Computer Networks 166 (2020) 106943

Contents lists available at ScienceDirect

Computer Networks

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

ACoPE: An adaptive semi-supervised learning approach for complex-policy enforcement in high-bandwidth networks

Morteza Noferesti, Rasool Jalili*

Data and Network Security Lab, Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 4 April 2019
Revised 3 October 2019
Accepted 4 October 2019
Available online 5 October 2019

Keywords:
High-bandwidth network analyzing
Data stream processing
Complex-policy
Semi-supervised learning
Adaptive learning

ABSTRACT

Today's high-bandwidth networks require adaptive analyzing approaches to recognize the network variable behaviors. The analyzing approaches should be robust against the lack of prior knowledge and provide data to impose more complex policies. In this paper, ACoPE is proposed as an adaptive semi-supervised learning approach for complex-policy enforcement in high-bandwidth networks. ACoPE detects and maintains inter-flows relationships to impose complex-policies. It employs a statistical process control technique to monitor accuracy. Whenever the accuracy decreased, ACoPE considers it as a changed behavior and uses data from a deep packet inspection module to adapt itself with the change.

The performance of ACoPE in analyzing network traffic is evaluated through UNB ISCX VPN-nonVPN and UNB ISCX Tor-nonTor datasets. The performance is compared with 10 different stream and traditional classification algorithms. ACoPE outperforms the stream classifiers, with 95.92% accuracy, 86.21% precision, and 73.29% recall in VPN dataset, and with 81.12% accuracy, 73.59% precision, and 61.08% recall in Tor dataset. The effectiveness of ACoPE to address the main constraints in analyzing of high-bandwidth networks to enforce security policies, namely *comprehensive processing* and *adaptive learning*, are confirmed through three different scenarios. Efficiency and accuracy of ACoPE in real high-bandwidth networks are evaluated by a pilot study, which indicates its efficiency and accuracy in analyzing high-bandwidth networks.

1. Introduction

As the size and complexity of networks increase, new challenges appear to analyze and manage network activities. According to the Cisco network forecasting report [1], the annual global traffic will exceed two zeta-byte in 2020. Real-time high-bandwidth network traffic analyzing is crucial for security services such as attack patterns and denial of service attack detection [2]. It is also essential for network management services such as tariff management which requires different applications detection accurately across the bandwidth.

Network administrators manage and control the traffic regarding network policies. The policies are defined in the form of "IF *condition* THEN *action*". When *condition* is held, *action* is performed by security mechanism units such as firewalls, intrusion prevention systems, or privileged access management systems. Security policy enforcement tools analyze high-bandwidth network traffics and maintain network behavior state to check the *condition*s.

Traditionally, *condition*s have been checked using the well-known ports numbers. Quantifying network traffic on major TCP and UDP ports, Alcock et al. in [3], confirmed that port-based approaches are ineffective. Traffic deep packet inspection methods, known as DPI techniques, are proposed as an alternative. The techniques looking for application-specific data, called signature, within traffic's payload. These approaches rely on two related assumptions, visibility of payloads and the syntax of known application packets payload. Although DPI-based analysis is claimed to achieve high accuracy, the assumptions are not valid in real networks.

The limitations of port-based and payload-based approaches have motivated the use of machine-learning techniques for traffic analysis [4]. Using statistical characteristics has lower computational cost than payload inspection, and it can apply on encrypted traffics. Current machine-learning-based methods can be classified in supervised, unsupervised, and semi-supervised.

The common supervised approaches rely on the Bayesian Decision Theory. According to the theory, assigning a label l to a flow f can be described by the prior probabilities of the labels $p(l)$ and the label conditional probability density function $p(f|l)$ for all possible labels $l \in LabelSet$. The classification decision performed accord-

* Corresponding author.
E-mail addresses: jalili@sharif.edu, jalili@sharif.ir (R. Jalili).

https://doi.org/10.1016/j.comnet.2019.106943
1389-1286/

ing to the posterior probabilities of labels, which represented as follows for label l

$$p(l|f) = \frac{p(l)p(f|l)}{p(f)} \quad (1)$$

where $p(f) = \sum_{l \in LabelSet} p(l)p(f|l)$. In dynamically changing environments such as high-bandwidth networks, the underlying distributions can change dynamically over time. In this environment, adaptive models require mechanisms to detect and handle the evolving data over time and to forget old irrelevant data.

The unsupervised learning approaches cluster traffic flows into groups that have similar patterns. Basically, three main clustering methods have been used in the network traffic clustering literature [5–8]: (1) the classic k-means algorithm that partitions instances into disjoint clusters (2) the clustering algorithms that generates a hierarchical grouping of instances, (3) the probability based method that assigns instances into classes probabilistically, not deterministically. Some researches try to adopt unsupervised learning techniques to recognize malicious nodes in wireless sensor networks [9], or to analyze new environments such as smart home [10,11], or to share knowledge in social networks [12].

Semi-supervised approaches fall between unsupervised and supervised approaches. These approaches use a large amount of unlabeled flows, together with labeled flows, to build a better classifier [13,14]. They are robust and can handle both previously unseen applications and behavior changed of existing applications [13,15]. Furthermore, semi-supervised approaches update the classifier by adding unlabeled flows to increase the classifier's performance [16]. The k-means and kNN algorithms are the most frequent in this category [14]. The popularity of k-means is due to its variability, which allows it to be fine-tuned for various purposes. It is often combined with genetic algorithms to find the best setting [14].

Semi-supervised learning has been used widely in network security mechanisms [17–19]. In intrusion detection techniques, Chen et al. in [20] demonstrated that custom semi-supervised algorithms were able to outperform both supervised and unsupervised learning methods during classifying unknown attack types. Wang et al. in [18] demonstrated a significant performance improvement when using semi-supervised learning to perform network traffic clustering.

High-bandwidth networks create dynamic environments, where incoming traffic flows are forming a potentially infinite stream with unexpected changes. Moreover, unbounded high-bandwidth network traffic requires that each packet be processed once (sometimes called one-pass processing restriction). Consequently, the analyzing approaches for security policy enforcement in high-bandwidth networks should satisfy the followings:

- *Comprehensive Processing*: Regarding the one-pass processing restriction, all the data required to check the *conditions* of security policies should be accumulated by only one processing mechanism.
- *Adaptive Learning*: The analyzing approaches should be periodically updated to adapt to changed behaviors, to detect new traffic behaviors, and to forget the expired learning information.

This paper introduces a complex-policy relation which comprehensively defines all the administrators' requirements. The *conditions* of complex policies are specified by a between-flows relation. The requirements from a simple access policy to a distributed denial-of-service attack detection can be expressed as a between-flows relation. Maintaining the relation to complex policy enforcement in high-bandwidth networks is a challenging task due to the high volume and dynamic nature of the traffic.

This paper proposes ACoPE as an **A**daptive semi-supervised learning approach for **Co**mplex-**P**olicy **E**nforcement in High-Bandwidth Networks. ACoPE maintains statistics and information about the recent traffic in the abstract data structure called meta-flow. It divides the meta-flows into three states which are named *in-control*, *warning*, or *out-of-control*. Utilizing the statistical process control technique, ACoPE monitors and controls states of the traffic. It uses applications detected by the DPI, Deep Packet Inspection technique, as ground truth data and updates the meta-flows accordingly. ACoPE accumulated all data required to check the *conditions* of complex policies by only one inline processing mechanism (*comprehensive* property). It also labels recent flows by the inline DPI technique and updates the network traffic states (*adaptive* property).

We provide two main contributions in this paper.

(1) Complex-policy relation is proposed and defined formally. The relation affords a *comprehensive* set of policies, where all administrators' requirements could represent by complex-policies. The complex-policy relation could be learned incrementally, to consider the evolution of traffic over time.
(2) An adaptive high-bandwidth network behavior detection approach is proposed based on the statistical process control technique. The approach monitors and controls the evolving traffic of arriving flows and maintains the behavior state to enforce complex-policies.

The rest of this paper is organized as follows. An overview of practical methodologies employed in the high-bandwidth network analysis is given in Section 2. Complex-policy relation is defined in Section 3. The adaptive learning algorithm in high-bandwidth networks, ACoPE, is introduced in Section 4. Empirical simulation and real-network implementation of ACoPE are explained in Section 5. Finally, we indicate the ongoing challenges and discuss the future directions of this area in Section 6.

2. Related works

Adaptive traffic analyzing for security policy enforcement in today's high-bandwidth networks is challenging due to the high volume, velocity, and variety characteristics of the networks traffics. Utilizing adaptive machine learning algorithms and comprehensive processing of the high-bandwidth traffic have not been much addressed in the literature.

Mahdavi et al. in [16] proposed a semi-supervised approach to classify traffic of encrypted applications. Graph theory and minimum spanning tree algorithm are used to make clusters over the traffic. Afterward, the labeled data instances are used for labeling clusters. The approach provides a precise and accurate performance in classification of encrypted traffic but is not adapted with the network traffic changes. The approach does not support *incremental learning* and *comprehensiveness* required for complex-policy enforcement in high-bandwidth networks.

The infinite stream of incoming traffic forms a dynamic environment with unexpected changes which requires analyzing approaches to satisfy the high-bandwidth network processing challenges such as incremental learning, inline processing, and outlier handling. Noferesti and Jalili in [21], represented a formal definition of a behavior detection system for high-bandwidth networks. Considering the behavior detection system constraints, the authors proposed an approach where a data structure called meta-event is defined whose inter-relationships are used to mine end-user behaviors.

Loo and Marsono in [22] proposed an incremental data stream mining algorithm for inline network traffic classification. The algorithm uses a self-training semi-supervised method and the concept of incremental k-means to continuously update the classification

Table 1
Network analyzing approaches and traffic dynamic behavior issues.

Name	*incremental*	*outlier*	*inline*	*comprehensive*
Loo and Marsono [22]	√	√	-	-
HB^{2}DS [21]	√	-	-	-
Mahdavi et al.[16]	√	-	-	-
Noferesti and Jalili [25]	√	√	-	-
Nazari et al. [24]	√	-	√	-
ACoPE	√	√	√	√

model after receiving new flow instances. The proposed algorithm uses the sufficient statistic known as clustering feature as proposed by [23] to compress the detected clusters. For each receiving flow, The nearest cluster and second nearest cluster are then determined. If both nearest clusters belong to the same class, a confidence level measure will increase. Only flow instances with a high confidence level are used in incremental learning and merged with the nearest cluster. The approach achieves high performance and accuracy in analyzing high-bandwidth networks; however, it does not address adaptive learning and comprehensiveness issues required for complex-policy enforcement in high-bandwidth networks.

Nazari et al. [24] apply different stream-classification algorithms such as Random Hoeffding Tree, Hoeffding Adaptive Tree, Hoeffding Tree, and kNN With PAW to identify applications over encrypted traffics. They utilize applications detected by the deep packet inspection technique, as ground truth data and update the classification model accordingly. Although adaptive learning is studied in the proposed approach, it does not have explicit *outlier handling* mechanism. Besides, complex policies cannot be defined in the proposed approach comprehensively.

The main difficulty in handling outliers in high-bandwidth networks is the absence of any information about the incoming flows in the future. Maintaining extended-meta-events as abstracting data structures over a sliding window, Nofereti and Jalili in [25] proposed an inline high-bandwidth network stream clustering algorithm designed to incrementally mine large amounts of continuously transmitting network traffic when some outliers can be dropped before determining the network traffic behavior. The extended-meta-events have a notion of a buffer, intended to handle outliers in the algorithm. The outliers would be caught in the buffer or left outside of the already existing extended-meta-events and put in a global buffer called window.

While state of the art in network traffic analysis tries to improve accuracy and/or performance, high-bandwidth network dynamic behaviors are not considered. While DPI modules attempt to increase the traffic identification accuracy, the *adaptivity* and lack of *comprehensiveness* are main obstacles of their applicability in high-bandwidth networks. This paper proposes ACoPE as an adaptive learning approach which monitors its error rate regarding the DPI module outputs. Adaptive learning regarding the deep packet inspection labels and a mechanism for comprehensive complex-policy enforcement in high-bandwidth networks are the main enrichments provided by ACoPE.

Table 1 summarizes how the current network analysis approaches address the high-bandwidth network traffic dynamic behavior issues, including *incremental learning* in which input data is continuously used to update to changes, *outlier handling, inline learning*, and *comprehensive processing*. The main difference between *inline learning* and *incremental learning* is that in *inline learning* the model is updated after each new data. However, *incremental learning* approaches is updated in a batch mode or an inline mode.

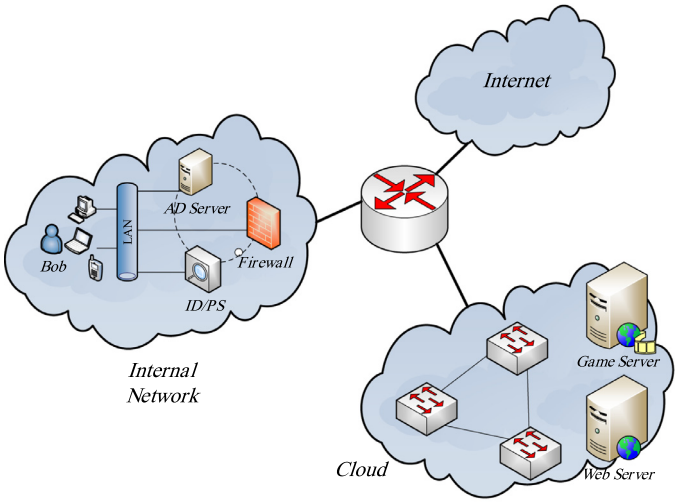

Fig. 1. A common network topology where *AD Server* is an active directory server, and *ID/PS* is an intrusion detection or prevention system.

3. Complex-Policy

High-bandwidth network traffic is composed of a massive and unbounded sequence of flows, continuously generated at rapid rates, called a flow stream [26–29]. Successive packets having the same 5-tuple $< source-IP, source-port, destination-IP, destination-port, TCP/UDP\ protocol >$ construct a flow. In network traffic analyzing systems, flow is enriched with some data to construct a data structure which is defined by Definition 1.

Definition 1 (Flow)**.** A flow data structure f follows the schema $f =< source-IP, source-port, destination-IP, destination-port, TCP/UDP\ protocol, ea_1, \ldots, ea_d >$ where the first five attributes are the traffic flow specifications and $ea_1, \ldots, ea_d$ are arbitrary extended attributes used to describe the flow more accurately.

An example of a typical network topology is depicted in Fig. 1. The dotted-circle shows the network administrator tools which manage *Internal Network* resources. *AD server* authenticates the network users. *ID/PS* detects and prevents network attacks. *Firewall* controls accesses between *Internal Network, Cloud* where enterprise customized services are deployed, and *Internet*. In the typical topology of Fig. 1, all traffics to *Cloud* is modeled by $<$ALL,ALL,ALL,*IP(Cloud)*,ALL$>$ where *IP*(*Cloud*) gets IP address of the *Cloud* gateway, or a traffic flow from *Bob* to the *Game Server* on source port 23 and destination port 5000 with 2*Mb* upload traffic and 10*Mb* download traffic is modeled by $<$*IP(Bob)*,23,*IP(Game Server)*,5000,TCP,2Mb,10Mb$>$ where *IP*(.) determines IP address (The arbitrary attributes are usually used to maintain flow features for machine-leaning based applications).

Definition 2 (Network traffic)**.** Network traffic is a potentially infinite ordered sequence of flows which is denoted by *FlowSet*.

Generally, a network security policy is defined in the form of "IF *condition* THEN *action*". Suppose that *ActionSet* is the set of all possible actions that can be enforced on the traffic. According to the *condition* part, we categorize the network policies into simple, semi-complex, and complex ones.

Definition 3 (Simple policy)**.** A simple security policy is a member of 2-ary relation over the sets *FlowSet* and *ActionSet* which is denoted by *SimplePolicy* where

$$SimplePolicy \in FlowSet * ActionSet \quad (2)$$

The condition of a simple security policy is defined based on the flow attributes such as *source – IP* or *destination – port*. Policies related to flow information, such as access to certain port numbers or servers, can be mapped into simple policies. Suppose that the network administrator in Fig. 1, requires to prohibit *Bob*'s access to web pages with financial contents on *Cloud*. Because this requirement depends on more information than flow attributes, it cannot be defined by the simple policies.

Lack of *comprehensiveness* of simple policies force the analyzing tools to prepare semi-complex policies which can satisfy more administrator requirements. Semi-complex policies are based on a binary relation denoted by *SPR*. Considering *LabelSet* as a set of labels meaningful to administrators, such as traffic applications, *SPR* is a function from *FlowSet* to *LabelSet*. More formally,

$$SPR : FlowSet \rightarrow LabelSet \tag{3}$$

For every $fs \in FlowSet$ there is exactly one element $label \in LabelSet$ such that the ordered pair $< fs, label >$ is contained in the subset defining *SPR*.

Definition 4 (Semi-complex policy)**.** A semi-complex policy is a relation over the sets *SPR* and *ActionSet* denoted by $SemiComplexPolicy \in SPR^{*}ActionSet$ where *SPR* is defined by Eq. (3).

Traffic packet inspection methods, known as DPI techniques, try to define relation *SPR* by looking for application-specific patterns within packets' payload (it is costly to be *adaptive*). ID/PS systems use a set of predefined application signatures to assign each flow to a label in *LabelSet*. While encrypted traffic cannot bet inspected where machine learning approaches try to help, DPI-based approaches produce a heavy operational load, especially in high-bandwidth networks. Considering Fig. 1, the network administrator can prohibit *Bob*'s access to the web pages with financial contents with a semi-complex policy. For this, it should modify relation *SPR* and assign each flows with 'money' keyword to the financial web content.

Although *SPR* can define more administrators requirements, it has two main drawbacks.

1. *SPR* relation between *FlowSet* and *LabelSet* is defined as a function where there is exactly one label in *LabelSet* for every flow. As a result, semi-complex policies do not consider relations between flows such as the sequence of flows or set of flows. Suppose that a network administrator requires to prevent the network from distributed denial of service attacks. The requirement cannot be defined by *SPR*.
2. In high-bandwidth networks, where the traffic generating processes are not strictly stationary, the concept we are predicting such as the relation between flows and labels may change over time. Consequently, the relation should be learned incrementally and adapt to the evolution of traffic generating process over time.

These drawbacks motivate us to define complex policies. The complex-policy is based on a relation *CPR* which is a subset of the cartesian product $P(FlowSet)^{*}LabelSet$ where $P(FlowSet)$ is the power set of *FlowSet*.

Definition 5 (Complex-policy)**.** A complex-policy is a member of a relation over *CPR* and *ActionSet* which is denoted by $ComplexPolicy \in CPR^{*}ActionSet$ where

$$CPR \subset P(FlowSet) * LabelSet \tag{4}$$

Table 2 summarizes the notations and their meaning used to define complex policies.

Complex policies satisfy a wide range of administrator requirements, while all relations between flows can be modeled by a member of $P(FlowSet)$. For example, a distributed denial of service attack can be defined as a large set of flows having the same destination address. Furthermore, it can be incrementally updated which maintains the status of the high-bandwidth network traffics.

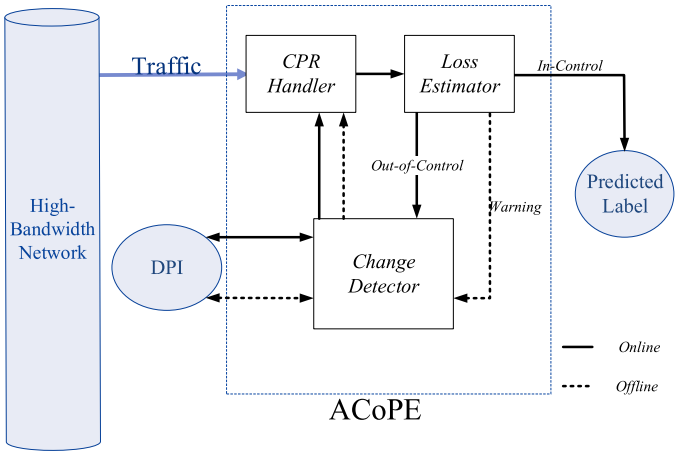

Fig. 2. The architecture of ACoPE.

4. ACoPe: adaptive learning approach for complex-policy enforcement

This section proposes the details of an Adaptive semi-supervised learning approach for Complex-Policy Enforcement in High-Bandwidth networks which is called ACoPE. ACoPE maintains complex-policy relation *CPR* to analyze the infinite stream of traffics in real-time with a limited amount of computational and storage overhead.

Suppose that for a set of flow *FlowSet* and set of label *LabelSet*, at time *t*, the complex-policy relation CPR_t is a set of $\{< \{f_1^1, \ldots, f_i^1\}, l_1 >, \ldots, < \{f_1^m, \ldots, f_j^m\}, l_m >\}$ where *m* is the size of CPR_t. For each $< \{f_1^z, \ldots, f_n^z\}, l_z > \in CPR_t$, *n* is the size of flow set $\{f_1^z, \ldots, f_n^z\}$, for all $1 < b < i$, $f_b^z \in FlowSet$, and $l_z \in LabelSet$. To summarize the set of flow $\{f_1^z, \ldots, f_n^z\}$, ACoPE defines the meta-flow data structure in Definition 6.

Definition 6 (Meta-flow)**.** A meta-flow is a summarization of a set of flows $\{f_1, ..., f_n\}$; which is defined as a tuple $(c, r, b, ts, ld, \sigma_i, p_i)$ where *c* (the meta-flow center) corresponds to a tuple with flow's entries, *r* (radius) is the maximum distance of flows from *c*, *b* is a buffer containing at most *m* flows of the stream, and *ts* is the time-stamp of meta-flow last update. *ld* named label-set is an array of the form $\{< label_1, count_1 >, \ldots, < label_w, count_w >\}$ where $label_i \in LabelSet$ and $count_i$ counts the number of flow with label $label_i$ in the flow set. For each dimension, the average of attributes is maintained in *c*. The *k*th entry of *c* is equal to $\frac{\sum_{i=1}^{n}(f_i^k)}{n}$ where (f_i^k) is the value of f_i in the *k*th dimension. σ_i and p_i are two statistics used to monitor the state of the meta-flow.

Considering the meta-flow data structure, the relation $CPR_t = \{< \{f_1^1, \ldots, f_i^1\}, l_1 >, \ldots, < \{f_1^m, \ldots, f_j^m\}, l_m >\}$ is converted into $\{< MF_1, l_1 >, \ldots, < MF_m, l_m >\}$ where MF_i is a meta-flow which is summarized the set of flows $\{f_1^i, \ldots, f_z^i\}$ with size *z*. At time *t*, for a flow *f* a prediction label l' is made using the current relation CPR_t. After some time, ACoPE receives the true label *l* and can estimate the loss by a function $loss(l', l)$ and it can use the true pair $< f, l >$ for model update to obtain CPR_{t+1}.

The ACoPE architecture is depicted in Fig. 2, the online and offline operations are depicted by straight and dotted lines respectively. ACoPE has three main modules, namely *CPR Handler, Loss Estimator*, and *Change Detector*. When a new flow f_{new} arrives at time

Table 2
Notations and their meaning.

Notation	Meaning
LabelSet	A set of possible labels meaningful for network administrator
FlowSet	A set of all flows in the network traffic
ActionSet	A set of all possible actions that can be enforced on the traffic
SPR	A function from *FlowSet* to *LabelSet*
SimplePolicy	A member of a 2-ary relation over the sets *FlowSet* and *ActionSet*
SemiComplexPolicy	A relation over the sets *SPR* and *ActionSet*
CPR	A subset of the cartesian product $P(FlowSet)^* LabelSet$.
ComplexPolicy	A member of a relation over *CPR* and *ActionSet*
MF	A meta-flow

t, *CPR Handler* finds the closest meta-flow to f_{new} among meta-flows in CPR_t (*first step*). *Loss Estimator* module estimates the meta-flow error and assigns a label to the flow (*second step*). And finally, updating the meta-flow according to f_{new} is performed in *Change Detector* module which utilizes the DPI module to find the flow true label (*third step*). The details of ACoPE steps are explained in the following sub-sections.

4.1. The first step: Finding the closest meta-flow

In the first step, ACoPE finds the closest meta-flow to the new arrived flow f_{new}, denoted by MF_c. If the distance between f_{new} and MF_c is less than the MF_c's radius, f_{new} is inserted into MF_c. Otherwise, f_{new} is inserted into window W. The process window algorithm proposed by [25] is used to create new meta-flows over the sliding window W. Initially, the meta-flow's state is *Warning* and its label is specified by the DPI module.

Finding the closest meta-flow relies on the ability to measure similarity or distance between flows and meta-flows' center. The distance between a flow and a meta-flow is defined by Definition 7. The distance function is defined based on the fractional l_p-metric which is proposed by Weller-Fahy et al. in [30] to improve the network classification results.

Definition 7 (Distance Function)**.** The distance between flow f and meta-flow MF with center c is calculated as:

$$distance(f, MF) = \left(\sum_{k=1}^{d+5} w_k * \left(\frac{|f^k - c^k|}{|domain(k)|} \right)^{0.5} \right)^2 \quad (5)$$

d is the number of extended flow attributes, $domain(k)$ is the domain of the kth attribute used to normalize the attribute values; w_k is the weight of the kth attribute, and $\sum_{k=1}^{d+5} w_k = 1$. In this paper, the equal weights are considered for all attributes.

4.2. The second step: Estimating the meta-flow error

Considering $< MF_i, l_i >$ as a pair of meta-flow and label, f_{new} as the arriving flow, and l_{new} as its true label, ACoPE assigns l_i to f_{new}, that can be either *true* ($l_i = l_{new}$) or *false* ($l_i \neq l_{new}$). The assignment error is a random variable from Bernoulli trials because : (1) each assignment results in one of two possible outcomes, *true* or *false* (2) if drift does not happen, the probability of *false* remains constant from assignment-to-assignment, (3) the assignments are independent [31]. For the random variable, the Binomial distribution gives a general form of probability that represents the number of the assignment errors in a set of n assignments. At time t, for the pair $< MF_i, l_i >$, the error-rate is the probability p_t^i of observing *false* ($l_i \neq l_{new}$) with the standard deviation $\sigma_t^i = \sqrt{p_t^i(1-p_t^i)/t}$ [31].

For a sufficiently large number of observations (> 30), the Binomial distribution is closely approximated by the Normal distribution with the same mean and variance. Considering that the probability distribution should not change unless a drift happens, the $(1-\alpha)/2$ confidence level for p_t^i is approximately $p_t^i \pm z^* * \sigma_t^i$. z^* value is calculated according to α as confidence level and z-table [32]. For instance, to construct a 95% confidence level, we have $(1-0.95)/2 = 0.025$. Regarding the z-table, z^* value is $1.96 \approx 2$. Now the upper rage and the lower range of p_t^i is approximately $p_t^i \pm 2 * \sigma_t^i$ wit confidence level 95%. According to z-table [32], z^* value is $2.576 \approx 3$ for confidence level 99%.

For each meta-flow MF_i, ACoPE maintains two registers during the operation, $p_{\min}^i$ and $\sigma_{\min}^i$. At time t, after predicting the current flow label and verifying the assignment error for the meta-flow, if $p_t^i + \sigma_t^i$ (determined from Normal distribution) is lower than $p_{\min}^i + \sigma_{\min}^i$, then $p_{\min}^i = p_t^i$ and $\sigma_{\min}^i = \sigma_t^i$.

ACoPE uses confidence level 95% and 99% for *Warning* and *Out-of-Control* states. At time t, a meta-flow MF_i can have one of the following states:

1. The meta-flow is in the *In-Control* state where all flows in MF_i is deemed to come from the same distribution if
$$p_t^i + \sigma_t^i < p_{min}^i + 2 * \sigma_{min}^i \quad (6)$$
In this state, the flow f_{new} is released with the meta-flow label l_i. With probability p_{dpi}, the flow is sent into the DPI module to determine its label and update the meta-flow.
2. The state is *Out-of-Control* whenever
$$p_t^i + \sigma_t^i \geq p_{min}^i + 3 * \sigma_{min}^i \quad (7)$$
With the probability of $(1-\alpha)/2$ the recent examples come from a different distribution than the previous examples. Flow f_{new} is sent into DPI and released with the DPI label. Moreover, the meta-flow MF_i is removed from the complex-policy relation.
3. Between the two previous states, the meta-flow is in the *Warning* state. In this state, the flow is released with DPI module output, but the meta-flow is updated against the output. Note that, while the confidence level is valid for the Normal distribution, if the meta-flow size is less than 30, then ACoPE considers the meta-flow state as *Warning*.

4.3. The third step: Updating the meta-flow

To summarize the continuously arriving flows and, handle outliers properly, ACoPE uses the robust buffering technique proposed in [25]. Considering the meta-flow's buffer $MF_c.b$ as a set of flow where are flow are recently inserted into the meta-flow MF_c. When the buffer size becomes greater than a predefined threshold, named max_buffer_size, a meta-flow MF_b is defined over buffer $MF_c.b$. Then, meta-flow MF_b is merged into MF_c. Consider $distance(.)$ as a distance function and $\zeta = MF_c.r + MF_b.r - distance(MF_c.c, MF_b.c)$, a flow f can be defined as $f = MF_c.c + \frac{MF_c.r}{distance(MF_c.c, MF_b.c)} * (MF_c.c - MF_b.c)$. In [21], it is shown that the merging operation is equal to inserting f into MF_c. Suppose that $\sigma = 1/2 * (distance(f_{new}, MF_c.c) - MF_c.r)$, after inserting f into MF_c its radius and center calculated as $MF_c.r = MF_c.r + \sigma$ and $MF_c.c = f + \frac{MF_c.r}{MF_c.r + \sigma} * distance(f, MF_c.c)$.

ACoPE uses the partial memory approach mechanism [31] to remove outdated meta-flows. The outdated meta-flows are detected regarding the time duration that flows are not inserted into them. To forget an outdated meta-flow MF_{out} from CPR_t, the current timestamp is subtracted from $MF_{out}.ts$. If the subtracted value is more than a predefined threshold, $Outlier_{THR}$, the meta-flow MF_{out} is removed from CPR_t.

To incrementally compute the number of each label in a meta-flow, ACoPE uses the hash-map data structure. Consider MF as a meta-flow and the assigned true label l, ACoPE increases the number of label l in $MF.ld$ by one (initiated by one). At time t, the label with maximum count in $MF.ld$ is the meta-flow's label.

4.4. ACoPE procedure

The details of ACoPE algorithm is described in Algorithm 1. ACoPE reads traffic flows continuously from the flow stream FS. All the meta-flows detected by the algorithm is stored in a set called MFS_t. W is the window data structure. The value of $Outlier_{THR}$ and represent the outlier detection threshold. For each arriving flow f_{new}, ACoPE finds the closest meta-flow named MF_c. If the meta-flow covers f_{new} (the flow distance to the meta-flow center is less than the meta-flow radius), it is inserted into the meta-flow buffer. When the buffer becomes full, the algorithm creates a meta-flow over traffic flows in the buffer and merge it with MF_c. Regarding the state of MF_c, ACoPE decides to release the f_{new} with a DPI detected label or the MF_c's label.

Algorithm 1 ACoPE (FS, MFS_t, W, $Outlier_{THR}$).

for each arriving flow f_{new} in FS **do**
  find the closest meta-flow MF_c in CPR to f_{new};
  if $distance(f_{new}, MF_c) \leq MF_c.r$ **then**
    insert f_{new} into the buffer $MF_c.b$;
    update $EM.t$;
    if MF_c is in *In-Control* state **then**
      release f_{new} with the label with the highest count in $MF_c.ld$;
      with probability p_{dpi} process f_{new} with DPI module and update $MF_c.ld$;
    end if
    if MF_c is in *Out-of-Control* state **then**
      release f_{new} with the DPI detected label
      remove MF_c from the relation CPR;
    end if
    if MF_c is in *Warning* state **then**
      release f_{new} with the DPI detected label
      update $MF_c.ld$ regarding f_{new} and its detected label;
    end if
    if the size of $MF_c.b$ is more than max_buffer_size **then**
      create meta-flow MF_b over the flows in the buffer;
      merge meta-flows MF_b and MF_c;
    end if
  else
    insert f_{new} into the window W and label it by DPI module;
    if W is full **then**
      call *ProcessWindow* algorithm proposed by [25];
      free W;
    end if
  end if
  remove the outdated meta-flows every $Outlier_{THR}$ duration;
end for

Table 3
Features extracted from network traffic.

Description	Count
Number and size of uploaded traffic	2
Number and size of downloaded traffic	2
Number of downloaded and uploaded traffic carrying payload	2
First byte of payload of upstream and downstream traffic	2
Min, max, mean, and var of uploaded traffic size	4
Min, max, mean, and var of downloaded traffic size	4
Min, max, mean, and var of uploaded inter-packet time	4
Min, max, mean, and var of downloaded inter-packet time	4
Min, max, mean, and var of RTT	4

5. Evaluation

Three different experiments are executed to evaluate the proposed approach. At first, the performance of ACoPE in classifying network traffics is compared with 10 different stream and traditional classification algorithms performances reported in [24]. Afterward, the effectiveness of ACoPE to address the main constraints exist in analyzing of high-bandwidth networks to enforce security policies, namely *comprehensive processing* and *adaptive learning*, is evaluated through simulation. In the simulation, the total system satisfaction of administrators requirements in a high-bandwidth network with three different scenarios, named *Heavy-loaded application, Modified-Applications*, and *Distributed-Denial-of-Service* is examined. And finally, the performance and accuracy of ACoPE are evaluated through a pilot study in a real high-bandwidth network.

ACoPE is implemented in C++ (GCC version 5.2.1) on Intel Xeon E5620 CPU with Linux kernel version 3.17.0.3, and 42GB of main memory. All devices are connected directly with 10Gbps fiber links. To read packets directly from the interface drivers, the DPDK library [33](version 18.2) is used which receives and sends packets within the minimum number of CPU cycles. The nDPI library version 1.8 [34] is used as DPI module which is an open source deep packet inspection tool which specifies the session protocol. We select nDPI as DPI module while it is ranked as the first open-source DPI in [35,36] which studied the performance of different DPI tools in various scenarios. However, ACoPE is independent of the DPI tool and one can choose any other tools.

5.1. Feature list

In this paper, we use the feature list proposed by Wang et al. in [37]. Accordingly, 28 features are extracted for each flow which are shown in Table 3. In the feature list, the round trip time (RTT) of a TCP flow is calculated as the total time between a sender transmitting a packet and the reception of its corresponding ack packet.

5.2. Comparing different classification algorithms

To evaluate ACoPE in terms of classifying network traffics, the UNB ISCX VPN-nonVPN dataset [38] and UNB ISCX Tor-nonTor dataset [39] are used. The datasets are constructed from different categories and applications such as file transfer (Skype, SFTP, FTPS), chat (Facebook, Hangout, Skype, AIM, ICQ), email (SMTP, POPS, IMAPS), and P2P (Bittorrent, Vuze).

ACoPE is compared with 10 different batch and stream classification algorithms, named Naive Bayes (NB), Random Hoeffding Tree (RHT), Hoeffding Adaptive Tree (HAT), Hoeffding Tree (HT), kNN With PAW (KWP), kNN, Oza Bag ADWIN ML (OBAML), HeterogeneousEnsembleBlast (HEB), Oza Boost ADWIN (OBA), and AdaptiveRandomForest (ARF). Nazari *et el.* applied the classification algorithms on the datasets in [24] and reported the results. The re-

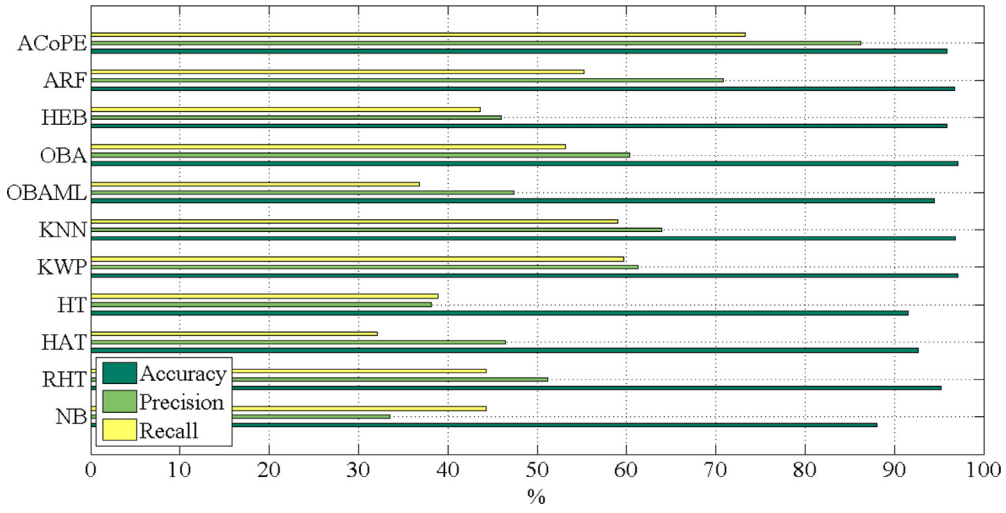

Fig. 3. Classification evaluation parameters on VPN dataset.

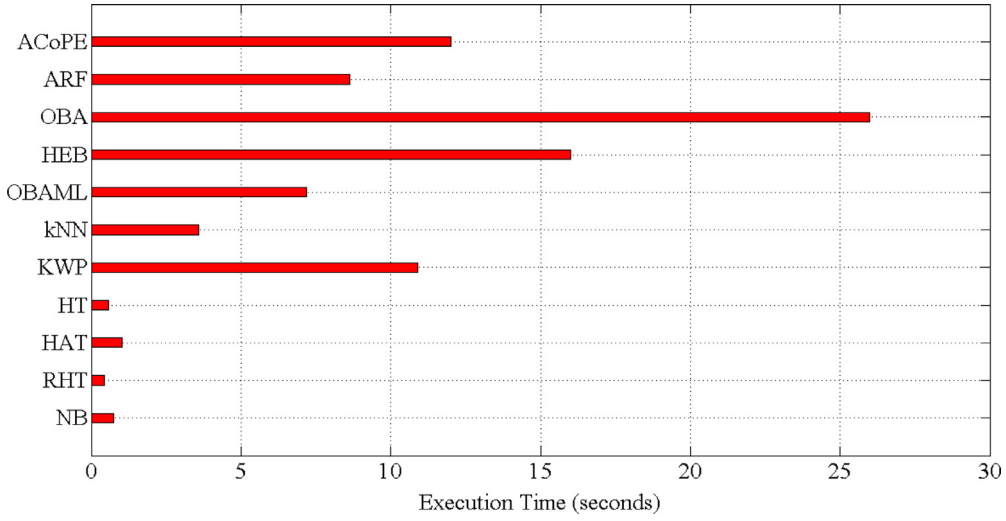

Fig. 4. Execution time of classification algorithms on the VPN dataset.

ported results are used to evaluate the performance of ACoPE in classifying network traffics.

Three evaluation measures are used to evaluate the quality of classifiers: (1) *Accuracy*, (2) *Precision*, and (3) *Recall*. Considering *TP, TN, FP* and *FN* as true positive, true negative, false positive and false negative, *accuracy=(TP+TN)/(TP+FP+FN+TN)* is used to evaluate the overall performance of a classifier. *Precision=(TP)/(TP+FP)* and *recall=(TP)/(TP+FN)* are used to evaluate the performance of each classes.

Fig. 3 shows the evaluation measures for different classification approaches on VPN dataset. Except Naive Bayes, the accuracy of stream classification approaches on VPN dataset is above 90%. OBA and KWP with 97% accuracy can detect the applications with less error rate. The precisions of ACoPE (86.21%) and ARF approach (70%) are more than the precision of other approaches. The recall of all approaches except ACoPE is less than 60%, while HAT with 32% has the worst recall measure.

The execution time of different classification algorithms on VPN dataset is demonstrated in Fig. 4. As the figure shows, HeterogeneousEnsembleBlast and Oza Boost ADWIN have the most execution time. Hoeffding Tree and Random Hoeffding Tree processed the datasets as fast as the Naïve Bayes approach did. ACoPE processed the dataset in 12 s which is promising.

The evaluation results and execution time of different approaches on Tor dataset are represented by Figs. 5 and 6. Most algorithms accuracy are between 80% and 90%. Oza Boost ADWIN with 87% accuracy and 73% recall, and Oza Bag ADWIN ML with 80% precision and 73% recall reported the most evaluation measures on Tor dataset. The execution times are depicted on a logarithmic scale. The slowest approaches are Oza Boost ADWIN with 1920s execution time and AdaptiveRandomForest with 1135s execution time. ACoPE processed the Tor dataset in 21 s with 81% accuracy, 73% precision, and 61% recall.

Table 4 compares the results of different network classification approaches on VPN and Tor datasets. The values of accuracy, precision, recall, and the number of detected classes are inserted as the approaches are claimed. All the approach except Hoeffding Adaptive Tree [24], kNN With PAW [24], and ACoPE process the datasets in batch mode, where all the traffic is available before starting the classification algorithm. In batch mode, C4.5 with 99.2% accuracy on VPN dataset and 97% accuracy on Tor dataset outperforms the other algorithms.

The main two contributions of ACoPE *comprehensive processing* and *adaptive leaning* which are essential for security policy enforcement on high-bandwidth networks. As the evaluation results and Table 4 show, the outputs of ACoPE is similar to other stream

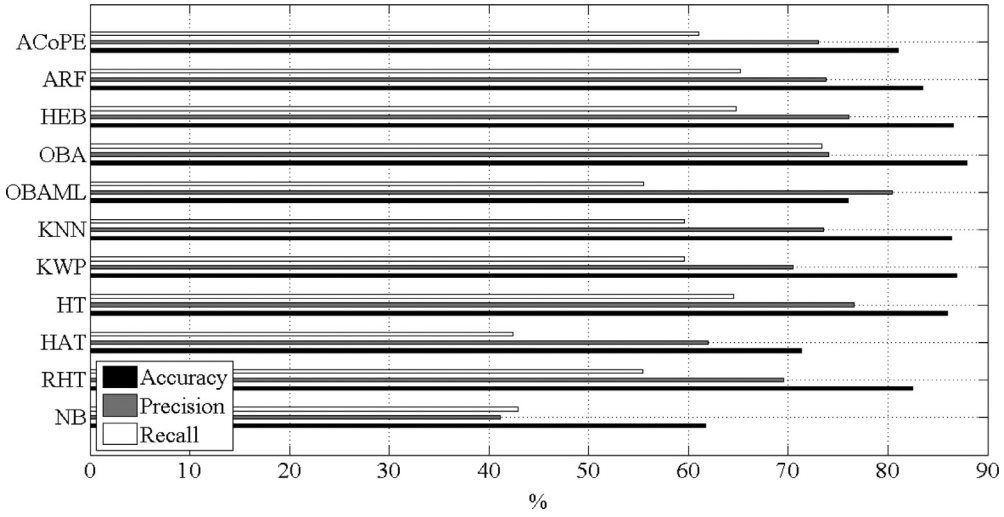

Fig. 5. Classification evaluation parameters on Tor dataset.

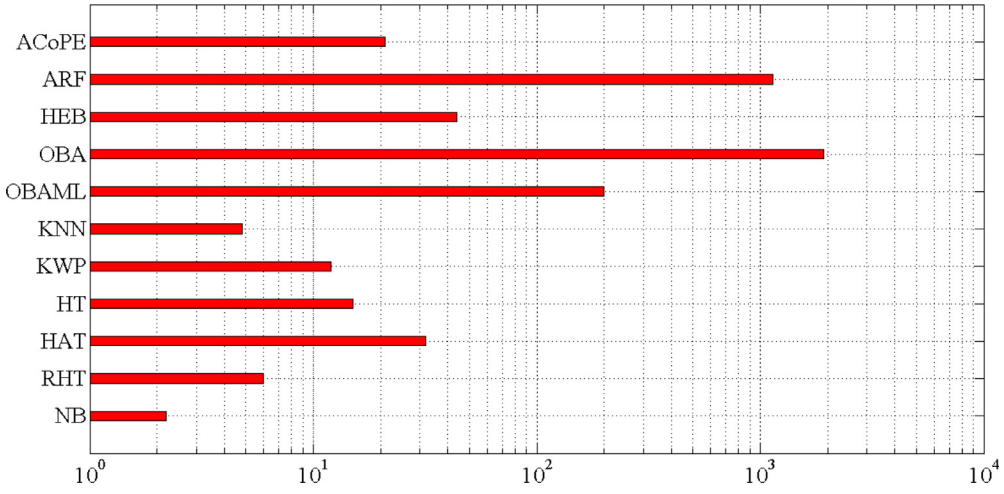

Fig. 6. Logarithmic execution time of classification algorithms on the Tor dataset.

Table 4
Network classification approaches outputs reported on the VPN and Tor datasets.

Dataset	Classification Approach	Accuracy	Precision	Recall	Detected Classes
VPN	Draper-Gil et al. [40]	80%	-	-	14
	Yamansavascilar et al. [40]	93.94%	-	-	14
	Wang et al. [41]	-	95.2%	92.0%	17
	Lotfollahi et al. [42]	95%	95%	95%	17
	C4.5 classifier [24]	99.2%	99.2%	99.2%	16
	Hoeffding Adaptive Tree [24]	96.75%	70.81%	55.28%	16
	ACoPE	95.92%	86.21%	73.29%	16
Tor	Lashkari et al. [43]	-	83%	84%	14
	Hodo et al. [44]	88.0%	79.4%	89.4%	8
	C4.5 classifier [24]	97.1%	97.3%	97.2%	17
	kNN With PAW [24]	86.92%	70.48%	59.61%	17
	ACoPE	81.12%	73.59%	61.08%	17

classifiers. To validate the contributions, we simulate ACoPE and evaluate how it addresses the *comprehensive processing* and *adaptive leaning* challenges in high-bandwidth network analyzing.

5.3. ACoPE simulation

In order to validate ACoPE, we examine how ACoPE addresses different administrators' requirements. Three most common network management requirements, named *Heavy-loaded application, Modified-Applications*, and *Distributed-Denial-of-Service* are defined. The CPU and memory usage, the number of dropped sessions, and the number of sessions processed by DPI are measured during the experiment.

The network test-bed topology used to implement the policies is depicted in Fig. 7. The topology is carried out using the Mininet network emulator [45]. Four hosts, H_1 to H_4, are linked to servers, S_1 and S_2, via three different switches. In the test-bed, *Client* in H_1 requests web pages from a web server which is implemented in S_1. To generate a realistic network workload, the distributed Internet traffic generator tool called D-ITG is used [46]. Table 5 represents

Table 5
The simulation workload details.

Source	Destination(s)	Flow(s) Specifications
H_1	S_1 and S_2	DNS and Telnet flows towards the servers.
H_1	S_2	A UDP flow in passive mode with constant inter-departure time between packets and constant packets size.
H_2	S_2	A TCP flow with constant inter-departure time between packets and uniformly distributed packet size betwee 500 and 700 bytes.
H_2	S_1 and S_2	A Single DCCP flow with constant inter-departure time between packets, and constant packet size.
H_3	S_1 and S_2	Three UDP flows with different constant bit rates.
H_3	S_1 and S_2	VoIP, Telnet, DNS flows towards the servers.
H_4	S_2	A Single SCTP flow, with association Id 3 and max outband stream.
H_4	S_2	A *Counter-Strike* (a multiplayer game) traffic flow towards the server.
H_4	S_1	A Single UDP flow with bursty inter-departure time between packets where the period durations are random variables [48].

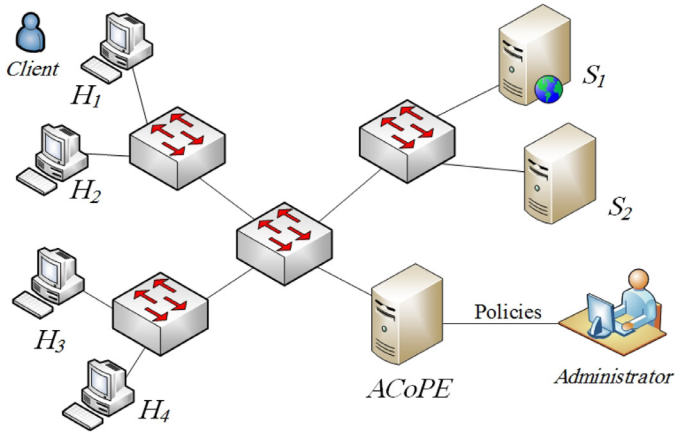

Fig. 7. The topology is used to simulate the complex-policy scenarios.

the simulation workload details (all the flows are generated by D-ITG tool version 2.8.1 regarding its manual [47]). The workload is passed through ACoPE which addresses the *Administrator* requirements by analyzing the traffic flows.

5.3.1. Heavy-loaded application scenario

One of network administrators requirement is to process the network traffic with deep packet inspection tools. The tools impose heavy overhead to process the high-bandwidth network traffics. ACoPE reduces the overhead by using complex-policy relation and keeps the accuracy high through using the statistical process control technique. To evaluate the performance of ACoPE in reducing the percentage of traffic processed by DPI, *Heavy-loaded application scenario* is designed.

In this scenario, *Administrator* wants to control the *Client* accesses to the web pages in a high-bandwidth network. To apply the security policy to the topology depicted in Fig. 7, a large web-page of size 10*MB* is created at server S_1. A signature for the DPI module is developed to parse the web content. Matching the signature for the large web-page is costly. *Client* sends a large number of requests to S_1. According to our server resources, the DPI module can process 8 flow concurrently.

To process the number of requests, ACoPE defines a meta-flow over the web-page related flows. When the meta-flow state goes to *In-Control*, ACoPE only sends a limited number of flows to the DPI module and a large number of flows are classified according to their similarity to meta-flow. In this setting, ACoPE is able to process over 50 simultaneous web-page requests. The CPU usage of the DPI module and ACoPE is depicted in Fig. 8. As the figure shows, DPI CPU usage increases to 100% after 15 web-page requests, but ACoPE can handle more than 7-times with the same resource usage. For the first flows, ACoPE sends them to the DPI module to maintain its precision which makes its resource usage high at the beginning.

5.3.2. Modified-application scenario

Applications and policy changes are inalienable in high-bandwidth network management. Security policy enforcement approaches have to control the changes with low overhead. To compare ACoPE with deep packet inspection modules to adapt to traffic changes, the *Modified-Application scenario* is designed. Suppose that in Fig. 7, *Administrator* denies access to web pages with financial contents from clients. Hence, deep packet inspection approaches

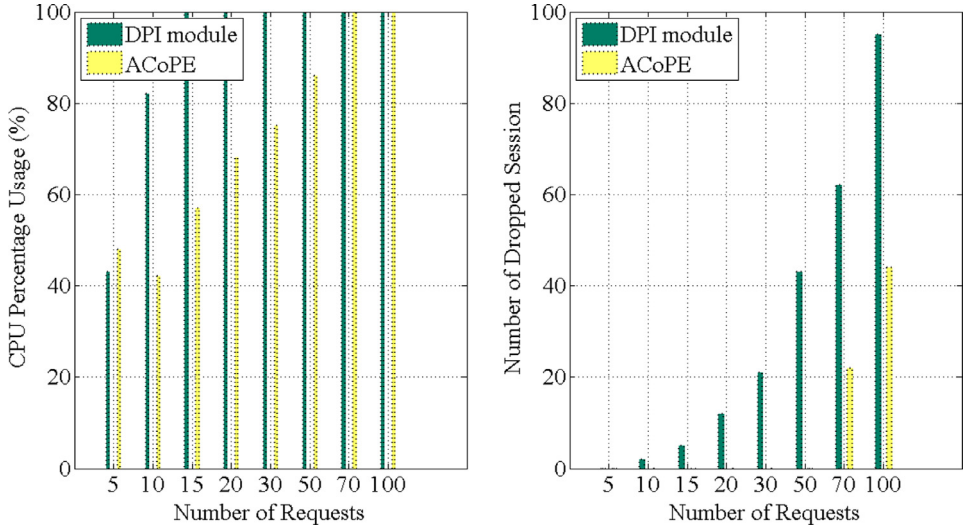

Fig. 8. The CPU usage and the number of dropped sessions during *Heavy-loaded application* scenario.

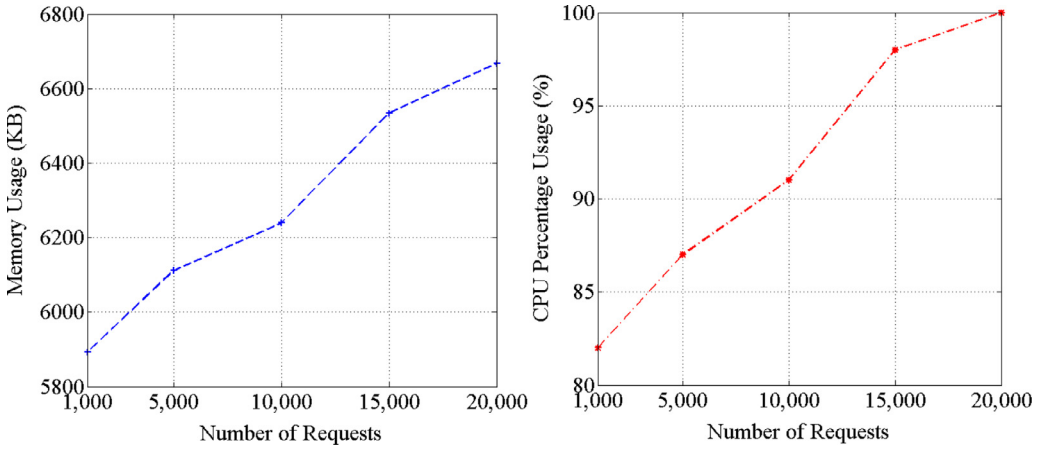

Fig. 9. The memory and CPU usage of ACoPE during *Distributed-Denial-of-Service* scenario.

Table 6
Modified-Application scenario results.

output	nDPI	ACoPE outputs (first experiment)	ACoPE outputs (second experiment)
Number of Sessions labeled as financial	1000	1874	1291
Precision	50%	77.6%	69.5%
Warning state	-	Once after 1259 sessions	1629 times
Number of sessions processed by nDPI	2000	749	1712

defines a signature to mark all the web traffic with 'money' keyword as financial traffic and all are dropped through the switch. In *Modified-Application* test, two web-pages are created in server S_1 where both have the same content but the 'money' keywords are replaced by 'modified' in the second one. As the common firewall and DPI modules are so fragile, *Client* can access the second web-page with financial content which violates the policy.

To detect financial web contents via ACoPE, a complex-policy is defined. When the DPI module marks a flow as financial, the associated meta-flow label is set to financial. When the second page is requested, it is marked by DPI as unknown, but because the page is similar to the first one, it is also detected by ACoPE as a financial web page. After a while, the meta-flow state changes to *Warning*,and then *Out-of-Control*. Although after the *Out-of-Control* state, all the requests are allowed, ACoPE persists more against DPI shortcomings and it can send an alarm to *Administrator* when it reaches *Out-of-Control* state and reports the shortcomings.

In this scenario, host H_1 sends 1000 web requests for page-one and 1000 requests for page-two. In the first experiment, requests of page-two are sent after requests of page-one. In the second, the request are sent randomly. The test is run for ten times. Table 6 summarizes the results. During the first experiment, 1552 requests are detected as financial requests (precision is 77.6%); the state goes to *Warning* after 1259 requests, and 749 flows are processed by the DPI module. In the second experiment, 1391 flows are detected as financial (precision is 69.5%), the state goes to *Warning* for 1629 times, and 1712 flows are processed by the DPI module.

5.3.3. Distributed-denial-of-service scenario

DDoS attack mitigation is one of the common requirements in high-bandwidth networks where multiple compromised computer systems attack a target and cause a denial of service for users of the targeted resource. Despite most commercial devices try to detect the attack, ACoPE answers the requirement by defining a complex-policy. Consequently, the size of the flow set in a policy of ACoPE should not pass over a specific threshold. When an arrived flow is inserted into a meta-flow, meta-flow's size is checked according to the threshold.

To set an upper bound for the server concurrent connections, we defined a complex-policy where the size of each meta-flow cannot pass over 10000. Host H_1 sends different numbers of *SYN* packets with random source addresses to a specific destination address through Hping tool which is a TCP/IP packet assembler [49]. At first, the number of *SYN* packets was set to 5000 as normal behavior of the network. Then, another instance of hping was executed to send another 5000 *SYN* packets. We increased the number of flows to 20,000 with four instances of Hping process. We executed the test for five rounds and monitor the ACoPE state and its resource usage.

In all experiments, ACoPE detects the attack with only one meta-flow where its status is *In-Control* because the flows are similar (attacker sends a large number of similar flows in DDoS attack). Moreover, after the traffic rate passes over the threshold, the *SYN* packets are dropped. The CPU and RAM usage of ACoPE during the test are calculated and depicted in Fig. 9. The memory and CPU usage are almost linear which is increased by the number of flows.

5.4. ACoPE pilot study

We examined ACoPE in a pilot study to evaluate its capability for analyzing a real high-bandwidth network. The network traffic of an enterprise institute for one week which resulted in 2TB traffic data is fed into ACoPE. We compared the applications detected by ACoPE with the applications detected by nDPI library version 1.8 [34]. About $2.4^{*}10^{6}$ flows are labeled toward 68 different applications by the DPI module. After every 50,000 flows, the evaluation measures are calculated. More frequent applications along with the number of meta-flows labeled with the same application and the purity are represented in Table 7.

The number of miss-classified and unknown flows is depicted in Fig. 10. Unknown flows in ACoPE are categorized in some meta-flows, which DPI module is unable to recognize. Amongst meta-flows labeled as unknown, we analyzed the three largest meta-flows. Flows in the largest meta-flow are mostly (93.9%) related to Background Intelligent Transfer Service (BITS) which facilitates asynchronous, prioritized, and throttled transfer of files between

Table 7
Applications detected by ACoPE.

Name	Count	Meta-flow	Purity	Name	Count	Meta-flow	Purity
DNS	102	757	74%	HTTP	86	1255	90%
NETBIOS	70	340	89%	NTP	63	655	94%
SAMBA	43	279	97%	SOCKSv5	40	364	95%
ICMP	28	113	99%	SSH	26	181	96%
IMAP	25	87	91%	RTMP	20	94	86%
POP3	19	77	99%	SMTP	15	63	98%

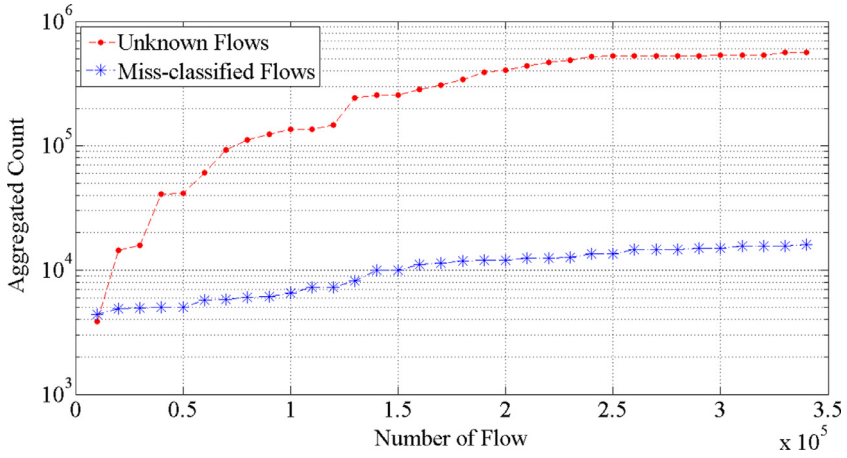

Fig. 10. The aggregated number of unknown and miss-classified flows as traffic flows proceed.

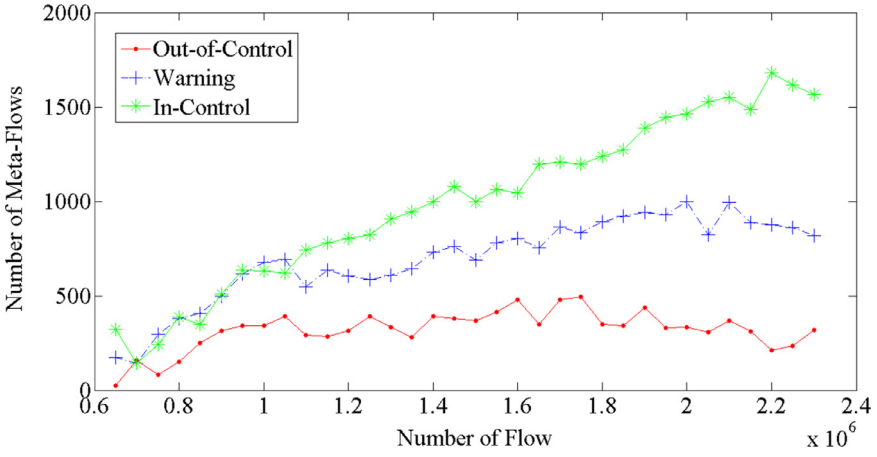

Fig. 11. The number of *In-Control*, *Warning*, and *Out-of-Control* meta-flows as traffic flows proceed.

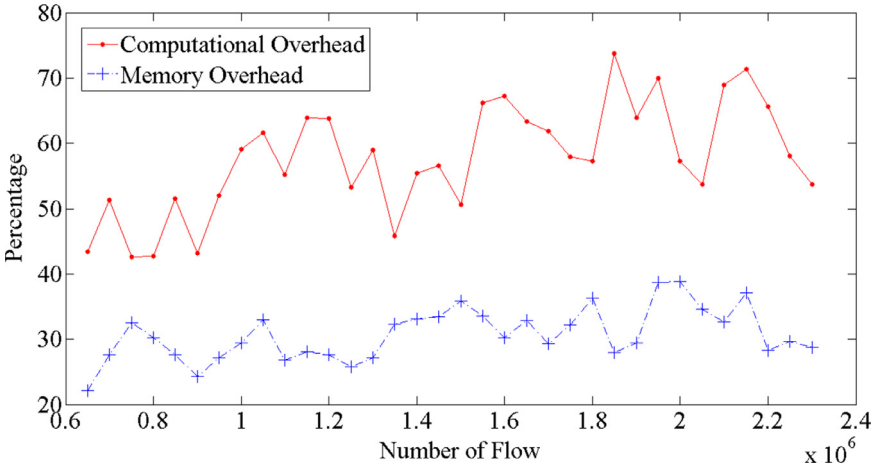

Fig. 12. The computational and memory overhead of ACoPE as traffic flows proceed.

machines using idle network bandwidth. Where nDPI does not have the BITS pattern, it cannot be recognized. The second meta-flows contain customized web-services developed by *Java* programming language. All the flows in the meta-flow are related to the customized service easily detected by specific port numbers. However, the third meta-flow cannot be easily analyzed where some flows are related to *Git, Dropbox, Spotify*, and Bluekai.

The number of *In-Control, Warning*, and *Out-of-Control* meta-flows are maintained by ACoPE as stream proceed is shown in Fig. 11. Usually the number of *In-Control* meta-flows is more than the number of *Warning* meta-flows which is also more than *Out-of-Control* ones. Regarding the pilot study results, about 40 percent of the flows are not processed by the DPI module, because they are covered by meta-flows in *In-Control* state which is decreased the computational overhead. When a flow is covered by a meta-flow in *Warning* state, it occurs about 32 percent in the pilot study, ACoPE sends it to the DPI module to keep the accuracy level high. If a meta-flow reaches *Out-of-Control* state it implies that the performance of the particular meta-flow has degraded significantly, therefore it is removed from the complex-policy relation.

Fig. 12 represents the computational and memory overhead of ACoPE on Intel Xeon E5620 CPU and 42GB of main memory. Accordingly, ACoPE processes the flow and maintain the complex-policy relation up-to-date, it runs under computational and memory constraints. The limitation of resources as memory and computational power is considered during the designing if ACoPE, it only stores an abstraction of the network traffic behavior by complex-policy relation that can be easily updated incrementally.

6. Conclusion and future works

Adaptive and robust analyzing of network traffics to enforce complex policies is critical for high-bandwidth network management. This study presented ACoPE as an adaptive semi-supervised learning approach for complex-policy enforcement in high-bandwidth networks. Administrators' requirements are comprehensively defined by the proposed complex-policy relation. ACoPE detects and maintains inter-flows relationships by an abstracting data structure named meta-flow. Utilizing deep packet inspection techniques, ACoPE adaptively assigns applications for each detected meta-flow. A statistical process control technique is used to monitor accuracy. Whenever the accuracy decreased; it is considered as a changed behavior, and the classification model is updated regarding the data from the deep packet inspection module. Experimental results have confirmed that developing ACoPE produces high effectiveness, accuracy, and efficiency to enforce complex policies in real high-bandwidth networks.

As further research, we plan to develop an expression language with which network administrators can describe their required policies as some relationships between traffic flows. ACoPE detects the flow relationships by a set membership where the sequential patterns of flows are dismissed. Introducing a data structure to abstract traffic flows including their orders could be considered as a feature work. The visualization of network traffic status according to meta-flows maintained by ACoPE is another future work which can make a better view for administrators about live traffic behaviors such as top fastest-growing applications or network resource usage. Besides, conflict resolution between different complex-policies is one of the challenges of policy-based management approaches in dynamic environments which should be studied in ACoPE.

Declaration of Competing Interest

None.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.comnet.2019.106943.

References

[1] Cisco VNI Forecast and Methodology, 2015–2020, 2017. available at http://www.cisco.com/.
[2] E. Viegas, A. Santin, N. Neves, A. Bessani, V. Abreu, A resilient stream learning intrusion detection mechanism for real-time analysis of network traffic, 2017, pp. 1–6.
[3] S. Alcock, J.-P. Möller, R. Nelson, Sneaking past the firewall: quantifying the unexpected traffic on major TCP and UDP ports, in: Proceedings of the 2016 ACM on Internet Measurement Conference, ACM, 2016, pp. 231–237.
[4] J. Zhang, X. Chen, Y. Xiang, W. Zhou, J. Wu, Robust network traffic classification, IEEE/ACM Trans. Netw. 23 (4) (2015) 1257–1270, doi:10.1109/TNET.2014.2320577.
[5] P. Perera, Y.-C. Tian, C. Fidge, W. Kelly, A comparison of supervised machine learning algorithms for classification of communications network traffic, in: International Conference on Neural Information Processing, Springer, 2017, pp. 445–454.
[6] A. Dainotti, A. Pescape, K.C. Claffy, Issues and future directions in traffic classification, IEEE Netw. 26 (1) (2012) 35–40, doi:10.1109/MNET.2012.6135854.
[7] H. Gharaee, H. Hosseinvand, A new feature selection IDS based on genetic algorithm and SVM, in: Telecommunications (IST), 2016 8th International Symposium on, IEEE, 2016, pp. 139–144.
[8] A. Tongaonkar, R. Torres, M. Iliofotou, R. Keralapura, A. Nucci, Towards self adaptive network traffic classification, Comput. Commun. 56 (2015) 35–46, doi:10.1016/j.comcom.2014.03.026.
[9] S. Sicari, A. Rizzardi, L.A. Grieco, A. Coen-Porisini, Gone: dealing with node behavior, in: 2015 IEEE 5th International Conference on Consumer Electronics - Berlin (ICCE-Berlin), 2015, pp. 358–362, doi:10.1109/ICCE-Berlin.2015.7391280.
[10] S. Majumder, E. Aghayi, M. Noferesti, H. Memarzadeh-Tehran, T. Mondal, Z. Pang, M.J. Deen, Smart homes for elderly healthcare–recent advances and research challenges, Sensors 17 (11) (2017) 2496.
[11] E. Aghayi, M. Khansari, H. Memarzadeh-Tehran, A new back-off mechanism for the S-MAC protocol with applications in healthcare, in: 7'th International Symposium on Telecommunications (IST'2014), IEEE, 2014, pp. 569–573.
[12] M. Hoseini, F. Saghafi, E. Aghayi, A multidimensional model of knowledge sharing behavior in mobile social networks, Kybernetes 48 (5) (2019) 906–929.
[13] N.H. Duong, H.D. Hai, A semi-supervised model for network traffic anomaly detection, in: 2015 17th International Conference on Advanced Communication Technology (ICACT), IEEE, 2015, pp. 70–75.
[14] P. Velan, M. Čermák, P. Čeleda, M. Drašar, A survey of methods for encrypted traffic classification and analysis, Int. J. Netw. Manage. 25 (5) (2015) 355–374.
[15] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, C. Williamson, Offline/realtime traffic classification using semi-supervised learning, Perform. Eval. 64 (9–12) (2007) 1194–1213.
[16] E. Mahdavi, A. Fanian, H. Hassannejad, Classification of encrypted traffic for applications based on statistical features, ISC Int. J. Inf. Secur. 10 (1) (2018) 29–43.
[17] C.-Y. Chiu, Y.-J. Lee, C.-C. Chang, W.-Y. Luo, H.-C. Huang, Semi-supervised learning for false alarm reduction, in: Industrial Conference on Data Mining, Springer, 2010, pp. 595–605.
[18] Y. Wang, Y. Xiang, J. Zhang, S. Yu, A novel semi-supervised approach for network traffic clustering, in: 2011 5th International Conference on Network and System Security, IEEE, 2011, pp. 169–175.
[19] P.V. Amoli, T. Hämäläinen, A real time unsupervised NIDS for detecting unknown and encrypted network attacks in high speed network, in: 2013 IEEE International Workshop on Measurements & Networking (M&N), IEEE, 2013, pp. 149–154.
[20] C. Chen, Y. Gong, Y. Tian, Semi-supervised learning methods for network intrusion detection, in: 2008 IEEE International Conference on Systems, Man and Cybernetics, IEEE, 2008, pp. 2603–2608.
[21] M. Noferesti, R. Jalili, *HB²DS*: a behavior-driven high-bandwidth network mining system, J. Syst. Softw. 127 (2017) 266–277.
[22] H.R. Loo, M.N. Marsono, Online network traffic classification with incremental learning, Evol. Syst. 7 (2) (2016) 129–143.
[23] T. Zhang, R. Ramakrishnan, M. Livny, Birch: an efficient data clustering method for very large databases, in: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, in: SIGMOD '96, ACM, New York, NY, USA, 1996, pp. 103–114, doi:10.1145/233269.233324.
[24] Z. Nazari, M. Noferesti, R. Jalili, DSCA: an inline and adaptive application identification approach in encrypted network traffic, in: Proceedings of the 3nd International Conference on Cryptography, Security and Privacy, ACM, 2019.
[25] M. Noferesti, R. Jalili, Inline high-bandwidth network analysis using a robust stream clustering algorithm, IET Inf. Secur. (2019).
[26] J.A. Silva, E.R. Faria, R.C. Barros, E.R. Hruschka, A.C.P.L.F. de Carvalho, J. Gama, Data Stream Clustering: A Survey, ACM Comput. Surv. 46 (1) (2013) 13:1–13:31, doi:10.1145/2522968.2522981.
[27] J. Gama, Knowledge Discovery from Data Streams, CRC Press, 2010.
[28] C.C. Aggarwal, C.K. Reddy, Data Clustering: Algorithms and Applications, CRC Press, 2013.
[29] C.C. Aggarwal, Data Mining: the Textbook, Springer, 2015.

[30] D.J. Weller-Fahy, B.J. Borghetti, A.A. Sodemann, A survey of distance and similarity measures used within network intrusion anomaly detection, IEEE Commun. Surv. Tutorials 17 (1) (2015) 70–91.
[31] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. (CSUR) 46 (4) (2014) 44.
[32] B. Ci, Confidence intervals, Lancet 1 (1987) 494–497.
[33] Data plane developement kit, (Accessed: 2018-02-24). https://dpdk.org.
[34] L. Deri, M. Martinelli, T. Bujlow, A. Cardigliano, nDPI: open-source high-speed deep packet inspection, in: 2014 International Wireless Communications and Mobile Computing Conference (IWCMC), IEEE, 2014, pp. 617–622.
[35] T. Bujlow, V. Carela-Español, P. Barlet-Ros, Independent comparison of popular DPI tools for traffic classification, Comput. Netw. 76 (2015) 75–89.
[36] T. Bujlow, V. Carela-Español, P. Barlet-Ros, Extended independent comparison of popular deep packet inspection (DPI) tools for traffic classification, 2014.
[37] B. Wang, J. Zhang, Z. Zhang, W. Luo, D. Xia, Traffic identification in big internet data, in: Big Data Concepts, Theories, and Applications, 2016, p. 129.
[38] VPN-nonVPN dataset (ISCXVPN2016), 2016. https://www.unb.ca/cic/datasets/vpn.html.
[39] Tor-nonTor dataset (ISCXTor2016), 2016. https://www.unb.ca/cic/datasets/tor.html.
[40] G. Draper-Gil, A.H. Lashkari, M.S.I. Mamun, A.A. Ghorbani, Characterization of encrypted and VPN traffic using time-related, in: Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP), 2016, pp. 407–414.
[41] W. Wang, M. Zhu, J. Wang, X. Zeng, Z. Yang, End-to-end encrypted traffic classification with one-dimensional convolution neural networks, in: 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), IEEE, 2017, pp. 43–48.
[42] M. Lotfollahi, M.J. Siavoshani, R.S.H. Zade, M. Saberian, Deep packet: a novel approach for encrypted traffic classification using deep learning, Soft Comput. (2017) 1–14.
[43] A.H. Lashkari, G. Draper-Gil, M.S.I. Mamun, A.A. Ghorbani, Characterization of tor traffic using time based features, in: ICISSP, 2017, pp. 253–262.
[44] E. Hodo, X. Bellekens, E. Iorkyase, A. Hamilton, C. Tachtatzis, R. Atkinson, Machine learning approach for detection of nontor traffic, in: Proceedings of the 12th International Conference on Availability, Reliability and Security, ACM, 2017, p. 85.
[45] B. Lantz, B. Heller, N. McKeown, A network in a laptop: rapid prototyping for software-defined networks, in: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, in: Hotnets-IX, ACM, New York, NY, USA, 2010, pp. 19:1–19:6, doi:10.1145/1868447.1868466.
[46] A. Botta, A. Dainotti, A. Pescape, A tool for the generation of realistic network workload for emerging networking scenarios, Comput. Netw. 56 (15) (2012) 3531–3547.
[47] Distributed Internet Traffic Generator, (Accessed: 2019-09-15). http://www.grid.unina.it/software/ITG/.
[48] F. Farabi, M. Mosavi, S. Karami, Optimal choice of random variables in D-ITG traffic generating tool using evolutionary algorithms, Iran. J. Electr. Electron. Eng. 11 (2) (2015) 101–108.
[49] Hping Command Description, (Accessed: 2018-04-08). http://www.hping.org.

Morteza Noferesti received his bachelor's degree in information technology from Shiraz University of Technology, Shiraz, Iran, in 2009. He also received his master's degree from Sharif University of Technology, Tehran, Iran in 2011. He received his Ph.D. in computer engineering from Sharif University of Technology in 2019. His research interests include big data security, highbandwidth network processing, intrusion detection, and big stream mining.

Rasool Jalili received his bachelor's degree in computer science from Ferdowsi University of Mashhad in 1985, and his master's degree in computer engineering from Sharif University of Technology in 1989. He received his Ph.D. in computer science from The University of Sydney, Australia, in 1995. He then joined the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran, in 1995. He has published more than 150 papers in Computer Security and Pervasive Computing in international journals and conferences proceedings. He is now an associate professor in Sharif. His research interests include access control, vulnerability analysis, and database security-which he conducts at his network security laboratory (DNSL, dnsl.sharif.ir).