Journal of High Speed Networks 30 (2024) 517–533
DOI 10.3233/JHS-230145
IOS Press

HClass: Fast hybrid network traffic classification with bit and keyword level signatures

Pratibha Khandait and Neminath Hubballi *
Department of Computer Science and Engineering, Indian Institute of Technology Indore, India
E-mails: phd1801101001@iiti.ac.in, neminath@iiti.ac.in

Received 1 September 2023
Accepted 14 April 2024

Abstract. Deep Packet Inspection (DPI) methods are extensively used in traffic classification. These methods extract unique application content either at byte or bit level granularity and represent them as signatures. DPI involves string or regular expression matching, which is computationally expensive, and evaluating signatures at bit-level granularity makes it even more inefficient. With the ever-increasing bandwidth and the high-speed internet traffic, the software implementations of DPI have become a performance bottleneck. In this paper, we propose HClass, a DPI-based network traffic classifier completely implemented in software to speed up signature matching. Our contributions with HClass are three-fold. First, we propose a hybrid signature matching technique with a combination of bit and byte-level signatures. Second, we propose methods to perform bit-level signature matching with byte/word level operations to cope with software implementations and be compatible with general-purpose CPU operations. Third, it uses a two-phase signature matching where first-phase signatures are short and quickly identify the potential application(s), and the second-phase signatures verify the potential application(s) to reduce false positives. We perform experiments with HClass on three datasets and report classification performance and execution time improvement of HClass with our implementations in C language.

Keywords: Accelerating traffic classification, Deep Packet Inspection, computational efficiency, bit-level signatures, byte-level signature

1. Introduction

Real-time network traffic classification is important for Internet Service Providers (ISPs) and network administrators for network management operations. ISPs use traffic classification primarily for providing differentiated treatment, meeting the Quality of Service (QoS) demands of applications, and other network management operations like security monitoring. Traditional port number based traffic classification is no longer reliable as various protocols use non-standard port numbers for different reasons [26]. Thus recent works use other techniques like machine learning algorithms [1,21,23], application behavior profiling [18] and deep packet inspection [12,19] to classify network traffic. Machine learning algorithms use a set of flow level characteristics like packet inter-arrival rate and the number of bytes/packets within a flow for uniquely identifying applications. A major limitation of machine learning methods is that they have not been implemented in real networks [31]. Profiling applications will be costly both in terms of computation and storage.

Deep Packet Inspection (DPI) analyzes the application payload for identifying the set of keywords [19,28] or short sequence of bytes [33] or invariant bits [15] to generate signatures. Thus DPI is an effective method for traffic

*Corresponding author. E-mail: neminath@iiti.ac.in.

classification as it relies on unique keywords or bits specific to an application. These application signatures are usually represented in the form of Finite State Machine (FSM) and regular expressions. Hence signature evaluation with payload content is computationally expensive [7]. Thus a challenge for the DPI-based traffic classifiers is to reduce the computational overhead. In order to achieve the execution speed, recent works propose to limit the amount of data screened to a few initial bytes [4]. It has been shown in the literature that keyword-based signatures can identify only text-based protocols, while bit-level signatures can identify both binary and text-based protocols [13,14,35]. However, bit-level signature matching is very expensive to implement in software. In this paper, we propose HClass – a hybrid method that has both bit-level and keyword-based signatures to identify the application traffic. Our contributions in this paper are as follows.

(1) We propose a hybrid network traffic classifier HClass, which uses both keyword-based and bit-level signatures for application identification.
(2) Signatures of HClass are organized into two levels. First level contains application-specific short bit-level signatures and second-level signatures is a collection of keyword-based and bit-level signatures used for identifying text protocols and binary protocols respectively.
(3) We propose a very efficient signature matching technique at the bit level using word-level operations tailored for software implementation.
(4) We experiment with a set of protocols and show that HClass accomplishes improved execution performance over other bit-level signature matching techniques.

We structure the remainder of this paper as follows. In Section 2, we brief the related work on DPI and machine learning based traffic classification techniques. In Section 3, we describe our proposed hybrid traffic classifier HClass, bit-level and keyword-based signature generation, and how signatures are used for classifying the network flows. Experiments and evaluation done with HClass are outlined in Section 4. Finally, this paper is concluded in Section 5 with a summary and future scope of work.

2. Related work

In this section, we describe prior works on network traffic classification techniques. These works mainly are of two types as ones based on Deep Packet Inspection and others based on machine learning and deep learning. We elaborate works of these two categories below.

(I) Deep Packet Inspection based Techniques: These works can be grouped into three categories as detailed below.

(i) Using Application Keyword based Signatures: These methods identify a commonly found yet meaningful set of keywords [28,30] from application flows. Words that are part of the application header are good candidates for such keywords. For example, in SMTP protocol, the words like "helo", "ehlo", "mail", "from", and "to" are good candidate keywords. These keywords are extracted considering their frequency of occurrence in the flows. SANTaClass [30] and RDClass [28] are two methods which identify keywords after parsing the payload using a collection of delimiters. Once the keywords are identified, they are represented in the form of a state transition machine as signatures. A disadvantage of these methods is that they make multiple scans of payload where in the first scan, words are extracted, and in the second scan, keywords and their ordering is verified. In order to speed up signature matching, a single scan algorithm is described in [12,19]. There are software-based algorithms designed [8,32] to meet the computational need of such keyword-based and regular expression based signatures. Few other works like DPFEE [17] implement hardware-based solutions for Deep Packet Inspection. Ren et al. [25] proposed outsourcing the DPI operations to a third party and using it as a service.

(ii) Using n-grams Extracted from Payload: These methods extract short sequences (n-grams) from the payload of application flows. These short sequences of the payload portion can be used along with their sequence for flow classification. Few works [33,37] also generate keyword-based signatures using these n-grams. ProDecoder [33] and Securitas [37] use the latent relationship between these n-grams to generate keywords which are subsequently used for classification.

(iii) Using Bit-level Signatures from Payload: Some of the recent works generate signatures using bit-level content. These bit-level signatures can handle new application encoding and data formats. Further bit-level signatures will be short compared to the other two types and hence are better for preserving users' privacy. BitMiner [36] and Bits Learning [35] learn the bit value at a particular position. *BitCoding* [13,14] identifies invariant bits as signatures which are encoded with run-length encoding. A state transition machine based bitwise matching is done to classify unknown flows. BitProb [15] generates probabilistic bit signatures to classify network traffic. It extracts the first 'n' bits of an application flow and calculates the probability of a bit being 0 or 1 at a particular position. A probabilistic state transition machine is generated with all bit probability values to classify the network traffic. These methods are not computationally efficient as they compare a single bit at a time and hence require as many comparison operations as there are bits in the signature.

II. Machine Learning based Techniques: Another major category of works use machine learning algorithms for network traffic classification [2]. These algorithms mainly fall into supervised [6], unsupervised [38], ensemble methods [11]. Supervised algorithms use labeled application flows or custom feature vectors for classification or identification of applications. However, these methods can not identify new applications which are not used for training purpose. This issue is addressed in unsupervised methods which group the flows or feature vectors based on similarity. New or unknown applications are identified if new applications do not show similarity with existing set of applications. Ensemble and hybrid methods claim to improve the performance by using more than one type of classifiers. Jin et al. [16], build an accurate and robust classifier with multiple binary classifiers trained on labeled flow level data. Some recent works also use deep learning models like transfer learning [27], multimodal learning [1], recurrent neural networks [20], deep convolution networks [9] for traffic classification showing varied accuracy and performance.

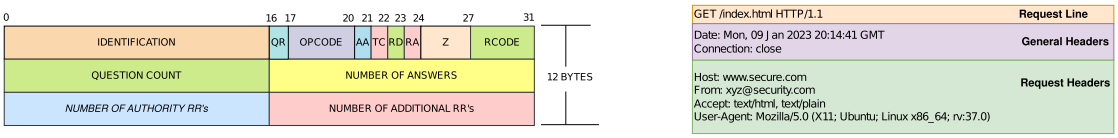
The work of Hussein et al. [24] highlights the limitations of machine learning methods. These limitations appear in the form of not being able to identify few applications correctly [3] (for e.g., some methods can not distinguish between gmail and yahoo mail traffic). Further, a large number of video streaming applications like youtube, amazon prime, etc may have overlapping characteristics. Similarly mobile applications have very large set running into millions in count. We believe that such scenarios prevent the machine learning based methods from scaling. To the best of our knowledge there are no studies which have validated these algorithms on large scale experiments. Further, machine learning methods require feature extraction which often requires entire flow to end incurring significant delays in classification.

3. Proposed traffic classification

This section describes the proposed hybrid traffic classifier `HClass`. In the next four subsections, we present the design rationale of the proposed hybrid traffic classifier `HClass`, a high-level description of its working, details of how different types of signatures are generated in `HClass` and signature matching at run time.

3.1. Design rationale

Application Protocols are of two types, binary and text-based. Binary protocols set specific values at specific bit positions. For example, flags in the header fields (TCP SYN flag) are examples of binary representation. Hence, the data in binary protocols can be viewed as a series of bits with fixed values (0 or 1) at specific positions. An example of a binary application protocol is DNS, whose header is shown in Fig. 1a. The DNS header has flags with fixed bit values positioned at a specific offset from the beginning. For instance, the QR flag in the DNS header is a one-bit field specifying the type of message. A value '0' indicates the query, and the value '1' is used for the response. Prior works [13–15] have shown that these protocols can be identified with bit-level signatures made up of 0's, 1's, and $*$'s generated from the initial few bytes of the payload. However, extracting signatures at the bit-level granularity and performing bit-by-bit signature comparisons (methods used in the existing literature) is computationally expensive.



(a) DNS Header (b) HTTP Header

Fig. 1. Binary and text-based protocol headers.

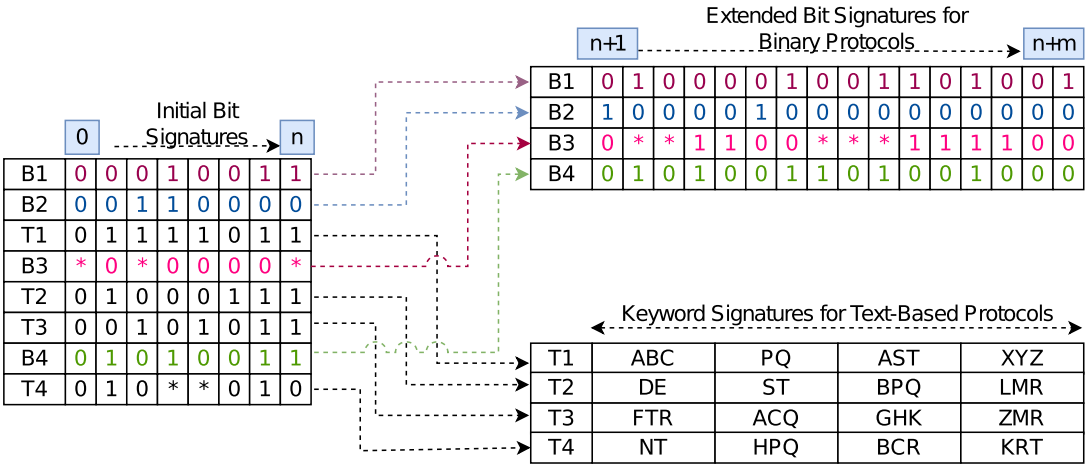


Fig. 2. Example showing application specific bit signatures and keywords.

On the other hand, text-based protocols contain plain text data in a human-readable format. Text-based protocols use only the values corresponding to human-readable ASCII/Unicode characters and can be classified efficiently using keyword-based signatures [28,30]. HTTP is an example of a text-based protocol. A sample HTTP header is Shown in Fig. 1b. It is easy to see that there are words like “GET”, “HTTP”, “Host”, and “User-Agent”. These words can serve as candidates for identifying HTTP traffic. Keyword-based signatures use such words as part of their signatures. Keyword-based signatures are reliable and easy to implement with software programs running on general-purpose CPUs. However, searching keywords in the payload involves string matching, which is a computationally expensive operation that can lead to higher time complexity. The larger the payload size, the longer it takes for the search operation and classification. Taking motivation from these, in this paper, we propose `HClass`, a fast hybrid network traffic classifier that efficiently implements both bit-level and keyword-based signatures to quickly classify both binary and text-based protocols.

3.2. *Working of hybrid classifier*

`HClass` is a hybrid network traffic classifier aimed to identify both text-based and binary protocols using a combination of keyword-based and bit-level signatures. `HClass` signatures are organized into two levels. The first-level signatures are binary signatures generated using application payloads of both text and binary protocols, and second-level signatures are separated with keyword-based signatures for text-based protocols and binary signatures for binary protocols. The idea here is to use the best type of signatures for identifying the respective class of protocols. Further, the first level signatures are in binary format, as text-based signatures can also be converted to binary signatures by converting the keyword(s) into a binary representation. Figure 2 shows the two-stage signature structure of `HClass`. The first stage bit signatures are short with fixed number of bits whose purpose is to quickly identify the probable application(s). As these signatures are binary irrespective of underlying application type,

bit-level comparison is done in the first phase. Here the initials B_i represent the signature of a binary protocol and T_i represents the signature of a text-based protocol. In Fig. 2 there are four protocols of each type are shown. It is evident to note that bit level signatures are made up of a series of 0's, 1's, and $*$'s. Once the initial bit-level signatures are matched, subsequently expanded/second-level signatures are matched.

The main idea of having the first-level signatures is to quickly determine potential application matches and whether the flow belongs to a binary or a text-based protocol. First phase signature matching assigns a temporary label to a test flow indicating the type of protocol matched (binary or text-based) and the name of the application to which the flow belongs. Using this labeling the second phase signature matching is done. If the first level matching identifies a test flow as a binary protocol, then the second level bit-level signatures are matched to the probable application(s). The second level signature matching omits the first n bits already matched in the first phase. On the other hand, if the flow is labeled as belonging to a text-based protocol, the classifier opts for a keyword-based signature matching in the second phase for verification. If the keywords corresponding to a probable application protocol identified in the first phase are found in the second level keyword based signature matching, then the flow is identified to be of that application. Thus, the first-level signature matching helps narrow the potential matches by speeding up the comparisons.

In the subsequent paragraphs, we elaborate the details of how signature matching is performed and other heuristic-based optimizations done to speed up this operation.

3.3. Signature generation for hybrid classifier

As mentioned earlier, `HClass` uses two level signatures for classifying the network traffic. The signatures are generated by analyzing the flow payload content of applications. `HClass` classifies both binary and text-based protocols. It uses bit-level signatures for binary protocols, a combination of bit-level and keyword-based signatures for text-based protocols. These signatures are arranged in three parts: a) common bit-level signature, b) extended bit-level signature, and c) keyword-based signature. These three types of signature generation is explained in the subsequent paragraphs. Signature generation uses flow content, and the first step is to reconstruct the flows.

`1. Flow Reconstruction: HClass` generates signatures using payload content extracted from specific application flows. Thus, the first step is to reconstruct bidirectional flows from network packets. This is done by considering five tuples, namely `SrcIP`, `DstIP`, `SrcPort`, `DstPort`, `Protocol` from an individual packet. All packets sharing these common attributes will be part of the flow. `HClass` uses bidirectional flows, hence the packets in either direction with `SrcIP`, `DstIP` and `SrcPort`, `DstPort` interchanged will be part of the flow. In general, there are two ways to identify and reconstruct flows. First, by identifying the connection from start to end, and second, is time-bound. In the first case, all packets from start to end sharing common attributes will be part of a flow. In the time-bound case, all packets exchanged between two hosts sharing common attributes and having inter-packet timing of not more than a threshold ζ will be part of a flow. For TCP, which is a connection-oriented protocol, the flow involves all the packets exchanged between connection establishment with a three-way handshake and connection closure with a four-way handshake or till the RST packet is seen. In the case of UDP, all packets having inter-packet timing of not more than ζ and sharing common attributes will be part of the flow.

`2. Common Bit-Level Signature: HClass` generates common bit level application protocol signatures using the first 'n' bits of payload. Thus it extracts first 'n' bits from every application flow (both binary and text-based protocols) and identifies bits that are invariant as shown in Fig. 3. In this sample figure, three applica-

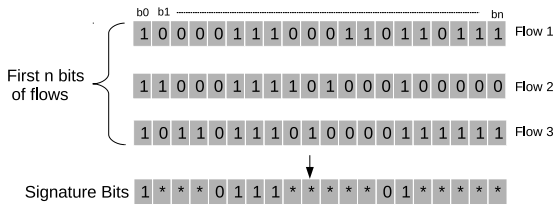


Fig. 3. Bit level signature.

tion flows of an application are shown, each having a bit sequence of length n. Using these bit sequences, a 'n' bit signature sequence is generated. This signature bit sequence will have bits marked with 1, 0, and $*$s. Those bit positions which are consistently either 0 or 1 across the flows will be marked with 1 or 0, and those bit positions which do not have consistently either 1 or 0 will be marked with $*$. It is worth noting that bit-level signatures can be generated for text-based protocols which use keywords as part of signatures. For example, HTTP protocol may use 'GET' as a keyword as part of its signature. This can be viewed as a series of bits for initial bit signature generation. Sometimes, it may happen that the entire keyword 'GET' may not appear within the 'n' number of bits chosen for first-level signatures. However, this is handled as keyword matching at the second level happens from the beginning of the payload.

`3. Extended Bit-Level Signatures for Binary Protocols:` For the binary protocols, additional m bit binary signatures are generated by identifying invariant bits as in the previous case.

`4. Keyword Signature for Text-based Protocol:` Signatures for text-based protocols are generated by identifying unique keywords appearing in the application payloads. For identifying keywords also, `HClass` uses the reconstructed flows. The keywords are selected with the following procedure

a) Parsing the Payload for Word Extraction: The payload text of the application is parsed using a set of delimiters such as 'blank', 'coma', 'colon', 'semicolon', 'tab', 'newline', etc.
b) High-Frequency Word Identification: The words that appear most frequently will be identified as potential candidates for keywords.
c) Word Filtering: Some of the words like 'the', 'and', date, names, etc. may appear frequently across the flows. However, they do not make up as good candidates for signatures. These words will be filtered by having a dictionary of bad words.
d) Final Keyword Selection: The remaining set of words is screened by an expert to select meaningful keywords for every application.
e) Keyword Offset Calculation: Once the keywords are selected for an application, a second round of payload search is done to identify the byte offset values of each keyword. Although offset calculation is optional, it will help in accelerating the search operation as certain portion of the payload can be skipped from evaluation.

`5. Bit Mask Generation for Bit-Level Signatures:` Matching the signatures in real-time is a very expensive operation as it involves comparing the payload of new packets against the signatures of different applications. As mentioned earlier, bit-level signatures are not efficient if compared bit by bit [14] using general-purpose processors with software implementations due to the following two reasons.

a) These processors do not support bit-level operations as there are no instructions to perform bit-level comparison operations.
b) The signatures have $*$ bits which need to be ignored while comparing the two bit sequences (signature and payload content). Thus we design a technique where we leverage the byte level operations to perform several bits comparison.

We address these two problems by deriving two masks (Mask-1 and Mask-2) from the signature bit sequence represented in Fig. 3. Mask-1 and Mask-2 bit sequences we use during signature matching. The two masks are generated as follows.

(i) Mask-1 will have those bits which are marked as $*$s in the signature filled with ones, and remaining bits which are consistently either 0 or 1 will be filled with zeros.
(ii) Mask-2 is a complement of Mask-1, i.e., it will have those bits which are marked as $*$s in the signature filled with zeros, and the remaining bits which are consistently either 0 or 1 will be filled with ones.

The idea of using Mask-1 and Mask-2 is to ignore $*$ bits from comparison with word-level operations. The two masks generated for the example bit signature sequence of Fig. 3 are shown in Fig. 4. These masks will help in efficient signature matching, as elaborated in subsequent sections.

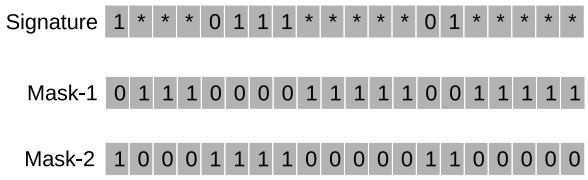


Fig. 4. Bit Mask-1 and bit Mask-2 from signature.

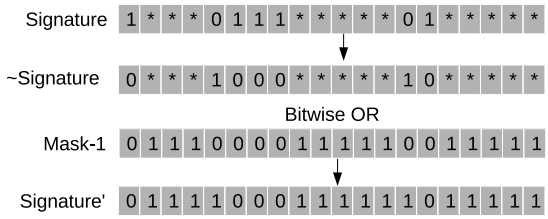


Fig. 5. Signature′ generation with bitwise OR of signature complement and Mask-1.

Fig. 6. Input′ generation with bitwise AND of input bit sequence and Mask-2.

3.4. Signature matching for real-time traffic classification

The traffic classification phase classifies flows into different applications using both common bit signatures and extended bit signatures or keyword-based signatures. As in the signature generation phase, flows are generated with packets sharing common attributes, and payload data is used for signature matching. From the flow data, the first '*n*' bits are extracted and compared with common bit signatures. Using this, the potential application signatures are determined and subsequently matched with either an extended bit signature or a keyword-based signature. Both bit-level signatures (common and extended) will make use of masks generated using the method described in Section 3.3. The details of bit-level signature matching and keyword-based signature matching are described below.

`1. Bit-Level Signature Matching:` Bit-level signature matching is done by comparing the bits of payload to bit signatures. As mentioned previously, classification happens in real-time and is performance critical operation; thus needs to be efficient with a minimum number of operations. The mask bit sequences, along with signature bits, help in improving the operational speed by optimizing bit sequence comparisons with word-level bitwise operations. Signature matching happens on the manipulated bit sequences of signature and test bit (input) sequence, as detailed below.

(1) *Signature′* bit-sequence Generation: Let *Signature* be the signature bit sequence (either common bit-signature or extended bit signature. The signature bits are complemented (every bit is toggled) to get ∼*Signature* representing the complement bit sequence. This is bitwise ORed with Mask-1 bits to generate *Signature′* bit sequence as shown in Fig. 5.
(2) *Input′* Bit-Sequence Generation: Input bit sequence extracted from test payload is bitwise ANDed with Mask-2 bit sequence to generate *Input′* bit sequence as shown in Fig. 6.

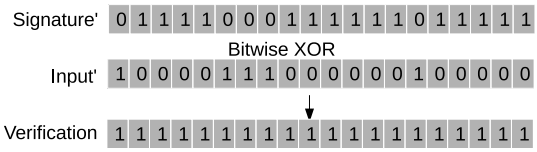


Fig. 7. Final bit sequence generated with bitwise XOR of Signature′ and Input′ bit sequences.

Table 1
Bitwise operations required for signature comparison

S. no	Operation	Details	Online/offline	Frequency
1	Complement	Signature bits are complemented	Offline	Once
2	OR	Bitwise OR of Mask-1 with complement of signature	Offline	Once
3	AND	Bitwise AND of test bit sequence with Mask-2	Online	Once per test flow
4	XOR	Bitwise XOR of output of Step-2 with output of Step-3	Online	Once per test flow

(3) Final Evaluation Bit Sequence: Final evaluation bit sequence is generated with bitwise XOR operation of *Signature′* and *Input′*. The input bit string matches with signature if the final evaluation bit string has all 1's as shown in Fig. 7; otherwise, it does not match with signature bits.[1]

These bitwise operations help us implement signature matching efficiently on commodity hardware architectures. For example, a 64-bit word architecture can do 64-bit operations with the execution of one instruction. This will help in verifying all invariant bits up to 64 bits in a single operation instead of checking them one by one. Thus any signature length which is multiple of 64 can be implemented with a sequence of bitwise operations. Table 1 lists the number of bitwise operations required to verify a one *Wordlength* bit signature. We can notice that, out of the four operations required, two are offline and can be done as part of signature generation once, and the other two are done in online mode (real-time) for every test flow. Thus *Signature′* can be computed offline for every application signature and used. For a signature that is $X \times$ *Wordlength* of target processor architecture, X number of online operations are required. This linear increase in the number of bitwise operations required enables the scaling of signature evaluation. While working with larger signature lengths divided into equal-sized bits of *Wordlength*, further optimizations can be done. In this, every *Wordlength* number of bits can be compared sequentially, and if Kth *Wordlength* bit sequence of test flow does not match with Kth *Wordlength* signature bit sequence, all subsequent comparison operations can be skipped.

`2. Keyword Signature Matching:` If the common portion of bit sequences indicates that the potential protocol is a text protocol, then the keyword-based signatures are used for identifying the application. The keywords within these signatures are ordered and are matched against the payload in the same order. Unlike the extended bit-level signatures, here the keyword-based matching is done from the beginning of the payload (including the first 'n' bits used for common bit signature) for the applications identified as potential matches in the first stage. Keyword searching can be done with any efficient string matching algorithm. Here, we use Boyer–Moore [5] string matching algorithm for comparing the test payload with the keywords of an application.

4. Experiments and evaluation

In this section, we describe the experiments and evaluation done to assess the performance of the proposed bit-level and keyword-level signature matching technique `HClass`. Datasets used for experiments and evaluation done are elaborated in the subsequent paragraphs.

[1] Mask-1 and Mask-2 values are the complements of each other at $*$ bit positions, hence result into a bit values of one with XOR operation.

Table 2
Dataset-1 details

S. no	Protocol	Flows training	Flows testing
	Binary protocols		
1	Bootp	81	86
2	DNS	25469	25850
3	NBNS	982	982
4	NTP	201	201
5	BACnet	9	11
6	BJNP	34	38
7	QUIC	106	90
	Text-based protocols		
8	MWBP	8	8
9	CUPS	47	45
10	HTTP	13233	17964
11	SMTP	520	521
	Total	**40690**	**45796**

4.1. Dataset details and implementation

For our experiments with HClass, network traffic traces were collected from different sources with ground truth. We collected network traffic traces from two public repositories, 'Digital Corpora' [10], and NETRESEC [22], and we also generated few application traces in the network security lab at IIT Indore. The entire collection of traces is organized into three datasets. Dataset-1 is from the Digital Corpora homepage under the link heading "Network Packet Dumps". Dataset-2 is collected from Research PCAP Dataset published by 'FOI Information Warfare Lab (FOI is The Swedish Defence Research Agency)'. This dataset is hosted by NETRESEC under the heading Resources and subheading Uncategorized PCAPs, 21st dataset in the list. Dataset-3 contains the network traffic generated in the network security lab at IIT Indore, and few application traffic were taken from Digital Corpora and FOI Information Warfare Lab sources. Packets in these traces are filtered with Wireshark [34] tool and saved as application-specific traces. Each application protocol trace is divided into two parts for training and testing. The training part was used to generate bit-level and keyword-based signatures, and the testing part was used to evaluate the classification performance of HClass. The statistics of flows pertaining to different applications and their count in training and testing portions in Dataset-1, Dataset-2, and Dataset-3 are shown in Table 2, Table 3, and Table 4, respectively. We can notice from these three tables that training and testing portions contain roughly equal number of flows.

Implementation and Evaluation: We implemented HClass in C language using LibPcap programming library [29]. Flow reconstruction in TCP is done by aggregating all packets between SYN and FIN/RST packets sharing five common attributes. In the case of UDP, flows are generated by aggregating all packets sharing common attributes using the threshold ζ set to six seconds. All the experiments reported in this paper are performed on a machine with an Intel Core i7-8565U 8-core processor running at a clock speed of 1.80 GHz having 16 GB RAM, and running Ubuntu 20.04 LTS operating system.

For the evaluation of HClass, we used flows from the training datasets to generate bit-level and keyword-based signatures. In the Experiments, we used the first 64 bits from the labeled application flows to generate the common bit-level signatures for each labeled application(both binary and text-based) in the training datasets. Extended bit-level signature for binary protocols is generated considering the next 64 bits of data from the flow payload of labeled applications. For generating keyword-based signatures, the labeled flows from each application are grouped based on similarity, and meaningful keywords with a frequency of occurrence greater than 70% across all flows are selected as candidates for keywords. These keywords are arranged in order and stored along with their offset

Table 3
Dataset-2 details

S. no	Protocol	Flows training	Flows testing
	Binary protocols		
1	Bootp	91	91
2	DNS	956	958
3	Kerberos	670	673
4	NBNS	289	290
5	NBSS	377	373
6	NTP	202	200
7	RPC	7	7
	Text-based protocols		
8	GSMIPA	9	9
9	HTTP	253	250
10	POP	56	57
	Total	**2903**	**2908**

Table 4
Dataset-3 details

S. no.	Protocol	Flow training	Flows testing
	Binary protocols		
1	SMB	3141	3111
2	SSH	1104	1000
3	CLDAP	910	918
	Text-based protocols		
4	BitTorrent	608	946
5	DropBox	788	1518
6	HTTP	44452	107428
7	POP	497	542
8	SIP	429	650
9	SMTP	1163	940
	Total Flows	**53092**	**117053**

values to generate keyword-based signatures. We considered the payload up to 1024 bytes for keyword extraction. We evaluate the performance of HClass by conducting three studies as elaborated in Section 4.2, Section 4.3 and Section 4.4.

4.2. Classification performance comparison with other DPI based techniques

In this subsection, we describe the experiments performed to evaluate the classification performance of HClass and compare its performance with BitCoding, a bit-level signature-based classifier, and KeyClass, a keyword-based classifier. BitCoding performs bit-by-bit signature matching against a new test payload to identify the application to which the flow belongs. KeyClass, on the other hand, uses a set of keyword-based signatures to classify the network traffic. It operates in two phases. In the first phase, the potential application matches against the test payload are identified quickly using the Aho–Corasick FSM constructed with the first keyword of each application. In the second phase, rest of the keywords of the potential application are searched in the payload using the Boyer–Moore

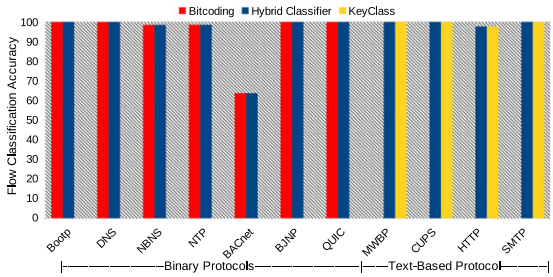


Fig. 8. Flow classification accuracy comparison for HClass, BitCoding, and KeyClass for Dataset-1.

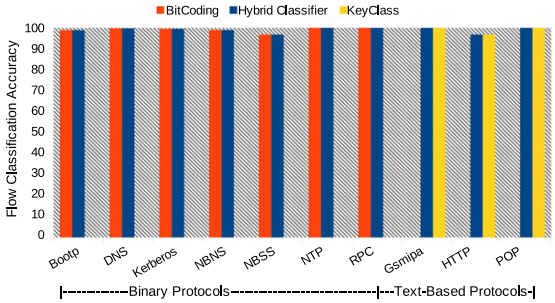


Fig. 9. Flow classification accuracy comparison for HClass, BitCoding, and KeyClass for Dataset-2.

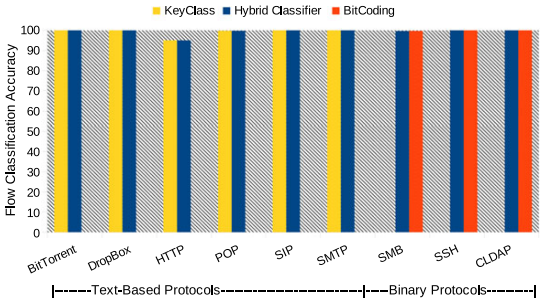


Fig. 10. Flow classification accuracy comparison for HClass, BitCoding, and KeyClass for Dataset-3.

string matching algorithm along with byte offsets of the keywords. We evaluated HClass for binary and text-based protocols and compared it against BitCoding for binary protocols and KeyClass for text-based protocols for each dataset.

Figures 8, 9 and 10 show the classification accuracy comparison between HClass, BitCoding, and KeyClass for Dataset-1, Dataset-2, and Dataset-3, respectively. We can notice from the figures that HClass(for both binary and text-based protocols), BitCoding(for binary protocols), and KeyClass(For text-based protocols) show equal performance in classification accuracy; however, the limitation of KeyClass is that it can only classify text based protocols with greater accuracy but not the binary protocols, and BitCoding is computationally expensive as it matches one bit at a time, unlike HClass that performs word-level signature matching. The details on the execution time comparisons for all three methods are discussed in Section 4.3

For clarity on the classification performance of HClass, we also represented the Classification performance of HClass in terms of confusion matrices as shown in Table 5, 6, and 7 on Dataset-1, Dataset-2, and Dataset-3 respectively. Confusion matrices give insights into the details of flows classified for each application and helps to understand any misclassification and unclassified instances of test flow in the network traffic. In these matrices,

Table 5
Flow classification confusion matrix for Dataset-1

	Bootp (86)	DNS (25850)	NBNS (982)	NTP (201)	BACnet (11)	BJNP (38)	QUIC (90)	MWBP (8)	CUPS (45)	HTTP (17964)	SMTP (521)
Bootp	86	0	0	0	0	0	0	0	0	0	0
DNS	0	25842	0	0	0	0	0	0	0	0	0
NBNS	0	0	967	0	0	0	0	0	0	0	0
NTP	0	0	14	198	0	0	0	0	0	0	0
BACnet	0	0	0	0	7	0	0	0	0	0	0
BJNP	0	0	0	0	0	38	0	0	0	0	0
QUIC	0	0	0	0	0	0	90	0	0	0	0
MWBP	0	0	0	0	0	0	0	8	0	0	0
CUPS	0	0	0	0	0	0	0	0	45	0	0
HTTP	0	0	0	0	0	0	0	0	0	17551	0
SMTP	0	0	0	0	0	0	0	0	0	0	521
Unclassified	0	8	1	3	4	0	0	0	0	413	0

Table 6
Flow classification confusion matrix for Dataset-2

	Bootp (91)	DNS (958)	Gsmipa (9)	HTTP (250)	Kerberos (673)	NBNS (290)	NBSS (373)	NTP (200)	POP (57)	RPC (7)
Bootp	90	0	0	0	0	0	0	0	0	0
DNS	0	956	0	0	0	0	0	0	0	0
Gsmipa	0	0	9	0	0	0	0	0	0	0
HTTP	0	0	0	242	0	0	0	0	0	0
Kerberos	0	0	0	0	670	0	0	8	0	0
NBNS	0	0	0	0	0	287	0	0	0	0
NBSS	0	0	0	0	0	0	361	0	0	0
NTP	0	0	0	0	0	0	0	192	0	0
POP	0	0	0	0	0	0	0	0	57	0
RPC	0	0	0	0	0	0	0	0	0	7
Unclassified	1	2	0	8	3	3	12	0	0	0

the rows and columns header represent the protocols, and the number within the bracket in the column header represents the number of test flows used for classification. Each entry (other than the header) in the matrix denotes the number of flow instances classified as the protocol in the row header out of total the number of flows indicated in the column header. For example, in Table 5, the entry in the fourth row and fourth column denotes out of 982 NBNS protocol flows 967 flows are classified as NBNS, and the entry in the fourth column and fifth row represents 14 NBNS flows misclassified as NTP and the entry at the bottom of the column represents one unclassified instance of NBNS protocol. We can notice from the table that there are very few misclassifications and some unclassified flows. These unclassified instances are mainly the cases where a common portion of the signature fails to generate possible matches, or there is a mismatch during extended bit-level signature matching for binary protocols or the desired keyword is not found for text-based protocol verification.

4.3. Execution time performance comparison with other DPI based techniques

We experimented with HClass and compared the execution time performance of HClass with BitCoding and KeyClass on the three datasets mentioned in the Section 4.1. The time comparison results for these experiments are shown in Table 8, 9, and 10 for Dataset-1, Dataset-2, and Dataset-3 respectively. In these tables, we reported

Table 7
Flow classification confusion matrix for Dataset-3

	BitTorrent (946)	DropBox (1518)	HTTP (107428)	POP (542)	SIP (650)	SMTP (940)	SMB (3111)	SSH (1000)	CLDAP (918)
BitTorrent	946	0	0	0	0	0	0	0	0
DropBox	0	1518	0	0	0	0	0	0	0
HTTP	0	0	102083	0	0	0	0	0	0
POP	0	0	0	541	0	0	0	0	0
SIP	0	0	0	0	650	0	0	0	0
SMTP	0	0	0	0	0	940	0	0	0
SMB	0	0	0	0	0	0	3102	0	0
SSH	0	0	0	0	0	0	0	1000	0
CLDAP	0	0	0	0	0	0	0	0	918
Not classified	0	0	5345	1	0	0	9	0	0

Table 8
Execution time (in sec) for Dataset-1

	HClass User time	Bitcoding User time	KeyClass User time
Bootp	0.005	0.071	NA
DNS	4.414	4.622	NA
NBNS	0.04	0.095	NA
NTP	0.007	0.016	NA
BACnet	0.003	0.004	NA
BJNP	0.003	0.004	NA
QUIC	0.003	0.003	NA
MWBP	0.005	NA	0.009
CUPS	0.005	NA	0.006
HTTP	2.485	NA	2.486
SMTP	0.053	NA	0.053

the user time for each protocol on all three datasets. As we evaluated BitCoding only for binary protocols and KeyClass only for the text-based protocols, the corresponding entries for binary protocols under the KeyClass column header are not applicable(NA). Similarly, the entries corresponding to text-based protocols under Bitcoding column headers are not relevant and hence are marked as NA. We can notice from Table 8, 9, and 10 that HClass outperformed BitCoding and showed a significant performance improvement over the BitCoding classifier. BitCoding performs bit-by-bit signature matching, which is computationally expensive and hence becomes the performance bottleneck. In contrast, HClass optimizes the bit signature comparisons and aligns it to word-level operations, thereby speeding up the signature matching. HClass also showed better performance in execution time compared to KeyClass for most of the protocols; however, there are a few cases where they showed nearly equal execution time, like in Table 8 for POP and SIP protocols. This is mainly due to the small number of flow instances for these protocols. There might be cases where the first keyword for such protocols may fall under a few initial bytes of the payload and performs as good as common bit-level signatures. For example, the keyword "GET" for HTTP protocol appears at the beginning of the payload and can also be part of common bit-level signatures.

4.4. Performance comparison with machine learning based classification

As many of the recent works use machine learning techniques for traffic classification, we also compare the performance of HClass with another recent work [24] which uses machine learning methods for traffic classification.

Table 9
Execution time (in sec) for Dataset-2

	HClass User time	BitCoding User time	KeyClass User time
Bootp	0.003	0.022	NA
DNS	0.025	0.088	NA
Kerberos	0.014	0.241	NA
NBNS	0.023	0.060	NA
NBSS	0.025	0.366	NA
NTP	0.016	0.069	NA
RPC	0.003	0.038	NA
Gsmipa	0.002	NA	0.003
HTTP	0.020	NA	0.036
POP	0.003	NA	0.004

Table 10
Execution time (in sec) for Dataset-3

	HClass (User time)	BitCoding (User time)	KeyClass (User time)
BitTorrent	0.275	NA	0.278
DropBox	1.339	NA	1.346
HTTP	13.158	NA	13.241
POP	0.024	NA	0.025
SIP	0.598	NA	0.599
SMTP	0.125	NA	0.133
SMB	0.25	1.748	NA
SSH	0.08	0.517	NA
CLDAP	0.053	0.055	NA

Table 11
Performance comparison of HClass with Hussein et al. [24]

Method	Accuracy
HClass	99.03
Hussein et al. [24]	94.35

It uses a set of statistics extracted from bursts of communication as features and subsequently feeds into a machine learning algorithm, namely C5.0 decision tree algorithm. A burst is simply a duration within which all packets in either direction are aggregated. A Number of features like packet count, byte count, ratio of packet count, average burst size, and inactive/idle time for each direction are used as features. While similar features are usually used in almost all machine learning based traffic classification methods, the selection of these statistics as features within a burst is based on the premise that different applications exhibit different distributions of bursts and hence can be separated.

For our experiments, we extracted the features as identified by Hussein et al. [24] from our dataset and compared the performance in terms of accuracy. We evaluated the performance of the classifiers based on "Accuracy" metric as defined in equation (1). Table 11 shows this performance comparison across the applications for Dataset-1. Table 12 shows the confusion matrix for the classification performance of C5.0 algorithm. We can notice that there are large number of misclassifications (entries shown in red color) among HTTP and DNS protocols. Similar

Table 12
Classification performance of Hussein et al. [24] on Dataset-1

Reference(→)	BACnet	BJNP	Bootp	CUPS	DNS	HTTP	MWBP	NBNS	NTP	QUIC	SMTP
Prediction(↓)	(12)	(32)	(81)	(31)	(41087)	(23904)	(9)	(822)	(281)	(106)	(825)
BACnet	4	0	0	0	0	0	0	0	0	0	0
BJNP	0	32	0	0	0	0	0	0	0	1	0
Bootp	0	0	32	0	1	0	0	1	0	0	0
CUPS	0	0	0	29	0	3	1	5	0	0	0
DNS	4	0	45	1	39848	2225	2	69	21	31	2
HTTP	0	0	0	1	1078	21618	2	0	0	0	69
MWBP	0	0	0	0	2	0	4	0	0	0	0
NBNS	1	0	4	0	116	2	0	741	0	1	0
NTP	0	0	0	0	34	0	0	5	257	0	0
QUIC	0	0	0	0	0	0	0	0	0	73	0
SMTP	3	0	0	0	8	54	0	1	3	0	754

results were obtained on the other two datasets which we omit showing here.

$$\texttt{Accuracy} = \frac{\texttt{Number of Correct Predictions}}{\texttt{Total Number of Predictions}} \tag{1}$$

Apart from the obvious low accuracy in the classification, the work of Hussein et al. [24] also shares similar limitations where it assumes that the applications of distinct classes can exhibit distinct burst characteristics. We believe that, as the number of applications increase, it is hard to find such well separated characteristics. For e.g a large number of video streaming applications like youtube, amazon prime, etc may have overlaps. Similarly mobile applications have very large set running into millions in count. These scenarios prevent the machine learning based methods from scaling. To the best of our knowledge there are no studies which have validated these algorithms on large scale experiments.

5. Conclusion

In this paper, we presented HClass, a DPI-based network traffic classifier that classifies both binary and text-based protocols. Binary protocols are classified using bit-level signatures, and text-based protocols use both bit-level signatures for initial screening and keyword-based signatures subsequently for application detection. HClass uses bit-level signatures with masks to perform bitwise operations on word length payload content and hence is computationally efficient. It can be run on commodity hardware by aligning the signature length to the word boundary. We experimented with three datasets and showed that HClass classification performance is reasonable and has a lower execution time compared to other methods. We are currently working on using these bit-level operations for other regular expression-based pattern matching and evaluating whether they result in performance optimizations.

Acknowledgement

This research is financially supported under the Indo-Norway SPARC project through grant number SPARC/2018-2019/P448/SL by the Government of India and also by a fellowship given by the Tata Consultancy Services(TCS) foundation to the first author.

Conflict of interest

Authors do not have any potential conflicts of interest with any one with the material included in this paper.

References

[1] G. Aceto, D. Ciuonzo, A. Montieri and A. Pescapè, MIMETIC: Mobile encrypted traffic classification using multimodal deep learning, *Computer Networks* **165** (2019), 1–12. doi:10.1016/j.comnet.2019.106944.

[2] A. Azab, M. Khasawneh, S. Alrabaee, K.K.R. Choo and M. Sarsour, Network traffic classification: Techniques, datasets, and challenges, *Digital Communications and Networks* (2023). doi:10.1016/j.dcan.2022.09.009.

[3] A. Bashir, C. Huang, B. Nandy and N. Seddigh, Classifying P2P activity in netflow records: A case study on BitTorrent, in: *ICC'13: Proceedings of the IEEE International Conference on Communications*, 2013, pp. 3018–3023. doi:10.1109/ICC.2013.6655003.

[4] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule and K. Salamatian, Traffic classification on the fly, *SIGCOMM Computer Communication Review* **36**(2) (2006), 23–26. doi:10.1145/1129582.1129589.

[5] R.S. Boyer and J.S. Moore, A fast string searching algorithm, *Communications of the ACM* **20**(10) (1977), 762–772. doi:10.1145/359842.359859.

[6] T. Bujlow, T. Riaz and J.M. Pedersen, A method for classification of network traffic based on C5.0 machine learning algorithm, in: *ICNC'12: Proceedings of the International Conference on Computing, Networking and Communications*, 2012, pp. 237–241. doi:10.1109/ICCNC.2012.6167418.

[7] N. Cascarano, A. Este, F. Gringoli, F. Risso and L. Salgarelli, An experimental evaluation of the computational cost of a DPI traffic classifier, in: *GLOBECOM'09: Proceedings of the 2009 IEEE Global Telecommunications Conference*, IEEE, 2009, pp. 1–8. doi:10.1109/GLOCOM.2009.5425469.

[8] B. Choi, J. Chae, M. Jamshed, K. Park and D. Han, DFC: Accelerating string pattern matching for network applications, in: *NSDI'16: Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, USENIX Association, 2016, pp. 551–565. doi:10.5555/2930611.2930647.

[9] G. D'Angelo and F. Palmieri, Network traffic classification using deep convolutional recurrent autoencoder neural networks for spatial–temporal features extraction, *Journal of Network and Computer Applications* **173** (2021), 102890. doi:10.1016/j.jnca.2020.102890.

[10] Digital Corpora, Accessed on 12-08-2023, http://digitalcorpora.org/.

[11] S.E. Gómez, B.C. Martínez, A.J. Sánchez-Esguevillas and L.H. Callejo, Ensemble network traffic classification: Algorithm comparison and novel ensemble scheme proposal, *Computer Networks* **127** (2017), 68–80. doi:10.1016/j.comnet.2017.07.018.

[12] N. Hubballi and P. Khandait, KeyClass: Efficient keyword matching for network traffic classification, *Computer Communications* **185** (2022), 79–91. doi:10.1016/j.comcom.2021.12.021.

[13] N. Hubballi and M. Swarnkar, BitCoding: Protocol type agnostic robust bit level signatures for traffic classification, in: *GLOBECOM 2017 – 2017 IEEE Global Communications Conference*, IEEE, 2017, pp. 1–6. doi:10.1109/GLOCOM.2017.8254001.

[14] N. Hubballi and M. Swarnkar, BitCoding: Network traffic classification through encoded bit level signatures, *IEEE/ACM Transaction on Networking* **26**(5) (2018), 2334–2346. doi:10.1109/TNET.2018.2868816.

[15] N. Hubballi, M. Swarnkar and M. Conti, BitProb: Probabilistic bit signatures for accurate application identification, *IEEE Transactions on Network and Service Management* **17**(3) (2020), 1730–1741. doi:10.1109/TNSM.2020.2999856.

[16] Y. Jin, N. Duffield, J. Erman, P. Haffner, S. Sen and Z.L. Zhang, A modular machine learning system for flow-level traffic classification in large networks, *ACM Transactions on Knowledge Discovery Data* **6**(1) (2012). doi:10.1145/2133360.2133364.

[17] V. Jyothi, S.K. Addepalli and R. Karri, DPFEE: A high performance scalable pre-processor for network security systems, *IEEE Transactions on Multi-Scale Computing Systems* **4**(1) (2018), 55–68. doi:10.1109/TMSCS.2017.2765324.

[18] T. Karagiannis, K. Papagiannaki and M. Faloutsos, BLINC: Multilevel traffic classification in the dark, in: *SIGCOMM '05: Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Association for Computing Machinery, 2005, pp. 229–240. doi:10.1145/1080091.1080119.

[19] P. Khandait, N. Hubballi and B. Mazumdar, Efficient keyword matching for Deep Packet Inspection based network traffic classification, in: *COMSNETS'20: International Conference on Communication Systems NetworkS*, IEEE, 2020, pp. 567–570. doi:10.1109/COMSNETS48256.2020.9027353.

[20] Y. Li, Y. Huang, S. Seneviratne, K. Thilakarathna, A. Cheng, G. Jourjon, D. Webb, D.B. Smith and R.Y.D. Xu, From traffic classes to content: A hierarchical approach for encrypted traffic classification, *Computer Networks* **212** (2022), 109017. doi:10.1016/j.comnet.2022.109017.

[21] C. Liu, L. He, G. Xiong, Z. Cao and Z. Li, FS-net: A flow sequence network for encrypted traffic classification, in: *INFOCOM'19: Proceedings of the IEEE Conference on Computer Communications*, IEEE, 2019, pp. 1171–1179. doi:10.1109/INFOCOM.2019.8737507.

[22] Netresec, Accessed on 25-08-2023, http://www.netresec.com/?page=pcapfiles.

[23] T.T.T. Nguyen and G. Armitage, A survey of techniques for Internet traffic classification using machine learning, *IEEE Communications Surveys & Tutorials* **10**(4) (2008), 56–76. doi:10.1109/SURV.2008.080406.

[24] H. Oudah, B. Ghita and T. Bakhshi, A novel features set for Internet traffic classification using burstiness, in: *ICISSP'19: In Proceedings of the 5th International Conference on Information Systems Security and Privacy*, SCITEPRESS – Science and Technology Publications, Lda., 2019, pp. 397–404. doi:10.5220/0007384203970404.

[25] H. Ren, H. Litt, D. Liu and X.S. Shen, Toward efficient and secure Deep Packet Inspection for outsourced middlebox, in: *ICC '19 – Proceedings of IEEE International Conference on Communications*, 2019, pp. 1–6. doi:10.1109/ICC.2019.8761954.

[26] M. Roughan, S. Sen, O. Spatscheck and N. Duffield, Class-of-service mapping for QoS: A statistical signature-based approach to IP traffic classification, in: *IMC '04: Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, ACM, 2004, pp. 135–148. doi:10.1145/1028788.1028805.

[27] G. Sun, L. Liang, T. Chen, F. Xiao and F. Lang, Network traffic classification based on transfer learning, *Computers & Electrical Engineering* **69** (2018), 920–927. doi:10.1016/j.compeleceng.2018.03.005.

[28] M. Swarnkar and N. Hubballi, RDClass: On using relative distance of keywords for accurate network traffic classification, *IET Networks* **7**(4) (2018), 273–279. doi:10.1049/iet-net.2017.0065.

[29] TCPDUMP, Accessed on 20-08-2023, https://www.tcpdump.org/pcap.html.

[30] A. Tongaonkar, R. Torres, M. Iliofotou, R. Keralapura and A. Nucci, Towards self adaptive network traffic classification, *Computer Communication* **56**(1) (2015), 35–46. doi:10.1016/j.comcom.2014.03.026.

[31] P. Wang, S.C. Lin and M. Luo, A framework for QoS-aware traffic classification using semi-supervised machine learning in SDNs, in: *2016 IEEE International Conference on Services Computing (SCC)*, IEEE, 2016, pp. 760–765. doi:10.1109/SCC.2016.133.

[32] X. Wang, Y. Hong, H. Chang, K. Park, G. Langdale, J. Hu and H. Zhu, Hyperscan: A fast multi-pattern regex matcher for modern CPUs, in: *NSDI'19: Proceedings of the 16th USENIX Conference on Networked Systems Design and Implementation*, USENIX Association, 2019, pp. 631–648. doi:10.5555/3323234.3323286.

[33] Y. Wang, X. Yun, M.Z. Shafiq, L. Wang, A.X. Liu, Z. Zhang, D. Yao, Y. Zhang and L. Guo, A semantics aware approach to automated reverse engineering unknown protocols, in: *ICNP '12: Proceedings of the 2012 IEEE International Conference on Network Protocols*, IEEE Computer Society, 2012, pp. 1–10. doi:10.1109/ICNP.2012.6459963.

[34] Wireshark, Accessed on 28-08-2023, https://www.wireshark.org.

[35] Z. Yuan, J. Xu, Y. Xue and M.V.D. Schaar, Bits learning: User-adjustable privacy versus accuracy in Internet traffic classification, *IEEE Communications Letters* **20**(4) (2016), 704–707. doi:10.1109/LCOMM.2016.2521837.

[36] Z. Yuan, Y. Xue and M.V.D. Schaar, BitMiner: Bits mining in Internet traffic classification, in: *SIGCOMM '15: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ACM, 2015, pp. 93–94. ISBN 9781450335423. doi:10.1145/2785956.2789997.

[37] X. Yun, Y. Wang, Y. Zhang and Y. Zhou, A semantics-aware approach to the automated network protocol identification, *IEEE/ACM Transactions on Networking* **24**(1) (2016), 583–595. doi:10.1109/TNET.2014.2381230.

[38] J. Zhang, Y. Xiang, W. Zhou and Y. Wang, Unsupervised traffic classification using flow statistical properties and IP packet payload, *Journal of Computer and System Sciences* **79**(5) (2013), 573–585. doi:10.1016/j.jcss.2012.11.004.