3-2004

Machine Learning Techniques for Characterizing IEEE 802.11b Encrypted Data Streams

Michael J. Henson

MACHINE LEARNING TECHNIQUES FOR CHARACTERIZING IEEE 802.11b ENCRYPTED DATA STREAMS

THESIS

Michael J. Henson, 2d Lt, USAF

AFIT/GCS/ENG/04-08

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

MACHINE LEARNING TECHNIQUES FOR CHARACTERIZING IEEE 802.11b ENCRYPTED DATA STREAMS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science

Michael J. Henson, BS

2d Lt, USAF

March 2004

MACHINE LEARNING TECHNIQUES FOR

CHARACTERIZING IEEE 802.11b ENCRYPTED

DATA STREAMS

Michael J. Henson, BS
2d Lt, USAF

Approved:

/SIGNED/

Major Rusty O. Baldwin, PhD (Chairman)	date
/SIGNED/	
Dr. Gilbert L. Peterson (Member)	date
/SIGNED/	
Dr. Richard A. Raines (Member)	date

Acknowledgments

First and foremost, I must thank my Heavenly Father without whom I could achieve nothing. Next, I would like to thank my wife for her support throughout this trying time. Without that support, I certainly would have failed at this endeavor. I would also like to express my sincere gratitude to my advisor, Major Rusty Baldwin, for his guidance throughout the course of this thesis effort. Another debt of gratitude is owed to Dr. Gilbert Peterson for giving me the necessary foundation of knowledge in Artificial Intelligence and Machine Learning techniques, a subject of which I knew nothing before coming to AFIT. I would also like to thank Captain John Wagnon for discussing his work and corresponding with me as to the setup of my experimental wireless network.

Michael J. Henson

Table of Contents

Acknowledgments v
List of Figures ix
List of Tables x
Abstract xi

I. Introduction 1

1.1 Motivation 1

1.2 Overview 2

1.3 Thesis Organization 4

II. Literature Review 5

2.1 Background 5

2.2 Current Related Research 6

2.3 Data Analysis Techniques 11

2.3.1 Statistical Pattern Recognition 11
2.3.2 Artificial Neural Networks 15
2.3.3 Decision Trees 18
2.3.4 Discovery of Frequent Episodes in Event Sequences 21
2.3.5 Hidden Markov Models 26

2.4 Summary 29

III. Methodology 30

3.1 Background 30

3.2 Problem Definition 30

3.2.1 Goals and Hypothesis 30
3.2.2 Approach 31

3.3 System Boundaries 31

3.4 System Services 33

3.5 Performance Metrics 33

3.6 Parameters 34

3.6.1 System 34
3.6.2 Workload 35

3.7 Factors 35

3.8 Evaluation Technique 36

3.9 Workload 36

3.10 Experimental Design 37

3.11 Analyze and Interpret Results 39

3.12 Summary 39

IV. Analysis and Findings 41

4.1 802.11b Ad-Hoc Network Topology 41

4.2 Collected Data 42

4.3 Data Preparation 45

4.4 Neural Network Configuration 47

4.5 Neural Network Results 49

4.6 Decision Tree Configuration 55

4.6 Decision Tree Results 59

4.7 Comparative Analysis 61

4.8 Analysis of Variance 62

4.9 Summary 63

V. Conclusions and Recommendations 64

5.1 Problem Summary 64

5.2 Findings 64

5.3 Limitations 65

5.4 Recommendations for Future Research 66

Appendix A 68
Appendix B 76
References 77

List of Figures

Figure Page

1-1. Wireless Ad-Hoc Network 2

2-1. Format of the IP Header 7

2-2. Histogram of HTML page sizes served by U.C. Berkeley Web Site 8

2-3. "Signature" of SSH Password Prompt Entry....9

2-4. Model for Statistical Pattern Recognition 12

2-5. Typical Neural Network Architecture 16

2-6. Outcome Variable for Outlook Example....20

2-7. Example Sequence of Events 22

2-8. Example Sequence with Two Windows of Width 5....23

2-9. Episodes α, β, and γ 23

2-10. Frequent Episodes as a Function of Window Width 25

2-11. One Coin Hidden Markov Model Example....27

3-1. Application Determination System 32

4-1. Neural Network with Desired Output for Email Sample 46

4-2. Sample ARFF File....47

4-3. Portion of J48 Induced Decision Tree 59

List of Tables

Table Page

2-1. Error Estimation Methods 14

2-2. Weather Data 19

2-3. Experiments with Various Data Sets 25

3-1. System Parameters for ADS 34

3-2. Factors Varied....36

4-1. Combinations of Data Collected....42

4-2. Unique Sizes for Unencrypted vs. Encrypted E-Mail 44

4-3. Sample Confusion Matrix....50

4-4. Node Output for Email-Http Sample....50

4-5. Neural Network Performance for Window = 11 51

4-6. Averages for All Replications of Neural Network Performance....52

4-7. Averages for Single Application Neural Network Performance 53

4-8. Packet Sizes by Application 54

4-9. Outcome Variable for Outlook 56

4-10. Classification Percentages for All Decision Tree Experiments....60

Abstract

As wireless networks become an increasingly common part of the infrastructure in industrialized nations, the vulnerabilities of this technology need to be evaluated. Even though there have been major advancements in encryption technology, security protocols and packet header obfuscation techniques, other distinguishing characteristics do exist in wireless network traffic. These characteristics include packet size, signal strength, channel utilization and others. Using these characteristics, windows of size 11, 31, and 51 packets are collected and machine learning (ML) techniques are trained to classify applications accessing the 802.11b wireless channel. The four applications used for this study included E-Mail, FTP, HTTP, and Print. Using neural networks and decision trees, the overall success (correct identification of applications) of the ML systems ranged from a low average of 65.8% for neural networks to a high of 85.9% for decision trees. These averages are a result of all classification attempts including the case where only one application is accessing the medium and also the unique combinations of two and three different applications.

MACHINE LEARNING TECHNIQUES FOR CHARACTERIZING 802.11B ENCRYPTED DATA STREAMS

I. Introduction

1.1 Motivation

As wireless networks become an increasingly common part of the infrastructure in industrialized nations, the capabilities of this technology needs evaluation. Due to the inherent mobility of these networks, they have been implemented in many tactically mobile sectors such as the medical community and the military [JVZ01]. These areas often require more secure communications than other users of this technology due to the sensitive nature of the missions. Wireless networks are unique; the transmission channel is not secure. Also, wireless networks typically have lower data rates and higher error rates than a wired network. The broadcast nature of wireless networks means they are much simpler targets for information warfare attacks such as jamming, interjection of spurious traffic, and traffic analysis [Gei02].

Most traffic analysis attacks rely on header information such as the sender and receiver IP addresses and protocol in use to gain information about a network [ChA99]. Several methods of restricting access to headers have been proposed [FKK96], [GFX01], [GLX99], [JVZ01], [WoV91]. However, there is other information available in wireless packets to would be attackers. This information includes packet size, channel utilization, signal strength, and packet inter-arrival times. For groups implementing wireless

networks (especially the military), an analysis of the ability to exploit these other characteristics is clearly important. Such an analysis has implications for both offensive and defensive information attack.

1.2 Overview

The IEEE 802.11b [P802.11] protocol is the most common wireless network used today [And98]. This protocol uses the 2.4 GHz industrial, scientific and medical (ISM) band and a direct sequence spread spectrum modulation scheme. The 802.11 protocol supports data rates ranging from 1 to 54 Mbps with the 802.11b specification supporting 1, 2, 5.5 and 11 Mbps. There are two operating modes for 802.11b networks: infrastructure and ad-hoc. In infrastructure mode, wireless nodes are connected to an existing wired network through a wireless access point. In ad-hoc mode, nodes in the network communicate directly to each other without the use of any traditional wired network. By definition, this type of network must include at least two computers. An example of this is shown below in Figure 1-1. This is the type of network considered in

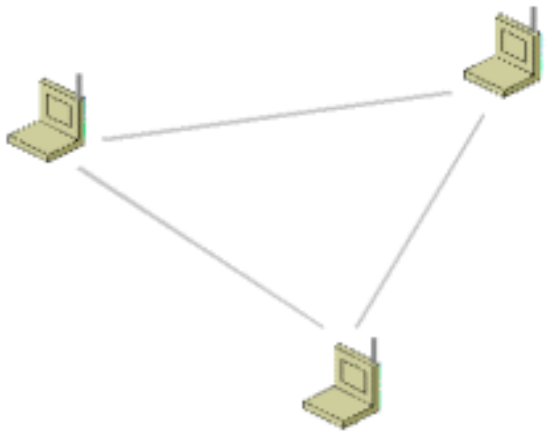

Figure 1-1 802.11b Wireless Ad-Hoc Network

this research. The 802.11b wireless networks are generally capable of two different encryption and authentication schemes. The first type of encryption known as Wired Equivalent Privacy (WEP), uses a shared secret key to both encrypt and decrypt messages. This key must be configured on all the wireless clients attempting to communicate in the wireless network and is usually between 40 and 152 bits in size depending on the vendor. A 40-bit key is specified in the 802.11b standard. Although WEP is known to be vulnerable to various attacks, other standards for wireless encryption are under development. IEEE 802.11i specifies certain improvements to wireless networking security. While this standard is being developed, wireless vendors have agreed upon an interoperable interim standard known as Wi-Fi Protected Access (WPA) [Gri02]. WPA replaces WEP's weaker encryption algorithm with the Temporal Key Integrity Protocol (TKIP). Unlike WEP, TKIP provides a unique starting key for each authentication and also synchronized changing of the encryption key for each frame.

The trend of moving toward tougher encryption standards is sure to continue as wireless networks begin to carry the same sensitive data as wired networks. It is this trend and the trend of obscuring header information that drives the need for other analysis techniques for wireless transmissions.

Research has shown that attributes other than the packet headers provide valuable information about the nature of transmissions on a wireless network [Bel97], [ChA99]. Information such as packet size can be captured easily due to the broadcast nature of the wireless medium and used in inferring information about the nature of the transmissions. There are techniques that could hide such information like traffic padding, changing the maximum transfer unit (MTU) size, and making all packets the same size, but these

techniques all require significant overhead and cause a reduction in useful throughput [KeA98]. It is the combination of negative bandwidth implications and the failure to appreciate the rich information to be gained from packet analysis that makes the use of such techniques to defeat that analysis unlikely [JVZ01], [KeA98].

The goal of this research is to develop an automated algorithm to characterize wireless traffic. More specifically, this algorithm will identify what applications are accessing the wireless channel. In order to automate such a system, machine learning (ML) techniques are used. Machine learning deduces patterns, regularities or rules from past "experiences" or samples. Neural networks and decision trees are used to infer information from 802.11b packet attributes without examining the data contained in the packets themselves which are assumed encrypted. Taking some of the unique attributes described above as input, these techniques will classify wireless transmissions into applications that are accessing the channel.

1.3 Thesis Organization

This chapter presents the motivation for the research and an overview of the concepts involved. Chapter II provides background in the area of traffic analysis techniques, and further discusses different methods of automatic data classification and machine learning. Chapter III discusses the methodology used in this research. System boundaries, parameters, workloads, and factors to be varied in the research are explained. Chapter IV contains the data collected and results of analysis techniques performed as well as an analysis of the variance of the data. Finally, Chapter V presents conclusions, limitations, and suggestions for future research in this area.

II. Literature Review

2.1 Background

Whether discussing voice communications or packet information on a network, obtaining the source information has always been more difficult than obtaining meta information, or information about things such as routing and timing characteristics. For example, wiretaps are so expensive to implement, and require such a high level of evidence to be presented to a judge before being authorized; police use them only as a tool of last resort. In contrast, the phone numbers a suspect calls, and the numbers of those who call him provide valuable information. The police use this meta data to infer information about the suspect and those he contacts. In 1998, there were 1,329 full wiretap applications approved, while there were 4,886 warrants for pen registers (devices that record all the numbers dialed from a particular phone line) and 2,437 warrants for trap-and-trace devices (which record the calling-line phone number of incoming calls) [And01]. This means that there were approximately 11 times as many warrants for communications data as there were for actual content.

Disregarding the legal issues of reading the contents of packets on a network, there are still the technical problems. It is fairly difficult to examine high data rate packets even when they are not encrypted. Add to this the trend towards tougher encryption standards for wireless traffic, especially since the relaxation of the US encryption export rules in 1999, and it becomes much more difficult to decrypt these packets and will probably soon require an inordinate amount of time and resources [GFX01]. For this reason, other methods for analyzing traffic are needed. Many intrusion detection systems use unencrypted header information to create classification

rules for attacks [Mar01]. However, there has been recent work on hiding this information for both wired and wireless networks including IPsec, Onion Routing, the Non-Disclosure Method (NDM), and the Dynamic Mix Method for Wireless Ad Hoc Networks [KeA98], [RSG98], [FKK96], [JVZ01]. In light of this, this research assumes that all packets have both the contents and the header information encrypted.

The following sections contain an in depth review of techniques used to monitor and characterize network traffic as well as techniques to characterize and classify information in general.

2.2 Current Related Research

There are many papers and articles on traffic analysis techniques. Many of these use routing information from packet headers [FKK96], [GFX01], [GLX99]. There are also a number of papers dealing with the prevention of traffic analysis [JVZ01], [WoV91], [WoV93]. Most prevention techniques aim to hide source and destination node information. There are several interesting research efforts that deal with other aspects of traffic analysis and data characterization that provide support and motivation for this study. These include a probability based attack on encrypted IP headers, a signature based attack for analysis of SSL encrypted web browsing, and a timing analysis attack on SSH [Bel97], [ChA99], [SWT01].

A probability based attack on encrypted IP headers is more effective if traffic analysis is done first. Different protocols and applications have their own characteristic traffic patterns or signatures. For example, SMTP has a series of short data packet exchanges between the two nodes, followed by a longer message from the client, and another set of brief exchanges. HTTP exchanges consist of a few hundred bytes sent in

one direction, followed by at least several hundred bytes in the other direction [Bel97]. By identifying the type of protocol in use, more information is available about the probable contents of the headers. The format of the IP header is shown below in Figure 2-1. The idea is that certain areas in an IP header can be predicted even though

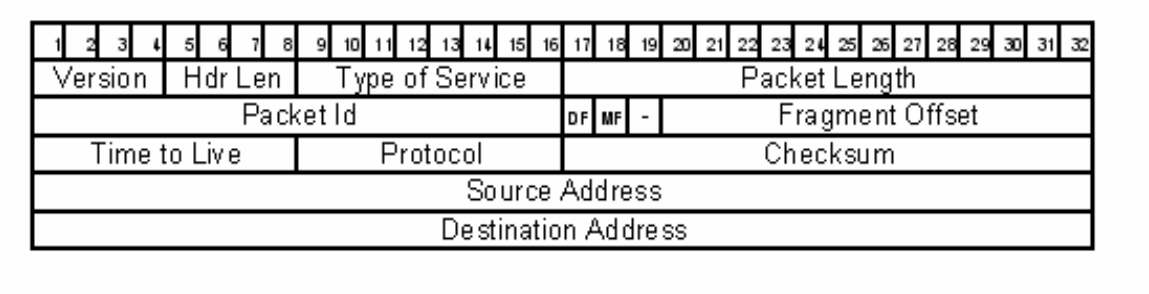

Figure 2-1 Format of the IP Header

encrypted. For example, the version number value is always 4_{16} (representing IP version 4); the header length is very often 5_{16} (which shows that there are 5 32 bit words or 20 bytes in the header), and the type of service field value is very often 10_{16} [Bel97]. This reveals useful information to use in conjunction with various cryptanalysis techniques and software. If padding is not used, the length attribute can be determined for certain packets based on traffic analysis of the actual packet size. Even in the presence of traffic padding, analyzing the distribution of lengths should yield information since the relative number of certain packets in traffic have been analyzed [Bel97]. For example, ACK packets are known to represent about 30 to 40 percent of those on the Internet and these have a length of 28_{16} (40 bytes).

The next area of related research is the traffic analysis of secure sockets layer (SSL) encrypted web browsing. The SSL protocol is an application layer mechanism widely used for encrypted Web browsing. However, SSL was not designed with traffic analysis in mind. One approach used to analyze SSL traffic uses the sizes of known Web

pages and identifies when those pages are downloaded by a user [ChA99]. Since HTML files can be of arbitrary length, the sizes of particular web pages are often unique among files at a site. Figure 2-2 below shows the size of Web pages served by the U.C.

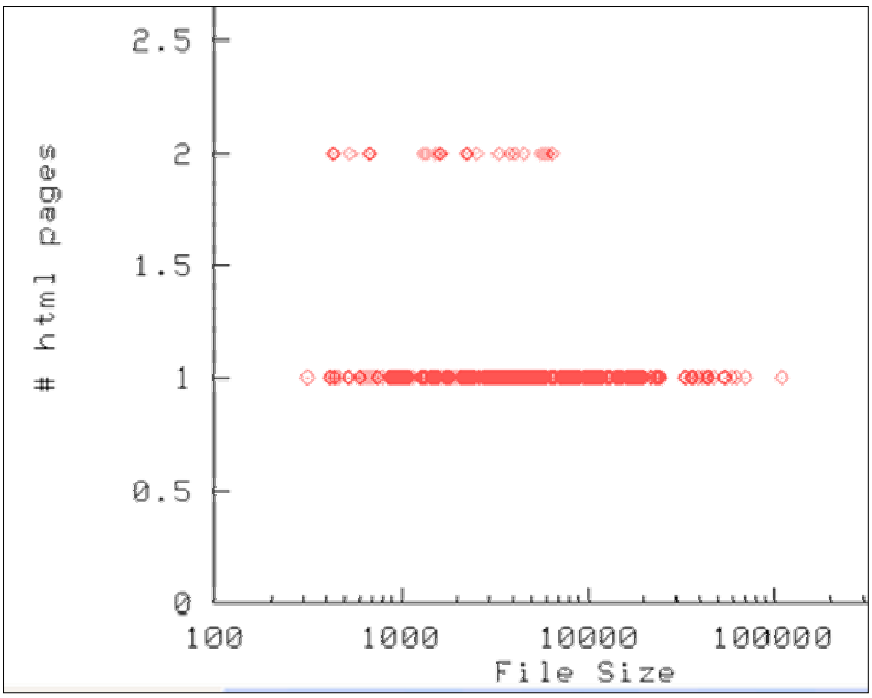

Figure 2-2 Histogram of HTML page sizes served by U.C. Berkeley Web Site Berkeley Extension Web site. Out of the approximately 500 pages on the site, only about ten percent are not unique.

To understand how this particular form of traffic analysis attack is performed, a thorough understanding of the protocols used for Web browsing is needed. HTTP, for example, has a simple procedure for downloading Web pages. First, the client browser sends a request for a page. The server responds with a stream of IP packets containing the HTML code for the page. This code contains references to other embedded objects, such as images, which the browser must also fetch from the server. After receiving and

parsing the HTML, the browser issues requests for all of the embedded objects. The characteristics of the protocol are used to create a database with size information for various encrypted and unencrypted Web pages taken from captured traffic. It is interesting to note that the only difference between encrypted and unencrypted sizes is that the encrypted files are a constant byte amount larger than the unencrypted ones [ChA99]. This suggests that a hidden Markov model (HMM) with pages visited corresponding to hidden states and the hyperlinks to state transitions would work well. HMMs are discussed in Section 2.3.

In what is perhaps the most novel of these related analysis techniques, timing analysis attacks performed on packets from an interactive Secure Shell (SSH) session revealed inter-keystroke timings [SWT01]. Traffic analysis reveals the exact timing of the transmission of the password since every keystroke is sent in a separate packet. Knowledge of the SSH password protocol allows a recognizable "signature" to be created to indicate when passwords are about to be entered. The signature consists of three twenty byte packets sent for the password request, followed by an echo of two twenty byte packets from the remote host and a twenty-eight byte packet for the "password:" prompt as shown in Figure 2-3. The local machine sends each character of the password

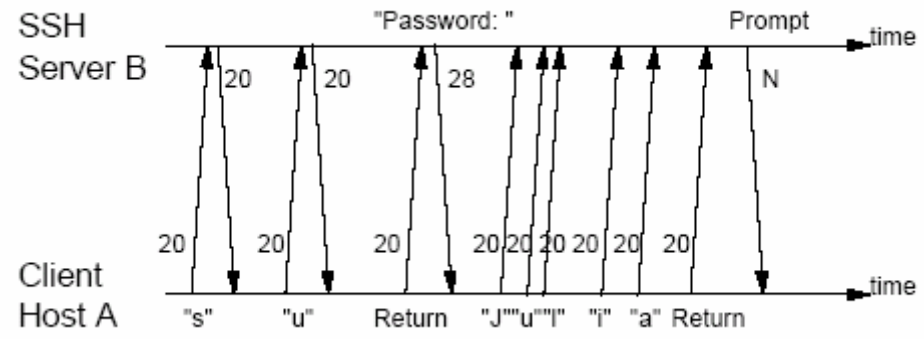

Figure 2-3 "Signature" of SSH Password Prompt Entry

in twenty byte packets until it either successfully logs into the local host or fails indicated by other size packets from the host.

A hidden Markov model is used to model the behavior of the inter-keystroke timings. In this example, the pair of keys is considered the hidden state, and the inter-keystroke timing the observable output. Using this approach, only $1/50^{th}$ the number of guesses were required to identify the correct password compared to a brute force search. This represents a gain of 5.7 bits of information per password guessed using the latency information [SWT01]. Thus, information other than that available in packet headers can assist in inferring important characteristics about network traffic.

Although these traffic analysis techniques are promising note that there are also numerous papers and articles revealing methods for preventing this type of analysis. Many of these traffic analysis prevention techniques involve the information assumed to be unavailable in the context of this effort, namely the packet header routing information. Techniques used to hide this information include link encryption and the insertion of dummy traffic into the network (traffic padding) [GLX99], [JVZ01]. Another technique used to defeat traffic analysis is to force all the packets in a given network to be a certain size. This can also happen as a result of fragmentation due to underlying networks' maximum transfer units (MTU) [Mar01]. Forcing constant packet sizes results in a reduction in useful bandwidth since many packets have useless padding material [WoV91]. To defend against timing attacks, round trip times (RTT) could be padded, increasing all RTTs to worst case round-trip times. Of course, this is also a major inconvenience for users. Given these inconveniences, and more importantly, worst-case delay requirements for some systems, it is questionable whether networks system

administrators will employ methods to defeat all traffic analysis techniques [GFX01]. Before any serious traffic analysis can be performed, a decision has to be made about how to collect and analyze the data. The next section covers several different methods for data analysis that have been applied to network traffic information in various ways.

2.3 Data Analysis Techniques

There are many interesting characteristics that can be captured and analyzed in networks. Some of these include source and destination addresses, source and destination port numbers, packet size, packet inter-arrival times, channel utilization, and signal strength in wireless traffic. Techniques used to analyze this data range from simple empirical observation to complex models which infer information from the traffic stream using machine learning. Statistics involves fitting models to data and making inferences from those models [Mar01]. Since this coincides well with the objective in this research, several of these statistical methods are described below.

2.3.1 Statistical Pattern Recognition

Pattern recognition is one subset of the larger class of data mining techniques. In general, data mining is performed on much larger sets of data than pattern recognition. The automatic recognition and classification of patterns by machines represent statistical pattern recognition [JDM00]. The targets of pattern recognition techniques include fingerprints, handwritten letters and words, the human face, speech signals, and network information among others. In pattern recognition, there are two major approaches: supervised and unsupervised learning [HTF01]. So called supervised learning gains its name from the presence of an outcome variable to guide the learning process of the model. For example, an outcome measurement could be heart attack versus no heart

attack or something more complex like different categories of network traffic. Unsupervised learning examines data without any knowledge of the outcome and characterizes it based on how it is organized or clustered. Sometimes it is useful to use both techniques. The unsupervised technique might be used to determine unique characteristics in network traffic that could be used for inferring useful data. Evaluation of the resulting categories or clusters of data found could be used to train a model in a supervised way by deciding on certain outcome variables that can be determined.

The design of a pattern recognition system generally includes the following three aspects: 1) data acquisition and preprocessing, 2) data representation, and 3) decision making [JDM00]. One of the simplest techniques in pattern recognition, template matching compares a captured sample against templates or prototypes stored in a database. Quite often, templates are learned from a training set of data. If a relatively small number of parameters are observed in network traffic, this method may prove to be an accurate way to infer important characteristics.

More formally, a pattern recognition system operates in two modes that include training or learning mode and classification or testing mode as seen in Figure 2-4. The

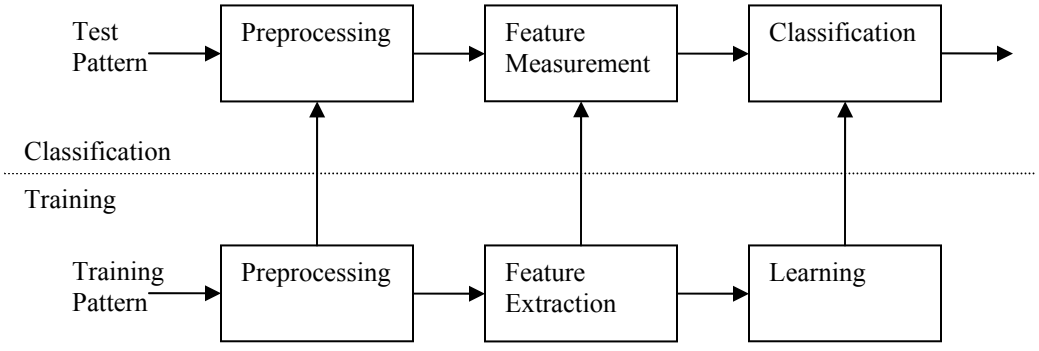

Figure 2-4 Model for Statistical Pattern Recognition

role of preprocessing is to separate interesting features of the data from those that are not statistically interesting. The user can intervene by examining these interesting features and selecting those appropriate for extraction. Next, the system is trained to separate the feature space according to the selected features. Classification mode works in a similar fashion by separating the feature space and assigning the results to specific categories learned in the training process.

Perhaps one of the most important aspects of pattern recognition is the error estimation of the classifications. This is especially true in the design phase of the pattern recognition model where different classifiers and combinations thereof can be selected from among the available features. The error rate of a recognition system can be estimated by using the percentage of misclassifications of the test data. The training and test sets need to be sufficiently large and independent in order for this estimation to be reliable in predicting future classification performance. The following example attributed to [JDM00] helps to clarify this idea. Given a classifier, suppose that τ is the number of samples out of a total of n that are misclassified. It can be shown that the probability density function of τ fits a binomial distribution. It follows that the maximum likelihood estimate, $\hat{P}_e$ is given by τ/n, with $E(\hat{P}_e) = P_e$ and $Var(\hat{P}_e) = P_e(1-P_e)/n$. This shows that $\hat{P}_e$ is an unbiased estimator and a confidence interval can be calculated. For example, if $n = 250$ and $\tau = 50$, then $\hat{P}_e$ is 0.2 and the 95% confidence interval for $\hat{P}_e$ would be (0.15, 0.25). Given two classifiers, if the mean of either is included in the others' confidence interval, their performance is statistically equivalent. Since a larger n reduces the confidence interval, a large sample is desirable. However, one has to consider the partitioning of the total set of available data as well. In other words, using too few

training examples so that more can be used for testing to decrease the confidence interval will likely lead to higher error levels. Some method for determining the ratio of training sets to test sets must be determined. Table 2-1 discusses some possibilities. If the amount of data used for training is too small, the classification technique will have poor generalization ability. This is an intuitive result especially if the samples used for training do not represent all of the class possibilities. On the other hand, using too much

Table 2-1 Error Estimation Methods

Method	Property	Comments
Resubstitution Method	All the available data is used for training and test sets are the same.	Optimistically biased estimate, especially when the ratio of sample size to dimensionality is small.
Holdout Method	Half the data is used for training and the remaining data is used for testing; training and test sets are independent.	Pessimistically biased estimate; different partitioning will give different estimates.
Leave-one-out Method	A classifier is designed using (n-1) samples and evaluated on the one remaining sample; this is repeated n times with different training sets of size (n-1).	Estimate is unbiased but it has a large variance; large computational requirement because n different classifiers have to be designed.
Rotation Method, n-fold cross validation	A compromise between holdout and leave-one-out methods; divide the available samples into P disjoint subsets, $1 \leq P \leq n$. Use (P-1) subsets for training and the remaining subset for test.	Estimate has lower bias than the holdout method and is cheaper to implement than the leave-one-out method.
Bootstrap Method	Generate many bootstrap sample sets of size n by sampling with replacement.	Bootstrap estimates can have lower variance than the leave-one-out method; computationally more demanding; useful in small sample size situations.

data to train and only a small amount to test may lead to an inaccurate estimated error rate. Deciding which method to use is more of an art than a science and may require

experimentation with the collected data [JDM00]. If a large number of data sets are available, all of the techniques described above are likely to approach the same error estimations.

Statistical pattern recognition works very well especially when the feature space (*d*) is small compared to the number of training samples. This is known as the peaking phenomenon [JaC87]. One problem with statistical pattern recognition is determining the period to be examined. In other words, what amount of time is one sample for the purposes of training and testing? In order to answer that question, a thorough understanding of the data to be used and the problem being solved is required. Consider Internet traffic for example. Making the decision of how many packets to use for one sample depends on the purpose or goal of the classification. Once the goal of the classification attempt is understood, the salient features can be analyzed. Those features that yield the most information toward the goal of the classification would be used for training. Examining the Internet traffic, it can be determined how often the classifying characteristics appear. This information is then used to determine an appropriate size for the samples. Another closely related and popular machine learning technique used for classification is artificial neural networks (ANN).

2.3.2 Artificial Neural Networks

Neural networks fall into the artificial intelligence (AI) and machine learning categories. Neural networks are learning and classification structures modeled after the human mind. As such, they are composed of small components or neurons which accept an input, make a decision on that input, and forward an appropriate response. These neurons function together to process information in a parallel fashion. The neurons of an

ANN are connected by weighted links over which signals can pass and it is in these links that the actual learning and intelligence of the ANN resides. There are different configurations of connections for these links but one common one, the feed forward network, is shown in Figure 2-5. Signals are presented to the network at the input layer. In a typical network, the input neurons do little but forward the incoming signals. Changing these signals from input to an output response is the job of the transfer function

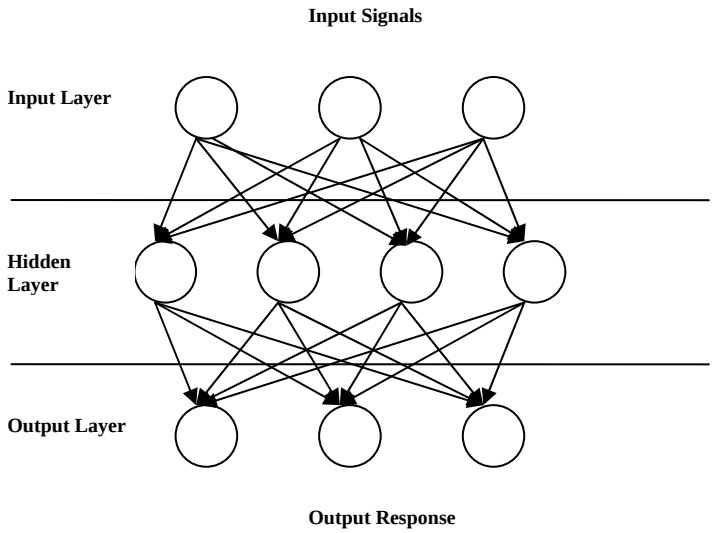

Figure 2-5 Typical Neural Network Architecture

which operates in three steps. In the first step, the neuron computes the net weighted input it received from its input connections. This can be done in different ways but a common formula is

$$Ii = \sum_{j=1}^{n} w_{ij} x_j \qquad (2.1)$$

where *Ii* is the net weight of the inputs received at neuron *i* from the *n* nodes in the network, w_{ij} is the weight from neuron *j* to neuron *i*, and x_j is the output from neuron *j*. In other words, each neuron receives input from every neuron in the layer before it and

these inputs are summed together after being multiplied by the weight of the corresponding connection. Weights can be negative and therefore inhibitory in nature or positive and excitatory. During the second step, the transfer function converts *Ii* to an activation level. There are several different functions for doing this, but one commonly used function is that of a sigmoid or S-shaped curve. A common formula used for this is

$$f(I) = \frac{1}{1+e^{-I}} \tag{2.2}$$

which effectively maps the signal to something between 0 and 1. The third step for transferring the signal takes this function output and compares it to a global threshold value. If the value exceeds the threshold, a common practice is to simply forward the value itself although any new value could be used while a 0 is forwarded if the threshold is not surpassed. In this way, a signal propagates through the neural network and is transformed to an output signal.

The way a neural network "learns" its classification behavior is through modification of the weights on the connections. There are many different learning algorithms that can be implemented within the neural network, but the most common one is backpropagation. All of these algorithms work by updating the weights on the connections between neurons according to the amount of error calculated by taking the difference between the actual output of the system and the correct output.

Neural networks have been applied successfully to a wide range of data mining, prediction, and classification problems including hand writing or optical character recognition, speech recognition, stock market forecasting and many others. Neural Networks are appealing because the restricted hypothesis space bias or constraints on the hypotheses that an algorithm is able to construct, is well suited for sequential and

temporal prediction or classification tasks [CrS97].

On the negative side, neural networks suffer from the fact that the reasoning behind their results is often difficult to understand or explain. This problem is often referred to as the "black box" effect of neural networks and limits their use in some areas where information about how classifications or decisions are being made is as important as the decisions themselves. A data mining technique which does not suffer from the black box effect is that of decision trees.

2.3.3 Decision Trees

Decision trees are often used in data mining for prediction and classification purposes. One of the most powerful characteristics of decision trees is the simple model structure they have. It is a straight-forward matter to transform a decision tree into a set of logical "if-then" rules.

Decision trees are produced in a recursive manner by selecting an attribute to split on and placing it at the root node. One branch is created for each possible value. If the data is discrete, the number of branches is equal to the number of categories for that attribute whereas continuous numerical data is normally split in a binary fashion with one branch being less than and the other greater than or equal to the chosen split. In order to determine which attribute from a set to split on, decision trees make use of different measures of node purity. Node purity has several factors associated with it. First, the major goal of node purity is to have only one of the classes of the decision tree model represented. In other words, if the classes were yes and no, a node with all no instances is pure. This purity could be achieved simply by partitioning the samples so that there was one node for each. This would quickly lead to large decision trees for anything other

than a trivial case. To achieve models which make correct classifications while maintaining more reasonable decision tree sizes requires that the test for purity also favor those nodes which are not only pure but also have more members. There are different measures of purity in use today, but one of the most common ones is called the information of the node and is measured in bits. This measure of purity will be explained with the introduction of a simple decision tree example from [Qui86].

Table 2-2 below summarizes the weather data used for this example. The class

Table 2-2 Weather Data

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes

being determined is whether or not a game is played given certain conditions. To determine the information gain from splitting on the categories, the purity of each split is tested by arranging the outcome variable within each category as shown in Figure 2-6. The numbers of yes versus no for the classes of outlook are then [2, 3], [3, 0], and [3, 1] respectively and the information values of these nodes are determined by a calculation of the nodes' *entropy* as seen by the samples. Entropy is defined as

$$\text{entropy}\,(p_1,p_2,\ldots,p_n) = -p_1\log p_1 - p_2\log p_2 \ldots - p_n\log p_n \qquad (2.3)$$

where (p_1,p_2,…,p_n) are the fractions of the values at the leaf nodes of a particular split.

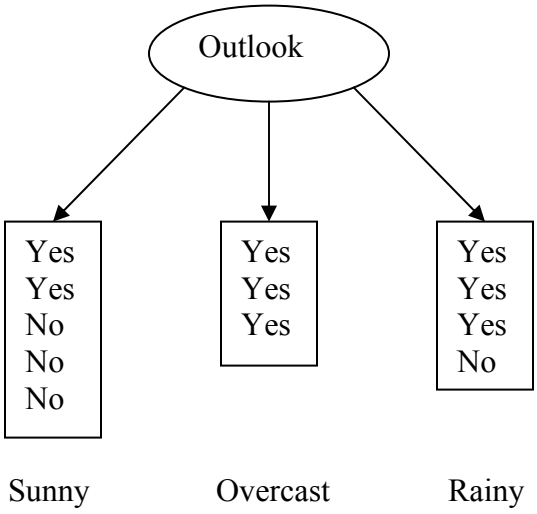

Figure 2-6 Outcome Variable For Outlook

These fractions always sum to 1. Since the logarithms used in equation (2.3) are base 2, the entropy values will always be between 0 and 1. The negative coefficient in front of each logarithm is required to get a positive value for the entropy since the log of a fraction is negative. Using (2.3) we have

$$\text{entropy } ((2/5), (3/5)) = 0.971 \tag{2.4}$$

$$\text{entropy } ((3/3), (0/3)) = 0.0 \tag{2.5}$$

$$\text{entropy } ((3/4), (1/4)) = 0.811 \tag{2.6}$$

The next step is to find an average information value for this particular split. This is done by multiplying the entropy for a particular branch by the fraction of total samples contained by it which gives

$$(5/12) * 0.971 + (3/12) * 0 + (4/12) * 0.811 = 0.675 \tag{2.7}$$

This result means that 0.675 bits of information are needed to classify a new example according to the outlook variable. The original unsplit yes and no data contained eight

yes and four no nodes corresponding to an entropy of 0.918 bits using equation (2.6). By splitting the data on the outlook variable, and simply subtracting the 0.675 bits from the original 0.918 we see that uncertainty is reduced by 0.243 bits. This is the information gain from splitting on the outlook attribute. This same procedure is carried out for each of the remaining splits (temperature, humidity, windy) and the one that has the best information gain (i.e., least uncertainty) is selected. The procedure repeats with the remaining factors until no more splits can be made. If the data presented for the decision tree model was ideal, leaf nodes would have entropy of 0 meaning that there was only one possible classification for any sample reaching that node. Of course, this rarely occurs with real data sets. Decision trees are a greedy divide and conquer approach to data mining and classification and have been made more popular in the past two decades by the proliferation of decision tree programs with good performance.

2.3.4 Discovery of Frequent Episodes in Event Sequences

Many data mining and machine learning techniques are used to discover correlations in unordered collections of data [MTV97]. In many situations, there is order to the data being examined in the form of sequences of events. Certainly, packets in a network may be viewed in this way. This is especially easy to recognize at lower channel utilization levels where a stream of packets in the network represents only a few exchanges between users and hosts. The sequence of events still holds true even for fully utilized channels, but recognizing those sequences is more difficult.

The following example, modified to be more appropriate to this work, is due to [MTV97]. Suppose that the initial handshaking of a protocol makes up events A and B, the data that follows makes up event C, D, and the end of the transfer makes up E and F.

Examining a stream of data could yield a sequence such as shown in Figure 2-7. What is

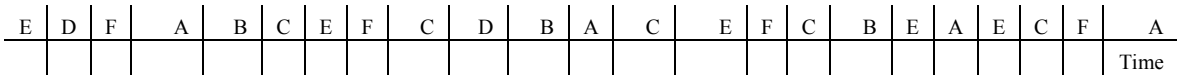

Figure 2-7 Example Sequence of Events

required is a way to examine this sequence of events and determine if there are any frequent episodes (collections of events occurring frequently together). In this example, the events correspond to particular signatures for known network events. However, the events could conceivably correspond to anything as long as they yield interesting or useful information about the data. From the above sample, we could make several inferences. For example, the E and F events occur several times together. Also, it can be observed that whenever A and B occur (in no particular order), C soon follows. It is obvious that some consideration must be given to the period or event window considered when discovering these frequent episodes.

The input is considered as a sequence of events where each has an associated time of occurrence. Given a set of X event types, where an event is a pair (A,t) and A is the type (A, B, C, etc.) and t is an integer representing the occurrence time of the event. An event sequence s on X is a triple (s,T_s,T_e), where $s = \langle (A_1,t_1),(A_2,t_2),\ldots,(A_n,t_n) \rangle$ is an ordered sequence of events [MTV97] and T_s and T_e are integers, $T_s < T_e$, where T_s is the starting time and T_e is the ending time for the sequence. Applying this formalism to the example sequence above results in Figure 2-8. In this figure, the event sequence $s = \langle (E,31),(D,32),\ldots,(A,58) \rangle$ is represented graphically. For each event occurring in the time interval [30, 58), the event type and time of occurrence have been recorded. At this point, the periodicity or length of window is considered. Of interest are all frequent

episodes in a given sequence. In order to be interesting, episodes must occur close

Figure 2-8 Example Sequence with Two Windows of Width 5

enough together in relation to time to yield useful information about the events. This choice is arbitrary, and will require experimentation with the actual data sets before choices yielding useful information can be made. In Figure 2-8, the time chosen (known as the window width) is 5. One way to proceed in this type of experimentation is to start with very small window widths, where it is unlikely many episodes will be found, increasing the size until the number of episodes becomes too large to yield any useful results. This provides a good upper and lower bound for useful window definitions. Note that the windows overlap since dividing the sequence into non-overlapping sub-sequences could cause lead to missing important episodes. Another parameter that can be varied to achieve different results is the number of windows an episode must occur in before being considered frequent. Episodes are partially ordered collections of events occurring together and can be described as directed acyclic graphs. Consider episodes α, β, and γ in Figure 2-9. Episode α is a serial episode in which event E occurs first, followed by event F with other events possibly occurring in between. This is important

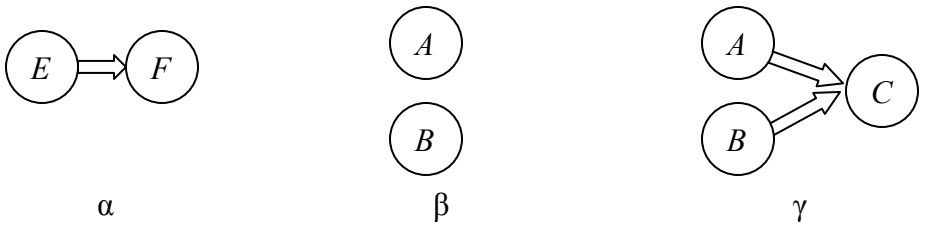

Figure 2-9 Episodes α, β, and γ

especially when considering the multi-access nature of networking. Episode β is a concurrent episode with no constraints on the order of A and B. Event A could have preceded B or B could have preceded A. Episode γ is a combination of the others where A and B occur in some order before C. There are definite parallels to this type of "rule" in the arena of intrusion detection systems where known attack signatures occur in a certain order.

Serial episode candidates can be recognized by using state automata that accept them and ignore all other input. Any number of automatons for the same episode can exist at the same time. The basic idea is each time the first event of an episode comes into the current window, a new automaton is initialized. When that event leaves the current window, the automaton is removed. In this way, the automata can keep track of what may be different episodes reaching different levels of completion. This will be conducive to discovering multiple similar events in a stream of packet data.

Keeping an array, for example α.initialized[i], for each automaton with values of initialization times for individual instances will allow the removal of the automata at the proper time. Further, a list (waits(A)) of automata waiting to reach their next states is kept. A number of algorithms have been developed for the discovery of episodes and frequent episodes in data [MTV97].

Experiments performed on sequences residing in flat text files representing a telecommunication network fault management database with 73679 alarms over 7 weeks showed greatly varying numbers of frequent episodes resulting from changing the frequency threshold from 0.001 to 0.100 [MTV97]. However, the number of frequent episodes increased as a function of the window width as can be seen in Figure 2-10. In

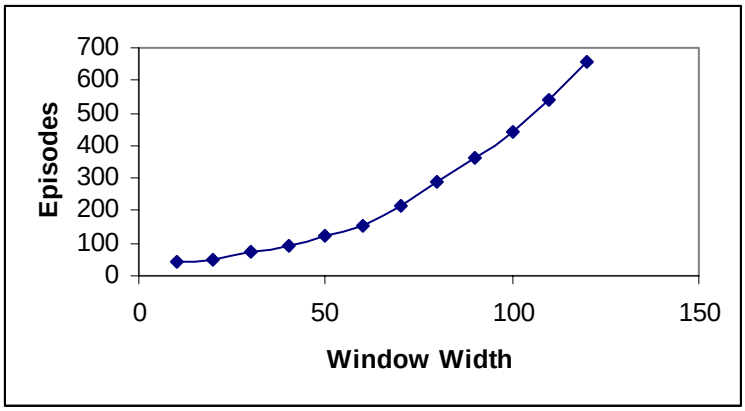

Figure 2-10 Frequent Episodes as a Function of Window Width

addition to the experiments on the alarm database, others were run on a variety of different data to show the methods general applicability. These other data sets are shown in Table 2-3. The WWW row in the table corresponds to part of the server log from the Department of Computer Science at the University of Helsinki. The researchers considered the page fetched as the event type. The slower input rate of users in the WWW row as compared with the alarm data in the telecommunications experiment

Table 2-3 Experiments with Various Data Sets

Data Set	Events	Event Type	Time Win.	Threshold	Freq. Epis.	Rules
Alarms	73679	287	60	0.8	826	6303
WWW	116308	7634	120	0.2	454	316
text	5417	1102	20	0.2	127	19
protein	4941	22	10		21234	

required a doubling of the window time as shown in the data above. However, it is appropriate to use a relatively small window to reduce the probability of incorrectly correlating unconnected events. Some interesting information discovered with the WWW experiments include the fact that students rarely ever use bookmarks to access

course pages, but rather navigate there from department home pages. The ability to vary the window width and frequency used for discovery of frequent episodes could be used to help discern multiple instances of similar events in relation to packet information streams may be useful.

2.3.5 Hidden Markov Models

Hidden Markov models were initially introduced and studied in the late 1960s and early 1970s [Edd00]. They are extremely rich mathematical structures and can therefore form the theoretical basis for many applications, especially where formalism is important. The underlying assumption of the statistical model used in Markov and other stochastic models is the signal examined can be characterized as a parametric random process, and the parameters of the stochastic process can be determined or estimated in a well-defined manner.

Of course, HMMs are based on Markov chains. A collection of discrete-valued random variables $\{Q_t \geq 1\}$ forms an n^{th} order Markov chain if $P(Q_t = q_t | Q_{t-1} = q_{t-1}, Q_{t-2} = q_{t-2}, \ldots, Q_1 = q_1) = P(Q_t = q_t | Q_{t-1} = q_{t-1}, Q_{t-2} = q_{t-2}, \ldots, Q_{t-n} = q_{t-n})$ for all $t \geq 1$, and all $q_1, q_2, \ldots, q_t$ [Bil02]. In other words, given the previous n random variables, the current variable is conditionally independent of every variable earlier than the previous n. It is noteworthy that this conditional independence is sometimes not strictly adhered to in practice while still achieving very good results from a HMM. For example, words and sentences follow sets of grammars. Clearly, there are some parts of a word or of a sentence that are conditionally dependent on what comes before them. For example, if we start a valid word with "ch" there is a set number of possibilities for what can follow. However, HMMs have still been used for the recognition of the spoken and written word

with much success.

As a simple example of how HMMs can be used consider tossing a coin behind a curtain [Rab89]. In this scenario, the number of coins and mechanism of tossing is unknown, while the output of the tossing can be observed. To design a HMM for this scenario, the number of states needed to represent the process must be decided. For example, should a single coin or multiple coins be used to represent the unknown process. A single coin model is shown in Figure 2-11 below. In this model, the

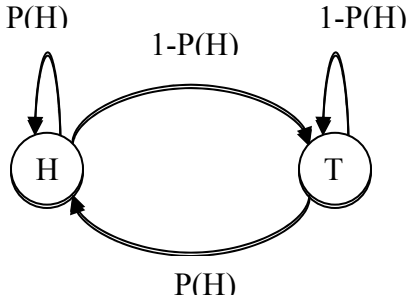

Figure 2-11 1 Coin Hidden Markov Model

probabilities are written as P(H) and P(T) where P(T) = 1-P(H). Since the number of actual coins being flipped is not known, other models may also be appropriate such as two, three, or *n* coin models. The decision of which is the best model can only be accomplished once the different proposed models have been created and tested on the available data. It is in this way that the use of hidden Markov models is considered more of a practice or an art form than a science. Only through experimentation with the underlying model can the best representation be found. The number of states and how they are connected is normally a task that is accomplished by hand with extensive knowledge of the characteristics of the data to be searched. However, efforts have been made to develop algorithms to learn the needed architecture for general HMMs [Bil02].

Once a proper model is created, there are two ways it can be used. A HMM can be used to produce a stream of output that is similar to what is produced by the system it is modeling. Using the coin example, the HMM could produce a stream of heads and tails that should represent one possible outcome in the real system. Another way to use HMMs is to take the observations from the system and choose a corresponding state sequence that best explains the observations [Rab89]. In other words, HMMs can be used for classification purposes.

HMMs have been used effectively for many different problems such as hand writing recognition and speech recognition. They have also been applied to packet information as in the case of the keystroke timing attack mentioned in Section 2.2. In this specific example, the character pairs are considered to be unobservable states and the latency between keystrokes are the observable output [SWT01]. Several assumptions have to be made in order to model the data in this fashion. First, the probability of transitioning to any other state or key has to be independent of the previous states as mentioned above. This assumption is true for passwords chosen at random, but not in the case where passwords are chosen based on dictionary words or close groupings of letters. However, HMMs work well even when the conditional independence rule is not strictly followed as mentioned earlier. Second, the probability distribution of the latency timing is only dependent on the current pair and not on any previous characters in the sequence. Of course, this is another relaxation of the formal independence rule since reaching for a far away letter in a previous sequence can have some effect on the latency. Once again, this does not seem to affect the ability of the HMM to provide useful results [SWT01].

HMM parameters can either be obtained by training initially unlabeled sequences

or built from sequences where the state paths are assumed to be known. Training algorithms are often used when a plausible alignment for the sequences in question is not already known. The standard training algorithm is a Baum-Welch expectation maximization based on gradient descent.

2.4 Summary

This chapter discusses techniques to characterize encrypted packet streams. The chapter begins with some background into why this research is needed. Next, current research in the area of traffic analysis that relates directly to this effort is covered. Finally, techniques of data analysis and classification methods are discussed and shown to be applicable in the area of network traffic analysis.

III. Methodology

3.1 Background

In Chapter II, current research in network traffic analysis and methods for the analysis and interpretation of network data are presented. Three different types of traffic analysis are described to include a probability based attack on encrypted IP headers [Bel97], a signature based attack for analysis of SSL encrypted web browsing [ChA99], and a timing analysis attack on SSH [SWT01]. This research shows that the study of traffic characteristics yields useful information about the network and that further research into this area is warranted.

The techniques used to analyze and interpret the wireless 802.11 traffic in this research include training a neural network and a decision tree model, both of which are machine learning approaches. Pilot studies are conducted to determine the relevant factors and the settings for the different techniques that provide appropriate performance and these will be discussed in Section 3.2.2 and in more detail in Chapter 4.

As the trend toward wireless communications continues to increase, the possibilities of in depth traffic analysis, even of encrypted data, need exploration in order to provide both offensive and defensive capabilities.

3.2 Problem Definition

3.2.1 Goals and Hypothesis

The primary goal of this research is to determine which applications and how many instances of each are accessing the wireless medium during a given time window under the assumption that the traffic is encrypted. It has been shown that unique characteristics of applications are manifested within packet transmissions due to the

802.11 protocol [And98]. This research shows that such characteristics can be recognized using machine learning concepts and therefore demonstrates that the identification process can be automated.

3.2.2 Approach

Wireless data is captured and machine learning techniques trained to recognize which applications are accessing the channel. The two techniques used for testing are neural networks and decision trees. Data streams are analyzed based on a sliding time window where the size of the window varies. For example, the window may be set to a size of 11 packets if that is sufficient information for the neural network and decision trees to make correct decisions a given percentage of the time at a certain confidence level. A sliding window for network data is important for determining how many instances of applications are accessing the network and has been discussed in Chapter II [MTV97]. The results from the different algorithms and settings within algorithms are compared and analyzed in order to find the most effective technique. For example, the neural network technique shows an appropriate level of performance when using back propagation to reduce the mean squared error after each training epoch. Back propagation is a form of supervised learning used in neural networks whereby the inputs to the network must include the sample to be analyzed and the anticipated or desired outputs in order to calculate the error for a particular training cycle. An important choice when using decision trees is whether or not to allow pruning. Pruning both reduces the size of decision trees and also increases the ability of the system to generalize.

3.3 System Boundaries

The System Under Test (SUT) is called the Application Determination System

(ADS) as shown in Figure 3-1. This system includes the machine learning algorithm used for identification, size of the time window used for training, percentage of

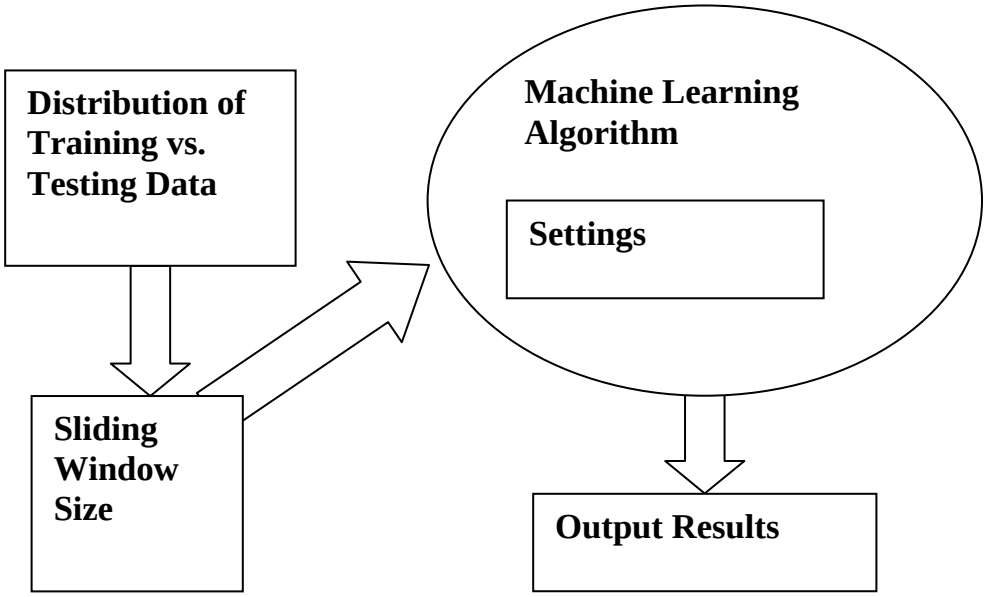

Figure 3-1: Application Determination System

data used for training vs. testing, and parameters that apply to the algorithms used. Within this system, the Component Under Study (CUS) is the algorithm used (neural network vs. decision trees) to classify the data.

Although this work is being carried out on an ad-hoc IEEE 802.11b wireless network, the results are applicable to any network where packets have not been padded to prevent analysis. Machine learning techniques are inherently resistant to network protocol differences [KaV94]. No attempt is made to decrypt information in the packets or the headers. It is assumed this information is too difficult to obtain [Mul02]. It should be noted that machine learning techniques are equally adept at learning both encrypted and unencrypted traffic characteristics. This study limits the applications accessing the medium to e-mail, ftp, http, and print jobs.

3.4 System Services

The ADS system identifies the type of applications accessing the medium. There are three possible outcomes of this identification process. One possible outcome is the correct identification of all applications accessing the channel. A second possibility is the false positive identification of applications. False positives occur when the system reports that applications are present on the channel when they are not. The third possibility is the combination of both true and false positives. Since the designs of both the neural networks and decision trees in this experiment force all samples to be placed into one of the four categories, there is no possibility of an unknown classification.

3.5 Performance Metrics

The ADS is similar to some commercially available intrusion detection systems except that it looks for a "signature" to identify an application rather than an attack [And01]. Therefore, one way to measure the performance of the ADS is to use a Receiver Operating Characteristic (ROC) curve. A ROC is a graph of correct identifications (true positives) as a function of incorrect ones (false positives) or, in other words, the sensitivity versus specificity of the system. While in an intrusion detection system, the ROC measures attacks identified over false alarms, in the ADS it instead measures the ratio of correct identification of applications to incorrect ones. Consider the following modified extension of Bayes' theorem:

$$\mathrm{P}(M|IDM) = \frac{P(M)P(IDM \mid M)}{P(M)P(IDM / M) + P(\neg M)P(IDM \mid \neg M)} \tag{3.1}$$

where M is e-mail accessing the medium and IDM is the correct identification of e-mail [Mar01]. For example, if the probability of e-mail accessing the medium, correctly identifying mail, and false identification as e-mail are 0.25, 0.80, and 0.05 respectively,

the probability of correctly identifying the e-mail given an e-mail accessing the medium is 0.842. Another metric of interest is the ratio of correct identifications to the total number of applications for a given time frame. For example, the system may correctly identify 70 out of 100 applications sent in some arbitrary time frame which results in a 70% success rate. This identifies the performance of the system as the third type of possible outcome listed in the system services section. In a similar fashion, a 100% success rate would identify the first outcome, while 0% would identify the second. Since four applications are being targeted for this study, another value to compare the system against is the probability of correct random guessing which is 0.25. In other words, 0.25 is the baseline from which success is measured and a success rate of 30% is only 5% better than random guessing. A combination of these techniques is used to determine the outcomes of the system.

3.6 Parameters

3.6.1 System

The system parameters for the ADS are enumerated in Table 3-1 and include the algorithm used for identification (neural network vs. decision tree model), size of the time window used for training, percentage of data used for training vs. testing, and

Table 3-1 System Parameters for ADS

Algorithm Used:	Neural Network, Decision Trees
Sliding Window Size:	Small, Medium, Large (11, 31, 51 Packets)
Distribution of Data:	Holdout Method, Rotation Method
Algorithm Parameters:	Underlying Model, Learning Method, Activation Function, Etc.

parameters that apply to the particular algorithms used. For the neural network, these parameters include the design of the network itself which includes the number of input, hidden and output nodes, the selection of training method (i.e., back propagation vs. radial basis learning), number of training cycles (epochs), and type of activation function. For the decision trees, the parameters also include the design of the underlying network. Although there are many parameters to consider for each algorithm, pilot studies are conducted to determine the appropriate settings to use.

3.6.2 Workload

Typically, workload parameters are characteristics of service requests to the system. However, in the case of the ADS it is more appropriate to define the workload by the amount of data on the wireless medium. Since packets are either present or not, "service requests" exhibit an on-off characteristic. Thus, the packets on the channel are the workload. The workload, then, varies from no traffic on the medium to many clients and applications competing for the medium at the same time. This workload will affect the way the recognition algorithms are set up. In an 802.11b network, the absence of any applications accessing the medium typically results in a majority of traffic being beacon packets (61 bytes). Since the detection of this "non-application" state is fairly trivial it is not incorporated into these experiments.

3.7 Factors

Table 3-2 shows the factors that are varied. The algorithms used to learn and recognize applications are a feed forward, back propagating neural network and a pruned decision tree model. The sliding window size has been shown to have a major impact in other similar studies of pattern detection and is varied between a small, medium and large

Table 3-2 Factors Varied

Algorithm Used:	Neural Network, Decision Tree
Sliding Window Size:	Small, Medium, Large (11, 31, 51 Packets)
Type and Number of Application(s)	E-mail, Ftp, Http, Print (1, 2, or 3)

level [MTV97]. These levels are 11, 31, and 51 packets respectively. The task of recognizing which applications are accessing the medium is more complicated as the number of applications to be identified increases. The levels for this factor are varied from one to four and the relative performance at each level measured. It is likely that this factor is heavily dependent on the window size.

3.8 Evaluation Technique

For this study, direct measurement is the most appropriate evaluation technique. While techniques for constructing analytical models for common performance metrics such as system throughput and delay are abundant, few if any such techniques exist for modeling the pattern recognition ability of a system. A wireless encrypted data stream is captured and analyzed. Significant characteristics are parsed and separated into groups of data for the testing and training of the machine learning techniques. Since the implementation of both techniques used in this research rely on supervised learning, the knowledge about what applications are accessing the network is used to assist in the learning and evaluation of those techniques.

3.9 Workload

The workload is the number of applications and clients accessing the channel. It ranges from one to three systems. The difficulty of identifying the correct number of

applications is expected to increase significantly with the inclusion of each additional system. It may also be the case that the ideal window size used in training and testing will vary according to the workload. For example, with only one system accessing the medium, only one of each type of application can be accessing the medium in a given window. This will likely have a positive effect on the classification ability of the different algorithms and also affect the size of the window required. Another possibility suggested by the pilot studies is the existence of unique packet size "signatures" which can be used to determine the approximate start, end and therefore also the number of applications present on the channel.

Text files taken from the early chapters of this work are used to create the files for the workload. These file sizes included 1, 10, 50, and 100 Kbytes. The text is copied and pasted into the body of the e-mail messages versus being included as an attachment. The copied text is pasted into word pad files and saved for use with ftp. These same word pad files are printed for that portion of the workload. Very simple web pages are also created using the same text. Since the type of files used is not a factor in this study, the same text is used for all applications.

3.10 Experimental Design

There are four factors in this experiment: detection algorithm, sliding window size, types of applications accessing the medium and number of nodes. For the detection algorithm, there are two levels which include a back propagating feed forward neural network and a decision tree model. Pilot studies show that the appropriate levels for the size of the sliding window are 11, 31, and 51 packets. Using a sliding window results in samples that overlap from the data stream. For example, using a window size of eleven,

the first segment analyzed is packet one through eleven and the next segment is packet two through twelve. For the next factor there are four levels: e-mail, ftp, http, and print. The last factor, number of nodes, ranges from one to three. This means that the number of experiments, n, for a full factorial design is

$$n = \prod_{i=1}^{k} m_i = 84 \tag{3.2}$$

where $k = 3$ is the number of factors and m is the number of levels for each factor. The number of experiments indicate a full factorial design is appropriate.

Based on pilot studies, the variability in the performance of the machine learning algorithms using different training and testing sets is fairly small with a maximum range of approximately thirteen percent observed with neural networks and only four percent with decision trees. From these pilot studies, settings that produced good results for the different algorithms were determined. For example, back propagation is used for the training algorithm within the neural network because that technique repeatedly demonstrated good performance with low variability compared to other techniques such as radial basis learning. This low variability means a fairly narrow confidence interval at a 0.05 significance level can be achieved to characterize the performance of ADS with only four replications of the neural network experiments. Since the algorithm used to build the decision trees is deterministic, no replications are necessary. Assuming an SSE of 370 for a sample size of 1000, (values taken from pilot study) the standard deviation of error is

$$s_e = \sqrt{\frac{SSE}{2^k (r-1)}} = 3.40 \tag{3.3}$$

where $k = 3$ is the number of factors and $r = 4$ is the number of replications. The standard deviation of effects is

$$s_{qi} = s_e / \sqrt{2^k r} = 0.491 \tag{3.4}$$

The t-value at 16 degrees of freedom and a 95% confidence is 2.120. Multiplying this by the standard deviation of effect obtained in (3.4) gives the resulting value of 1.041 which gives a confidence interval of

$$q_i \mp 1.041. \tag{3.5}$$

Using the case of the response variable in this study, this means it has a range of (61.96, 64.04) which is acceptable and verifies that a total of 210 experiments (4 replications for NN experiments for 168 plus 42 for decision trees) is appropriate.

3.11 Analyze and Interpret Results

The data gathered is used to calculate confidence levels as indicated above. The confidence intervals are used to show that the system will perform within a specified range 95% of the time. The ratio of correct responses to total number of applications will be compared against both actual data (100%) and random guessing (25%) to determine the performance of the system.

3.12 Summary

This chapter presents the methodology for conducting this research. The goal is to use machine learning techniques to determine the number of each of four different application types accessing the wireless medium. Based on this goal, a procedure is developed to make use of insights gained from pilot studies and direct the rest of the research and experimentation. The ABS is described as the SUT with the specific CUS being the algorithms used to learn the traffic patterns. The system parameters are

described and factors to be varied are selected and explained. Finally, an analysis of the experimental design is covered to include the number of experiments and the number of repetitions required to achieve the goals of the study.

IV. Analysis and Findings

The purpose of this chapter is to present and interpret the findings from this research. The chapter begins with a brief description of the 802.11b network topology used for data collection. Next, an analysis of the collected data describing unique characteristics which allow the machine learning (ML) techniques to classify the different applications is presented. Following this is a more detailed description of the specific algorithm settings used for the neural network and the decision trees. After that, results from the machine learning classifications are presented and analyzed with the final section presenting the ANOVA for the data.

4.1 802.11b Ad-Hoc Network Topology

Three computers, two laptops and one desktop, are configured as members of an Independent Basic Service Set (IBSS) or ad-hoc work group. The computers use Enterasys Networks csi6d-aa-128 IEEE 802.11b cards with a maximum bit rate of 11 Mbps. A fourth computer is used as a passive "sniffer" using Airopeek NX software by Wildpackets and a Cisco Aironet 350 series PCMCIA wireless local area network (LAN) card. Experiments are run with both encryption disabled and also with Wired Equivalent Privacy encryption enabled.

Two of the computers from the work group are set up as servers for ftp, e-mail, and http using Microsoft Internet Information Services (IIS). Both of those computers have printers configured and shared although no real printers are attached to them. Using IIS, ftp servers are used to transfer files. For e-mail, Simple Mail Transfer Protocol (SMTP) is configured via IIS and Microsoft Outlook Express Version 6 is used. Web traffic is accessed via a folder on the server machine. Privileges are set so any user may

access them. The "phantom" printers are added to each of the other computers in the work group.

4.2 Collected Data

The data used consists of text only for the four applications used (e-mail, ftp, http, print). Text taken from the first several chapters of this work is placed into the body of the e-mail messages. For the file transfers (ftp), the text files (.txt) are saved in Microsoft Notepad version 5.1. The DOS ftp put command is the only protocol used in the ftp research. The web pages are created using the text data with links between different pages of text. Print jobs are executed directly from Notepad.

The total combinations of collected traffic are shown in Table 4-1. Note that the e-mail, e-mail combination (EE) is only tested once to show that this case is not significantly different than the single e-mail case and show why other multiple cases are excluded from this research.

Table 4-1 Combinations of Data Collected

EMAIL (E)	EMAIL-PRINT (EP)
FTP (F)	FTP-HTTP (FH)
HTTP (H)	FTP-PRINT (FP)
PRINT (P)	HTTP-PRINT (HP)
EMAIL-EMAIL (EE)	EMAIL-FTP-HTTP (EFH)
EMAIL-FTP (EF)	EMAIL-FTP-PRINT (EFP)
EMAIL-HTTP (EH)	EMAIL-HTTP-PRINT (EHP)
	FTP-HTTP-PRINT (FHP)

Collecting the samples of single applications is straight-forward as instances of each application are sent on an otherwise empty channel (except for 802.11b beacon packets). Several sizes of the files are sent or viewed (in the case of http) in order to analyze unique characteristics of the different applications that would permit proper

classification via machine learning techniques. The file sizes used for each application ranged from 1 kilobyte to 100 kilobytes.

Collecting samples of applications two at a time results in a total of six unique combinations of applications. One experiment was run to show that there is little or no difference between the single application and two instances of the same application with respect to identification via the machine learning techniques. For the three application case, four unique combinations result.

Although signal strength, channel utilization, throughput and other characteristics are collected, only the packet size is used for analysis and classification of the applications. This is because packet size proved to be the strongest indicator of the classification goal (application type). An initial analysis of the distribution of packet sizes amongst the four applications revealed several interesting facts. First, there are definitely packet sizes unique to each application. Of course, this is limited to only the four applications studied. There are generally two classes of unique packet sizes which occur in each type of transmission. The first kind are the "leftover" packets which occur in application transmissions of fairly large (>20Kbyte) size. These occur in different ratios after the 1544 byte maximum transfer units (MTU) of the transmission. For example, the leftover packets for e-mail transmissions are 976 bytes. These unique sizes are probably due to the underlying protocols of the application programs. One problem, however, is that these unique size packets do not occur when transferring smaller files such as the 1 and 10 kilobyte transmissions. The second kind of unique packet sizes found occur in both the small and large transmissions. These packet sizes are not only unique among the four applications, but among the individual transmissions as well.

Furthermore, these sizes occur near the beginning or the end of such transmissions. Some of these unique packet sizes occur in "signature" sequences which allow the identification of the start and end of an application transmission. For example, the packet size 182 is in every e-mail sent regardless of size and the packet size sequence 154-14-82-14-132 is always near the end. The 182 byte packet near the beginning of every e-mail transmission is the SMTP application announcing itself as a mail service. The basis of this research is that unique characteristics exist in encrypted traffic and as such a comparison is made between unencrypted traffic sizes and encrypted traffic sizes for the same files. The results in Table 4-2 show that the unique sizes found in the unencrypted traffic also occur in the encrypted version in the same numbers. For all but the 802.11b

Table 4-2 Unique Sizes for Unencrypted vs. Encrypted E-Mail

Packet Type	Unencrypted	Encrypted
802.11 ACK	14	14
802.11 BEACON	61	61
SMTP	76	84
SMTP	81	89
SMTP QUIT	82	90
SMTP	84	92
SMTP	90	98
RCPT TO	98	106
HELLO	105	113
MAIL FROM	106	114
SENDER OK	116	124
START MAIL INPUT	122	130
SERVICE CLOSING CHANNEL	132	140
MAIL FOR DELIVERY	154	162
MAIL SERVICE	182	190
SMTP DATA	968	976
SMTP DATA	1536	1544

acknowledgement packets (14 bytes) and beacon packets (61 bytes), the unique sizes in the encrypted traffic are 8 bytes larger which results from the encryption process. Although only WEP encryption is examined, the same unique characteristics likely exist

with other encryption techniques as well with different amounts added to the unencrypted packet size due to the particular encryption technique and strength used. Other information could be gained from examining packet size as well. For example, in file transfers (ftp), there is always one unique packet size from a range of sizes. Files named (1K.txt, 10K.txt, 50K.txt, 100K.txt) result in packet sizes of 89, 90, 90, and 91 bytes respectively. This packet contains the file name being transferred. The number of characters in the name of the file being transferred can be determined by simply subtracting 83 bytes from the size of this packet. Using encrypted files, the same result can be obtained by subtracting 122 bytes from this unique packet that appears near the beginning of a file transfer.

This is significant since machine learning techniques require some unique characteristic be present in the data to perform classifications. The decision of attributes to use for training is often much more important in these types of classifications than the design of the system itself.

4.3 Data Preparation

To use ML techniques for the classification of data, it must first be prepared. The program used to model neural networks is Java Neural Network Simulator (JNNS) which is a graphical version of the Stuttgart Neural Network Simulator (SNNS) [Fis98], [Zel94]. Data must be randomized and equally distributed for training to work effectively. After normalizing the data to fall between 0.0 and 1.0, the packet sizes for single cases of e-mail, ftp, http, and print applications are randomly selected as the next sample. As each of these samples is written out, the desired output is added. Since there are four nodes in the output of this neural network, the output corresponding to the

appropriate node is set to 1 while the others are left as 0 as shown in Figure 4-1. For

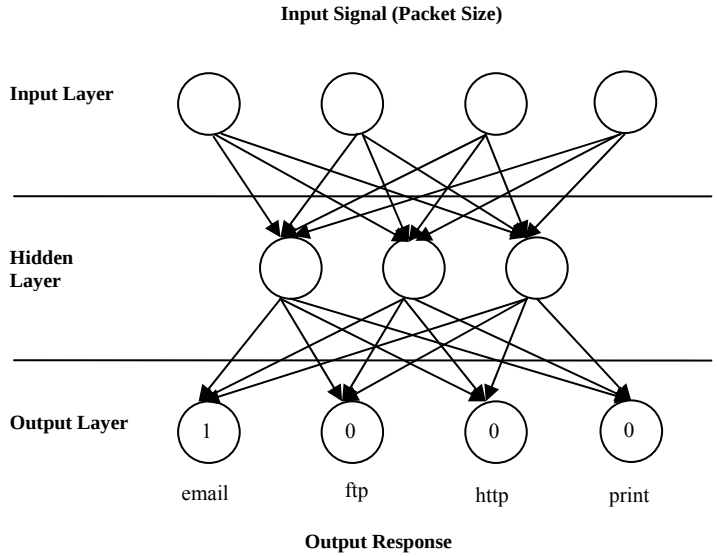

Figure 4-1: Neural Network with Desired Output for Email Sample

example, if an application is e-mail, the desired output is 1000. In the interest of space, only four input nodes are shown in Figure 4-1 though the true number is 11, 31, or 51. The files created for training and testing thus consisted of groups of 11, 31, or 51 packets and the appropriate output response for each based on which application it came from.

The Waikato Environment for Knowledge Analysis (WEKA) [WiF00] is used in the decision tree experiments. To prepare the data for this application, the files needed to be put into an attribute relation file format (ARFF) [WiF00]. In this format, attributes are listed at the beginning of the file preceded by an @ symbol as shown in Figure 4-2. If the data is nominal or categorical, categories must follow the name in curly braces. Otherwise the type of data must follow (i.e., numeric). Of course, the number of packets per sample is 11, 31 or 51 as with the neural networks but Figure 4-2 only shows 4 to conserve space. The final attribute in the relation portion of an ARFF file is the class to

```
% Comments
%
%
@RELATION PacketSize

@ATTRIBUTE packet1  NUMERIC
@ATTRIBUTE packet2  NUMERIC
@ATTRIBUTE packet3  NUMERIC
@ATTRIBUTE packet4  NUMERIC

@ATTRIBUTE application    {email, ftp, http, print}

@DATA
0.009067358,1,0.009067358,1,http
0.073834197,0.009067358,0.066709845,0.009067358,ftp
0.009067358,0.234455959,0.009067358,0.124352332,print
0.009067358,0.054404145,0.009067358,0.123056995,email
```

Figure 4-2: Sample ARFF File

be determined (application). After that, the data for training or testing follows. After each sample of packet size data is the class for that particular sample. Unlike the data for the neural networks, there is no reason to randomize the data for the decision tree induction algorithm.

4.4 Neural Network Configuration

Learning in a neural network can be implemented by any one of a large number of very different algorithms. Pilot studies with the 802.11b packet size data indicated backpropagation would be effective. Other learning algorithms used for the pilot studies included radial basis learning, quick propagation, and batch backpropagation. About 80 percent of all neural network projects use backpropagation because of its ability to classify well in diverse situations [BuC93].

The topology of the network is feed-forward. This means that the connections between the nodes only go in one direction, from the input to the hidden and then the output nodes. The network used consists of three layers. The input layer has 11, 31, or

51 nodes depending on the test being run. The nodes in the middle of the network, known as hidden nodes, allow the network to solve problems other than those that are linearly separable (the only problems solvable with single layer neural networks). The decision of how many nodes to include in the hidden layer is more of an art than a science, but most designers suggest somewhere between 1 and the number of input nodes in the network. Again, pilot studies measured the effects of changing the number of these nodes from one to ten. Five nodes showed the best results. The output layer represents the classes of applications and so there are four of them. The nodes correspond to e-mail, ftp, http, and print applications accordingly.

The transfer function is a combination of the activation function and the output function for a node. The default output function in JNNS is *identity* which means the input signal is propagated if the activation threshold is surpassed. The *logistic* activation function is used since it resulted in the best performance in pilot studies comparing it against the *tanH* function.

Backpropagation uses a generalized delta rule to update weights on connections. The generalized delta rule is

$$\Delta w_{ij} = \beta E f(I) \tag{4.1}$$

where w_{ij} is the weight between node i and j, β is the learning constant (a parameter between 0 and 1 set to a default 0.2), E is the error for the neuron, and $f(I)$ is the input to the neuron. Errors for neurons can be calculated in two ways. Output neuron errors are calculated by subtracting the actual output of the neuron from the desired output minus an error acceptance value. These are set to 1 and 0.1 respectively. In other words, if the output of a particular node is supposed to be 1, and the actual output is 0.89, the error

would be 0.9 – 0.89 or 0.01. This calculation cannot be done on the internal nodes of a network since the number of nodes differs from the number in the output layer. Instead, the error in the output layer neurons is passed back to the middle layer neurons and is weighted by the same connection weights that propagate the input forward. The overall error in the middle neurons is therefore the weighted sum of the errors for each of the output neurons. It is these weights on the connections that contain the "learning" of the system.

4.5 Neural Network Results

Since initialization of a neural network takes random weights, the learning of the network will vary. To determine a confidence interval for the results, four replications of the neural network experiments were done. An important consideration is how to split the data up for training and testing of the network. One common approach is known as cross validation and a four-fold cross validation is performed for the single application case in this research. The data for the single case (5,224 samples) is separated into four equal sections. Four different training procedures are applied such that each of the four data partitions takes its turn as the testing data and the rest are used for the training data. The results for all four tests are averaged for an overall result. Since the scope of this effort is to demonstrate a system can be automated to characterize encrypted data streams, the full combination of applications are not trained. The reason for this is that in this pedagogical example, only four applications are used and the unique combinations of those applications are already 14. In a real system, it would not be practical to train a ML algorithm on every combination of applications. Instead, the network is only trained on the individual applications.

To test the system, a confusion matrix is used. A confusion matrix is simply a table listing the true positives and false positives for a particular classification attempt as shown in Table 4-3. A DOS-based function from the SNNS suite called analyze is used. This function uses a winner takes all (WTA) approach to the confusion matrix. In other words, whichever node for a particular output is the highest is the one the system identifies a particular sample as. In Table 4-3, there are only single applications.

Table 4-3 Sample Confusion Matrix

	EMAIL	**FTP**	**HTTP**	**PRINT**	**% Correct**
EMAIL	900	312	27	27	0.710
FTP	294	870	40	62	0.687
HTTP	497	586	48	135	0.037
PRINT	5	49	7	1205	0.951

However, it is intuitive that if the first row sample actually contained e-mail and ftp data, the e-mail row should be added to the ftp row and divided by the row total. This is how the percentages are calculated in the results that follow.

To determine if the WTA approach might be effective in this case, the node outputs for a trained sample (e-mail and http) are examined as shown in Table 4-4. The

Table 4-4 Node Output for Email-Http Sample

EMAIL		*FTP*	
Mean	0.369	Mean	0.306
Standard Deviation	0.265	Standard Deviation	0.093
Confidence Level (95.0%)	0.006	Confidence Level (95.0%)	0.002
Confidence Interval	(0.363, 0.376)	Confidence Interval	(0.304, 0.309)

HTTP		*PRINT*	
Mean	0.336	Mean	0.050
Standard Deviation	0.171	Standard Deviation	0.161
Confidence Level (95.0%)	0.004	Confidence Level (95.0%)	0.004
Confidence Interval	(0.332, 0.340)	Confidence Interval	(0.046, 0.054)

means for the node outputs show that the e-mail and http nodes are higher on average. Further, none of the confidence intervals overlap which means these ranges are statistically significant. Using this approach, the number of correct classifications over the total number of samples in a file is the percent correct. Since the function is designed to only declare one node the winner, the confusion matrix had to be analyzed in another way for a multiple application case. The true positives of the one case allowed by the system and the "false positives" of those cases that actually are in the sample are added together and then divided by the total number of samples for a true classification rate.

The neural networks trained in this research performed on average 38 percentage points better than what would be expected for random guessing (25%) or at an average of 63% correct classifications. The 95% confidence interval for the classifications is (0.525, 0.733) which does not include 0.25 so the result is statistically significant. This is certainly enough to show that classification is possible, but not enough for any sort of real implementation of an ADS. Table 4-5 is an example of the overall confusion matrix for

Table 4-5 Neural Network Performance for Small Window (11)

	EMAIL	FTP	HTTP	PRINT	% Correct
EMAIL	11	70	941	284	0.008
FTP	26	72	724	484	0.055
HTTP	47	40	884	335	0.676
PRINT	6	5	251	1044	0.799
EF	118	43	2797	465	0.047
EH	123	75	6006	858	0.867
EP	134	70	5085	2242	0.315
FH	571	343	8589	1080	0.843
FP	630	270	8915	2234	0.207
HP	12	22	258	1013	0.973
EFH	711	488	11279	2978	0.807
EFP	894	724	14044	9514	0.442
EHP	150	174	4710	7810	0.986
HFP	595	613	8060	3893	0.954
				OVERALL	**0.657**

one replication of the neural network experiment at the small window size level. It is important to notice that the total numbers of samples for the various combinations are different. This is why obtaining the percentage correct by taking the sum of the number of correctly classified samples over the total number of samples is important. In other words, instead of just taking the average of averages we must take the sum of the numerators and divide this by the sum of the denominators (weighted average). These ratios are different when the bases are different. Although the file sizes used to create the different combinations are the same, the number of packets generated varies from application to application. For example, printing always takes from two to five times the number of packets for the same size file as the other three applications. Furthermore, it is intuitive that there are more packets when transmissions of more than one application are sent. Four replications were run for each of the three different window sizes (11, 31, and 51). Instead of showing the confusion matrix for each replication (available in Appendix A), the weighted average for each neural network is combined in Table 4-6. The sections

Table 4-6 Averages for All Replications of Neural Network Performance

X11		**X31**		**X51**	
EH	0.829	**EH**	0.939	**EH**	0.818
EF	0.346	**EF**	0.309	**EF**	0.853
EP	0.543	**EP**	0.480	**EP**	0.740
FH	0.547	**FH**	0.603	**FH**	0.380
FP	0.267	**FP**	0.203	**FP**	0.450
HP	0.881	**HP**	0.807	**HP**	0.648
FEH	0.792	**FEH**	0.808	**FEH**	0.793
FEP	0.647	**FEP**	0.634	**FEP**	0.895
PEH	0.966	**PEH**	0.960	**PEH**	0.811
PHF	0.784	**PHF**	0.757	**PHF**	0.725
EMAIL	0.251	**EMAIL**	0.395	**EMAIL**	0.701
FTP	0.213	**FTP**	0.220	**FTP**	0.595
HTTP	0.445	**HTTP**	0.622	**HTTP**	0.149
PRINT	0.821	**PRINT**	0.871	**PRINT**	0.900
OVERALL	0.657	**OVERALL**	0.658	**OVERALL**	0.723

of the table are grouped by window size and number of applications.

Using ML techniques for the classification of applications based on the size of packets relies on unique packet sizes present in those applications. It is intuitive that the larger the collected sample size is the more of these unique sized packets there should be and so the ML techniques should perform better. The weighted averages for the systems are 65.7%, 65.8%, and 72.3% respectively. The ML technique seems to perform better with more data available. Applying a 90% confidence level to these averages results in (62.2, 69.3), (64.1, 67.4), and (68.2, 76.4) for the three window sizes respectively. At the 90% confidence level, there is no statistically significant difference between the performances of the 11 and 31 window neural networks.

The mean for the single data seem to add credence to the hypothesis that the systems should perform better as the window size increases as shown in Table 4-7. In

Table 4-7 Averages for Single Application Neural Network Performance

X11		**X31**		**X51**	
EMAIL	0.251	**EMAIL**	0.395	**EMAIL**	0.701
FTP	0.213	**FTP**	0.220	**FTP**	0.595
HTTP	0.445	**HTTP**	0.622	**HTTP**	0.149
PRINT	0.821	**PRINT**	0.871	**PRINT**	0.900

every case except for HTTP the percentage of correct classifications does increase as the window increases from 11 to 51. It is not clear why the HTTP percentage correctly classified decreases at the window level of 51. The variability of the HTTP percentages is high as the values for the four replications ranged from about 1.4% to 51.3%. All three systems performed very well on the print data. This is probably because the number of unique packet sizes in print transmissions is much higher than in the other applications. A graph of inter-application packet sizes is shown in Table 4-8. Note that this is not an

Table 4-8 Packet Sizes by Application

EMAIL	HTTP	PRINT	FTP
14	14	14	14
20	20	20	20
28	61	61	61
61	84	84	84
72	88	172	92
84	92	176	102
90	123	180	104
92	129	184	108
98	147	188	114
112	148	192	141
113	150	200	364
114	156	216	1544
124	160	224	
130	173	236	
190	184	272	
976	188	288	
1544	204	336	
	205	382	
	210	398	
	212	404	
	216	418	
	218	432	
	221	484	
	222	520	
	223	712	
	264	720	
	272	848	
	276	860	
	288	888	
	375	908	
	384	920	
	1316	940	
	1324	968	
	1401	1012	
	1544	1140	
		1156	
		1308	
		1348	
		1396	
		1400	
		1416	
		1452	
		1488	
		1544	

all inclusive list but contains those packets observed throughout this effort. Unique

packet sizes are highlighted for each application.

As for the multiple application data, no clear trends can be observed. In fact, occasionally the 11 window system performs better than the 31 which performs better than the 51. This is the reverse of what is expected since more information should be available as the window size increases. The percentages for the multiple cases could probably be improved by training the network on those samples. However, this would make the system impractical for anything other than a "toy" implementation since the number of training instances would increase exponentially with the number of applications used.

4.6 Decision Tree Configuration

Decision trees incorporate divide and conquer techniques. Data samples to be classified are analyzed based on some measure of the pureness of the nodes produced. The particular tree induction algorithm used herein is an implementation of the C4.5 algorithm in the Waikato Environment for Knowledge Analysis system known as J48 [WiF00].

The idea of purity involves entropy and the information gain achieved by splitting on the different nodes available. However, this method of determining the purity of a split suffers from a strong bias toward tests with many outcomes [Qui93]. In other words, if a particular attribute is made up of unique entries (such as a phone number for different people) the information gain will favor that attribute for the next split even though that attribute may or may not be a good indicator of class since the entropy for a node with only a single case is 0. However, another goal of decision trees is to produce models with the least number of splits while achieving low entropy.

To correct this deficiency, the decision tree implementation in WEKA uses what is known as the *gain ratio* criterion [Qui93]. The gain ratio tests to ensure the information gain is at least as large as the average gain over all tests examined. Split information is

$$split\ info\ \ (\mathrm{X}) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \log_2 \left(\frac{|T_i|}{|T|} \right) \tag{4.2}$$

where n is the number of outcomes possible for a particular attribute, T is the set of training samples, and T_i is the number of cases which take each path if the attribute is selected. The gain ratio then becomes

$$gain\ ratio\ (X) = \frac{gain(X)}{split(X)} \tag{4.3}$$

Revisiting an example from Chapter 2, the information gain for splitting on the outlook data is the entropy of the original unsplit data minus the entropy after splitting on the outlook variable so that we have 0.918 – 0.675 = 0.243. The data for splitting on the outlook variable is shown in Table 4-9. To calculate the split information using this data

Table 4-9 Outcome Variable for Outlook

Sunny	Overcast	Rainy
Yes	**Yes**	**Yes**
Yes	**Yes**	**Yes**
No	**Yes**	**Yes**
No		**No**
No		

gives

$$-(5/12) * \log_2(5/12) - (3/12) * \log_2(3/12) - (4/12) * \log_2(4/12) = 1.554 \tag{4.4}$$

Applying (4.3) to get the gain ratio gives 0.243/1.554 = 0.156. This modification has the effect of increasing the denominator and therefore reducing the overall gain ratio when

the number of splits increases. This ratio is still a "bigger is better" metric so splits that tend to separate the data into single case nodes should be avoided.

Since the data used is continuous, another point must be made about the way decision trees deal with this type of data. So far, the examples have discussed only discrete data. The difference with continuous, numerical data is that there are as many possible thresholds in the data as there are unique numbers in the range of that data [WiF00]. The training instances *T* are sorted on the values *v* of the attribute being considered. If there are *m* such values, then there are a total of *m*-1 possible splits on the attribute and each one is considered as described above for the discrete data. This work is necessary to find the right split for every attribute in the data. The analysis of packet sizes from e-mail, ftp, http and print showed packet sizes ranged from 14 for an 802.11b acknowledgement to 1544 for a data packet with approximately 72 unique sizes in between. Since the window size used for samples is 11, 31, and 51, the numbers of tests needed are 869, 2449, and 4029 respectively. Each packet in the collection window is treated as a separate attribute. Since the values are arranged in order, the tests can be performed in one pass while the class distributions are updated on the fly and the training of the network usually takes less than 30 seconds.

Pruning reduces a decision tree's size. This makes the model more comprehensible, and may correct any over-fitting of the model which boosts its generalization ability. There are two methods of pruning, prepruning and postpruning. The decision tree algorithm employed in WEKA used postpruning to reduce the size of the tree model. Trees are usually "pruned" by removing one or more subtrees and replacing them with leaves. The test used to decide whether or not a branch should be

pruned should be “taken with a large grain of salt” and “does violence to statistical notions of sampling and confidence limits” [Qui93]. However, the test does frequently produce good results. In the author’s own words:

> When *N* training cases are covered by a leaf, *E* of them incorrectly, the resubstitution error rate for this leaf is *E*/*N*. However, we can regard this somewhat naively as observing *E* “events” in *N* trials. If this set of *N* training cases could be regarded as a sample (which, of course, it is not), we could ask what this result tells us about the probability of an event (error) over the entire population of cases covered by this leaf. The probability of error cannot be determined exactly, but has itself a (posterior) probability distribution that is usually summarized by a pair of confidence limits. For a given confidence level (CF), the upper limit on this probability can be found from the confidence limits for the binomial distribution. Then, C4.5 simply equates the predicted error rate at a leaf with this upper limit, on the argument that the tree has been constructed to minimize the observed error rate.

More detail on the C4.5 decision tree algorithm can be found in [Qui93].

The pilot studies for the decision tree experiments led to using a boosting algorithm known as AdaBoost.M1 [Qui96]. Boosting is a technique applicable to many ML algorithms. It was originally intended for taking relatively weak (slightly better than random guessing) classifiers and turning them into strong ones. However, the technique has been applied to decision trees in general, and the C4.5 (and so J48) algorithm in particular with good results [Qui96]. In general, boosting keeps all training instances per training cycle, but increases the weight or how important the sample is each time it is misclassified. In this way, a system is forced to “work harder” to classify those samples that are more difficult.

Many of the different settings in WEKA simply relate to the output format of the data, but some have an impact on the performance of the algorithms and so all the settings used for this research are included in Appendix B.

4.6 Decision Tree Results

Unlike neural networks, the decision tree induction algorithm is deterministic and only required one replication at each window setting (11, 31, and 51). Splitting the data up into portions for cross-validation purposes is still important though and a four fold split was performed just as with the neural network data.

Although decision trees are more comprehensible in nature than neural networks, the size of the trees for large attribute cases makes them difficult to visualize. The assumption that part of the reasoning behind the split decisions has to do with the unique packet sizes present in the different application transmissions is true when looking at a small portion of an induced decision tree as in Figure 4-3. Since the data has been normalized by dividing by the MTU (1544), we are actually looking for 976/1544 (the "leftover" packets for e-mail) or approximately 0.6321. In the small portion of decision tree shown, the value for this unique packet size is highlighted and the rule for this particular value shows that about 65 samples of e-mail were correctly classified based on

```
|   |   |   |   |   packet11 <= 0.494819: ftp (2.11)
|   |   |   |   |   packet11 > 0.494819: http (7.39/1.58)
|   |   |   |   packet10 > 0.022021: http (5.28)
|   |   packet4 > 0.012953
|   |   |   packet1 <= 0.906736: http (4.22/0.53)
|   |   |   packet1 > 0.906736
|   |   |   |   packet3 <= 0.039508
|   |   |   |   |   packet6 <= 0.963731: http (2.11)
|   |   |   |   |   packet6 > 0.963731
|   |   |   |   |   |   packet3 <= 0.012953
|   |   |   |   |   |   |   packet8 <= 0.906736: http (8.97/1.06)
|   |   |   |   |   |   |   packet8 > 0.906736: ftp (6.33/1.58)
|   |   |   |   |   |   packet3 > 0.012953: ftp (2.64/0.53)
|   |   |   |   packet3 > 0.039508: ftp (6.33/1.06)
packet3 > 0.595855
|   packet3 <= 0.632124: email (64.99/4.22)
```

Figure 4-3 Portion of J48 Induced Decision Tree

it while 4.22 were incorrect. Further analysis of the tree revealed that the unique packet sizes are exclusively used for the identification of e-mail.

The same confusion matrices were used to determine the percent of correct classifications for the decision tree models as with the neural networks. The decision trees performed much better and took less time to train than the neural networks. The classification levels for the single replications of the three window experiments are shown in Table 4-10. The variability among the three different window sizes is much less than that for the neural networks ranging from 86.4% to 88%. An interesting similarity in the performance of the two algorithms is that print jobs are the most correctly classified ranging from 97.6% to 99.8% success. Since there are more unique packet sizes in print transmissions than any of the others, this result is expected. Looking

Table 4-10 Classification Percentages for All Decision Tree Experiments

X11		**X31**		**X51**	
EH	0.832	**EH**	0.663	**EH**	0.930
EF	0.879	**EF**	0.934	**EF**	0.922
EP	0.769	**EP**	0.866	**EP**	0.873
FH	0.765	**FH**	0.797	**FH**	0.681
FP	0.687	**FP**	0.723	**FP**	0.605
HP	0.924	**HP**	0.941	**HP**	0.948
FEH	0.995	**FEH**	0.995	**FEH**	0.998
FEP	0.838	**FEP**	0.879	**FEP**	0.882
PEH	0.902	**PEH**	0.915	**PEH**	0.925
PHF	0.822	**PHF**	0.734	**PHF**	0.677
EMAIL	0.828	**EMAIL**	0.908	**EMAIL**	0.932
FTP	0.900	**FTP**	0.984	**FTP**	0.987
HTTP	0.984	**HTTP**	0.986	**HTTP**	0.976
PRINT	0.976	**PRINT**	0.994	**PRINT**	0.997
OVERALL	0.843	**OVERALL**	0.859	**OVERALL**	0.839

at the weighted mean for the systems the performance is 84.3%, 85.9% and 83.9% respectively. Since the decision tree algorithm is deterministic, no confidence intervals are required. The first two systems perform as expected (in relation to each other) in that

the second performs better. However, the 51 window system performs worse than both the 11 and 31 window systems.

A similarity in the performance of the different algorithms is apparent in the single data classifications. As expected, the percentage of correct classifications rises with the number of packets used in the training data except for in the HTTP case. The difference is not as dramatic as in the neural network system (44.5%, 62.2%, 14.9%), but the trend is still the same. Unfortunately, yet another similarity between the performance of the two algorithms is that there are no clear relationships among the multiple data cases.

4.7 Comparative Analysis

On 36 out of 42 (85.7%) experiments, the decision tree approach classifies the samples at a much better rate of up to 85% in some cases. In 26 of those experiments, decision trees are statistically better with the classification rate falling outside the confidence interval of the neural network performance. This means that in ten of the experiments, the decision trees perform better than the average neural network performance but the value falls within the 90% confidence interval of the NN response. On the six experiments where the neural network performs better, the maximum difference is 6% and only three of these cases are statistically significant.

The decision tree algorithm is a better candidate for an ADS implementation. Decision trees classify at a higher percentage, train faster, and are more comprehendible than neural networks. However, it is clear that both ML techniques have successfully learned their classification tasks when compared to the baseline of 25% for correct random guessing.

This does not imply that the methods outlined in this research are ready to implement in an ADS. There are still problems to be overcome such as how to determine which of the different cases (single, double, triple) are occurring at any given time on the network. With the decision trees, it is a fairly simple process to determine if only one sample is accessing the medium since the other three application levels should account for less than 10% of the overall classifications. These difficulties occur due to the differences in training a model on a "flat" file of collected transmissions versus attempting to classify samples "on the fly". For example, the decision trees are forced to make a determination among the four applications for every sample presented to them. Out of a large number of samples of combined traffic, this method is successful since the algorithm votes fairly equally among the correct choices. This presents a problem for the real-time detection of multiple applications via decision trees. For example, if e-mail and http traffic are combined on the channel, the decision tree will give a classification of one or the other. It may be the case that the oscillation of the decision tree classification could act as the input to another layer of programming or ML which could make the ultimate determination about which applications and how many are accessing the channel. Perhaps the nature of the neural network output nodes could be useful in multiple application decisions.

4.8 Analysis of Variance

Statistical methods are used throughout the analysis of the classification results obtained and they reveal that there is little difference in system performance among the different window sizes for each ML technique. An analysis of variance between the two systems is not necessary since the outcome would be obvious: the choice of algorithm

accounts for nearly all of the variance in performance.

4.9 Summary

This chapter presents and analyzes the results of this research. Two kinds of unique packet sizes occur within transmissions of the four applications used, "leftover" packets and "signature" ones. Decision trees and neural networks use these unique packets to achieve correct classification rates that are statistically significant ranging from 65.8% to 85.9%. The single application classifications perform as expected in regard to the increase in percent correct corresponding to an increase in window size. The multiple application classifications achieve statistically significant results. However, no other clear trends can be observed. Statistically, the choice of window size changes little of the performance of the algorithms. An ANOVA is not necessary since confidence levels are used for each result and the major source of variance clearly results from the choice of algorithm.

.

V. Conclusions and Recommendations

This chapter summarizes the problem, research contributions, limitations and recommendations for future research in this area.

5.1 Problem Summary

As wireless networks become an increasingly common part of our nation's infrastructure, we need to evaluate the vulnerabilities of this technology. Wireless networks are unique in that the channel is not physically secure and typically has lower data rates and higher error rates when compared to a wired network. There is a major push toward tougher encryption for wireless networks embodied in such documents as 802.11i. As encryption grows stronger, it becomes less and less likely that packets can be decrypted in a timely manner. For this reason other techniques for analyzing wireless network traffic need to be discovered. Such study is of value from both the offensive and defensive standpoint. In other words, what might our adversaries be learning from our wireless networks? Conversely, what might we learn from the networks of our adversaries? Unique characteristics have been shown to exist within such wireless packet attributes as the packet size and these are used to train ML techniques for the automatic detection of applications accessing the channel.

5.2 Findings

This research shows that ML techniques can be applied successfully to the problem of inferring important information from 802.11b encrypted transmissions. The overall success of the neural networks and decision trees ranges from 65% to 86% correct classification of the applications accessing the channel. For the single application case, the success varies a great deal in the neural networks ranging from an average low of

approximately 15% to a high of 90%. The success of the decision tree algorithm varied much less and ranged from a low of approximately 83% to a high of 99.8% for the single application case. In both ML techniques, the print jobs are the most successfully classified.

To understand how the ML techniques determined their rules, a thorough analysis of the packet sizes within the four applications studied is undertaken. This analysis reveals a number of unique inter-application packet sizes and some "signature" intra-application packet sizes. Since these signature packets appear in every size of the applications tested they can be used to identify even small (1-30 KByte) transmissions. Further, these signature packet sizes tend to mark the beginning and end of transmissions. For example, the packet size of 108 bytes is unique among all four applications studied (e-mail, ftp, http, print) and occurred near the end of every ftp. By analyzing unencrypted versions of the same e-mail transmission, it is revealed that this packet is an "end of ftp service" packet. Since the hypothesis of this research is that encryption can not hide all the information in packets, an analysis between the unencrypted and WEP encrypted transmissions of all four applications is also performed. The results show that the same unique packet size characteristics do indeed exist in the encrypted versions of 802.11b transmissions, and that the encrypted versions are simply 8 bytes larger which is due to the encryption overhead. In other words, the 100 byte end of ftp service packet is 108 bytes in the encrypted version. The ML techniques for this research are trained using the encrypted version of all traffic.

5.3 Limitations

Since this research is carried out on an ad-hoc 802.11b network, the results

might not apply to an infrastructure where the wireless computers communicate via an access point to a wired network since the packet sizes could be changed once they are processed by the access point. However, it is likely that unique packet sizes occur in most networks where the sizes of the packets vary.

Another limitation of this research is that only four applications are studied. Further, only text data is used for the applications. The work likely generalizes to more applications, but the number would have to be bounded since the number of unique packet sizes obviously is. In other words, it is possible that several applications will have similar packet sizes and so confuse any ML technique trying to classify them.

A third limitation is that the data transformed and classified by the ML techniques is collected and resides in "flat" files. Identifying the number of applications accessing the medium at any one time is inherently problematic for the ML techniques. This is true even when viewing a transmission with a-priori knowledge and with the benefit of knowing the applications that make up each file of samples. This problem is worsened when transformed to the real-time application detection scenario.

5.4 Recommendations for Future Research

One area of future research should include the analysis of more applications to see if there are unique packets available for classification. Along the same lines, other types of data should be used in order to ensure that the classification process can handle these. Perhaps even more detailed classifications are possible with the inclusion of other types of data. For example, we may learn that a print job is accessing the channel and that the job includes graphics.

Future research could also include the eventual production of an ADS. One

possibility for creating such a system is to use the unique intra-application packets with some sort of finite state automatons (FSA). These FSAs would be responsible for identifying how many applications were accessing the channel. In a pristine environment (where only one transmission is occurring) these FSAs could also track the approximate sizes of application transmissions. By combining FSAs with decision trees (or another appropriate ML technique) the accuracy of the system would likely be improved.

Another interesting area for research is the combination of information gained via the headers with the data gleaned from the packet sizes. For example, if the IP address of a computer is identified as commonly being associated with http traffic, that computer can possibly be identified as a web host machine.

This research relies on unique packet sizes in order to infer information from transmissions. Other characteristics of packet transmissions should be studied to determine if they can provide as much information. One such characteristic, signal strength, could be used to help determine the number of applications accessing the channel. For example, if the ML technique identified e-mail as the application, but there were two signal strengths involved then there could be two e-mails on the channel.

Another possibility for future research could involve improved methods for defeating this type of traffic analysis. In other words, methods need to be developed that are less costly in terms of performance and overhead so that they will more likely be used in the field.

Appendix A

Neural Network Results:Window=11, Replication 1					
	EMAIL	**FTP**	**HTTP**	**PRINT**	**Overall %**
EH	123	75	6006	858	0.867
EF	118	43	2797	465	0.047
EP	134	70	5085	2242	0.315
FH	571	343	8589	1080	0.843
FP	630	270	8915	2234	0.207
HP	12	22	258	1013	0.973
FEH	711	488	11279	2978	0.807
FEP	894	724	14044	9514	0.442
PEH	150	174	4710	7810	0.986
PHF	595	613	8060	3893	0.954
				Total	**0.646**

Single Application Confusion Matrix

	EMAIL	**FTP**	**HTTP**	**PRINT**	
EMAIL	11	70	941	284	0.008
FTP	26	72	724	484	0.055
HTTP	47	40	884	335	0.676
PRINT	6	5	251	1044	0.799
				Total	**0.384**
				Overall	**0.634**

Neural Network Results:Window=11, Replication 2					
	EMAIL	**FTP**	**HTTP**	**PRINT**	**Overall %**
EH	658	206	5343	855	0.849
EF	275	37	2470	641	0.091
EP	486	173	4593	2280	0.367
FH	756	102	7916	1809	0.757
FP	676	91	8309	2973	0.254
HP	17	95	217	976	0.914
FEH	911	363	10432	3750	0.757
FEP	714	642	13174	10646	0.476
PEH	278	333	4335	7898	0.974
PHF	464	460	7371	4866	0.964
				Total	**0.647**

Single Application Confusion Matrix

	EMAIL	**FTP**	**HTTP**	**PRINT**	
EMAIL	111	198	843	154	0.084
FTP	77	310	631	288	0.237
HTTP	54	50	854	348	0.653
PRINT	8	22	186	1090	0.834
				Total	**0.452**
				Overall	**0.638**

Neural Network Results:Window=11, Replication 3					
	EMAIL	**FTP**	**HTTP**	**PRINT**	**Overall %**
EH	4151	937	972	1002	0.725
EF	1879	492	465	587	0.692
EP	3410	833	904	2385	0.769
FH	5815	2167	1125	1476	0.311
FP	5872	2235	1331	2610	0.402
HP	137	120	65	983	0.803
FEH	7382	2918	1723	3433	0.777
FEP	8952	3759	2413	10050	0.904
PEH	2996	931	802	8115	0.927
PHF	5025	2640	1282	4213	0.618
				Total	**0.711**

Single Application Confusion Matrix

	EMAIL	**FTP**	**HTTP**	**PRINT**	
EMAIL	617	317	186	186	0.472
FTP	492	479	69	266	0.366
HTTP	585	177	201	343	0.153
PRINT	122	42	36	1106	0.846
				Total	**0.459**
				Overall	**0.700**

Neural Network Results:Window=11, Replication 4					
	EMAIL	**FTP**	**HTTP**	**PRINT**	**Overall %**
EH	3875	190	2302	694	0.874
EF	1860	37	1049	477	0.554
EP	3255	148	1946	2183	0.721
FH	6465	120	2827	1171	0.278
FP	6466	84	3081	2418	0.207
HP	140	74	136	955	0.836
FEH	7758	453	4598	2647	0.828
FEP	9425	744	5828	9179	0.768
PEH	3081	276	2033	7452	0.978
PHF	5260	530	3733	3638	0.600
				Total	**0.669**

Single Application Confusion Matrix

	EMAIL	**FTP**	**HTTP**	**PRINT**	
EMAIL	575	175	399	157	0.440
FTP	525	252	260	269	0.192
HTTP	600	52	386	268	0.295
PRINT	112	8	136	1050	0.803
				Total	**0.433**
				Overall	**0.658**

Neural Network Results:Window=31, Replication 1					
	EMAIL	**FTP**	**HTTP**	**PRINT**	**Overall %**
EH	3318	111	3254	349	0.934
EF	1422	7	1571	410	0.419
EP	2691	217	2773	1397	0.577
FH	4513	20	5033	996	0.478
FP	4349	48	5301	2331	0.197
HP	214	231	116	724	0.653
FEH	11160	402	898	2976	0.807
FEP	11750	1095	1871	8283	0.918
PEH	3435	815	981	7593	0.936
PHF	7220	556	1133	4240	0.450
				Total	**0.679**

Single Application Confusion Matrix

	EMAIL	**FTP**	**HTTP**	**PRINT**	
EMAIL	863	276	96	51	0.671
FTP	378	354	428	126	0.275
HTTP	339	38	706	203	0.548
PRINT	56	28	63	1139	0.885
				Total	**0.595**
				Overall	**0.675**

Neural Network Results:Window=31, Replication 2					
	EMAIL	**FTP**	**HTTP**	**PRINT**	**Overall %**
EH	684	55	5983	310	0.948
EF	224	12	2882	292	0.069
EP	463	100	4794	1721	0.308
FH	431	21	9043	1067	0.858
FP	383	32	9142	2472	0.208
HP	55	58	128	1044	0.912
FEH	704	73	11563	3096	0.799
FEP	629	189	12985	9196	0.435
PEH	313	215	3753	8543	0.983
PHF	369	105	7909	4766	0.971
				Total	**0.657**

Single Application Confusion Matrix

	EMAIL	**FTP**	**HTTP**	**PRINT**	
EMAIL	206	101	907	72	0.160
FTP	88	208	827	163	0.161
HTTP	16	30	1078	162	0.838
PRINT	1	4	124	1157	0.899
				Total	**0.514**
				Overall	**0.650**

Neural Network Results:Window=31, Replication 3					
	EMAIL	**FTP**	**HTTP**	**PRINT**	**Overall %**
EH	141	132	6570	189	0.954
EF	7	24	3121	258	0.009
EP	94	172	5244	1568	0.234
FH	20	53	9579	910	0.911
FP	21	60	9819	2129	0.181
HP	74	73	140	998	0.885
FEH	100	150	12472	2714	0.824
FEP	164	411	13996	8428	0.391
PEH	119	277	4020	8408	0.978
PHF	76	226	8559	4288	0.994
				Total	**0.649**

Single Application Confusion Matrix

	EMAIL	**FTP**	**HTTP**	**PRINT**	
EMAIL	156	78	1017	35	0.121
FTP	248	153	824	61	0.118
HTTP	22	28	1098	138	0.853
PRINT	7	9	124	1146	0.891
				Total	**0.496**
				Overall	**0.642**

Neural Network Results:Window=31, Replication 4					
	EMAIL	**FTP**	**HTTP**	**PRINT**	**Overall %**
EH	5614	295	857	266	0.920
EF	2489	43	575	303	0.742
EP	4115	384	1010	1569	0.803
FH	7808	181	1586	986	0.167
FP	6886	239	2434	2470	0.225
HP	87	198	80	920	0.778
FEH	8750	440	3184	3060	0.801
FEP	8431	1096	4750	8722	0.793
PEH	2339	731	1688	8066	0.942
PHF	5112	592	2967	4478	0.611
				Total	**0.670**

Single Application Confusion Matrix

	EMAIL	**FTP**	**HTTP**	**PRINT**	
EMAIL	811	244	192	39	0.630
FTP	594	420	201	71	0.326
HTTP	775	64	321	126	0.249
PRINT	96	68	76	1046	0.813
				Total	**0.505**
				Overall	**0.662**

Neural Network Results:Window=51, Replication 1					
	EMAIL	**FTP**	**HTTP**	**PRINT**	**Overall %**
EH	4148	611	1961	307	0.869
EF	1813	396	986	195	0.651
EP	2785	1332	1772	1597	0.585
FH	4856	1792	3227	590	0.479
FP	4178	1900	3414	2516	0.367
HP	52	650	50	513	0.445
FEH	5707	2451	3950	3308	0.785
FEP	4314	4584	4895	10141	0.795
PEH	1401	3560	1433	6410	0.721
PHF	2705	2636	3198	4574	0.793
				Total	**0.687**

EFHP	**EMAIL**	**FTP**	**HTTP**	**PRINT**	
EMAIL	779	277	137	73	0.615
FTP	333	641	221	71	0.506
HTTP	268	248	649	101	0.512
PRINT	4	203	32	1027	0.811
				Total	**0.611**
				Overall	**0.684**

Neural Network Results:Window=51, Replication 2					
	EMAIL	**FTP**	**HTTP**	**PRINT**	**Overall %**
EH	5236	1364	124	304	0.762
EF	2221	862	104	203	0.909
EP	3635	1404	267	2180	0.776
FH	5314	4247	246	658	0.429
FP	4882	3625	693	2809	0.535
HP	67	208	6	984	0.782
FEH	6630	4358	1017	3411	0.778
FEP	5247	5362	1821	11504	0.923
PEH	1592	1640	417	9155	0.871
PHF	3268	3828	1143	4876	0.750
				Total	**0.760**

EFHP	**EMAIL**	**FTP**	**HTTP**	**PRINT**	
EMAIL	900	312	27	27	0.710
FTP	294	870	40	62	0.687
HTTP	497	586	48	135	0.037
PRINT	5	49	7	1205	0.951
				Total	**0.596**
				Overall	**0.753**

Neural Network Results:Window=51, Replication 3					
	EMAIL	**FTP**	**HTTP**	**PRINT**	**Overall %**
EH	5391	1268	73	296	0.777
EF	2333	811	65	181	0.927
EP	3809	1373	232	2072	0.785
FH	5618	4111	138	598	0.406
FP	5334	3592	491	2592	0.514
HP	70	232	2	961	0.761
FEH	7728	4338	789	3061	0.807
FEP	6148	5498	1497	10791	0.937
PEH	1817	1699	343	8945	0.867
PHF	3837	3915	888	4475	0.707
				Total	**0.759**

	EMAIL	**FTP**	**HTTP**	**PRINT**	
EMAIL	915	302	10	39	0.722
FTP	323	857	12	74	0.676
HTTP	553	560	18	135	0.014
PRINT	7	56	19	1184	0.935
				Total	**0.587**
				Overall	**0.751**

Neural Network Results:Window=51, Replication 4					
	EMAIL	**FTP**	**HTTP**	**PRINT**	**Overall %**
EH	6033	698	52	245	0.865
EF	2723	411	84	171	0.924
EP	4278	1169	236	1803	0.812
FH	7694	2003	174	594	0.208
FP	6805	2016	613	2575	0.382
HP	77	423	5	760	0.604
FEH	8687	2704	980	3044	0.802
FEP	7115	4628	1794	10397	0.925
PEH	2166	2766	414	7458	0.783
PHF	4600	2945	1075	4495	0.649
				Total	**0.709**

	EMAIL	**FTP**	**HTTP**	**PRINT**	
EMAIL	959	293	1	13	0.757
FTP	600	647	13	6	0.511
HTTP	770	396	42	58	0.033
PRINT	3	91	27	1145	0.904
				Total	**0.551**
				Overall	**0.702**

Decision Tree Results:Window=11					
	EMAIL	**FTP**	**HTTP**	**PRINT**	**Overall %**
EH	5234	1153	648	32	0.832
EF	1781	1233	402	12	0.879
EP	3801	1060	677	1993	0.769
FH	2481	7270	826	5	0.765
FP	2513	6419	1257	1859	0.687
HP	70	29	156	1049	0.924
FEH	6081	7606	1697	62	0.995
FEP	5420	8293	4059	7401	0.838
PEH	2670	1255	1207	7711	0.902
PHF	2339	6686	2017	2126	0.822
				Total	**0.839**

EFHP	**EMAIL**	**FTP**	**HTTP**	**PRINT**	
EMAIL	1081	170	46	8	0.828
FTP	119	1175	9	2	0.900
HTTP	12	6	1285	2	0.984
PRINT	12	10	9	1274	0.976
				Total	**0.922**
				Overall	**0.843**

Decision Tree Results:Window=31					
	EMAIL	**FTP**	**HTTP**	**PRINT**	**Overall %**
EH	2019	1150	249	2	0.663
EF	6069	527	429	33	0.934
EP	4369	616	389	2148	0.866
FH	2140	7664	769	0	0.797
FP	2434	6824	894	1887	0.723
HP	54	22	159	1060	0.941
FEH	6081	7606	1697	62	0.995
FEP	6027	8540	3045	7555	0.879
PEH	3089	1081	958	7706	0.915
PHF	3488	6654	1762	1244	0.734
				Total	**0.854**

EFHP	**EMAIL**	**FTP**	**HTTP**	**PRINT**	
EMAIL	1178	99	16	3	0.908
FTP	12	1276	8	0	0.984
HTTP	10	8	1278	0	0.986
PRINT	3	3	1	1299	0.994
				Total	**0.968**
				Overall	**0.859**

Decision Tree Results:Window=51					
	EMAIL	**FTP**	**HTTP**	**PRINT**	**Overall %**
EH	6139	458	400	30	0.930
EF	2151	977	261	0	0.922
EP	4458	548	402	2083	0.873
FH	3331	6275	857	1	0.681
FP	3652	5633	1088	1635	0.605
HP	37	28	130	1069	0.948
FEH	7515	6186	1697	17	0.998
FEP	7616	7373	2966	7181	0.882
PEH	3285	957	974	7587	0.925
PHF	4232	6110	1696	1090	0.677
				Total	**0.833**

EFHP	**EMAIL**	**FTP**	**HTTP**	**PRINT**	
EMAIL	1179	68	17	1	0.932
FTP	39	1224	2	0	0.967
HTTP	16	14	1235	0	0.976
PRINT	3	0	0	1262	0.997
				Total	**0.968**
				Overall	**0.839**

Appendix B

Settings for WEKA Decision Tree Algorithm J48 and AdaBoost.M1

AdaBoostM1	
Classifier	J48
debug	FALSE
maxIterations	10
useResampling	FALSE
weightThreshold	100
J48	
binarySplits	FALSE
confidenceFactor	0.25
minNumObj	2
numFolds	3
reducedErrorPruning	FALSE
saveInstandeData	FALSE
subtreeRaising	TRUE
unpruned	FALSE
useLaplace	FALSE

References

[P802.11] "IEEE Standard for Wireless LAN Medium Access Control and Physical Layer Specification," P802.11*, Institute of Electrical and Electronics Engineers*, November 1997.

[And98] Andren C., "IEEE 802.11 Wireless LAN: can we use it for multimedia," *IEEE Multimedia*, pp. 84-89, 1998.

[And01] Anderson Ross. *Security Engineering: A Guide to Building Dependable Distributed Systems*, John Wiley & Sons, Inc. New York, NY, 2001.

[Bel97] Bellovin, S., "Probable Plaintext Cryptanalysis of the IP Security Protocols," *Proceedings of the Symposium on Network and Distributed System Security,* San Diego, CA, pp. 155-160, 1997.

[BGS00] Back, Adam, Goldberg, Ian, Shostack, Adam "Freedom 2.0 Security Issues and Analysis," *Zero-Knowledge Systems, Inc.*, November, 2000.

[Bil02] Bilmes, Jeff, "What HMMs Can Do," *University of Washington School of Engineering Technical Report-UWEE*, January 2002.

[BuC93] Butler, C., Caudill, M. *Understanding Neural Networks*. Cambridge, MA, The MIT Press, 1993.

[ChA99] Cheng, H., Avnur, R., "Traffic Analysis of SSL Encrypted Web Browsing," *Berkeley School of Computer Science*, Unpublished Report, 2000.

[CrS97] Craven, S., Shavlik, J., "Using Neural Networks for Data Mining," *Future Generation Computer Systems*, 13(2-3):211-229, 1997.

[Edd00] Eddy, Sean, "Profile Hidden Markov Models," *Bioinformatics* 14: 755-763, 1998.

[Ent02] Enterasys Networks, *802.11 Wireless Networking Guide*, Rochester, NH, 2002.

[FHB98] Fischer, Igor, Hennecke, Fabian, Bannes, Christian, Zell, Andreas. *JNNS, Java Neural Network Simulator, User Manual*. University of Stuttgart, 1998.

[FKK96] Fasbender, A., Kesdogan, D., Kubitz, O., "Variable and Scalable Security: Protection of Location Information in Mobile IP," *IEEE 46th Vehicular Technology Conference*, Atlanta, GA, March 1996.

[Gei02] Geier, Jim. *Wireless Lans*. Indianapolis, Indiana, Sams, 2002.

[GFX01] Guan Yong et al., "Net Camo: Camouflaging Network Traffic for QoS-Guaranteed Mission Critical Applications," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, Vol. 31, No. 4, July 2001.

[GLX99] Guan Yong et al., "Preventing Traffic Analysis for Real-Time Communication Networks," *IEEE Military Communications*, 1:744-750, November 1999.

[Gri02] Grimm, Brian, "Wi-Fi Protected Access," White Paper, *Wi-Fi Alliance*, 2002.

[Gun00] Gunsch, Trace, "Emerging Technology Future View Convergence," U.S. Army Information Systems Engineering Command, Technology Integration Center Report, March 2000.

[HTF02] Hastie, T., Tibshirani, R., Friedman, Jerome. *The Elements of Statistical Learning*. New York, NY, Springer-Verlag, 2001.

[JDM00] Jain, A., Duin, R., Mao, J., "Statistical Pattern Recognition: A Review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 1, pp. 4-37, January, 2000.

[JVZ01] Jiang, Shu, Vaidya, Nitin, Zhao, Wei, "A Dynamic Mix Method for Wireless Ad Hoc Networks," *MILCOM Proceedings*, McLean, VA, 2:873-877, October, 2001.

[KaV94] Karr, C. L., Vann, P. A., "Inferring Difficult to Measure Parameters Using Neural Networks and Genetic Algorithms," *Artificial Neural Networks in Engineering Conference*, St. Louis, MO, 4:313-319. 1994

[KeA98] Kent, S., Atkinson, R., "RFC 2406 IP Encapsulating Security Payload," *The Internet Engineering Task Force*, RFC-2406, November 1998.

[Lou95] Lou Hui-Ling, "Implementing the Viterbi Algorithm," *IEEE Signal Processing Magazine*, 1995.

[Mar01] Marchette, David. *Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint*, New York, NY, Springer-Verlag, 2001.

[Mit97] Mitchell, T. M. *Machine Learning*, New York, NY, McGraw Hill, 1997.

[MTV97] Mannila, Heikki, Toivonen, Hannu, Verkamo, Inkeri, "Discovery of Frequent Episodes in Event Sequences," *Data Mining and Knowledge Discovery*, Vol. 1, No. 3, pp. 259-289, 1997.

[Mul02] Mullins, Justin, "Making Unbreakable Code," *IEEE Spectrum*, pp. 40-45, May 2002.

[Qui86] Quinlan, R., "Induction of Decision Trees," *Machine Learning*, 1, pp. 81-106, 1986.

[Qui93] Quinlan, R. *Programs for Machine Learning: C4.5*, San Mateo, CA, Kaufmann Publishers, 1993.

[Qui96] Quinlan, R., "Bagging, Boosting, and C4.5," *Proceedings of the National Conference on Artificial Intelligence* (AAAI-96), Portland, OR, pp. 725-730, 1996.

[Rab89] Rabiner, Lawrence, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, February 1989.

[RSG98] Reed, M., Syverson, P., Goldschlag, D., "Anonymous Connections and Onion Routing," *IEEE Journal on Selected Areas in Communications*, Vol. 16, No. 4, pp. 482-494, May 1998.

[SWT01] Song, Dawn, Wagner, David, Tian, Xuqing, "Timing Analysis of Keystrokes and Timing Attacks on SSH," *Proceedings of the 10th USENIX Security Symposium*, Washington D.C., 2001.

[WiF00] Witten, I., Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, San Francisco, CA, Morgan Kaufmann, 2000.

[WoV91] Newman-Wolfe, R. E., Venkatraman, Balaji, "High Level Prevention of Traffic Analysis," *Seventh Annual Computer Security and Applications Conference*, San Antonio, TX, December 1991.

[WoV92] Newman-Wolfe, R. E., Venkatraman, Balaji, "Performance Analysis of a Method for High Level Prevention of Traffic Analysis," *Eighth Annual Computer Security and Applications Conference*, San Antonio, TX, December 1992.

[WoV93] Newman-Wolfe, R. E., Venkatraman, Balaji, "Transmission Schedules to Prevent Traffic Analysis," *Ninth Annual Computer Security and Applications Conference,* San Antonio, TX, December 1993.

[Zel94] Zell, A. et al. *SNNS, Stuttgart Neural Network Simulator, User Manual*. University of Stuttgart, 1994.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
23-03-2004	**Master's Thesis**	Apr 2003 – Mar 2004

4. TITLE AND SUBTITLE

MACHINE LEARNING TECHNIQUES FOR CHARACTERIZING IEEE 802.11b ENCRYPTED DATA STREAMS

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Henson, Michael, J., Second Lieutenant, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, Building 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCS/ENG/04-08

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Security Agency
Attn: Mr. William Kroah NSA/R5
9800 Savage Road
Ft. George G. Meade, MD 20755-6799
Comm: (301) 688-0348
e-mail: wtk@afterlife.ncsc.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

As wireless networks become an increasingly common part of the infrastructure in industrialized nations, the vulnerabilities of this technology need to be evaluated. Even though there have been major advancements in encryption technology, security protocols and packet header obfuscation techniques, other distinguishing characteristics do exist in wireless network traffic. These characteristics include packet size, signal strength, channel utilization and others. Using these characteristics, windows of size 11, 31, and 51 packets are collected and machine learning (ML) techniques are trained to classify applications accessing the 802.11b wireless channel. The four applications used for this study included E-Mail, FTP, HTTP, and Print. Using neural networks and decision trees, the overall success (correct identification of applications) of the ML systems ranged from a low average of 65.8% for neural networks to a high of 85.9% for decision trees. These averages are a result of all classification attempts including the case where only one application is accessing the medium and also the unique combinations of two and three different applications.

15. SUBJECT TERMS

Artificial Intelligence, Computer Networks, Data Processing, Wireless Communications, Local Area Networks, Information Security

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Rusty O. Baldwin, Maj, USAF (ENG)
U	**U**	**U**	**UU**	92	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636, ext 4612; e-mail: rusty.baldwin@afit.edu

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18