GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Congestion Control Using Network Based Protocol

A Report for the Evaluation of Project

Submitted by

Rohit kumar

(1613101590)

In partial fulfilment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of

Prof. Anvesha Katti

April/May-2020

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report "**CONGESTION CONTROL USING NETWORK BASED PROTOCL** " is the bonafide work of "**ROHIT KUMAR (1613101590)**" who carried out the project work under my supervision.

SIGNATURE OF HEAD

School of Computing Science & Engineering

SIGNATURE OF SUPERVISOR

Prof. Anvesha Katti

Professor

School of Computing Science & Engineering

TABLE OF CONTENTS:

	Page No.
1. Abstract	1
2. Introduction	2
(i) Overall description	6
(ii) Purpose	11
(iii) Motivation and scope	14
3. Literature survey	15
4. Purposed model	20
5. Result and Discussion	29
6.Conclusion and Future Works	34
7. References	35

ABSTRACT

Network traffic characterization is nothing but a methodology to classify network traffic into a no of classes agreeing distinctive highlights or attributes. In this paper a payload based classification is utilized for network traffic characterization methods. It has a wide scope of utilization like network security examination, intrusion identification, QoS supplier et cetera; and furthermore has significance in investigation of different suspicious network movement. Already numerous supervised classifications like Support Vector Machines and unsupervised bunching calculation like K-Means connected in method of Network Congestion Control. In modern network condition, the performance of customary classification procedure influenced because of minimal supervised data, and unfamiliar applications. In this paper we proposed an approach for network Network Congestion Control by consolidating clustering, feature extraction and classification systems. An unsupervised grouping calculation, K-Means is used for clustering the network traffic dataset and on grouped information the feature extraction system is connected. For classification of network traffic in view of extracted features we used K-means, Naïve Bayes classification calculation. This thesis presents a performance measurement between these classification algorithms.

INTRODUCTION

Network Congestion Control is the way toward distinguishing network applications and characterizing the relating traffic, which is thought to be the most essential usefulness in current network administration and security frameworks. Or on the other hand Traffic grouping is a programmed technique which classifies network traffic as per different requirements into various traffic. Application related Network Congestion Control is essential innovation for on-going network security. The traffic order can be utilized to discover the worm engendering, intrusion discovery, and examples characteristic of DOS assaults, spam spread, suspicious activity around network, QOS supplier et cetera. Network Congestion Control strategies incorporate payload-based profound classification. Payload based classification also known as Deep Packet Inspection (DPI) technique and provides accurate classification result in Network Congestion Control process. The Internet continually evolves in scope and complexity, much faster than our ability to characterize, understand, control, or predict it. The field of Internet Network Congestion Control research includes many papers representing various attempts to classify whatever traffic samples a given researcher has access to, with no systematic integration of results. Here we provide a rough taxonomy of papers, and explain some issues and challenges in Network Congestion Control. The flow feature-based Network Congestion Control can be achieved by using supervised classification algorithms or unsupervised classification (clustering) algorithms. In unsupervised Network Congestion Control, it is very difficult to construct an application oriented traffic classifier by using the clustering results without knowing the real traffic classes. In the last decade, considerable research works were reported on the application of machine learning techniques to Network Congestion Control. These works can be categorized as supervised methods or unsupervised methods.

Supervised Methods:

The supervised Network Congestion Control methods analyze the supervised training data and produce an inferred function which can predict the output class for any testing flow. In supervised Network Congestion Control, sufficient supervised training data is a general assumption. To

address the problems suffered by payload-based Network Congestion Control, such as encrypted applications and user data privacy, Moore and Zuev applied the supervised naive Bayes techniques to classify network traffic based on flow statistical features. Williams et al. evaluated the supervised algorithms including naive Bayes with discretization, naive Bayes with kernel density estimation, C4.5 decision tree, Bayesian network, and naive Bayes tree. Nguyen and Armitage proposed to conduct Network Congestion Control based on the recent packets of a flow for real-time purpose. Auld et al. extended the work of with the application of Bayesian neural networks for accurate Network Congestion Control. Erman et al. used unidirectional statistical features for Network Congestion Control in the network core and proposed analgorithm with the capability of estimating the missing features. Bernaille and Teixeira proposed to use only the size of the first packets of an SSL connection to recognize the encrypted applications. Bonfiglio et al. proposed to analyze the message content randomness introduced by the encryption processing using Pearson's chi-Square test-based technique. Crotti et al. proposed the probability density function (PDF)-based protocol fingerprints to express three traffic statistical properties in a compact way. Their work is extended with a parametric optimization procedure . Este et al. applied oneclass SVMs to Network Congestion Control and presented a simple optimization algorithm for each set of SVM working parameters. Valenti et al. proposed to classify P2P-TV traffic using the count of packets exchanged with other peers during the small time windows. Pietrzyk et al. evaluated three supervised methods for an ADSL provider managing many points of presence, the results of which are comparable to deep inspection solutions. These works use parametric machine learning algorithms, which require an intensive training procedure for the classifier parameters and need the retraining for new discovered applications. There are a few works using nonparametric machine learning algorithms. Roughan et al.have tested NN and LDA methods for Network Congestion Control using five categories of statistical features. Kim et al. extensively evaluated ports-based Corel Reef method, host behaviour-based BLINC method and seven common statistical feature-based methods using supervised algorithms on seven different traffic traces. The performance of the NN-based traffic classifier is comparable to two outstanding parametric classifiers, SVM and neural nets. Although nonparametric methods have several important advantages which are not shared by parametric methods, their capabilities have been considered undervalued in the area of Network Congestion Control. Besides, supervised learning has also been applied to payload-based Network Congestion Control. Although Network Congestion

Control by searching application signatures in payload content is more accurate, deriving the signatures manually is very time consuming. To address this problem, Haffner et al. proposed to apply the supervised learning algorithms to automatically identify signatures for a range of applications. Finamore et al. proposed application signatures using statistical characterization of payload and applied supervised algorithms, such as SVM, to conduct Network Congestion Control. Similar to the supervised methods based on flow statistical features, these payload-based methods require sufficient supervised training data.

Unsupervised Methods:

The unsupervised methods (or clustering) try to find cluster structure in unlabeled traffic data and assign any testing flow to the application-based class of its nearest cluster. McGregor et al. proposed to group traffic flows into a small number of clusters using the expectation maximization (EM) algorithm and manually label each cluster to an application. Zander et al.used Auto Class to group traffic flows and proposed a metric called intra-class homogeneity for cluster evaluation. Bernaille et al. applied the k-means algorithm to traffic clustering and labeled the clusters to applications using a payload analysis tool. Erman et al. evaluated the k-means, DBSCAN and AutoClass algorithms for traffic clustering on two empirical data traces. The empirical research showed that traffic clustering can produce high-purity clusters when the number of clusters is set as much larger than the number of real applications. Generally, the clustering techniques can be used to discover traffic from previously unknown applications . Wang et al. proposed to integrate statistical feature-based flow clustering with payload signature matching method, so as to eliminate the requirement of supervised training data. Finamore et al. combined flow statistical feature-based clustering and payload statistical feature-based clustering for mining unidentified traffic. However, the clustering methods suffer from a problem of mapping from a large number of clusters to real applications. This problem is very difficult to solve without knowing any information about real applications.Erman et al. proposed to use a set of supervised training data in an unsupervised approach to address the problem of mapping from flow clusters to real applications. However, the mapping method will produce a large proportion of “unknown” clusters, especially when the supervised training data is very small. In this paper, we study the problem of supervised Network Congestion Control using very few training samples. From the supervised learning point of view, several supervised samples are available for each class. Without the process of unsupervised clustering, the mapping between clusters and applications can be avoided. Our work focuses on

nonparametric classification methods and addresses the difficult problem of Network Congestion Control using very few training samples. The motivations are twofold. First, as mentioned in Section 1, nonparametric NN method has three important advantages which are suitable for Network Congestion Control in current complex network situation. Second, labelling training data is time consuming and the capability of classification using very few training sample is very useful. The classification method based on well-known port numbers is the most common Network Congestio Control method because many traditional applications are associated with a known port number, for example the web traffic is associated with TCP port 80. The port-based classification method is quite simple and easily extended by adding new application-port pairs to its database. However, the identification method based on well-known ports has become more and more difficult to adapt to the development of the network technology. At present there appears to be increasing Internet applications which do not use well-known port numbers and use dynamic ports to communicate with each other. Thus the network traffic which use dynamic ports to communicate with each other cannot be easily detected by the port-based method. In addition to this, the traffic which is classified as Web may easily be something else carried over TCP port 80. So the port-based method can't adapt to a variety of network traffic. Another Network Congestion Control method is the Deep Packet Inspection (DPI) approach using complete protocol parsing. However, there are many difficulties to use this method. For example, for the purpose of the security, some protocols are employed to encrypt the network data stream, and the contents of the data stream are intentionally hidden from sniffer programs.

i. Overall Description

System Specification

Software Requirements Specifications

- Java1.4
- Swings
- Windows 98
- Python

Hardware Requirements Specifications

- Hard disk: 40 GB
- RAM: 128mb
- Processor: Pentium 4
- Monitor: 15″ color monitor
- Floppy drive: 1.44 MB
- Mouse: HCL
- CD Drive: LG 52X
- Printer: Laser

Software Description

For outline our framework we utilized MATLAB for advancement. MATLAB is most appropriate for our proposed technique because of these worries:

JAVA:-The software can be executed in the Windows 2000 (Professional or Server) or Windows XP Professional environment.

A high-level programming language developed by Sun Microsystems. Java was originally called OAK, and was designed for handheld devices and set-top boxes. Oak was unsuccessful so in 1995 Sun changed the name to Java and modified the language to take advantage of the burgeoning World Wide Web.Java is an object-oriented language similar to C++, but simplified to eliminate language features that cause common programming errors. Java source code files (files with a .java extension) are compiled into a format called bytecode (files with a .class extension), which can then be executed by a Java interpreter. Compiled Java code can run on most computers because

Java interpreters and runtime environments, known as Java Virtual Machines (VMs), exist for most operating systems, including UNIX, the Macintosh OS, and Windows. Byte code can also be converted directly into machine language instructions by a just-in-time compiler (JIT).Java is a general purpose programming language with a number of features that make the language well suited for use on the World Wide Web. Small Java applications are called Java applets and can be downloaded from a Web server and run on your computer by a Java-compatible Web browser, such as Netscape or Microsoft Internet Explorer.Java is a programming language originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from Cand C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to bytecode (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere". Java is currently one of the most popular programming languages in use, and is widely used from application software to web applications.The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java, GNU Classpath, and Dalvik.

Principles of Java

There were five primary goals in the creation of the Java language:

1. It should be "simple, object oriented and familiar".

2. It should be "robust and secure".

3. It should be "architecture neutral and portable".

4. It should execute with "high performance".

5. It should be "interpreted, threaded, and dynamic".

Java Platform

One characteristic of Java is portability, which means that computer programs written in the Java language must run similarly on any supported hardware/operating-system platform. This is achieved by compiling the Java language code to an intermediate representation called Java bytecode, instead of directly to platform-specific machine code. Java bytecode instructions are analogous to machine code, but are intended to be interpreted by a virtual machine (VM) written specifically for the host hardware. End-users commonly use a Java Runtime Environment (JRE) installed on their own machine for standalone Java applications, or in a Web browser for Java applets.Standardized libraries provide a generic way to access host-specific features such as graphics, threading, and networking.A major benefit of using bytecode is porting. However, the overhead of interpretation means that interpreted programs almost always run more slowly than programs compiled to native executables would. Just-in-Time compilers were introduced from an early stage that compile bytecodes to machine code during runtime. Over the years, this JVM built-in feature has been optimized to a point where the JVM's performance competes with natively compiled C code.

Implementations

Sun Microsystems officially licenses the Java Standard Edition platform for Linux, Mac OS X and Solaris. Although in the past Sun has licensed Java to Microsoft, the license has expired and has not been renewed. Through a network of third-party vendors and licensees,[24] alternative Java environments are available for these and other platforms.Sun's trademark license for usage of the Java brand insists that all implementations be "compatible". This resulted in a legal dispute with Microsoft after Sun claimed that the Microsoft implementation did not support RMI or JNI and had added platform-specific features of their own. Sun sued in 1997, and in 2001 won a settlement of US$20 million, as well as a court order enforcing the terms of the license from Sun.As a result, Microsoft no longer ships Java with Windows, and in recent versions of

Windows, Internet Explorer cannot support Java applets without a third-party plugin. Sun, and others, have made available free Java run-time systems for those and other versions of

Windows.Platform-independent Java is essential to the Java EE strategy, and an even more rigorous validation is required to certify an implementation. This environment enables portable server-side applications, such as Web services, Java Servlets, and Enterprise JavaBeans, as well as with embedded systems based on OSGi, using Embedded Java environments. Through the new GlassFish project, Sun is working to create a fully functional, unified open source implementation of the Java EE technologies.Sun also distributes a superset of the JRE called the Java Development Kit (commonly known as the JDK), which includes development tools such as the Java compiler, Javadoc, Jar, and debugger.

Performance

Programs written in Java have a reputation for being slower and requiring more memory than those written in C. However, Java programs' execution speed improved significantly with the introduction of Just-in-time compilation in 1997/1998 for Java 1.1,the addition of language features supporting better code analysis (such as inner classes, StringBuffer class, optional assertions, etc.), and optimizations in the Java Virtual Machine itself, such as HotSpot becoming the default for Sun's JVM in 2000.To boost even further the speed performances that can be achieved using the Java language, Systronix made JStik, a microcontroller based on the aJile Systems line of embeddedJava processors. In addition, the widely used ARM family of CPUs has hardware support for executing Java bytecode through its Jazelle option.

Automatic memory management

Java uses an automatic garbage collector to manage memory in the object lifecycle. The programmer determines when objects are created, and the Java runtime is responsible for recovering the memory once objects are no longer in use. Once no references to an object remain, the unreachable memory becomes eligible to be freed automatically by the garbage collector. Something similar to a memory leak may still occur if a programmer's code holds a reference to an object that is no longer needed, typically when objects that are no longer needed are stored in containers that are still in use. If methods for a nonexistent object are called, a "null pointer exception" is thrown.

One of the ideas behind Java's automatic memory management model is that programmers can be spared the burden of having to perform manual memory management. In some languages, memory for the creation of objects is implicitly allocated on the stack, or explicitly allocated and deallocated from the heap. In the latter case the responsibility of managing memory resides with the programmer. If the program does not deallocate an object, a memory leak occurs. If the program attempts to access or deallocate memory that has already been deallocated, the result is undefined and difficult to predict, and the program is likely to become unstable and/or crash. This can be partially remedied by the use of smart pointers, but these add overhead and complexity. Note that garbage collection does not prevent "logical" memory leaks, i.e. those where the memory is still referenced but never used. Garbage collection may happen at any time. Ideally, it will occur when a program is idle. It is guaranteed to be triggered if there is insufficient free memory on the heap to allocate a new object; this can cause a program to stall momentarily. Explicit memory management is not possible in. Java does not support C/C++ style pointer arithmetic, where object addresses and unsigned integers (usually long integers) can be used interchangeably. This allows the garbage collector to relocate referenced objects and ensures type safety and security. As in C++ and some other object-oriented languages, variables of Java's primitive data types are not objects. Values of primitive types are either stored directly in fields (for objects) or on the stack (for methods) rather than on the heap, as commonly true for objects (but see Escape analysis). This was a conscious decision by Java's designers for performance reasons. Because of this, Java was not considered to be a pure object-oriented programming language. However, as of Java 5.0, auto boxing enables programmers to proceed as if primitive types were instances of their wrapper class.

Purpose

a) Existing system:

As a result of its strict adherence to end-to-end congestion control, the current Internet suffers from two maladies:

Congestion collapse from undelivered packets, and unfair allocations of bandwidth between competing traffic flows.

The first malady — congestion collapse from undelivered packets — arises when packets that are dropped before reaching their ultimate continually consume bandwidth destinations.

The second malady—unfair bandwidth allocation to competing network flows—arises in the Internet for a variety of reasons, one of which is the existence of applications that do not respond properly to congestion. Adaptive applications (e.g., TCP-based applications) that respond to congestion by rapidly reducing their transmission rates are likely to receive unfairly small bandwidth allocations when competing with unresponsive applications. The Internet protocols themselves can also introduce unfairness. The TCP algorithm, for instance, inherently causes each TCP flow to receive a bandwidth that is inversely proportional to its round-trip time [6]. Hence, TCP connections with short round-trip times may receive unfairly large allocations of network bandwidth when compared to connections with longer round-trip times.

The impact of emerging streaming media traffic on traditional data traffic is of growing concern in the Internet community. Streaming media traffic is unresponsive to the congestion in a network, and it can aggravate congestion collapse and unfair bandwidth allocation.

Proposed system: To address the maladies of congestion collapse we introduce and investigate a novel Internet traffic control protocol called *Congestion Free Router* (CFR). The basic principle of CFR is to compare, at the borders of a network, the rates at which packets from each application flow are entering and leaving the network. If a flow's packets are entering the network faster than they are leaving it, then the network is likely buffering or, worse yet, discarding the flow's packets. In other words, the network is receiving more packets than it is capable of handling. CFR prevents this scenario by "patrolling" the network's borders, ensuring that each flow's packets do not enter the network at a rate greater than they are able to leave the network. This patrolling prevents congestion collapse from undelivered packets, because unresponsive flow's otherwise undeliverable packets never enter the network in the first place.

Although CFR is capable of preventing congestion collapse and improving the fairness of bandwidth allocations, these improvements do not come for free. CFR solves these problems at the expense of some additional network complexity, since routers at the border of the network are expected to monitor and control the rates of individual flows in CFR. CFR also introduces added communication overhead, since in order for an edge outer to know the rate at which its packets are leaving the network, it must exchange feedback with other edge routers. Unlike some existing approaches trying to solve congestion collapse, however, CFR's added complexity is isolated to edge routers; routers within the core of the network do not participate in the prevention of congestion collapse. Moreover, end systems operate in total ignorance of the fact that CFR is implemented in the network, so no changes to transport protocols are necessary at end systems.

TRADITIONAL METHOD

Port-Based Application Deduction: Application is combining with a port number, the common network Network Congestion Control method is based on port number and it uses only packet header. In this method the classification is based on well-known port number to a given traffic type, for example web traffic is associated with TCP port 80.The port number of any application is not default, which leads to failure of the port based method.

Packet-Based Analysis: The packet based analysis will give most accurate solution but, there are many pitfalls with this method. Some protocols are encrypted, thus their data stream is intentionally hidden from sniffer programs. Another problem is that, there is no protocol description for organization protocols. Parsing all protocols for all users separately is a computationally unavailable and risky task. In this method, parsing the payload of network packets is highly privacy violation. Signature method of Network Congestion Control consumes less resource which is searching for specific byte patterns, even when packet s in the stateless manner. . The signature based methods is very difficult to maintain signatures with maximum ratio of true positives and minimum ratio of false positives.

RECENT METHOD

Statistics based classification: Packet level trace generates n number of zero payload flows where peer try to connect each other. In this case some statistical feature of the packet-level-trace is grabbed and used to classify the network traffic. This approach is feasible to determine the application type, but specific application/client cannot be determined in general. These flow

characteristics can be highly coded manually or another way is to automatically extract the features of a particular kind of traffic. This method is achieved by combining statistical method with artificial intelligence. There is various data mining approaches combination to apply statistical based classification. Applying statistical based classification will give high accuracy for Network Congestion Control, but the result cannot be exact and accept minor classification errors.

Flow-based Classification: Traffic application based on flow-level data with the same and high level of accuracy is very difficult, because it consist of less detailed input. For application behavior, analyzing the application constraints makes the classification more feasible. The connection patterns is the novel approach to classify traffic based on the application groups, It is represented by graphs, where nodes provides ip address and port pairs information and edge represents flows between source and destination nodes. Connection patterns are analyzed at three levels of details, the social, the functional and the application level. This method operates within the information having no access to payload information, no knowledge about port number and no information behind what current flow collectors give. On the other hand, connection patterns require a high quality of flow information and finished flow period to perform the analyses.

Motivations And Scope:

Network congestion in data networking and queueing theory is the reduced quality of service that occurs when a network node or link is carrying more data than it can handle. Typical effects include queueing delay, packet loss or the blocking of new connections. A consequence of congestion is that an incremental increase in offered load leads either only to a small increase or even a decrease in network throughput. Network protocols that use aggressive retransmissions to compensate for packet loss due to congestion can increase congestion, even after the initial load has been reduced to a level that would not normally have induced network congestion. Such networks exhibit two stable states under the same level of load. The stable state with low throughput is known as **congestive collapse**. Networks use **congestion control** and **congestion avoidance** techniques to try to avoid collapse. These include: exponential backoff in protocols such as CSMA/CA in 802.11 and the similar CSMA/CD in the original Ethernet, window reduction in TCP, and fair queueing in devices such as routers and network switches. Other techniques that address congestion include priority schemes which transmit some packets with higher priority ahead of others and the explicit allocation of network resources to specific flows through the use of admission control. This paper presented result executing clustering, feature extracting and classification process in terms of accuracy. The accurate classification depends upon how effectively the clustering or features extracted and every classification algorithm relies on a good accuracy.We use clustering and feature to improve the classification process.The implementation process is described phase by phase with GUI development.

3. LITERATURE SURVEY

1) Congestion Free Router (CFR): The paper publish in 2012 , they introduced the novel called as congestion free router(CFR). CFR is capable of preventing congestion collapse and improving the fairness of bandwidth allocations, these improvement do not come for free.

2) Core Stateless Fair Queuing: The university of California propose the mechanism Core Stateless Fair Queuing (CSFQ). It used to provide fair bandwidth allocation. They introduce this to promoting Fairness in the internet. This unfair bandwidth arises in the network by the variety of the reasons, like application which do not adapt to congestion.

3) Round Trip Time (RTT): In 2013, to maintain scalability they used the time interval feedback i.e. Round Trip Time(RTT). It is a shortest time observing algorithm. They also use the Time Sliding Window is a rate estimate algorithm that monitored each flow bit rate.

4) Packet Drop Prevention (PDP): In research paper publish in 2015, They propose the novel for Congestion avoidance known as Packet Drop Prevention(PDP). This help in faster data transfer without the loss of packets.

5) RESEARCH PAPERS: There is a huge collection of papers related to Network Congestion Control; K-means; Naïve Bayes; Decision Tree and K-Nearest Neighbour. After reviewing them, following papers are found relevant for the present work of my dissertation:-

R.S.AnuGowsalya,Dr.S.Miruna Joe Amali [01], In this paper they explain, Network Congestion Control is an automated process which categorizes computer network traffic according to various parameters into a number of traffic classes. Many supervised classification algorithms and unsupervised clustering algorithms have been applied to categorize Internet traffic. Traditional Network Congestion Control methods include the port-based prediction methods and payload-based deep inspection methods. In current network environment, the traditional methods suffer from a number of practical problems, such as dynamic ports and encrypted applications. In order to improve the classification accuracy, Support Vector Machine (SVM) estimator is proposed to categorize the traffic by application. In this, traffic flows are described using the discretized statistical features and flow correlation information is modeled by bag-of-flow (BoF). This methodology uses flow statistical feature based Network Congestion Control to enhance feature discretization. This approach for Network Congestion Control improves the classification performance effectively by incorporating correlated information into the classification process. The experimental results show that the proposed scheme can achieve much better classification performance than existing state-of-the-art Network Congestion Control methods.

Jun Zhang, Yang Xiang, Yu Wang [02], In this paper they explain, Network Congestion Control has wide applications in network management, from security monitoring to quality of service measurements. Recent research tends to apply machine learning techniques to flow statistical feature based classification methods. The nearest neighbor (NN)-based method has exhibited superior classification performance. It also has several important advantages, such as no requirements of training procedure, no risk of overfitting of parameters, and naturally being able to handle a huge number of classes. However, the performance of NN classifier can be severely affected if the size of training data is small. In this paper, we propose a novel nonparametric approach for Network Congestion Control, which can improve the classification performance effectively by incorporating correlated information into the classification process. We analyze the new classification approach and its performance benefit from both theoretical and empirical perspectives. A large number of experiments are carried out on two real-world traffic data sets to validate the proposed approach. The results show the Network Congestion Control performance can be improved significantly even under the extreme difficult circumstance of very few training samples.

R.S. ANU GOWSALYA, S. MIRUNA JOE AMALI [03], In this paper they explain, Network Congestio Control is of fundamental importance to numerous other network activities, from security monitoring to accounting, and from Quality of Service to providing operators with useful forecasts for long-term provisioning. Naive Bayes estimator is applied to categorize the traffic by application. Uniquely, this work capitalizes on hand-classified network data, using it as input to a supervised Naive Bayes estimator. A novel Network Congestion Control scheme is used to improve classification performance when few training data are available. In the proposed scheme, traffic flows are described using the discretized statistical features and flow correlation information is modeled by bag-of-flow (BoF). A novel parametric approach for Network Congestion Control, which can improve the classification performance effectively by incorporating correlated information into the classification process. Then analyze the new classification approach and its performance benefit from both theoretical and empirical perspectives. Finally, a large number of experiments are carried out on large-scale real-world traffic datasets to evaluate the proposed scheme. The experimental results show that the proposed scheme can achieve much better classification performance than existing state-of-the-art Network Congestion Control methods.

Ms.ZebaAtiqueShaikh, Prof. Dr. D.G. Harkut [04], In this paper they explain, Network Network Congestion Control can be used to identify different applications and protocols that exist in a network. Actions such as monitoring, discovery, control and optimization can be performed by using classified network traffic. The overall goal of network Network Congestion Control is improving the network performance. Once the packets are classified as belonging to a particular application, they are marked. These markings or flags help the router determine appropriate service policies to be applied for those flows. This paper gives an overview of available network classification methods and techniques. Researchers can utilize this paper for approaching real time network Network Congestion Control. Network Congestion Control using payload, statistical analysis, deep packet inspection, naïve Bayesian estimator and Bayesian neural networks are reviewed in this paper.

PallaviSinghal, Rajeev Mathur [05], In this Paper they explain, Network Network Congestion Control is extensively required mainly for many network management tasks such as flow prioritization, traffic shaping/policing, and diagnostic monitoring. Similar to network management tasks, many network engineering problems such as workload characterization and modeling, capacity planning, and route provisioning also benefit from accurate identification of network traffic .This paper presents review on all the work done related to Network Traffic Management since 1993 to 2013 in various fields like artificial intelligence, neural network, ATM and wireless networks.

G.Suganya [06], In this paper they explain, Network Congestion Control technique is an important tool for network and system security in the environments such as cloud computing based environment. Modern Network Congestion Control methods plans to take the gain of flow statistical features and machine learning methods, but the classification performance is affected by reduced supervised information, and unfamiliar applications. In addition detection of anomalies in the flow level is not considered in earlier approaches. Current work proposes Flow-level anomaly detection with the framework of Unknown Flow Detection approaches. Flow-level anomaly can be detected by using Synthetic flow-level traffic trace generation approach(SG –FLT). The two major challenges with such an approach are to characterize normal and anomalous network behavior, and to discover realistic models defining normal and anomalous traffic at the flow level. Unknown flow detection approach has been performed by Flow level propagation and finding the

correlated flows to boost the classification accuracy. Performance evaluation is conducted on real-world network traffic datasets which demonstrates that the proposed scheme provides efficient performance than existing methods in the complex network environment.

Andrew W. Moore [07], In this paper they explain, accurate Network Congestion Control is of fundamental importance to numerous other network activities, from security monitoring to accounting, and from Quality of Service to providing operators with useful forecasts for long-term provisioning. We apply a Naive Bayes estimator to categorize traf_c by application. Uniquely, our work capitalizes on hand-classi_ed network data, using it as input to a supervised Naive Bayes estimator. In this paper we illustrate the high level of accuracy achievable with the Naive Bayes estimator. We further illustrate the improved accuracy of refined variants of this estimator. Our results indicate that with the simplest of Naive Bayes estimator we are able to achieve about 65% accuracy on per-flow classification and with two powerful refinements we can improve this value to better than 95%; this is a vast improvement over traditional techniques that achieve 50.70%. While our technique uses training data, with categories derived from packet-content, all of our training and testing was done using header-derived discriminators. We emphasize this as a powerful aspect of our approach: using samples of well-known traffic to allow the categorization of traffic using commonly-available information alone.

M. Tamilkili [08], In this paper they explain, recent research tends to apply machine learning techniques to classify network traffic using flow statistical features and IP packet payload. This survey paper looks into the use of Machine Learning (ML) algorithms for Network Congestion Control in the period 2009 to 2013. It provides motivation for the application of ML techniques to network Network Congestion Control, and reviews significant work in this area. The recent techniques focus on statistical features of network packets to provide security. A review of ML techniques has been done.

G. Rubadevi, R. Amsaveni [09], in this paper they explain, the classification and identification of network application from network traffic flow provides various advantages to a number of fields such as security monitoring, intrusion detection and to tackle a number of network security problems including lawful interception. In this paper traffic flow is described by using the discretized statistical features. The flow correlation information of the network traffic flow is

modeled by Flow Container (FC). In this paper novel hybrid aggregated classifier is proposed. First, low density flow and high density flow is analyzed. For Low density flow C4.5 classifier is used and high density flow Naïve Bayesian classifier is used and finally aggregated result is provided. The aggregated result is compared with machine learning algorithm such as Single Naïve Bayesian predictor. The proposed system enhances the accuracy rate as well as improves the performance of the system.

Bin Hu, Yi Shen [10], In this paper they explain, the classification of network applications is essential to numerous network activities, including quality of service, accounting, and intrusion detection. Originally, port-match was regarded as the most popular and effective method. After that, with the booming of new Internet businesses, researchers shifted to the use of payload analysis. However, the payload analysis is unable to process encrypted traffic, which motivated scientists to develop more general and effective solutions. To do so, Network Congestion Control has been of great concern to academia and industry and gradually formed a relatively independent area of research. Due to the ability of handling large number of flow samples and multidimensional feature spaces efficiently, Machine Learning based classification has stood out among multitude research findings. This paper aims to provide an overview of recent advances in such study area. We focus on how to divide machine learning based classification into two categories: supervised and clustering. We also present the algorithms of creating specific feature sets and classification models such as Genetic Algorithm and Bayesian algorithm. Finally, we compare the efficiency of these algorithms and discuss the future direction of machine.

PROPOSED MODEL

In this proposed framework, we introduced a new strategy to investigate network traffic activities utilizing Java framework. Proposed framework examines suspicious activities and malignant information travelling over the networks by utilizing different technique like clustering and classification. This framework load and process traffic data into proposed framework and examining the information and the network traffic data is differentiated as IP Wise, Port Wise and Protocol Wise. K-Means algorithm is utilized for clustering and following this phase feature extraction is done and for Network Congestion Control we utilized three classification algorithms K-means and Naïve Bayes. This paper also evaluates the classification performance between all three algorithms. The proposed framework is successfully implemented utilizing different phases which are described below.

Execution flow

Step 1:- Load network traffic dataset

This phase load the network traffic dataset into proposed framework. The network traffic dataset is taken from KDD Cup 1999. After loading dataset, the framework differentiated the dataset into IP wise, Port wise and Protocol wise.

Data pre-processing is performed to remove the noises or we can say data cleaning is utilized to remove the unnecessary information contains in network traffic dataset.

Step 2: - Clustering

This phase perform clustering on network traffic dataset utilizing K-Means algorithm. The earlier work that reviewed the algorithms in network traffic classification suggested that K-means for clustering is robust and fast and help the classification phase to recognize unfamiliar applications hence, improving the classification accuracy.

K-means is an unsupervised machine learning algorithm mostly used when the dataset is unlabelled. It works by iteratively assign each data point to one of K group based on provided features. K is groups in data and each data point clustered based on feature measurement and assign

to a group K. The value of K can be found by calculating the mean distance between data points and their cluster centroid.

$$\mu(x) = \frac{1}{n}\sum_{k=1}^{n} x$$

Where ***n*** represents the no of data points and the ***x*** corresponds to the centroids.

Clustering assigns a label to each traffic data so that it shares similar characteristics that represent their traffic application. After this step the traffic data is divided into different clusters where each data in cluster shares common characteristics hence, strongly interrelate to feature of interest.

Step 3: - Feature Extraction

This phase utilized to extract the features from datasets. The features which are extracted described in table below.

Features	Comment
Src_addr	source address
Src_port	source port
Dst_addr	destination address
Dst_port	destination port
Ip_prot	ip protocol
Tcp_ags	xor of TCP flags
N_ack	number of ACK flag
N_psh	number of PSH flag
Duration_sr	Duration
Pkt_max_lg	max of packet size
Pkt_mean_sr	mean of packet size

pktvar_sr	variance of packet size
Packets_lg	number of packets
Octets_lg	number of bytes
Speed_sr	bytes per second
Pkt_rate_sr	packets per second

Naive Bayes

Naive Bayes classifiers, a family of classifiers that are based on the popular Bayes' probability theorem, are known for creating simple yet well performing models, especially in the fields of document classification and disease prediction.Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, an analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of naive Bayes classifiers.[5] Still, a comprehensive comparison with other classification algorithms in 2006 showed that Bayes classification is outperformed by other approaches, such as boosted trees or random forests.One of the recent approaches classifies the traffic by using the simple and effective probabilistic Naive

Bayes (NB) classifier. It employs the Bayes'theorem with naive feature independence assumptions. The main reason for the underperformance of a number of traditional classifiers including NB is the lack of the feature discretization process. NB algorithm is used to produce a set of posterior probabilities as predictions for each testing flow. It is different to the conventional NB classifier which directly assigns a testing flow to a class with the maximum posterior probability. Considering correlated flows, the predictions of multiple flows will be aggregated to make a final prediction

The Mathematics of the Naive Bayes Algorithm:

The basis of Naive Bayes algorithm is Bayes' theorem or alternatively known as Bayes' rule or Bayes' law. It gives us a method to calculate the conditional probability, i.e., the probability of an event based on previous knowledge available on the events. More formally, Bayes' Theorem is stated as the following equation:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

The components of the above statement are:

- $P(A|B)$: Probability (conditional probability) of occurrence of event A given the event B is true
- $P(A)$ and $P(B)$: Probabilities of the occurrence of event A and B respectively
- $P(B|A)$: Probability of the occurrence of event B given the event A is true

The terminology in the Bayesian method of probability (more commonly used) is as follows:

- A is called the **proposition** and B is called the **evidence.**
- $P(A)$ is called the **prior** probability of proposition and $P(B)$ is called the **prior** probability of evidence.
- $P(A|B)$ is called the **posterior.**
- $P(B|A)$ is the **likelihood**.

This sums the Bayes' theorem as

$$\text{Posterior} = \frac{(\text{Likelihood}).(\text{Proposition prior probability})}{\text{Evidence prior probability}}$$

Naive Bayes Classifier uses Bayes Theorem, which finds the probability of an event given the probability of another event that has already occurred. Naive Bayes classifier performs extremely well for problems which are linearly separable and even for problems which are non-linearly separable it performs reasonably well. At the beginning of the analysis of the Naive Bayes algorithm, it is important to establish the accuracy of it. Naive Bayes algorithm performed on real Internet traffic described by all discriminators is having a good accuracy. We also illustrate the performance of Naive Bayes technique by considering how many bytes were classified correctly. Here again, we have used all datasets to estimate the average amount of bytes correctly classified.

Leaky Bucket Algorithm

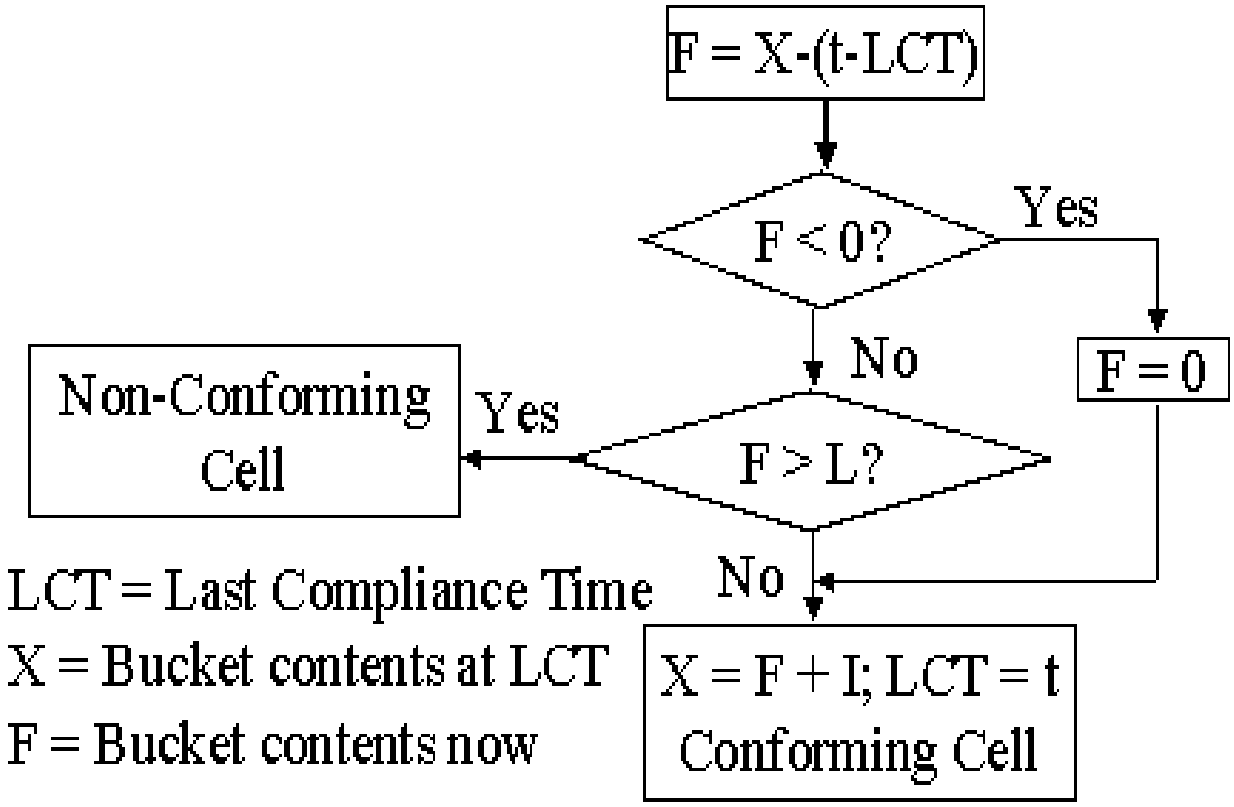
Flow rate: If packets are transmitted in unregulated flow, the router holds the packet and transmits the packet in regulated flow. In Leaky Bucket algorithm, router applies a constant output rate regardless of the input flow rate. In networks, packets can arrive faster than router flow rate, cause bucket overflow and packets are dropped.

Bucket size: The bucket size is important in controlling buffer overflow and reducing maximum number of packets dropped. With small bucket size, number of packets dropped increases. With large bucket size, number of packets dropped decreases. Adjustable bucket size, allows the router to release packets into the network according to the burstiness of the network.

Consider bucket size of 1000. The queued packets are as follows 200, 800, 450, 400, 200. In step 1, the bucket size is greater than packet size (1000>200). Remaining bucket size will be 800 (1000-200). In step 2, the bucket size is greater than packet size (800>400). Remaining bucket size will be 400 (800-400). In step 3, the bucket size is smaller than packet size (400<450). The process stopped. The remaining packets will be dropped.

In leaky bucket algorithm, Adjustable bucket size and flow rate decreases the number of packets dropped in the network.

Adaptive Router: Non-adaptive router, do not base their routing decisions on different flow rate. In non-adaptive router, if number of packets transmitted by the transmitter, only some of the packets able to reach receiver because of variable flow rate. Every node has a buffer for every destination and buffer has maximum holding capacity of packets. Because of non-adaptive router, number of buffer requirement will increase. Non Adaptive router gives poor performance if volume or topologies change over time. In non-adaptive router, current state of network ignored. Adaptive Router changes their routing decisions to reflect changes in the flow rate. In adaptive router, high throughput is achieved with small buffer size and zero packets dropped. An adaptive routing strategy can aid in congestion control. Adaptive Router is flow-aware router. Adaptive router improves network performance. A network model with two sources called transmitter node and receiver node with adaptive router. When the transmission begins, both source nodes will begin transmission simultaneously. When the router notices a possibility change in flow rate occurrence, it adjusts the packet size and alerts the sources.



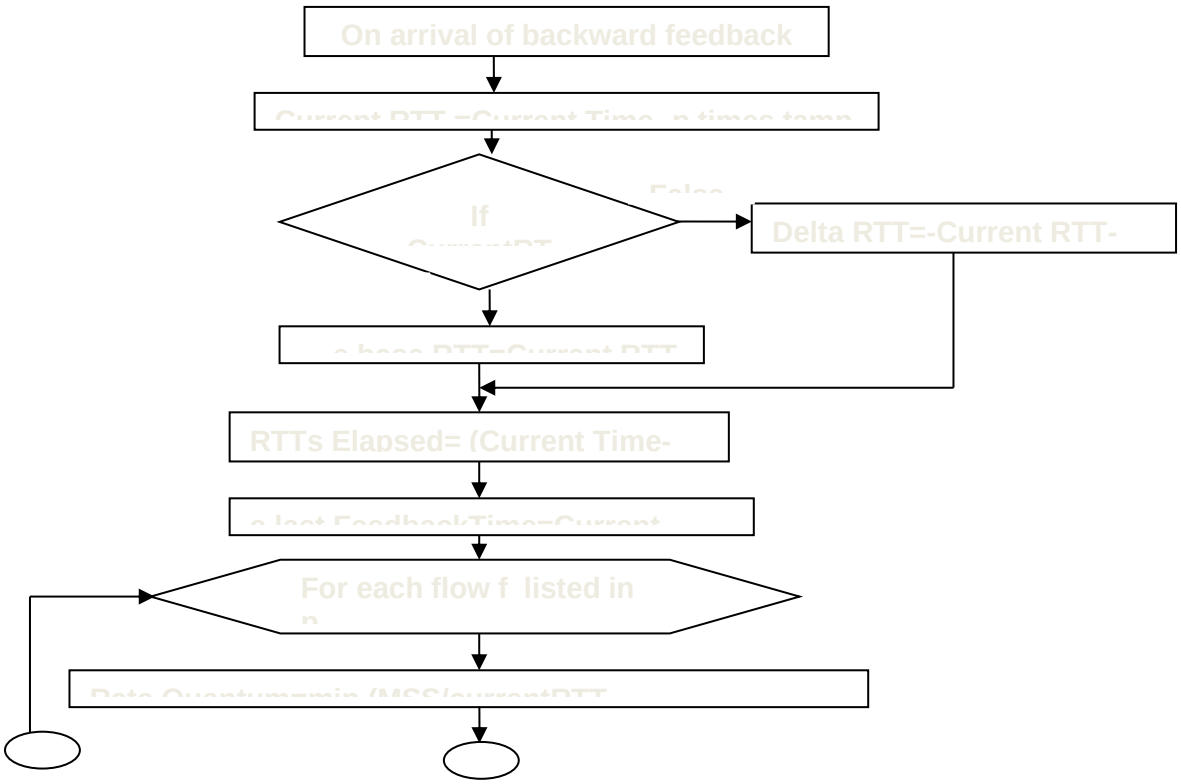
Rate Control Protocol

Rate Control Protocol (RCP) is a congestion control algorithm designed for fast download times (i.e. aka user response times, or flow-completion times). Whereas other modifications to TCP (e.g. STCP, Fast TCP, XCP) are designed to work for specialized applications that use long-lived flows (scientific applications and supercomputer centers), RCP is designed for the typical flows of typical users in the Internet today. For example, a mid-size flow in the Internet today contains 1000 packets and TCP typically makes them last 10x longer than need-be (XCP is even worse). RCP makes flows finish close to the minimum possible, leading to a perceptible improvement for web users, distributed computing, and distributed file-systems.

The foremost goal in RCP is short flow completion times, or in other words to make a flow complete as quickly—after all this is what most users experience and care about when they interact with the network in any way. shows RCP's pros and cons. The biggest plus of RCP is the short flow completion times under a wide range of network and traffic characteristics—which are in fact quite close to what flows would achieve if they were processor-shared. It turns out that as a consequence of this, RCP also achieves the other goals that high-speed TCPs and XCP achieve, which is efficient and fair network usage in the presence of a few high-bandwidth transfers. In summary, there are four main features of RCP that make it an appealing and practical congestion control algorithm Protocol (RCP) is a congestion control algorithm designed for fast download times (i.e. aka user response times, or flow-completion times). Whereas other modifications to TCP (e.g. STCP, Fast TCP, XCP) are designed to work for specialized applications that use long-lived flows (scientific applications and supercomputer centers), RCP is designed for the typical

flows of typical users in the Internet today. For example, a mid-size flow in the Internet today contains 1000 packets and TCP typically makes them last 10x longer than need-be (XCP is even worse). RCP makes flows finish close to the minimum possible, leading to a perceptible improvement for web users, distributed computing, and distributed file-systems.

Flow Chart



Module Split-Up:

PROJECT MODULES:

The various modules in the protocol are as follows:

Module 1: -

SOURCE MODULE.

Module 2: -

INROUTER ROUTER MODULE.

Module 3: -

ROUTER MODULE.

Module 4: -

OUTROUTER ROUTER MODULE.

Module 5: -

DESTINATION MODULE.

SOURCE MODULE:-

The task of this Module is to send the packet to the InRouter router.

INROUTER ROUTER MODULE:-

An edge router operating on a flow passing into a network is called an InRouter router. CFR prevents congestion collapse through a combination of per-flow rate monitoring at OutRouter routers and per-flow rate control at In Router routers. Rate control allows an InRouter router to police the rate at which each flow's packets enter the network. InRouter Router contains a flow classifier, per-flow traffic shapers (e.g., leaky buckets), a feedback controller, and a rate controller

ROUTER MODULE:-

The task of this Module is to accept the packet from the InRouter router and send it to the OutRouter router.

OUTROUTER ROUTER MODULE:-

An edge router operating on a flow passing out of a network is called an OutRouter router. CFR prevents congestion collapse through a combination of per-flow rate monitoring at OutRouter

routers and per-flow rate control at InRouter routers. Rate monitoring allows an OutRouter router to determine how rapidly each flow's packets are leaving the network. Rate monitored using a rate estimation algorithm such as the Time Sliding Window (TSW) algorithm. OutRouter Router contains a flow classifier, Rate monitor, and a feedback controller.

DESTINATION MODULE:-

The task of this Module is to accept the packet from the OutRouter router and stored in a file in the Destination machine. We distinguish a semantic gap where modern network observing applications need to work at the transport layer furthermore, past, while existing checking frameworks work at the network layer. To bridge this hole and empower forceful enhancements, here we presented the idea of clustering, in view of the key reflection of the classification and process it. Following are the key goals;

1. Traffic capture and classification

2. Find network traffic usage,

3. To analyse the network traffic, measure, and classify it according to the need of different ISPs.

4. Present the performance evaluation.

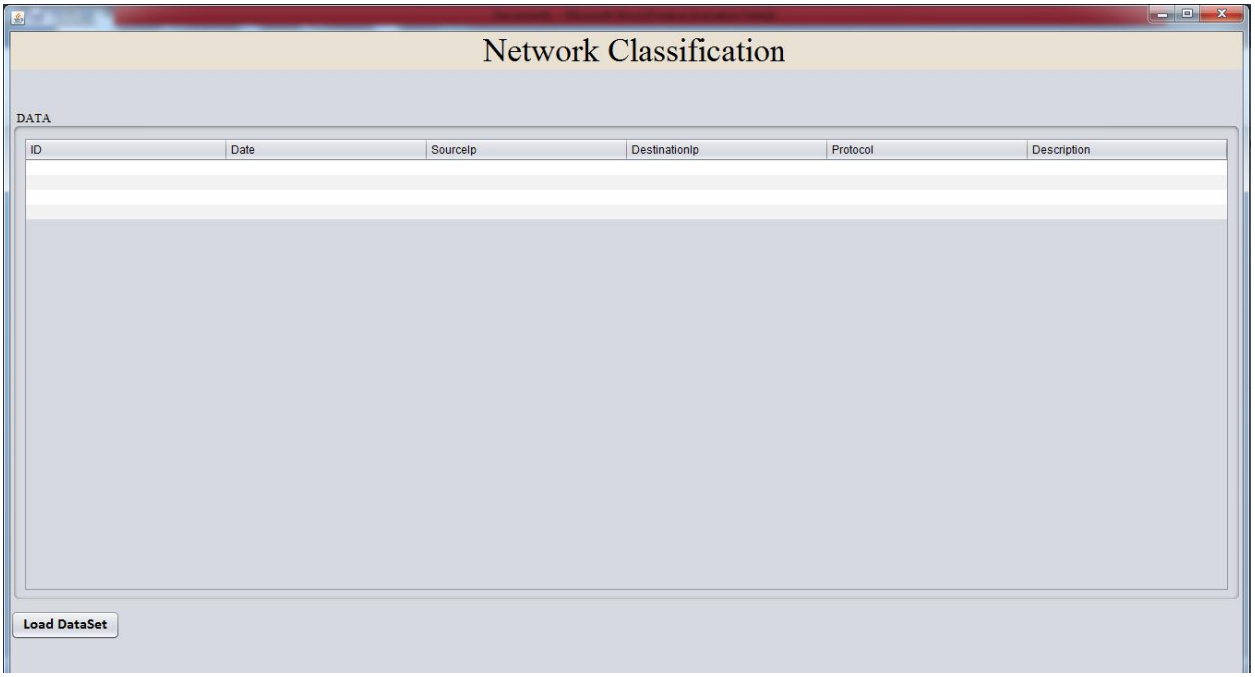
Result and Discussion

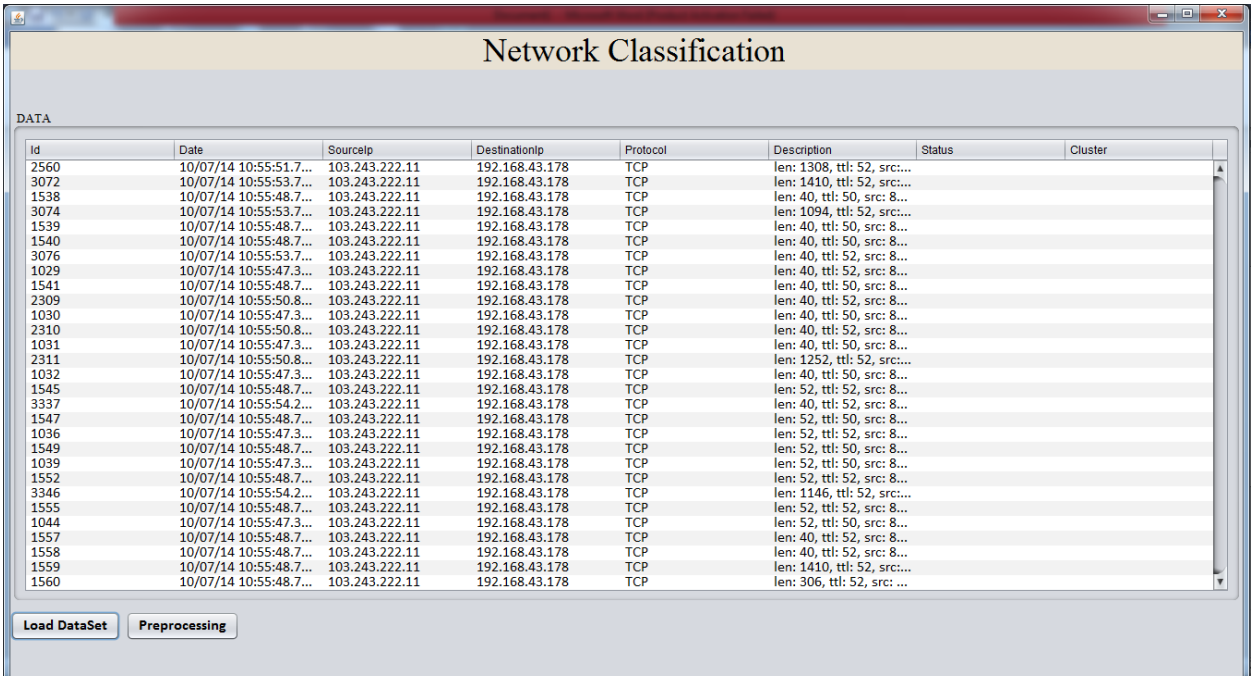
In this chapter the result analysis of proposed work is described. This paper presented result executing clustering, feature extracting and classification process in terms of accuracy. The accurate classification depends upon how effectively the clustering or features extracted and every classification algorithm relies on a good accuracy.We use clustering and feature to improve the classification process.

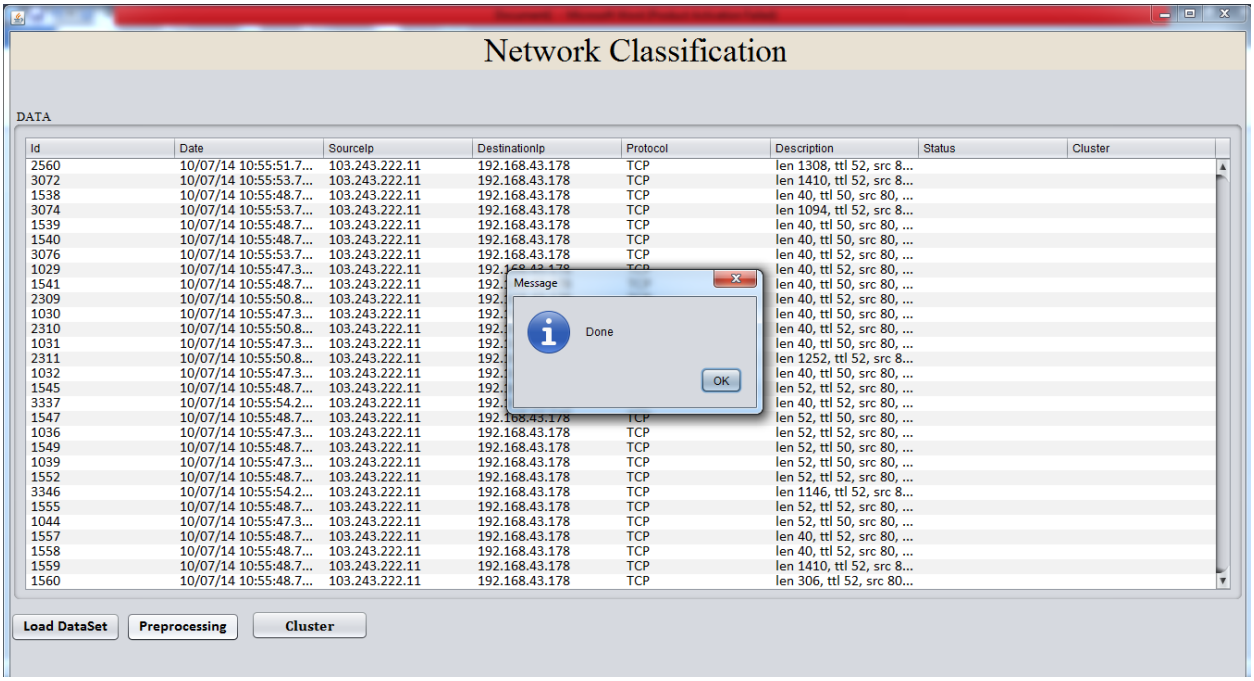
The implementation process is described phase by phase with GUI development:-

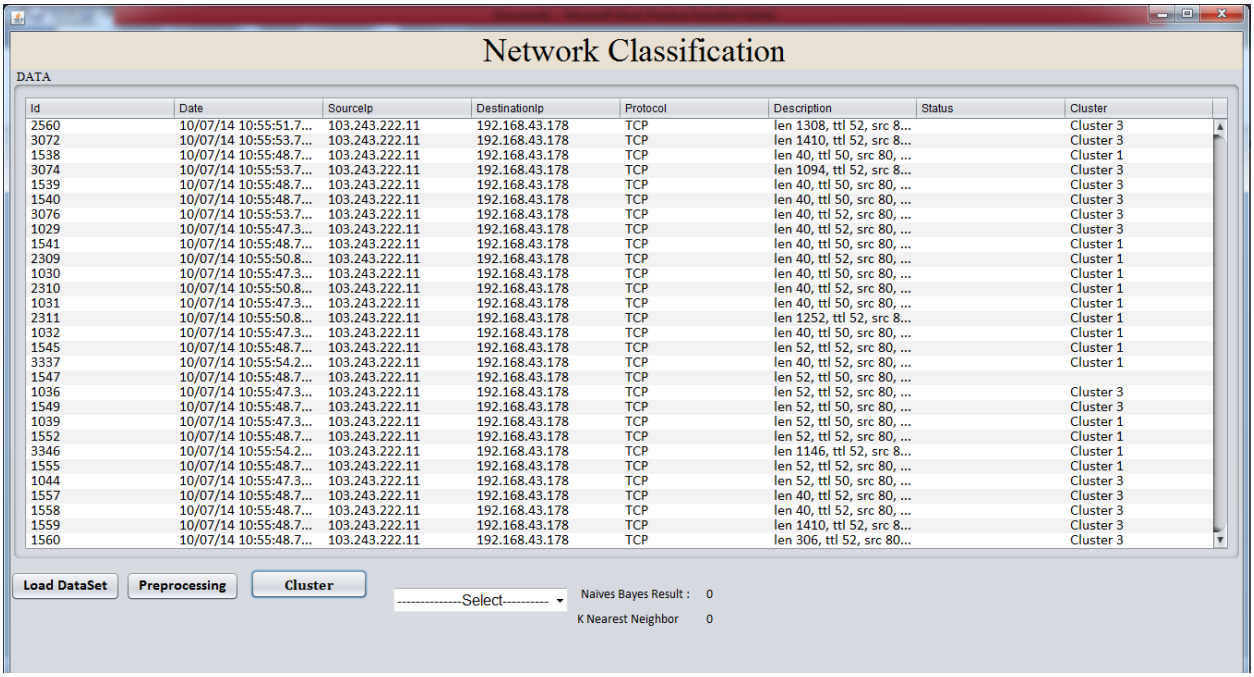
Output

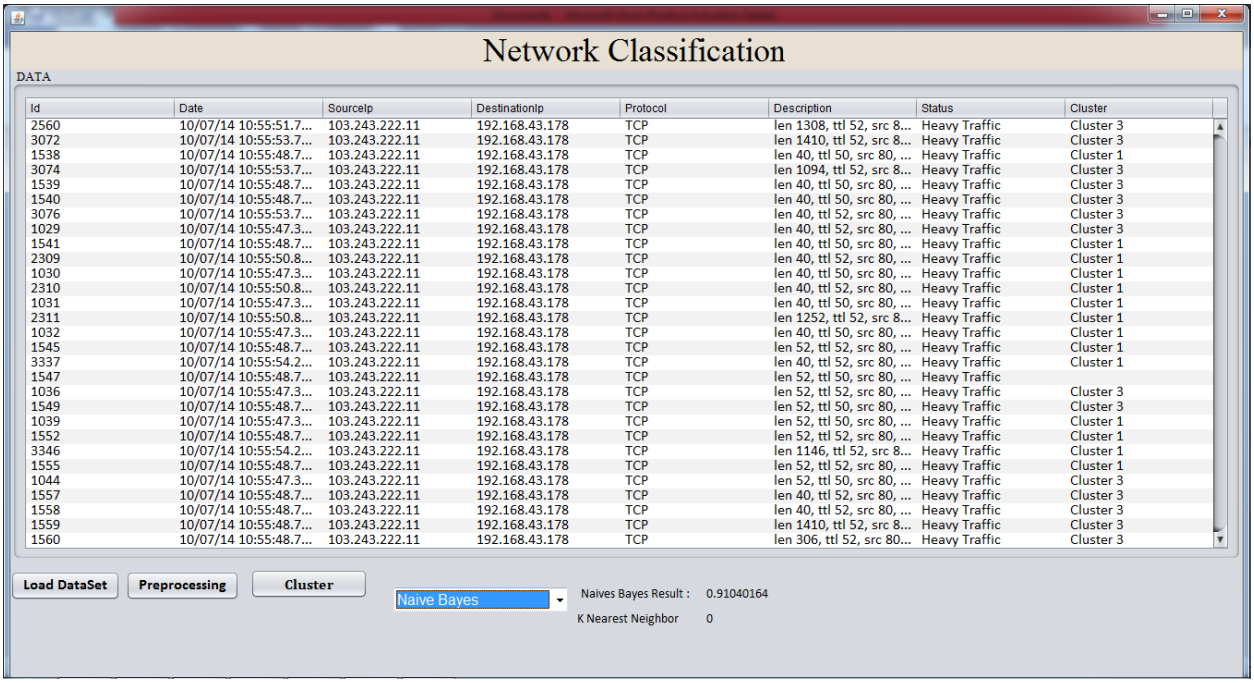

35%

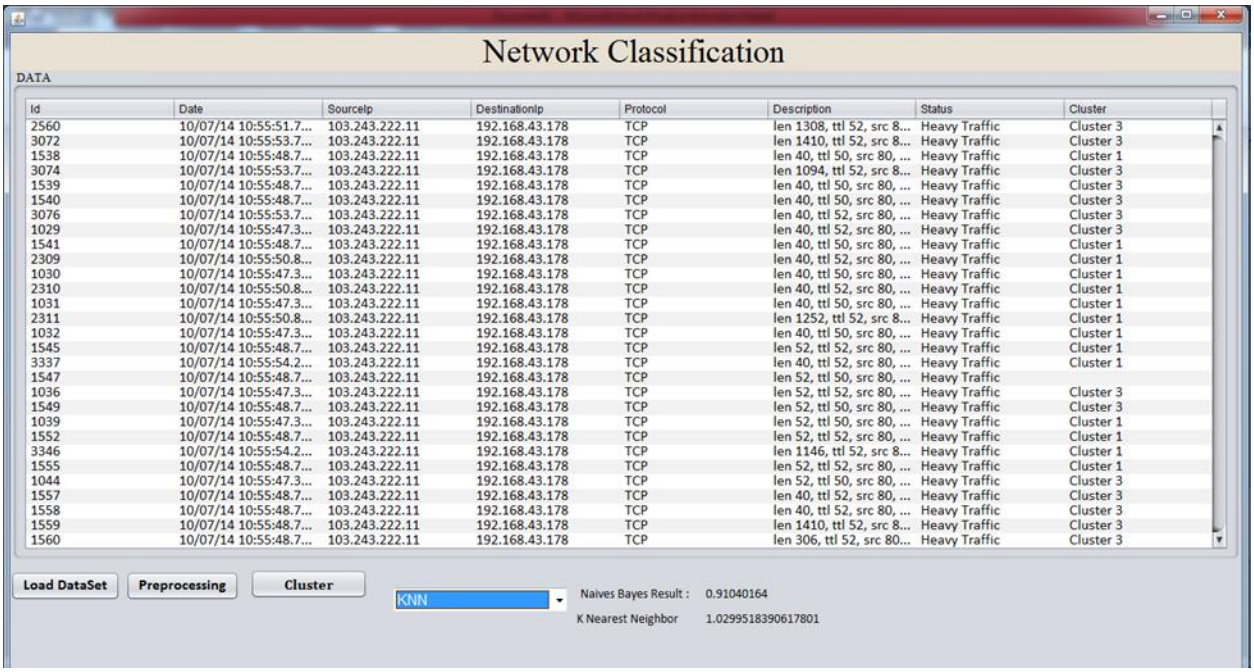

Network Classification
DATA
ID
Date
SourceIp
DestinationIp
Protocol
Description
Load DataSet


Network Classification
DATA
Id Date SourceIp DestinationIp Protocol Description Status Cluster
2560 10/07/14 10:55:51.7... 103.243.222.11 192.168.43.178 TCP len: 1308, ttl: 52, src:...
3072 10/07/14 10:55:53.7... 103.243.222.11 192.168.43.178 TCP len: 1410, ttl: 52, src:...
1538 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len: 40, ttl: 50, src: 8...
3074 10/07/14 10:55:53.7... 103.243.222.11 192.168.43.178 TCP len: 1094, ttl: 52, src:...
1539 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len: 40, ttl: 50, src: 8...
1540 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len: 40, ttl: 50, src: 8...
3076 10/07/14 10:55:53.7... 103.243.222.11 192.168.43.178 TCP len: 40, ttl: 52, src: 8...
1029 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len: 40, ttl: 52, src: 8...
1541 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len: 40, ttl: 50, src: 8...
2309 10/07/14 10:55:50.8... 103.243.222.11 192.168.43.178 TCP len: 40, ttl: 52, src: 8...
1030 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len: 40, ttl: 50, src: 8...
2310 10/07/14 10:55:50.8... 103.243.222.11 192.168.43.178 TCP len: 40, ttl: 52, src: 8...
1031 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len: 40, ttl: 50, src: 8...
2311 10/07/14 10:55:50.8... 103.243.222.11 192.168.43.178 TCP len: 1252, ttl: 52, src:...
1032 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len: 40, ttl: 50, src: 8...
1545 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len: 52, ttl: 52, src: 8...
3337 10/07/14 10:55:54.2... 103.243.222.11 192.168.43.178 TCP len: 40, ttl: 52, src: 8...
1547 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len: 52, ttl: 50, src: 8...
1036 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len: 52, ttl: 52, src: 8...
1549 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len: 52, ttl: 50, src: 8...
1039 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len: 52, ttl: 50, src: 8...
1552 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len: 52, ttl: 52, src: 8...
3346 10/07/14 10:55:54.2... 103.243.222.11 192.168.43.178 TCP len: 1146, ttl: 52, src:...
1555 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len: 52, ttl: 52, src: 8...
1044 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len: 52, ttl: 50, src: 8...
1557 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len: 40, ttl: 52, src: 8...
1558 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len: 40, ttl: 52, src: 8...
1559 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len: 1410, ttl: 52, src:...
1560 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len: 306, ttl: 52, src: ...
Load DataSet
Preprocessing


Network Classification
DATA
Id Date SourceIp DestinationIp Protocol Description Status Cluster
2560 10/07/14 10:55:51.7... 103.243.222.11 192.168.43.178 TCP len 1308, ttl 52, src 8...
3072 10/07/14 10:55:53.7... 103.243.222.11 192.168.43.178 TCP len 1410, ttl 52, src 8...
1538 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ...
3074 10/07/14 10:55:53.7... 103.243.222.11 192.168.43.178 TCP len 1094, ttl 52, src 8...
1539 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ...
1540 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ...
3076 10/07/14 10:55:53.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ...
1029 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ...
1541 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ...
2309 10/07/14 10:55:50.8... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ...
1030 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ...
2310 10/07/14 10:55:50.8... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ...
1031 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ...
2311 10/07/14 10:55:50.8... 103.243.222.11 192.168.43.178 TCP len 1252, ttl 52, src 8...
1032 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ...
1545 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 52, ttl 52, src 80, ...
3337 10/07/14 10:55:54.2... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ...
1547 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 52, ttl 50, src 80, ...
1036 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 52, ttl 52, src 80, ...
1549 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 52, ttl 50, src 80, ...
1039 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 52, ttl 50, src 80, ...
1552 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 52, ttl 52, src 80, ...
3346 10/07/14 10:55:54.2... 103.243.222.11 192.168.43.178 TCP len 1146, ttl 52, src 8...
1555 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 52, ttl 52, src 80, ...
1044 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 52, ttl 50, src 80, ...
1557 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ...
1558 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ...
1559 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 1410, ttl 52, src 8...
1560 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 306, ttl 52, src 80...
Message
Done
OK
Load DataSet
Preprocessing
Cluster


Network Classification
DATA
Id Date SourceIp DestinationIp Protocol Description Status Cluster
2560 10/07/14 10:55:51.7... 103.243.222.11 192.168.43.178 TCP len 1308, ttl 52, src 8... Cluster 3
3072 10/07/14 10:55:53.7... 103.243.222.11 192.168.43.178 TCP len 1410, ttl 52, src 8... Cluster 3
1538 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Cluster 1
3074 10/07/14 10:55:53.7... 103.243.222.11 192.168.43.178 TCP len 1094, ttl 52, src 8... Cluster 3
1539 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Cluster 3
1540 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Cluster 3
3076 10/07/14 10:55:53.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Cluster 3
1029 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Cluster 3
1541 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Cluster 1
2309 10/07/14 10:55:50.8... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Cluster 1
1030 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Cluster 1
2310 10/07/14 10:55:50.8... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Cluster 1
1031 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Cluster 1
2311 10/07/14 10:55:50.8... 103.243.222.11 192.168.43.178 TCP len 1252, ttl 52, src 8... Cluster 1
1032 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Cluster 1
1545 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 52, ttl 52, src 80, ... Cluster 1
3337 10/07/14 10:55:54.2... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Cluster 1
1547 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 52, ttl 50, src 80, ...
1036 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 52, ttl 52, src 80, ... Cluster 3
1549 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 52, ttl 50, src 80, ... Cluster 3
1039 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 52, ttl 50, src 80, ... Cluster 1
1552 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 52, ttl 52, src 80, ... Cluster 1
3346 10/07/14 10:55:54.2... 103.243.222.11 192.168.43.178 TCP len 1146, ttl 52, src 8... Cluster 1
1555 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 52, ttl 52, src 80, ... Cluster 1
1044 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 52, ttl 50, src 80, ... Cluster 3
1557 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Cluster 3
1558 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Cluster 3
1559 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 1410, ttl 52, src 8... Cluster 3
1560 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 306, ttl 52, src 80... Cluster 3
Load DataSet
Preprocessing
Cluster
----------Select----------
Naives Bayes Result : 0
K Nearest Neighbor 0


Network Classification
DATA
Id Date SourceIp DestinationIp Protocol Description Status Cluster
2560 10/07/14 10:55:51.7... 103.243.222.11 192.168.43.178 TCP len 1308, ttl 52, src 8... Heavy Traffic Cluster 3
3072 10/07/14 10:55:53.7... 103.243.222.11 192.168.43.178 TCP len 1410, ttl 52, src 8... Heavy Traffic Cluster 3
1538 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Heavy Traffic Cluster 1
3074 10/07/14 10:55:53.7... 103.243.222.11 192.168.43.178 TCP len 1094, ttl 52, src 8... Heavy Traffic Cluster 3
1539 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Heavy Traffic Cluster 3
1540 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Heavy Traffic Cluster 3
3076 10/07/14 10:55:53.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Heavy Traffic Cluster 3
1029 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Heavy Traffic Cluster 3
1541 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Heavy Traffic Cluster 1
2309 10/07/14 10:55:50.8... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Heavy Traffic Cluster 1
1030 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Heavy Traffic Cluster 1
2310 10/07/14 10:55:50.8... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Heavy Traffic Cluster 1
1031 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Heavy Traffic Cluster 1
2311 10/07/14 10:55:50.8... 103.243.222.11 192.168.43.178 TCP len 1252, ttl 52, src 8... Heavy Traffic Cluster 1
1032 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Heavy Traffic Cluster 1
1545 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 52, ttl 52, src 80, ... Heavy Traffic Cluster 1
3337 10/07/14 10:55:54.2... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Heavy Traffic Cluster 1
1547 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 52, ttl 50, src 80, ... Heavy Traffic
1036 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 52, ttl 52, src 80, ... Heavy Traffic Cluster 3
1549 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 52, ttl 50, src 80, ... Heavy Traffic Cluster 3
1039 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 52, ttl 50, src 80, ... Heavy Traffic Cluster 1
1552 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 52, ttl 52, src 80, ... Heavy Traffic Cluster 1
3346 10/07/14 10:55:54.2... 103.243.222.11 192.168.43.178 TCP len 1146, ttl 52, src 8... Heavy Traffic Cluster 1
1555 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 52, ttl 52, src 80, ... Heavy Traffic Cluster 1
1044 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 52, ttl 50, src 80, ... Heavy Traffic Cluster 3
1557 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Heavy Traffic Cluster 3
1558 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Heavy Traffic Cluster 3
1559 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 1410, ttl 52, src 8... Heavy Traffic Cluster 3
1560 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 306, ttl 52, src 80... Heavy Traffic Cluster 3
Load DataSet
Preprocessing
Cluster
Naive Bayes
Naives Bayes Result : 0.91040164
K Nearest Neighbor 0


Network Classification
DATA
Id Date SourceIp DestinationIp Protocol Description Status Cluster
2560 10/07/14 10:55:51.7... 103.243.222.11 192.168.43.178 TCP len 1308, ttl 52, src 8... Heavy Traffic Cluster 3
3072 10/07/14 10:55:53.7... 103.243.222.11 192.168.43.178 TCP len 1410, ttl 52, src 8... Heavy Traffic Cluster 3
1538 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Heavy Traffic Cluster 1
3074 10/07/14 10:55:53.7... 103.243.222.11 192.168.43.178 TCP len 1094, ttl 52, src 8... Heavy Traffic Cluster 3
1539 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Heavy Traffic Cluster 3
1540 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Heavy Traffic Cluster 3
3076 10/07/14 10:55:53.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Heavy Traffic Cluster 3
1029 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Heavy Traffic Cluster 3
1541 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Heavy Traffic Cluster 1
2309 10/07/14 10:55:50.8... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Heavy Traffic Cluster 1
1030 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Heavy Traffic Cluster 1
2310 10/07/14 10:55:50.8... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Heavy Traffic Cluster 1
1031 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Heavy Traffic Cluster 1
2311 10/07/14 10:55:50.8... 103.243.222.11 192.168.43.178 TCP len 1252, ttl 52, src 8... Heavy Traffic Cluster 1
1032 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 40, ttl 50, src 80, ... Heavy Traffic Cluster 1
1545 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 52, ttl 52, src 80, ... Heavy Traffic Cluster 1
3337 10/07/14 10:55:54.2... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Heavy Traffic Cluster 1
1547 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 52, ttl 50, src 80, ... Heavy Traffic
1036 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 52, ttl 52, src 80, ... Heavy Traffic Cluster 3
1549 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 52, ttl 50, src 80, ... Heavy Traffic Cluster 3
1039 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 52, ttl 50, src 80, ... Heavy Traffic Cluster 1
1552 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 52, ttl 52, src 80, ... Heavy Traffic Cluster 1
3346 10/07/14 10:55:54.2... 103.243.222.11 192.168.43.178 TCP len 1146, ttl 52, src 8... Heavy Traffic Cluster 1
1555 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 52, ttl 52, src 80, ... Heavy Traffic Cluster 1
1044 10/07/14 10:55:47.3... 103.243.222.11 192.168.43.178 TCP len 52, ttl 50, src 80, ... Heavy Traffic Cluster 3
1557 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Heavy Traffic Cluster 3
1558 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 40, ttl 52, src 80, ... Heavy Traffic Cluster 3
1559 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 1410, ttl 52, src 8... Heavy Traffic Cluster 3
1560 10/07/14 10:55:48.7... 103.243.222.11 192.168.43.178 TCP len 306, ttl 52, src 80... Heavy Traffic Cluster 3
Load DataSet
Preprocessing
Cluster
KNN
Naives Bayes Result : 0.91040164
K Nearest Neighbor 1.0299518390617801

Conclusion and Future Works

This paper proposed a new network Network Congestion Control based on K-means clustering and different classification techniques. This paper performs clustering and features extraction to extract some relevant feature like Src,_addr, Src_port, Dst_addr, Dst_port and so on; which will be used in classification and the effective feature extraction help improving the classification accuracy and network performance. We perform the classification based on Naïve Bayes, K-means by incorporating the features exacted. The classification performance of all techniques are measured and presented in result analysis phase. The classification accuracy among Naïve Bayes, K-means are demonstrated that among all these decision Tree has higher classification accuracy than all others.

References:

[01] R.S.AnuGowsalya, Dr.S.Miruna Joe Amali, **"SVM Based Network Network Congestion Control Using Correlation Information",** International Journal of Research in Electronics and Communication Technology (IJRECT 2014), Vol. 1, Issue 2 April - June 2014,

[02] Jun Zhang, Yang Xiang, Yu Wang, **"Network Network Congestion Control Using Correlation Information",** Parallel and Distributed Systems, IEEE Transactions on , vol.24, no.1, pp.104-117, Jan. 2013,

[03] R.S. ANU GOWSALYA, S. MIRUNA JOE AMALI, "**Naive Bayes Based Network Network Congestion Control Using Correlation Information",** International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 3, March 2014 ISSN: 2277 128X,

[04] Ms.ZebaAtiqueShaikh, Prof. Dr. D.G. Harkut, "**An Overview of Network Network Congestion Control Methods",** International Journal on Recent and Innovation Trends in Computing and Communication, February 2015, ISSN: 2321-8169, Volume: 3 Issue: 2,

[05] PallaviSinghal, Rajeev Mathur, "**State of the Art Review of Network Network Congestion Control based on Machine Learning Approach",** International Journal of Computer Applications (0975 – 8887) International Conference on Recent Trends in engineering & Technology - 2013(ICRTET'2013),

[06] G.Suganya, **"An Efficient Network Network Congestion Control Based on Unknown andAnomaly Flow Detection Mechanism",** International Journal of Computer Trends and Technology (IJCTT) – volume 10 number 4 – Apr 2014,

[07] Andrew W. Moore, **"Internet Traffic Classication Using Bayesian Analysis Techniques",** SIGMETRICS'05, June 6.10, 2005, Banff, Alberta, Canada. Copyright 2005 ACM 1-59593-022-1/05/0006

[08] M. Tamilkili, **"A Survey on Recent Network Congestion Control Techniques Using Machine Learning Methods",** International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 12, December 2013, ISSN: 2277 128X

[09] G. Rubadevi, R. Amsavenim, **"A Novel Hybrid Aggregated Classifier For Internet Network Congestion Control",** International Journal of Computer Engineering and Applications, Volume VII, Issue II, Part II, August 14

[10] Bin Hu, Yi Shen, **"Machine Learning Based Network Network Congestion Control: A Survey",** Journal of Information & Computational Science 9: 11, October 1, 2012, 3161–3170.