Fall 12-1-2012

Identification of TCP Protocols

Juan Shao
University of Nebraska-Lincoln, jshao111@gmail.com

IDENTIFICATION OF TCP PROTOCOLS

by

Juan Shao

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Master of Science

Major: Computer Science

Under the Supervision of Professor Lisong Xu and Professor Dong Wang

Lincoln, Nebraska

December, 2012

IDENTIFICATION OF TCP PROTOCOLS

Juan Shao, M. S.

University of Nebraska, 2012

Adviser: Lisong Xu

Recently, many new TCP algorithms, such as BIC, CUBIC, and CTCP, have been deployed in the Internet. Investigating the deployment statistics of these TCP algorithms is meaningful to study the performance and stability of the Internet. Currently, there is a tool named Congestion Avoidance Algorithm Identification (CAAI) for identifying the TCP algorithm of a web server and then for investigating the TCP deployment statistics. However, CAAI using a simple k-NN algorithm can not achieve a high identification accuracy. In this thesis, we comprehensively study the identification accuracy of five popular machine learning models. We find that the random forest model achieves the highest identification accuracy among these five models, and its identification accuracy is much higher than that of CAAI.

ACKNOWLEDGMENTS

I would like to express my appreciation to my advisory committee: Professor Lisong Xu, Professor Dong Wang, Professor Ying Lu and Professor Byrav Ramamurthy. Special thank to my advisor Dr. Lisong Xu for his expertise and patience. It is he who taught me how to find problems and solve problems which will benefit my whole life. We had many discussions on this work and he was always open, helpful and patient. This work would not have been done without his guidance. It is my honor to work in his group. Also I thank my co-advisor Dr. Dong Wang who gives me many suggestions to spark new ideas. I thank Professor Ying Lu and Professor Byrav Ramamurthy for reviewing my thesis and serving on my masters thesis defense committee.

Furthermore, during this work, many people gave me good suggestions and technical help, I want to thank them here. I would like to thank Mr. Peng Yang who helps me in collecting the raw data and understanding the details of CAAI. His help greatly facilitates my work. I would like to thank all the people who always accompany me and support me. They are my families, professors, CSE staff, my friends and my classmates. My final words go to my parents who always support me and give me confidence.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, first we introduce TCP (Transmission Control Protocol), one of the most widely deployed protocols in the Internet, and various TCP algorithms. Then we show the importance of conducting census of TCP algorithm variants in the Internet. Furthermore we introduce CAAI, the current most powerful TCP identification tool for TCP census. Also, we give a simple introduction of several popular machine learning techniques or models which are used in our thesis. At last, we summarize our contributions.

1.1 TCP and Congestion Avoidance Algorithms

TCP, as one of the major components of Internet Protocol Suites (TCP/IP), has been widely used in all kinds of web applications such as Email, webpage browsing and file download due to its capability to provide reliable and sequential end-to-end data transfer. TCP maintains a congestion window (cwnd) for each connection. By continuously adjusting the cwnd size and detecting packet loss, TCP tries to probe the available bandwidth along the path and controls the data rate entering the network to avoid triggering network congestion [1].

TCP is a complicated algorithm, which consists of several sub-algorithms, including

congestion avoidance algorithm, slow start algorithm, loss recovery algorithm, and so on. Among these algorithms, TCP congestion avoidance algorithm is the most important one and will be referred to as TCP algorithm in the rest of this thesis. It deals with how to increase and decrease the cwnd size to rapidly adapt to the current available bandwidth. The originally designed TCP algorithm is AIMD (Additive-Increase-Multiplicative-Decrease). Later with the development of the Internet, many new TCP algorithms are developed and deployed in the Internet in order to not only better exploit the high bandwidth of Internet links but also adapt to various new network environments.

A survey of TCP algorithm census in the current Internet is very important for the following two major reasons. First, it helps us in designing new TCP algorithms and designing other congestion control algorithms for the future Internet. Second, it helps us in setting the optimal parameters for the Internet devices, such as the buffer capacity of Internet routers. In order to conduct a survey of TCP algorithm census, one important and necessary step is to develop a tool which can identify the TCP algorithm of a web server.

1.2 CAAI and Identification of TCP Algorithms

There are very few works on identifying the TCP algorithm of a web server by using machine learning methods. CAAI (Congestion Avoidance Algorithm Identification) is the most powerful one, and it "can identify all default TCP algorithms (i.e. AIMD, BIC, CUBIC and CTCP) and most non-default TCP algorithms of major operating system families" [2].

The basic idea of CAAI is to distinguish among different TCP algorithms using a pair of TCP features: multiplicative decrease parameter and window growth function which will be explained in Chapter 2. For each TCP algorithm, CAAI obtains its unique pair of features from the local web server in a lab testbed, which is referred to as a pair of training features. Then, for a remote web server in the Internet, CAAI obtains the pair of features of the web

server, and compares it with all pairs of training features. CAAI finds the most similar pair of training features, and reports the corresponding TCP algorithm as the identification result. The most important contribution of CAAI is that it is the first one that can identify almost all TCP algorithms. However, the weakness of CAAI is that its identification accuracy is not very high. The fundamental reason is that CAAI uses the simple nearest neighbor algorithm to classify the TCP algorithm of a web server. Therefore, in the thesis, we will investigate the identification accuracy of other more powerful machine learning models, which are briefly summarized in the next section.

1.3 Introduction of Machine Learning Models

Machine learning is a generalization or prediction process of automatically discovering the patterns, characteristics or relationships of some given samples (training data) and automatically predicting the classes of some new cases or making correct actions on the new inputs (test data).

CAAI uses the nearest neighbor algorithm, which is a special case of the k-NN (k-Nearest Neighbor)algorithm with $k = 1$. Besides the k-NN algorithm, many other machine learning models have been developed and deployed such as Decision Tree (DT) models, Random Forest (RF) models, Artificial Neural Network (ANN) models, Naive Bayes (NB) models and Support Vector Machine (SVM) models, etc. However, there is no universal machine learning model which fits all applications. Each of these models has its own advantage, disadvantage and specific fitting application scenario. In the following, I provide a simple introduction of these popular machine learning models.

A DT model is a tree-shape model in which each non-leaf node represents a feature. An RF model is a forest model which is composed of multiple sub decision trees. An RF model makes the final prediction decision based on the mode voting of its sub decision

trees. An ANN model is a multi-layer connection structure based model which simulates the biological neuron system. “In most cases a neural network is an adaptive system that changes its structure during a learning phase” [3]. A SVM model is a space and hyperplane based classification model in which all the examples are represented as the points in the space and the features comprises a hyperplane. The target of a SVM model is to find out a hyperplane to “best” separate the space points. An NB model “is a simple probabilistic classifier based on applying Bayes’ theorem with strong (naive) independence assumptions” [4]. An NB model assumes that “the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature, given the class variable” [4].

1.4 Contribution

In this thesis, we look into the TCP protocol identification problem, and make the following contributions.

- We study and compare the identification accuracy of two different methods to extract the feature vectors. The first method is proposed by CAAI, which extracts two TCP features from the raw TCP cwnd data. The second method is to directly use the original raw TCP cwnd data. We find that the second method achieves higher identification accuracy than the first one.

- We study and compare the identification accuracy of five popular machine learning models including the DT model, RF model, ANN model, NB model, and SVM model. We find that the RF model achieves the highest identification accuracy. Its accuracy is 97% with the first feature extract method and is 94% with the second one. Both are much higher than the accuracy of CAAI (about 70%).

Chapter 2

Background and Related Work

In this chapter, we discuss the TCP congestion avoidance algorithms, the previous research on TCP protocol identification, popular machine learning models and the machine learning applications in the network area.

2.1 TCP Congestion Avoidance Algorithms

In a TCP congestion control algorithm, a special congestion window (cwnd) is configured in the sender and the cwnd size controls the number of packets that can be simultaneously sent out in an RTT (Round Trip Time), i.e., the data sending rate. At the beginning of the transmission, TCP slowly probes the available network bandwidth and the cwnd is additively increased during uncongested periods and multiplicatively decreased upon congestion detection [5]. After detecting a congestion event, TCP halves its cwnd size and enters the "congestion avoidance" phase. The above procedure is the original TCP congestion control design, which is called AIMD (Additive-Increase-Multiplicative-Decrease) [6] [7]. AIMD halves the cwnd size upon detecting congestion and increases the cwnd size only by one for each successful received ACK, which increases too slowly and is wasteful for

a large congestion window. Thus, AIMD does not perform efficiently in high bandwidth links. Recently, a host of novel TCP protocols have been developed to adapt to the rapidly-increasing broadband networks and various network conditions. As a result, current TCP congestion avoidance algorithms have been evolved from homogeneous congestion control (the initial standard AIMD algorithm) to heterogeneous congestion control (AIMD, BIC [8], CUBIC [9], CTCP [10] and VEGAS [11], etc.) [2].

Coexistence of these different congestion avoidance algorithms on the same physical network infrastructure brings out the TCP protocols' fairness and friendness problems. Most of these new TCP protocols are designed to be friendly and work fairly with AIMD since they assume AIMD still dominates the Internet while that might not be the truth. TCP protocol census carried out by identifying the Internet web servers' TCP protocols not only helps in the performance evaluation of current TCP protocols but also helps in the design of new TCP protocols and the other TCP modules.

2.2 Related Work on TCP Protocol Identification

Many works have been done on identifying the TCP configurations of web servers. Oshio et al. [12] identify the TCP versions at a router by analyzing the statistic information of the packets passing through the router. Padhye et al. [13] develop a famous tool named TBIT (TCP Behavior Inference Tool) to identify the initial window size and the loss recovery components of a web server. Feyzabadi et al. [14] extend the TBIT by the active probing method, which is to actively send data to the web server and analyze the responses returning from the web server. Yang et al. [2] further extend the active probing mechanism and develop a very powerful TCP identification tool, called CAAI, which infers the servers' TCP configurations by analyzing the collected cwnd size traces of web servers. CAAI is the most related work, so below we discuss the detail procedures of the CAAI tool.

To collect the cwnd size traces, CAAI repeatedly conducts a sequence of file-downloading operations in a lab test bed. Basically, the lab test bed is composed of several computers connected in the same LAN: two web servers (one installed with Windows OS and one installed with Linux OS), a host machine and a software router. CAAI sets up two web servers because some TCP protocols only work under some specific OS (Windows or Linux). The web servers provide the file-downloading service and are separately configured by one of the fourteen TCP protocols. The software router is configured with some network parameters (such as the link delay, the data loss rate and the reordering rate) to emulate various network conditions. When the network parameters are all set to zero, CAAI emulates a perfect network condition; otherwise, CAAI emulates an imperfect network condition. For each file-downloading operation, a large-size file is downloaded from one of the pre-configured web servers to the host machine. During that process, a congestion avoidance event occurs and the corresponding cwnd size trace is recorded.

In CAAI, two special lab network environments (environment A and environment B) with specific settings for RTT and MSS of the web server are chosen to help generating TCP behaviors which could be easily distinguished. By deliberately delaying sending out DATA/ACK, CAAI guarantees that the operations in the our experiment follow the assigned sequences of RTTs. When all the operations finish, the raw data set (cwnd size traces) is collected.

Based on the collected cwnd size traces, CAAI extracts two types of features: β (multiplicative decrease parameter) "which determines the slow start threshold (i.e., the boundary congestion window size between the slow start and congestion avoidance states)" [2] and g(.) (window growth function) "which determines how a TCP algorithm grows its congestion window size in the congestion avoidance state" [2]. CAAI fits the curve of g(.) by a fifth-degree polynomial expression: $g(.) = \alpha_5 * x^5 + \alpha_4 * x^4 + \alpha_3 * x^3 + \alpha_2 * x^2 + \alpha_1 * x^1 + \alpha_0$. CAAI utilizes the coefficiencies of g(.) as the features, and thus, two six-tuple features,

$(\alpha_0^A, \alpha_1^A, \alpha_2^A, \alpha_3^A, \alpha_4^A, \alpha_5^A)$ and $(\alpha_0^B, \alpha_1^B, \alpha_2^B, \alpha_3^B, \alpha_4^B, \alpha_5^B)$ are separately extracted from the specific network environment A and environment B, respectively. Finally, for each operation, the extracted feature vector is a 14-tuple: $(\beta^A, \alpha_0^A, \alpha_1^A, \alpha_2^A, \alpha_3^A, \alpha_4^A, \alpha_5^A, \beta^B, \alpha_0^B, \alpha_1^B, \alpha_2^B, \alpha_3^B, \alpha_4^B, \alpha_5^B)$. CAAI extracts all the 14-tuple feature vectors as the whole data set for the future machine learning process.

CAAI deploys the k-NN algorithm on the whole data set. CAAI can identify a total of 14 TCP algorithms. CAAI stores its feature vectors extracted from the perfect network conditions (no reordering, no data loss, no duplication and no time delay) as the training set. CAAI stores the feature vectors of 14 TCP algorithms extracted from the imperfect network conditions (with at least one non-zero network parameter) as the validation data set. CAAI defines a special distance function and for each feature vector in the validation data set, CAAI calculates the distances between the feature vector in the validation data set and all the the feature vectors in the training set. The TCP algorithm of the training feature vector with the shortest distance is viewed as the TCP algorithm configured for that specific web server.

The nature of TCP identification actually is a machine learning process: given the behavior of a black box in some cases where the box's configurations is already known; predict the configurations of the black box in some other cases where the configuration information is hidden and is only allowed for observing its behavior.

In this thesis, based on the CAAI tools, we attempt to develop novel machine learning method based approaches for TCP protocol identification.

2.3 Machine Learning Techniques in Network Research

Machine learning techniques have been widely deployed in all kinds of research area which need classification, pattern recognition or automation of learning actions. Here, we

emphasize on introducing five popular machine learning models: the DT model, RF model , ANN model, SVM model and NB model.

A DT model is a tree-shape prediction model in which each non-leave node represents a specific condition (feature label), and each branch of that non-leave node represents different value of that feature; each leave node represents a specific class label. A path from the root to the leave node represents that the cases which satisfy the conjunctions of all the branched condition values belong to that class labeled in the leave node. DT is good at visually showing the decision-making factors and might have the over-fitting problem especially with too many features or training cases.

A RF model, as its name implied, is composed of a big number of trees and makes the final decision on the mode of all the decision trees. To construct each single decision tree of RF, it randomly samples some features and some cases to train and grow a decision tree, thus, each tree can be viewed as a specialist with some specific knowledge. For each test case, all the decision trees in the forest will give out their classification decisions (predicted class) based on their knowledge. The mode class of the predicted classes are viewed as the final class classification of RF. By constructing many simplified trees, RF avoids the over-fitting problem.

An ANN model is a connection based mathematical model. Normally in the model there are several layers such as the input layer, hidden layer and output layer which are laid out in sequence. Between each two neighbor layers there are some weighted connections which forward input impulse and feedback responses between the two layers and normally the hidden layer has a non-linear core function. Each time an ANN model is fed with a training case, the input signal will pass through all the layers and the related connections and generate the corresponding output, the predicted class. Then that predicted class is compared with the pre-known correct class. If they match, nothing needs to be changed; otherwise, the class difference will feed back and pass through all the layers, and during this

process, it could change the weights of some connections by some rules. It is kind of a self feedback system. With enough number of training cases, the ANN model might gradually come to a stable status in which the predicted class error is pretty small or the system will stop automatically after enough number of trained cases. At that time, the ANN model can be used for class prediction.

"An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on." [15]

"A naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions" [4]. One prerequisite of this method is that all the features are conditionally independent. "An advantage of the naive Bayes classifier is that it only requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification" [4].

Machine learning models have been widely deployed in the network research, such as to identify network traffic [16], [17], or to detect the specific network behavior (malware [18], spam [19], network anomaly intrusion [20] and DDoS (Distributed Denial of Service) attack [21], [22], etc.).

2.4 Summary

In this chapter, we introduced the previous TCP identification research work, several popular machine learning models and the related machine learning applications in the network research. Specifically, we discussed the technical details of CAAI tools which is the primary base of this thesis.

Chapter 3

Problem Statement and My Approach

In this chapter, we explain the specific problems to be solved in this thesis, and discuss our approach to solve the problems.

3.1 Problem Statement

This thesis aims at finding out the best machine learning model to classify the fourteen "mainstream" TCP protocols (BIC, COMPOUND, CTCP, CUBIC, CUBICB, HIGHSPEED, HTCP, ILLINOIS, RENO, SCALABLE, VEGAS, VENO, WESTWOOD, YEAH). In the Internet, not all the web servers are configured with the 14 "mainstream" TCP protocols that we investigate. Some web servers might be configured with rarely used or even unpublished TCP protocols which we call "unusual" TCP protocols. Extending the best machine learning model for classifying "unusual" TCP protocols are left as future work.

Finally, the problems that we target in this thesis can be stated as follows:

Given the cwnd size traces collected from the web servers which are configured with various TCP protocols, can we develop some machine learning model based methods which outperform the k-NN algorithm based CAAI method?

3.2 Typical Machine Learning Process

A typical machine learning process has three stages. The first stage is to collect the raw data set. The second stage is to preprocess the raw data (such as making up the missing data and data normalization) and to extract the features from the preprocessed data set. The third stage is to deploy the machine learning model on the feature set. First, the samples in the feature set are divided into two sub sets: a training set and a validation set. Then, the training set is used to train the model. After that, the established model validates each sample in the validation set and outputs the model's prediction accuracy.

There are two common methods to divide the feature set: The first method is to divide the whole feature set with a fixed ratio. For example, we can randomly select 90% of the samples in the feature set as the training set and use the remaining samples as the validation set. The second method is the n-fold cross-validation. In the n-fold cross-validation, the whole data set is evenly and randomly divided into n groups. Then we run the training and testing process for n rounds. In each round, a group of data is selected as the validation set while the other $n-1$ groups of data are served as the training set. Then the model runs on these two sets and outputs the predication accuracy for this round. For the n rounds, each group has one and only one opportunity to be selected as the validation group. In the end, the n rounds' prediction accuracies are averaged as the final prediction accuracy of the model.

3.3 My Approach

In this section, we describe the three stages of our proposed TCP protocol identification method which is based on machine learning models.

3.3.1 Raw data collection

We use CAAI to collect the raw cwnd data of a web server, which contains the TCP congestion window traces of the web server in two experiments: A and B. Experiments A and B are carefully designed by CAAI so that different TCP algorithms have different congestion window traces in either experiment A or B or both. For each experiment, CAAI collects the trace of TCP congestion window sizes under a TCP timeout for 35 round trip times (RTTs). The TCP timeout is one of the following four possible values: 60, 120, 240, and 480, and depends on the longest web page size of a web server. The value of 35 is chosen by CAAI so that different TCP algorithms have different congestion window sizes in at least one RTT.

3.3.2 Data preprocessing and features extraction

After the raw data is collected, we need to define the features and extract the features from the raw data set. But before that, we might need some data preprocessing work to deal with the data impurity such as the missing data and the noise data. The data preprocessing is related to the features to be extracted. Therefore, we firstly discuss our feature extraction methods, then we discuss the corresponding data preprocessing work. In this thesis, we study two feature extraction methods.

The first method directly uses the raw cwnd data as a features. We call these features raw cwnd features. Specifically, a raw cwnd feature vector contains the raw TCP congestion window sizes in the two experiments, the corresponding timeout value, and the corresponding TCP protocol. Therefore, a raw cwnd feature vector is a 72 tuple in the form of $(C_1^A, C_2^A, ..., C_{35}^A, C_1^B, C_2^B, ..., C_{35}^B, timeout, protocol)$, where C_i^A is the raw TCP congestion window size in RTT i of experiment A, and C_i^B is the raw TCP congestion window size in RTT i of experiment B. $timeout$ is one of the four values: 60, 120, 240, and 480.

Sometimes, CAAI could not collect the cwnd trace for 35 RTTs for an experiment. In this case, we remove that trace from the data set.

The second method is to use the features extracted by CAAI from the raw cwnd data. We call these features processed cwnd features. Specifically, a processed cwnd feature vector contains the smoothed TCP congestion window sizes and multiplicative decrease parameters in the two experiments, the corresponding timeout value, and the corresponding TCP protocol. Recall that a raw cwnd trace of an experiment contains 35 RTTs of data. Because all TCP algorithms have very similar first 20 RTTs of cwnd, CAAI uses only the last 15 RTTs of cwnd. Because a raw cwnd trace may have noise data, CAAI fits the last 15 RTTs of cwnd with a fifth-degree polynomial to obtain a smoothed cwnd trace. In addition, CAAI also extracts the multiplicative decrease parameter from the raw cwnd trace, which is an important TCP parameter. Therefore, a processed cwnd feature vector is a 34 tuple in the form of $(\beta^A, E_1^A, E_2^A, ..., E_{15}^A, \beta^B, E_1^B, E_2^B, ..., E_{15}^B, timeout, protocol)$, where β^A and β^B are the multiplicative decrease parameters of experiments A and B, respectively, E_i^A and E_i^B are the smoothed TCP congestion window sizes in RTT $20 + i$ of experiments A and B, respectively.

3.3.3 Apply machine learning models

After we obtain the feature vectors, we use the 10-fold cross validation method, which is described in Chapter 3.2, to divide all feature vectors into two sets: the training set and the validation set. Then we use five popular machine learning models on these feature vectors, including the DT model, RF model, ANN model, NB model, and SVM model. For each model, the training set is used to train the model, and then the trained model validates the validation set and outputs the prediction accuracy. Since we have two types of features: raw cwnd features and processed cwnd features. For each model, we run it for two types

of cwnd features. Therefore, we run a total of $5 \times 2 = 10$ experiments. In addition to these five models, we also run the k-NN algorithm which is used by CAAI, compare the prediction accuracy of these five models with that of k-NN algorithm, and finally select the best machine learning model with the highest predictions accuracy.

3.4 Summary

In this chapter, I discussed the statement of the problem that we target. Also I discussed the common machine learning process and the procedures of our machine learning model based approaches in details.

Chapter 4

Experiment and Result Analysis

In this chapter, we conduct the experiments as described in Chapter 3, and discuss the experiment results.

As described in Chapter 3, we conduct two groups of experiments using two types of features: raw cwnd features and processed cwnd features, respectively. For each group of experiments, we use five popular machine learning models with the 10-fold cross validation. Finally, we also run the k-NN algorithm of CAAI, and use it as a reference model.

The experimental results show that among the five machine learning models that we chose, the RF model performs best and our approach outperforms the k-NN algorithm based CAAI approach in terms of prediction accuracy.

4.1 Raw Data Collection for Experiments

We collect our raw cwnd data from the web servers in our lab testbed instead of Internet for the following reasons. For a web server in the Internet, we do not know its actual TCP algorithm, and therefore, we could not validate the results of our models. For a web server in our lab testbed, we know exactly its TCP algorithm, and therefore, we could validate the

results of our models.

A total of 44800 experiments were performed on the testbed, and correspondingly a total of 44800 raw cwnd traces were collected. Specifically, there were 14 web servers running 14 TCP algorithms, respectively. For each TCP algorithm (or each web sever), 4 timeout values were tested. For each combination of TCP algorithm and timeout, 400 random network conditions were emulated on the testbed. For each combination of TCP algorithm, timeout, and network condition, we conducted a pair of experiments A and B. Therefore, there were a total of $14 \times 4 \times 400 \times 2 = 44800$ experiments. This work was jointly conducted with Peng Yang.

4.2 Data Preprocessing and Feature Extraction

Below we describe how we extract two types of feature vectors, raw cwnd feature vectors and processed cwnd feature vectors, from the 44800 raw cwnd traces.

Recall that a raw cwnd feature vector contains the raw TCP congestion window traces in a pair of experiments A and B, the corresponding timeout value, and the corresponding TCP protocol. Specifically, a raw cwnd feature vector is a 72 tuple in the form of $(C_1^A, C_2^A, ..., C_{35}^A, C_1^B, C_2^B, ..., C_{35}^B, timeout, protocol)$. Since a raw cwnd feature vector contains the two raw cwnd traces of a pair of experiments A and B, a raw cwnd feature vector is created for each pair of experiments A and B. Therefore, for a total of 44800 raw cwnd traces, there should be a total of $44800/2 = 22400$ raw cwnd feature vectors. However, due to various reasons, such as poor network conditions, a raw cwnd trace file may contain less than 35 RTTs of data, and thus the corresponding raw cwnd feature vector could not be extracted. Finally, we successfully extracted a total of 17250 raw cwnd feature vectors.

Recall that a processed cwnd feature vector contains the smoothed TCP congestion window traces and multiplicative decrease parameters in a pair of experiments A and B, the corre-

sponding timeout value, and the corresponding TCP protocol. Specifically, a processed cwnd feature vector is a 34 tuple $(\beta^A, E_1^A, E_2^A, ..., E_{15}^A, \beta^B, E_1^B, E_2^B, ..., E_{15}^B, timeout, protocol)$. Since a processed cwnd feature vector contains the two smoothed cwnd traces and multiplicative decrease parameters of a pair of experiments A and B, a processed cwnd feature vector is created for each pair of experiments A and B. Therefore, for a total of 44800 raw cwnd traces, there should be a total of $44800/2 = 22400$ processed cwnd feature vectors. However, due to various reasons, such as poor network conditions, multiplicative decrease parameters or smoothed cwnd traces may not be successfully extracted from a raw cwnd trace, and thus the corresponding processed cwnd feature could not be extracted. Finally, we successfully extracted a total of 16989 processed cwnd feature vectors.

We note that the total number of raw cwnd feature vectors is slightly different from the total number of processed cwnd feature vectors. Because the difference is very small, we believe that it will not have any significant impact on our comparison between machine learning models using these two types of feature vectors.

4.3 Results of Machine Learning Models Using the Raw Cwnd Features

4.3.1 Models' training and validation

Instead of writing codes for all the machine learning models from scratch, it is common to apply the machine learning models by using some machine learning tools. Most popular machine learning models have been implemented in these machine learning tools (such as Weka [23], Matlab [24] and R language [25]) in the form of packages or functions. In our experiments, we utilize a popular open source machine learning tool, Weka (Waikato Environment for Knowledge Analysis), which is developed and maintained by the University

of Waikato in New Zealand. Weka not only implements most of the popular machine learning algorithms but also provides some other useful functionalities such as the data preprocessing and the feature visualization. Specifically, we used the Weka 3.7.5 developer version which supports all the five popular machine learning models that we chose. Also, Weka provides the option to use the n-fold cross-validation method. As the Weka default setting, we use the 10-fold cross-validation for our experiments.

In the following, we discuss the details of applying Weka machine learning models in our experiments. For the DT model, one early popular DT algorithm is the ID3 algorithm and later it is extended to the C4.5 algorithm. In our experiments, we apply the Weka J48 module which is a Java implementation of the C4.5 algorithm. For the RF model, we apply the Weka "RandomForest" module which is only supported by the Weka developer version. For the ANN model, we apply the Weka "MultilayerPerceptron" module which is a typical ANN model implementation. Running Weka "MultilayerPerceptron" module consumes quite a lot of memory which might exceed the default memory setting of the Java virtual machine and incur the "out of memory" exception. We launch the Weka by a special command to increase Java virtual machine's initial available memory as follows:

```
$ java -Xmx512m -classpath weka:wekaclassalgos.jar weka.gui.GUIChooser
```

For the NB and the SVM models, we separately apply the Weka "Naive Bayes" module and the Weka "LibSVM" module.

In our experiments, we wrote a program in Matlab to preprocess the raw data set, extract the features and transform the feature vectors into a Weka input file (.arff file). Then we use the Weka GUI interface to load the input file and separately apply the five machine learning models on it using the 10-fold cross-validation method. Depending on which model is running, Weka might take time from a few seconds to half an hour to complete the process. Finally, Weka outputs the statistical analysis of the final prediction results and the details

of the established models. We also run Weka to output the prediction results of all the validation feature vectors.

4.3.2 Results of using raw cwnd features

Fig. 4.1, Fig. 4.2, Fig. 4.3, Fig. 4.4 and Fig. 4.5 show the screenshots of the results of Weka DT, RF, ANN, SVM and NB modules using the raw cwnd features, respectively. Each result picture contains three parts. In the first part, the value of the "Correctly Classified Instance" indicates the prediction accuracy. For example, in Fig. 4.2, the "Total Number of Instances" is 17250. Among them, the number of "Correctly Classified Instances" is 16838 and the number of "Incorrectly Classified Instances" is 412. Thus, Weka calculates the prediction accuracy as 97.6116%. In the second part, the "Detailed Accuracy By Class" shows the credibility of each TCP protocol's prediction. The "TP Rate" (True Positive Rate) represents that among all the cases of protocol x, how many percentage of them are truly classified as protocol x. The "FP Rate" (False Positive Rate) represents that among all the cases which are not of protocol x, how many percentage of them are falsely classified as protocol x. The "Precision" represents the ratio of examples truly belong to protocol x to the number of examples which are classified as protocol x. The "recall" is the same as the "TP Rate". In Fig. 4.2, the "TP Rates" of all the protocols are pretty high which means the prediction result is precise. The third part is the "Confusion Matrix" in which the diagonals represent the correctly classified cases while the other elements in the matrix represent incorrectly classified cases. In Fig. 4.2, the diagonals dominate the matrix.

Table. 4.1 summarizes the prediction accuracy of five models. From which we can see that the RF model achieves the highest prediction accuracy (97.6116%), and it builds the model within a pretty short time (2.33 seconds). Thus, the RF model is viewed as the best model when using the raw cwnd features.

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

Correctly Classified Instances       16425               95.2174 %
Incorrectly Classified Instances       825                4.7826 %
Kappa statistic                          0.9481
Mean absolute error                      0.0075
Root mean squared error                  0.079
Relative absolute error                  5.7096 %
Root relative squared error             30.7931 %
Coverage of cases (0.95 level)          96.058  %
Mean rel. region size (0.95 level)       7.5217 %
Total Number of Instances            17250

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                 0.976    0.003    0.96       0.976   0.968      0.965    0.99      0.945     bic
                 0.971    0.004    0.957      0.971   0.964      0.96     0.986     0.957     compound
                 0.916    0.002    0.912      0.916   0.914      0.912    0.963     0.877     ctcp
                 0.97     0.003    0.969      0.97    0.97       0.967    0.989     0.95      cubic
                 0.943    0.001    0.953      0.943   0.948      0.947    0.981     0.94      cubicb
                 0.916    0.007    0.912      0.916   0.914      0.907    0.963     0.871     highspeed
                 0.92     0.005    0.94       0.92    0.93       0.923    0.967     0.899     htcp
                 0.92     0.005    0.943      0.92    0.932      0.925    0.968     0.906     illinois
                 0.927    0.009    0.918      0.927   0.922      0.914    0.974     0.914     reno
                 0.961    0.002    0.971      0.961   0.966      0.964    0.985     0.954     scalable
                 0.999    0          1        0.999   0.999      0.999    0.999     0.999     vegas
                 0.933    0.006    0.925      0.933   0.929      0.923    0.975     0.893     veno
                 0.999    0          0.998    0.999   0.998      0.998    0.999     0.996     westwood
                 0.958    0.004    0.954      0.958   0.956      0.953    0.984     0.938     yeah
Weighted Avg.    0.952    0.004    0.952      0.952   0.952      0.948    0.981     0.933

=== Confusion Matrix ===

    a    b    c    d    e    f    g    h    i    j    k    l    m    n   <-- classified as
 1286    0    0    5    3    0   19    0    0    4    0    0    0    0     a = bic
    0 1547    1    0    1   15    0    2   12    0    0   10    1    5     b = compound
    0    7  373    0    0    9    1    4    7    2    0    3    0    1     c = ctcp
    4    0    0 1327    3    0   34    0    0    0    0    0    0    0     d = cubic
    4    1    2    1  363    1    6    1    2    2    0    0    0    2     e = cubicb
    0   17   10    0    2 1158    5   13   46    1    0    9    0    3     f = highspeed
   41    1    3   34    3   10 1348    9    0    5    0    6    0    6     g = htcp
    0   11    8    2    2   16    8 1423   15   10    0   25    0   26     h = illinois
    0   20   10    0    0   40    0    8 1488    0    0   40    0    0     i = reno
    5    0    1    0    1    3    9    7    1 1086    0    2    0   15     j = scalable
    0    0    0    0    0    0    0    0    0    0 1365    0    2    0     k = vegas
    0    9    1    0    0   15    1   11   48    0    0 1177    0    0     l = veno
    0    0    0    0    0    0    0    0    1    0    0    0 1270    0     m = westwood
    0    3    0    0    3    3    3   31    1    8    0    1    0 1214     n = yeah
```

Figure 4.1: Weka output of DT using raw cwnd features

Table 4.1: Five models' prediction accuracy when using raw cwnd features

	DT	RF	ANN	SVM	NB
Prediction Accuracy (%)	95.2174	97.6116	75.5942	53.1072	44
Build-Model-Time (sec)	3.16	2.33	564.59	128.89	0.27

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

Correctly Classified Instances       16838               97.6116 %
Incorrectly Classified Instances       412                2.3884 %
Kappa statistic                          0.9741
Mean absolute error                      0.0082
Root mean squared error                  0.0557
Relative absolute error                  6.1928 %
Root relative squared error             21.701  %
Coverage of cases (0.95 level)          99.6464 %
Mean rel. region size (0.95 level)       9.46   %
Total Number of Instances            17250

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                 0.989    0.001    0.988      0.989   0.988      0.987    1         0.998     bic
                 0.991    0.004    0.965      0.991   0.978      0.976    0.999     0.997     compound
                 0.929    0.001    0.967      0.929   0.947      0.946    0.988     0.96      ctcp
                 0.99     0        0.995      0.99    0.993      0.992    0.999     0.998     cubic
                 0.974    0        0.992      0.974   0.983      0.983    0.999     0.997     cubicb
                 0.966    0.003    0.958      0.966   0.962      0.959    0.997     0.988     highspeed
                 0.971    0.003    0.971      0.971   0.971      0.968    0.998     0.991     htcp
                 0.965    0.004    0.961      0.965   0.963      0.96     0.997     0.988     illinois
                 0.953    0.005    0.947      0.953   0.95       0.945    0.996     0.985     reno
                 0.981    0.001    0.99       0.981   0.985      0.984    0.999     0.997     scalable
                 1        0        0.999      1       0.999      0.999    1         1         vegas
                 0.948    0.003    0.965      0.948   0.957      0.954    0.994     0.981     veno
                 1        0        1          1       1          1        1         1         westwood
                 0.977    0.001    0.985      0.977   0.981      0.979    0.999     0.997     yeah
Weighted Avg.    0.976    0.002    0.976      0.976   0.976      0.974    0.998     0.992

=== Confusion Matrix ===

    a    b    c    d    e    f    g    h    i    j    k    l    m    n   <-- classified as
 1302    0    0    1    0    0   13    1    0    0    0    0    0    0 |    a = bic
    0 1580    2    0    2    1    0    0    8    0    1    0    0    0 |    b = compound
    0    4  378    0    0   10    1    2    6    1    0    5    0    0 |    c = ctcp
    0    0    0 1355    0    0   13    0    0    0    0    0    0    0 |    d = cubic
    0    2    0    0  375    0    6    1    0    0    1    0    0    0 |    e = cubicb
    0   21    3    0    0 1221    1    1   14    0    0    3    0    0 |    f = highspeed
   16    2    1    6    1    7 1423    7    1    2    0    0    0    0 |    g = htcp
    0    4    3    0    0    8    3 1492    9    5    0   12    0   10 |    h = illinois
    0   22    3    0    0   19    0    9 1531    0    0   22    0    0 |    i = reno
    0    0    0    0    0    1    4    8    0 1108    0    0    0    9 |    j = scalable
    0    0    0    0    0    0    0    0    0    0 1367    0    0    0 |    k = vegas
    0    2    1    0    0    5    0   10   47    0    0 1197    0    0 |    l = veno
    0    0    0    0    0    0    0    0    0    0    0    0 1271    0 |    m = westwood
    0    0    0    0    0    3    1   21    0    3    0    1    0 1238 |    n = yeah
```

Figure 4.2: Weka output of RF using raw cwnd features

4.4 Results of Machine Learning Models Using the Processed Cwnd Features

Fig. 4.6, Fig. 4.7, Fig. 4.8, Fig. 4.9 and Fig. 4.10 show the screenshots of the results of the Weka DT, RF, ANN, SVM and NB modules using the processed cwnd features, respectively.

Table 4.2 summarizes the five models' prediction accuracy. As we can see from Table 4.2, the RF model still achieves the highest predication accuracy (93.9902%) when using the

```
Time taken to build model: 564.59 seconds

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

Correctly Classified Instances       13040               75.5942 %
Incorrectly Classified Instances      4210               24.4058 %
Kappa statistic                          0.7348
Mean absolute error                      0.0426
Root mean squared error                  0.1552
Relative absolute error                 32.3056 %
Root relative squared error             60.4562 %
Coverage of cases (0.95 level)          92.7652 %
Mean rel. region size (0.95 level)      15.0621 %
Total Number of Instances            17250

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall  F-Measure  MCC    ROC Area  PRC Area  Class
                 0.97     0.004    0.949      0.97    0.959      0.956  0.999     0.988     bic
                 0.556    0.084    0.402      0.556   0.467      0.409  0.901     0.366     compound
                 0.096    0.004    0.39       0.096   0.154      0.184  0.888     0.196     ctcp
                 0.981    0.002    0.981      0.981   0.981      0.979  0.999     0.992     cubic
                 0.906    0        0.983      0.906   0.943      0.943  0.975     0.943     cubicb
                 0.566    0.046    0.495      0.566   0.528      0.489  0.941     0.515     highspeed
                 0.887    0.006    0.937      0.887   0.911      0.903  0.985     0.947     htcp
                 0.843    0.024    0.777      0.843   0.809      0.79   0.966     0.817     illinois
                 0.539    0.053    0.511      0.539   0.524      0.474  0.882     0.508     reno
                 0.752    0.013    0.797      0.752   0.774      0.759  0.988     0.847     scalable
                 1        0        0.996      1       0.998      0.998  1         0.999     vegas
                 0.398    0.006    0.844      0.398   0.541      0.559  0.811     0.552     veno
                 0.998    0.001    0.987      0.998   0.992      0.992  1         0.997     westwood
                 0.77     0.024    0.722      0.77    0.745      0.724  0.977     0.819     yeah
Weighted Avg.    0.756    0.022    0.77       0.756   0.753      0.737  0.952     0.761

=== Confusion Matrix ===

    a    b    c    d    e    f    g    h    i    j    k    l    m    n   <-- classified as
 1277    0    0    0    0    0   39    0    0    0    0    0    0    1 |    a = bic
    0  887    4    0    0  379    1    3  305    0    0    4    1   10 |    b = compound
    0  142   39    0    0   60    0    0  157    2    0    4    0    3 |    c = ctcp
    2    0    0 1342    1    0   23    0    0    0    0    0    0    0 |    d = cubic
    2    1    1    0  349    0   10    2    1    3    1    0    8    7 |    e = cubicb
    0  378    4    0    0  715    0    5  150    0    0    3    0    9 |    f = highspeed
   64    7    0   25    1   18 1300   20    2   23    0    0    0    6 |    g = htcp
    0   22    4    0    4   41    0 1304    9    8    1   62    2   89 |    h = illinois
    0  507   13    0    0  192    0    1  865    0    1   17    3    7 |    i = reno
    0    0    1    1    0    0   11   29    0  850    0    0    0  238 |    j = scalable
    0    0    0    0    0    0    0    0    0    0 1367    0    0    0 |    k = vegas
    0  262   32    0    0   39    0  219  199    1    1  502    1    6 |    l = veno
    0    0    0    0    0    0    0    0    3    0    0    0 1268    0 |    m = westwood
    0    2    2    0    0    1    4   96    2  179    1    3    2  975 |    n = yeah
```

Figure 4.3: Weka output of ANN using raw cwnd features

Table 4.2: Five models' prediction accuracy when using processed cwnd features

	DT	RF	ANN	SVM	NB
Prediction Accuracy (%)	92.2715	93.9902	48.4372	65.6425	42.5746
Build-Model-Time (sec)	2.8	3.9	275.32	109.64	0.28

processed cwnd features and it builds the model in a pretty short time (3.9 seconds). Thus, the RF model is viewed as the best model when using the processed cwnd features.

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

Correctly Classified Instances        9161               53.1072 %
Incorrectly Classified Instances      8089               46.8928 %
Kappa statistic                          0.4895
Mean absolute error                      0.067
Root mean squared error                  0.2588
Relative absolute error                 50.8458 %
Root relative squared error            100.8425 %
Coverage of cases (0.95 level)          53.1072 %
Mean rel. region size (0.95 level)       7.1429 %
Total Number of Instances            17250

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                 0.061    0        1          0.061   0.115      0.237    0.53      0.132     bic
                 0.616    0.001    0.991      0.616   0.76       0.766    0.808     0.646     compound
                 0.619    0        1          0.619   0.765      0.783    0.81      0.628     ctcp
                 0.01     0        1          0.01    0.02       0.097    0.505     0.089     cubic
                 0.397    0        1          0.397   0.569      0.626    0.699     0.411     cubicb
                 0.524    0        1          0.524   0.687      0.71     0.762     0.559     highspeed
                 1        0.511    0.154      1       0.266      0.274    0.744     0.154     htcp
                 0.307    0        1          0.307   0.469      0.536    0.653     0.369     illinois
                 0.671    0        0.994      0.671   0.801      0.803    0.835     0.698     reno
                 0.582    0        1          0.582   0.736      0.752    0.791     0.61      scalable
                 0.649    0        1          0.649   0.787      0.794    0.824     0.677     vegas
                 0.687    0        1          0.687   0.814      0.819    0.844     0.71      veno
                 0.786    0        1          0.786   0.88       0.879    0.893     0.802     westwood
                 0.465    0        1          0.465   0.635      0.668    0.732     0.504     yeah
Weighted Avg.    0.531    0.044    0.927      0.531   0.583      0.613    0.744     0.494

=== Confusion Matrix ===

    a    b    c    d    e    f    g    h    i    j    k    l    m    n   <-- classified as
   80    0    0    0    0    0 1237    0    0    0    0    0    0    0 |    a = bic
    0  982    0    0    0    0  606    0    6    0    0    0    0    0 |    b = compound
    0    0  252    0    0    0  154    0    1    0    0    0    0    0 |    c = ctcp
    0    0    0   14    0    0 1354    0    0    0    0    0    0    0 |    d = cubic
    0    0    0    0  153    0  232    0    0    0    0    0    0    0 |    e = cubicb
    0    0    0    0    0  662  602    0    0    0    0    0    0    0 |    f = highspeed
    0    0    0    0    0    0 1466    0    0    0    0    0    0    0 |    g = htcp
    0    0    0    0    0    0 1072  474    0    0    0    0    0    0 |    h = illinois
    0    9    0    0    0    0  519    0 1078    0    0    0    0    0 |    i = reno
    0    0    0    0    0    0  472    0    0  658    0    0    0    0 |    j = scalable
    0    0    0    0    0    0  480    0    0    0  887    0    0    0 |    k = vegas
    0    0    0    0    0    0  395    0    0    0    0  867    0    0 |    l = veno
    0    0    0    0    0    0  272    0    0    0    0    0  999    0 |    m = westwood
    0    0    0    0    0    0  678    0    0    0    0    0    0  589 |    n = yeah
```

Figure 4.4: Weka output of SVM using raw cwnd features

4.5 Discussion of the Results

In the following, we analyze the performance of the five machine learning models.

In the experiments, the NB model has the lowest prediction accuracy. This is because the NB model assumes all the features are independent identically distributed. While in our case, the cwnd size traces are collected in the continuous time slots and actually they are dependent. Also the cwnd sizes are not identically distributed. That is why the NB model performs poor in the experiments.

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

Correctly Classified Instances        7590               44      %
Incorrectly Classified Instances      9660               56      %
Kappa statistic                          0.3948
Mean absolute error                      0.0794
Root mean squared error                  0.2763
Relative absolute error                 60.288  %
Root relative squared error            107.665  %
Coverage of cases (0.95 level)          47.8841 %
Mean rel. region size (0.95 level)       8.2025 %
Total Number of Instances            17250

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                 0.909    0.029    0.723      0.909   0.806      0.794    0.973     0.878     bic
                 0.009    0.004    0.2        0.009   0.018      0.025    0.811     0.207     compound
                 0.042    0.015    0.064      0.042   0.051      0.033    0.796     0.057     ctcp
                 0.661    0.001    0.976      0.661   0.788      0.791    0.995     0.962     cubic
                 0.2      0.001    0.755      0.2     0.316      0.382    0.748     0.203     cubicb
                 0.09     0.064    0.1        0.09    0.095      0.027    0.775     0.147     highspeed
                 0.662    0.048    0.561      0.662   0.607      0.57     0.931     0.569     htcp
                 0.047    0.002    0.702      0.047   0.088      0.167    0.786     0.264     illinois
                 0.006    0.003    0.161      0.006   0.012      0.014    0.742     0.162     reno
                 0.012    0.007    0.115      0.012   0.022      0.017    0.729     0.111     scalable
                 0.991    0        1          0.991   0.996      0.995    0.997     0.995     vegas
                 0.888    0.347    0.168      0.888   0.283      0.29     0.82      0.209     veno
                 0.987    0        1          0.987   0.993      0.993    1         0.999     westwood
                 0.37     0.083    0.262      0.37    0.307      0.246    0.808     0.263     yeah
Weighted Avg.    0.44     0.044    0.494      0.44    0.398      0.394    0.859     0.461

=== Confusion Matrix ===

    a    b    c    d    e    f    g    h    i    j    k    l    m    n   <-- classified as
 1197    0    0    0    1    0  116    0    0    1    0    0    0    2 |    a = bic
    0   15   43    0    2  129    0    0    0    0    0 1105    0  300 |    b = compound
    0    4   17    0    0   53    0    0    0    1    0  287    0   45 |    c = ctcp
    5    0    0  904    2    0  457    0    0    0    0    0    0    0 |    d = cubic
    2    0   11    4   77   61    0    0    8   10    0  184    0   28 |    e = cubicb
    0    6   54    0    3  114    0    7    6   20    0  766    0  288 |    f = highspeed
  318    2   19   18    5    9  970    7    2   59    0    9    0   48 |    g = htcp
    0    9   17    0    3  218    4   73    2   12    0  832    0  376 |    h = illinois
    0    6   51    0    4  245    0    2   10    1    0 1127    0  160 |    i = reno
  133    1    3    0    0   23  149    1    3   14    0  777    0   26 |    j = scalable
    0    0    0    0    0    0    0    0   12    0 1355    0    0    0 |    k = vegas
    0   11   22    0    0   51    0    2    6    0    0 1121    0   49 |    l = veno
    0    4    7    0    0    0    0    0    6    0    0    0 1254    0 |    m = westwood
    0   17   21    0    5  240   33   12    7    4    0  459    0  469 |    n = yeah
```

Figure 4.5: Weka output of NB using raw cwnd features

For the DT model, whether its features are dependent or independent does not influence its performance. That is why the DT model achieves much better prediction accuracy than the NB model. The DT model makes the predictions relying on one decision tree.

The RF model further improves the prediction accuracy by making the final classification decision based on the mode voting of all its sub decision trees instead of one decision tree. The RF model normally is composed of many sub decision trees and each sub decision tree is established by partial features of a portion of the whole feature vectors. As a result, some sub decision trees might be good at differentiating some specific TCP protocols and their

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

Correctly Classified Instances       15676               92.2715 %
Incorrectly Classified Instances      1313                7.7285 %
Kappa statistic                          0.9162
Mean absolute error                      0.0137
Root mean squared error                  0.0961
Relative absolute error                 10.372  %
Root relative squared error             37.4311 %
Coverage of cases (0.95 level)          95.8561 %
Mean rel. region size (0.95 level)       8.5635 %
Total Number of Instances            16989

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall  F-Measure  MCC    ROC Area  PRC Area  Class
                 0.964    0.003    0.959      0.964   0.961      0.958  0.989     0.928     bic
                 0.761    0.012    0.868      0.761   0.811      0.795  0.965     0.83      compound
                 0.985    0.002    0.979      0.985   0.982      0.98   0.996     0.978     cubic
                 0.884    0.002    0.912      0.884   0.898      0.896  0.954     0.841     cubicb
                 0.855    0.003    0.877      0.855   0.866      0.862  0.945     0.797     ctcp
                 0.969    0.004    0.952      0.969   0.961      0.958  0.988     0.935     highspeed
                 0.919    0.007    0.927      0.919   0.923      0.916  0.97      0.894     htcp
                 0.927    0.005    0.946      0.927   0.937      0.931  0.974     0.906     illinois
                 0.865    0.026    0.776      0.865   0.818      0.799  0.975     0.846     reno
                 0.946    0.004    0.945      0.946   0.945      0.941  0.98      0.913     scalable
                 1        0        1          1       1          1      1         1         vegas
                 0.975    0.004    0.951      0.975   0.962      0.96   0.99      0.928     veno
                 0.947    0.004    0.95       0.947   0.948      0.944  0.99      0.938     westwood
                 0.908    0.008    0.9        0.908   0.904      0.896  0.964     0.852     yeah
Weighted Avg.    0.923    0.007    0.924      0.923   0.923      0.916  0.979     0.905

=== Confusion Matrix ===

    a    b    c    d    e    f    g    h    i    j    k    l    m    n   <-- classified as
 1270    0    1    2    0    0   44    0    0    0    0    0    0    1 |    a = bic
    0 1212    0    1   12    7    0   20  289    3    0    4    5   39 |    b = compound
    2    0 1345    0    0    0   18    0    0    0    0    0    0    1 |    c = cubic
    8    0    2  320    0    2   18    3    0    6    0    0    0    3 |    d = cubicb
    0   10    0    0  348   12    5    5   11    6    0    2    0    8 |    e = ctcp
    0    3    0    0   12 1207    1    1    0   14    0    0    0    7 |    f = highspeed
   44    1   26   16    5    4 1353    1    0   20    0    0    0    3 |    g = htcp
    0   24    0    3    3    5    1 1420    4    9    0   36    1   25 |    h = illinois
    0  107    0    0    4    0    0    2 1386    0    0   15   55   34 |    i = reno
    0    5    0    7    2   21   15    5    3 1059    0    0    0    3 |    j = scalable
    0    0    0    0    0    0    0    0    0    0 1170    0    0    0 |    k = vegas
    0    1    0    0    1    0    0   21    6    0    0 1231    2    1 |    l = veno
    0    3    0    0    2    0    0    0   52    0    0    7 1205    3 |    m = westwood
    0   30    0    2    8   10    4   23   35    4    0    0    1 1150 |    n = yeah
```

Figure 4.6: Weka output of DT using processed cwnd features

voting could strengthen the influence of those specific features on the final classification decision. By the observations in [2], some TCP protocols might generate similar cwnd size traces which only differ at a few cwnd sizes. Those specific cwnd sizes are critical for successfully classifying those TCP protocols. The sub decision trees focusing on those specific cwnd sizes have bigger chance to successfully identify those similar TCP protocols and their voting will influence the final classification decision and increase the prediction accuracy. Otherwise, using only one decision tree to identify so many TCP protocols, the small parts of critical features will be submerged by other prominent features. That is the

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

Correctly Classified Instances       15968               93.9902 %
Incorrectly Classified Instances      1021                6.0098 %
Kappa statistic                          0.9348
Mean absolute error                      0.0141
Root mean squared error                  0.0811
Relative absolute error                 10.671  %
Root relative squared error             31.5827 %
Coverage of cases (0.95 level)          99.0347 %
Mean rel. region size (0.95 level)      10.0018 %
Total Number of Instances            16989

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall  F-Measure  MCC    ROC Area  PRC Area  Class
                 0.98     0.002    0.982      0.98    0.981      0.979  0.998     0.995     bic
                 0.779    0.012    0.874      0.779   0.824      0.808  0.984     0.908     compound
                 0.991    0.001    0.99       0.991   0.991      0.99   1         0.999     cubic
                 0.956    0.001    0.975      0.956   0.965      0.964  0.993     0.979     cubicb
                 0.907    0.002    0.937      0.907   0.921      0.92   0.988     0.947     ctcp
                 0.982    0.002    0.972      0.982   0.977      0.975  0.997     0.992     highspeed
                 0.954    0.004    0.953      0.954   0.954      0.949  0.997     0.981     htcp
                 0.969    0.005    0.954      0.969   0.961      0.958  0.997     0.987     illinois
                 0.872    0.024    0.791      0.872   0.83       0.812  0.984     0.899     reno
                 0.954    0.002    0.971      0.954   0.962      0.96   0.997     0.988     scalable
                 1        0        1          1       1          1      1         1         vegas
                 0.975    0.003    0.968      0.975   0.971      0.969  0.998     0.993     veno
                 0.962    0.004    0.955      0.962   0.958      0.955  0.993     0.981     westwood
                 0.919    0.006    0.925      0.919   0.922      0.916  0.987     0.954     yeah
Weighted Avg.    0.94     0.006    0.941      0.94    0.94       0.935  0.994     0.97

=== Confusion Matrix ===

    a    b    c    d    e    f    g    h    i    j    k    l    m    n   <-- classified as
 1292    0    0    0    0    0   25    0    0    0    0    0    0    1 |    a = bic
    0 1240    0    0    4    1    0    9  291    2    0    2    7   36 |    b = compound
    0    0 1354    0    0    0   12    0    0    0    0    0    0    0 |    c = cubic
    0    0    0  346    0    1   10    2    0    2    0    0    0    1 |    d = cubicb
    0   11    0    0  369    7    3    3    4    1    0    3    0    6 |    e = ctcp
    0    2    0    0    8 1222    1    1    0    5    0    0    0    6 |    f = highspeed
   24    0   13    5    2    3 1405    3    0   18    0    0    0    0 |    g = htcp
    0    7    0    0    0    2    1 1484    3    2    0   21    0   11 |    h = illinois
    0  117    0    0    2    0    0    1 1398    0    0   10   47   28 |    i = reno
    0    2    0    4    0   20   13    8    1 1068    0    0    0    4 |    j = scalable
    0    0    0    0    0    0    0    0    0    0 1170    0    0    0 |    k = vegas
    0    0    0    0    2    0    0   22    5    0    0 1231    3    0 |    l = veno
    0    2    0    0    0    0    0    0   39    0    0    5 1224    2 |    m = westwood
    0   38    0    0    7    1    4   23   26    2    0    0    1 1165 |    n = yeah
```

Figure 4.7: Weka output of RF using processed cwnd features

major reason that the RF model provides higher prediction accuracy than the DT model and in fact RF model has the highest prediction accuracy among the five machine learning models we chose.

Applying the ANN model is not quite autonomous. Several parameters need to be repeatedly adjusted in the experiments to achieve the optimal performance, such as the layers' structure, the kernel algorithm and the learning rate. Adjusting these parameters needs experience and takes time. In our experiments, we simply use the default setting of Weka ANN module and by changing the parameter values to improve Weka ANN module's

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

Correctly Classified Instances        8229               48.4372 %
Incorrectly Classified Instances      8760               51.5628 %
Kappa statistic                          0.4386
Mean absolute error                      0.0832
Root mean squared error                  0.206
Relative absolute error                 63.1408 %
Root relative squared error             80.2866 %
Coverage of cases (0.95 level)          93.4016 %
Mean rel. region size (0.95 level)      36.0464 %
Total Number of Instances           16989

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                 0.901    0.012    0.862      0.901   0.881      0.871    0.973     0.879     bic
                 0.393    0.105    0.279      0.393   0.326      0.248    0.806     0.285     compound
                 0.987    0.003    0.961      0.987   0.974      0.972    0.998     0.986     cubic
                 0.18     0.004    0.478      0.18    0.261      0.284    0.669     0.215     cubicb
                 0        0        0          0       0         -0.003    0.761     0.06      ctcp
                 0.288    0.028    0.451      0.288   0.351      0.321    0.878     0.354     highspeed
                 0.758    0.019    0.792      0.758   0.775      0.754    0.967     0.836     htcp
                 0.312    0.023    0.572      0.312   0.404      0.383    0.773     0.41      illinois
                 0.464    0.14     0.256      0.464   0.33       0.251    0.841     0.332     reno
                 0.352    0.046    0.351      0.352   0.351      0.305    0.895     0.365     scalable
                 1        0        0.996      1       0.998      0.998    1         0.999     vegas
                 0.272    0.074    0.227      0.272   0.248      0.182    0.851     0.276     veno
                 0.212    0.072    0.192      0.212   0.202      0.134    0.852     0.243     westwood
                 0.1      0.034    0.191      0.1     0.131      0.089    0.809     0.216     yeah
Weighted Avg.    0.484    0.047    0.495      0.484   0.479      0.441    0.876     0.495

=== Confusion Matrix ===

    a    b    c    d    e    f    g    h    i    j    k    l    m    n   <-- classified as
 1188    0    3    0    0    0  127    0    0    0    0    0    0    0 |    a = bic
    0  626    0    1    0   76    0   14  301  136    0  164  212   62 |    b = compound
    7    0 1348    0    0    0   11    0    0    0    0    0    0    0 |    c = cubic
    1    2    0   65    4    0   61  152   15   15    2    0    0   45 |    d = cubicb
    0   44    0    0    0   21    2    0  133   25    0   73   89   20 |    e = ctcp
    0  429    0    0    0  358    1   29   98  168    0   35    9  118 |    f = highspeed
  181   15   51   25    0    2 1117   36    6   38    0    0    1    1 |    g = htcp
    1  228    0   31    1   68   20  478  333  125    0   85   60  101 |    h = illinois
    0  124    0    0    0    0    0   49  744   69    0  314  265   38 |    i = reno
    0  142    0    0    0  139   44   45  154  394    0   42   71   89 |    j = scalable
    0    0    0    0    0    0    0    0    0    0 1170    0    0    0 |    k = vegas
    0  105    0    1    0    1    0    8  442   39    0  344  289   34 |    l = veno
    0  120    0    0    0    0    0    9  473   36    0  333  270   31 |    m = westwood
    0  409    0   13    0  128   27   15  204   78    3  123  140  127 |    n = yeah
```

Figure 4.8: Weka output of ANN using processed cwnd features

performance is possible. Another disadvantage of ANN model is its slowness. From Table. 4.1 and Table. 4.2, we can see that in our experiments, the ANN model takes much longer model-build-time than the other models.

Normally the SVM model is good at the classification of two classes, and its prediction performance is closely related to the balance of the data set (the number of negative examples is close to the number of positive examples). In our case, there are 14 classes (TCP protocols) and for each class, the data set may not be balanced. These factors lead to the poor performance of the SVM model in our experiments.

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

Correctly Classified Instances       11152               65.6425 %
Incorrectly Classified Instances      5837               34.3575 %
Kappa statistic                          0.6262
Mean absolute error                      0.0491
Root mean squared error                  0.2215
Relative absolute error                 37.2634 %
Root relative squared error             86.3292 %
Coverage of cases (0.95 level)          65.6425 %
Mean rel. region size (0.95 level)       7.1429 %
Total Number of Instances            16989

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall  F-Measure  MCC    ROC Area  PRC Area  Class
                 0.127    0        1          0.127   0.225      0.344  0.563     0.194     bic
                 0.561    0.014    0.8        0.561   0.66       0.643  0.773     0.49      compound
                 0.015    0        1          0.015   0.029      0.116  0.507     0.094     cubic
                 0.481    0        0.994      0.481   0.648      0.687  0.74      0.489     cubicb
                 0.71     0.001    0.944      0.71    0.811      0.815  0.855     0.678     ctcp
                 0.862    0.002    0.969      0.862   0.912      0.908  0.93      0.846     highspeed
                 0.999    0.322    0.227      0.999   0.371      0.392  0.839     0.227     htcp
                 0.572    0.001    0.974      0.572   0.721      0.73   0.785     0.596     illinois
                 0.831    0.021    0.803      0.831   0.817      0.798  0.905     0.684     reno
                 0.668    0.004    0.918      0.668   0.773      0.771  0.832     0.635     scalable
                 0.842    0        1          0.842   0.914      0.912  0.921     0.853     vegas
                 0.955    0.002    0.971      0.955   0.963      0.96   0.976     0.931     veno
                 0.946    0.003    0.962      0.946   0.954      0.95   0.971     0.914     westwood
                 0.564    0.004    0.919      0.564   0.699      0.704  0.78      0.55      yeah
Weighted Avg.    0.656    0.033    0.873      0.656   0.665      0.681  0.812     0.574

=== Confusion Matrix ===

    a    b    c    d    e    f    g    h    i    j    k    l    m    n   <-- classified as
  167    0    0    0    0    0 1151    0    0    0    0    0    0    0 |    a = bic
    0  893    0    0    0    0  398    9  249    0    0    2   10   31 |    b = compound
    0    0   20    0    0    0 1346    0    0    0    0    0    0    0 |    c = cubic
    0    0    0  174    0    0  174    0    0   14    0    0    0    0 |    d = cubicb
    0    1    0    0  289   17   60    0    7   21    0    5    0    7 |    e = ctcp
    0    0    0    0    8 1073  138    0    0   26    0    0    0    0 |    f = highspeed
    0    0    0    0    0    0 1472    0    0    1    0    0    0    0 |    g = htcp
    0    1    0    0    0    0  629  876    0    1    0   18    0    6 |    h = illinois
    0  177    0    0    3    0   37    0 1332    0    0    5   35   14 |    i = reno
    0    1    0    1    2   17  348    0    0  748    0    0    0    3 |    j = scalable
    0    0    0    0    0    0  185    0    0    0  985    0    0    0 |    k = vegas
    0    0    0    0    2    0   47    7    0    0    0 1206    1    0 |    l = veno
    0    3    0    0    1    0    4    5   48    0    0    6 1203    2 |    m = westwood
    0   40    0    0    1    0  483    2   22    4    0    0    1  714 |    n = yeah
```

Figure 4.9: Weka output of SVM using processed cwnd features

In conclusion, the RF model is the best model for classifying TCP algorithms.

4.6 Experiments Applying the k-NN Algorithm

To compare the prediction performance between our RF model based approaches and that of the previous k-NN algorithm based CAAI approach, we conduct an experiment which follows the CAAI approach and uses the k-NN algorithm.

CAAI does not use 10-fold cross validation. Instead, it uses only the processed cwnd

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

Correctly Classified Instances        7233               42.5746 %
Incorrectly Classified Instances      9756               57.4254 %
Kappa statistic                          0.3826
Mean absolute error                      0.0828
Root mean squared error                  0.271
Relative absolute error                 62.8356 %
Root relative squared error            105.61   %
Coverage of cases (0.95 level)          48.9611 %
Mean rel. region size (0.95 level)      10.0964 %
Total Number of Instances           16989

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall  F-Measure  MCC    ROC Area  PRC Area  Class
                 0.926    0.018    0.811      0.926   0.865      0.855  0.984     0.957     bic
                 0.041    0.024    0.15       0.041   0.064      0.031  0.709     0.172     compound
                 0.976    0.003    0.966      0.976   0.971      0.968  0.997     0.992     cubic
                 0.149    0.001    0.831      0.149   0.253      0.347  0.727     0.156     cubicb
                 0.128    0.133    0.023      0.128   0.039     -0.003  0.643     0.034     ctcp
                 0.072    0.016    0.259      0.072   0.113      0.103  0.894     0.291     highspeed
                 0.726    0.015    0.819      0.726   0.77       0.751  0.98      0.831     htcp
                 0.131    0.016    0.456      0.131   0.204      0.208  0.867     0.348     illinois
                 0.051    0.022    0.193      0.051   0.081      0.054  0.775     0.178     reno
                 0.19     0.04     0.252      0.19    0.217      0.172  0.792     0.204     scalable
                 0.999    0        1          0.999   1          1      0.999     0.999     vegas
                 0.268    0.078    0.215      0.268   0.239      0.171  0.805     0.179     veno
                 0.985    0.244    0.246      0.985   0.394      0.426  0.897     0.323     westwood
                 0.074    0.004    0.614      0.074   0.132      0.196  0.739     0.233     yeah
Weighted Avg.    0.426    0.04     0.488      0.426   0.399      0.39   0.859     0.453

=== Confusion Matrix ===

    a    b    c    d    e    f    g    h    i    j    k    l    m    n   <-- classified as
 1220    0    0    0    0    0   97    0    0    1    0    0    0    0 |    a = bic
    0   65    0    1  405   81    0   55   15  164    0   34  760   12 |    b = compound
    8    0 1333    0    0    0   25    0    0    0    0    0    0    0 |    c = cubic
   19   19    0   54  109   12    1   58    0   86    0    4    0    0 |    d = cubicb
    0   12    0    0   52   13    5   13   42    0    0  104  166    0 |    e = ctcp
    0  105    0    3  547   90    0    2   56    6    0  434    1    1 |    f = highspeed
  249    4   47    1    8    2 1069   26    1   54    0    0    1   11 |    g = htcp
    0   34    0    4  536   56   16  201   84  258    0  108  204   30 |    h = illinois
    0    1    0    1   14    6    0    1   82    0    0   13 1485    0 |    i = reno
    7   56    0    0  272   36   48   44   42  213    0  394    3    5 |    j = scalable
    0    0    0    1    0    0    0    0    0    0 1169    0    0    0 |    k = vegas
    0    1    0    0    9   12    0    5   80    0    0  338  818    0 |    l = veno
    0    0    0    0    1    0    0    0   13    0    0    5 1253    0 |    m = westwood
    1  136    0    0  310   40   44   36   10   63    0  136  397   94 |    n = yeah
```

Figure 4.10: Weka output of NB using processed cwnd features

features obtained under the perfect network condition. We wrote a Matlab program to extract such processed cwnd features from our raw cwnd data, and then used the same k-NN algorithm as CAAI to classify TCP algorithms. Finally, our experiment shows that the prediction accuracy of k-NN is 71%.

The experimental result proves that comparing to the previous k-NN algorithm based CAAI method, our RF model based methods greatly improve the prediction accuracy of TCP identification.

4.7 Summary

In this chapter, we described the details of our experiments. We conducted experiments using two types of features and using five popular machine learning models. Our results show that the RF model with the raw cwnd features achieve the highest prediction accuracy (97.6%) which is much higher than that of CAAI (about 71%).

Chapter 5

Future Work

There are several potential improvements for this research as the future work.

First, our experiments only go through five machine learning models. There could be some other machine learning models which outperform the RF model. Also, in the experiments, we utilize the default Weka settings for each of the five Weka modules. Adjusting the parameters of these models might further improve their prediction performance.

Second, we can extend the RF model to recognize the "unusual" TCP protocols.

Third, all the experiments are conducted in the lab test bed in which we emulate various network conditions. However, it is not guaranteed that these emulated network conditions can well represent the various network environments in the real Internet. There might be some unnoticed or unknown network configurations in the Internet which influence the behavior of remote web servers and impair the performance of our methods. It is meaningful to validate our methods' prediction performance in the real Internet. The basic idea is to set up some specific TCP version configured web servers at the different locations of the Internet, and applying our methods to validate their performance. To reduce the cost of setting up remote web servers, we could rent some geographically distributed web servers of some cloud systems such as the Amazon EC2 platform.

Bibliography

[1] Wikipedia, "Transmission control protocol," http://en.wikipedia.org/wiki/Transmission_Control_Protocol. 1.1

[2] P. Yang, W. Luo, L. Xu, J. Deogun, and Y. Lu, "TCP congestion avoidance algorithm identification," in *Proceedings of IEEE ICDCS*, 2011, pp. 310 –321. 1.2, 2.1, 2.2, 4.5

[3] Wikipedia, "Artifical neural network," http://en.wikipedia.org/wiki/Artificial_neural_network. 1.3

[4] ——, "Naive bayes," http://en.wikipedia.org/wiki/Naive_Bayes_classifier. 1.3, 2.3

[5] T. Henderson, E. Sahouria, S. McCanne, and R. Katz, "On improving the fairness of TCP congestion avoidance," in *Proceedings of IEEE GLOBECOM*, 1998, pp. 539 –544. 2.1

[6] V. Jacobson, "Congestion avoidance and control," in *Proceedings of ACM SIGCOMM*, Aug. 1988. 2.1

[7] D. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN systems*, vol. 17, no. 1, pp. 1–14, 1989. 2.1

[8] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control for fast long-distance networks," in *Proceedings of IEEE INFOCOM*, Mar. 2004. 2.1

[9] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 64–74, Jul. 2008. 2.1

[10] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound TCP approach for high-speed and long distance networks," in *Proceedings of IEEE INFOCOM*, Apr. 2006. 2.1

[11] L. Brakmo, S. O'Malley, and L. Peterson, "TCP vegas: New techniques for congestion detection and avoidance," in *Proceedings of ACM SIGCOMM*, Aug. 1994, pp. 24 –35. 2.1

[12] J. Oshio, S. Ata, and I. Oka, "Identification of different TCP versions based on cluster analysis," in *Proceedings of IEEE ICCCN*, Aug. 2009. 2.2

[13] J. Padhye and S. Floyd, "On inferring TCP behavior," in *Proceedings of ACM SIGCOMM*, Aug. 2001. 2.2

[14] S. Feyzabadi and J. Schonwalder, "Identifying TCP congestion control algorithms using active probing," in *Proceedings of Passive and Active Measurement Conference (PAM), Poster*, Apr. 2010. 2.2

[15] Wikipedia, "Support vector machine," http://en.wikipedia.org/wiki/Support_vector_machine. 2.3

[16] D. Arndt and A. Zincir-Heywood, "A comparison of three machine learning techniques for encrypted network traffic analysis," in *Proceedings of IEEE CISDA*, Apr. 2011, pp. 107 –114. 2.3

[17] J. Zheng and Y. Xu, "Identification of network traffic based on support vector machine," in *Proceedings of IEEE ICACTE*, vol. 3, Aug. 2010, pp. 286 –290. 2.3

[18] I. Firdausi, C. Lim, A. Erwin, and A. Nugroho, "Analysis of machine learning techniques used in behavior-based malware detection," in *Proceedings of IEEE ACT*, Dec. 2010, pp. 201 –203. 2.3

[19] D. Renuka, T. Hamsapriya, M. Chakkaravarthi, and P. Surya, "Spam classification based on supervised learning using machine learning techniques," in *Proceedings of IEEE PACC*, Jul. 2011, pp. 1 –7. 2.3

[20] Y. Meng, "The practice on using machine learning for network anomaly intrusion detection," in *Proceedings of IEEE ICMLC*, vol. 2, Jul. 2011, pp. 576 –581. 2.3

[21] T. Subbulakshmi, S. Shalinie, V. GanapathiSubramanian, K. BalaKrishnan, D. Anand-Kumar, and K. Kannathal, "Detection of DDoS attacks using enhanced support vector machines with real time generated dataset," in *Proceedings of IEEE ICoAC*, Dec. 2011, pp. 17 –22. 2.3

[22] S. Seufert and D. O'Brien, "Machine learning for automatic defence against distributed denial of service attacks," in *Proceedings of IEEE ICC*, 2007, pp. 1217–1222. 2.3

[23] Wikipedia, "Weka," http://www.cs.waikato.ac.nz/ml/weka/. 4.3.1

[24] Matlab, "Matlab," http://www.mathworks.com/. 4.3.1

[25] R. Language, "R," http://www.r-project.org/. 4.3.1