iJRASET
International Journal For Research in
Applied Science and Engineering Technology

INTERNATIONAL JOURNAL
FOR RESEARCH
IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 Issue: VII Month of publication: July 2017

DOI:

Flow-Based Network Traffic Classification Using Clustering Technique with MLA Approach

Triveni Pujari[1], Nalinakshi B. G[2], Dr.D Jayaramaiah[3]
[1, 2,3]Dept. Of ISE, The Oxford College of Engineering, Bengaluru-68, India.

Abstract*: Network traffic classification is the process of categorizing network traffic according to various parameters into a number of traffic classes and it is necessary to maintain smooth operation of the network. There are so many methods to classify the network traffic. The proposed methods use machine learning algorithm i.e (MLA) approach. In MLA approach, a system learns from empirical data to automatically associate objects with corresponding classes. There are two types of MLAs one is supervised and the other is unsupervised. Supervised methods consist of labeled data to classify any flows into pre-defined traffic classes, but they cannot deal with unknown flows hence we use unsupervised clustering method along with supervised approach to cluster and classify both known and unknown flows. We applied hybridization of both supervised and unsupervised algorithm to achieve better accuracy. A number of real world traffic traces have been used to show the assessment of traffic classes and to test the proposed approach. The experimental results indicate that by incorporating special features of data packets in the course of clustering, enhances accuracy and cluster purity with significant improvement.

***Keyword* : Traffic classification, unsupervised machine learning, clustering, iterative approach, Wireshark, Tranalyzer**

I. INTRODUCTION

Traffic classification (i.e., associating traffic flows with their source applications) has attracted increasing research efforts in the last decade. The explosion of this research area started when the traditional approach of relying on transport-level protocol ports became unreliable, mainly because of the increasing variety and complexity of modern Internet traffic and application-level protocols. The reason for the growth in Internet traffic data is due to the bandwidth-hungry applications like File Transfer applications, Video Streaming, Social Media Network (Facebook, Twitter etc.), Mobile applications, E-commerce websites, Stock Exchange data and much more. As the traffic data increases it is necessary to analyse, measure, and classify it as ISP and Network Administrators need it for various perspectives like network planning, traffic shaping, billing and to extract useful information. This task needs to be performed with various tools available in market (Ex: tcpdump, Wireshark etc.)These tools capture the network traffic and store it onto a local server for further processing.

Network traffic classification is the process of classifying traffic based on their applications. Nowadays due to the growth of internet users and bandwidth hungry applications the traffic generated in the network is very high. There are many methods for classification of network traffic, they all try to classify the network traffic accurately but classification accuracy is less. From the very beginning of the internet, since there were not so many users and therefore, not so many applications, traffic classification was done using the well-known ports defined by IANA [16]. Classification based on well-known TCP or UDP ports is becoming increasingly less effective, due to the numbers of networked applications are port-agile (allocating dynamic ports as needed), end users are deliberately using non-standard ports to hide their traffic, and use of Network Address Port Translation (NAPT) is widespread (for example a large amount of peer-to-peer file sharing traffic is using non-default ports).

Payload-based classification relies on some knowledge about the payload formats for every application of interest: protocol decoding requires knowing and decoding the payload format while signature matching relies on knowledge of at least some characteristic patterns in the payload. This approach is limited by the fact that classification rules must be updated whenever an application implements even a trivial protocol change, and privacy laws and encryption can effectively make the payload inaccessible.

To overcome the above issues we are using machine learning approach to classify the traffic. In our work we are implementing an unsupervised clustering approach to classify the network traffic because unsupervised is that of trying to find hidden structure in unlabelled dataset. Clustering analysis is one of the unsupervised approaches and it is the process of making set objects in such a way that objects in the same group are more similar to each other than to those in other group. In our work we proposed automatic-learning algorithm using clustering techniques. Each flow indicates the packet size, packet length, inter-arrival time etc. So we can easily get the flow information for classification because these features are known to carry valuable information about the protocol and the applications that generated the flow.

II. LITERATURE SURVEY

The author in paper [1] proposes internet traffic classification using supervised learning algorithm and makes comparative analysis between machine learning algorithms. The main focus in this paper is the selection of feature sets. Hence it is concluded that the supervised decision tree based algorithms provide better performance and accuracy than the other supervised algorithms like KNN, naive Bayes etc. But accuracy of these decision tree based algorithms is poor while applying them for classifying P2P applications.

According to author in paper [2] Self-learning classifier is an unsupervised clustering algorithm with an adaptive seeding approach. It helps to automatically identify the classes of traffic being checked and labelled. This algorithm automatically groups flows into pure clusters using statistical features. Hence this paper acts as a base to our project as it summarises the state of the art of cluster analysis and here the main target of the classification is flows. Each flow is characterised by simple metrics, like segment size and inter arrival times.

According to Zhang, jun, et al. in paper [3] the classification of network traffic was done by correlation information. The traditional methods suffer from a number of practical problems, such as dynamic ports and encrypted applications so machine learning techniques have been focused to classify the traffic. Machine learning can automatically search for and describe useful structural patterns in a supplied traffic dataset. The correlation analysis can improve the classification accuracy and system flexibility. The proposed approaches can be used for recognising unknown application from captured network traffic and semi supervised data mining for processing network packets.

We selected paper [4] to understand the self-adaptive approach for network traffic classification. The author presented a novel, fully automated Packet Payload Content (PPC) Based network traffic classification system. System learns new application in the network where classification is desired. Hence the proposed algorithms are for distilling the generated signatures, and showed that these signatures are practical for real time classification in the real world.

According to author in paper [5], this paper can facilitate collaboration, convergence on standard definitions and procedures. The described Traffic Identification Engine (TIE), an open source tool for network traffic classification, can be applied to both live traffic and previously captured traffic traces. It is also investigated that the performance of multi classification systems when applied to early classification. TIE has the ability to configure from which portion of traffic the features passed to the classifier can be extracted.

The author in paper [7] proposes an unsupervised learning, which is traffic clustering for classification, where labelled training data is difficult and new patterns keep emerging. In order to improve the accuracy of traffic clustering, they constrained clustering schemes, which make decisions by considering some background information are proposed. They use Gaussian mixture density and adapt an appropriate algorithm for estimating the parameters.

III. SYSTEM DESIGN AND METHODOLOGY

The proposed system overcomes the limitation of iterative classifiers. The semi supervised classifiers are using the internet traffic and also overcome the internet bots. The iterative filtering and multi batch seeding is applied to improve the performance. We propose an unsupervised traffic classification that uses both flow features and packet payload. Using a bag-of-words approach and latent semantic analysis, some clusters are identified. Auto-Learning achieves better results in terms of classification performance, provides fine grained visibility on traffic, and offers a simple self-seeding mechanism that naturally allows the system to increase its knowledge. The proposed homogeneous clustering algorithm achieves much better classification performance than existing traffic classification like k-means methods. Homogeneous algorithm improves the overall system performance and resource efficient since traffic reduction is used. The system architecture is as shown in figure.1.

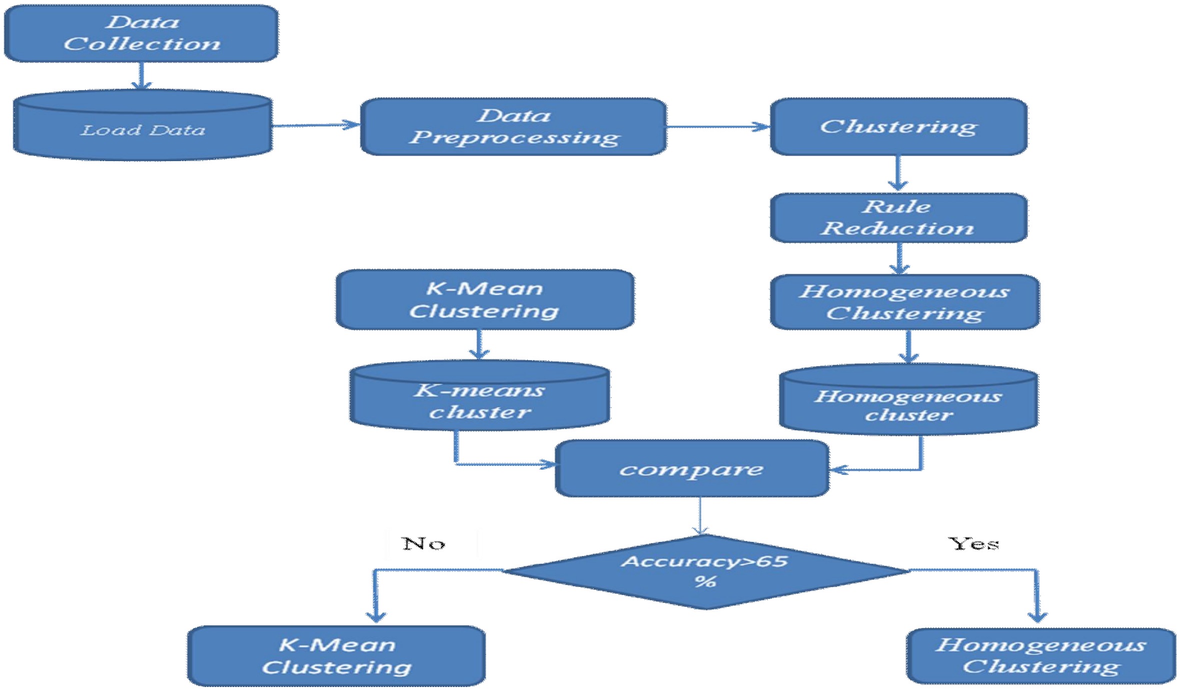


Fig.1 System Architecture

The functionalities of various software modules are explained below which includes four modules namely collection of dataset, data pre-processing, traffic reduction, and feature extraction.

A. Collect Dataset

Most commonly a data set corresponds to the contents of a single database table, or a single statistical data matrix, where every column of the table represents a particular variable, and each row corresponds to a given member of the data set. The data set lists values for each of the variables, such as height and weight of an object, for each member of the data set. Here the Wireshark tool is used to collect the dataset.

B. Data Pre-Processing

Data preparation and filtering steps can take considerable amount of processing time. It includes cleaning, normalization, transformation, feature extraction, and selection etc. Analysing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running an analysis.

C. Traffic Reduction

Traffic reduction method is one of the filtering method, it can reduce the data needed to be processed and hence increases the overall system performance. However, if a filter eliminates data improperly, bot detection rates could increase. Traffic Reduction can significantly improve the classification performance of many supervised classification algorithms.

D. Feature Extraction

Some of the behavior is distinguishable from normal behavior and hence features of the behavior can be extracted to detect bots. An ideal feature should be applicable to as many bots as possible. The features are collected in the feature extraction stage, like packet size, protocol, server port number, IP addresses etc and then the max membership principle is applied to the features to identify malicious ones. A packet is sent to the feature extraction stage if and only if its source or destination address is listed in the IP address list.

E. Clustering

Clustering is the process of grouping objects with similar features. In this paper the iterative clustering algorithm is used to classify the network traffic. In our work we are going to demonstrate how cluster analysis can be used to effectively identify groups of traffic. We are considering two unsupervised clustering algorithms, namely K-means and iterative homogeneous clustering for network traffic classification. We evaluate these two algorithms and compare them with the previously used auto class algorithm, using empirical internet traces. Number of parameters used is six and based on these parameters there are three clusters formed with four classes namely FTP, HTTP, Telnet and SMTP. The comparison analysis made between existing K-means algorithm and proposed cluster algorithm. The graph (fig.7) shows the efficiency of proposed algorithm is better than that of existing algorithm. Clustering Algorithm analyses each batch of newly collected flows via the ProcessBatch(). It uses doiterative() function to make pure clusters the below algorithm shows the main loop of proposed algorithm.

This function takes in input

_ *B*, the batch of new flows or dataset;

_ *U*, the set of previous outliers that were not assigned to any class when processing the previous batch;

_ *S*, the set of *seeding flows*, i.e., flows already analysed in past batches for which iterative clustering algorithm was able to provide a label;

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

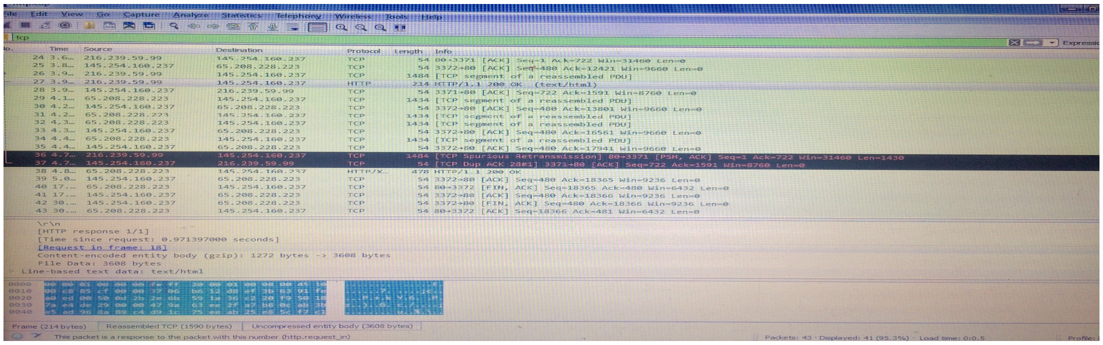
1) Clustering the dataset to get homogeneous subsets of flows,
2) Flow label assignment (function doLabeling()), and
3) Extraction of a new set of seeds (function extractSeeds()).Note that flows that are not assigned to any cluster are returned in the *U* set. Those flows are then aggregated in the next batch, so that they can eventually be aggregated to some cluster. In the following it details each step of the batch processing. Steps for malware detection or to detect the internet bots
4) Clustering the dataset to get homogeneous subsets of flows,
5) After homogeneous clustering rule reduction method is used
6) Extraction of a new set of rules (the source and destination address should be present in the IP address list)
7) Comparison between k-means and proposed homogeneous cluster method

IV. EXPERIMENTAL RESULTS

To carry out the experiment we have installed JDK 1.8 on our machine with net beans-IDE. It consists of packet sniffer program to capture and generate a Dataset. The implementation part consists of the following modules namely packet capturing, parameter selection, iterative clustering, labelling and classification.

A. *Packet Capturing*

This module is mainly used for capturing the packets to classify the traffic. It uses packet sniffer algorithm to capture the packet or packet capturing tools like Wireshark, netflow etc. Here Wireshark tool is used to capture the packet, it is one of the data capturing tools used to provide the structure of different networking protocols. It can also parse and display the fields, along with their meanings as specified by different networking protocols which are shown in the figure .2.

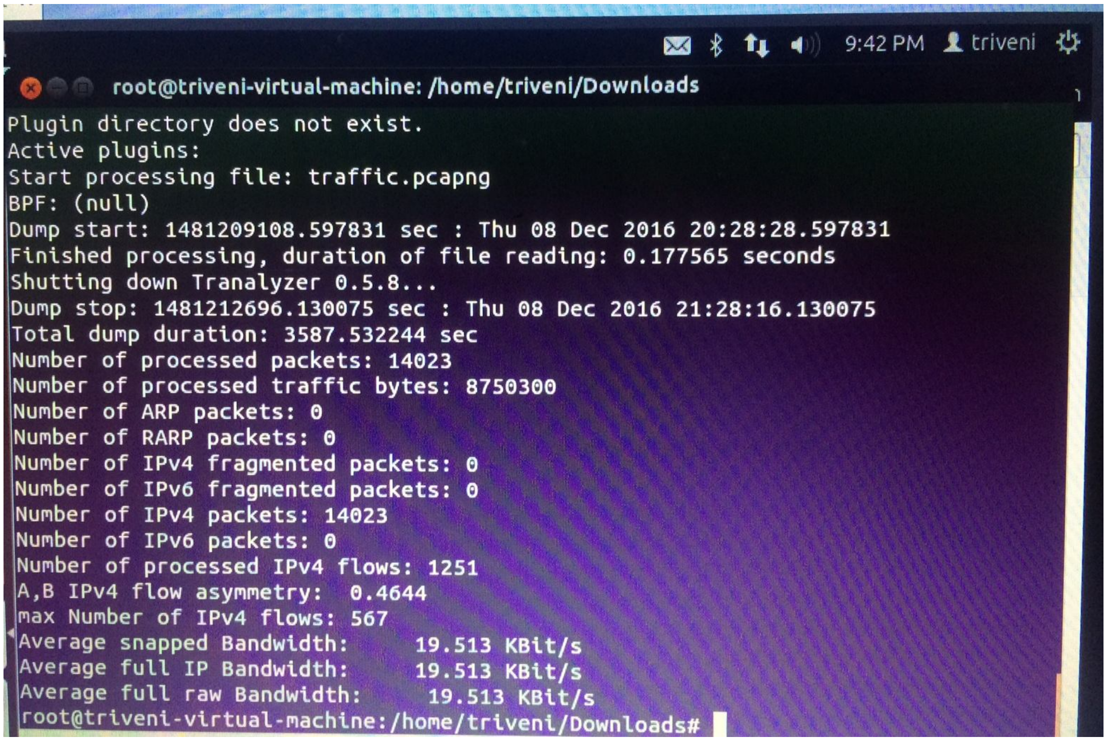


```
No.  Time    Source            Destination       Protocol  Length  Info
 24 3.6...  216.239.59.99     145.254.160.237   TCP           54  80→3371 [ACK] Seq=1 Ack=722 Win=31460 Len=0
 25 3.8...  145.254.160.237   65.208.228.223    TCP           54  3372→80 [ACK] Seq=480 Ack=12421 Win=9660 Len=0
 26 3.9...  216.239.59.99     145.254.160.237   TCP         1484  [TCP segment of a reassembled PDU]
 27 3.9...  216.239.59.99     145.254.160.237   HTTP         214  HTTP/1.1 200 OK  (text/html)
 28 3.9...  145.254.160.237   216.239.59.99     TCP           54  3371→80 [ACK] Seq=722 Ack=1591 Win=8760 Len=0
 29 4.1...  65.208.228.223    145.254.160.237   TCP         1434  [TCP segment of a reassembled PDU]
 30 4.2...  145.254.160.237   65.208.228.223    TCP           54  3372→80 [ACK] Seq=480 Ack=13801 Win=9660 Len=0
 31 4.2...  65.208.228.223    145.254.160.237   TCP         1434  [TCP segment of a reassembled PDU]
 32 4.3...  65.208.228.223    145.254.160.237   TCP         1434  [TCP segment of a reassembled PDU]
 33 4.3...  145.254.160.237   65.208.228.223    TCP           54  3372→80 [ACK] Seq=480 Ack=16561 Win=9660 Len=0
 34 4.4...  65.208.228.223    145.254.160.237   TCP         1434  [TCP segment of a reassembled PDU]
 35 4.4...  145.254.160.237   65.208.228.223    TCP           54  3372→80 [ACK] Seq=480 Ack=17941 Win=9660 Len=0
 36 4.7...  216.239.59.99     145.254.160.237   TCP         1484  [TCP Spurious Retransmission] 80→3371 [PSH, ACK] Seq=1 Ack=722 Win=31460 Len=1430
 37 4.7...  145.254.160.237   216.239.59.99     TCP           54  [TCP Dup ACK 28#1] 3371→80 [ACK] Seq=722 Ack=1591 Win=8760 Len=0
 38 4.8...  65.208.228.223    145.254.160.237   HTTP/X...    478  HTTP/1.1 200 OK
 39 5.0...  145.254.160.237   65.208.228.223    TCP           54  3372→80 [ACK] Seq=480 Ack=18365 Win=9236 Len=0
 40 17....  65.208.228.223    145.254.160.237   TCP           54  80→3372 [FIN, ACK] Seq=18365 Ack=480 Win=6432 Len=0
 41 17....  145.254.160.237   65.208.228.223    TCP           54  3372→80 [ACK] Seq=480 Ack=18366 Win=9236 Len=0
 42 30....  145.254.160.237   65.208.228.223    TCP           54  3372→80 [FIN, ACK] Seq=480 Ack=18366 Win=9236 Len=0
 43 30....  65.208.228.223    145.254.160.237   TCP           54  80→3372 [ACK] Seq=18366 Ack=481 Win=6432 Len=0

      \r\n
      [HTTP response 1/1]
      [Time since request: 0.971397000 seconds]
      [Request in frame: 18]
      Content-encoded entity body (gzip): 1272 bytes -> 3608 bytes
      File Data: 3608 bytes
  ▷ Line-based text data: text/html

0000  00 00 01 00 00 00 fe ff  20 00 01 00 08 00 45 10
0010  00 c8 85 cf 00 00 37 06  b6 12 d8 ef 3b 63 91 fe
0020  a0 ed 00 50 0d 2b 2e 6b  59 1a 36 c2 20 f9 50 18
0030  7a e4 de 29 00 00 47 9a  63 ee 2f a7 b8 0c ab 3b
0040  e5 ad 96 8a 89 c4 d9 1c  75 ee ab 25 e8 5c f7 c1

Frame (214 bytes) | Reassembled TCP (1590 bytes) | Uncompressed entity body (3608 bytes)
This packet is a response to the packet with this number (http.request_in)    Packets: 43 · Displayed: 41 (95.3%) · Load time: 0:0.5    Profile:
```

Fig.2. Wireshark Traffic Dump

1) *Tranalyzer2:* Tranalyzer2 is a lightweight flow generator and packet analyzer designed for simplicity, performance and scalability. The program is written in C and built upon the libpcap library. It provides functionality to pre- and post-process

IPv4/IPv6 data into flows and enables a trained user to see anomalies and network defects even in very large datasets, this is shown in figure.3.



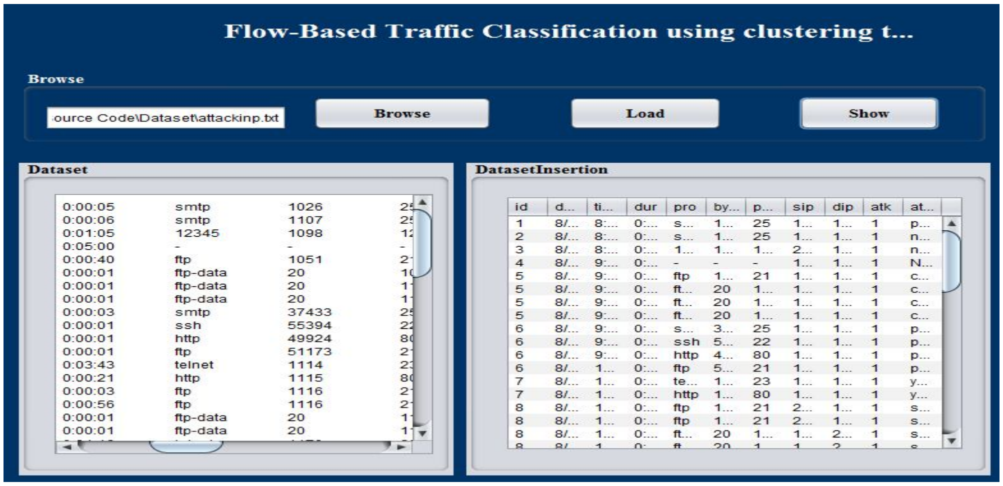
```
Plugin directory does not exist.
Active plugins:
Start processing file: traffic.pcapng
BPF: (null)
Dump start: 1481209108.597831 sec : Thu 08 Dec 2016 20:28:28.597831
Finished processing, duration of file reading: 0.177565 seconds
Shutting down Tranalyzer 0.5.8...
Dump stop: 1481212696.130075 sec : Thu 08 Dec 2016 21:28:16.130075
Total dump duration: 3587.532244 sec
Number of processed packets: 14023
Number of processed traffic bytes: 8750300
Number of ARP packets: 0
Number of RARP packets: 0
Number of IPv4 fragmented packets: 0
Number of IPv6 fragmented packets: 0
Number of IPv4 packets: 14023
Number of IPv6 packets: 0
Number of processed IPv4 flows: 1251
A,B IPv4 flow asymmetry:  0.4644
max Number of IPv4 flows: 567
Average snapped Bandwidth:     19.513 KBit/s
Average full IP Bandwidth:     19.513 KBit/s
Average full raw Bandwidth:     19.513 KBit/s
root@triveni-virtual-machine:/home/triveni/Downloads#
```

Fig.3. parameter selection using tranlyzer2

2) *Iterative Clustering:* It uses doiterative () function to make pure clusters the below algorithm shows how the iterative clustering will work.

3) *Labelling:* Once flows have been clustered, the doLabeling (C0) procedure assigns a label to each cluster. For each cluster I in C0, 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 flows. More complicated voting schemes may be adopted. The below fig.2 shows the accuracy of the proposed iterative algorithm is more than that of the existing algorithms.

The below snapshots show the results obtained in our work which includes data loading, data pre-processing, clustering and the comparison between existing and proposed system in the form of accuracy.



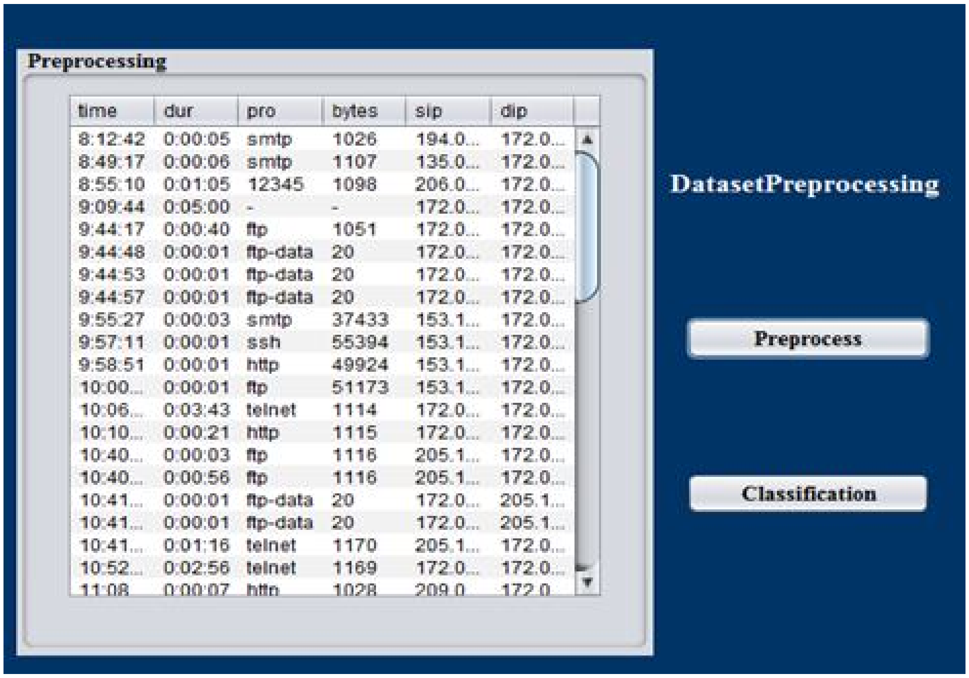
ource Code\Dataset\attackinp.txt

```
0:00:05     smtp        1026     25
0:00:06     smtp        1107     25
0:01:05     12345       1098     12
0:05:00     -           -        -
0:00:40     ftp         1051     21
0:00:01     ftp-data    20       10
0:00:01     ftp-data    20       1
0:00:01     ftp-data    20       1
0:00:03     smtp        37433    25
0:00:01     ssh         55394    22
0:00:01     http        49924    80
0:00:01     ftp         51173    21
0:03:43     telnet      1114     23
0:00:21     http        1115     80
0:00:03     ftp         1116     21
0:00:56     ftp         1116     21
0:00:01     ftp-data    20       1
0:00:01     ftp-data    20       1
```

id	d...	ti...	dur	pro	by...	p...	sip	dip	atk	at...
1	8/...	8:...	0:...	s...	1...	25	1...	1...	1	p...
2	8/...	8:...	0:...	s...	1...	25	1...	1...	1	n...
3	8/...	8:...	0:...	1...	1...	1...	2...	1...	1	n...
4	8/...	9:...	0:...	-	-	-	1...	1...	1	N...
5	8/...	9:...	0:...	ftp	1...	21	1...	1...	1	c...
5	8/...	9:...	0:...	ft...	20	1...	1...	1...	1	c...
5	8/...	9:...	0:...	ft...	20	1...	1...	1...	1	c...
5	8/...	9:...	0:...	ft...	20	1...	1...	1...	1	c...
6	8/...	9:...	0:...	s...	3...	25	1...	1...	1	p...
6	8/...	9:...	0:...	ssh	5...	22	1...	1...	1	p...
6	8/...	9:...	0:...	http	4...	80	1...	1...	1	p...
6	8/...	1...	0:...	ftp	5...	21	1...	1...	1	p...
7	8/...	1...	0:...	te...	1...	23	1...	1...	1	y...
7	8/...	1...	0:...	http	1...	80	1...	1...	1	y...
8	8/...	1...	0:...	ftp	1...	21	2...	1...	1	s...
8	8/...	1...	0:...	ftp	1...	21	2...	1...	1	s...
8	8/...	1...	0:...	ft...	20	1...	1...	2...	1	s...

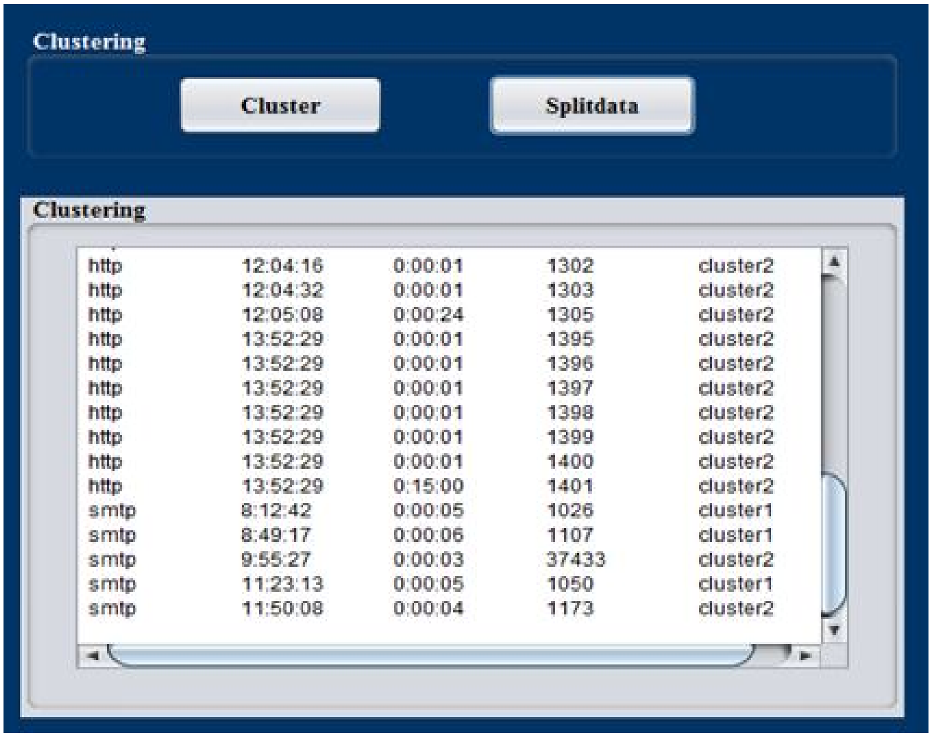
Fig.4 load data

First we need to load the dataset for data pre-processing and for feature extraction which is shown in the fig.2. After loading the data, the data pre-processing phase takes place which is shown in fig.3. It includes cleaning, normalization, transformation, feature extraction, and selection etc



time	dur	pro	bytes	sip	dip
8:12:42	0:00:05	smtp	1026	194.0...	172.0...
8:49:17	0:00:06	smtp	1107	135.0...	172.0...
8:55:10	0:01:05	12345	1098	206.0...	172.0...
9:09:44	0:05:00	-	-	172.0...	172.0...
9:44:17	0:00:40	ftp	1051	172.0...	172.0...
9:44:48	0:00:01	ftp-data	20	172.0...	172.0...
9:44:53	0:00:01	ftp-data	20	172.0...	172.0...
9:44:57	0:00:01	ftp-data	20	172.0...	172.0...
9:55:27	0:00:03	smtp	37433	153.1...	172.0...
9:57:11	0:00:01	ssh	55394	153.1...	172.0...
9:58:51	0:00:01	http	49924	153.1...	172.0...
10:00...	0:00:01	ftp	51173	153.1...	172.0...
10:06...	0:03:43	telnet	1114	172.0...	172.0...
10:10...	0:00:21	http	1115	172.0...	172.0...
10:40...	0:00:03	ftp	1116	205.1...	172.0...
10:40...	0:00:56	ftp	1116	205.1...	172.0...
10:41...	0:00:01	ftp-data	20	172.0...	205.1...
10:41...	0:00:01	ftp-data	20	172.0...	205.1...
10:41...	0:01:16	telnet	1170	205.1...	172.0...
10:52...	0:02:56	telnet	1169	172.0...	172.0...
11:08	0:00:07	http	1028	209.0	172.0

Fig.5 data pre-processing



http	12:04:16	0:00:01	1302	cluster2
http	12:04:32	0:00:01	1303	cluster2
http	12:05:08	0:00:24	1305	cluster2
http	13:52:29	0:00:01	1395	cluster2
http	13:52:29	0:00:01	1396	cluster2
http	13:52:29	0:00:01	1397	cluster2
http	13:52:29	0:00:01	1398	cluster2
http	13:52:29	0:00:01	1399	cluster2
http	13:52:29	0:00:01	1400	cluster2
http	13:52:29	0:15:00	1401	cluster2
smtp	8:12:42	0:00:05	1026	cluster1
smtp	8:49:17	0:00:06	1107	cluster1
smtp	9:55:27	0:00:03	37433	cluster2
smtp	11:23:13	0:00:05	1050	cluster1
smtp	11:50:08	0:00:04	1173	cluster2

Fig.6 clustering

Cluster process group objects with similar characteristics. Objects are described by means of selected features which are shown in fig.4. In fig.5 shows the comparison between the proposed clustering algorithm and the existing K-means algorithm hence it is concluded that our proposed algorithm is more effective than existing algorithm.

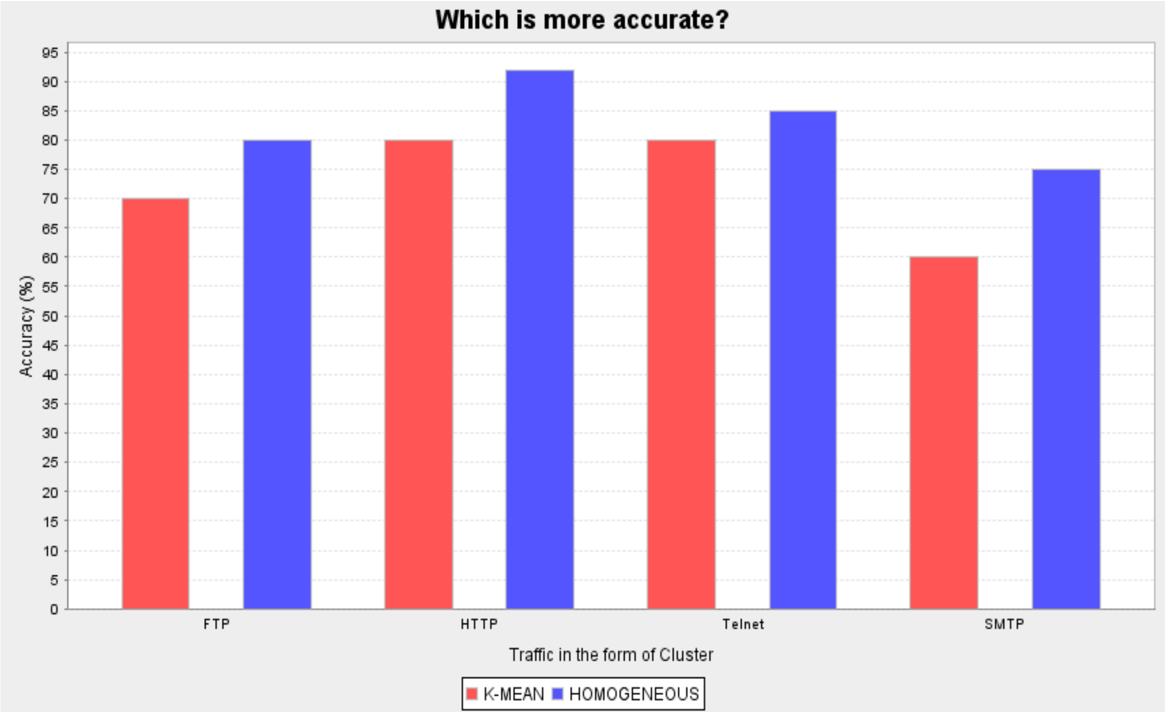


Fig.7 comparison between existing and proposed system

The accuracy graph shown in the figure .7 is the comparison between existing k-means algorithm and proposed clustering algorithm. Here X-axis indicates the generated clusters and Y-axis indicates the accuracy percentage. In this graph there are 4 clusters namely FTP, HTTP, Telnet and SMTP and it shows the percentage values.

V. CONCLUSION

The proposed homogeneous clustering MLA is used for distinguishing different kinds of traffic in a computer network. Here we are focusing on four different applications like FTP, HTTP, Telnet and SMTP. The homogeneous clustering method gives 90% accuracy than the existing k-means method in terms of purity or homogeneity of clusters and it also able to distinguish the traffic which appears to be similar, where an existing system cannot do.

VI. FUTURE SCOPE

The proposed algorithm which is considerably reduces the network traffic. Now the future work will be focusing on providing a good QOS, network security and minimize the network delay

REFERENCES

[1] Kalaiselvi, T., and P. Shanmugaraja. "Internet Traffic Classification Using supervised Learning Algorithms–A Survey." (2016).

[2] Grimaudo, Luigi, et al. "Select: Self-learning classifier for internet traffic." IEEE Transactions on Network and Service Management 11.2 (2014): 144-157.

[3] Zhang, Jun, et al. "Network traffic classification using correlation information." IEEE Transactions on Parallel and Distributed Systems 24.1 (2013): 104-117.

[4] Tongaonkar, Alok, et al. "Towards self adaptive network traffic classification." Computer Communications 56 (2015): 35-46.

[5] De Donato, Walter, Antonio Pescapé, and Alberto Dainotti. "Traffic identification engine: an open platform for traffic classification." IEEE Network 28.2 (2014): 56-64.

[6] Dainotti, Alberto, Antonio Pescape, and Kimberly C. Claffy. "Issues and future directions in traffic classification." IEEE network 26.1 (2012).

[7] Wang, Yu, et al. "Internet traffic classification using constrained clustering." IEEE Transactions on Parallel and Distributed Systems 25.11 (2014): 2932-2943.

[8] Kim, Jeankyung, Jinsoo Hwang, and Kichang Kim. "High-Performance Internet Traffic Classification Using a Markov Model and Kullback-Leibler Divergence." Mobile Information Systems 2016 (2016).

[9] Zhang, Jun, et al. "Network traffic classification using correlation information." IEEE Transactions on Parallel and Distributed Systems 24.1 (2013): 104-117.

[10] Shafiq, Muhammad, et al. "Network Traffic Classification techniques and comparative analysis using Machine Learning algorithms." Computer and Communications (ICCC), 2016 2nd IEEE International Conference on. IEEE, 2016.

[11] Alejandre, Francisco Villegas, Nareli Cruz Cortés, and Eleazar Aguirre Anaya. "Botnet Detection using Clustering Algorithms." Research in Computing Science 118 (2016): 65-75.

[12] Hosseinpour, Farhoud, et al. "Artificial immune system based intrusion detection: Innate immunity using an unsupervised learning approach." International Journal of Digital Content Technology and its Applications 8.5 (2014): 1.

[13] Katal, Supriya, and Asstt Prof Hardeep Singh. "A Survey of Machine Learning Algorithm in Network Traffic Classification." International Journal of Computer Trends and Technology (IJCTT).–Madurai, India: Seventh Sense Research Group 9.6 (2014): 301-304.
[14] Finsterbusch, Michael, et al. "A survey of payload-based traffic classification approaches." IEEE Communications Surveys & Tutorials 16.2 (2014): 1135-1156.
[15] Ranjan, Supranamaya, Joshua Robinson, and Feilong Chen. "Machine learning based botnet detection using real-time connectivity graph based traffic features." U.S. Patent No. 8,762,298. 24 Jun. 2014.
[16] Narten, Thomas. "Guidelines for writing an IANA Considerations Section in RFCs." (2008).