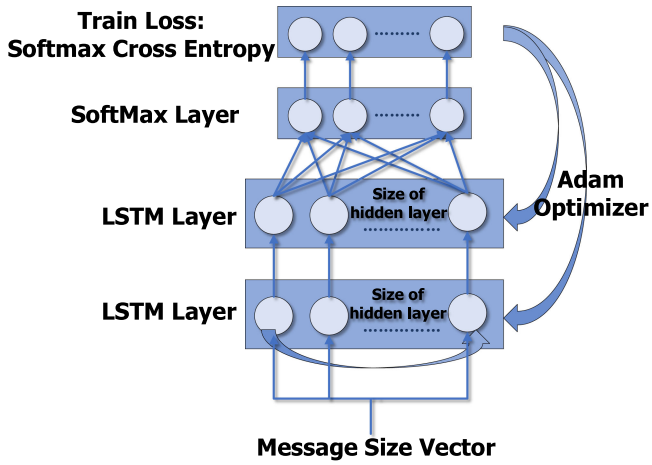

Fig. 8. Training process in the LSTM neural network.

a *long term*. Specifically, for HTTP traffic flows, to capture all correlation knowledge, the previous at least 8 measures should be well learned. For video traffic flows, even though learning previous 2 measures can obtain most of the correlation knowledge, other bits of correlation knowledge would last for more than 15 orders.

3) Training Traffic Models by LSTM Neural Network.: As LSTM can capture the long-term dependencies among time series data [27], we adopt it among traditional recurrent neural networks, HMMs or other sequence learning methods, to train our traffic models. After message size normalization, we can extract the message size vectors for all traffic flows. In order to obtain simple and scalable models, we employ one-to-one matching between model and traffic type, where each traffic type is trained independently with the LSTM network, generating an according LSTM model. Fig. 8 shows the LSTM network structure for model training, consisting of the input layer, multiple LSTM layers, the SoftMax layer, and the Train loss layer. Particularly, when inputting a message vector $S = \{s_1, s_2, \ldots, s_n\}$, the conditional situations of $s_1 \rightarrow s_2$, $(s_1, s_2) \rightarrow s_3$, $(s_1, s_2, s_3) \rightarrow s_4$,..., $(s_1, s_2, \ldots, s_{n-1}) \rightarrow s_n$, can be sequentially indicated. These sequential situations are input to the LSTM model for statistical processing and training, where the size of hidden units and the depth of LSTM layers are tunable hyper-parameters to debug performance. To prevent over fitting, the dropout regularization is included into the LSTM hidden layer. LSTM model can well address the gradient disappearing problem, but the gradient exploding problem may appear. To cope with it, the gradient clipping is utilized by setting a threshold as the maximum gradient value. If the size of sequential inputs is M and the size of LSTM hidden units is L, there is a full connection between the LSTM layer and SoftMax layer to resize the $1 * L$ vector to the $1 * M$ vector. The cross entropy is constructed as the loss function to train the LSTM model. With an input vector, the well trained LSTM model can predict a probability matrix indicating conditional probabilities that the successional next message size would be. In addition, the weights of LSTM layer parameters can be further optimized through the Adam Optimizer [28]. By conducting deep learning on each type of traffic flows based

Algorithm 2: The LSTM Neural Network Training Algorithm.

Input: $Layer\_dept$: the depth of the LSTM neural network
$LSTM\_size$: the size of each LSTM layer $Input$: the training data set Max_loss : the maximum allowed loss value
$Input$: the training data set Max_loss : the maximum allowed loss value
Max_loss : the maximum allowed loss value
Output: $LSTM\_network$: the well-trained LSTM-traffic model

function DO_TRAINING($Layer\_dept, LSTM\_size, Input, Max\_loss$)
  $Observation\_space =$ TO_SET($Input$)
  $Space\_size =$ Length$Observation_space$
  $LSTM\_network =$ Build_LSTM$Layer_dept, LSTM_size, Space_size$
  $Train\_loss = Integer.Max$
  $Input =$ One_Hot_Encode$Input$
  while $Train\_loss > Max\_loss$ **do**
    $Train\_loss = LSTM\_network.train(Input)$
  end while
  return $LSTM\_network$
end function

function BUILD_LSTM($Layer\_dept, LSTM\_size, Output\_size$)
  $network \leftarrow Empty\ Network$
  while $Layer\_dept > 0$
    $LSTM\_layer =$ LSTM_CELL($LSTM\_size$)
    $dropout\_layer =$ DROPOUT_WRAPPER($LSTM\_layer$)
    $network \leftarrow network + dropout\_layer$
    $Layer\_dept = Layer\_dept - 1$
  end while
  $SoftMax\_layer =$ SOFTMAX($LSTM\_size, Output\_size$)
  $network \leftarrow network + SoftMax\_layer$
  $Loss\_layer \leftarrow$ SOFTMAX_CROSS_ENTROPY($SoftMax\_layer$)
  $network \leftarrow network + Loss\_layer$
  return $network$
end function

on collected data, the model forest can be generated offline. **Algorithm 2** gives the pseudocode of LSTM neural network training. During the model training, the parameters of LSTM network depth and size of each LSTM layer are key factors to affect the model performance, which have been evaluated in our experiments section.

C. Classifying Traffic Flow at Online Stage

When conducting online traffic classification, message segments are extracted in real time to output a message size vector,

Algorithm 3: Online Traffic Classification Algorithm.

Input: $Model\_forest$: all types of LSTM-traffic model
$Size\_vec$: the input message size vector
Output: $Traff\_type$: the classified traffic type

function ONLINE_CLASSIFY($Model\_forest, Size\_vec$)
  Initialize: $Traff\_type = null, max\_prob = -1$
  for λ_i in $Model\_forest$ **do**
    p=1
    for s_j in $Size\_vec$ **do**
      if $j < len(Size\_vec) - 1$ **then**
        $Pro\_profile \leftarrow \lambda_i(s_0, \ldots, s_j)$
        p=$p \times Pro\_profile(s_{j+1})$
      end if
    end for
    if $p > max\_prob$ **then**
      $Traff\_type = \lambda_i$
      $max\_prob = p$
    end if
  end for
  return $Traff\_type$
end function

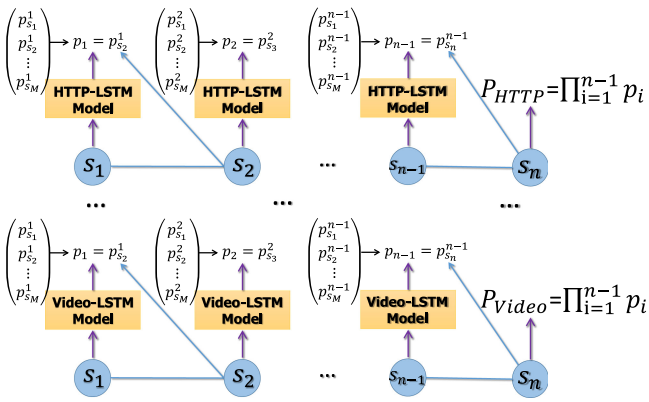

Fig. 9. Traffic type classification process.

which is then input to the model forest. Note that, the input message size vector has no necessary to start from the beginning or cover the complete message sequence of the traffic flow. *SMC* can classify the traffic flow at the early stage after the traffic starts, and the early classifying experiments have been carried out in the evaluation section. After testing the input message size vector, each LSTM-traffic model can output a probability profile, representing the prediction result for each message segment within the input sequence.

Specifically, we show the model testing process in Fig. 9, where we denote the model for traffic type i as λ_i. Given a message size vector $S = \{s_1, s_2, \ldots, s_n\}$, the model λ_i can output a probability vector $P = \{(p_1, p_2, \ldots, p_{n-1}), \lambda_i\}$, where the value p_j for $1 \leq j \leq n-1$ is the probability of the $(j+1)$-th message being predicted with the size of s_{j+1} when inputting $((s_1, s_2, \ldots, s_j), s_{j+1})$ to the model λ_i. More specifically, for an input of message sequence $(s_1, s_2, \ldots, s_j)$, the model λ_i can predict its next message size and output a probability matrix

P_{LSTM}, i.e., $\{p^j_{s_i}, p^j_{s_2}, \ldots, p^j_{s_M}\}$ meaning the probability of the next message size going to be $s_1, s_2, \ldots, s_M$, respectively, where M is the number of all possible size values. Then, with knowing the size of s_{j+1}, the value p_j is filtered out to be $p^j_{s_{j+1}}$, i.e., $p_j = P_{LSTM}(p^j_{s_{j+1}}|((s_1, s_2, \ldots, s_j), s_{j+1}))$.

With probability profiles achieved by all types of LSTM-traffic models, the final classification result can be filtered out. Particularly, as the probability of the message sequence being the traffic type i can be calculated as

$$\prod_{j=1}^{n-1} P_{LSTM}(p^j_{s_{j+1}}|((s_1, s_2, \ldots, s_j), s_{j+1}, \lambda_i)), \tag{6}$$

the model who gets the maximum probability, will be determined as the true traffic type for the message sequence. Then, the traffic type t can be represented as

$$t = \arg\ \max_i \prod_{j=1}^{n-1} P_{LSTM}(p^j_{s_{j+1}}|((s_1, s_2, \ldots, s_j), s_{j+1}, \lambda_i)). \tag{7}$$

Algorithm 3 shows the pseudocode of the algorithm.

V. Performance Evaluation

A. Methodology

We implement *SMC* by `Python`, which includes the offline LSTM neural network training part (using library tools of TensorFlow [29]), and online traffic classification part.

Benchmark Approach. To compare with our proposed *SMC* with other schemes, we develop a HMM-based method as a benchmark approach, in which a two-order Markov chain is adopted to process the same input vector as the LSTM model. The HMM-based approach is implemented by *Python* based on the Viterbi Algorithm [30].

For experiment setup, the traffic flow set (in Table I) is first divided into 10 subsets equally, which can be used to conduct cross validation experiments and vary the training size. Particularly, the set of data subsets is denoted by $\mathcal{D}$, and for cross validation experiments, if the i-th subset is chosen for data training, the data set of $\mathcal{D} - i$ will be used for performance evaluation. Therefore, by changing the value of i, the cross validation experiments can be conducted.

Metrics. The following four metrics are used to evaluate classification performance, where p_{ij} denotes the situation that a traffic flow type of j in ground truth is identified by the classifier as the traffic flow type i :

1) **Accuracy** means the probability that for all K types of traffic flow, a traffic flow is correctly identified, i.e., $\text{Accuracy} = \frac{\sum_{i=1}^{K} p_{ii}}{\sum_{j=1}^{K}\sum_{i=1}^{K} p_{ij}}$.
2) **Precision** means the probability that the classification for a traffic flow type A is exactly A in the ground truth, i.e., $\text{Precision}_k = \frac{p_{kk}}{\sum_{i=1}^{K} p_{ik}}$.
3) **Recall** means the probability that an event A in ground truth is identified as A, i.e., $\text{Recall}_k = \frac{p_{kk}}{\sum_{j=1}^{K} p_{kj}}$.
4) **F-Score** is a metric that combines precision and recall, i.e., $\text{F-Score}_k = 2 * \frac{Precision * Recall}{Precision + Recall}$.

B. Impact of LSTM Parameters

In this subsection, the impact of the depth of the LSTM network and the size of hidden units, on the detection accuracy and training time are investigated. Particularly, we adopt 9 subsets data to train the model and use the remaining 1 subset data for evaluation.

Fig. 10 shows the performance of classification precision and training complexity by investigating the impact of LSTM neural network depth. Particularly, from Fig. 10(a), we can see that a satisfied classification accuracy can be achieved with the two-layer LSTM neural network. For instance, when the number of layers increases from 2 to 3, 4, and 5, the overall accuracy will decrease from 96.9% to 53.8%, 21.4%, and 21.7%, respectively. It should be noted that, with increasing the depth of LSTM neural network, the learning capacity will increase which could result in the overfitting for the learning model. Therefore, the noise is unexpectedly injected into the model, which cloud decrease the learning precision. Fig. 10(b) shows the side impact of the inappropriate depth setting for the neural network, where much longer training time is required with the number of layers increasing. The training time means the time elapsed when finishing the training of deep learning model by a server. In our experiments, we adopt the server with the GPU of tesla k20. Taking the SSH traffic as an example, the training time can increase significantly to more than 4300 s with the five-layer LSTM model while only 40 s is required with the two-layer LSTM model. To this end, we adopt a two-layer LSTM neural network in the following evaluation experiments, to implement *SMC*.

Fig. 11 shows the impact of the size of hidden units at each LSTM layer. In Fig. 11(a), we can observe that the with the size of 400, most of traffic flow types can achieve the highest precision. Particularly, when the size is set to 20, 200, 400, and 800, respectively, the overall accuracy increases from 93.3% to 94.9% and 96.9%, and then decreases to 92.3%. For different sizes of hidden units, the required training time is shown in Fig. 11(b). It can be observed that when the size is 400, the increase of training time is negligible, but there is a significant increasement for the training time when the sized is increased to 800. To this end, for further performance evaluation, the size of hidden units of LSTM layer is set to be 400 in *SMC*.

C. Impact of Training Size

After achieving suitable parameters for the neural network, we show the impact of the training size. For this experiment, we use one traffic flow subset as the testing set, and increase the training size from one subset to ten subsets.

Fig. 12 shows the overall classification accuracy with varying the training size, and we can have two major observations. First, regardless of the training size, our classification approach outperforms the HMM-based approach significantly. For instance, with the training size of 0.5, *SMC* can achieve the accuracy about 92.7% while the accuracy is only about 26% in HMM-based approach, which is a significant gap. The reason is that a n-order Markov Chain can only 'remember' the impact of the latest n states in the time sequence while our LSTM model does not have

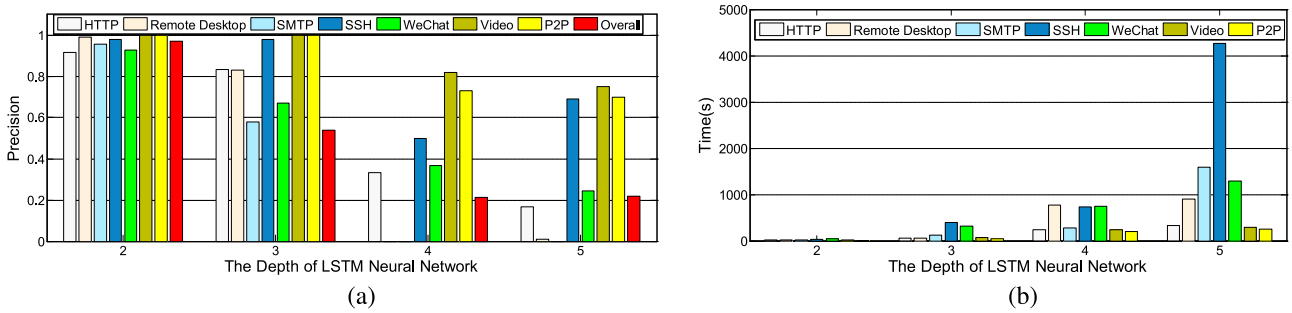

Fig. 10. Impact of the depth of LSTM neural network. (a) Precision under different depths. (b) Training time under different depths.

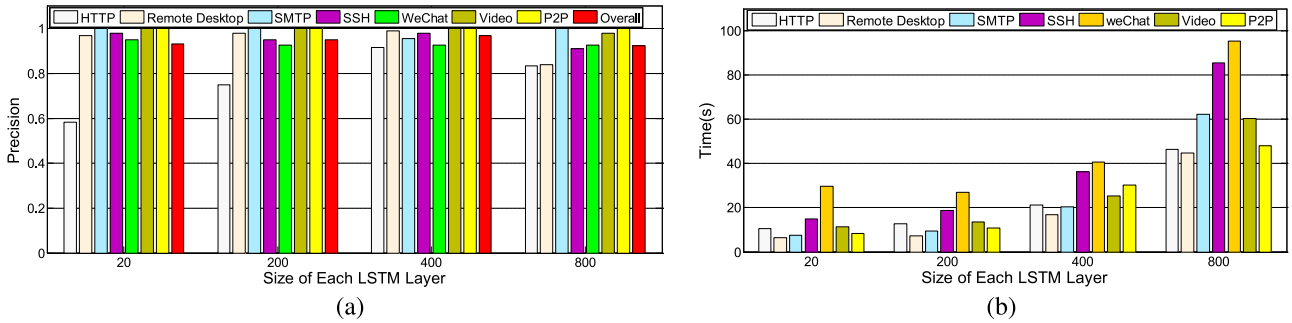

Fig. 11. Impact of the LSTM layer size. (a) Precision under different sizes of LSTM layers. (b) Training time under different sizes of LSTM layer.

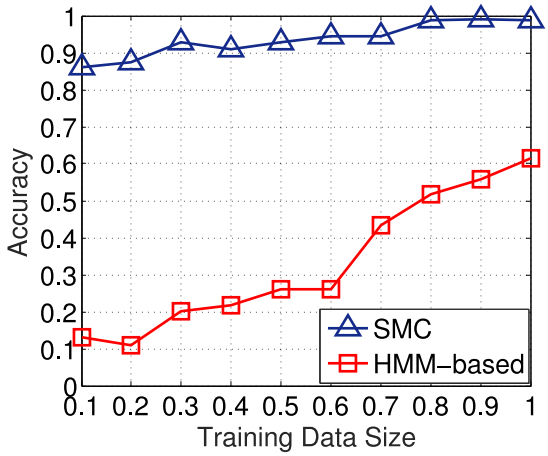

Fig. 12. Classification accuracy with varying training size.

a fixed weight on specific position of the previous states. Second, *SMC* just slightly relies on the size of training set. Specifically, when only one subset is used for training, *SMC* also achieves the accuracy above 86%. In addition, with increasing the training size, the accuracy achieved by *SMC* gradually increases and at last reaches the maximum value up to 98.9%. To better understand the impact of training size for each type of traffic flow, we calculate the precision results of all traffic flow types with varying training size, which are shown in Table II. We can see that only HTTP and WeChat traffic flows heavily rely on the size of training data. It is reasonable since the HTTP contents and online chatting contents are more diversified. Even so, with enough training data, their classification precision can be also well guaranteed. Specifically, when the training size becomes large enough, the precision of HTTP traffic classification can rise from 27.5% to 92.9%, and the precision of WeChat classification can be also enhanced from 15.6% to 97.4%. However, for other types of traffic flows, i.e., Remote Desktop, SMTP, SSH, Video and P2P, even with a small training data size (e.g., 10%), the classification precision can also reach as high as 99.2%, 91.4%, 95.3%, 88.2% and 96.4%, respectively.

D. Performance Comparison

In this subsection, we carry out the performance comparison when the hyper-parameters of the LSTM model is well tuned with enough training data. For cross validation, in each experiment round, we choose a different subset data for test, and the remaining subsets data are adopted for training.

Fig. 13(a) presents the overall accuracy of *SMC* and the HMM-based approach under different test data, where two major observations should be pointed out. First, under all rounds of experiments, there is a significant accuracy gap (larger than 30%) between *SMC* and the HMM-based approach. Second, in different experiment rounds, *SMC* is able to achieve a stable performance with only a slight fluctuation, where all accuracy scores are above 95% and the maximum accuracy score reaches as high as 98.9%. However, in the HMM-based approach, the accuracy score fluctuates dramatically, where the accuracy score can sometimes reach 62% while can also drop down to 48%, indicating that the HMM-based appraoch is deeply influenced by the sample diversity.

Fig. 13(b) shows the average classification results in precision, recall, and F-score schemes. It can be seen that for all traffic flow

TABLE II
CLASSIFICATION PRECISION RESULTS OF ALL TRAFFIC FLOW TYPES WITH VARYING TRAINING SIZE

Training Data Size	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HTTP	27.5%	34.4%	36.7%	36.7%	56.7%	56.7%	55%	56.7%	91.7%	92.9%
Remote Desktop	99.2%	99%	99.4%	99.7%	99.7%	99.3%	99.3%	99.5%	99.2%	99.6%
SMTP	91.4%	95.9%	97.5%	96.5%	96.8%	96.5%	96.4%	97.2%	97.3%	97.3%
SSH	95.3%	97.4%	98.1%	98.7%	98.9%	99.1%	99.3%	99.4%	99.4%	99.4%
WeChat	15.6%	78.8%	93.4%	96.3%	94.1%	94.1%	95.3%	94.3%	96.1%	97.3%
Video	88.2%	92.4%	100%	100%	100%	100%	100%	100%	100%	100%
P2P	96.4%	100%	100%	100%	100%	100%	100%	100%	100%	100%

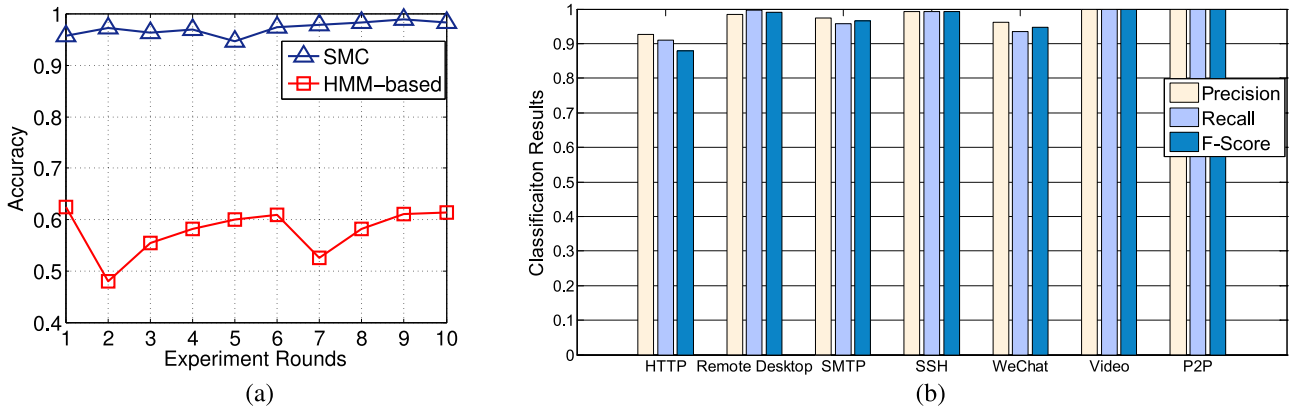

Fig. 13. Overall results under cross validation. (a) Total accuracy. (b) Average classification results for all traffic flow types.

types, the proposed *SMC* can recognize them well. For instance, although the HTTP classification has the minimum accuracy since the HTTP content transmission is the most complex, the classification performance is still acceptable with the precision score over 90%. On the other hand, the traffic for multimedia P2P and Video can be perfectly recognized whose precisions reach 100%. It is quite valuable since the multimedia traffic takes up the most traffic bandwidth. Besides, for other traffic types, they are also well identified with the precision between 95% and 98%. For all traffic types, recall and F-score scores are also very impressive, both of which have the same variation trend with the precision score.

E. Classifying Traffic Flow at Early Stage

To further improve the performance, we study how early the traffic flow can be classified, which is highly important for online traffic identification. Generally, for online traffic classification, the earlier the classifier comes to a correct conclusion, the better the classifier is. In this experiment, we adopt half of subsets as the training set and the other half of subsets as the testing set. The overall classification accuracy when varying the input length of message size vector as show in Fig. 14. As *SMC* requires at least two message segments as input, we examine the performance with the vector length ranging from 2 to 10. It can be observed that even with the minimal input length, i.e., 2 message segments, the accuracy score still reaches over 73%. In addition, the overall classification accuracy increases significantly until the vector length reaches 6 message segments. After that, the

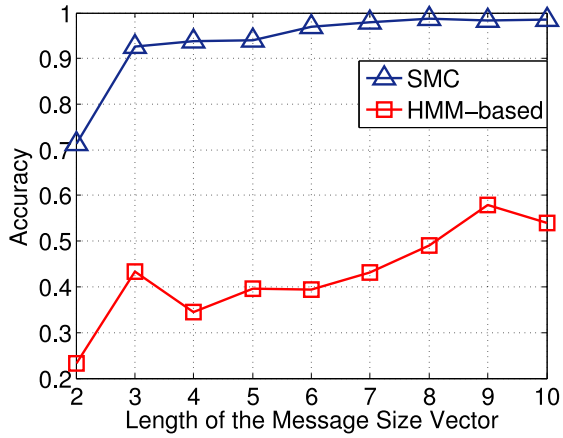

Fig. 14. Impact of the input length of message size vector.

accuracy score will fluctuate around 98%, which means that *SMC* can well classify the traffic flow with only 6 continuous message segments being extracted as input. The result is quite impressive, as *SMC* is capable of recognizing traffic flows at very early stage, especially for HTTP, Video, and P2P traffic flows, whose message sequence lengths are normally large with the median length of 15, 36, and 70 message segments, respectively (shown in Fig. 15). However, in the HMM-based approach, with the minimal input length, only 23% accuracy is achieved. When increasing the input length to 9 message segments, the maximum accuracy only reaches 58%, which is much smaller than that of our proposed *SMC* system.

TABLE III
CLASSIFICATION PRECISION RESULTS OF ALL TRAFFIC FLOW TYPES WITH VARYING INPUT LENGTH OF MESSAGE SIZE VECTOR

Input Length	2	3	4	5	6	7	8	9	10
HTTP	77.4%	88.5%	92.2%	93%	94.2%	94.1%	94.1%	94.1%	94.1%
Remote Desktop	45%	89.2%	92.1%	92.3%	96.2%	97.1%	97.1%	97.1%	97.2%
SMTP	86.1%	92.2%	95%	95%	98%	98%	99%	99%	99%
SSH	44.5%	92%	95.9%	95%	98.9%	98.9%	99.1%	99.2%	99.3%
WeChat	93.8%	91.5%	92.9%	91.9%	94.9%	95%	95%	95.1%	95.2%
Video	88.7%	96%	98%	100%	100%	100%	100%	100%	100%
P2P	76.9%	98.9%	100%	100%	100%	100%	100%	100%	100%

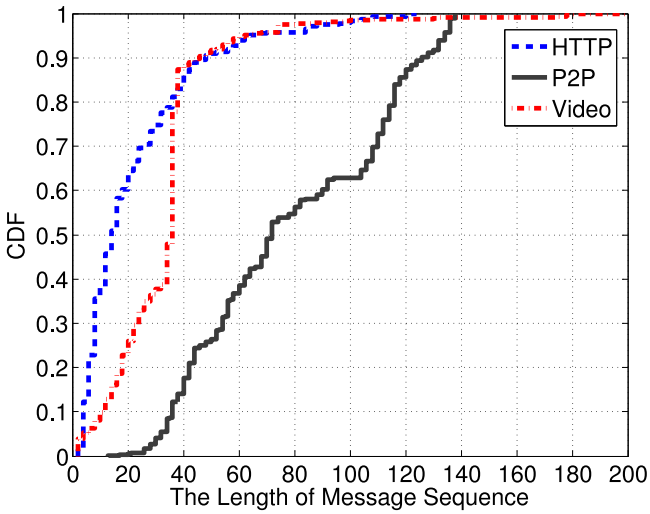

Fig. 15. CDFs of message sequence length of traffic flows.

To further understand how the classification precision of each traffic flow type is affected by the input length, we calculate classification precision results (in *SMC*) of all traffic flow types with varying input length, which are shown in Table III. We have the major observations as follows. First, for all traffic flow types, with larger input length, higher classification precision scores can be achieved. Second, for most of traffic flow types (except for Video and P2P traffic flows as they can be perfectly recognized), when the input length arrives at a certain value, the classification precision score cannot be further improved and will fluctuate within a small range (possibly restrained by the upper-bound caused by intrinsic feature complexity). For instance, for HTTP, Remote Desktop, SMTP, SSH and WeChat traffic flow types, the precision score will fluctuate around 94%, 97%, 99%, 99% and 95%, respectively, when the input length is large enough. Third, for all traffic flow types, the highest precision score can be achieved after the input length reaching 6 message segments.

VI. RELATED WORK

In the literature, the major traffic classification approaches, i.e., deep packet inspection, statistical and behavioral analysis on packet features, and port matching, which have been widely studied. In this paper, we review our related work mainly in the first two categories as current Internet applications normally adopt customized ports and port matching based methods are no longer efficient [31].

A. Deep Packet Inspection (DPI)

To cope with the port disguise problem, DPI-based approaches are proposed. Particularly, as conducting DPI on each packet payload requires significant CPU and memory resources, Alcock *et al.* proposed a lightweight classification approach, named *PACE*, which tries to examine the first four bytes of packet payload observed in each direction [14]. Risso *et al.* first summarized a taxonomy for DPI-based classification approaches with considering their characteristics, strength and weaknesses, as well as processing and memory requirements, and then carried out a comparison for the packet-based and message-based inspection approaches [15]. Normally, Network Processors (NPs) are essential to perform DPI as they have a good packet processing performance; to achieve high performance DPI, Piyachon *et al.* exploited NP's on-chip memory resource for parallel processing, to facilitate the processing efficiency [23]. In addition, Bujlow *et al.* organized a comprehensive comparison of several well-known DPI approaches, by studying their performance at various classification levels (e.g., protocol level and application level) [16].

However, to achieve a high-level classification performance, DPI-based approaches normally suffers two weaknesses: 1) DPI on each packet payload is computationally-costly and calls for many CPU and memory resources; 2) DPI on payload is invasive for network user privacy and might be invalid when meets encrypted protocols. In this paper, we focus on encrypted Internet traffic flows, which may restrain the usage of those DPI-based approaches.

B. Statistical and Behavioral Analysis on Packet Features

Statistical and behavioral analysis methods are proposed to characterize measurable properties of traffic flows such as packet inter-arrival time or packet size, where machine learning algorithms are employed to for traffic classification.

Regarding to supervised learning, Dusi *et al.* proposed a statistical classification mechanism, named *Tunnel Hunter*, to recognize whether a traffic flow is HTTP or SSH, in which three properties of IP packets, i.e., packet size, arrival order, and inter-arrival time, are extracted and trained offline to generate a "fingerprint" for HTTP and SSH traffic type [17]. To identify

P2P applications (e.g., Skype and Thunder) from other traffic flows, Wang *et al.* proposed an Application Behavior Characterization (*ABC*) technique, in which for a specific application, they extract application behavior features from multiple flows to achieve the correlation among these traffic flows. Based on these features, they trained a two-layer classification model based on decision tree to classify P2P application traffic [32]. To identify Web Real-Time Communication (*WebRTC*) which normally adopts dynamic ports and transmits under an encryption protocol, Mauro *et al.* devised a classifier, which can partition the traffic into *WebRTC* and *normal* two categories, where features such as inter-arrival times, the number of packets, and packet lengths are utilized, and then trained by the random forest algorithm [33]. In addition to the above algorithms, Support Vector Machine (SVM)-based algorithms are also popular in the literature. Specifically, with extracting the payload size of packets as statistical features, Este *et al.* devised a multi-class traffic classification approach based on SVM, which tries to perform correctly with as little training as possible [18]. Likewise, Finamore *et al.* also leveraged on statistical characterization of payload, and proposed an Internet classification engine, named *KISS*, to classify UDP traffic (mainly stemming from the momentum of P2P applications) [19]. Besides, neural networks are also applied to conduct traffic classification. Based on header-derived statistics (without port or host (IP address) information), Auld *et al.* trained a sophisticated Bayesian neural network to classify traffic flows, which is able to achieve an overall accuracy of 95% [34]. Using a set of inter-arrival time and packet size related statistical features to characterize traffic flows, Sun *et al.* built a probabilistic neural network to identify Web and P2P traffics, and confirmed it is an effective technique for traffic classification [35].

On the other hand, unsupervised learning is also studied for Internet traffic classification, where clustering algorithms are dominant approaches. Specifically, Liu *et al.* adopted the payload-independent statistical characters and experimented with unsupervised K-means to evaluate the classification performance, which is able to achieve an overall accuracy of 80% [36]. Using only transport layer statistics, Erman *et al.* conducted cluster analysis to identify groups of similar traffic, where two unsupervised clustering algorithms, i.e., K-Means and DBSCAN, are adopted and compared; they revealed that in comparison with K-Means, even though DBSCAN achieves a lower accuracy, it can produce better clusters [37]. Paramita *et al.* first adopted the Fuzzy C-Mean algorithm to conduct the clustering for data set, then used the k-nearest neighbors (K-NN) algorithm to classify traffic [38]. With doing so, the disadvantages of K-NN in computation process can be conquered. In addition, the principal component analysis algorithm is employed for feature selection, which can further improve the clustering and classification performance. Although unsupervised learning is independent of data labeling, its classification accuracy is usually lower than that of supervised learning [39].

For statistical based approaches, the selection of statistical features are critical to classification performance. In current approaches, many selected features are related to measurable properties in time domain. However, time-related metrics are normally unstable, which may be affected by network conditions such as bursty heavy loads and traffic congestion. Therefore, those algorithms have a robustness weakness, especially under dynamic network conditions. Differently, in *SMC*, the model input only contains the size of TCP packets, the classification performance of which is independent of port disguise, content encryption, and network status fluctuation. In addition, to the best of our knowledge, we are the first to investigate the early traffic classification, which is rather valuable for online network management.

In our previous work [40], we have demonstrated the efficacy of LSTM neural network for traffic classification by learning sequential message features. In this paper, we further improve it by conducting more data analytics on message sizes for feature extraction, elaborating on the design of *SMC*, and evaluating the capability of *SMC* for early online traffic classification. Moreover, up-to-date research works have been surveyed and the paper related work has been re-organized to well motivate our work. At last, we have revised the presentation of the whole paper to fit this journal version.

VII. Conclusion and Future Work

In this paper, we have collected extensive traffic flows from a public router, and labeled them into many application categories. After extracting the message sequence for all collected traffic flows, we have identified the unique *message sequential feature* for different traffic flow types, based on which we have developed an online Internet traffic classification system, named *SMC*. In *SMC*, the deep learning of LSTM is leveraged to model each traffic type as we have confirmed the long-term dependency among their message segments. Extensive experiments have been conducted and the results demonstrate the efficacy of *SMC*. Generally, *SMC* is able to achieve 97% of average classification accuracy, and can reach its approximately maximum efficacy when more than 6 pieces of message size information are input. It enables online early-stage traffic classification especially for heavy-traffic flows with long message sequences. Our designed *SMC* can be the basis of extensive intelligent network applications, including traffic monitoring, efficient load balancing, etc. For future work, we will collect more types of traffic flow to evaluate and enhance the robustness of *SMC*. Besides, we will exploit *SMC* to develop upper-layer network management applications.

Acknowledgment

The authors would like to thank student Renjie Jin for his helps in conducting experiments.

References

[1] A. Sivanathan *et al.*, "Classifying IoT devices in smart environments using network traffic characteristics," *IEEE Trans. Mobile Comput.*, vol. 18, no. 8, pp. 1745–1759, Aug. 2019.

[2] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang, and X. Shen, "Content popularity prediction towards location-aware mobile edge caching," *IEEE Trans. Multimedia*, vol. 21, no. 4, pp. 915–929, Apr. 2019.

[3] Y. Qiao, Y. Cheng, J. Yang, J. Liu, and N. Kato, "A mobility analytical framework for big mobile data in densely populated area," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 1443–1455, Feb. 2017.
[4] S. E. Coull and K. P. Dyer, "Traffic analysis of encrypted messaging services: Apple imessage and beyond," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 5–11, 2014.
[5] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Comput. Surv.*, vol. 52, no. 6, pp. 125:1–125:36, Oct. 2019.
[6] Y. Wang, Y. Xiang, J. Zhang, W. Zhou, G. Wei, and L. T. Yang, "Internet traffic classification using constrained clustering," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 2932–2943, Nov. 2014.
[7] D. Liu, A. Alahmadi, J. Ni, X. Lin, and X. Shen, "Anonymous reputation system for IIoT-Enabled retail marketing atop PoS blockchain," *IEEE Trans Ind. Inform.*, vol. 15, no. 6, pp. 3527–3537, Jun. 2019.
[8] P. Yang, F. Lyu, W. Wu, N. Zhang, L. Yu, and X. Shen, "Edge coordinated query configuration for low-latency and accurate video analytics," *IEEE Trans. Ind. Inform.*, vol. 16, no. 7, pp. 4855–4864, Jul. 2020.
[9] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the edge: A scalable IoT architecture based on transparent computing," *IEEE Netw.*, vol. 31, no. 5, pp. 96–105, Aug. 2017.
[10] W. Xu *et al.*, "Internet of vehicles in big data era," *IEEE/CAA J. Automatica Sinica*, vol. 5, no. 1, pp. 19–35, Jan. 2018.
[11] K. Kaneko, H. Nishiyama, N. Kato, A. Miura, and M. Toyoshima, "Construction of a flexibility analysis model for flexible high-throughput satellite communication systems with a digital channelizer," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2097–2107, Mar. 2018.
[12] N. Cheng *et al.*, "Big data driven vehicular networks," *IEEE Netw.*, vol. 32, no. 6, pp. 160–167, Nov. 2018.
[13] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "Blinc: Multilevel traffic classification in the dark," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, 2005, pp. 229–240.
[14] A. Shane and R. Nelson, "Libprotoident: Traffic classification using lightweight packet inspection," WAND Network Res. Group, Tech. Rep., Univ. Waikato, 2012.
[15] F. Risso, M. Baldi, O. Morandi, A. Baldini, and P. Monclus, "Lightweight, payload-based traffic classification: An experimental evaluation," in *Proc. IEEE Int. Conf. Commun.*, 2008, pp. 5869–5875.
[16] T. Bujlow, V. Carela-Español, and P. Barlet-Ros, "Independent comparison of popular DPI tools for traffic classification," *Comput. Netw.*, vol. 76, pp. 75–89, Jan. 2015.
[17] M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli, "Tunnel hunter: Detecting application-layer tunnels with statistical fingerprinting," *Comput. Netw.*, vol. 53, no. 1, pp. 81–97, 2009.
[18] A. Este, F. Gringoli, and L. Salgarelli, "Support vector machines for TCP traffic classification," *Comput. Netw.*, vol. 53, no. 14, pp. 2476–2490, 2009.
[19] A. Finamore, M. Mellia, M. Meo, and D. Rossi, "Kiss: Stochastic packet inspection classifier for UDP traffic," *IEEE/ACM Trans. Netw.*, vol. 18, no. 5, pp. 1505–1515, Oct. 2010.
[20] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, "Offline/Realtime traffic classification using semi-supervised learning," *Perform. Eval.*, vol. 64, no. 9-12, pp. 1194–1213, 2007.
[21] Z. Xiong, Y. Zhang, D. Niyato, R. Deng, P. Wang, and L. Wang, "Deep reinforcement learning for mobile 5 G and beyond: Fundamentals, applications, and challenges," *IEEE Veh. Technol. Mag.*, vol. 14, no. 2, pp. 44–52, Apr. 2019.
[22] Z. Xiong, Y. Zhang, N. C. Luong, D. Niyato, P. Wang, and N. Guizani, "The best of both worlds: A general architecture for data management in blockchain-enabled internet-of-things," *IEEE Netw.*, vol. 34, no. 1, pp. 166–173, Jan. 2020.
[23] P. Piyachon and Y. Luo, "Efficient memory utilization on network processors for deep packet inspection," in *Proc. ACM/IEEE Symp. Architecture Netw. Commun. Syst.*, 2006, pp. 71–80.
[24] Y. Jing, G. Xue, and S. Qian, "Noff: A novel extendible parallel library for high-performance network traffic monitoring," in *Proc. 24th Asia-Pacific Softw. Eng. Conf.*, Dec. 2017, pp. 130–139.
[25] F. Lyu *et al.*, "LEAD: Large-scale edge cache deployment based on spatio-temporal WiFi traffic statistics," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2020.2984261.
[26] J. Qin, H. Zhu, Y. Zhu, L. Lu, G. Xue, and M. Li, "POST: Exploiting dynamic sociality for mobile advertising in vehicular networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 6, pp. 1770–1782, Jun. 2016.
[27] X. Zhang, F. Yin, Y. Zhang, C. Liu, and Y. Bengio, "Drawing and recognizing chinese characters with recurrent neural network," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 849–862, Apr. 2018.
[28] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Netw.*, vol. 18, no. 5, pp. 602–610, Aug. 2005.
[29] TensorFlow Community, "Tensorflow apis," Accessed: Mar. 2018. [Online]. Available: https://www.tensorflow.org
[30] G. D. Forney, "The viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268–278, 1973.
[31] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in-network identification of P2P traffic using application signatures," in *Proc. 13th Int. Conf. World Wide Web*, 2004, pp. 512–521.
[32] D. Wang, L. Zhang, Z. Yuan, Y. Xue, and Y. Dong, "Characterizing application behaviors for classifying P2P traffic," in *Proc. IEEE Int. Conf. Comput., Netw. Commun.*, 2014, pp. 21–25.
[33] M. D. Mauro and M. Longo, "Revealing encrypted WebRTC traffic via machine learning tools," in *Proc. IEEE 12th Int. Joint Conf. e-Business. Telecommun.*, vol. 4, 2015, pp. 259–266.
[34] T. Auld, A. W. Moore, and S. F. Gull, "Bayesian neural networks for internet traffic classification," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 223–239, Jan. 2007.
[35] R. Sun, B. Yang, L. Peng, Z. Chen, L. Zhang, and S. Jing, "Traffic classification using probabilistic neural networks," in *Proc. IEEE 6th Int. Conf. Natural Comput.*, vol. 4, 2010, pp. 1914–1919.
[36] Y. Liu, W. Li, and Y.-C. Li, "Network traffic classification using K-means clustering," in *Proc. IEEE 2nd Int. Multi-Symp. Comput. Comput. Sci.*, 2007, pp. 360–365.
[37] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in *Proc. ACM SIGCOMM workshop Mining Netw. Data*, 2006, pp. 281–286.
[38] A. S. Paramita, "Improving K-NN internet traffic classification using clustering and principle component analysis," *Bull. Elect. Eng. Inform.*, vol. 6, no. 2, pp. 159–165, 2017.
[39] S. Valenti, D. Rossi, A. Dainotti, A. Pescapè, A. Finamore, and M. Mellia, "Reviewing traffic classification," in *Data Traffic Monitoring and Analysis*. Berlin, Heidelberg, Germany: Springer, 2013, pp. 123–147.
[40] R. Jin, G. Xue, F. Lyu, H. Sheng, G. Liu, and M. Li, "Leveraging interconnection of message sequence for traffic classification: A deep learning approach," in *Proc. IEEE 24th Int. Conf. Parallel Distrib. Syst.*, Dec. 2018, pp. 77–84.

Wenxiong Chen (Member, IEEE) received the B.Sc. degree in communication engineering and the M.Sc. degree in translation from Hunan Normal University, Changsha, China, in 2007 and 2011, respectively, and is currently working toward the Ph.D. degree with the Business School, Central South University, Changsha, China. He is currently a Lecturer with the Business School, Hunan Normal University. His research interests include information management, big data analysis, and personalized recommendation.

Feng Lyu (Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2018 and the B.S. degree in software engineering from Central South University, Changsha, China, in 2013. From September 2018 to December 2019 and from October 2016 to October 2017, he was a Postdoctoral Fellow and a Visiting Ph.D. Student with BBCR Group, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. He is currently a Professor with the School of Computer Science and Engineering, Central South University, Changsha, China. His research interests include vehicular networks, beyond 5G networks, big data measurement and application design, and could or edge computing. He was the recipient of the Best Paper Award of the IEEE ICC 2019. He is currently an Associate Editor for the IEEE SYSTEMS JOURNAL and the Leading Guest Editor of the *Peer-to-Peer Networking and Applications*. He is a Member of the IEEE Computer Society, the Communication Society, and the Vehicular Technology Society.

Fan Wu (Member, IEEE) received the Ph.D. degree in computer science and technology from Central South University, Changsha, China, in 2020. From 2018 to 2019, he was a Visiting Ph.D. Student with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. Since July 2020, he has been a Postdoctoral Fellow with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include information-centric networking and network protocol.

Peng Yang (Member, IEEE) received the B.E. degree in communication engineering and the Ph.D. degree in information and communication engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2013 and 2018, respectively. He was a Visiting Ph.D. Student with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada from September 2015 to September 2017 and a Postdoctoral Fellow from September 2018 to December 2019. Since 2020, he has been a Faculty Member with the School of Electronic Information and Communications, HUST. His current research interests include mobile edge computing, video streaming and analytics, and machine learning.

Guangtao Xue (Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2004. He is currently a Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research interests include vehicular networks, wireless networks, mobile computing, and distributed computing. He is a Member of the IEEE Computer Society and the IEEE Communication Society.

Minglu Li (Senior Member, IEEE) received the Ph.D. degree in computer software from Shanghai Jiao Tong University, Shanghai, China, in 1996. He is currently a Full Professor and the Director of Network Computing Center, Shanghai Jiao Tong University. He has authored or coauthored more than 400 papers in academic journals and international conferences. His research interests include vehicular networks, big data, cloud computing, and wireless sensor networks. He was the Chairman of the Technical Committee on Services Computing from 2004 to 2016, the Technical Committee on Distributed Processing from 2005 to 2017 , and of the IEEE Computer Society in Great China region. He was the General Co-Chair of the IEEE International Conference on Services Computing, the IEEE International Symposium on Cluster Computing and the Grid, the IEEE International Conference on Parallel and Distributed Systems, and the IEEE International Parallel and Distributed Processing Symposium, and the Vice Chair of the IEEE International Conference on Computer Communications. He was also a Program Committee Member of more than 50 international conferences including the IEEE International Conference on Computer Communications from 2009 to 2016 and the IEEE International Symposium on Cluster Computing and the Grid in 2008.