The ISC Int'l Journal of Information Security
ISeCure

January 2018, Volume 10, Number 1 (pp. 29–43)
http://www.isecure-journal.org

Encrypted Traffic Classification Using Statistical Features

Ehsan Mahdavi [1,*], Ali Fanian [1], and Homa Hassannejad [1]
[1] *Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran*

A R T I C L E I N F O.

Article history:
Received: 13 August 2017
First Revised: 31 December 2017
Last Revised: 4 January 2018
Accepted: 28 January 2018
Published Online: 31 January 2018

Keywords:
Internet Traffic Classification, Encrypted Traffic, Graph Theory.

Abstract

Traffic classification plays an important role in many aspects of network management such as identifying type of the transferred data, detection of malware applications, applying policies to restrict network accesses and so on. Basic methods in this field were using some obvious traffic features like port number and protocol type to classify the traffic type. However, recent changes in applications make these features imperfect for such tasks. As a remedy, network traffic classification using machine learning techniques is now evolving. In this article, a new semi-supervised learning is proposed which utilizes clustering algorithms and label propagation techniques. The clustering part is based on graph theory and minimum spanning tree algorithm. In the next level, some pivot data instances are selected for the expert to vote for their classes, and the identified class labels will be used for similar data instances with no labels. In the last part, the decision tree algorithm is used to construct the classification model. The results show that the proposed method has a precise and accurate performance in classification of encrypted traffic for the network applications. It also provides desirable results for plain un-encrypted traffic classification, especially for unbalanced streams of data.

1 Introduction

Network traffic classification provides many features for network administrators. Many network management tasks like bandwidth management and guarantee for quality of services, rely on accurate network traffic classification. An accurate network traffic classification provides Internet Service Providers (ISPs) the ability to provide reliable methods for improving quality of their services with respect to application specific requirements. Therefore, the first step to achieve these goals is to identify and classify the traffic of applications and services in the network. On the other hand, growth of malwares and their techniques to hide themselves from being detected by Intrusion Detection Systems (IDSs) and to bypass network firewalls makes traffic classification an important role in network security and cyber threats. Current network traffic classification methods can be categorized into the following four major classes:

- Port based classification: traditionally, traffic classification was done by port numbers defined by IANA for network transport layer. Although this is a quick and simple method to classify traffic of basic applications, with the growth of new applications such as Peer-To-Peer (P2P) applications it is becoming more and more inefficient. The nature of P2P applications tends to bypass firewalls and IDSs by using non-standard

* Corresponding author.
Email addresses: e.mahdavi@ec.iut.ac.ir (E. Mahdavi), a.fanian@cc.iut.ac.ir (A. Fanian), h.hassannejad@ec.iut.ac.ir (H. Hassannejad)
ISSN: 2008-2045 © 2018 ISC. All rights reserved.

port numbers or masquerading other standard application port numbers [1].

- Payload based classification: to identify traffic type, the whole payload of network packet is being analyzed in this category. The first major challenge ahead of this method is to analyze the payload of every single network packets, which requires heavy processing efforts. In addition, contents of encrypted packets are ambiguous for these methods. These restrictions make them unusable for such cases. Respecting the privacy of the users is another important challenge which these methods face.
- Host based classification: in this category, the host behavioral patterns are being modeled. The job is done by utilizing heuristic methods without using network packet contents. For example, these patterns could contain information about connected hosts, number of different ports in a connection and the transport protocols used in a connection. The accuracy of these methods is strictly tied to the topological position of assets being watched. In recent researches, these methods are not used by themselves but in conjunction with other methods.
- Feature based classification: considering the stated limitations for the above categories, feature based classification methods got more attention. In this category a set of statistical features are used as classification criterion. This approach assumes that network layer traffic of each application has unique statistical features which can be used to distinguish them. Packet length distribution, the idle time between streams, inter arrival time of the packets and the length of the packets are among these features. These methods analyze the data values for these features instead of analyzing packet payloads. This brings high efficiency for fast network transactions and also privacy for the users [2–5]. Moreover, as there is no need of the packet payloads to obtain the features, it is also possible to utilize these methods to classify encrypted network traffic [3, 4, 6–8].

Many feature based classification methods, use machine learning algorithms to determine the traffic labels. In this regard, machine learning algorithms fall in three categories of supervised, unsupervised and semi-supervised algorithms [2–4, 9]. Supervised algorithms need pre-classified data instances to build a classification model. The model can be used later to classify new instances. Although supervised algorithms are highly accurate, generating fully labeled data is cumbersome and in some cases impossible. Un-supervised algorithms put data instances with similar features in same groups. These methods do not need pre-labeled data, so they are considered cheap and simple methods but their efficiency is not suitable. Semi-supervised algorithms are somewhere in the middle. Semi-supervised methods tend to utilize benefits of both other categories while avoiding their deficiencies.

A new classification method proposed in this article to classify application layer protocols. The method which is a semi-supervised algorithm can efficiently be used for encrypted traffic like SSH [10]. The proposed method, will construct its classification model with datasets that only 10% of their instances are labeled. At the first step, the method clusters the dataset, after labeling clusters and streams in them, a classification algorithm uses labeled streams and the constructed classification model can be used to classify new network streams.

Network streams for each Internet protocol, make clusters with different forms. Utilizing a graph based clustering algorithm and applying MST algorithms on the clusters, the proposed method is able to distinguish clusters with different shapes and density and also clusters with imbalance sizes. Technical results show the quality of the clusters and classification model built using them.

The remaining sections of this article are organized as follows. In Section 2 we review the related work in the literature. The proposed methods are explained in Section 3. Section 4 assesses the efficiency of the proposed method and finally in the last section we conclude the paper.

2 Related Work

As mentioned before, port based and payload based methods are facing important limitations. As a result, recent methods mostly rely on statistical features to identify different applications. The rationale behind these methods is based on the fact that traffic in different network applications have some unique features which make them distinguishable from each other.

More and Zoe [5] utilized Bayesian analysis and showed the naÃŕve bayes classifier can be efficient with slight modifications. These modifications contained two plugins. The first add-on was to use kernel density estimation. While naÃŕve bayes classifier builds the model using normal distribution, kernel estimation methods can utilize different kernel functions to build it. The second trick was to use fast correlation technique to select optimal features. By these modifications, bayes classifier achieved classification accuracy of 96% for different network traffic.

Utilizing naÃŕve bayes classifiers is an ongoing method. Zhang *et al.* in [11] showed that it can be efficiently used for encrypted traffic classification with few labeled data in hand. They correlate flows by naÃŕve bayes classifier in Bag of Flows (BoF). Then they can process these BoFs and aggregate the naÃŕve bayes predictions of those bags. Extensive experiments on real world data sets proved performance of their method.

In [12] the efficiency of C4.5 decision tree algorithm, neural networks, naÃŕve bayes and naÃŕve bayes tree classifiers are compared. Results showed that C4.5 algorithm is faster than others. Kim *et al.* [13] compared results of 9 different learning algorithms. The results showed that Support Vector Machine (SVM) classifiers are more accurate. The total accuracy of SVM classifiers was reported to be 94.2%.

Speaking of supervised learning, deep learning is in a rise. Lotfollahi and others claim to be the first who proposed a method based on deep learning for encrypted traffic classification in [14]. Deep packet, proposes a framework that combines stacked autoencoder and convolution neural network (CNN) for network traffic classification. They also claim that deep learning outperforms other methods in encrypted network traffic classification.

A recent study in [15] compared 6 classification models namely logistic regression, support vector machine, NaÃŕve Bayes, k-nearest neighbor, Random Forest (RF) and Gradient Boosting Tree (GBT). They showed ensemble methods RF and GBT with some optimization can prove efficiency over other mentioned classifiers. Those optimizations worked for datasets time-related features.

One of the antecedent methods in unsupervised learning for network traffic classification was proposed by McGregor in [16]. This research concentrated on common applications like HTTP, FTP, SMTP, IMAP, NTP and DNS.

In [17] some clustering algorithms like k-means, Gaussian Mixture Model (GMM), hidden markov model and spectral clustering was used to cluster TCP connections. The learning phase outputs two different sets. The first set contains a description of the clusters and the second contain applications distinguished be each cluster. In the classification part, a stream of packet headers is used as input and some information like source and destination port number, source and destination IP headers, connection protocol and packet length of first P packets of each TCP connection are extracted.

In [18] a new unsupervised learning method is introduced. In the first step of the method the statistical features are used to build the clusters and the bag of words model is used to describe the contents of each cluster. Bag of words is a common model to describe the documents in information retrieval and natural language processing.

In [19] a semi-supervised algorithm is introduced which consists of the two step of clustering and classification. The latter one assigns cluster labels. For the clustering, the K-means algorithm was used. The data instances are divided in K clusters with respect to their feature values. Data instances in the same cluster have similar feature values. K is a native integer which specifies the number of different clusters and should be defined somehow prior for the algorithm. For each cluster a central instance is defined and other data instances will be divided into cluster with respect to their distance with these central instances. Each new data added to a cluster and the central instance might update and change until the algorithm becomes stable and no changes are made during different iterations.

In [20] a semi-supervised method is proposed using constrained clustering algorithm. Constrained clustering works according to Must-link and Cannot-link. Must-link relations specify which data instances should be in same cluster. On the other hand, Cannot-link constraints will forbid some data instance to be in same clusters. In network flow context, the correlation between data streams might be as used Must-link relations.

3 The Proposed Method

In this section, a novel semi-supervised based method is proposed to classify network traffic. The purpose of the method is to classify encrypted traffic for application layer protocols. In this regard, although it is shown in the performance evaluation section that the proposed method can classify different application layer protocols, considering the availability of SSH encrypted datasets, it is taken under the focus. The proposed classification method will utilize graph theory clustering, MST algorithm and C4.5 decision tree. The method will construct a classification model in two major steps of clustering and classification consequently. In the proposed method the class of network traffic is automatically determined. Figure 1 shows a generalized structure of the proposed method.

The classification model is built in two steps of clustering and classification. In the first step, a clustering is done with respect to statistical feature values. Using un-labeled network flows, we try to build high quality clusters. The clustering is done using graph theory and MST algorithm. Afterwards, the limited number of labeled data instances are used for labeling clusters. Some clusters might not contain any labeled

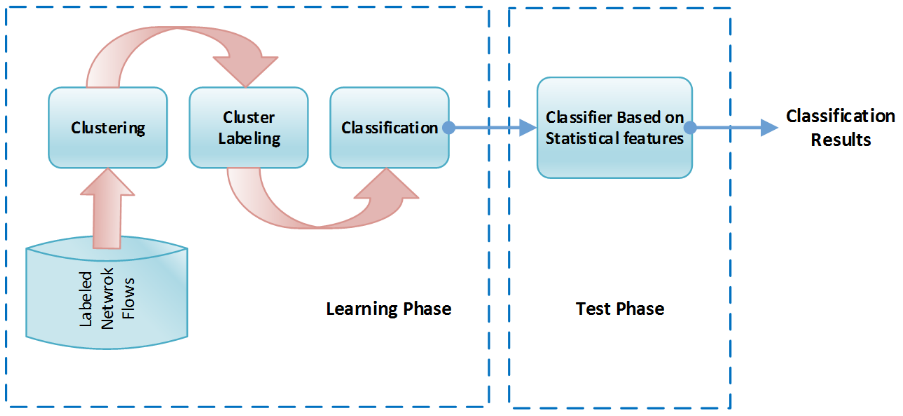

Figure 1. General structure of the proposed method

data instances so their label is determined via asking expert. Then regarding the intra-cluster distances, clusters with the same label might get merged. At the end, labeling the clusters results in all data instances to have labels and get used as input to the classification step. The classification model is obtained using C4.5. Finally, this semi-supervised model is able to classify new network streams using their statistical features. In next subsections each part of the proposed model will be explained.

3.1 Learning Phase

The learning phase in the proposed method is a semi-supervised classification. The input to this phase is a collection of training data instances which at most 10% of them are labeled and the learning is done with respect the values of their features. If D is the training set, then $D = L \cup U$ in which L is the collection of labeled and U is the collection of un-labeled network streams. The learning phase of the proposed scheme can be divided into this 3 blocks:

- Clustering,
- Labeling clusters and merging them,
- Building the classifier model.

The first part of the learning phase, i.e., clustering, utilizes a graph based algorithm. Nodes in this graph represent the labeled network flows and the edges show the distance of these flows. The graph of flows is clustered using graph algorithms such as MST algorithm. Using the labeled network flows the label of other un-labeled flows are assigned. Finally utilizing these labeled data, a supervised learning method is used the build a classifier. The details of each parts are exposed in the following.

3.1.1 Clustering

The final goal of the clustering part is to divide data instances into relative clusters with respect to the structure of the dataset. This is done by utilizing data similarity measures. For the clustering we assume that there is no initial knowledge about the dataset such as number of clusters and the distribution of data instances amongst them. Clustering is a powerful mean in the fields of machine learning and data mining.

Difference in distributions of datasets causes the constructed clusters to be of arbitrary shapes, different densities and even un-balanced sizes. However, in the most clustering methods, some initial information such as number of clusters is assumed to be available, they donâĂŹt consider some issues related to the unbalanced data. So, the proposed method somehow covers these issues. In this section clustering algorithm based on graph theory and MST is used.

As shown in Figure 2, the final clusters are yield after two level of clustering. In both levels the network flows are considered as nodes in the graph and distance between them construct the edges of the trees. Dissociation of the flows and formation of the clusters is done by eliminating the edges of the resulting trees. Elimination of the edges is inspired by the method proposed in [21]. In the following part, if needed the impure clusters of the first part, are decomposed into new smaller clusters using a new method.

As mentioned before, $D = L \cup U$ is a set consisting of labeled network flows $L = \{x_i^{(l)}, i = 1, \ldots, n\}$ and un-labeled network flows $U = \{x_i, i = 1, \ldots, m\}$. For each network flow, $x_i = (x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{id}) \in ??^d$ represents the set statistical features and x_{ij} is a single feature of the flow x_i. The labels of the flows are from the set $B(l \in B)$.

First a weighted graph is built using the set of training samples D. As noted before the set of the

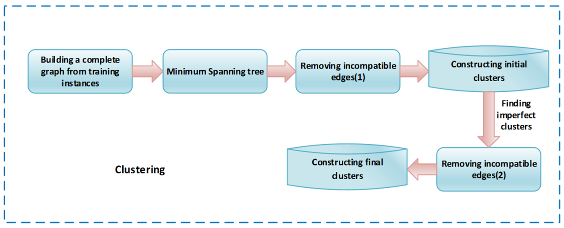
Building a complete graph from training instances
Minimum Spanning tree
Removing incompatible edges(1)
Constructing initial clusters
Finding imperfect clusters
Removing incompatible edges(2)
Constructing final clusters
Clustering

Figure 2. clustering scheme in the proposed method.

network flows (consisting of labeled and unlabeled) are the nodes of the graph and the distance between them construct the edges of the graph.

Definition 1. Weighted Undirected Graph: Graph $G(X) = (V, E)$ is a weighted undirected graph in which V is the set of vertices and $E = (x_i, x_j)x_i, x_j \in X, i \neq j$ is the set of graph edges. Each edge $e = (x_i, x_j)$ in graph has the length (weight) of $d(x_i, x_j)$.

In the proposed method, $G(X) = (V, E)$ is complete weighted undirected graph. In G, $V = D$ is the set of graph vertices and $E = \{(x_i, x_j) \mid x_i, x_j \in D, i \neq j\} e = (x_i, x_j)$ is the set edges of the graph. Each edge defined distance between the two network is the d_{mix} in which $d_{mix} = (x_i, x_j)$ graph has a length of the in streams. Common distance functions like Euclidean, Mahalanobis or other arbitrary distance functions can be used. In this research, instances are network flows and their features can be either numerical or nominal. So we use a mixed distance function of Euclidian and stream distances. The distance function is shown in relation Equation 1.

$$d_{mix} = \sqrt{d_n(x_i, x_j) + d_c(x_i, x_j)} \tag{1}$$

$$d_n(x_i, x_j) = \sum_{d_k \in NUM} (x_{i,k} - x_{j,k})^2 \tag{2}$$

$$d_c(x_i, x_j) = \sum_{d_k \in CAT} \delta(x_{i,k} - x_{j,k}) \tag{3}$$

where $\delta(a_1, a_2) = 1$ if $a_1 \neq a_2$ and $\delta(a_1, a_2) = 0$ if $a_1 = a_2$. Also, d_{mix} is the distance function that we use in this article, and d_n is the Euclidian distance of numerical features and d_c is the distance for nominal feature values. A clustering algorithm will divide dataset X into K clusters of $C_1, C_2, \ldots, C_K$. In this division these conditions exist:

$$\begin{cases} C_i \neq \emptyset \\ C_i \cap C_j = \emptyset \\ X = C_1 \cup C_2 \cup \ldots \cup C_k \\ i = 1 : k; j = 1 : k \end{cases} \tag{4}$$

As we represented the collection of network stream by means of graphs, so clustering them results in dividing the graph into K subgraphs, each representing a cluster. To construct the clusters in clustering one needs to define a separation criterion. In graph of network streams, definition of separation criteria has the certain meaning of finding incompatible edges. Incompatible edges are the edges which connect irrelevant network flows and clustering process should be the best candidates for elimination. Topologically, incompatible edges are the inter-cluster edges that eliminating them to form subgraphs (clusters) correctly. Proper separation criterion distinguishes inter-cluster edges from intra-cluster ones and makes correct network traffic classes by eliminating them.

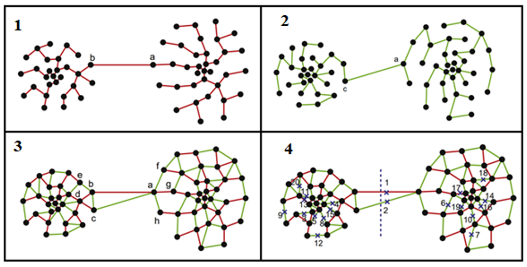

Figure 3. first and second MSTs [22].

The proposed method uses Minimum Spanning Trees in graph clustering phase. A minimum spanning tree (MST) or minimum weight spanning tree is a subset of the edges of a connected, edge-weighted (un)directed graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight. However, using only the first MST may not produce reliable clusters. Given the nature of a tree, removing a single edge results in two separate subgraphs. On the other hand, the 1st MST might not contain enough edges to represent the original graph. It means that it cannot represent the network flows correctly. So we use 1st and 2nd MSTs together and apply the clustering algorithm on it.

In Figure 3 first and second MSTs of an arbitrary graph are shown respectively in section 1 and 2. 3rd and 4th section of that figure show the graph of those MSTs combined together. Specifically, the 4th section of the figure, shows the subsets that can be separated with respect to their distances. Edge ab is a correct candidate for elimination which results a valid separation. This edge can be considered incompatible.

Regarding the edge ab of Figure 3, a criterion for determining incompatible edges can be defined as follows: "Each edge with length significantly greater than the average length of other edges connected to the both ends of that edge, are considered as incompatible and should be eliminated" [21, 22].

This criterion only covers separation of clusters with respect to their distances. Another measurement that commonly is used as separation criterion in clustering algorithms is cluster densities. According to Figure 3, in [22], it was stated that the major difference between the cluster separation using two measures of cluster distances and cluster densities, is that the average length of connected edges to the ending vertices of incompatible edges behave differently with respect to these measures. In clusters that are distinguishable using cluster distances, it is common that the stated average lengths are homogenous while with respect to the density they differ significantly.

To take these differences into account, a new criterion can be defined which covers both measures. In combined graph of first and second MSTs together, eliminating at least two edges yields a graph cut.

Definition 2. In graph $G(X) = (V, E)$ and $e \in E$, if eliminating edge e yields two subgraphs $G_1(X) = (V_1, E_1)$ and $G_2(X) = (V_2, E_2)$ such that $V = V_1 + V_2$, then a graph cut is happened. To identify the inter-cluster, for each edge itâĂŹs length is compared to the minimum value of the average length of the connected edges to the vertices of both sides of that edge. This comparison could be done by their relation ratio. The weight of each edge in the combined graph of 1st and 2nd MSTs is defined as follows:

Graph $G_{mst}(X) = (V, E_{mst})$ is the graph from combining the 1st and 2nd MSTs in which $e_{ab} \in E_{mst}$ and $a, b \in V$. Weight of edge ab is defined as follows:

$$w(e_{ab}) = \frac{d(e_{ab}) - \min(avg(E_a - \{e_{ab}\}), avg(E_b - \{e_{ab}\}))}{d(e_{ab})} \qquad (5)$$

where $E_a = \{e_{ij} \mid (e_{ij} \in E_{mst}) \wedge (i = a \vee j = a)\}$, $E_b = \{e_{ij} \mid (e_{ij} \in E_{mst}) \wedge (i = b \vee j = b)\}$ and $avg(E) = \frac{1}{|E|} \sum_{e \in E} d(e)$. As mentioned before $d(e)$ is the defined distance between network flows. To define weight of an edge, consider the connected edges to both ends of that edge. Then consider average length of those edges for both sides and find which is minimum. The weight of an edge is defined to be the ratio of its length the mentioned minimum value. The smaller this ratio is, the mentioned weight will be greater and vice versa. So, the larger the weight of an edge, it is most probable to be an inter-cluster edge. Then it is a good candidate to be eliminated. Figure 3 shows a problem of cluster separation by distances. According to Figure 1, edge ab is inter-cluster and should be removed. Edge cd is an intra-cluster edge and should remain. The weight of ab and cd are 10 and 5 respectively. Therefore, edge ab is a better candidate to be eliminated.

The weight defined in (3-5) considers only local edges connecting to both ends of the edge being checked. This is where some exceptions arise. If the edge being checked is an intra-cluster edge far from any inter-cluster edge, but with weight greater than average of that local edges, it will be removed. To remedy this exception and some other likewise problems, the weight of an edge in 1st and 2nd MSTs based graph is defined as:

$$w(e_{ab}) = \delta \times \frac{d(e_{ab}) - \min(avg(E'_a - \{e'_a\}), avg(E'_b - \{e'_b\}))}{d(e_{ab})} + (1 - \delta) \times d(e_{ab}) \qquad (6)$$

where $E'_a = E_a - \{e_{ab}\}$, $e'_a = argmax_{e \in E'_a}(d(e))$, $E'_b = E_b - \{e_{ab}\}$, $e'_b = argmax_{e \in E'_b(d(e))}$ and δ is a penalty factor for the length of the edge which resides in $0 \leq \delta \leq 1$. Terms $E'_a - \{e'_a\}$ and $E'_b - \{e'_b\}$ cause omitting longest edge length in computing weight e_{ab}. The penalty factor δ holds a trade-off between length of the edge and the ratio of $\frac{d(e_{ab}) - \min(avg(E'_a - \{e'_a\}), avg(E'_b - \{e'_b\}))}{d(e_{ab})}$.

The defined weight in 1st and 2nd MSTs based graph exposes useful concepts:

- In a graph, weight of inter-cluster edges is extensively larger than others. This helps to find graph cuts and to constitute subgraphs(cluster) by removing the longer edges sorted by their weights in a descending order.
- Inter-cluster edges are distributed equally between T_1(1st MST) and T_2(2nd MST). This feature facilitates determining incompatible edges.
- Except the inter-cluster edges, most weighted edges reside in T_2.

According to above benefits, a suitable criterion for determining incompatible edges in 1st and 2nd MSTs based graph is finding an edge with most weight in that graph which removing it results in a valid graph cut. The edges are candidate to be eliminated with respect to their weight in a descending order. $Rank(E_{mst})$ is list of the edges in 1st and 2nd MSTs based graph, which is sorted in a descending order.

After removing each edge, obtaining a graph cut is examined. In 1st and 2nd MSTs based graph, each cut is yield be removing at least 2 edges. So on constitution of each graph cut, we have a set of removed edges. Assume that is the mentioned set. If the following equation is established the graph cut is valid and there will be 2 subgraphs in which each one is considered as cluster:

$$Ratio(E_{gcut}) = \frac{\min(|E_{gcut} \cap T_1|, |E_{gcut} \cap T_2|)}{|E_{gcut}|} \geq \lambda \qquad (7)$$

Otherwise the cut is invalid and the whole graph must remain as a cluster. λ is an empirical threshold. According to above, incompatible edges are the ones with most weight in $Rank(E_{mst})$ which hold 3-7. This means removing them makes a valid graph cut. In graph separation with respect to the distances, number of the edges that are removed from T_1 is approximately equal to the number of eliminated edges from

T_2. It is expected that $\lambda = 0.5$ is a suitable choice. The criterion is not held about graph separation with respect to cluster densities. Our experiments showed that $\lambda = 0.33$ is suitable choice for the threshold which best covers both measures. By eliminating incompatible edges, 1st and 2nd MSTs based graph will be divided to some clusters. For each cluster, the incompatible edge removal is continued until no more incompatible edges could be found. This means that the clusters could not be divided to smaller ones. The algorithm of the process is as below:

Algorithm 1 Clustering algorithm

```
Input: $G(X) = (V, E)$ the graph of the training network flow
Output: $out$ a set of clusters(subgraphs)
 1: Find 1st and 2nd MSTs $T_1$ and $T_2$
 2: Build 1st and 2nd MSTs based graph $G_{mst}(X)$
 3: Create tables $open = G_{mst}X)$ and $closed = \emptyset$
 4: If $open = \emptyset$ and $out = closed$ then end the algorithm
 5: If $open = \emptyset$ then grab $G'_{mst} = (V', E')$ from $open$
 6: Calculate weight of the edges in $G'_{mst} = (V', E')$
 7: Create list $Rank(E')$
 8: Find the best edge candidate for removal from $Rank(E')$
 9: If a graph cut happened go to 11.
10: If a graph cut not happened go to 8.
11: If the graph cut is valid subgraphs $G'_1$ and $G'_2$ is added to the table $open$ and go to 4.
12: If the graph cut is not valid move $G'_{mst} = (V', E')$ from $open$ to $closed$ and go to 4.
```

Finding imperfect clusters:
At the end of the clustering, network flows might fall within clusters of different size and shapes. The proposed method tends to build an accurate classification model and finding perfect clusters has an important effect on this matter. Perfect clusters are the ones that all or most of their instances belong to the same class of traffic. In this regard, the quality of produced clusters with respect to standard deviation of the length of the intra-cluster edges is measured.

Definition 3. If $G' = (V', E')$ is a produced cluster or subgraph (subtree) of the clustering phase and N is the number of the edges in graph $G'(X)$ and $d(e')$ is the length of and edge, then the standard deviation of the length of the intra-cluster edges are measured like:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(d(e_i) - \overline{d(e)})^2} \qquad (8)$$

Here $\overline{d(e)}$ in the average length of edges and can be calculated as below:

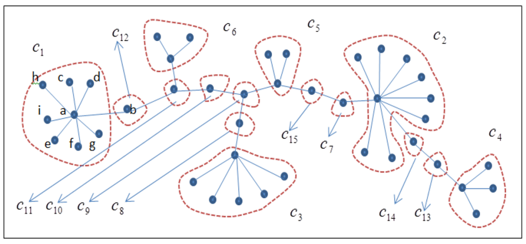

Figure 4. primary clusters in second phase of clustering.

$$\overline{d(e)} = \frac{1}{N}\sum_{i=1}^{N} d(e_i) \qquad (9)$$

Considering a threshold β, if for a cluster $\sigma \geq \beta$ then the cluster will be checked to see if it can be divided into smaller clusters.

Procedure of removing incompatible edges
Assume $C = \{c_1, c_2, c_3, \ldots, c_n\}$ is the set of the produced clusters of the last steps. $\sigma \geq \beta$ is checked on all produced clusters. If it holds for a cluster, then the cluster will be checked to see if it can be divided into smaller clusters. Assume c_i is a cluster with $\sigma \geq \beta$. Each cluster in C is an MST, so c_i can be showed as $T = (V, E)$ in which V is the set of it's vertices and E the set of edges. In T, $deg(v_i)$ is the degree of each vertex where $v_i \in V$.

In this step the proposed method is about to reduce impurity of the clusters by dividing them into smaller clusters. For this purpose, degree of each vertex in T is calculated and they get sorted in a descending order with respect to the calculated degree. Considering the vertex with most degree, if all adjacent vertices are of degree 1, they will be in same cluster with that vertex. If one of them has degree more than 1 then it will be a cluster (Figure 4).

As shown in Figure 4, each vertex with its adjacent vertices of degree 1 is in the same cluster. For example, consider c_1. Vertex a with degree of 8 and its 7 adjacent vertices of degree 1 ($cdefgh$) are forming a single cluster. Vertex b is adjacent to a but its degree is greater than 1. So it creates its own cluster named c_{12} and as it hasn't any adjacent vertices of degree 1, cluster c_{12} has only one member b.

Consider vertex $v_i \in V$ in tree and assume $N_i = \{n_{i1}, n_{i2}, \ldots, n_{ij}, \ldots, n_{im}\}$, $j = 1\ldots, m$ is the set of adjacent vertices of v_i. While creating clusters of phase 2, if $deg(n_{ij}) = 1$, $n_{ij} \in N_i$ then it will be in the same cluster with v_i.

To obtain an optimal number of clusters, after creating these clusters, if necessary, some of the clusters get merged together. For this purpose, the following parameters are calculated and if a merge happened they should be updated.

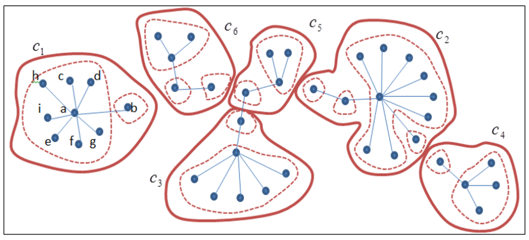

Figure 5. Final clusters of second phase of clustering.

- Cluster center $center_i$
- Adjacent vertices N_i
- Inter-cluster edges out_i
- Intra-cluster vertices vin_i
- Cluster standard deviation sd_i (respect to the length of Intra-cluster edges)
- Cluster Degree $deg\ c_i$ (maximum degree of vertices in cluster)

To start merging, the clusters are sorted descending with respect to their degrees and merge procedure starts with cluster with smallest degree. As shown in Figure 4, clusters c_7 to c_{15} have the minimum degree value, so each one can be the starting candidate. Assume c_{12} is the choice. The distance of center for c_{12} ($center_{12}$) should be calculated with neighboring cluster centers ($center_{11}$) and ($center_1$). Two neighbors with nearest clusters now get merged. To ensure that the merge was useful difference of standard deviation for merged clusters sd_{merg} and stand deviation of two initial clusters sd_1, sd_{12} is calculated.

$$sd_{merg} - sd_{12} = a_1 \quad (10)$$
$$sd_{merg} - sd_1 = a_2 \quad (11)$$

The merge is successful if 3-11 is hold, else the merge should be reverted and the edge between them eliminated.

$$\begin{cases} a_1 \wedge a_2 \geq \theta \\ a_1 \wedge a_2 < \theta \end{cases} \quad (12)$$

The θ in 3-12 is a threshold which is studied in section âĂŐ4.

After merging two clusters, all mentioned parameters should be updated. Merge procedure is continued until no inter-cluster edges exist. In other words, no more clusters could be merged. For example, Figure 5 shows the results of the algorithm for the last example.

As can be seen in Figure 5, initial 15 clusters are reduced to 6 after applying this algorithm. The resulting clusters are expected to be more accurate in comparison with first step clusters.

The merge algorithm is as follows:

Algorithm 2 Merge step

```
Input: Clusters produced by algorithm 1 (MST $T$)
Output: More accurate clusters
1: 1. Build the primitive clusters with respect to vertices degrees and store in List *initial.*
2: Calculate the following parameters for each $Cluster_i$ in *initial* List. Center for each $cluster_i$, Adjacent clusters $N_i$, Intra-cluster edges $in_i$, Inter-cluster edges $out_i$, Intra-cluster vertices $vin_i$, Standard deviation of the cluster $sd_i$ (With respect to length of the edges in cluster), Degree of the cluster $degc_i$ (Maximum degree of the vertices in the cluster)
3: Sort the clusters based on cluster rank in ascending order and store them in *clusters* list.
4: Store all inter-cluster edges in $inter\_clusters\_edges$ list.
5: If $inter\_clusters\_edges \neq \emptyset$, go to 7.
6: If $inter\_clusters\_edges = \emptyset$, end the algorithm.

7: Calculate the distance of center of first cluster $c_i$ in *clusters* list to center of its adjacent clusters $N_i$.
8: Select the cluster that satisfies $c_{select} = argmin_{j=1,\dots,m,n_{ij}\in N}\{d(center_i, center_{n_{ij}})]$ and store it in $c_{select}$
9: Merge $c_i$ and $c_{select}$.
10: If the merge is not appropriate, assume the edge between $c_i$ and $c_{select}$ as incompatible, remove it and go to 4.
```

Finally, clusters obtained from algorithm 1 and 2 will go as input of labeling step. In next section the labeling phase is explained.

3.1.2 Labeling

The resulting clusters of the clustering part will be used as input for classification phase. Classification is a supervised learning method. So, it needs labeled network flows. In this step, the flows must be labeled properly. As mentioned before, we assume that at least 10The clustering phase has nothing to do with the data labels, so labeled data instances are probably distributed among the clusters. To label clusters, labeled instances in each are used.

Some clusters might not have any labeled instances, but if a cluster has some, the instances should be examined if they are determinative or not. The determinative instance is the one with maximum degree. In other words, the vertices with maximum number of neighbors are determinative ones. Determinative instances are also called dense instances. If there is a labeled dense instance in a cluster, itâĂŹs label is propagated to other instances of that cluster. Other-

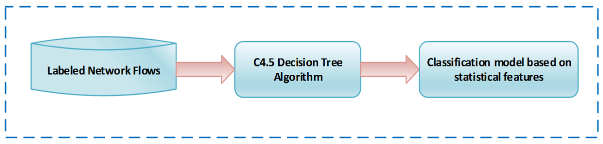
Labeled Network Flows
C4.5 Decision Tree Algorithm
Classification model based on statistical features

Figure 6. General structure of classification phase in proposed method.

wise, the cluster label could be obtained by asking experts. Then the label will be propagated to all cluster members.

3.1.3 Classification

In last steps, the training data instances were clustered and by label propagation, a set of labeled network flows was produced. These labeled instances can now be used for classification purposes. The goal of the classification phase is to build a classification model by statistical features of the dataset. For classification model, C4.5 supervised algorithm is used. As mentioned before, C4.5 algorithm showed proper performance in comparison with Bayesian network, naÃŕve bayes and SVMs. C4.5 algorithm generates a tree structure based on statistical features of the network flows. The structure can be used later in test phase to classify unseen instances. These instances are called test flows. The structure of the classification phase is shown in Figure 6.

3.2 Test Phase

The classification model built by C4.5 decision tree in now examines. Test data instances, are a subset of initial data set which doesn't contain training instances.

One method for evaluating classification algorithm is holdout. In this method, initial data set is divided into two disjoint data sets training and test. The division ratio depends on the view of analyst. It is common that 2/3 of the data set is used for training and the rest for evaluation. Holdout is simple and fast but has its deficiencies. The choice of training and test sets affects each other. Instances chosen for test have no chance to take part in training and vice versa. Random Sub-sampling is about running holdout repeatedly. The flaw here is that some instances might more frequently be chosen for training or test and others less. Cross Validation, which will be used in this article for evaluation purposes, controls this frequency and tends to maintain a balance.

4 Evaluating the Proposed Method

The proposed method is implemented using JAVA programming language on a machine with windows operating system. To evaluate it, datasets NLANR(AMP) [23], MAWI [24], DARPA99 [25] and Moore [26] was used. Although the size of SSH and NOTSSH types are imbalanced, SSH network flows are widely present in all these datasets. For the first 3 datasets, the binary classification problem SSH and NOTSSH was evaluated, but in Moore dataset there are various classes of traffic. So, classifying all types of network flows was evaluated using the proposed method.

To evaluate the proposed method, measures Recall or Detection Rate (DR), False Positive Rate (FPR), Precision and F-measure are used. FPR reveals the rate false prediction for the negative (NOTSSH) class. DR shows the percentage of correct predictions of all classes and precision states how accurate a classifier performs regarding a specific class. Finally, F-measure balances a tradeoff between Recall and Precision and can be calculated by the following equation:

$$F - Measure = \frac{(1 = \beta^2) \times Recall \times Precision}{\beta^2 \times (Recall + Precision)} \tag{13}$$

β in 4-1 is a tradeoff parameter between recall and precision which is normally set to be equal to 1.

Experiments and results

First 3 datasets are perused first. Each dataset contains statistical features of some network flows. SSH label corresponds to SSH network flows and NOTSSH is used for other classes of flows.

For cross validation purposes, 5 files each containing 4000 network flows are randomly sampled and prepared. Each file consists of 3000 NOTSSH instances and 1000 SSH flows. The following results are average values obtained for these prepared datasets.

Figure 7 illustrates detection rate of the proposed method for SSH and NOTSSH classes for various standard deviations which was used in the second phase of clustering. The range for standard deviation in these experiments was in $[0.02, 0.1]$. As can be seen, various detection rates obtained by varying standard deviation. Smaller values of standard deviation result in more accurate clusters, so it is reasonable for the method to achieve better results for such values. For MAWI dataset as can be seen in Figure 7, acceptable detection rates achieved for values than 0.07 of standard deviation, so 0.07 can be considered as threshold for this parameter.

Another effect of changing standard deviation is variation of the cluster numbers. Lesser values of standard deviation results in more clusters and greater values for it yields fewer clusters. Figure 8, shows this effect for data set MAWI. It is shown that the minimum number of clusters was obtained for value 0.1 of standard deviation and maximum number for value 0.02.

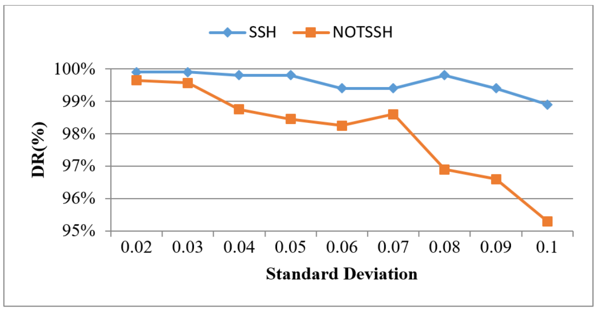

Figure 7. Detection rate of SSH and NOTSSH for various standard deviations in MAWI dataset

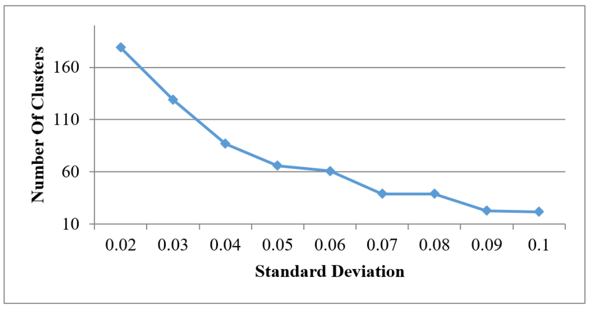

Figure 8. Different cluster numbers for different values of standard deviation in MAWI dataset

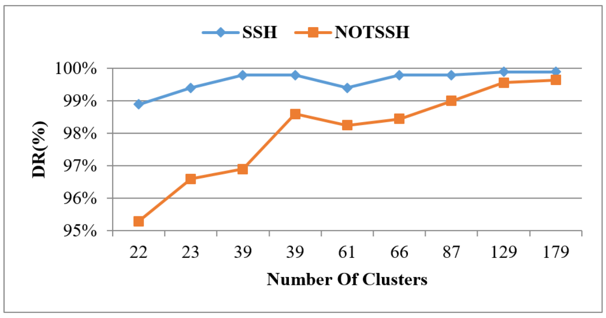

Figure 9. Detection rate of SSH and NOTSSH for various cluster numbers in MAWI dataset

Figure 9 shows trend for detection rate of SSH and NOTSSH while varying number of clusters. Values for standard deviation which result in fewer clusters and better separation for SSH and NOTSSH are suitable ones. Considering Figure 9, for 39 clusters or more, acceptable detection rate of SSH and NOTSSH is obtained. As mentioned before, the standard deviation for MAWI dataset varied in range of $[0.02, 0.07]$, so range of $[0.05, 0.07]$ that produces less clusters with acceptable results was the best range.

Figure 10, similarly shows detection rate of the proposed method for SSH and NOTSSH classes for various standard deviations in DARPA dataset. The figure shows that for higher standard deviation values, detection rate for NOTSSH increases and decreases for SSH. This shows that choosing smaller standard deviation values leads in misclassifying more

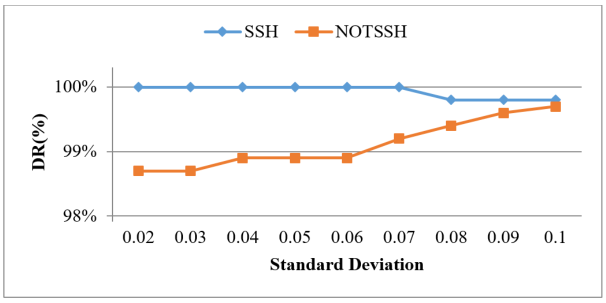

Figure 10. Detection rate of SSH and NOTSSH for various standard deviations in DARPA dataset

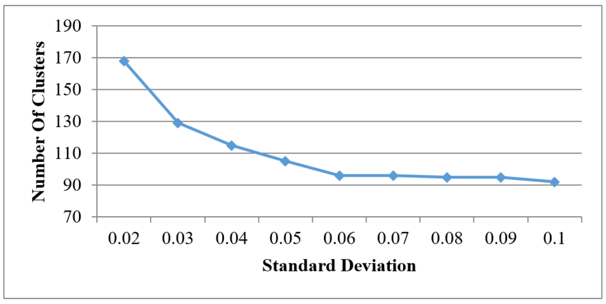

Figure 11. Different cluster numbers for different values of standard deviation in DARPA dataset

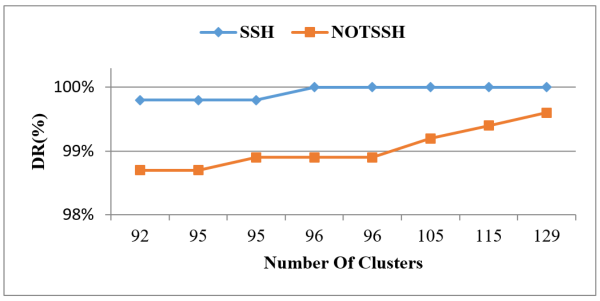

Figure 12. Detection rate of SSH and NOTSSH for various cluster numbers in DARPA dataset

of NOTSSH flows as SSH flows.

Also Figure 11, 12 show cluster numbers for different standard deviation values and corresponding detection rates respectively in DARPA dataset.

Like the results of MAWI dataset, according to results shown in Figure 10 to Figure 12, standard deviation values in range $[0.07, 0.09]$ produces acceptable results for lesser cluster numbers. The proposed method is evaluated using AMP dataset and the results are shown in Figure 13 to Figure 15.

According to Figure 13, in AMP dataset smaller values of standard deviation results in better detection rates. Also smaller values of standard deviation produce more cluster number. According to Figure 15, minimum number of clusters which produce better results are in range of 47 to 89.

According to Figure 14 and Figure 15, the best

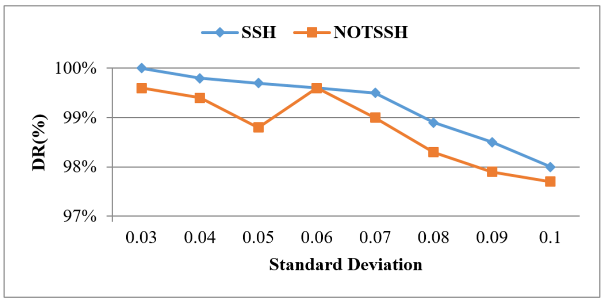

Figure 13. Detection rate of SSH and NOTSSH for various standard deviations in AMP dataset

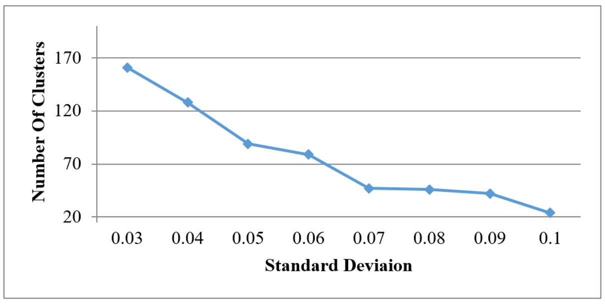

Figure 14. Different cluster numbers for different values of standard deviation in AMP dataset

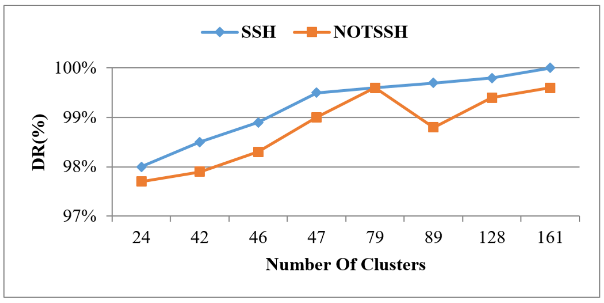

Figure 15. Detection rate of SSH and NOTSSH for various cluster numbers in AMP dataset

range of standard deviation for noted cluster number is $[0.05, 0.07]$. Considering all mentioned results, this could be stated that the range of $[0.05, 0.07]$ is suitable for standard deviation in all 3 datasets.

To better analyze the performance of the classifiers, specifically the binary ones, Receiver Operating Characteristics (ROC) curves are used. The observation mentioned before were all binary classification problems, so we analyze the ROC for them.

In this article, all datasets consist of two parts of labeled and un-labeled instances with the assumption that at least 10% of them are labeled.

After clustering part, as mentioned before, each cluster is checked to see if they contain any labeled dense instances. If there is one, its label is propagated to other instances of the cluster. If not, the cluster label is determined by asking experts. Initial assumption was that the dataset consists of at least 10% labeled data. After label propagation, this number arises. Considering above statement, the minimum and maximum number of labeled instance for datasets MAWI, DARPA and AMP are shown in Figure 17.

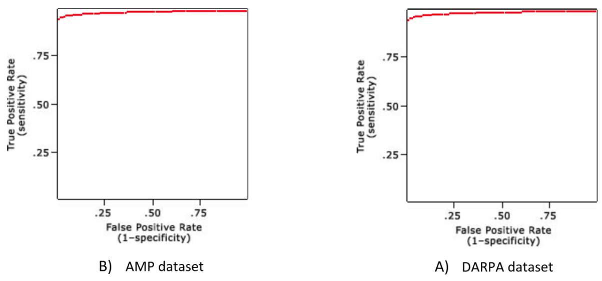

Figure 16. ROC for datasets DARPA and AMP

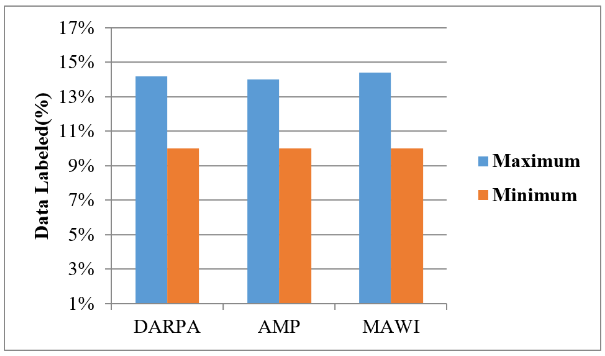

Figure 17. minimum and maximum amount of labeled data in datasets MAWI, AMP, DARPA

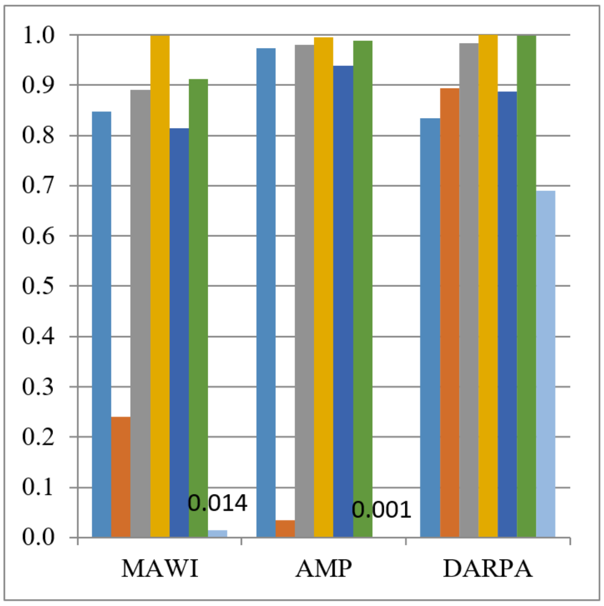

Figure 18. Comparing detection rate of SSH network flows

According to Figure 17, for the mentioned experiments, at most 14% data instances were labeled for classification step.

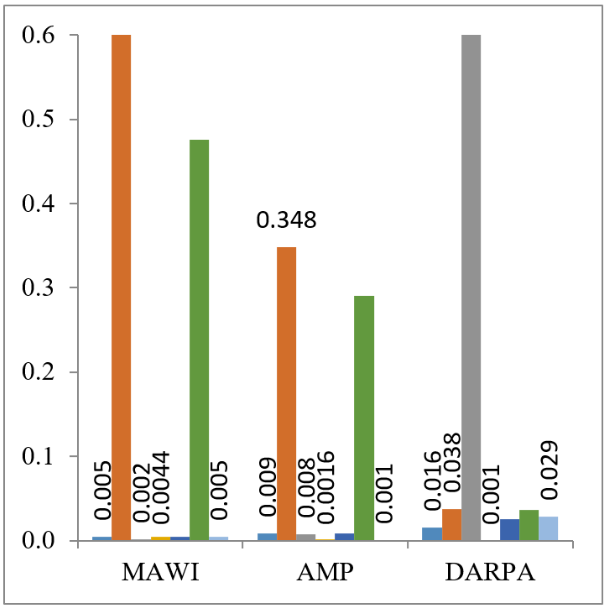

Figure 19. Comparing false positive rate for SSH

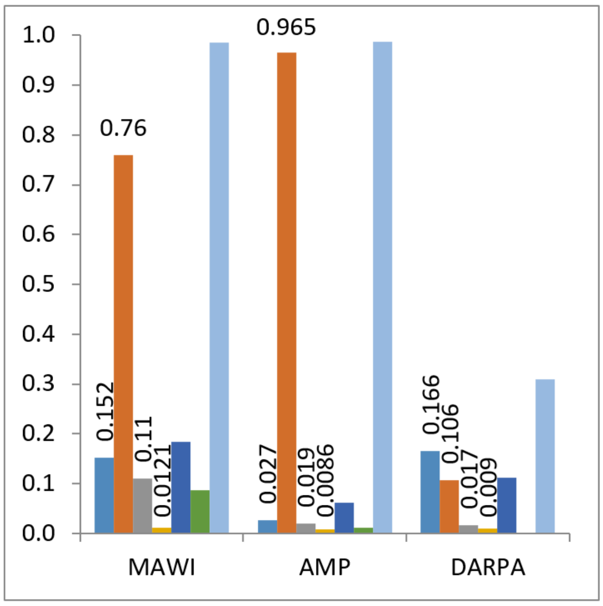

Figure 21. Comparing false positive rate for NOTSSH

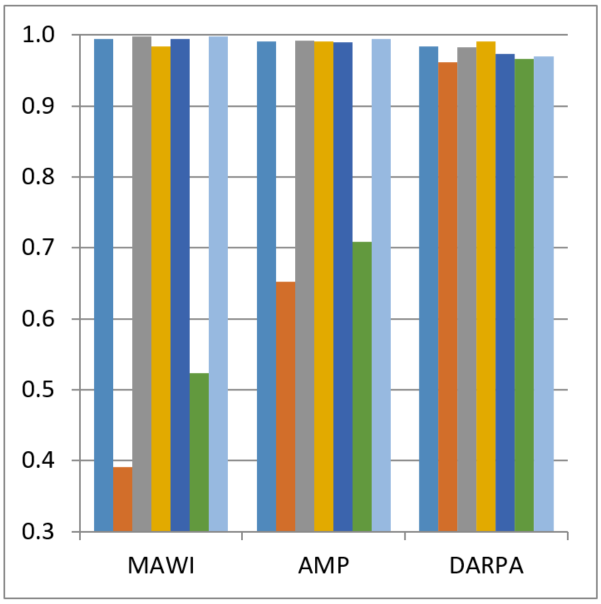

Figure 20. Comparing detection rate of NOTSSH network flows

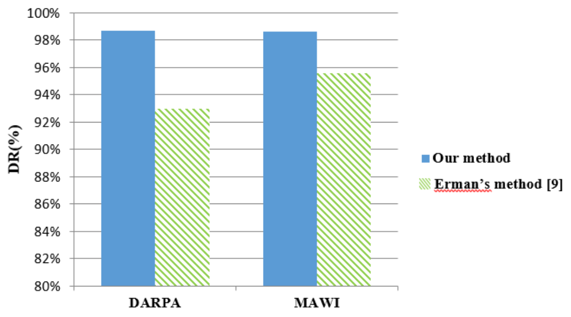

Figure 22. Comparing detection rate of the proposed method with a K-means based semi-supervised method [9]

To present a better view of the proposed method performance, in the following paragraphs, the proposed method is compared to some other well-known methods. Figure 18 and Figure 19 compare detection rate and false positive rate of the proposed method with 6 methods namely GP [6], C4.5, AdaBoost, RIPPER, SVM and NaÃŕve Bayesian for SSH network flows. Similarly, Figure 20 and Figure 21 compare detection rate and false positive rate of the proposed method with 6 methods namely GP [6], C4.5, AdaBoost, RIPPER, SVM and NaÃŕve Bayesian for NOTSSH network flows.

According to these comparisons, the proposed method reached better results considering both DR and FPR. Due to the fact that our method had only 10% of its data instances labeled clears better performance of this method. Considering the fact that the proposed method is a semi-supervised method, we compare its performance with another semi-supervised method [9]. Figure 22 compares detection rate of SSH network flows for datasets DARPA and MAWI.

As mentioned before, the proposed method consists of two clustering algorithms in the first part. In both levels the network flows are considered as nodes for the graph and distance between them construct the edges of the trees. Dissociation of the flows and formation of the clusters is done by eliminating the edges of the resulting trees. Elimination of the edges is inspired by the method proposed in [22]. In the following part, if needed the impure clusters of the first part, are decomposed into new smaller clusters using a new method. The evaluation results for the

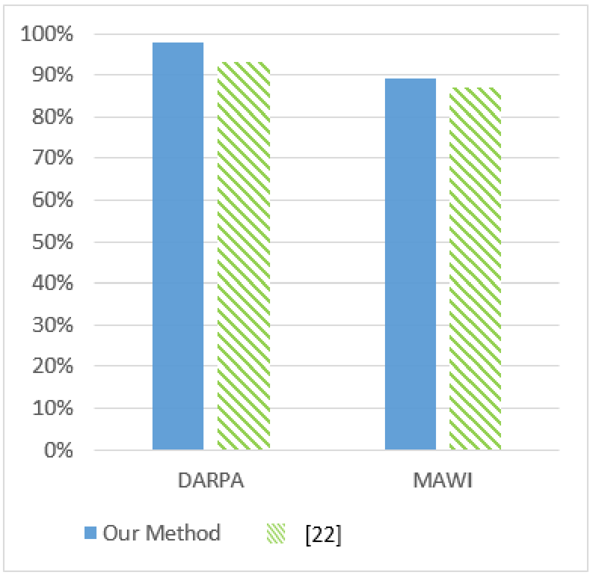

Figure 23. Comparing SSH detection rate of the proposed method with [22]

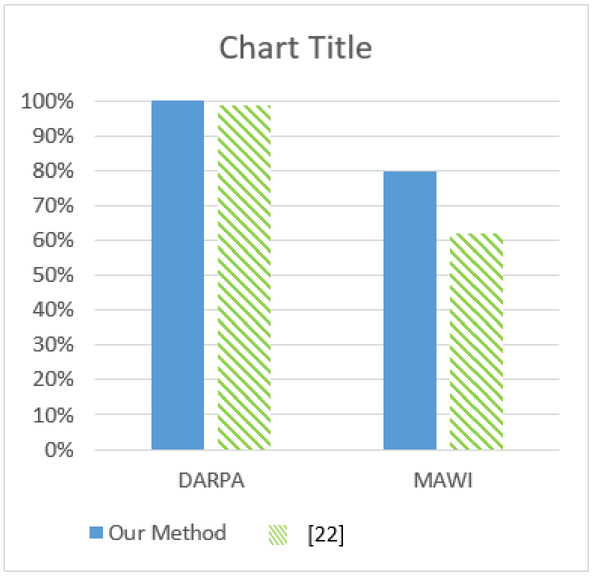

Figure 24. Comparing NOTSSH detection rate of the proposed method with [22]

proposed method and [22] are presented in Figure 23 for SSH network flows and in Figure 24 for NOTSSH network flows.

Another useful comparison is shown by Figure 25. It compares average F-measure for SSH class in MAWI dataset. It also shows the effect of majority of labeled samples. In addition to comparing with C4.5 and Erman's [9], Figure 25 shows slight superiority of the method compared to [11]. As can be seen, F-measure for the proposed method in noticeably higher than method of [11] for fewer labeled instances and like other methods increases with more instances.

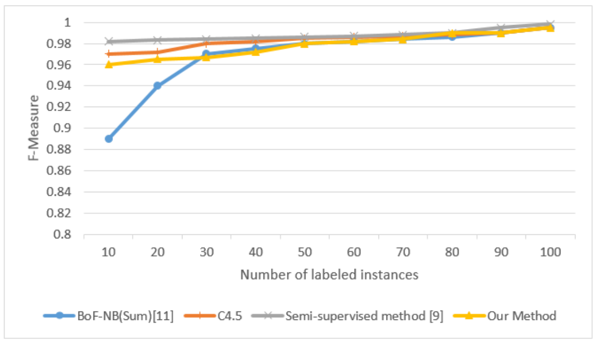

Figure 25. Average F-Measure of SSH class on MAWI dataset

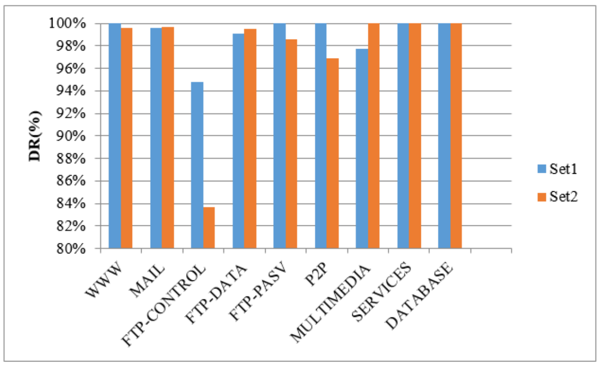

Figure 26. Detection rate of the method for classes of the Moore dataset on two disjoint random subsets of it

As mentioned before, the proposed method is applied to Moore dataset to classify different classes of network flows. In this regard, two distinct randomly generated subset of Moore dataset is prepared and the results of the proposed method for these subsets are shown in Figure 26.

The maximum number of generated clusters on first subset was 400. Like other experiments 10% of data instances were initially labeled and after label propagation, only 15.5% of instances had labels. For the second subset, the maximum number of clusters reached 295 and at most 15% of data instances were labeled for classification purposes. As shown in Figure 26, the proposed method reached suitable results by having less than 16% of its instances labeled.

Figure 27 compares detection rate of the proposed method with C4.5 and K-means algorithm. According to Figure 27, K-means reached detection rate 100% for 4 of the classes, but poor results for some other classes. This shows high FPR for those classes which shows poor performance of K-means.

In addition, more experiments have been carried out using other common dataset from UCI Repository [26] like Seeds, Breast Cancer, Wisconsin, Banknote and others. Datasets with various dimensions of features, classes and instance numbers form a few to hundreds of features and thousands of instances

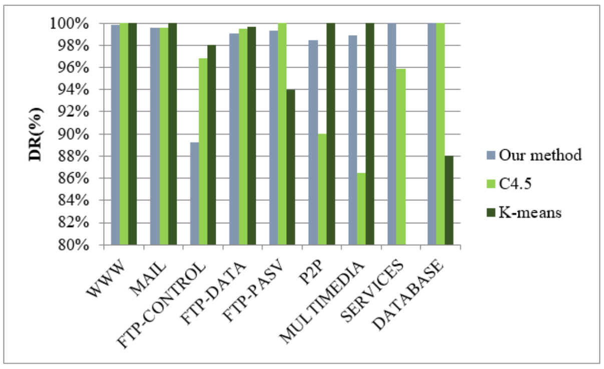

Figure 27. comparing performance of the proposed method with c4.5 and K-means algorithm

were used and the method obtained more or less the same results. The above experiments prove appropriate level of scalability.

5 Conclusion

Internet network traffic classification methods were discussed in this article. The main goal of the paper was to devise a novel method for detecting and classifying traffic of encrypted application such as SSH. The proposed method, which is a semi-supervised method based on graph theory, was implemented. The algorithm classifies both encrypted and plain network traffics by means of concepts of graph theory, minimum spanning trees and C4.5 decision trees. The method first clusters network flows based on their statistical features. According to the results, generated clusters of this method were perfect enough to cover most classes in related clusters. Having dense data instances, the label of each cluster could be obtained and propagated to other instances of the same cluster. The experimental results showed that the new method has accurate and suitable performance on classifying different network flows. Considering the main goal of the paper to classify encrypted network traffics, the experimental results confirm that the algorithm classifies the instances successfully.

References

[1] A. Madhukar and C. Williamson, "A longitudinal study of P2P traffic classification," in 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2006.

[2] A. Callado and *et al.*, "A survey on internet traffic identification," Communications Surveys & Tutorials IEEE, vol. 11, pp. 37-52, 2009.

[3] T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," Communications Surveys & Tutorials, IEEE, vol. 10, pp. 56-76, 2008.

[4] A. Dainotti, A. Pescape and K. C. Claffy, "Issues and future directions in traffic classification," IEEE Network, vol. 26, no. 1, pp. 35-40, 2012.

[5] A. W. Moore and D. Zuev, "Internet Traffic Classification Using Bayesian Analysis Techniques," SIGMETRICS Perform. Eval. Rev., vol. 33, no. 1, pp. 50-60, 2005.

[6] R. Alshammari and A. N. Zincir-Heywood, "Can encrypted traffic be identified without port numbers, IP addresses and payload inspection?," Computer networks, vol. 55, pp. 1326-1350, 2011.

[7] C. Zigang, C. Shoufeng, X. Gang and G. Li, "Progress in Study of Encrypted Traffic Classification," in Trustworthy Computing and Services: International Conference, Beijing, 2013.

[8] Z. Meng, H. Zhang, B. Zhang and G. Lu, "Encrypted Traffic Classification Based on an Improved Clustering Algorithm," in Trustworthy Computing and Services: International Conference, Beijing, 2012.

[9] J. Erman, A. Mahanti, M. Arlitt and L. Cohen, "Offline/realtime traffic classification using semi-supervised learning," Performance Evaluation, vol. 64, no. 9-12, p. 1194âĂŞ1213, 2007.

[10] "SSH," [Online]. Available: http://www.rfc-archive.org/getrfc.php?rfc=4251.

[11] C. Chao, J. Zhang, Y. Xiang, W. Zhou and Y. Xiang, "Internet traffic classification by aggregating correlated naive bayes predictions," IEEE Transactions on Information Forensics and Security, vol. 8, no. 1, pp. 5-15, 2013.

[12] N. Williams, S. Zander and G. Armitage, "A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification," SIGCOMM Comput. Commun. Rev., vol. 36, no. 5, pp. 5-16, 2006.

[13] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos and K. Lee, "Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices," in CoNEXT '08, New York, 2008.

[14] M. Lotfollahi, R. S. Hossein Zade, M. Jafari Siavoshani and M. Saberian, "Deep Packet: A Novel Approach For Encrypted Traffic Classification Using Deep Learning," eprint arXiv, vol. 1709.02656, no. 2, 2017.

[15] S. Bagui, X. Fang, K. Ezhil, S. C. Bagui and J. Sheehan, "Comparison of machine-learning algorithms for classification of VPN network using time-related features," Journal of Cyber Security Technology, vol. 1, no. 2, pp. 108-126, 2017.

[16] A. McGregor, M. Hall, P. Lorier and J. Brunskill, "Flow Clustering Using Machine Learning Techniques," in Passive and Active Network Measurement: 5th International Workshop, Berlin, Heidelberg, Springer Berlin Heidelberg, 2004, pp. 205-214.

[17] L. Bernaille, R. Teixeira and K. Salamatian,

"Early Application Identification," in Proceedings of the 2006 ACM CoNEXT Conference, New York, NY, USA, 2006.

[18] Z. Jun, X. Yang, Z. Wanlei and W. Yu, "Unsupervised traffic classification using flow statistical properties and IP packet payload," Journal of Computer and System Sciences, vol. 79, no. 5, pp. 573-585, 2013.

[19] A. Shrivastav and A. Tiwari, "Network traffic classification using semi-supervised approach," in Machine Learning and Computing (ICMLC), Bangalore, 2010.

[20] Y. Wang, Y. Xiang, J. Zhang and S. Yu, "A novel semi-supervised approach for network traffic clustering," in 5th International Conference on Network and System Security (NSS), Milan, 2011.

[21] C. T. Zahn, "Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters," IEEE Transactions on Computers, Vols. C-20, no. 1, pp. 68-86, 1971.

[22] C. Zhong and *et al.*, "A graph-theoretical clustering method based on two rounds of minimum spanning trees," Pattern Recognition, vol. 43, pp. 752-766, 2010.

[23] "NLANR," [Online]. Available: http://pma.nlanr.net.

[24] "MAWI," [Online]. Available: http://mawi.wide.ad.jp/mawi/.

[25] "DARPA 1999 intrusion detection evaluation data," [Online]. Available: https://www.ll.mit.edu/ideval/data/. "BRASIL," [Online]. Available:

[26] https://www.cl.cam.ac.uk/research/srg/netos/projects/brasil/.

Ehsan Mahdavi is a Ph.D. candidate with Isfahan University of technology. He received his B.S. and M.S. degrees in computer science from Shahid Beheshti University,Tehran, Iran. He then returned back to his hometown Isfahan and pursuit Ph.D. study in the field of network security with Isfahan University of Technology which is ongoing now.

Ali Fanian received the B.S., M.S. and Ph.D. degrees in computer engineering in 1999, 2001 and 2011, respectively from Isfahan University of Technology, Isfahan, Iran. He started his work in the same department as an assistant professor from that time. Different aspects of computer architecture and network security are his research interests; specially, adhoc networks, wireless network security and hardware design.

Homa Hasannejad received her B.S. in computer engineering from the department of electrical & computer engineering at Shiraz University, Iran. She received her M.S. degrees in artificial intelligence at the department of electrical & computer engineering at Isfahan University of Technology, Iran. Her research interests include the study of increasing the accuracy of Internet traffic classifications using machine learning methods.