iJRASET
International Journal For Research in
Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 3 Issue: VIII Month of publication: August 2015

DOI:

www.ijraset.com Volume 3 Issue VIII, August 2015
IC Value: 13.98 ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Self-Learning Network Traffic Classification

Vandana M[1], Sruthy Manmadhan[2]
[1,2]*Dept of Computer Science & Engineering,*
NSS College of Engineering, Palakkad, Kerala

Abstract*— Network management is part of traffic engineering and security. The current solutions - Deep Packet Inspection (DPI) and statistical classification rely on the availability of a training set. In case of these there is a cumbersome need to regularly update the signatures. Further their visibility is limited to classes the classifier has been trained for. Unsupervised algorithms have been envisioned as an alternative to automatically identify classes of traffic. To address these issues Unsupervised Self Learning Network Traffic Classification is proposed. It uses unsupervised algorithms along with an adaptive seeding approach to automatically let classes of traffic to emerge, making them identified and labelled. Unlike traditional classifiers, there is no need of a-priori knowledge of neither signatures nor a training set to extract the signatures. Instead, Unsupervised Self Learning Network Traffic Classification automatically groups flows into pure (or homogeneous) clusters using simple statistical features. This label assignment (which is still based on some manual intervention) ensures that class labels can be easily discovered. Furthermore, Unsupervised Self Learning Network Traffic Classification uses an iterative seeding approach which will boost its ability to cope with new protocols and applications. Unlike state-of-art classifiers, the biggest advantage of Unsupervised Self Learning Network Traffic Classification is its ability to discover new protocols and applications in an almost automated fashion.

***Keywords*— Traffic classification; clustering; self-seeding; unsupervised machine learning..**

I. INTRODUCTION

In a typical network, the traffic through the network is heterogeneous and consists of flows from multiple applications and utilities. Many of these applications are unique and have their own requirements with respect to network parameters such as delay, jitter, etc. Unless these requirements are met, the quality and usability of these applications will be severely compromised. While meeting these requirements in a Local Area Network (LAN) with its huge bandwidth might be easy, it usually is a challenge to meet them on the WANs, which have bandwidth constraints. Thus, traffic management on the WANs must exist in order to properly prioritize different applications across the limited WAN bandwidth and ensure that these requirements are met. In addition, a proper understanding of the applications and protocols in the network traffic is essential for any network manager to implement appropriate security policies. In a real network, user perception also matters. Although a user application might allow large delays or jitter, the user might be very sensitive to long wait times. Managing network traffic thus requires a judicious balance of all these priorities.

Classification of traffic is only the first step that helps identify different applications and protocols that exist in a network. Various actions, such as monitoring, discovery, control, and optimization can then be performed on the identified traffic with the end goal of improving the network performance. Typically, once the packets are classified (identified) as belonging to a particular application or protocol, they are marked or flagged. These markings or flags help the router determine appropriate service policies to be applied for those flows.

In other words:

Classification is a technique that identifies the application or protocol, and
Marking is the process that colors the packets (or just lets them through untouched) based on certain classification policies, which are used by the routers internally, or further downstream (depending on the kind of coloring) to provide appropriate treatment to those packets.

There are two other approaches to classifying traffic:

Classifying the packet based on the payload, that is, Payload-Based Classification. In this method, packets are classified based on the fields of the payload, such as Layer 4 ports (source or destination or both)
Classification based on a statistical method that uses statistical analysis of the traffic behaviour like inter-packet arrival, session time, and so on

The payload-based method is most prevalent. However, more often than not, it fails with encrypted and tunnelled traffic. The Payload-Based Classification technique can be divided into generic or basic payload analysis or advanced payload analysis. The generic approach to traffic classification is based on information in the IP header. Typically, the following information is looked at.

Layer 3 address (IP address)

Layer 2 address (MAC)

Protocols

This technique is very rudimentary and does not provide classification for most of the applications. A classification method based on the placement of traffic (ingress interface) also exists, but is not widely used. (Therefore, it will not be examined here) [3].

All generic classification techniques based on Destination IP address, Source IP address, or IP protocol, etc. are limited in their ability as the inspection is limited to the IP header only. Similarly, classifying based on Layer 4 ports only is also limited. The problem with this approach is that not all current applications use standard ports. Some applications even obfuscate themselves by using well the defined ports of other applications (e.g., IM applications may run over TCP port 80, which is generally used for HTTP). Hence, the Layer 4 port mechanism of application identification is not always reliable.

“Self Learning Network Traffic Classification” is a novel algorithm that overcomes the limitations highlighted above. The goal is to provide a deeper network visibility for operators. In other words, it is intended to offer the ability to semi-automatically identify prominent classes of traffic, targeting network management and traffic engineering operations1. “Self Learning Network Traffic Classification” proves to be able to expose classes of traffic which are very specific and possibly are not already known to the operator. For example, “Self Learning Network Traffic Classification” has been able to separate Google Mail traffic from other mail services. It thus automatically allows discovering new classes of traffic, allowing arbitrary definition of labels [5].

In “Self Learning Network Traffic Classification”, unsupervised data mining algorithms are used to automatically split traffic into homogeneous subsets or clusters. Flows are considered as the target of the classification. Each flow is characterized by using simple layer-4 metrics, like segment size and inter-arrival time. These features are known to carry valuable information about the protocol and/or application that generated the flow [2]. However, they perform not as good in the context of unsupervised (i.e. clustering) algorithms. Hence some ingenuity is adopted in order to improve cluster homogeneity. To overcome the limitation of off-the-shelf algorithms, an iterative clustering procedure is designed in which a filtering phase follows each clustering phase to eliminate possible outliers. Filtering is based on the still valuable information provided by port numbers. The port number information is not embedded in a metric space, e.g., the distance between port 79 and 80 is not different from the one between port 80 and 8080. As such, it is hard to integrate port number as a simple feature into classical clustering algorithms.

First, “Self Learning Network Traffic Classification” generated only a few cluster in each of these traces (typically less than 150). Second, clusters are very pure, i.e., the overall homogeneity of the clusters is close to 100%. This allows to easily inspect and label each cluster, thus assigning a proper label to all flows belonging to the same cluster. As soon as some labels are assigned to flows, it will automatically inherit them for classification of flows that arrive in the future. This can be referred to as adaptive or progressive seeding since flows labeled in the past are used to seed the subsequent datasets. Notably, this will minimize the bootstrapping effort required to label applications, and manual intervention is mainly required for the initial label assignment. This mechanism allows to naturally grow the intelligence of the system such that it is able to automatically adapt to the evolution of protocols and applications, as well as to discover new applications.

The idea of leveraging semi-supervised learning has been initially proposed in [4], where the authors leverage the standard k-means to construct clusters. Part of the flows to be clustered are assumed to be already labeled, and a simple voting scheme is used to extend the dominant label to the whole cluster.“Self Learning Network Traffic Classification” follows similar principles, extending the idea with

Iterative port filtering and

Multi-batch seeding,

which allow to significantly boost overall performance achieving 98% accuracy in practical cases. The iterative clustering algorithm and self-seeding approach provide several advantages: the number of clusters is reduced to less than 150, while at the same time homogeneity is significantly increased. This simplifies the labeling process so that manual inspection becomes almost trivial. Furthermore, the “Self Learning Network Traffic Classification” self-seeding process is more robust and results obtained from actual traffic traces show how the system helps in automatically identifying fine grained classes of traffic (e.g, IMAP vs POP3, XMPP vs Messenger), and even unveiling the presence of unknown/undesired classes (e.g., Apple push notification, Bot/Trojan, or Skype authentication traffic). In identifying standard protocols “Self Learning Network Traffic Classification” proves to be even more robust than professional DPI based tools which were fooled by non-English customizations of protocol error messages [2] [4].

II. RELATED WORKS

Real time traffic classification has the potential to solve difficult network management problems for Internet service providers (ISPs) and their equipment vendors. Network operators need to know what is flowing over their networks promptly so they can react quickly in support of their various business goals. Traffic classification may be a core part of automated intrusion detection systems [1] [2] [3], used to detect patterns indicative of denial of service attacks, trigger automated re-allocation of network resources for priority customers [4], or identify customer use of network resources that in some way contravenes the operator's terms of service. More recently, governments are also clarifying ISP obligations with respect to 'lawful interception' (LI) of IP data traffic [5]. Just as telephone companies must support interception of telephone usage, ISPs are increasingly subject to government requests for information on network use by particular individuals at particular points in time. IP traffic classification is an integral part of ISP-based LI solutions.

Commonly deployed IP traffic classification techniques have been based around direct inspection of each packet's contents at some point on the network. Successive IP packets having the same 5-tuple of protocol type, source address port and destination address:port are considered to belong to a flow whose controlling application to determine. Simple classification infers the controlling application's identity by assuming that most applications consistently use 'well known' TCP or UDP port numbers (visible in the TCP or UDP headers). However, many applications are increasingly using unpredictable (or at least obscure) port numbers [6].

A. Port Based Ip Traffic Classification

TCP and UDP provide for the multiplexing of multiple flows between common IP endpoints through the use of port numbers. Historically many applications utilise a 'well known' port on their local host as a rendezvous point to which other hosts may initiate communication. A classifier sitting in the middle of a network need only look for TCP SYN packets (the first step in TCP's three-way handshake during session establishment) to know the server side of a new client-server TCP connection. The application is then inferred by looking up the TCP SYN packet's target port number in the Internet Assigned Numbers Authority (IANA)'s list of registered ports [7]. UDP uses ports similarly (though without connection establishment nor the maintenance of connection state).

However, this approach has limitations. An application may use ports other than its well-known ports to avoid operating system access control restrictions (for example, non-privileged users on unix-like systems may be forced to run HTTP servers on ports other than port 80.) Also, in some cases server ports are dynamically allocated as needed. For example, the RealVideo streamer allows the dynamic negotiation of the server port used for the data transfer. This server port is negotiated on an initial TCP connection, which is established using the well-known RealVideo control port [9]. In some circumstances IP layer encryption may also obfuscate the TCP or UDP header, making it impossible to know the actual port numbers.

B. Payload Based IP Traffic Classification

Although the problem of determining the higher-layer protocol of a packet is commonly known as "traffic classification", in case of payload-based methods often it becomes a problem of "pattern verification". In the past, applications were detected by "classifying" the packet based on the value of a field (e.g. port 25 equal to SMTP traffic) [1]. These days, ports are meaningless on a vast portion of the traffic and the classification consists in guessing the application-layer protocol and verifying that this is correct by means of some appropriate controls (e.g. if the payload contains some well-known keywords). The difference between the payload-based technologies is due to the different types of controls that can be put in place, and the different processing methods that are required to perform these checks. The different complexity of these methods can be seen in the processing power that increases from left to right (e.g. the simplest method requires the parsing of headers, while the most complex one requires the processing of the application payload in all packets) and in the corresponding memory requirements. To avoid total reliance on the semantics of port numbers, many current industry products utilise stateful reconstruction of session and application information from each packet's content. Payload based classification of P2P traffic (by examining the signatures of the traffic at the application level) could reduce false positives and false negatives to 5% of total bytes for most P2P protocols studied. The classification procedure starts with the examination of a flow's port number. If no well-known port is used, the flow is passed through to the next stage. In the second stage, the first packet is examined to see whether it contains a known signature. If one is not found, then the packet is examined to see whether it contains a well-known protocol. If these tests fail, the protocol signatures in the first KByte of the flow are studied. Flows remaining unclassified after that stage will require inspection of the entire flow payload. Their results show that port information by itself is capable of correctly classifying 69% of the total bytes. Including the information observed in the first KByte of each flow increases the accuracy to almost 79%. Higher accuracy (upto nearly 100%) can only be achieved by investigating the remaining unclassified flows' entire payload. Although payload based inspection avoids reliance on fixed port numbers, it imposes significant complexity and processing load on the traffic identification device. It must be kept upto- date with extensive knowledge of application protocol semantics, and must be powerful enough to perform concurrent analysis of a potentially large number of flows. This approach can be difficult or impossible when dealing with proprietary protocols or encrypted traffic. Furthermore direct analysis of session and application layer content may represent a breach of organisational privacy policies or violation of relevant privacy legislation [3].

C. Hybrid Approaches

A semisupervised traffic classification approach which combines unsupervised and supervised methods. Motivations are due to two main reasons: Firstly labeled examples are scarce and difficult to obtain, while supervised learning methods do not generalise well when being trained with few examples in the dataset. Secondly, new applications may appear over time, and not all of them are known as a priori, traditional supervised methods map unseen flow instances into one of the known classes, without the ability to detect new types of flows [2].

To overcome the challenges, classification method consists of two steps. First, a training dataset consisting of labeled flows combined with unlabeled flows are fed into a clustering algorithm. Second, the available labeled flows are used to obtain a mapping from the clusters to the different known classes. This steps allows some clusters to be remained. To map a cluster with labeled flows back to an application type, a probabilistic assignment is used. Clusters without any labeled flows assigned to them are labeled 'Unknown' as application type. Finally a new unseen flow will be assigned to the nearest cluster with the distance metric chosen in the clustering step. This approach has promising results. Once the flows are clustered, a fixed number of random flows in each cluster are labeled. Results show that the approach results in 94% flow accuracy.

D. Deep Packet Inspection

Although most general applications can be determined or at least guessed based on L3 and L4 information, additional granular sub-classes within applications (like URLs) or specific kinds of messages within the application (like voice within IM streams) are required. For proper classification and sub-classification, it is necessary to do a deep packet inspection (DPI) and verify what the application is. Most DPI mechanisms use Signature Analysis to understand and verify different applications. Signatures are unique patterns that are associated with every application. In other words, each application is studied for its unique characteristics and a reference database is created. The classification engine then compares the traffic against this reference to identify the exact applications. That means the reference needs to be updated periodically to keep current with new applications as well as new developments in existing protocols.

III.PROPOSED SYSTEM

Data mining techniques may be grouped in two families: supervised and unsupervised techniques [6]. Supervised algorithms assume the availability of a training dataset in which each object is labeled, i.e., it is a-priori associated to a particular class. This information is used to create a suitable model describing groups of objects with the same label. Then, unlabeled objects can be classified, i.e., associated to a previously defined class, according to their features. For unsupervised algorithms, instead, grouping is performed without any a-priori knowledge of labels. Groups of objects are clustered based on a notion of distance evaluated among samples, so that objects with similar features are part of the same cluster.

Supervised algorithms achieve high classification accuracy, provided that the training set is representative of the objects. However, labeled data may be difficult, or time-consuming to obtain. Semi-supervised classification addresses this issue by exploiting the information available in unlabeled data to improve classifier performance. Many semi-supervised learning methods have been proposed [7], unfortunately, no single method fits all problems. Both labeled and unlabeled data are clustered by means of (variations of) known clustering algorithms (k-means in [8] and SOM in [9]). Next, labeled data in each cluster is exploited to assign labels to unlabeled data. Finally a new classifier is trained on the entire labeled dataset. While exploiting a different, iterative clustering approach to group data, our labeling process is similar to [8]. Due to its iterative refinement process, the approach adopted in "Self Learning Network Traffic Classification" is also particularly suited to model traffic flow changes, because it allows a seamless adaptation of the obtained traffic classes to traffic pattern evolution.

The application of unsupervised techniques is not new in the traffic classification field. [10] is one of the preliminary works and shows that clustering techniques are useful to obtain insights about the traffic. Supervised and unsupervised techniques are compared, demonstrating that unsupervised algorithms can achieve performance similar to the supervised algorithms. Other works compare the accuracy of different and standard unsupervised algorithms [12], In general, the techniques presented in these works achieve a moderate accuracy and they typically identify several hundreds of clusters, therefore questioning the applicability of this methodology in practice. Traffic is sniffed in real time and new flows enter the system continuously. Flows are processed in batches. A new batch B is formed as soon as a given number of valid flows is observed. The probe monitors packets and rebuilds flows. For a given flow, as soon as 6 data packets are observed the flow identifier and features are dispatched to a buffer where the batch is being formed.

```
 Main()
Output: set C of labeled clusters
S = Ø
while (newbatch B) do
ProcessBatch(B, U, S, C, NS)
S = NS
end while
 ProcessBatch(B, U, S, C, NS):
Input: Set B of new flows, set S of seeds
Output: set C of labeled clusters, set NS of new seeds
B' = B ∪S ∪U /* Merge new flow, seed set, past outliers */
C' = doIterativeClustering(B');
C = doLabeling(C');
NS = extractSeeds(C);
```

“Self Learning Network Traffic Classification” analyzes each batch of newly collected flows via the *ProcessBatch()* This function takes in input

B, the batch of new flows;
U, the set of previous outliers that were not assigned to any class
S, the set of seeding flows, i.e., flows already analysed in past batches
As output, it produces
C, the set of clusters;
NS, the set of new seeds that are extracted from each cluster;
U, which contains the set of new outliers
Its main steps are
Clustering batch data flows (*doIterativeClustering*()),
Flow label assignment (*doLabeling*()), and
Extraction of a new set of seeds (*extractSeeds*()).

A. The Filtering Procedure

Filtering is performed on the cluster I provided as input. First, *doFiltering*() discards clusters which have less than minPoints flows to avoid dealing with excessively small clusters. Discarded flows are returned in set U, the set of unclustered flows that will undergo a subsequent clustering phase Dominating Phase is a flag that is used to “Self Learning Network Traffic Classification “the type of filtering: when it is TRUE, the filtering processes only dominated Port clusters. To this aim, the srvPort distribution is checked. If the fraction of flows with the most frequent srvPort in I exceeds the threshold port Fraction, the cluster is a dominated Port cluster. The flows involving the dominant srvPort are clustered together and added to the set C of final clusters, while flows not involving the dominant srvPort are removed and put in U. The dominant port dp is included in the set DP of dominant ports . If there is no dominant port, all flows from I are put in U. When Dominating Phase is FALSE, random Port clusters are handled. In this case, cluster I (with all its flows) is simply added to the set of final clusters [3].

B. The Iterative Clustering Procedure

The iterative clustering procedure receives as input the current batch B of flows. It iteratively generates dominated port clusters alternating clustering and filtering phases. At last, it generates random port clusters. More specifically, the set of flows B to be clustered is processed for itermax iterations. At each iteration the set U of flows that are not yet assigned to any cluster is processed. k clusters are formed using the well-known k-means algorithm that returns the set C' of k clusters. Each cluster in C' undergoes a filtering phase , which is looking for dominated Port clusters at this stage. The *doFiltering()* procedure returns in U flows that did not pass the filter and must be processed at the next iteration. After itermax iterations, random Port clusters are handled. The clustering method used is a Hybrid of K means and KNN clustering algorithm.

1) *KNN Clustering Algorithm:* KNN are instance-based or lazy learners. It delays the process of modeling the training data until it is needed to classify the test samples. It can be used both for classification and prediction. The training samples are described by n-dimensional numeric attributes. The training samples are stored in an n-dimensional space. When a test sample (unknown class label) is given, the k-nearest neighbor classifier searches the k training samples which are closest to the unknown sample. Closeness is usually defined in terms of Euclidean distance.

```
k-nearest neighbor classification algorithm.
Let k be number of nearest neighbors and D be the set of training samples(yj).
for each test sample xi do compute d( xi , yj ) using Euclidean distance for every sample yj of D
Select the k closest training samples yj (neighbor's) to test sample xi
Classify the sample xi based on majority class among its nearest neighbors.
end for
```

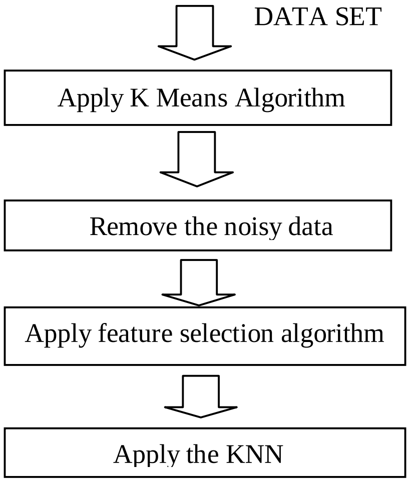
Some of the advantages of KNN are a)it is very to simple to implement and easy to justify the outcome of KNN. Although KNN has this advantages, it has some disadvantages such as: a) high Computation cost since it needs to compute distance of each test instance to all training samples b) requires large memory proportional to the size of training set c) Low accuracy rate in multidimensional data sets with irrelevant features d) there is no thumb rule to determine value of parameter K (number of nearest neighbors).

2) *K-Means Clustering Algorithm:* K-means is one of the simplest unsupervised learning algorithms and follows partitioning method for clustering. K-means algorithm takes the input parameter, k as number of clusters and partitions a dataset of n objects into k clusters, so that the resulting objects of one cluster are dissimilar to that of other cluster and similar to objects of the same cluster. In k-means algorithms begins with randomly selected k objects, representing the k initial cluster center or mean. Next each object is assigned to one the cluster based on the closeness of the object with cluster center. To assign the object to the closest center, a proximity measure namely Euclidean distance is used that quantifies the notion of closest. After all the objects are distributed to k clusters, the new k cluster centers are found by taking the mean of objects of k clusters respectively. The process is repeated till there is no change in k cluster centers.

```
K-means partitioning algorithm:
Input is k is the number of clusters, D is input data set Output is k clusters.
Randomly choose k objects from D as the initial cluster centers.
Repeat
Assign each object from D to one of k clusters to which the object is most similar based on the mean value of the objects in the cluster.
Update the cluster means by taking the mean value of the objects for each of k cluster.
Until no change in cluster means/ min error E is reached.
```

3) *Hybrid Algorithm*: In the first stage of proposed model, simple K-means clustering (with k = 2), is applied to 392 data samples. The wrongly classified samples are eliminated to get final 299 samples. The accuracy of the k-NN algorithm can be severely degraded by the presence of noisy or irrelevant features .In the second stage , relevant features are identified by cascaded GA_CFS. In supervised learning, feature selection aims to maximize classification accuracy. Mining on the reduced set of attributes has following benefits a)It reduces the number of attributes appearing in the discovered patterns, helping to make the patterns easier to understand b)It enhances the classification accuracy and c)It reduces classifier- learning time. GA is a stochastic general search method, capable of effectively exploring large search spaces, which is usually required in case of attribute selection. CFS evaluates the worth of a subset of attributes by considering the individual predictive ability of each

feature along with the degree of redundancy between them. Correlation coefficients is used to estimate correlation between subset of attributes and class, as well as inter-correlations between the features. Relevance of a group of features grows with the correlation between features and classes, and decreases with growing inter-correlation [12]. Authors have used GA_CFS to improve performance. Finally in the third stage, the correctly classified samples from first stage and the relevant features identified in the second stage were given as input to KNN classifier using 70-30 ratio partitioning method (training-test). Classification accuracy and Kappa statistics are used for performance evaluation of proposed classifier. When Kappa statistics K equals 1 , it means that there is complete agreement between the classifier output and the expected output (real world output). Kappa is always less than or equal to 1. A value less than 1 implies less than perfect agreement between the classifier output and the real world expected output.



Program diagram for hybrid Approach

C. Labelling

Once flows have been clustered, the *doLabeling(C')* procedure assigns a label to each cluster. For each cluster I in C' , flows are checked. If I contains some seeding flows, i.e., flows (extracted from S) that already have a label, a simple majority voting scheme is adopted: the seeding flow label with the largest frequency will be extended to all flows in I, possibly over-ruling a previous label for other seeding flow. More complicated voting schemes may be adopted (e.g., by requiring that the most frequent label wins by 50% or more). However, performance evaluation shows that the homogeneity of clusters produced by the iterative clustering procedure is so high that simple schemes work very nicely in practice .

Bootstrapping the labeling process

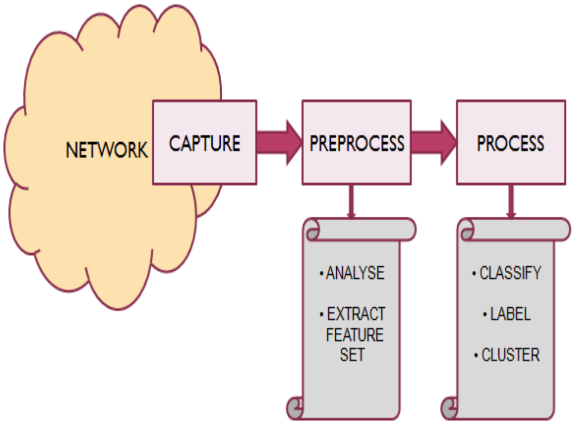
If no seeding flows are present, I is labeled as “unknown” and passed to the system administrator that should manually label the cluster. This will clearly happen during the bootstrapping of “Unsupervised Self Learning Network Traffic Classification”, when no labeled flows are present. To address this issue, several solutions can be envisioned. For example, labels can be manually assigned by using the domain knowledge of the system administrator, supported by all the available information on the flows in the cluster (e.g., port number, server IP addresses or even the flow payload, if available). A second option is to use a bootstrapping flow set from some active experiments in which traffic of a targeted application is generated. Similarly, a set of bootstrapping flows can be generated by providing labels obtained by some other available traffic classification tools.

D. Self-Seeding

Once some clusters have been labeled, “Self Learning Network Traffic Classification”is able to automatically reuse this information to process next batches. This is simply achieved by extracting some seeding flows from labeled clusters by means of the *extractSeeds(C)* procedure. It implements a stratified sampling technique, i.e., from each cluster, the number of extracted seeds is proportional to the cluster size. Stratified sampling ensures that at least one observation is picked from each of the cluster, even if probability of it being selected is far less than 1.

Thus, it guarantees that in the seeding set there are representatives of each cluster and avoids the bias due to classes having much more flows than others. Let numSeeds be the target number of seeding flows, i.e., numSeeds = ||NS||. For each labeled cluster I, a number NSI of labeled flows proportional to the cluster size is extracted at random. Flows are randomly selected from each cluster I. This mechanism enforces a self training process that allows the system to grow the set of labeled data and thus augment the coverage of the classification process.

E. Implementation



Architecture

Figure details about the system architecture. The entire system can be divided into mainly 3 modules.

1) *Capture:* The initial step in any the network traffic monitoring or analysing system will be to capture the data from the network (as shown in figure 4.2). Every network will be having lot of packets flowing through it including different protocols. The capturing is done using the JpCap library. JpCap *getDevicelist()* helps in monitoring, analyzing and listing out all the live interfaces in the network. The *openDevice()* helps in opening specified device interfaces and the packets from the particular device can be captured and processed using *processPacket().*
2) *Pre-Process:* This includes the process of analysing the data packets that have been captured . Also the the static feature set that have to be used for clustering purpose is selected. Based on these the feature set is being extracted.
3) *Processing:* Processing involves clustering, labelling and classification of clusters. Clustering algorithms group objects with similar characteristics [6]. Objects are described by means of features which map each object to a specific position in a hyperspace. The similarity between two objects is based on their distance. The closer the two objects are, the more likely they are similar and thus should to be grouped in the same cluster. Typically, the Euclidean distance is used. Iterative clustering is the core of system. It exploits the hybrid k-means clustering algorithm [6] to group flows into subsets or clusters which are possibly generated by the same applications. Hybrid k-means algorithm is selected since it is well understood. Once some clusters have been labeled, the system is able to automatically reuse this information to process next batches. This is simply achieved by extracting some seeding flows from labeled clusters by means of the *extractSeeds(C)* procedure. It implements a stratified sampling technique, i.e., from each cluster, the number of extracted seeds is proportional to the cluster size. Stratified sampling ensures that at least one observation is picked from each of the cluster, even if probability of it being selected is far less than 1. Thus, it guarantees that in the seeding set there are representatives of each cluster and avoids the bias due to classes having much more flows than others. Another important function is detecting the unreachable hosts flooding servers. One of the major problems faced by Internet hosts is denial of-service (DoS) caused by IP packet floods. Hosts in the Internet are unable to stop packets addressed to them. Once a host's network link becomes congested, IP routers respond to the overload by dropping packets arbitrarily. This is contrary to the goals of the host, which could respond more effectively to overload if it had control over which packets were dropped. For example, a host may reject new connections rather than accept excess load. A host running multiple services may give higher priority to some services than others (service differentiation). Also, a host may provide lower quality service rather than reject requests (service degradation). During the processing stage the system also identifies and lists out the hosts sending packets that are unreachable. Using this list further actions can be performed to block them and there by reducing ip flooding attacks.

IV. CONCLUSIONS AND FUTURE WORK

Unsupervised Self Learning Network Traffic Classification is a semi-automated Internet flow traffic classifier which leverages unsupervised clustering algorithms to automatically group's flows into clusters. Given that using unsupervised clustering algorithms does not result in high accuracy, that adding a filtering phase after clustering significantly improves the performance and coverage. Moreover, alternating the clustering and filtering phase's further results in very homogeneous clusters, while providing very high coverage. Labels for different clusters in "Self Learning Network Traffic Classification" can be bootstrapped using several different approaches (DPI, behavioral techniques, or human-in-the-middle). Once labels for some flows are provided, "Self Learning Network Traffic Classification" inherits previously labeled flows to automatically label new clusters. Furthermore, it adapts the model to traffic changes, and is able to automatically increase its knowledge. "Self Learning Network Traffic Classification" simplifies the manual bootstrapping of labels that, once provided to the system, lead to excellent performance [1]. At present traffic classification is widely concerned in many research fields such as network security, traffic scheduling and traffic accounting. How to identify network traffic fast and accurately is a very meaningful thing. But most machine learning based methods have a lower speed and efficiency, and can not guarantee their stability and usability. Principal Component Analysis (PCA) based method can be also used. At first the method use Fast Correlation-Based Filter (FCBF) algorithm to filter training data set to obtain suitable flow attributes. Then these flow attributes are processed by PCA to build feature subspace for each flow class. After that a nearest neighbour rule is used to accurately identify flow class of testing traffic sample. PCA based method has higher accuracy, stability and faster speed than Naive Bayes (NB) estimation method and Naive Bayes Kernel (NBK) [2] [14].

In the current Internet, hosts are practically defenceless against IP flooding attacks, as a sufficient amount of malicious traffic exhausts the last-hop link capacity and renders the link unusable. The reason flooding attacks can occur is that a host is unable to control which packets it receives. So if such packets from hosts could be identified priorly, then measures can be adopted to tame the ip flooding attack as well.

V. ACKNOWLEDGEMENT

The authors would like to thank professors of NSSCE, Palakkad for suggestions and support on this paper.

REFERENCES

[1] Grimaudo, Luigi, Marco Mellia, Elena Baralis, and Ram Keralapura. "SeLeCT: Self-Learning Classifier for Internet Traffic." 1-14..

[2] H. Kim, K. C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: myths, caveats, and the best practices," 2009 ACM CoNEXT.

[3] T. Karagiannis, D. Papagiannaki, and M. Faloutsos, "Blinc: multilevel traffic classification in the dark," 2005 ACM SIGCOMM.S.

[4] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, "Offline/ realtime traffic classification using semi-supervised learning," Perform. Eval., vol. 64, no. 9-12, pp. 1194–1213, Oct. 2007

[5] R. Dara, S. C. Kremer, and D. A. Stacey, "Clustering unlabeled data with SOMs improves classification of labeled real-world data," in Proc. 2002 IEEE IJCNN, pp. 2237–2242.

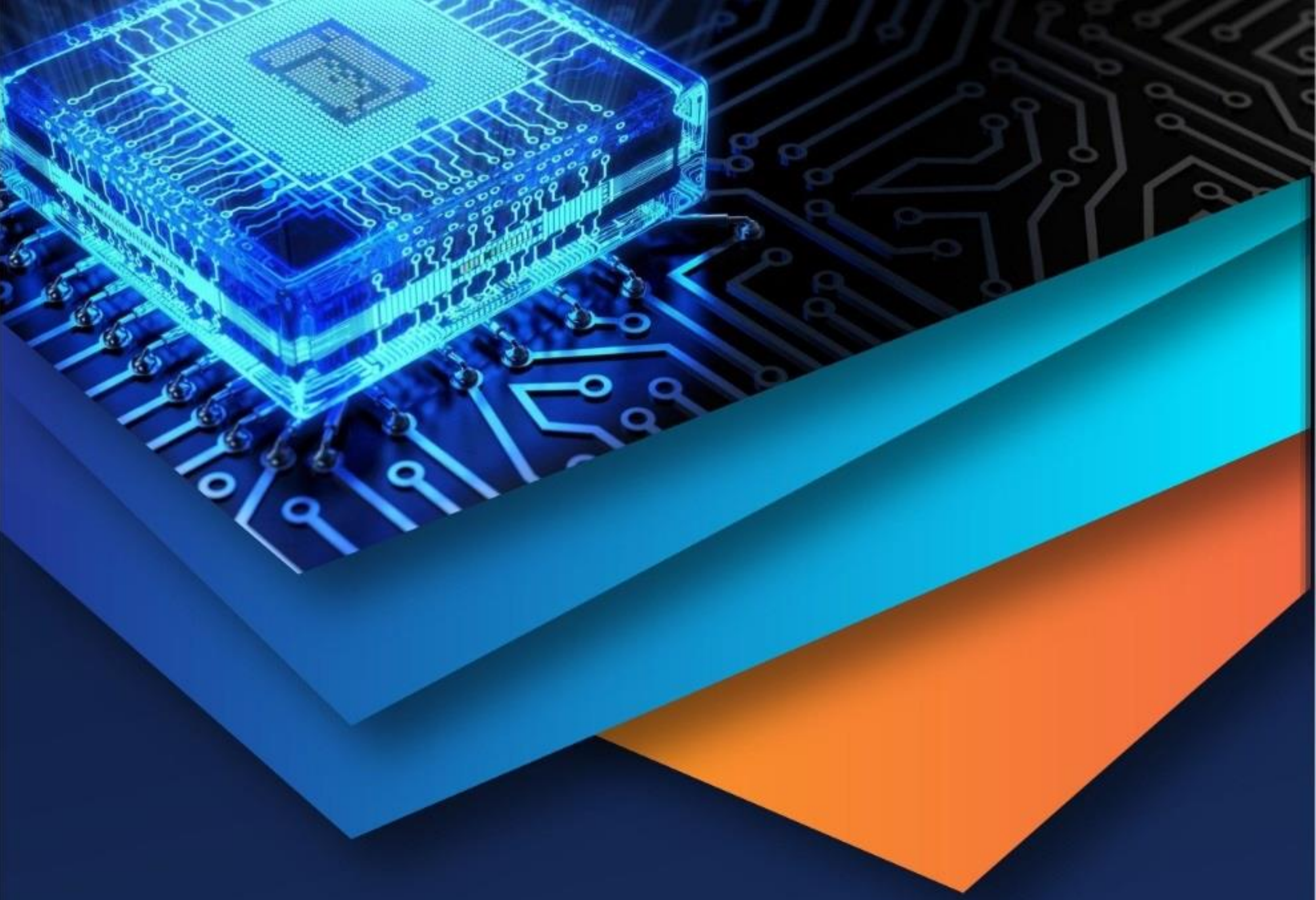
[6] T. Nguyen and G. Armitage, "A survey of techniques for Internet traffic classification using machine learning," IEEE Commun. Surveys and Tutorials, vol. 10, no. 4, 2008.

[7] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Traffic classification through simple statistical fingerprinting," SIGCOMM Comput. Commun. Rev., vol. 37, no. 1, pp. 5–16, 2007.

[8] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, "Flow clustering using machine learning techniques," in Proc. Passive and Active Measurement Workshop (PAM2004), Antibes Juan-les-Pins, France, April 2004.

[9] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service mapping for QoS: A statistical signature-based approach to IP traffic classification ," in Proc. ACM / SIGCOMM Internet Measurement Conference (IMC) 2004, Taormina, Sicily, Italy, October 2004

[10] S. Zander, T. Nguyen, and G. Armitage, "Automated traffic classification and application identification using machine learning," in IEEE 30th Conference on Local Computer Networks (LCN 2005), Sydney, Australia, November 2005.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY