uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Ali Mumtaz

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.Sc. (E-Business)

GRADE / DEGREE

Faculty of Graduate and Post Doctoral Studies

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Window-Based Stream Data Mining for Classification of Internet Traffic

TITRE DE LA THÈSE / TITLE OF THESIS

Dr. Bijan Raahemi

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Dr. Morad Benyoucef

Dr. David Wright

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

uOttawa

L'Université canadienne
Canada's university

WINDOW-BASED STREAM DATA MINING FOR CLASSIFICATION OF INTERNET TRAFFIC

By

Ali Mumtaz

Thesis Submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements

For the degree
MASTER OF SCIENCE (ELECTRONIC BUSINESS TECHNOLOGIES)

School of Information Technology & Engineering- Telfer School of Management
Faculty of Graduate and Post Doctoral Studies
UNIVERSITY OF OTTAWA

Thesis Supervisor
Dr. Bijan Raahemi

Your file Votre référence
ISBN: 978-0-494-41676-1
Our file Notre référence
ISBN: 978-0-494-41676-1

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Acknowledgements

First, I wish to thank Almighty God for bestowing the wisdom and courage that enabled me to accomplish my goals. My thesis supervisor Dr. Bijan Raahemi provided an excellent leadership, guidance and affection that helped me to complete my program of study with full dedication and peace of mind. Bijan involved me in this research problem in the first place. He taught me how to conduct a literature review, write a proposal, develop methodology and remain persistent. He provided prompt criticism, comments and solutions during all the stages of my research.

A special thanks for Dr Dominique Ferrand (Director of the E-Business program) for providing positive and constructive criticism during my thesis proposal stage. He taught me Research Methodologies and helped me to better redefine my research problem. I am also indebted to my teachers at the University of Ottawa: Dr. James Bowen, Professor Patricia Stirbys, Dr. Hamid Mehrvar, Moiz Syed and Hurol Sahin.

I would like to thank the members of my thesis committee, Dr. David Wright and Dr. Morad Benyoucef for accepting to review and comment on the thesis.

Let me say thanks to Monique Walker (Coordinator, High-tech Satellite Programs) for always answering my questions. She helped me a lot right from my initial application and up to the completion of the program.

University of Ottawa provided me the scholarship and let me say thank you to the people at UOttawa for confidence in me and supporting my research and academic activities.

My parents of course always pray for my success; my brother Ahmed and sister Nadia were always available to encourage me. I would like to appreciate the patience of my wife Uzma, being always happy and supportive during my stay at the University. My son, Shehryar and daughter Sofia supported me by allowing me to study in a peaceful environment at home.

ABSTRACT

Accurate classification of Internet applications is a fundamental requirement for network provisioning, network security, maintaining quality of services and network management. Increasingly, new applications are being introduced on the Internet. The traffic volume and patterns of some of the new applications such as Peer-to-Peer (P2P) file sharing put pressure on service providers' networks in terms of congestion and delay, to the point that maintaining Quality of Services (QoS) planned in the access network requires the provisioning of additional bandwidth sooner than planned. Peer-to-Peer applications enable users to communicate directly over the Internet, thus bypassing central server control implemented by service providers and poses threats in terms of network congestion, and creating an environment for malicious attacks on networks. One key challenge in this area is to adapt to the dynamic nature of Internet traffic. With the growth in Internet traffic, in terms of number and type of applications, traditional classification techniques such as port matching, protocol decoding or packet payload analysis are no longer effective For instance, P2P applications may use randomly selected non-standard ports to communicate which makes it difficult to distinguish from other types of traffic only by inspecting port number.

The present research introduces two new techniques to classify stream (online) data using K-means clustering and Fast Decision Tree (FDT). In the first technique, we first generate micro-clusters using k-means clustering with different values of k. Micro clusters are then merged into two clusters based on weighted averages of P2P and NonP2P population. This technique generates two merged clusters, each representing P2P or NonP2P traffic. The simulation results confirm that the two final clusters represent P2P and NonP2P traffic each with a good accuracy.

The second technique employs a two-stage architecture for classification of P2P traffic, where in the first stage, the traffic is filtered using standard port numbers and layer 4 port matching to label well-known P2P and NonP2P traffics, leaving the rest of the traffic as "Unknown". The labeled traffic generated in the first stage is used to train a Fast Decision Tree (FDT) classifier with high accuracy. The Unknown traffic is then applied to the FDT model which classifies the traffic into P2P and NonP2P with high accuracy. The two-stage architecture, therefore, not only classifies well-known P2P applications, it also classifies applications that use random or private (non standard) port numbers and can not be classified otherwise. We performed various experiments where we captured Internet traffic at a main gateway router, pre-processed the data and selected three most significant attributes, namely Packet Length, Source IP address and Destination IP address. We then applied the proposed technique to three different windows of records. Accuracy, Specificity and Sensitivity of the model are calculated. Our simulation results confirm that the predicted output represents P2P and NonP2P traffic with accuracy higher than 90%.

CHAPTER 1 – INTRODUCTION

Peer-to-Peer (P2P) applications enable users to communicate directly over the Internet, bypassing central server control implemented by service providers. 'It is a network architecture in which each computer has equivalent capabilities and responsibilities' [2]. P2P network contradicts the traditional client/server architecture in which one or more computers are dedicated to serve the others. In the traditional network one computer is termed as 'server' and clients are users of services rendered by a server. In P2P environment every computer is a client and server [2] without any computer functioning as a central server. It provides a decentralizing computing concept and all computers act as equal or peers. Furthermore, decentralization stimulates dispersion of content among peers thus eliminates a server assigned to host services and resources. In [1] authors mentioned that P2P traffic and its characteristics have changed the original assumptions under which the data networks were designed. They further elaborated that 'P2P traffic is more symmetric; and P2P traffic is less bursty, which makes it difficult to take advantage of statistical multiplexing'. Further, [2] grouped P2P into five distinct types such as; Instant messaging (windows messenger), File sharing/Swapping (Kaza, Napster), Distributed search engines (Open cola), Group collaboration (Net meeting, Intralinks), Distributed computing (United devices, Entropia).

1.1 Problem Definition

P2P applications bypass central server control implemented by service providers and pose threats in terms of network congestion, inability to enforce content ownership and creating

an environment for malicious attacks on networks. P2P applications may use randomly selected non-standard ports to communicate and consume network resources of service providers without creating additional revenues [3]. The classification method developed in the present work will be used to classify peer-to-peer traffic and can be adopted by Internet Service Providers (ISP), corporate wide network providers and data service providers in order to detect peer-to-peer traffic. Once the traffic is classified as P2P it helps service provider to implement their control policy in terms of restricting bandwidth or billing customers for extra usage.

A recent study has demonstrated that P2P traffic on the Internet is 70% of total broadband traffic [4]. The volume and traffic patterns are putting pressure on service providers' networks in terms of congestion and business models. For example, Quality of Service (QoS) based planned over subscription factors are under pressure in the access network causing the provisioning of additional bandwidth sooner than planned, and at backbone links P2P requests from outside the service provider's network for files residing on subscriber computers can result in massive amounts of outbound traffic, which is of no benefit to network subscribers.

1.2 Research Objectives

With the increased use of the Internet as a medium to communicate and to offset the costs of communications infrastructure, we witness a steady development of a problem – classification of traffic. Efforts were made to classify the traffic for various applications including classification of peer to peer traffic mostly faced by Internet Service Providers (ISPs) and corporate wide networks. We see that most of the work classifies traffic based on offline techniques. There are, however, many applications that require handling data in the

form of stream, such as stock tickers, point of sales systems, web data flow analysis and network intrusion detection. As such, stream (online) data mining techniques have gained much attention recently. Both techniques (offline and online data mining) have their own applications and we cannot deduce that stream classification will jeopardize offline techniques. As we see in literature surveys there has been an effort to develop techniques to address a specific problem for instance to solve the memory limitation problem or just comparing different algorithms. It is further found that efforts have been made to develop a technique utilizing window based phenomenon. This technique is still at an infancy stage and is a new research area. Application of window based technique to address classification of data streams poses several challenges, some of which are addressed in the present work.

The objective of this research are (a) to develop techniques for capturing data packets off the stream, (b) to apply window-based online data mining techniques , both clustering and classification, to classify P2P traffic, and (c) to implement the proposed algorithms, run simulations, and analyze the results to measure the accuracy and performance of the online classifiers.

1.3 Organization of the Thesis

Chapter 2 provides details about the related research and gives an understanding of work being done by other researchers. Chapter 3 describes the K-means clustering technique that was applied to classify the Internet traffic. This chapter also contains the pre-processing, modeling and simulation results related to the clustering technique. Chapter 4 provides an insight of our new technique named as "Two-Stage Classifier with Fast Decision Tree" and

subsections present the methodology for preprocessing and modeling. The chapter is concluded with discussion on simulation results. Chapter 5 is about the implementation of the concept and methodology. It mainly provides scripts and partial java code to work in an orderly manner. The thesis is concluded in chapter 6 with discussion on future directions and suggestions for development of an application that can be applied in various business models.

CHAPTER 2 – RELATED RESEARCH

Classification of P2P traffic has gained much attention in both academic and industrial research communities, and various solutions have been developed for P2P traffic classification. A popular approach is the TCP port based analysis where tools such as Netflow[5]and cflowd[6] are configured to read the service port numbers in the TCP/UDP packet headers , and compare them with the known (default) port numbers of the P2P applications. The packets are then classified as P2P if a match occurs. Although P2P applications have default port numbers, newer versions allow the user to change the port numbers, or choose a random port number within a specified range. Hence, port based analysis becomes inefficient and misleading.

Recently, researchers have considered the behavioral and statistical characteristics of the internet traffic to classify P2P applications. In[7]S. Zander, T. Nguyen, and G. Armitage proposed a framework for IP traffic classification based on a flow's statistical properties using an unsupervised machine learning (ML) technique. In this approach, the authors first classified packets into flows according to IP (Src, Des) addresses, port numbers and protocol. Then they used the attributes: packet inter-arrival time, packet length mean and variance, flow size, and duration to build the ML classifier model using the auto class classification system [8]. In[9] their findings demonstrated that current popular methods such as port number and payload-based identification exhibit a number of shortfalls and proposed that an alternative is to use machine learning (ML) techniques and identify

network applications based on per-flow statistics, derived from payload-independent features such as packet length and inter-arrival time distributions.

The concept of sliding window was adopted by Aggarwal, Jiawei, Wand and Yu in their classification model [10]. The model uses a sliding window which is equal to the history of the entire data stream. Their methodology is different in a way that it adapts to the fast evolving data streams. For the testing of on-demand classifier they divided the training streams into two parts; one for class specific statistical maintenance of microclusters and second for testing the nature of the horizon for getting the best classification accuracy. Later, they integrated these two portions to build a classifier. One important point that they demonstrated is that the classifier must be able to adjust rapidly to the changes so that accurate classification results can be provided to the user on-demand. This is the most important aspect of any online data stream classifier. If the classifier is not agile it will provide erroneous results. In my view the fundamental difference between an offline classifier and online is the agility factor of the classifier. It was also mentioned that "process of classification should be viewed as a continuous process in which the training stream and test stream are simultaneously generated by the underlying process." Adoption of "snapshots" technique allowed them to store the behavior of microclusters at different moments of time. They used geometric time frame to store snapshots at different level of granularity depending upon the recency. It was demonstrated that On-demand classifier has higher classification accuracy in comparison with the simple one pass classification algorithms. It has good scalability in terms of dimensionality and class labels. The third important result is that their model is space efficient.

The major strength of this algorithm is division of tasks into two parts and later merging to get a result. The agility of classifier makes it a good fit for online classification. It adjusts to the change in the flow. This model does not depict the methods to extract flows off the stream and an automatic mechanism to transform and input data to classifier.

Pedro Domingos and Geoff Hulten [11] utilized Very Fast Decision Tree (VFDT) to mine the continuous web access data from the University of Washington main campus. VFDT is a decision tree learning system that utilizes Hoeffding trees. VFDT uses Information gain (Entropy) or Gini index to evaluate the attributes. Data mining systems have limitations of memory, time and samples. This research was conducted in 2000 and at that time the authors observed that available algorithms were not able to cope with high stream traffic and most importantly, data traffic generated by ISPs, Banks, Telecommunication services which are high in volume. They introduced the concept of "One pass seeing" of data in order to solve problems of ever increasing data on the Internet. They described training and test data as "examples" and these examples were used to mine data in less time than inputting from the source. It can learn from each example by seeing it only once. Accuracy achieved by applying VFDT was 72.7%. It also does not require storing examples and therefore, solves problem of solving a memory limitation issues. This work does not take account of rapidly changing characteristics in data streams. However, it provides a good concept of addressing memory limitation issues inherent with stream data mining.

In 2001, VFDT based mining was improved by extending it with Concept-Adapting Very Fast Decision Tree Learner (CVFDT) [12]. It contributed by addressing the issue of

continuously changing data streams. This algorithm stays current by growing an alternate sub tree based on new data and whenever an old data becomes obsolete, it then, replaces the old data with new one once it becomes accurate. It works similar to VFDT but the major difference is that it keeps on reapplying VFDT algorithm every time new example is arrived. CVFDT keeps decision tree up-to-date by moving window of examples. Empirical results show that accuracy achieved was 72.3% as compared to VFDT, which was at 72.7%. This work focuses on solving the problem of time constraints as every time when new example is introduced the system does not start from scratch but it relies on updating the sufficient statistics at its nodes. This work does not address two major issues; drift in accuracy as we change the window size and secondly, time slots for extracting data to create an example.

An effort was made to extend VFDT in to two directions [13]; ability to deal with numerical attributes and ability to apply Naïve Bayes classifiers in tree leaves. It used Information Gain as the evaluation function. In this work major contribution is an application of VFDT using Naïve Bayes classifier and based on numerical attributes. A new term Very Fast Decision Tree based on Classifier Naïve Bayes (VFDTcNB) was introduced. Comparison was made with traditional C4.5 algorithm. Evaluation was based on three dimensions; error rate, learning time and tree size. Empirical results show that VFDTcNB out-performs C4.5. It was also determined that VFDTcNB is very fast in the training phase. It scans the entire training set once and time needed to process each example is negligible. In application phase there is an overhead due to use of naïve Bayes at leaves. Interestingly, on a different training set, C4.5 had an accuracy of 68.6% while VFDTcNB was at 62.4%. VFDT is an incremental algorithm and its variant VFDTcNB produces stable and predictive models. There are still few questions and limitations that need further work i.e., performance analysis with varied and drifted set. It is also required to use this method on particular application like Peer-to-

Peer traffic classification or any other Banking application. As we see that accuracy of classifier decreases as we change the dataset.

As we move further, we see that initial research phase was focused on overcoming memory, time and computation issues. Furthermore, effort was done on selecting the best algorithm to apply on stream data. We see a tremendous growth in data with an introduction of integration techniques and use of information sharing applications. It was realized that in Data Mining we cannot rely on just one-pass algorithm as we cannot predict the accuracy of results with one-pass methods. Ruoming Jin and Gagan Agrawal revisited one-pass problem and presented a two pronged approach [14]. Firstly, Numerical Pruning Approach (NIP) paved a way to efficiently process numerical attributes and secondly, with the use of gain function entropy sample size was reduced to 37%. Basis of NIP is 'to partition the range of a numerical attribute into intervals and then use statistical tests to prune these intervals'. In order to reduce the sample size authors used a theorem "Multivariant Delta". This is important in data streams that we can achieve accuracy by reducing the amount of samples required. In this work there is need to demonstrate the accuracy achieved using reduced samples if we to consider inter-arrival times of packets.

Thomas Karagiannis, Konstantina Papagiannaki and Michalis Faloutsos presented a very interesting concept and named it "BLINC" [15]. BLINd Classification (BLINC) approach is two fold, which first shifts the focus from classifying individual flows to associating Internet hosts with applications, and then classifying their flows accordingly. In their research authors demonstrated two key features of BLINC: Firstly, classification of hosts by capturing the fundamental patterns of their behavior at the transport layer. Secondly, they defined

methodology at three levels of host behavior: (i) the social level, (ii) the functional level, and (iii) the application level. The results were very encouraging and BLINC classified approximately 80%-90% of the total number of flows in each trace with 95% accuracy. Another important performance was demonstrated that identifies malicious behavior or previously unknown applications without having a prior knowledge or port specific information.

Although BLINC introduces new approach for traffic classification but it associates Internet hosts with applications. Laurent Bernaille, Renata Teixeira, Ismael Akodjenou
Augustin Soule, Kave Salamatian [16] argues that BLINC instead of studying TCP (or UDP) flows individually "All classification techniques that use flow statistics can only identify the nature of a flow when the flow is stopped. BLINC has to gather information from several flows for each host before it can decide on the role of a host generated by specific hosts. BLINC is able to accurately associate hosts with the services they provide or use (application server, web client, etc.). However, it cannot classify a single TCP flow". They proposed a traffic classification mechanism that works in two phases. First phase is an offline learning phase and the second phase performs online classification. During the training phase, training data is used to cluster TCP flows based on common behavior. Traffic classification stage uses these classes to determine the application associated with each TCP flow. Their classification method is aimed at identifying the application based on TCP flow as early as possible. They presented the classification accuracy based on just first five packet flows. Following table reflects the level of accuracy achieved:

Application	Accuracy
Edonkey	84.2%
ftp	87%
http	99%
Kazaa	95.24%
nntp	99.6%
Pop3	0%
Smtp	44.4%
Ssh	96.92%
https	81.8%
pop3	89.8%

Table 1: Accuracy level achieved for different applications.

They presented a structure of an online classifier as shown in Figure 1.

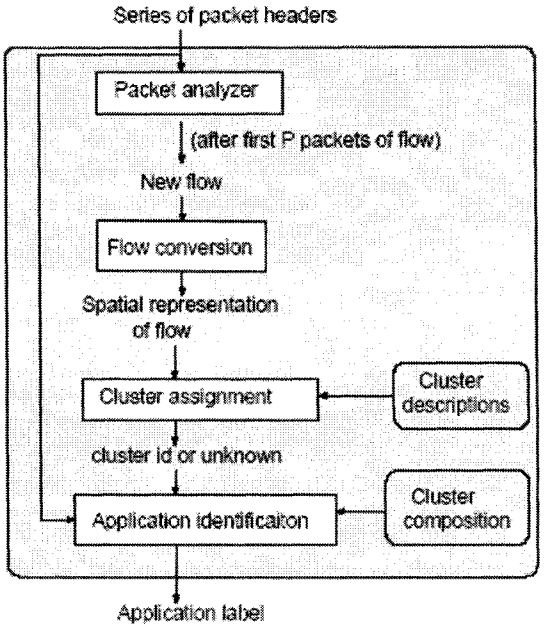


Figure 1: Structure of an online classifier Source*: Ref [16]*

They described the working of online classifier as "...Our classifier takes as input the series of packet headers for both directions of an edge link. A packet analyzer extracts the 5-tuple (protocol, source IP, destination IP, source port, destination port) and the packet size. The analyzer filters out control traffic (the three packets of the TCP handshake and ACKs without payload) and stores the size of every packet in both directions of the connection. When it [Packet analyzer] has the size for the first P packets of the connection, it sends this information to the flow conversion module, which maps the new flow to a spatial representation. Then, the cluster assignment module searches all the cluster descriptions to find the best fit for the new flow and the application identification module selects which application is the most likely for the flow given the set of applications that compose the cluster"[16]

Their method has few limitations such as:

Multi-homed networks: This relates to the multiple connections to the Internet. This technique is to be applied to all individual access links.

Packet order: Out of order packets impact the quality of classification. They observed that 4% of the TCP flows had an out of order packet within the first five data packets of the flow.

Sampling: The accuracy of model degrades quickly under packet sampling since on high speed links, monitors cannot collect all packets.

Applications with similar behaviors: If two distinct applications start by exchanging five packets of approximately the same size, then it is classified both with the same label.

An effort was done to profile Internet backbone traffic based on behavior models and applications [17]. This classifies Internet traffic flows and is related to present work in a way

that it tries to classify online traffic. In this study packet header traces were collected on Internet backbone links in a tier-1 ISP. These traces were aggregated into flows based on the four attributes; source IP address, destination IP address, source port and destination port. The objective was to profile the traffic based on communications patterns. They created four clusters based on these four features. The first two clusters based on source IP and destination IP addresses clusters represent host behaviors while source port and destination port clusters represent service behaviors. In the second stage structures among the clusters were discovered. They first developed a behavior classification scheme based on observed similarities or dissimilarities in communication patterns. The model was validated using data collected from a variety of links at the core of the Internet. They demonstrated that popular services and applications and certain type of malicious activities exhibit stable and distinctive behavior patterns. Oveissian, Salamatian, Soule and Taft presented a different approach [18]. They proposed a model based on applications of statistical inference techniques for identifying classes as well as deciding the likelihood that a particular flow belongs to a specific class. Internet traffic's aggregated flows are complex and hard to characterize. They focused on the macroscopic behavior of flows and classify traffic into small number of classes. In their work they focused on highly aggregated flows and developed a very general method for classifying them. They applied two classification methods; Hidden Markov Model (HMM) and Gaussian Mixture Model (GMM) to the 22,000 active flows in the network. They chose three values for the window size; 12 (60mins), 18 (90mins) and 24(120mins). The result of proposed model is a stable and useful classification with three classes of flows.[19] proposed a method for mining emerging patterns in data streams and achieved upto 10% increase in accuracy as compared to the other methods. They applied sliding window concept in building a model using C4.5 decision tree. Similarly, an effort was

made in [20] works by applying a Naïve Bayes estimator to categorize traffic by application. Furthermore, [21] extended the work by using the discriminators in flow-based classification. A different approach was introduced in [22] where their method classifies the current measured traffic to a "best-fit" model selected from a library of candidate models using statistical estimation techniques. Furthermore, [23] addressed the requirements of business processes for an online frequent pattern mining by introducing a method based on Block Depth First Search (BDFS). It provided some of the solutions to real-time frequent pattern mining.

Cost-Efficient Mining Techniques for Data Streams was presented by Mohamed, Shonali, Arkady[24]. They proposed an algorithm output granularity as a solution for mining data streams. Algorithm output granularity is the amount of mining results that fits in main memory before any incremental integration. The algorithm has two main components. The first component is the resource aware Rate Adaptation (RA) that uses the data adaptation techniques to catch up the high speed data stream and at the same time to achieve the optimum accuracy according to the available resources. Second component is One- Look clustering algorithm, named as Light-weight clustering algorithm (LWC). It was identified that we have only one look at the online-streams and algorithm output granularity must be able to identify the snap of stream [25] and looked at sliding windows from the perspective of preventing stale data from influencing analysis and statistics. They also mentioned that random samples can also be used as a summary structure where a small sample represents the characteristics of the data set. [26] presented a new approach for mining frequent item sets on data stream that allows users to issue requests for frequent item sets over an

arbitrary time interval. [27] work illustrates an approach for identifying the P2P application traffic through application level signatures. [28] worked on almost similar technique and proposed a method to ensure Quality of Service (QoS) on networks based on statistical application signatures. Troyano, Ruiz and Riquelme presented a scalable learning algorithm to classify numerical and time-changing data streams. They introduced classification learning algorithm based on decision rules and prototypes.

[29] proposed est-win method which is based on sliding windows. This method eliminates historical data packets and replaces it with new data items. Authors are of the view that previous knowledge gained is not very useful as stream data changes over time. The size of a sliding window defines the desired life-time of information in a new processing and they called it 'transaction'. Accordingly, only recently generated transactions in the range of the window are considered to find the recently frequent item sets of a data stream. Most recently, [1] classified P2P traffic based on decision tree and IP layer attributes. By detecting communities of peers, they achieved classification accuracy of higher than 98%. Their model relies on four attributes at the IP layer such as "Protocol", "Length", "Source IP" and "Destination IP".

Table-2 shows a summary of different techniques developed for stream data mining.

Pub.Year	Topic	Authors	
2000	Mining high-speed data streams	Domingos, P., & Hulten, G.	Addressed problem of constructing accurate decision tree models from data stream. Very Fast Decision Tree (VFDT) algorithm was presented which is used based on "one pass seeing". Accuracy 72.7%
2001	Mining time-changing data streams	Hulten, G., Spencer, L., & Domingos, P.	Concept-Adapting Very Fast Decision Tree Learner (CVFDT).Algorithm stays current by growing an alternate sub tree based on new data and whenever an old data becomes obsolete, it replaces the old data with new one once it becomes accurate. Accuracy 72.3%
2005	Profiling Internet backbone traffic: Behavior models and applications	Xu, K., Zhang, Z. L., & Bhattacharyya, S.	Presented methodology for building comprehensive behavior profile of Internet backbone traffic in terms of communication patterns of end-hosts and services. Four attributes were used to build four clusters(SrcIP, DstIP, SrcPrt, DstPrt). In the second stage structures among clusters were discovered
2006	Traffic Classification on the fly	Bernaille, L., Teixeira, R., Akodkenou, I., Soule, A., & Salamatian, K.	Presented two phases; offline learning phase and online classification phase. During training phase, training data is used to cluster TCP flows based on common behavior.
2006	A framework for on-demand classification of evolving data streams	Aggarwal, C. C., Han, J., & Jianyong, W.	Sliding Window was presented using snap shotsin which simultaneous training and test streams are used for dynamic classification of data sets. The training model quickly adapts to the changes of the data stream. Divided the training streams in to two parts; statistical maintenance of micro clusters and testing the nature of the horizon for getting the best classification accuracy. Later, both portions were integrated to build a classifier
2007	Accurate classification of the internet traffic based on the SVM method	Li, Z., Yuan, R., & Guan, X.	Support Vector Machine (SVM) method to train 7 classes of applications of different characteristics. Data was captured from a campus network. Discriminator selection algorithm was used to select best combination of attributes. There are about 200 features in a single flow of Internet.
2007	Peer-to-Peer IP traffic classification using Decision Tree and IP layer attributes	Raahemi,B.,Hayajneh,A.,Ra binovitch,P.	Built several models using a combination of various attribute sets for different ratios of P2P to non-P2P traffic in the training data. It was observed that accuracy of the model increases when attributes Source IP and Dest IP address are included in model. Community of Peers were detected with accuracy higher than 98%.

Table 2: summary of major stream data mining techniques presented in the literature review

In the above table researchers presented different techniques to address stream data mining problems. The most important contribution is the development of VFDT and CVFDT algorithms that mine the stream data in one pass. Also, the sliding window approach developed in [10] gives a new dimension to the stream mining problem.

These techniques also have their own limitations. Some of them require deep packet inspection which may not be possible if privacy is a major concern. Some are memory or computation intensive. In this thesis, we introduce classification techniques which are relatively fast and accurate, and at the same time, do not require deep packet inspection.

CHAPTER 3 – MICRO-CLUSTERING USING K-MEANS CLUSTERING

Clustering is a grouping of data items based on similarities [30].It is similar to classification in that data are grouped but unlike classification, groups are not defined. Clusters are set of like elements. Elements from different clusters are not alike. The distance between points in a cluster is less than the distance between a point in the cluster and any point outside it. The classification techniques presented in [1]are based on supervised learning, and as such, they utilize labeled data to classify P2P traffic. Online streams are continuous flow of data and traffic dynamics change with respect to time. Pre-labeled data is not suitable for an online classifier as each new window may introduce different traffic characteristics. Our first approach is based on K-mean clustering technique so as to avoid rely on labels because (a) classes are not predefined (b) labeling introduces delay in processing thus increasing cycle turn around time (c) port based labeling relies on prior knowledge of standard and non-standard ports, and some P2P applications change the port numbers randomly. K-means clustering is an unsupervised learning algorithm [31]. It classifies or groups objects based on attributes/features into K number of group. K is positive integer number. The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid.

3.1 Attribute selection

Our work relies on three attributes. We generate two data sets each consisting of mixture of P2P and Non P2P packets. The first dataset consists of 22593 records and the second set has 27292 records. In the first experiment, seven attributes were used; Packet inter-arrival

time, ID, Packet length, Source IP address, Destination IP address, Source Port and Destination port. It was observed that the K-Means clustering algorithm generated clusters based on port numbers. Since most of the P2P applications are capable of using random or user defined ports [1], we do not rely on port numbers. We therefore, eliminated source and destination port numbers. Since we are not doing sequence analysis, packet inter-arrival time and ID do not play significant role in our data analysis. As such, the clustering model is built with only three attributes; Source IP, Destination IP and Packet length. Packet length plays a significant role in cluster creation as P2P clusters have lower average packet length as compared to Not P2P clusters.

We use Weka application software [32] to analyze and select most suitable attributes. Weka is an open source application software developed at the University of Waikato, New Zealand. It is a collection of machine learning algorithms for data mining tasks. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization

Table 3 shows list of attributes:

S/No	Attribute	Abbreviation	Description
1.	Time	Time	Time stamp of each packet
2.	Packet length	PL	Packet length ranges from 40 to 1500 bytes
3.	Average packet length	APL	Derived attribute based on packet length
4.	Source IP address	SrcIP	Sender's 32 bits IP address
5.	Source port	SrcPrt	16 bits source port number

6.	Destination IP address	DesIP	Receiver's 32 bits IP address
7.	Destination port	DesPrt	16 bits destination port number
8.	Protocol	Proc	Protocol in the data filed TCP is 6 UDP is 17
9.	Standard Deviation	StDev	Derived attribute based on packet length.

Table 3: Attributes and description

3.2 Data Preprocessing

Using Windump utility we generate datasets from the files captured at the campus router. Internet data is in binary format and in order to perform analysis tasks it is transformed into readable format. Following steps are followed to perform preprocessing task:

1. *Windump utility* is an open source program that transforms captured Internet traffic into readable format. It works under Command Line Interface (CLI) and following script is used:

Windump.exe –nvr <source file> -c<number of records> > Target File.txt

Where switch n = Do not convert IP addresses into names

v =Produce verbose output

r = read source file

c = count packets

File.txt = text file generated

2. File.txt is than inserted in PHP program and delimiters are inserted after each attribute.
3. Data file generated as a result of step-2 is than opened in Microsoft excel.
4. Data is cleaned and only attributes described in table-1. are retained.

5. Cleaned file is then inserted into Weka for analysis and model building. Weka supports data files based on Microsoft Excel, Comma Separated Variable (CSV) and Attribute-Relation File Format (ARFF).

We adopt the method [1] to preprocess data, discard non-informative attributes and unary fields; the sample extracted file is shown as follows :

```
04:38:15.298019 IP (tos 0x0, ttl 105, id 48442, offset 0, flags [DF], proto: TCP (6), length: 1500)
81.66.28.51.6346 > 137.122.70.23.4921: . 3732432837:3732434297(1460) ack 1274371307 win 65081

04:38:15.298116 IP (tos 0x0, ttl 127, id 34354, offset 0, flags [DF], proto: TCP (6), length: 40)
137.122.73.47.2075 > 216.120.241.17.80: ., cksum 0xc292 (correct), ack 2920 win 65535
```

The initial inspection of the spreadsheet with a simple statistic analysis showed that:

- Very few records (519 = 0.7%) have missing or scrambled data, so these entries were discarded. (Cleaning)
- The "tos", "flag" and "offset" fields have almost one single value for all records (unary attributes), and "ttl", "win", "cksum" and "Sequence number" contain no information to differentiate records (non-informative attributes), so these entries have non-distinguishing information so they were all dropped off also. (Filtering)
- The "Arrival time", "Identification", "protocol", "Packet Length", "Source IP", "Destination IP", "Source Port" and "Destination Port" fields have different values that can be utilized to discriminate records (informative attributes). Accordingly, they were retained during cluster creation stage.

3.3 Selecting K

K-means clustering requires deciding on values of K before we start. We generated 2 data sets having various population of P2P and NonP2P traffic. As shown in Fig-2, maximum accuracy is achieved by k=6. The accuracy drops as we increase the value of k. This graph is based on experiments conducted on 21593 records and three attributes; (packet length, Source IP and Destination IP). We repeated the experiment with 27292 records, and this time, maximum accuracy was achieved with k=9. We concluded that as we increase the number of records, higher value of k is more desirable. However, we should also keep in mind that the higher value of k means we need more memory and processing time to create and merge the k micro clusters.

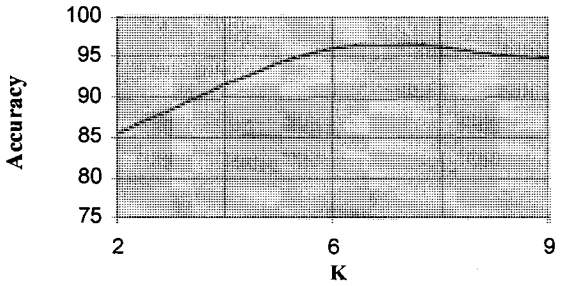


Figure 2: Accuracy Vs Number of Clusters (K)

3.4 Building the Merged Micro-Cluster Model

We present a two-stage method to solve clustering problem of P2P traffic. In the first stage, the input traffic is classified based on known port numbers using the technique introduced in [1] that is packets with port numbers less than 1024 are classified as Non P2P and packets with port numbers higher than 1024 are sent to the second stage. In the second stage, first micro clusters are created by selecting the value of K and a random seed.

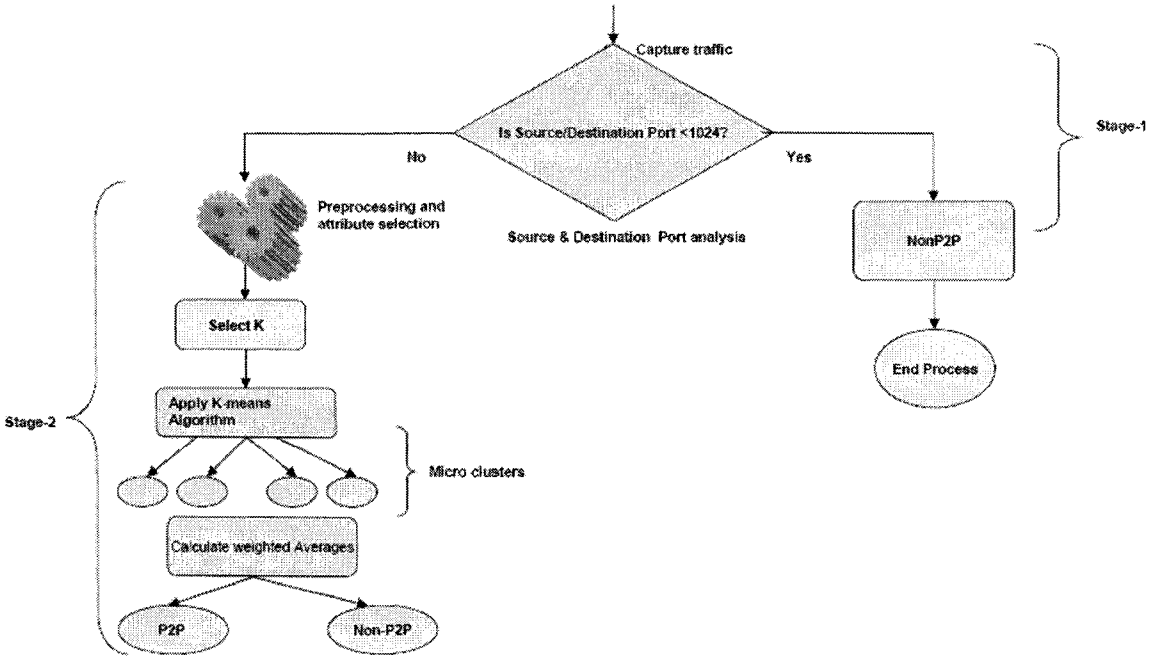


Figure 3: Two stage clustering model

We use Weka to generate clusters and conducted 9 experiments on two datasets and different combination of attributes. Each data set is split as 66/34; where 66% of the data is used to build the model and 34% is to test the model. We name micro clusters as C_0 .....C_k; where C_0 is the first micro-cluster and C_k is the last. Micro clusters contain varied P2P and Not P2P population. The weighted average is calculated based on the formula:

$$P_k = (P2P \text{ packets in micro cluster } k \ / \ Total\ P2P) \quad (1)$$

$$NP_k = (NonP2P \text{ packets in micro cluster } k \ / \ Total\ NonP2P) \quad (2)$$

We calculate weighted averages of P2P and NonP2P population in each micro cluster. If the population of any group (P2P or Non P2P) is dominant such as more than 50% we group them together and calculate average population of P2P and Non P2P packets.

Micro-clusters are created based on source IP address, destination IP address and packet length. K-means algorithm first build a model based on the training set and 66/34 training/testing data split. As we analyze each generated micro-cluster, we found grouping based on IP addresses. Micro clusters are then merged based on the following criteria. If P2P average is higher than NonP2P average in a given micro-cluster, we name such cluster as Cp. All Cps are then merged together called Merged_P. The same procedure is performed to merge micro-clusters having majority of NonP2P traffic to create final cluster Merged_NP. The pseudo code of the procedure is written below:

```
Merged_P = Null      /*initialize the cluster representing P2P
                        traffic*/
Merged_NP = Null   /*initialize the cluster representing NonP2P
                        traffic */
For i=1 to k       /*for every micro-clusters */
If Pi > NPi
Then
     Merge Ci with Merged_P
Else
   Merge Ci with Merged_NP
End
```

The merged cluster merged_P now represents the majority of P2P traffic, and the merged cluster Merged_NP represents NonP2P traffic. We calculate weighted average population based on formulas (1) and (2).

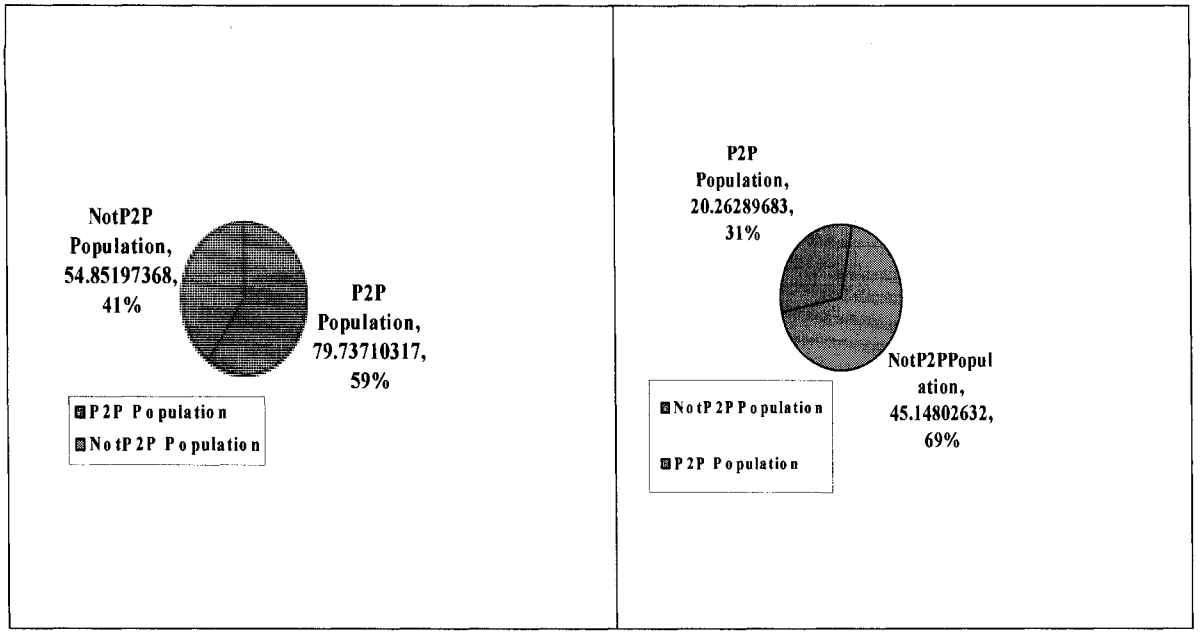
3.5 Simulation Results

Our experiment is based on 27292 records three attributes (packet length, Source IP, and Destination IP), and k=6. Table 4 shows distribution of P2P and NonP2P in all the 6 clusters. These clusters represent population distribution based on IP addresses.

Cluster	P2P (%)	NonP2P (%)
C0	91	9
C1	93	7
C2	90	10
C3	95	5
C4	17	83
C5	73	27

Table 4: Micro Clusters C0 to C6

Micro-clusters are then merged based on population. Figure 4 shows final clusters based on average population after we merge micro clusters. Our 2 final clusters represent P2P and Non P2P distribution in each cluster. P2P cluster has 59% of P2P population and 41% Non P2P population. In the second cluster NonP2P dominates P2P with 69% population.



P2P Cluster **NonP2P Cluster**

Figure 4: The two final merged clusters

Our analysis of two final clusters yielded that it still relies on labels during merger. During a stream communication we do not have a labeled traffic and new data items must be classified using unlabelled data. It chooses clusters based on three attributes Packet length, Source IP and Destination IP addresses. It was observed that the instances in each cluster represent a distinct packet length. For instance, clusters with higher P2P population have higher average packet lengths as compared to the ones with lower population of P2P traffic. In order to classify unlabelled data, we developed another method described in Chapter-4 which uses labeled data to train the classifier and then the model classifies unknown traffic as P2P or NonP2P.

CHAPTER 4 – TWO STAGE CLASSIFIER WITH FAST DECISION TREE

In [1], static data was used to classify the traffic. This technique is useful when there is no significant change in communication patterns and it relies on labeled data. In online stream communications discrete data flows and communication patterns change for different P2P and Non P2P applications. In [17], it was found that several most popular services and applications, as well as certain types of malicious activities, exhibit distinctive behavior patterns. We present a new two stage architecture to classify P2P and Non-P2P traffic based on knowledge of labeled data and then classifying unknown traffic based on trained model. In the first stage, known P2P and Non-P2P traffic are filtered out using a port filtering method on the basis of Known P2P and NonP2P port numbers as described in section 4.1.2. Once the traffic is classified as P2P, NonP2P and Unknown, the second stage is to build the classification model. For this purpose, classifier is trained/built using Fast Decision Tree (FDT) model (the training stage). The accuracy of the model is calculated and "Unknown" traffic is introduced as an input to the classifier. The Unknown traffic is not labeled and the classifier predicts which instance is most likely a P2P or NonP2P.

The model is kept in memory and the "Unknown" traffic is introduced to classify as P2P or Non-P2P. Furthermore, the model is constantly re-evaluated with the new window and is only rebuilt if accuracy drops below 90%.

4.1 Two stage classification model

In this section we present a new classification method for data streams named 'Two Stage Classifier with Decision Tree'. The two stage method is shown in figure 5 below:

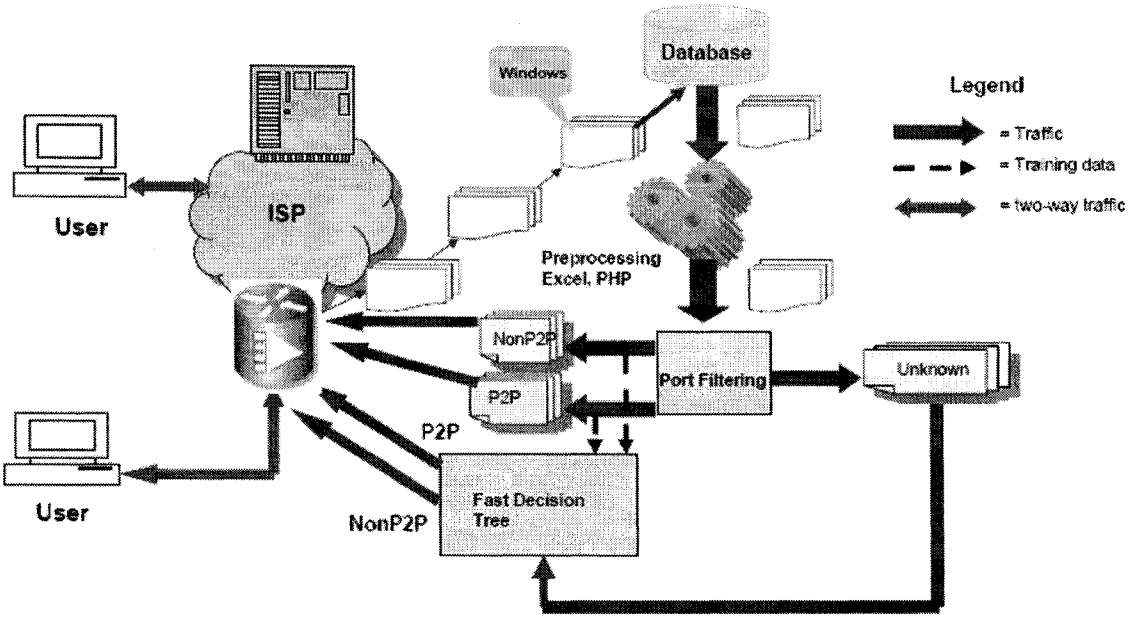


Figure 5: The two stage Classifier with Decision Tree

Architecture of Figure-5 addresses four challenges in stream data mining. First, it extracts the data and creates windows using TCPdump script described in subsection 4.1.1. Data streams generate continuous and high speed data items and therefore, it has not enough time to rescan the data or perform multiple scans. Data is extracted at different intervals using a controlled script and windows are processed using an application developed in KDD lab. This part solves the problem of extracting data on high speed links.

Second, the two stage architecture employs standard port filtering to specify well-known P2P and NonP2P applications based on their port numbers. This will generate an accurate pool

of training data that will be used in the second stage to train the FDT classifier. Furthermore, it eliminates the need to process the whole window which in turn improves the overall process time.

Third, due to the high speed characteristics of online streams, each window needs to be processed and classified as fast as possible. This in turn requires a faster algorithm to classify traffic than the coming window rate. It also trains the classifier based on examples generated as a result of port filtering. This makes sure that unknown traffic is classified on most recent instances.

Fourth, due to a continuous change in data stream characteristics, the classifier has to be constantly re-evaluated at periodic intervals. The requirement to re-evaluate is dictated based on accuracy produced by the classifier. We take the classified unknown traffic with the confidence level achieved during model analysis. If re-evaluation entails an accuracy of any thing less than 90%, a new model is built based on most recent windows. In [1] work average accuracy of the classifiers were above 90%, therefore, the model is kept at least 90% accurate. Re-evaluation of the model solves the memory, storage and CPU limitation problems as it works within available resources without compromising the process time and accuracy. It optimizes the memory space by keeping most recent model in memory and constantly re-evaluating the model. Classification result is also dependent on recently generated windows as it depicts most recent communication patterns so the model makes sure that classified output is within the acceptable accuracy range.

4.1.1 Window creation:

In [25], authors described that implementation of sliding windows on data streams is a natural method of approximation with attractive properties. One of the most important features of windows is that it emphasizes on the recent data as opposed to the older data thus suiting most of today's application.

We create windows of variable sizes, each size based on the number of records. The window size decision is based on available resources and requirement of the particular application. This technique is based on generating fixed windows based on number of records of recent data from the data stream rather than capturing the data over the entire range of data. It is useful because communication patterns change over time and that particular window will be used to train the classifier for the unknown instances within the stream. The first step consists of deciding window size based on number of records. Three experiments were performed with three different window sizes in the KDD laboratory using the following environment:

Machine	Operating System	RAM	Application	Algorithm
IBM–Think Center	Windows XP and Ubuntu Linux	512MB	Weka	REPTree

Table 5: Test environment

The data is extracted using a TCPdump tool along with different options as shown below:

tcpdump.exe –nv-c<number of records> host<host-name> > target-file.txt

Where switch n = Do not convert IP addresses into names

v =Produce verbose output

r = read source file

c = count packets

host = host name of the machine/router from where we want to capture traffic

Figure 6 shows pre-processed data representing a window that was created using TCPdump utility.

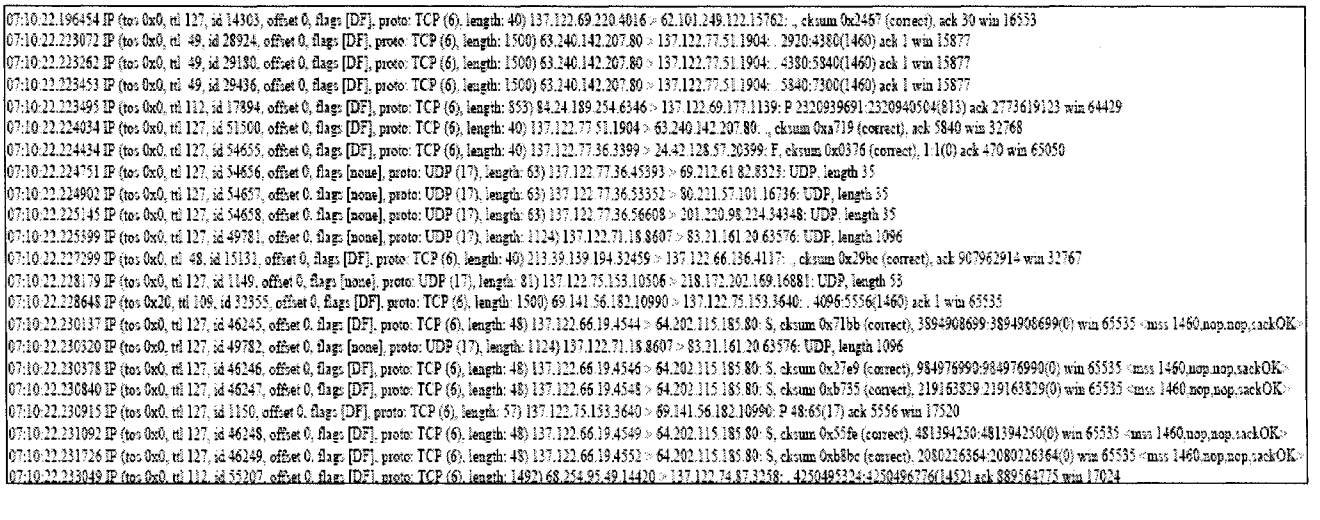


Figure 6: Abstract of a window generated by TCPdump

Each extracted window needs to be pre-processed before it is actually filtered for P2P, NonP2P and Unknown traffic. The methodology developed in [1] is used to pre-process the window and following five attributes are kept:

S/no	Name	Abbreviation	Description
1	Packet Length	'Len'	Packet Length ranging from 40 to 1500 bytes
2	Source IP Address	'SrcIP'	Sender's 32 bit IP address
3	Destination IP address	'DesIP'	Destination's 32 bit IP address
4	Source Port	'SrcPrt'	Sender's port
5	Destination Port	'DesPrt'	Destination port

Table 6: Attributes kept after Pre-processing

4.1.2 Stage -1 (Port filtering)

In this stage we input the pre-processed and clean data to the application developed in [1]. This application is based on PHP (Personal Home Page) which filters out P2P, NonP2P and Unknown traffic based on port numbers. The criterion of filtering is based on the four assumptions:

(a) Data instances having port numbers less than 1024 are classified as NonP2P because being well-known IANA port numbers[33]

(b) Some P2P applications are 'well-known' applications and use port numbers as shown in table 6 below:

P2P Application	Port Numbers
KaZaA	1214
BitTorrent	6881, 6889
Napster	6699, 6700, 6701
EDonkey 2000	4661, 4665

WinMX TCP/UDP	6257/6699
DirectConnect	411, 412
LimeWire (Gnutella Protocol)	6346, 6347
eMule TCP/UDP	4662/4672
Direct File Express	1044, 1045
WASTE	1337
CuteMX	2340
ShareDirect	2705
Abacast	4000-4100, 4500, 9000-9100
iMesh	4329
SongSpy	5190
Hotline Connect	5500-5503
Yoink	6666, 6667
Aimster/Madster	7668
BuddyShare	7788
Grouper	8038
hotComm	8080, 28864, 28864
Scour	8311
AudioGnome, OpenNap, Swaptor	8888, 8889
Blubster	41170
Messenger	1863
Netmeeting	1731

Table 7: Known P2P applications with their port numbers [1]

(c) The dynamic and private port numbers are those in the range 49152 to 65535. [33]P2P applications randomly select port numbers out of this range as these ports are not permanently assigned to any publicly defined application.

(d) If the data instance does not belong to (a) and (b) above then it is 'Unknown'

The port numbers are not to be used to train the classifier and were removed during a model building process. All instances were labeled as being P2P, NonP2P and Unknown traffic using the following pseudo code.

```
If (Src. port number OR Des. port number) < 1024
   Then Type = "NonP2P"
Else
   If (Src. port number OR Des. port number) = {well-known
standard port numbers such as
1214,6881,6889,6699,6700,6701,4661,4665,1337.....8888,8889,
41170}
    Type = "P2P"
Else
   Type = "UnKnown"
End
```

The purpose of introducing this stage is to decrease processing time by eliminating traffic comprised of known and standard port numbers. The second major objective of this stage is to generate a training data for the classifier containing a P2P/NonP2P instances. The information gained from these instances will be used in the second stage to build classifier and classify the future unknown and unlabelled instances in the data streams. Figure 7 shows the results of port filtering stage.

PacketLength	Source Address	Destination Address	Type
40	137.122.72.6	137.122.14.100	NOTP2P
603	137.122.72.6	137.122.14.100	NOTP2P
603	137.122.72.6	137.122.14.100	NOTP2P
1432	137.122.14.100	137.122.72.6	NOTP2P
48	137.122.71.120	71.232.235.50	P2P
48	137.122.71.120	86.0.197.178	P2P
48	137.122.71.120	24.42.38.243	P2P
1500	137.122.71.75	24.245.24.165	P2P
1500	137.122.71.75	207.201.244.20	P2P
1500	137.122.71.75	207.201.244.20	P2P
48	137.122.71.120	87.197.176.152	P2P
1500	216.167.232.212	137.122.76.174	P2P
48	137.122.70.52	66.110.61.21	UnKown
57	137.122.70.244	70.68.153.71	UnKown
81	137.122.70.244	84.70.133.158	UnKown
52	137.122.67.200	69.156.48.188	UnKown
1480	137.122.71.117	201.222.227.11	UnKown
40	137.122.66.12	24.77.58.97	UnKown
57	137.122.75.118	67.168.36.49	UnKown

Figure 7: Abstract of an output generated by the port filtering stage

4.1.3 Stage-2 (Model building and classification of unknown traffic)

In this stage a model is built based on P2P and NonP2P instances generated in stage-1. These instances are most current and exhibit most recent communication pattern.

4.1.3.1 Preprocessing:

IP addresses are converted into numeric numbers by using the following formula:

$$Numeric = A*(256^3) + B*(256^2) + C*(256) + D \quad \ldots\ldots (1)$$

Where, A.B.C.D is the octets of an IP address.

The purpose of conversion is to avoid IP addresses being considered as text during model building process.

4.1.3.2 Model building

We start building our model using a training data generated in stage-1.

Weka provides both Command Line Interface (CLI) and Graphical User Interface (GUI) to run an algorithm. The following script loads java class and starts building a model.

```
# Java weka.classifiers.REPTree –M 2 –V 0.0010 –N 3 –S 1 –L -1
```

Where;

REPTree is Fast Decision Tree Algorithm

M is maxDepth -- The maximum tree depth (-1 for no restriction).

2 is minNum -- The minimum total weight of the instances in a leaf.

V is minVarianceProp -- The minimum proportion of the variance on all the data that needs to be present at a node in order for splitting to be performed in regression trees.

noPruning -- Whether pruning is performed.

3 numFolds -- Determines the amount of data used for pruning. One fold is used for pruning, the rest for growing the rules.

S is seed -- The seed used for randomizing the data.

Testing Method: The testing method is 10-fold-cross validation in which a data is divided into ten folds and for each round of the ten fold cross validation, one fold is used for testing and the other nine folds are used for training.

Fast Decision Tree (FDT) [32] builds a decision tree using information gain/variance and prunes it using reduced-error pruning (with back fitting). It sorts values for numeric attributes once and missing values are dealt with by splitting the corresponding instances into pieces. It generates a confusion matrix as an output and a model is evaluated based on True Positive, True Negative, False Positive and False Negative rates.

Confusion Matrix: Confusion Matrix is measure of actual and predicted classifications done by a classification system. The confusion matrix helps to validate the model accuracy before the model is deployed and promoted for business solution in real world scenarios. The proposed model maximizes the True-Positive and True-Negative percentages by iterative assessment and tuning processes.

4.1.3.3 Model evaluation:

In this section performance of the proposed model is analyzed by three window sizes. We conducted three experiments to measure the accuracy, specificity and sensitivity of our classifier. Table 8 shows the window size and distribution of P2P/NonP2P instances:

Experiment	Window Size(records)	P2P	NonP2P
Window #1	1087	715	372
Window#2	5135	3723	1412
Window#3	11436	9329	2107

Table 8 : Various window sizes

Accuracy:

The accuracy (correctness) of the model is calculated based on the following formula:

$$Accuracy = (TP+TN)/(TP+TN+FP+FN) \quad \ldots (2)$$

Where, TP= True Positive, TN=True Negative, FN=False Negative, FP=False Positive

Figure 8 shows accuracy/ correctness of the classifier

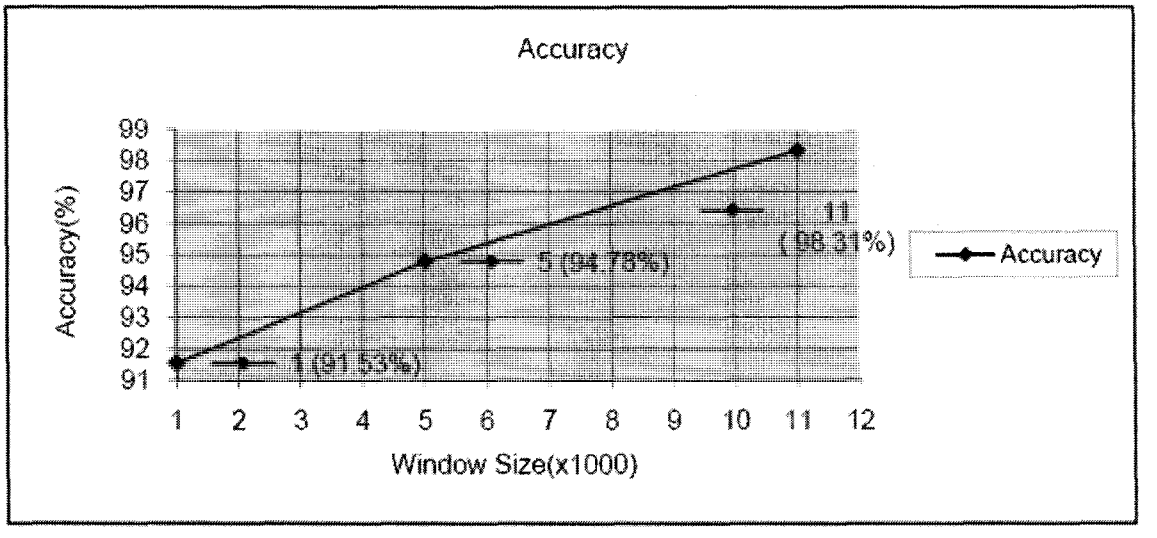


Figure 8: Accuracy/Correctness of FDT for various window sizes

It is important to observe that with the increase of window size accuracy increases because the classifier has more examples to build a model. Window size of 1K shows a decrease in accuracy to 91.35%. As we increase the window size accuracy starts increasing and reaches to 94.78% with a window size of 5K. It further reached to 98% with a window size of 11K.

During the experiments it was stipulated that memory constraints are addressed by using a window size less than 11K. The lab environment enabled performing tests on computers having 512MB of RAM which was able to handle 11K window size without any problems. With the increase of window size above 11K significant deterioration was observed in system performance with respect to RAM and system halted. Although, the accuracy of the classifier increases when the window size increases (because the classifier has more examples to build a model), in practice, memory constraints and the time-to-build the model are critical factors in deciding the size of the windows. Figure-9, for instance, shows the time-to-build models in seconds for three different window sizes. :

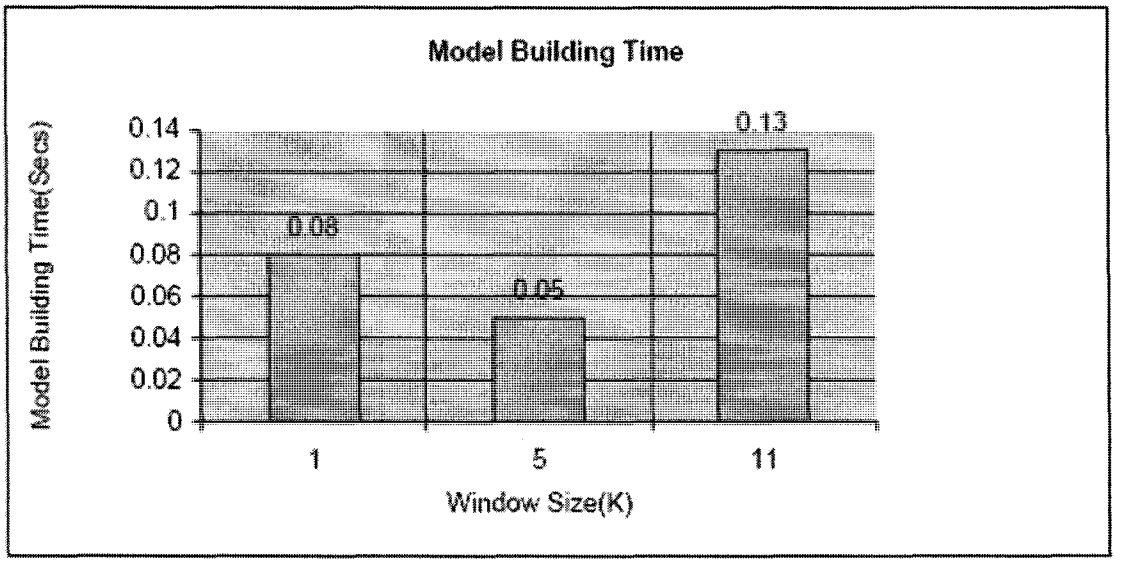


Figure 9: Model building time for 3 (three) window sizes

Window size of 1K exhibits model building time of .08 seconds which is higher than 5K window size. It is because 1K window size has not enough instances to build a model. If we move to higher window size of 5K and 11K accuracy of model increases without

considerable increase in model building time. In order to optimize memory and increase accuracy an ideal window size should be between 5K and 11K records.

Specificity:

Specificity is a measure of True negative rate and is calculated as follows:

$$\textit{Specificity (True Negative Rate)} = TN/(FP+TN) \quad \ldots \quad (3)$$

Figure 10 shows specificity of a classifier based on three window sizes i.e., 1K, 5K and 11K.

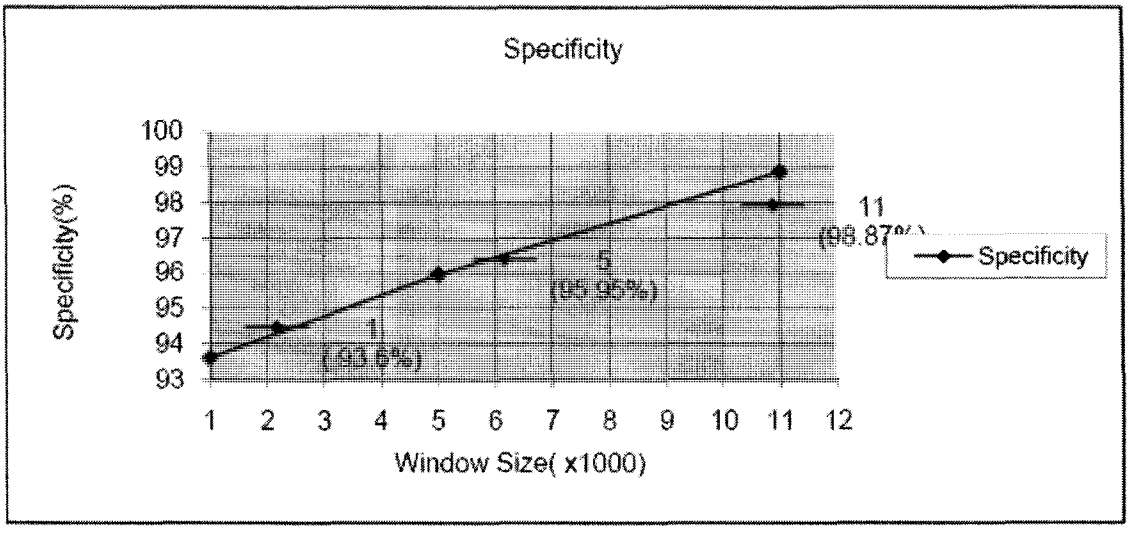


Figure 10: Specificity of Fast Decision Tree classifier

Specificity or True negative rate measures the percentage of instances correctly identified as NonP2P to the total number of NonP2P instances [1]. Window sizes with higher specificity depicts more suitable model as compared to the ones with low percentages.

Sensitivity:

Sensitivity, also called the True Positive rate, is measured by the percentage of the correctly P2P classified number of instances to the total number of P2P instances.

Sensitivity is measured as follows:

Sensitivity (True Positive Rate) = TP/ (TP +FN) ... (4)

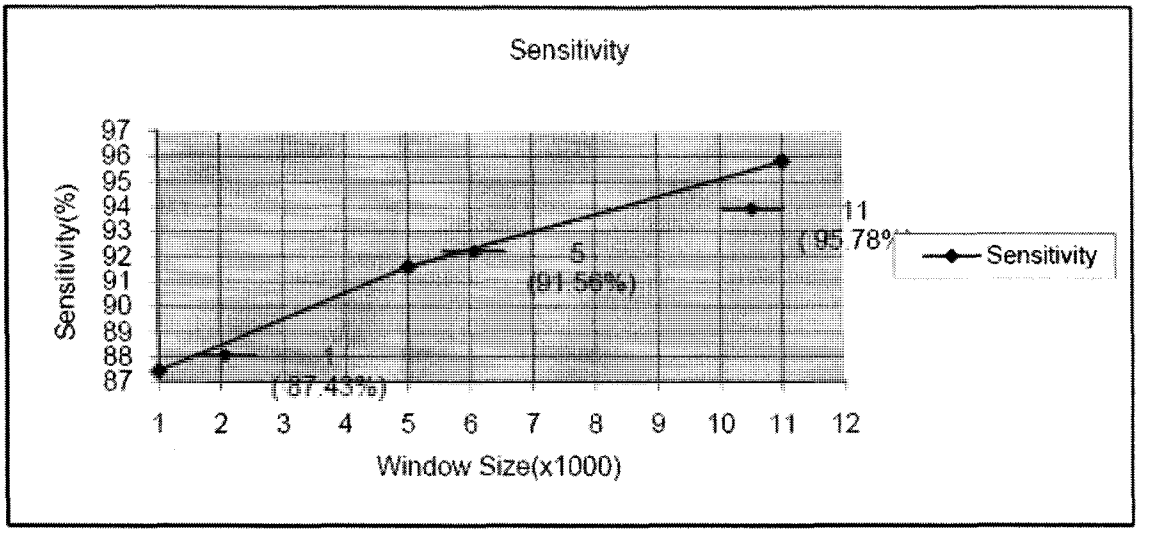


Figure 11: Sensitivity of the Fast Decision Tree Classifier

4.2 Classifying the Unknown traffic

In this section Unknown traffic is classified based on the model trained in stage -2. The essence of this stage is the use of unlabelled data and only three attributes; 'SrcIP', 'DesIP'

and 'Len'. Unknown traffic is generated when applications disguise their traffic by using non-standard port numbers. The motive behind disguising is to avoid being detected by the firewalls. We start our classification method by first doing preprocessing described in preceding section:

(a) *Preprocessing:* IP addresses are converted to numeric values using a formula 1 and later the unknown file is converted into CSV format. Classification being a supervised learning method requires pre-defined classes and this condition was satisfied during the training stage. Classifier still expects a 'Type' field in order to run the Fast Decision Tree while unknown traffic does not contain any 'Type' information. We insert " ? " in 'Type' field as shown in CSV file extract below:

Len	SrcIP	DesIP	Type
48	2306491956	1114520853	?
57	2306492148	1178900807	?
81	2306492148	1413907870	?
52	2306491336	1167863996	?
1480	2306492277	3386827531	?
40	2306490892	407714401	?
57	2306493302	1135092785	?
40	2306494009	2410144096	?
40	2306492179	3272224271	?
48	2306493636	1094499351	?
48	2306491030	1409904833	?

Figure 12: Abstract of Unknown file

(b) *Conversion to ARFF format:* CSV file is then converted to ARFF format using Weka GUI and the following abstract shows converted file :

```
@relation REALUNKNOWN_IPCONVERTED

@attribute Len numeric
@attribute SrcIP numeric
@attribute DesIP numeric
@attribute Type {NOTP2P,P2P}

@data

48,2306491956,1114520853,?
57,2306492148,1178900807,?
81,2306492148,1413907870,?
52,2306491336,1167863996,?
1480,2306492277,3386827531,?
40,2306490892,407714401,?
57,2306493302,1135092785,?
40,2306494009,2410144096,?
40,2306492179,3272224271,?
```

Figure 13: Unknown file in ARFF format

The file header contains declaration of attributes and data portion is comprised of Unknown instances.

(c) *Classifying Unknown Pre-Processed file:* In this subsection we demonstrate the classification of 'unknown' traffic using Fast Decision Tree. Our classifier predicts the class of every instance based on the knowledge gained during training stage. Unknown file is introduced to the classifier using GUI interface and output is generated in the form of a text file. Figure 14 shows the predicted output of unknown file:

```
@relation ReducedP2PFile3attributes_predicted

@attribute Instance_number numeric
@attribute Len numeric
@attribute SrcIP numeric
@attribute DesIP numeric
@attribute predictedType {NOTP2P,P2P}
@attribute Type {NOTP2P,P2P}

@data
0,48,2306491956,1114520853,P2P,?
1,57,2306492148,1178900807,P2P,?
2,81,2306492148,1413907870,P2P,?
3,52,2306491336,1167863996,P2P,?
4,1480,2306492277,3386827531,P2P,?
5,40,2306490892,407714401,P2P,?
6,57,2306493302,1135092785,P2P,?
7,40,2306494009,2410144096,P2P,?
8,40,2306492179,3272224271,P2P,?
9,48,2306493636,1094499351,P2P,?
10,48,2306491030,1409904833,P2P,?
11,51,2306491570,3475899948,NOTP2P,?
12,52,2306492716,1372308242,P2P,?
13,40,2306492153,3475898949,NOTP2P,?
14,40,2306491760,2392947918,P2P,?
15,40,2306493621,3475905301,NOTP2P,?
16,40,2306493360,1072824149,NOTP2P,?
17,40,2306492601,644055514,P2P,?
18,40,2306492601,1072824149,NOTP2P,?
19,40,2306492726,3475900012,NOTP2P,?
```

Figure 14: Predicted classes of Unknown file

The accuracy of the predicted output is based on the classifier's accuracy. It is observed from the output generated above that classifier predicts the class based on training it received during the model building process. It has no information of classes of each 'unknown' instance and decides a class based on packet length and IP addresses. This also proves the concept that Peers on the network exhibit distinct behaviors and can be grouped together/detected based on just three attributes. However, we still need to re-evaluate the classifier in order to make sure that our

output is within acceptable accuracy (90% and above). Section 4.3 describes the method of constantly re-evaluating the classifier.

4.3 Model Re-evaluation

The classifier accuracy is susceptible to diminish over period of time because of change in communication patterns and nature of an online stream, therefore, a constant re-evaluation is required to keep it accurate and up to date. In order to solve this problem we present an additional functionality based on keeping the model in memory and re-evaluate it using new instances. The purpose of this stage is two-folded:

(a) Save model rebuilding time: Model re-evaluation does not start model building from scratch and thus reduces overall process time.

(b) Gives confidence to the classifier's predicted output.

It is not necessary to re-evaluate the model with each upcoming window. We will rely on the model as long as it exhibits accuracy higher than 90% on periodically selected windows of traffic. When the accuracy drops below 90%, the FDT classifier is re-built using the upcoming windows of instances. In Figure-15, the model was built with accuracy of 91% and 1087 records. We will rely on the model until the accuracy is drops below 90%.

```
Relation:     ReducedP2PFile3attributes1K
Instances:    1087
Attributes:   4
              Len
              SIP
              DIP
              Type
Test mode:    10-fold cross-validation

=== Summary ===

Correctly Classified Instances         995               91.5363 %
Incorrectly Classified Instances        92                8.4637 %
```

Figure 15: Abstract of an output generated by FDT classifier model

Re-evaluation is conducted by keeping the model in memory and inputting a new preprocessed window of instances by selecting an option 'Re-evaluate the model'. The file is loaded and an output is generated which is appended to the model's output generated during model building stage. We tested the model based on second file containing 5135 records and it was observed that accuracy was down to 82% and model was not suitable to further classify the traffic as shown in figure 16. Furthermore, it also depends on business requirements of a particular application where accuracy is not very critical and any threshold value can be ascertained for accuracy.

```
=== Re-evaluation on test set ===

User supplied test set
Relation:     ReducedP2PFile3attributes5K
Instances:    5135
Attributes:   4

=== Summary ===

Correctly Classified Instances         4235               82.4732 %
Incorrectly Classified Instances        900               17.5268 %
```

Figure 16: Re-evaluation of model

Once the model's accuracy declines it has to be removed from memory and new model be built from scratch. The benefits of removing the model from memory are two folded:

(a) It overcomes memory space limitations by keeping up-to-date and accurate model.

(b) The new model reflects change in communication pattern.

4.4 A comparison of K-means clustering vs. the two-stage classifier

Table -9 shows a comparison of the two methods developed in this research.

K-Means Clustering	Two- stage FDT classifier
Uses un-labeled data during micro cluster creation and labeled data later at the merger phase.	Uses labels during classification and classifies upcoming windows based on un-labeled data
Labeling introduces delay in processing thus increasing cycle turnaround time.	Does not require labels once a classifier is trained.
Performs clustering on window sizes higher than 10,000 instances without requiring additional memory(<=128MB)	Requires additional memory (>128MB) for windows higher than 10,000 instances
Merged clusters represent a mix population of P2P and Non-P2P instances. It is sometimes difficult to pronounce a cluster as P2P/Non-P2P	Accuracy of P2P and Non-P2P classified instances is above 90% thus helps in characterizing traffic as a P2P or Non-P2P.
It still requires a new methodology to cluster unknown data instances. Even in previous work [10] authors used labels to generate final clusters.	Classifies unknown traffic after it is trained with accurately labeled data.
It requires pre-determination of k value. Accuracy is greatly dependent on correct selection of K-value. Thus requiring an additional work for every distinct data type.	It is not dependent on any pre-determined parameter and thus works well for distinct data types.

Table 9: A comparison of K-means clustering and the two-stage classifier

Table-10 also illustrates some of the differences between the present work and previous methods. This table does not include the work in [1] because their method is not based on stream data classification.

Method Name	Accuracy	Time to Build(seconds)	Attributes	Labeled/Un-labeled data	Year
K-means Clustering(Chap3)	96% with K=6	3	3	Un-Labeled Data	2007
Fast Decision Tree(Chap4)	98.31% with 11K window size	13.0(11k window)	3	labeled data	2008
On-Demand Classifier[10]	85-98%(depends on dataset)	Info N/A	42	Labeled	2006
Very Fast Decision Tree(VFDT)[11]	72.7%	18.3	21	Labeled	2000
Concept Adapting Very Fast Decision Tree(CVFDT)[12]	72.3%	Info N/A	21	Info N/A	2001

Table 10: A comparison of the present and previous methods.

CHAPTER 5 –IMPLEMENTATION

Implementation is motivated by the desire to integrate all the components and create a cohesive system that should be able to function in an orderly manner. The proposed system includes data extraction module, data transformation engine and finally the classification system. All these modules are based on methodology and design described in previous chapters.

In order to implement the proposed system we present a 3 module architecture shown in figure 17.

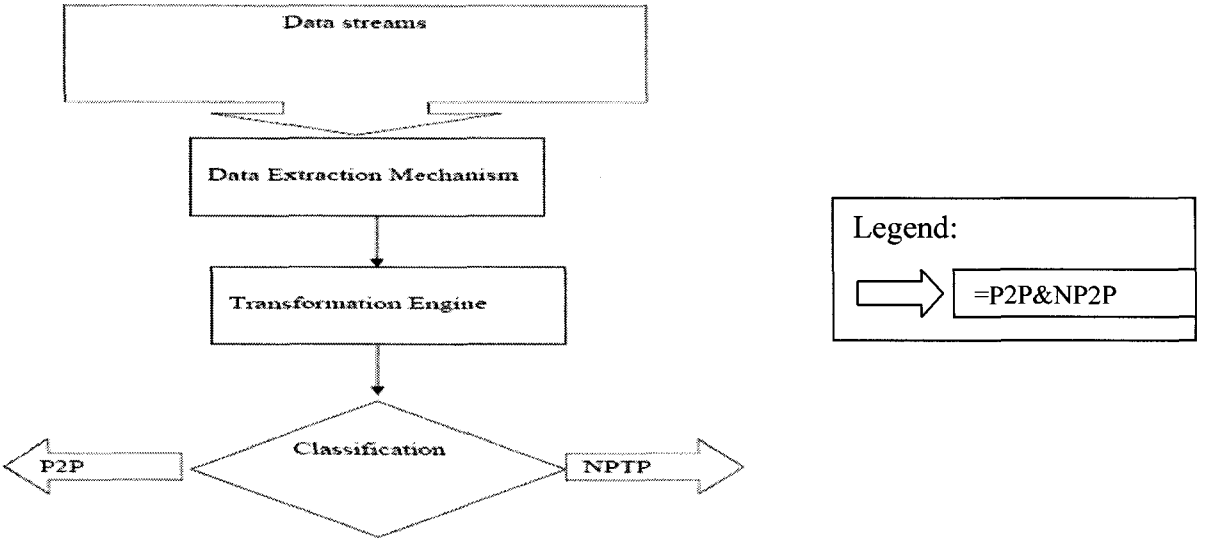


Figure 17: Implementation Architecture

In the above mentioned architecture data streams are first captured using a script which is controlled by a cron. Second, once the data is extracted it is copied to the directory where it is preprocessed and this stage is named as 'Transformation' stage. Third, a classification is performed which includes both training and classifying unknown traffic stages.

5.1 Data Extraction Mechanism

In this section we present a method to extract data and create an environment for window pre-processing. Major hindrance in analyzing and importing data into Microsoft Windows for performing data mining tasks is the incompatibility of 'instance windows' generated by routers or computer interfaces and Windows operating system. Following steps were performed to address the incompatibility problem:

1. Installed Ubunto Linux version 6.10 on one the lab machines with configuration; HP pavilion xt953, Pentium-3, 256MB RAM and 28GB hard disk.
2. Installed and configured Samba on Linux Ubunto version 6.01.
3. Plugged in portable hard disk through USB interface.
4. Shared the portable drive.
5. Logged to Linux machine through windows using username "smb" and password.
6. Data files were accessible through windows operating system and was able to be further processed.

Samba: Samba is a software that runs on a platform other than Microsoft Windows, for example, UNIX, Linux, IBM System 390, OpenVMS, and other operating systems. Samba

uses the TCP/IP protocol installed on the host server. It allows the host to interact with a Microsoft Windows client or server as if it is a Windows file and print server.

Samba Installation: Samba is based on Server Message Block (SMB) protocol that facilitates the sharing of files, folders, volumes, and the sharing of printers throughout the network.

5.1.1 Samba Installation steps:

(i) At the command prompt type:

```
apt-get install samba
```

(ii) SAMBA file that has all configuration and tune up setting resides at the following path:

```
/etc/samba/smb.conf
```

Script at step (i) will install SAMBA

(iii) Restart SAMBA using following command line:

```
/etc/init.d/samba restart
```

Or

```
/etc/init.d/samba stop
```

```
/etc/init.d/samba start
```

(iv) The computer account has to be added to the SAMBA password file.

```
smbpasswd –a –m kddlab
```

Where 'kddlab' is a computer running Linux machine

(v) Create a user smb in Linux machine.

(vi) Change password of user smb

```
smbpasswd –a smb
```

5.2 Data extraction scheduling

Cron is programmed to start running TCPdump script at user defined timings.

Cron: Cron is a program that enables UNIX users to run commands/scripts automatically at a specified time and date. [34]

Crontab: Crontab files contain list of jobs to be executed by the cron. Cron scans the crontab files and loads the job into memory for execution at a specified time and date. Crontab has six fields each of which is separated by space:

1. Minute (0-59)
2. Hour (0-23)
3. Day of month (1-31)
4. Month of the year (1-12 or Jan, Feb etc)
5. Day of week (0-6; where as 0=Monday and 6=Sunday)
6. Command to run.

In the present work crontab is modified as follows:

(i) Crontab file is invoked for editing

```
# crontab –e
```

(ii) Following line is added in crontab

```
5 * * * * /tcpdump –nv -c<number of records> host <host-name> > target-file.txt
```

"5" above shows that window is created after every 5 minutes. It was selected arbitrarily and can be changed to any numeric value between 0-59. Second an "*" would run a tcpdump script every hour after 5 minutes interval. Third "*' would run every day of month (1-31). Fourth "*" is for month and it would run every month (1-12) and fifth "*" represents that it would run every day of week (Monday to Sunday). In order to create a window every minute the following line is added in crontab.

```
* * * * * /tcpdump –nv -c<number of records> host <host-name> > target-file.txt
```

In the above crontab line the first "*" would create the window at the 1 minute mark, meaning we will generate 60 windows per hour.

The output of the cron job is a text file containing a certain number of records for performing pre-processing and classification tasks.

5.3 Transformation

Once the window is generated as a result of script defined in subsection 5.2(ii) and TCPdump we move the file to the directory where pre-processing is done

```
# mv /usr/justextracted/win1  /usr/tobeprocessed/win1
# mv /usr/justextracted/win2  /usr/tobeprocessed/win2
```

In the above script 'mv' command moves win1 file from '/usr/justextracted' directory to the '/usr/tobeprocessed' directory and keeps the same file name. The purpose of using 'mv' command instead of copy (cp) command is that mv deletes the original file and thus enables us to conserve memory storage space.

Once the file is in 'tobeprocessed' directory we import the file to the machine running Miscrosoft Windows XP using the samba connection. We run PHP code and generate field separators. The PHP application generates a text file as an output and we open this file in Excel so as to save it in CSV format. The purpose of creating a CSV file is to make it compatible with Weka recognizable format.

5.4 Training

We then move the file to the directory "C:\internet" to perform port filtering by using the PHP code.

As a result we get a file consisting of P2P, NonP2P and Unknown instances. This file is now ready to train the classifier based on P2P and NonP2P instances. In order to cross – validate the model using ten folds we use the following code [32]

```
import weka.classifiers.Evaluation;
import java.util.Random;
...
Evaluation eval = new Evaluation(newData);
eval.crossValidateModel(tree, newData, 10, new Random(1)); // 10 fold cross validation
```

The following partial code trains the model based on P2P and NonP2P instances. It uses FDT and 10 folds cross validation method. It first invokes java classes for instances, evaluation and FDT algorithm. In the second block it inputs the file containing P2P and NonP2P file 'win1'. It starts building a model by calling a method cls.buildClassifier(train). Once the model is build it evaluates the model built during the model building process.

```
import weka.core.Instances;
import weka.classifiers.Evaluation;
import weka.classifiers.trees.REPTree;  // FDT algorithm
...
Instances train = c:\internet\win1     // CSV file generated as a result of port filtering
Instances test = c:\internet\win1
```

```
// we first train our classifier based on the P2P and NonP2P instances
Classifier cls = new REPTree();
cls.buildClassifier(train);

// evaluate classifier
Evaluation eval = new Evaluation(train);
eval.crossValidateModel(tree, newData, 10, new Random(1)); // 10 fold cross validation

// Out put is printed
System.out.println(eval.toSummaryString("\nResults\n==\n", false)); // Print on new line
```

5.5 Classifying unknown traffic

In this subsection we present a partial code that classifies the unknown traffic. First, it invokes 5 classes and then loads 'unknown' file and starts classifying the data instances based on training provided in the first stage. The output is 'predicted instances' which predicts the classes of all unknown traffic.

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import weka.core.Instances;
...
// load Unknown data
Instances unlabeled = new Instances(
                    new BufferedReader(
                     new FileReader("/internet/unknown.arff"))); // loads the unknown
instances file

// set class attribute
unlabeled.setClassIndex(unlabeled.numAttributes() - 1);

// create copy
Instances labeled = new Instances(unlabeled);

// label predicted  instances
for (int i = 0; i < unlabeled.numInstances(); i++) {
  clsLabel = tree.classifyInstance(unlabeled.instance(i));
  labeled.instance(i).setClassValue(clsLabel);
}

// save predicted labeled data
```

```
BufferedWriter writer = new BufferedWriter(
                    new FileWriter("/internet/unknownpredicted.arff"));
writer.write(labeled.toString());
writer.newLine();
writer.flush();
writer.close();
```

CHAPTER 6 – CONCLUSION AND FUTURE WORK

Accurate classification of Internet traffic is a fundamental requirement for network provisioning, network security, maintaining quality of services and network management. The traffic volume and patterns of some of the new applications such as peer-to-peer (P2P) file sharing put pressure on service providers' networks in terms of congestion and delay, to the point that maintaining Quality of Services (QoS) planned in the access network requires the provisioning of additional bandwidth sooner than planned. The focused of this thesis is mainly on handling stream data and classification of P2P applications in the internet traffic using data mining techniques. The two techniques presented in this thesis, merged micro-clustering and two-stage classifier with fast decision tree, contribute significantly to the field of study as follows:

1. Clustering technique establishes a method to create two distinct clusters representing P2P and NonP2P population. The two final clusters are generated from several micro clusters that contained mix of P2P and NonP2P instances. In future work we recommend extending this technique to include time stamps as an attribute and merge the clusters based on unlabeled unknown instances.
2. The second technique is a two-stage window-based architecture for classification of Peer-to-Peer traffic where in the first stage we apply port filtering to label well-known P2P and NonP2P traffic leaving the rest of the traffic as Unknown. In the second stage, we train a Fast Decision Tree classifier using labeled records produced in the previous stage. Unknown traffic is then applied to the FDT classifier which predicts the classes of

Unknown instances based on accuracy of the model. We conducted experiments where the Internet traffic was captured at the campus gateway, preprocessed, and applied to the proposed two-stage architecture. The analysis showed that we can achieve an accuracy of higher than 90% in classification of Unknown traffic. The accuracy of the model is periodically re-evaluated on recent windows. When the accuracy falls below than 90%, a new FDT model is built using upcoming windows. This will keep the model up-to-date and accurate. The new classification technique only relies on three attributes, namely source IP address, destination IP address and packet length thus eliminating the need for deep packet inspection and the privacy issues associated with it.

3. The window-based capturing of live data was proposed where the application runs on a Linux (ubunto) platform and captures internet traffic periodically. It then pre-processes the data captured within the window to be applied to the two-stage classifier with FDT.

As the continuation of this work, we propose developing an integrated application where the codes for window-based capturing of live traffic (cron and samba) will be integrated with the codes for the two-stage architecture (the filtering stage, and the fast decision tree stage) to create an integrated executable code that can be imported in routers and commercial products for traffic classification. The integrated code can be in Java or C++. The input to the integrated system will be stream of internet traffic and the output will be one of the two classes assigned to the traffic (P2P or NonP2P).

The techniques and methodology developed in this thesis for classification of P2P traffic can be extended in the future to a wider scope, for instance, application signature detection and network security (anomaly detection).

PUBLICATIONS

Accepted for Publication:

B.Raahemi, A.Mumtaz, " A Two Stage Window-based Architecture for Classification of Peer-to-Peer traffic using Fast Decision Tree", *The 4th International Conference on Data Mining, LasVegas, Nevada, USA, 2008*

Under Review:

B.Raahemi, A.Mumtaz,A.Hayajneh,P.Rabinovitch, " Classification of Peer-to-Peer Traffic Using a Two-Stage Window-based Classifier with Fast Decision Tree and IP layer Attributes", *PMECT08, 2nd International Workshop on Performance Modeling and Evaluation in Computer and Telecommunication Networks, Virgin Islands, USA, August 4-7, 2008*

REFERENCES

[1] B. Raahemi, A. Hayajneh and P. Rabinovitch, "Peer-to-Peer Traffic Classification Using Decision Tree and IP Layer Attributes," *International Journal of Business Data Communications and Networking,* vol. 3, pp. 60-72, 2007.

[2] M. Miller, *Discovering P2P.* USA: Sybex, 2001.

[3] Cloud Shield, *Peer-to-peer traffic control. 2007(October),* Available: http://www.cloudshield.com/solutions/p2pcontrol.asp

[4] Azzouna, N.B.; Guillemin, F., "Impact of peer-to-peer applications on wide area network traffic: an experimental approach", *IEEE Global Telecommunications Conference, GLOBECOM '04,* Vol. 3, P1544-1548 (29 Nov.-3 Dec. 2004)

[5] __"Cisco IOS NetFlow Overview", Whitepaper, available at www.Cisco.com, Cisco Systems Inc., 2006.

[6] M. Corvella and B. Krishnamurthy, *Internet Measurement: Infrastructure, Traffic and Applications.* West Sussex, England: John Wiley and sons Ltd, 2006,

[7] S. Zander, T. Nguyen and G. Armitage, "Self-learning IP traffic classification based on statistical flow characteristics, in Passive and Active Network Measurement" , vol. 3431/2005, Springer Berlin / Heidelberg, 2005, pp. 325-328.

[8] Cheeseman, P. and Stutz J, *"Bayesian classification (Autoclass): theory and results, Advances in knowledge discovery and data mining*", American Association for Artificial Intelligence, Menlo Park, CA, 1996.

[9] N. Williams, S. Zander and G. Armitage., "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification". *ACM SIGCOMM Computer Communication Review 36(5),* pp. 5-16, 2006

[10] C. C. Aggarwal, J. Han and W. Jianyong., "A framework for on-demand classification of evolving data streams". *IEEE Transactions on Knowledge and Data Engineering 18(5),* pp577-589, 2006

[11] P. Domingos and G. Hulten. "Mining high-speed data streams". Presented at *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ,pp 71-80,2000.

[12] G. Hulten, L. Spencer and P. Domingos. "Mining time-changing data streams". Presented at *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge*

Discovery and Data Mining. pp97-106, 2001

[13] J. Gama, R. Rocha and P. Medas. "Accurate decision trees for mining high-speed data streams". Presented at *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.,pp 523-528, 2003

[14] R. Jin and G. Agrawal. "Efficient decision tree construction on streaming data". Presented at *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.pp 571-576, 2003.

[15] T. Karagiannis, K. Papagiannaki and M. Faloutsos. "BLINC: Multilevel traffic classification in the dark". *ACM SIGCOMM Computer Communication Review 35(4),2005,* pp. 229-240.

[16] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule and K. Salamatian. "Traffic classification on the fly". *ACM SIGCOMM Computer Communication Review 36(2),* pp. 23-26.

[17] K. Xu, Z. L. Zhang and S. Bhattacharyya. "Profiling internet backbone traffic: Behavior models and applications". Presented at P*roceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* .

[18] A. Oveissian, K. Salamatian and A. Soule. "Fast flow classification over internet". Presented at *Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04)*.

[19] H. Alhammady and K. Ramamohanarao. "Mining emerging patterns and classification in data streams." Presented at *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence* .

[20] A. Moore and D. Zuev. "Internet traffic classification using bayesian analysis techniques". Presented at *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*.

[21] A. Moore, D. Zuev and M. Crogan. 2005, "Discriminators for use in flow-based classification". Available: www.cl.cam.ac.uk/~awm22/publications/moore2005discriminators.pdf.

[22] Y. Zeng and T. M. Chen. "Automatic model classification of measured internet traffic". Presented at *IPOM, IEEE Workshop*. Available: http://engr.smu.edu/~tchen/papers/IPOM2002_Oct2002.pdf

[23] R. Dass and A. Mahanti, "An efficient technique for frequent pattern mining in real-time business applications," in *Proceedings of the 38th Hawaii International Conference on System Sciences,* 2005

[24] Gaber M.M, East.C, Krishnaswamy.C and Zaslavsky."A. Cost-efficient mining techniques for data streams". Presented at *Proceedings of the Second Workshop on Australasian Information Security, Data Mining and Web Intelligence, and Software Internationalisation*

[25] B. Babcock, S. Babu, M. Datar, R. Motwani and J. Widom. "Models and issues in data stream systems". Presented at *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*.

[26] C. Raïssi, N. Cedex, P. Poncelet and M. Teisseire. « Towards a new approach for mining frequent itemsets on data stream". *28(1),2007,* pp. 23-36

[27] S. Sen, O. Spatscheck and W. D. "Accurate, scalable in-network identification of p2p traffic using application signatures". *Presented at Proceedings of the 13th International Conference on World Wide Web.* Available: www.portal.acm.org/citation.cfm?id=988742

[28] M. Roughan, S. Sen, O. Spatscheck and N. Duffield. "Class-of-service mapping for QoS: A statistical signature-based approach to IP traffic classification". Presented at *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*.

[29] J. Chang H. and W. Lee S. "estWin: Online data stream mining of recent frequent itemsets by sliding window method". *Journal of Information Science 31(2), 2005*

[30] M. H. Dunham, *Data Mining.* Prentice Hall, 2002.

[31] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of 5-Th Berkeley Symposium on Mathematical Statistics and Probability,* 1967, pp. 281-297.

[32] Ian H. Witten & Frank, E. "Data Mining, Practical Machine Learning Tool and Techniques", Publisher: Elsevier printing, 2005.

[33] Wikipedia, < www.wikipedia.org> , accessed November 2007.

[34]< http://www.unixgeeks.org/security/newbie/unix/cron-1.html> accessed March 2008.

APPENDIX

A. Run information based on 1087 instances and 10 fold cross validation

=== Run information ===

Scheme: weka.classifiers.trees.REPTree -M 2 -V 0.0010 -N 3 -S 1 -L -1
Relation: ReducedP2PFile3attributes1K
Instances: 1087
Attributes: 4
Len
SIP
DIP
Type
Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

REPTree
==========

```
Len < 340
|  Len < 46.5
|  |  DIP < 2219371306.5
|  |  |  SIP < 2306493467.5
|  |  |  |  SIP < 2306491443.5
|  |  |  |  |  SIP < 2306491390 : P2P (11/0) [5/0]
|  |  |  |  |  SIP >= 2306491390 : NONP2P (9/0) [2/0]
|  |  |  |  SIP >= 2306491443.5 : P2P (54/2) [20/2]
|  |  |  SIP >= 2306493467.5
|  |  |  |  SIP < 2306493598.5 : NONP2P (14/2) [6/2]
|  |  |  |  SIP >= 2306493598.5 : P2P (3/0) [5/1]
|  |  DIP >= 2219371306.5
|  |  |  DIP < 2306491553.5 : NONP2P (35/1) [26/7]
|  |  |  DIP >= 2306491553.5
|  |  |  |  DIP < 2755139017
|  |  |  |  |  SIP < 3333317582.5 : P2P (39/3) [16/3]
|  |  |  |  |  SIP >= 3333317582.5
|  |  |  |  |  |  SIP < 3547559962.5 : NONP2P (6/1) [8/3]
|  |  |  |  |  |  SIP >= 3547559962.5 : P2P (5/0) [3/0]
|  |  |  |  DIP >= 2755139017
|  |  |  |  |  DIP < 3486538045 : NONP2P (32/2) [12/0]
|  |  |  |  |  DIP >= 3486538045 : P2P (11/1) [8/0]
|  Len >= 46.5
|  |  Len < 177
|  |  |  DIP < 3408419239.5
|  |  |  |  DIP < 2306491036.5
|  |  |  |  |  SIP < 2306490747.5
|  |  |  |  |  |  SIP < 1895102457 : P2P (4/0) [1/0]
|  |  |  |  |  |  SIP >= 1895102457 : NONP2P (7/2) [2/0]
|  |  |  |  SIP >= 2306490747.5 : P2P (84/2) [52/0]
|  |  |  DIP >= 2306491036.5 : P2P (114/1) [55/2]
```

```
|   |   |   DIP >= 3408419239.5
|   |   |   |   DIP < 3543895864.5 : NONP2P (3/0) [5/2]
|   |   |   |   DIP >= 3543895864.5 : P2P (5/0) [3/0]
|   |   |  Len >= 177
|   |   |   SIP < 1845329300.5 : P2P (13/1) [3/1]
|   |   |   SIP >= 1845329300.5
|   |   |   |   Len < 230 : NONP2P (5/0) [4/1]
|   |   |   |   Len >= 230
|   |   |   |   |   SIP < 2704284295.5
|   |   |   |   |   |   Len < 298 : P2P (13/0) [4/1]
|   |   |   |   |   |   Len >= 298 : NONP2P (3/1) [5/1]
|   |   |   |   |   SIP >= 2704284295.5
|   |   |   |   |   |   Len < 279 : NONP2P (6/1) [0/0]
|   |   |   |   |   |   Len >= 279 : P2P (2/0) [2/0]
Len >= 340
|  Len < 1359.5
|   |  DIP < 2306492652.5
|   |   |   DIP < 746592482 : P2P (7/0) [1/0]
|   |   |   DIP >= 746592482
|   |   |   |   DIP < 2306492322.5
|   |   |   |   |   SIP < 1173401394.5
|   |   |   |   |   |   Len < 642 : P2P (3/0) [1/0]
|   |   |   |   |   |   Len >= 642 : NONP2P (16/0) [8/0]
|   |   |   |   |   SIP >= 1173401394.5
|   |   |   |   |   |   SIP < 1883210741.5 : P2P (13/0) [4/0]
|   |   |   |   |   |   SIP >= 1883210741.5
|   |   |   |   |   |   |   SIP < 2306492203.5
|   |   |   |   |   |   |   |   SIP < 2306491390
|   |   |   |   |   |   |   |   |   SIP < 2306491041.5 : NONP2P (6/0) [3/1]
|   |   |   |   |   |   |   |   |   SIP >= 2306491041.5 : P2P (3/0) [1/0]
|   |   |   |   |   |   |   |   SIP >= 2306491390 : NONP2P (10/0) [4/2]
|   |   |   |   |   |   |   SIP >= 2306492203.5
|   |   |   |   |   |   |   |   SIP < 2306493157 : P2P (10/0) [3/0]
|   |   |   |   |   |   |   |   SIP >= 2306493157
|   |   |   |   |   |   |   |   |   SIP < 3533529246.5 : NONP2P (15/3) [2/0]
|   |   |   |   |   |   |   |   |   SIP >= 3533529246.5 : P2P (3/0) [2/0]
|   |   |   |   DIP >= 2306492322.5 : P2P (6/0) [1/0]
|   |   |  DIP >= 2306492652.5
|   |   |   |   Len < 1283
|   |   |   |   |   SIP < 2445760013 : NONP2P (24/5) [12/5]
|   |   |   |   |   SIP >= 2445760013 : P2P (5/0) [1/0]
|   |   |   |  Len >= 1283 : NONP2P (52/1) [31/1]
|   |  Len >= 1359.5
|   |   |  SIP < 3354060896.5 : P2P (57/1) [30/0]
|   |   |  SIP >= 3354060896.5
|   |   |   |   SIP < 3502291513 : NONP2P (13/0) [9/0]
|   |   |   |   SIP >= 3502291513 : P2P (3/0) [3/0]

Size of the tree : 79

Time taken to build model: 0.08 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         995               91.5363 %
Incorrectly Classified Instances        92                8.4637 %
```

```
Kappa statistic                  0.8123
Mean absolute error              0.1147
Root mean squared error          0.2702
Relative absolute error         25.4733 %
Root relative squared error     56.9549 %
Total Number of Instances        1087

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
 0.879    0.066     0.874     0.879    0.877     NONP2P
 0.934    0.121     0.937     0.934    0.936     P2P

=== Confusion Matrix ===

   a   b   <-- classified as
 327  45 |   a = NONP2P
  47 668 |   b = P2P
```

B. Run information based on 5135 records and 10 fold cross validation

```
=== Run information ===

Scheme:       weka.classifiers.trees.REPTree -M 2 -V 0.0010 -N 3 -S 1 -L -1
Relation:     ReducedP2PFile3attributes5K
Instances:    5135
Attributes:   4
              Len
              SIP
              DIP
              Type
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===


REPTree
============

Len < 407.5
|   Len < 48.5
|   |   DIP < 2219371303
|   |   |   DIP < 1166215728
|   |   |   |   DIP < 1149322072
|   |   |   |   |   DIP < 211690922.5 : NONP2P (4/0) [3/0]
|   |   |   |   |   DIP >= 211690922.5
|   |   |   |   |   |   DIP < 1054191136.5
|   |   |   |   |   |   |   SIP < 2306491919
|   |   |   |   |   |   |   |   SIP < 2306491835 : P2P (22/0) [11/0]
|   |   |   |   |   |   |   |   SIP >= 2306491835 : NONP2P (3/1) [1/0]
|   |   |   |   |   |   |   SIP >= 2306491919 : P2P (68/0) [36/0]
|   |   |   |   |   |   DIP >= 1054191136.5
|   |   |   |   |   |   |   DIP < 1125000729
|   |   |   |   |   |   |   |   DIP < 1120649218.5
|   |   |   |   |   |   |   |   |   DIP < 1106984674.5
|   |   |   |   |   |   |   |   |   |   DIP < 1093932359.5
|   |   |   |   |   |   |   |   |   |   |   DIP < 1080085003.5 : NONP2P (6/2) [5/0]
```

```
|  |  |  |  |  |  |  |  |  |  |  |  DIP >= 1080085003.5 : P2P (18/0) [5/0]
|  |  |  |  |  |  |  |  |  |  |  DIP >= 1093932359.5
|  |  |  |  |  |  |  |  |  |  |  |  DIP < 1095358363 : NONP2P (6/0) [7/0]
|  |  |  |  |  |  |  |  |  |  |  |  DIP >= 1095358363
|  |  |  |  |  |  |  |  |  |  |  |  |  DIP < 1100664428.5 : P2P (2/0) [2/0]
|  |  |  |  |  |  |  |  |  |  |  |  |  DIP >= 1100664428.5 : NONP2P (3/0) [3/0]
|  |  |  |  |  |  |  |  |  |  DIP >= 1106984674.5 : P2P (16/0) [9/1]
|  |  |  |  |  |  |  |  |  DIP >= 1120649218.5 : NONP2P (7/2) [2/1]
|  |  |  |  |  |  |  |  DIP >= 1125000729 : P2P (44/0) [24/0]
|  |  |  |  |  DIP >= 1149322072
|  |  |  |  |  |  SIP < 2306492318
|  |  |  |  |  |  |  SIP < 2306491134 : P2P (9/0) [1/0]
|  |  |  |  |  |  |  SIP >= 2306491134 : NONP2P (72/2) [30/0]
|  |  |  |  |  |  SIP >= 2306492318 : P2P (18/1) [11/0]
|  |  |  |  DIP >= 1166215728
|  |  |  |  |  SIP < 2306493496 : P2P (262/0) [135/1]
|  |  |  |  |  SIP >= 2306493496
|  |  |  |  |  |  SIP < 2306493578 : NONP2P (17/6) [6/0]
|  |  |  |  |  |  SIP >= 2306493578 : P2P (17/1) [8/0]
|  |  DIP >= 2219371303
|  |  |  DIP < 3519157672.5
|  |  |  |  DIP < 3421712537.5
|  |  |  |  |  DIP < 2306492204.5
|  |  |  |  |  |  SIP < 3481366663.5
|  |  |  |  |  |  |  SIP < 3417961378
|  |  |  |  |  |  |  |  DIP < 2230986525 : NONP2P (18/0) [8/0]
|  |  |  |  |  |  |  |  DIP >= 2230986525
|  |  |  |  |  |  |  |  |  SIP < 1167787027
|  |  |  |  |  |  |  |  |  |  SIP < 1147110962
|  |  |  |  |  |  |  |  |  |  |  DIP < 2306491458 : P2P (17/0) [7/0]
|  |  |  |  |  |  |  |  |  |  |  DIP >= 2306491458
|  |  |  |  |  |  |  |  |  |  |  |  DIP < 2306491671.5 : NONP2P (6/0) [2/0]
|  |  |  |  |  |  |  |  |  |  |  |  DIP >= 2306491671.5
|  |  |  |  |  |  |  |  |  |  |  |  |  SIP < 1049138155 : P2P (5/0) [3/0]
|  |  |  |  |  |  |  |  |  |  |  |  |  SIP >= 1049138155
|  |  |  |  |  |  |  |  |  |  |  |  |  |  DIP < 2306491809 : P2P (2/0) [3/0]
|  |  |  |  |  |  |  |  |  |  |  |  |  |  DIP >= 2306491809 : NONP2P (8/4) [10/3]
|  |  |  |  |  |  |  |  |  |  SIP >= 1147110962 : NONP2P (43/0) [23/3]
|  |  |  |  |  |  |  |  |  SIP >= 1167787027
|  |  |  |  |  |  |  |  |  |  DIP < 2306490822.5 : NONP2P (10/2) [6/0]
|  |  |  |  |  |  |  |  |  |  DIP >= 2306490822.5 : P2P (45/0) [29/0]
|  |  |  |  |  |  |  SIP >= 3417961378 : NONP2P (32/0) [13/1]
|  |  |  |  |  |  SIP >= 3481366663.5 : P2P (16/0) [9/0]
|  |  |  |  |  DIP >= 2306492204.5
|  |  |  |  |  |  DIP < 2306493441.5 : P2P (146/10) [68/3]
|  |  |  |  |  |  DIP >= 2306493441.5
|  |  |  |  |  |  |  DIP < 3314579613.5
|  |  |  |  |  |  |  |  DIP < 3262743942.5
|  |  |  |  |  |  |  |  |  DIP < 2309014449
|  |  |  |  |  |  |  |  |  |  DIP < 2306503345.5
|  |  |  |  |  |  |  |  |  |  |  DIP < 2306493857
|  |  |  |  |  |  |  |  |  |  |  |  DIP < 2306493490 : NONP2P (8/0) [2/0]
|  |  |  |  |  |  |  |  |  |  |  |  DIP >= 2306493490
|  |  |  |  |  |  |  |  |  |  |  |  |  SIP < 715526029.5 : P2P (5/0) [3/0]
|  |  |  |  |  |  |  |  |  |  |  |  |  SIP >= 715526029.5
|  |  |  |  |  |  |  |  |  |  |  |  |  |  SIP < 1102066727.5 : NONP2P (12/1) [2/0]
|  |  |  |  |  |  |  |  |  |  |  |  |  |  SIP >= 1102066727.5
```

```
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  SIP < 3407694231 : P2P (18/3) [5/0]
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  SIP >= 3407694231
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  SIP < 3528093423.5 : NONP2P (9/0) [1/0]
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  SIP >= 3528093423.5 : P2P (4/1) [1/0]
|  |  |  |  |  |  |  |  |  |  |  DIP >= 2306493857 : P2P (12/0) [4/0]
|  |  |  |  |  |  |  |  |  |  DIP >= 2306503345.5 : NONP2P (24/0) [12/0]
|  |  |  |  |  |  |  |  |  DIP >= 2309014449
|  |  |  |  |  |  |  |  |  |  SIP < 2306493529 : P2P (17/0) [12/0]
|  |  |  |  |  |  |  |  |  |  SIP >= 2306493529
|  |  |  |  |  |  |  |  |  |  |  SIP < 2306493660 : NONP2P (2/0) [3/1]
|  |  |  |  |  |  |  |  |  |  |  SIP >= 2306493660 : P2P (5/0) [6/0]
|  |  |  |  |  |  |  |  DIP >= 3262743942.5 : NONP2P (10/0) [3/0]
|  |  |  |  |  |  |  DIP >= 3314579613.5 : P2P (27/0) [17/0]
|  |  |  |  DIP >= 3421712537.5
|  |  |  |  |  Len < 42 : NONP2P (101/2) [52/5]
|  |  |  |  |  Len >= 42
|  |  |  |  |  |  DIP < 3478810233.5 : NONP2P (9/0) [3/0]
|  |  |  |  |  |  DIP >= 3478810233.5 : P2P (8/1) [6/1]
|  |  |  DIP >= 3519157672.5 : P2P (71/0) [26/0]
|  Len >= 48.5
|  |  Len < 59.5 : P2P (396/6) [198/5]
|  |  Len >= 59.5
|  |  |  Len < 190
|  |  |  |  SIP < 2306491626
|  |  |  |  |  DIP < 2306490750.5
|  |  |  |  |  |  SIP < 2306491390
|  |  |  |  |  |  |  SIP < 2306490549.5
|  |  |  |  |  |  |  |  DIP < 2306490549.5 : NONP2P (13/0) [8/0]
|  |  |  |  |  |  |  |  DIP >= 2306490549.5 : P2P (8/2) [3/0]
|  |  |  |  |  |  |  SIP >= 2306490549.5 : P2P (45/4) [15/0]
|  |  |  |  |  |  SIP >= 2306491390 : NONP2P (8/0) [3/0]
|  |  |  |  |  DIP >= 2306490750.5
|  |  |  |  |  |  DIP < 2640255613
|  |  |  |  |  |  |  SIP < 1166215728
|  |  |  |  |  |  |  |  SIP < 1164020587 : P2P (73/5) [33/0]
|  |  |  |  |  |  |  |  SIP >= 1164020587 : NONP2P (2/0) [2/0]
|  |  |  |  |  |  |  SIP >= 1166215728 : P2P (89/0) [43/0]
|  |  |  |  |  |  DIP >= 2640255613
|  |  |  |  |  |  |  SIP < 2306490572.5 : NONP2P (3/0) [4/0]
|  |  |  |  |  |  |  SIP >= 2306490572.5 : P2P (8/0) [4/1]
|  |  |  |  SIP >= 2306491626
|  |  |  |  |  SIP < 3061826205.5 : P2P (229/2) [113/3]
|  |  |  |  |  SIP >= 3061826205.5
|  |  |  |  |  |  DIP < 2306490531.5 : NONP2P (4/0) [1/0]
|  |  |  |  |  |  DIP >= 2306490531.5 : P2P (49/4) [18/1]
|  |  |  Len >= 190
|  |  |  |  SIP < 2218301896.5 : P2P (82/3) [48/2]
|  |  |  |  SIP >= 2218301896.5
|  |  |  |  |  DIP < 1857224279
|  |  |  |  |  |  SIP < 2306491553.5
|  |  |  |  |  |  |  SIP < 2306491381 : P2P (12/1) [4/0]
|  |  |  |  |  |  |  SIP >= 2306491381 : NONP2P (8/0) [6/0]
|  |  |  |  |  |  SIP >= 2306491553.5 : P2P (50/1) [22/2]
|  |  |  |  |  DIP >= 1857224279
|  |  |  |  |  |  SIP < 3481498892
|  |  |  |  |  |  |  DIP < 3520370787.5
|  |  |  |  |  |  |  |  SIP < 3393577760.5
```

```
|  |  |  |  |  |  |  |  |  |  SIP < 3033324314
|  |  |  |  |  |  |  |  |  |  |  Len < 311.5
|  |  |  |  |  |  |  |  |  |  |  |  DIP < 2306493554 : NONP2P (12/1) [8/3]
|  |  |  |  |  |  |  |  |  |  |  |  DIP >= 2306493554 : P2P (8/2) [7/2]
|  |  |  |  |  |  |  |  |  |  |  Len >= 311.5 : NONP2P (10/0) [7/0]
|  |  |  |  |  |  |  |  |  |  SIP >= 3033324314 : P2P (5/0) [1/0]
|  |  |  |  |  |  |  |  |  SIP >= 3393577760.5 : NONP2P (13/0) [6/1]
|  |  |  |  |  |  |  |  DIP >= 3520370787.5 : P2P (8/0) [1/0]
|  |  |  |  |  |  |  SIP >= 3481498892 : P2P (10/0) [4/0]
Len >= 407.5
|  SIP < 3418667646.5
|  |  Len < 1304
|  |  |  SIP < 2306492215.5
|  |  |  |  SIP < 1848053299.5
|  |  |  |  |  SIP < 1166252826.5
|  |  |  |  |  |  SIP < 745730655.5 : P2P (22/1) [7/0]
|  |  |  |  |  |  SIP >= 745730655.5
|  |  |  |  |  |  |  SIP < 1159027017
|  |  |  |  |  |  |  |  SIP < 1081476372 : NONP2P (14/0) [9/1]
|  |  |  |  |  |  |  |  SIP >= 1081476372
|  |  |  |  |  |  |  |  |  DIP < 2306492322.5
|  |  |  |  |  |  |  |  |  |  DIP < 2306491906.5
|  |  |  |  |  |  |  |  |  |  |  SIP < 1093444604.5 : P2P (3/0) [0/0]
|  |  |  |  |  |  |  |  |  |  |  SIP >= 1093444604.5
|  |  |  |  |  |  |  |  |  |  |  |  SIP < 1106489840.5 : NONP2P (2/0) [3/0]
|  |  |  |  |  |  |  |  |  |  |  |  SIP >= 1106489840.5 : P2P (3/0) [1/0]
|  |  |  |  |  |  |  |  |  |  DIP >= 2306491906.5 : NONP2P (14/0) [9/2]
|  |  |  |  |  |  |  |  |  DIP >= 2306492322.5
|  |  |  |  |  |  |  |  |  |  SIP < 1114434430.5 : NONP2P (2/0) [1/0]
|  |  |  |  |  |  |  |  |  |  SIP >= 1114434430.5 : P2P (11/0) [4/0]
|  |  |  |  |  |  |  SIP >= 1159027017 : NONP2P (46/0) [29/0]
|  |  |  |  |  SIP >= 1166252826.5
|  |  |  |  |  |  DIP < 2306493496 : P2P (77/0) [37/0]
|  |  |  |  |  |  DIP >= 2306493496
|  |  |  |  |  |  |  DIP < 2306493772.5 : NONP2P (17/0) [5/0]
|  |  |  |  |  |  |  DIP >= 2306493772.5 : P2P (3/0) [1/0]
|  |  |  |  SIP >= 1848053299.5
|  |  |  |  |  DIP < 3520467590
|  |  |  |  |  |  DIP < 1881153193.5
|  |  |  |  |  |  |  DIP < 1166252826.5 : NONP2P (49/4) [22/0]
|  |  |  |  |  |  |  DIP >= 1166252826.5 : P2P (15/0) [10/0]
|  |  |  |  |  |  DIP >= 1881153193.5
|  |  |  |  |  |  |  SIP < 2306491462.5 : NONP2P (64/0) [23/1]
|  |  |  |  |  |  |  SIP >= 2306491462.5
|  |  |  |  |  |  |  |  Len < 990.5 : NONP2P (10/0) [2/0]
|  |  |  |  |  |  |  |  Len >= 990.5 : P2P (3/0) [2/0]
|  |  |  |  |  DIP >= 3520467590 : P2P (5/0) [0/0]
|  |  |  SIP >= 2306492215.5
|  |  |  |  DIP < 2306492594
|  |  |  |  |  Len < 471
|  |  |  |  |  |  Len < 426.5 : P2P (4/0) [4/1]
|  |  |  |  |  |  Len >= 426.5 : NONP2P (11/1) [14/3]
|  |  |  |  |  Len >= 471
|  |  |  |  |  |  DIP < 1278942564.5
|  |  |  |  |  |  |  SIP < 2306493154.5 : P2P (26/0) [12/0]
|  |  |  |  |  |  |  SIP >= 2306493154.5
|  |  |  |  |  |  |  |  Len < 1100
```

```
|  |  |  |  |  |  |  |  |  |  Len < 899 : P2P (5/0) [3/0]
|  |  |  |  |  |  |  |  |  |  Len >= 899 : NONP2P (3/0) [2/0]
|  |  |  |  |  |  |  |  |  Len >= 1100 : P2P (7/0) [6/0]
|  |  |  |  |  |  |  DIP >= 1278942564.5 : P2P (65/0) [37/2]
|  |  |  |  |  DIP >= 2306492594
|  |  |  |  |  |  DIP < 2306492991 : NONP2P (21/0) [13/0]
|  |  |  |  |  |  DIP >= 2306492991
|  |  |  |  |  |  |  SIP < 2306493251
|  |  |  |  |  |  |  |  SIP < 2306492561.5 : P2P (2/0) [1/0]
|  |  |  |  |  |  |  |  SIP >= 2306492561.5 : NONP2P (15/0) [7/1]
|  |  |  |  |  |  |  SIP >= 2306493251 : P2P (29/2) [21/4]
|  |  |  Len >= 1304
|  |  |  |  SIP < 2306493498.5
|  |  |  |  |  SIP < 1102583940.5
|  |  |  |  |  |  SIP < 529669792 : P2P (17/0) [16/0]
|  |  |  |  |  |  SIP >= 529669792
|  |  |  |  |  |  |  DIP < 2306492518
|  |  |  |  |  |  |  |  SIP < 868433754.5 : NONP2P (2/0) [2/0]
|  |  |  |  |  |  |  |  SIP >= 868433754.5 : P2P (4/0) [3/0]
|  |  |  |  |  |  |  DIP >= 2306492518 : NONP2P (4/0) [1/0]
|  |  |  |  |  SIP >= 1102583940.5
|  |  |  |  |  |  SIP < 2306493070.5 : P2P (159/0) [72/0]
|  |  |  |  |  |  SIP >= 2306493070.5
|  |  |  |  |  |  |  SIP < 2306493197 : NONP2P (5/0) [4/0]
|  |  |  |  |  |  |  SIP >= 2306493197 : P2P (35/0) [20/0]
|  |  |  |  SIP >= 2306493498.5
|  |  |  |  |  SIP < 2445769475.5
|  |  |  |  |  |  SIP < 2306503345.5
|  |  |  |  |  |  |  SIP < 2306493702 : NONP2P (4/0) [2/0]
|  |  |  |  |  |  |  SIP >= 2306493702 : P2P (10/0) [7/0]
|  |  |  |  |  |  SIP >= 2306503345.5 : NONP2P (37/0) [22/0]
|  |  |  |  SIP >= 2445769475.5 : P2P (26/0) [13/0]
|  SIP >= 3418667646.5
|  |  SIP < 3501283567 : NONP2P (115/1) [59/3]
|  |  SIP >= 3501283567 : P2P (28/2) [9/0]

Size of the tree : 225

Time taken to build model: 0.05 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        4867               94.7809 %
Incorrectly Classified Instances      268                5.2191 %
Kappa statistic                          0.8681
Mean absolute error                      0.0773
Root mean squared error                  0.2161
Relative absolute error                 19.3791 %
Root relative squared error             48.3884 %
Total Number of Instances             5135

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   Class
                 0.892    0.031     0.916     0.892     0.904     NONP2P
                 0.969    0.108     0.96      0.969     0.964     P2P
```

=== Confusion Matrix ===

```
   a    b   <-- classified as
1260  152 |    a = NONP2P
 116 3607 |    b = P2P
```

C. Run information based on 11436 records and 10 fold cross validation

=== Run information ===

Scheme: **weka.classifiers.trees.REPTree -M 2 -V 0.0010 -N 3 -S 1 -L -1**
Relation: ReducedP2PFile3attributes
Instances: 11436
Attributes: 4
Len
SIP
DIP
Type
Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

REPTree
==========

```
SIP < 3421367522.5
|  DIP < 3421712537.5
|  |  SIP < 1166215728
|  |  |  SIP < 1159161522.5
|  |  |  |  DIP < 2306493656.5
|  |  |  |  |  SIP < 1052135062
|  |  |  |  |  |  SIP < 77738505 : NONP2P (4/0) [1/0]
|  |  |  |  |  |  SIP >= 77738505
|  |  |  |  |  |  |  DIP < 2306491919
|  |  |  |  |  |  |  |  DIP < 2306491819 : P2P (71/0) [42/0]
|  |  |  |  |  |  |  |  DIP >= 2306491819
|  |  |  |  |  |  |  |  |  SIP < 810840100.5 : NONP2P (5/1) [1/0]
|  |  |  |  |  |  |  |  |  SIP >= 810840100.5 : P2P (6/0) [1/0]
|  |  |  |  |  |  |  DIP >= 2306491919 : P2P (221/0) [117/0]
|  |  |  |  |  SIP >= 1052135062
|  |  |  |  |  |  SIP < 1078664924.5 : NONP2P (24/1) [16/2]
|  |  |  |  |  |  SIP >= 1078664924.5
|  |  |  |  |  |  |  DIP < 2306492215.5
|  |  |  |  |  |  |  |  DIP < 2306491936
|  |  |  |  |  |  |  |  |  Len < 48.5
|  |  |  |  |  |  |  |  |  |  SIP < 1150178939.5
|  |  |  |  |  |  |  |  |  |  |  SIP < 1102041052.5
|  |  |  |  |  |  |  |  |  |  |  |  SIP < 1093444604.5 : P2P (9/0) [3/1]
|  |  |  |  |  |  |  |  |  |  |  |  SIP >= 1093444604.5 : NONP2P (5/0) [3/0]
|  |  |  |  |  |  |  |  |  |  |  SIP >= 1102041052.5 : P2P (22/0) [11/0]
|  |  |  |  |  |  |  |  |  |  SIP >= 1150178939.5
|  |  |  |  |  |  |  |  |  |  |  SIP < 1153407337 : NONP2P (8/0) [2/0]
|  |  |  |  |  |  |  |  |  |  |  SIP >= 1153407337 : P2P (3/0) [3/0]
|  |  |  |  |  |  |  |  |  Len >= 48.5
```

```
|  |  |  |  |  |  |  |  |  |  |  SIP < 1101596529
|  |  |  |  |  |  |  |  |  |  |  |  SIP < 1093444604.5 : P2P (29/0) [18/2]
|  |  |  |  |  |  |  |  |  |  |  |  SIP >= 1093444604.5 : NONP2P (2/0) [4/0]
|  |  |  |  |  |  |  |  |  |  |  SIP >= 1101596529 : P2P (85/0) [40/0]
|  |  |  |  |  |  |  |  |  DIP >= 2306491936
|  |  |  |  |  |  |  |  |  |  SIP < 1147278875.5
|  |  |  |  |  |  |  |  |  |  |  DIP < 2306492163.5 : P2P (12/1) [8/0]
|  |  |  |  |  |  |  |  |  |  |  DIP >= 2306492163.5 : NONP2P (12/1) [7/2]
|  |  |  |  |  |  |  |  |  |  SIP >= 1147278875.5 : NONP2P (28/0) [13/0]
|  |  |  |  |  |  |  |  DIP >= 2306492215.5
|  |  |  |  |  |  |  |  |  SIP < 1095358362.5
|  |  |  |  |  |  |  |  |  |  SIP < 1093120705 : P2P (19/0) [6/0]
|  |  |  |  |  |  |  |  |  |  SIP >= 1093120705 : NONP2P (3/0) [3/0]
|  |  |  |  |  |  |  |  |  SIP >= 1095358362.5 : P2P (232/0) [104/0]
|  |  |  |  DIP >= 2306493656.5
|  |  |  |  |  Len < 1400
|  |  |  |  |  |  DIP < 2306493670.5 : NONP2P (15/0) [4/0]
|  |  |  |  |  |  DIP >= 2306493670.5
|  |  |  |  |  |  |  SIP < 1149220144 : P2P (72/3) [31/1]
|  |  |  |  |  |  |  SIP >= 1149220144
|  |  |  |  |  |  |  |  SIP < 1152913178 : NONP2P (8/0) [7/0]
|  |  |  |  |  |  |  |  SIP >= 1152913178 : P2P (3/0) [2/0]
|  |  |  |  |  Len >= 1400 : NONP2P (57/0) [24/0]
|  |  |  SIP >= 1159161522.5
|  |  |  |  DIP < 2306492318 : NONP2P (119/1) [64/1]
|  |  |  |  DIP >= 2306492318
|  |  |  |  |  SIP < 1159780607 : NONP2P (22/0) [8/0]
|  |  |  |  |  SIP >= 1159780607 : P2P (13/0) [11/0]
|  |  SIP >= 1166215728
|  |  |  DIP < 1166215728
|  |  |  |  DIP < 1149615580.5
|  |  |  |  |  DIP < 1054340094.5
|  |  |  |  |  |  DIP < 211690922.5 : NONP2P (6/1) [3/1]
|  |  |  |  |  |  DIP >= 211690922.5 : P2P (412/3) [213/0]
|  |  |  |  |  DIP >= 1054340094.5
|  |  |  |  |  |  DIP < 1123933515.5
|  |  |  |  |  |  |  Len < 421.5
|  |  |  |  |  |  |  |  SIP < 2306493655
|  |  |  |  |  |  |  |  |  DIP < 1122381453.5
|  |  |  |  |  |  |  |  |  |  DIP < 1080765137 : NONP2P (8/3) [8/3]
|  |  |  |  |  |  |  |  |  |  DIP >= 1080765137
|  |  |  |  |  |  |  |  |  |  |  SIP < 2306492215.5
|  |  |  |  |  |  |  |  |  |  |  |  SIP < 2306492175.5
|  |  |  |  |  |  |  |  |  |  |  |  |  SIP < 2306491621.5
|  |  |  |  |  |  |  |  |  |  |  |  |  |  SIP < 2306491542 : P2P (11/1) [12/1]
|  |  |  |  |  |  |  |  |  |  |  |  |  |  SIP >= 2306491542 : NONP2P (3/0) [1/0]
|  |  |  |  |  |  |  |  |  |  |  |  |  SIP >= 2306491621.5 : P2P (37/2) [22/0]
|  |  |  |  |  |  |  |  |  |  |  |  SIP >= 2306492175.5 : NONP2P (4/0) [2/0]
|  |  |  |  |  |  |  |  |  |  |  SIP >= 2306492215.5
|  |  |  |  |  |  |  |  |  |  |  |  DIP < 1094427601.5
|  |  |  |  |  |  |  |  |  |  |  |  |  DIP < 1093673338.5 : P2P (27/0) [13/0]
|  |  |  |  |  |  |  |  |  |  |  |  |  DIP >= 1093673338.5 : NONP2P (2/0) [2/0]
|  |  |  |  |  |  |  |  |  |  |  |  DIP >= 1094427601.5 : P2P (59/0) [36/0]
|  |  |  |  |  |  |  |  |  DIP >= 1122381453.5 : NONP2P (12/3) [5/2]
|  |  |  |  |  |  |  |  SIP >= 2306493655
|  |  |  |  |  |  |  |  |  SIP < 2306493670.5 : NONP2P (15/0) [12/0]
|  |  |  |  |  |  |  |  |  SIP >= 2306493670.5 : P2P (5/0) [4/1]
```

```
|  |  |  |  |  |  |  |  Len >= 421.5
|  |  |  |  |  |  |  |  |  DIP < 1095379598.5
|  |  |  |  |  |  |  |  |  |  SIP < 2306492957
|  |  |  |  |  |  |  |  |  |  |  SIP < 2306492322 : NONP2P (10/0) [9/0]
|  |  |  |  |  |  |  |  |  |  |  SIP >= 2306492322 : P2P (2/0) [2/0]
|  |  |  |  |  |  |  |  |  |  SIP >= 2306492957 : NONP2P (18/0) [14/0]
|  |  |  |  |  |  |  |  |  DIP >= 1095379598.5
|  |  |  |  |  |  |  |  |  |  Len < 594 : NONP2P (6/1) [5/1]
|  |  |  |  |  |  |  |  |  |  Len >= 594 : P2P (6/0) [3/0]
|  |  |  |  |  |  |  DIP >= 1123933515.5 : P2P (222/0) [113/0]
|  |  |  |  DIP >= 1149615580.5
|  |  |  |  |  DIP < 1165811022
|  |  |  |  |  |  DIP < 1152742633.5
|  |  |  |  |  |  |  DIP < 1150982871.5 : NONP2P (62/0) [24/0]
|  |  |  |  |  |  |  DIP >= 1150982871.5
|  |  |  |  |  |  |  |  SIP < 2306492525.5 : NONP2P (24/0) [16/0]
|  |  |  |  |  |  |  |  SIP >= 2306492525.5 : P2P (5/0) [1/0]
|  |  |  |  |  |  DIP >= 1152742633.5
|  |  |  |  |  |  |  DIP < 1159161522.5 : P2P (125/0) [52/0]
|  |  |  |  |  |  |  DIP >= 1159161522.5
|  |  |  |  |  |  |  |  DIP < 1159782702 : NONP2P (49/0) [30/0]
|  |  |  |  |  |  |  |  DIP >= 1159782702 : P2P (24/1) [7/1]
|  |  |  |  |  DIP >= 1165811022 : NONP2P (76/0) [41/0]
|  |  |  DIP >= 1166215728
|  |  |  |  DIP < 2306492727
|  |  |  |  |  SIP < 2306493505.5
|  |  |  |  |  |  SIP < 2306491070
|  |  |  |  |  |  |  DIP < 2306490786
|  |  |  |  |  |  |  |  DIP < 2195311476
|  |  |  |  |  |  |  |  |  |  SIP < 2306490541.5 : NONP2P (3/0) [2/0]
|  |  |  |  |  |  |  |  |  |  SIP >= 2306490541.5 : P2P (77/2) [27/0]
|  |  |  |  |  |  |  |  |  DIP >= 2195311476
|  |  |  |  |  |  |  |  |  |  SIP < 1904626668.5
|  |  |  |  |  |  |  |  |  |  |  DIP < 2306490541.5 : NONP2P (3/0) [2/0]
|  |  |  |  |  |  |  |  |  |  |  DIP >= 2306490541.5 : P2P (33/2) [16/0]
|  |  |  |  |  |  |  |  |  |  SIP >= 1904626668.5 : NONP2P (18/0) [10/0]
|  |  |  |  |  |  |  |  DIP >= 2306490786 : P2P (774/0) [403/1]
|  |  |  |  |  |  SIP >= 2306491070
|  |  |  |  |  |  |  DIP < 2229478464.5 : P2P (1388/2) [688/2]
|  |  |  |  |  |  |  DIP >= 2229478464.5 : NONP2P (3/1) [5/0]
|  |  |  |  |  SIP >= 2306493505.5
|  |  |  |  |  |  SIP < 2306493554
|  |  |  |  |  |  |  DIP < 1845395807.5
|  |  |  |  |  |  |  |  DIP < 1393231290
|  |  |  |  |  |  |  |  |  DIP < 1383777429.5
|  |  |  |  |  |  |  |  |  |  DIP < 1293794346 : NONP2P (5/0) [1/0]
|  |  |  |  |  |  |  |  |  |  DIP >= 1293794346 : P2P (8/0) [3/0]
|  |  |  |  |  |  |  |  |  DIP >= 1383777429.5 : NONP2P (10/0) [3/0]
|  |  |  |  |  |  |  |  DIP >= 1393231290 : P2P (5/0) [2/0]
|  |  |  |  |  |  |  DIP >= 1845395807.5 : NONP2P (16/0) [10/0]
|  |  |  |  |  |  SIP >= 2306493554 : P2P (346/4) [155/5]
|  |  |  |  DIP >= 2306492727
|  |  |  |  |  DIP < 2306492840
|  |  |  |  |  |  SIP < 1894980908.5 : P2P (40/0) [24/0]
|  |  |  |  |  |  SIP >= 1894980908.5
|  |  |  |  |  |  |  Len < 48
|  |  |  |  |  |  |  |  SIP < 2872000029 : P2P (2/0) [2/0]
```

```
|  |  |  |  |  |  |  |  |  SIP >= 2872000029 : NONP2P (4/0) [2/1]
|  |  |  |  |  |  |  |  Len >= 48 : NONP2P (81/0) [31/1]
|  |  |  |  |  |  DIP >= 2306492840
|  |  |  |  |  |  |  DIP < 2306493505.5
|  |  |  |  |  |  |  |  SIP < 2170167615.5 : P2P (338/0) [156/0]
|  |  |  |  |  |  |  |  SIP >= 2170167615.5
|  |  |  |  |  |  |  |  |  SIP < 2268624138.5 : NONP2P (3/0) [2/0]
|  |  |  |  |  |  |  |  |  SIP >= 2268624138.5 : P2P (181/1) [115/0]
|  |  |  |  |  |  |  DIP >= 2306493505.5
|  |  |  |  |  |  |  |  DIP < 2306493554 : NONP2P (61/13) [22/1]
|  |  |  |  |  |  |  |  DIP >= 2306493554
|  |  |  |  |  |  |  |  |  Len < 40.5
|  |  |  |  |  |  |  |  |  |  SIP < 2306492641.5 : P2P (58/1) [19/0]
|  |  |  |  |  |  |  |  |  |  SIP >= 2306492641.5
|  |  |  |  |  |  |  |  |  |  |  SIP < 2306492901.5 : NONP2P (39/2) [16/0]
|  |  |  |  |  |  |  |  |  |  |  SIP >= 2306492901.5
|  |  |  |  |  |  |  |  |  |  |  |  DIP < 2704284295.5 : P2P (29/0) [19/0]
|  |  |  |  |  |  |  |  |  |  |  |  DIP >= 2704284295.5
|  |  |  |  |  |  |  |  |  |  |  |  |  DIP < 3038174338.5 : NONP2P (3/0) [3/1]
|  |  |  |  |  |  |  |  |  |  |  |  |  DIP >= 3038174338.5 : P2P (17/0) [1/0]
|  |  |  |  |  |  |  |  Len >= 40.5
|  |  |  |  |  |  |  |  |  Len < 644
|  |  |  |  |  |  |  |  |  |  SIP < 2208783418.5 : P2P (138/0) [74/0]
|  |  |  |  |  |  |  |  |  |  SIP >= 2208783418.5
|  |  |  |  |  |  |  |  |  |  |  SIP < 2306490554 : NONP2P (3/0) [2/0]
|  |  |  |  |  |  |  |  |  |  |  SIP >= 2306490554 : P2P (325/3) [182/2]
|  |  |  |  |  |  |  |  |  Len >= 644
|  |  |  |  |  |  |  |  |  |  Len < 1310
|  |  |  |  |  |  |  |  |  |  |  SIP < 2306492641.5 : P2P (13/0) [2/0]
|  |  |  |  |  |  |  |  |  |  |  SIP >= 2306492641.5
|  |  |  |  |  |  |  |  |  |  |  |  SIP < 2306492901.5 : NONP2P (12/0) [9/0]
|  |  |  |  |  |  |  |  |  |  |  |  SIP >= 2306492901.5
|  |  |  |  |  |  |  |  |  |  |  |  |  Len < 1251.5 : P2P (9/0) [6/0]
|  |  |  |  |  |  |  |  |  |  |  |  |  Len >= 1251.5 : NONP2P (5/0) [1/0]
|  |  |  |  |  |  |  |  |  Len >= 1310 : P2P (107/0) [31/0]
|  DIP >= 3421712537.5
|  |  DIP < 3519157384
|  |  |  DIP < 3478810233.5
|  |  |  |  DIP < 3455259322.5 : NONP2P (94/0) [36/0]
|  |  |  |  DIP >= 3455259322.5
|  |  |  |  |  DIP < 3469822584.5 : P2P (6/0) [3/0]
|  |  |  |  |  DIP >= 3469822584.5 : NONP2P (81/0) [45/0]
|  |  |  DIP >= 3478810233.5
|  |  |  |  SIP < 2306491697.5
|  |  |  |  |  DIP < 3487225266
|  |  |  |  |  |  SIP < 2306491393.5 : P2P (9/0) [5/0]
|  |  |  |  |  |  SIP >= 2306491393.5 : NONP2P (12/0) [11/0]
|  |  |  |  |  DIP >= 3487225266 : NONP2P (59/0) [16/0]
|  |  |  |  SIP >= 2306491697.5
|  |  |  |  |  SIP < 2306493345 : P2P (34/0) [21/1]
|  |  |  |  |  SIP >= 2306493345 : NONP2P (7/1) [9/0]
|  |  DIP >= 3519157384 : P2P (277/1) [146/0]
SIP >= 3421367522.5
|  SIP < 3523705594.5
|  |  Len < 1452
|  |  |  SIP < 3486399946.5 : NONP2P (138/5) [48/3]
|  |  |  SIP >= 3486399946.5
```

```
|   |   |   |   SIP < 3487225266 : P2P (26/0) [12/0]
|   |   |   |   SIP >= 3487225266
|   |   |   |   |   DIP < 2306493345
|   |   |   |   |   |   SIP < 3501283567 : NONP2P (4/0) [2/0]
|   |   |   |   |   |   SIP >= 3501283567 : P2P (9/0) [8/0]
|   |   |   |   |   DIP >= 2306493345 : NONP2P (9/1) [5/0]
|   |   Len >= 1452 : NONP2P (131/0) [82/0]
|   SIP >= 3523705594.5 : P2P (232/1) [113/0]

Size of the tree : 209

Time taken to build model: 0.13 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11243               98.3123 %
Incorrectly Classified Instances       193                1.6877 %
Kappa statistic                          0.9437
Mean absolute error                      0.0266
Root mean squared error                  0.1246
Relative absolute error                  8.8459 %
Root relative squared error             32.1435 %
Total Number of Instances            11436

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   Class
                 0.95      0.009      0.958      0.95      0.954    NONP2P
                0.991       0.05      0.989     0.991       0.99    P2P

=== Confusion Matrix ===

    a    b   <-- classified as
 2002  105 |    a = NONP2P
   88 9241 |    b = P2P
```