RUMMAGE
THE PASSIVE EMAIL MONITORING SYSTEM
FOR ENTERPRISE NETWORKS

~~[TYPE THESIS TITLE HERE]~~

Formatted: Space Before: 0 pt, After: 0 pt

by

NC Ali Mustafa Qamar
NC Muhammad Alqama
NC Sana Tariq
NC Riffat Ara

~~[Your Name]~~

~~A thesis submitted in partial fulfillment of the requirements for the degree of~~
Submitted to the Faculty of Computer Science,
Military College of Signals, National University of Sciences and Technology, Rawalpindi

Formatted: Space After: 0 pt

in partial fulfillment for the requirements of B.E. degree in Software Engineering
May 2005

~~[Name of degree]~~
~~[Name of university]~~
~~[Year]~~

~~Approved by~~ _____
~~Chairperson of Supervisory Committee~~

_____

_____

_____

~~Program Authorized to Offer Degree~~ _____

~~Date~~ _____

Formatted: Space After: 0 pt

Formatted: Space After: 0 pt

Formatted: Space Before: 0 pt, After: 0 pt

[NAME OF UNIVERSITY]

ABSTRACT

Information retrieval systems have grown to cover complex tasks of information privacy and security. Where information is assuming more important role in corporate life, there is intense need for developing systems to monitor privacy breaches. RUMMAGE tries to attack the domain from the perspective of email analysis.

Data mining techniques have largely been deployed in the domain of network security. RUMMAGE carries a further step by extending the idea to email analysis domains. Information Retrieval systems provide the basic mechanism to be followed in the analysis of email communication.

Passive systems have proved to be a great asset in the domain of network security. Being passive provides a lot of power to administrators to analyze the system without imposing any burden on underlying network.

RUMMAGE is a passive system that analyses the email text intelligently. It proposes the idea of automated passive analysis of email information. The project attacks the problem by starting to capture the email traffic from network backbone by filtering out unwanted data. Captured emails are disintegrated into fundamental components for further analysis. Intelligent data mining techniques are employed to classify the text correctly.

Chairperson of the Supervisory Committee: Professor [Name]
Department of [Name]

Formatted: Space After: 0 pt

ACKNOWLEDGMENTS

First of all we thank ALLAH Almighty for his gracious blessings that enabled us to complete this project.

We gratefully acknowledge the continuous guidance and motivation provided to us by our internal project advisor Dr. Saeed Murtaza and external advisor Faisal Anwar. We would also like to express our appreciation to the guidance of Lt. Col. Naveed Sarfraz Khattak (MCS) for his suggestions and recommendations.

We are also deeply indebted to our parents for the strength that they gave us through their prayers.

~~[Click and type acknowledgments]~~

Formatted: Space After: 0 pt

Formatted: Footer distance from edge: 0.5"

Formatted: Indent: Left: 0.25", Space After: 0 pt

Formatted: Indent: Left: 0.25", Space After: 0 pt

Formatted: Space After: 0 pt

Formatted: Centered

TABLE OF CONTENTS

1 INTRODUCTION 1
1.1 OVERVIEW 1
1.2 BACKGROUND 2
1.3 PROBLEM FORMULATION 3
1.3.1 PROJECT OBJECTIVES 3
1.3.2 PRODUCT PERSPECTIVES 4
1.3.3 USER INTERFACES AND ASSUMPTIONS 4
1.3.4 PRODUCT FUNCTIONAL REQUIREMENTS 4
1.4 SOFTWARE SYSTEM ATTRIBUTES 4
1.4.1 RELIABILITY AND SYSTEM DEPENDENCY 4
1.4.2 COMPREHENSIVE GUI 4
1.4.3 INTELLIGENT AND CONFIGURABLE 5
1.5 PROJECT APPLICATIONS 5
1.6 MOTIVATION 5
2 LITERATURE REVIEW 6
2.1 EMAIL SYSTEM 6
2.1.1 EMAIL SYSTEM MODELS 6
2.1.2 EMAIL SYSTEM COMPONENTS 7
2.1.3 EMAIL PROTOCOLS 8
2.1.4 MULTIPURPOSE INTERNET MAIL EXTENSIONS (MIME) 9
2.2 DATABASE 12
2.2.1 MYSQL TABLES 13
2.3 PASSIVE NETWORK SYSTEMS 14
2.3.1 COMPONENTS OF PASSIVE NETWORK SYSTEMS 15
2.4 DATA MINING 20
2.4.1 DATA MINING DOMAINS 21
2.4.2 APPLICATIONS OF DATA MINING 21
2.4.3 TEXT MINING 21
2.4.4 FEATURE EXTRACTION 22
2.4.5 EMAIL CLASSIFICATION 24
3 ARCHITECTURAL DESIGN 29
3.1 DESIGN OVERVIEW 29
3.2 CONCEPTUAL MODEL 29
3.3 DISTINCT ARCHITECTURAL COMPONENTS 30
3.4 DETAILED ARCHITECTURAL DESIGN MODEL 30
3.5 IMPLEMENTATION PLATFORM 31
4 EMAIL CAPTURE ENGINE 33
4.1 INTRODUCTION 33
4.2 OBJECTIVES 33
4.3 DESIGN CONSIDERATIONS 33
4.4 ARCHITECTURAL BREAKDOWN AND IMPLEMENTATION 34
4.4.1 PACKET CAPTURE ENGINE 34
4.4.2 REASSEMBLY ENGINE 36
5 EMAIL PARSING AND STORAGE 39
5.1 INTRODUCTION 39
5.2 PROTOCOL DECODING 39
5.2.1 SMTP 39
5.2.2 POP 40

5.2.3 MIME....41
5.3 IMPLEMENTATION....44
5.3.1 DATABASE....44
6 FEATURE EXTRACTOR....46
6.1 INTRODUCTION....46
6.2 AIM OF FILTERS....47
6.3 STANDARD FORM FILTER....47
6.4 STOP LIST FILTER....48
6.5 REMOVAL OF REDUNDANT CHARACTERS....48
6.6 STEMMING TO THE ROOT FORMS....49
6.7 POST PROCESSING-WEIGHT ASSIGNMENT....49
6.7.1 FREQUENCY CALCULATION....49
6.7.2 WEIGHT ASSIGNMENT FORMULAE....50
6.8 CONCLUSIONS....55
7 EMAIL CLASSIFIER....57
7.1 INTRODUCTION....57
7.2 OBJECTIVES....57
7.3 CONCEPTUAL DESIGN....57
7.4 CLASSIFIERS IMPLEMENTED....58
7.4.1 DESIGN CONSIDERATIONS FOR ANN....58
7.4.2 FLOWCHART OF ANN....62
7.4.3 IMPLEMENTATION OF ANN....62
7.4.4 IMPLEMENTATION OF NAIVE BAYES ALGORITHM....63
7.4.5 TESTING OF ANN....66
8 PERFORMANCE ANALYSIS....67
8.1 ARTIFICIAL NEURAL NETWORK....67
8.2 NAIVE BAYES....69
8.3 COMPARISON OF WEIGHT ASSIGNMENT TECHNIQUES....70
9 SUMMARY AND FUTURE WORK....72
9.1 SUMMARY....72
9.2 FUTURE WORK....74
9.2.1 WEB MAIL ANALYSIS....74
9.2.2 AUTOMATED ANALYSIS OF FORMATTED TEXT DOCUMENT ATTACHMENTS....74
9.2.3 OPTIMIZATION TO WORK ON GIGABIT SPEED....74
9.2.4 HANDLING ENCRYPTED TRAFFIC....75
9.2.5 ENHANCEMENTS IN CLASSIFICATION....75
9.2.6 PORTING TO UNIFIED MESSAGING SYSTEMS....75
9.2.7 VOIP ANALYSIS....75
~~[CLICK AND INSERT TABLE OF CONTENTS]~~

LIST OF FIGURES

Figure Number *Page No.*
Figure 2-1: An Email System 6
Figure 2-2: Berkeley Packet Filter (BPF) 17
Figure 2-3: Linux Socket Filter (LSF) 18
Figure 2-4: Internet Header 19
Figure 2-5: TCP Header Format 20
Figure 2-6: An Artificial Neural Network 25
Figure 3-1: Conceptual Design 29
Figure 3-2: Detailed Architectural Design 31
Figure 4-1: Email Capture Engine Stand Point 33
Figure 4-2: Email Capture Engine Architecture Breakdown 34
Figure 4-3: Packet Capture Engine 34
Figure 4-4: Reassembly Engine 36
Figure 4-5: IP Reassembly Algorithm 38
Figure 5-1: MIME Decoder 41
Figure 6-1: Architecture Breakdown 46
Figure 6-2: Preprocessing in Feature Extractor 47
Figure 6-3: Weight Assignment Algorithm 50
Figure 6-4: Flow Chart of Feature Extractor 52
Figure 7-1: Conceptual Design of Email Classifier 58
Figure 7-2: Effect of Varying the Number of Hidden Units on the Performance of ANN 59
Figure 7-3: Effect of Changing Momentum on the Performance of ANN 59
Figure 7-4: Number of Misclassifications vs. Momentum 60
Figure 7-5: Effect of Changing Learning Rate on the Performance of ANN 60
Figure 7-6: Flowchart for Artificial Neural Network 62
Figure 7-7: Algorithm for Training and Classification Phases in ANN 63
Figure 7-8: Naive Bayes Algorithm 64
Figure 8-1: CPU User time (seconds) vs. Number of Examples for 1 and 2 Layers ANN 69
Figure 8-2: CPU User time (seconds) vs. Number of Examples for Naive Bayes 70
Figure 8-3: Comparison of Different Techniques 70

Formatted: Space After: 0 pt

Formatted: Tab stops: Not at 5.17"

LIST OF TABLES

Table Number *Page No.*

Table 8-1: Parameters for Analysis 67
Table 8-2: Results for ANN with 1 Hidden Layer 68
Table 8-3: Results for ANN with 2 Hidden Layers 68
Table 8-4: Different Weight Assignment Techniques and their Classification Accuracies 69

Formatted: Space After: 0 pt

Formatted: Tab stops: Not at 5.17"

ABBREVIATIONS AND ACRONYMS

Abbreviations and Acronyms	Definition
TCP	Transmission Control Protocol
BPF	Berkeley Packet Filter
LSF	Linux Socket Filter
TCB	Transmission Control Block
SMTP	Simple Mail Transfer Protocol
POP	Post Office Protocol
IMAP	Internet Mail Access Protocol
MIME	Multipurpose Internet Mail Extensions
FE	Feature Extractor
ANN	Artificial Neural Network
TST	Transaction Safe Table
NTST	Not Transaction Safe Table

~~[Click and insert List of Figures]~~

~~ACKNOWLEDGMENTS~~

~~The author wishes to [Click and type acknowledgments]~~

~~GLOSSARY~~

~~**Word**. [Click and type definition here.]~~

Chapter 1

1 Introduction

1.1 Overview

With the exponential growth in the quantity and complexity of information sources on the internet, information retrieval systems have evolved from a simple concern with the storage and distribution of artifacts, to encompass a broader concern with the transfer of meaningful information. The need for effective methods of automated information retrieval (IR) has grown in importance because of the tremendous explosion in the amount of unstructured data, both internal, corporate document collections, and the immense and growing number of document sources on the Internet.

Traditional data analysis is assumption driven as a hypothesis is formed and validated against the data. Data mining, in contrast, is discovery driven as the patterns are automatically extracted from data. Data mining is becoming a pervasive technology in activities to predict the success of a marketing campaign, looking for patterns in financial transactions to discover illegal activities. From this perspective, it was just a matter of time for the discipline to reach the important area of computer security. Also the popularity of the Internet was another most important development of the twentieth century that prompted an intensified effort in data security.

Because of the increased reliance on powerful, networked computers to help run businesses and keep track of personal information, industries have been formed around the practice of network and computer security. Enterprises have solicited the knowledge and skills of security experts to properly audit systems and tailor solutions to fit the operating requirements of the organization. Because most organizations are dynamic in nature, with workers accessing company IT resources locally and remotely, the need for secure computing environments has become more pronounced. In modern society security policies and mechanisms are not perfect and more and more organizations are becoming vulnerable to a wide variety of security breaches against information infrastructure. To protect against the security flaws and detect and prevent cyber attacks, the use of data mining techniques is the focus of the research in the areas of information security.

Formatted: Space After: 0 pt
Formatted: Space Before: 0 pt
Formatted: Space After: 0 pt
Formatted: Centered

The project aims at passively sniffing network traffic and applying packet reassembly techniques, implementation of email parser and saving the data into a database, implementing a feature extractor module for pre-classification and constructing an email classifier using Artificial Intelligence techniques. RUMMAGE can be effectively used by a defense organization for information retrieval and by an enterprise for email monitoring. Client email systems can use it to meet certain user preferences. The protocol decoding performed in this project is a valuable addition to the existing research done in this field. RUMMAGE is the first automated software of its kind that employs passive and efficient email capturing to intelligent data mining, satisfying the needs of various organizations in multiple domains.

1.2 Background

Formatted: Space After: 0 pt

Electronic surveillance has been extremely effective in securing the conviction of more than 25,600 dangerous felons over the past 13 years. In this modern day, law enforcement and intelligence communities rely heavily on computer technology to identify, monitor and react to both known and perceived threats. This technology renders the government capable of receiving broad access to the personal information of its people.

The development of Passive email monitoring system has been a major concern of the developed countries for the past few years. The U.S. Federal Bureau of Investigation (FBI) has been in particular been involved in Carnivore, Altivore and Echelon; which are the three big names in the world of network monitoring. Carnivore was responsible for tracking down terrorist activities and detaining 400-500 suspects immediately after the attacks on the World Trade Center in 2001. Despite the programs' occasional flaws in mistaking implicating an innocent person, it is still better to exhaust all possibilities than to miss one.

The world's first arrest resulting from passive monitoring of electronic communications is being reported by Globe Technology. In the article, sources reveal that an e-mail message intercepted by NSA spies precipitated a massive investigation by intelligence officials in several countries that culminated in the arrest of nine men in Britain and one in suburban Orleans, Ont. Email monitoring is also carried out by the powerful Gmail system by Google. The email text scanning infrastructure that Google has built is powerful and global

in reach. Google has proposed scanning the text of all incoming emails for ad placement. The basic idea is this: the service may scan the contents of people's email to figure out the most relevant targeted advertisement.

Also the famous Daniel Pearl murderer's arrest was guided by the scanning the emails sent by the murderer. People typically scan information when they perform searches. Elements that are bolded, in a different color, set off by bulleted or numbered lists, are usually the things that are seen first. This is important, if visitors don't see the terms they are looking for, they then move on to other websites to find the information that they are scanning for. This also works for the search engines. Google gives much weight to words placed in link texts, Title tag, bolded text and alt texts to images.

1.3 Problem Formulation

Email services have become a necessity today. The extensive use of email systems is the result of fast communication it offers, thus adding to its importance in the world today. The simplicity of email systems makes it vulnerable to misuse by sending fake emails using spoofing, spam emails or other web attacks. This illegitimate traffic further increases traffic load on the Internet and also consumes bandwidth resulting in poor service. All these issues are of great concern to an organization where information integrity is very crucial for its sustenance. RUMMAGE provides control and organization to analyze its email traffic to address potential threats of information leak and misuse of email services. It provides a passive system, which means no burden on network.

Formatted: Space After: 0 pt

~~1.4~~1.3.1 P

Formatted: Bullets and Numbering

roject Objectives

~~1.4.7~~The project objectives consisted of implementation of the filters to capture the email traffic from the sniffed raw data obtained from the mirror traffic using the packet filtering techniques; identifying and tracking the email sessions to retrieve the complete email messages from the captured email traffic; implementation of a decode system which is able to decode all three email protocols i.e. SMTP, IMAP and POP; implementation of email parser along with the development of an efficient database system to dump the email traffic; creation of a feature extractor module for pre classification; and analysis of the

captured traffic using sophisticated data mining techniques involving artificial intelligence algorithms.

1.3.2 Product Perspectives

RUMMAGE should be a Passive Implementation with Modular Architecture. It should be artificially intelligent and efficient. It should enable the organization to monitor all outgoing and incoming emails by analyzing SMTP, POP3 and IMAP protocols. It should also decode MIME encoded attachments. The captured email traffic should be dumped into database for offline analysis. The project should automate the process of analysis.

1.3.3 User Interfaces and Assumptions

The user of this software will be a network administrator who will configure the system according to the organizational requirements. The user will be able to directly deal with data base for storing and retrieving information. The only dependency is of its platform being a system fully developed on Linux.

1.3.4 Product Functional Requirements

The functional requirements for the architecture include sniffing the data from data link layer of various networks with different data rates efficiently. Processing of the captured packets should be fast enough to meet the higher data rates. Data base should be configurable through user interface. The intelligent parts of the software must be configurable according to user preferences. In its complete implementation RUMMAGE must have modest CPU and memory requirements.

1.4 Software System Attributes

1.4.1 Reliability and System Dependency

~~Rummage~~RUMMAGE is designed in such a way that makes it easy to deploy in various domains like ISP's, enterprises, defense organizations. It should be reliable software for the organization. RUMMAGE is UNIX based software. It must be able to run on various Linux and UNIX flavors. It must have GLibC installed on the system for its operation.

1.4.2 Comprehensive GUI

RUMMAGE should provide easy to use, comprehensive user interface. The interface should provide ease of monitoring and configuration. RUMMAGE should provide its user

Formatted: Body Text,Body Text Char Char, Line spacing: single

Formatted: Body Text,Body Text Char Char, Line spacing: single

Formatted: Body Text,Body Text Char Char, Line spacing: single

Formatted: Indent: Left: 0"

Formatted: Body Text,Body Text Char Char, Line spacing: single

with the facility to view and manipulate database contents. Email text and attachments should be available to be viewed. GUI should also summarize email traffic statistics.

1.4.3 Intelligent and Configurable

~~Rummage~~Rummage should be an intelligent and self adaptive system. Once configured according to user preferences in various domains it should adapt itself to meet the user requirements well with training and learning.

~~1.4~~1.5 Project Applications

RUMMAGE provides the ground for extensive research work in the field of email security as well as providing quality of service. It can be employed by the security agencies as a monitoring system, developed to intercept e-mail and monitor other online activities of suspected criminals. RUMMAGE may also be exploited by various enterprises. ISPs are preferred to conduct the intercepts and the software can be deployed by the Internet service providers. By analyzing the email attachments and the embedded scripts some work can be done to detect and prevent email viruses, worms etc. Solution for the problem of spam can be developed

Formatted: Space After: 0 pt

~~1.5~~1.6 Motivation

Extensive research is being done in the areas of network security and data mining. ~~RUMMAGE~~Rummage provides us an opportunity to extend the effort and deploy data mining techniques to email monitoring domains. Over the years information security and privacy has become of prime importance across the globe. Keeping in view the ever growing cyber attacks, the system can easily be extended for signature base monitoring. ~~RUMMAGE~~Rummage can prove to be of great importance in tracking down terrorist activities.

Chapter 2

2 Literature Review

~~1.6~~2.1 E-mail System

E-mail system consists of two different servers running on a server machine. One is called the SMTP server, where SMTP stands for Simple Mail Transfer Protocol. The SMTP server handles outgoing mail. The other is either a POP3 server or an IMAP server, both of which handle incoming mail. POP stands for Post Office Protocol, and IMAP stands for Internet Mail Access Protocol. A typical e-mail system is depicted in the Figure 2-1. The SMTP server listens on well-known port number 25, POP3 listens on port 110 and IMAP uses port 143.

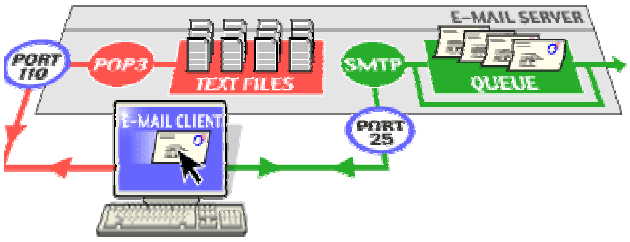


Figure 2-1~~.2-1~~: An Email System

~~1.6.1~~2.1.1 E-mail System Models

There are three models of using client/server electronic mail. Two of these models map directly into POP or IMAP.

The offline model has been the most popular form of client/server e-mail at UW to date, and is used by protocols such as POP3. In this model, a client application periodically connects to a server, downloads all pending messages to the client machine and then deletes these messages from the server. The connection is only periodic, even though the computer maintains a modem connection throughout. All of the mail is processed locally on the client computer.

The online model is most commonly associated with the IMAP mail protocol. In this model, a client application manipulates mailbox data on a server, maintaining a connection

Formatted: Space After: 0 pt

Formatted: Outline numbered + Level: 1 + Numbering Style: 1, 2, 3, … + Start at: 1 + Alignment: Left + Aligned at: 0" + Tab after: 0.3" + Indent at: 0.3"

Formatted: Space After: 0 pt

Formatted: Centered, Space After: 0 pt

Formatted: Space After: 0 pt, Tab stops: 0.5", Right

Formatted: Space After: 0 pt

throughout the session. The client stores no mailbox data and only retrieves data from the server as needed. Nothing can be done with the messages if the client is disconnected from the server.

The disconnected user model offers a hybrid of the offline and online models. In this model, a user can download some set of messages from the server, manipulate them offline, and then upload the changes at some later time. The server is again the main repository of the messages. This is the model required by people who travel a lot, and wish to process mail without maintaining a modem connection.

Most email servers conduct email services by running two separate processes on the same machine. One process is the POP3 server, which holds emails in a queue and delivers emails to the client when they are requested. There is the SMTP server that receives outgoing emails from clients and sends and receives emails from other SMTP servers. These two processes are linked by an internal mail delivery mechanism that moves mail between the POP3 and SMTP servers.

~~1.6.2~~2.1.2 **E**

mail System Components

The software programs that handle Internet messages are called agents. There are three types of Internet messaging agents: the Mail Transport Agent (MTA), Mail Delivery Agent MDA), and Mail User Agent (MUA).

Formatted: Space After: 0 pt

An MTA is a program that transmits and receives messages between messaging sites. The sending MTA accepts messages from end user client software and transmits it to a receiving MTA. The receiving MTA receives messages from the sending MTA, determines whether or not the recipient resides locally on the receiving MTA (server) system, and then hands off the message for delivery.

The MDA is the trench soldier: the grunt of Internet messaging. All the MDA knows is how to determine which local user the message is destined for and how to put the message in the correct place in the mail store. The MTA hands the MDA each Internet message destined for a local user and that the MDA is responsible for knowing where to place it in the mail store.

The MUA is the interface between the MTA and the most unpredictable component of the IMM: the user himself. The MUA typically retrieves messages from the mail store in one of three ways: by using a mail access protocol like IMAP or POP, by using a remote file access protocol, or by accessing local files.

~~1.6.3~~2.1.3 E

mail Protocols

There are two commonly used email protocols (the method by which the email client communicates with the email server) for receiving mail. These are Post Office Protocol (POP) and Internet Message Access Protocol (IMAP). The current versions of these protocols are POP3 and IMAP4 respectively. There is another protocol involved in email. It is the Simple Mail Transfer Protocol (SMTP) and is the protocol used by the mail server for sending email.

Formatted: Space After: 0 pt

A POP [1] mail server transfers a copy of all the messages in the user's inbox down to the client computer whenever a connection is made. The contents of the inbox on the server can then be deleted. (Options do exist to leave a copy of the messages on the server for some period of time). All other mail folders are stored on the client computer. The address book is on the client computer.

When a connection is made to an IMAP [2] mail server, a copy of the headers (date, from, subject) of the mail messages are transferred to the client computer. The actual messages remain in the inbox on the server until deleted. Mail folders can be stored on either the client computer or the server. Conceptually the address book could reside on either the client computer or the server. However, all current email clients require that the address book resides on the client computer, whether the POP or IMAP protocol is being used.

Simple Mail Transfer Protocol (SMTP) [3] is Internet's standard host-to-host mail transport protocol and traditionally operates over TCP, port 25. The objective of Simple Mail Transfer Protocol (SMTP) is to transfer mail reliably and efficiently. SMTP is independent of the particular transmission subsystem and requires only a reliable ordered data stream channel.

~~1.6.3.1~~2.1.4 M

Formatted: Heading 3

Formatted: Bullets and Numbering

ultipurpose Internet Mail Extensions (MIME)~~IME~~

MIME [4] is a supplementary protocol. It is an extension to SMTP and is not a mail protocol that can replace SMTP. The primary motivator for the creation of the working group that created MIME was to support non-ASCII character sets necessary for email in languages other than English. A secondary motivator was a requirement for a standard way to send attachments. Less important, but also a motivating factor, was the need for a standard way to send multimedia content. MIME came about through the realization that a single solution could address all three needs.

With MIME, users can send attachments containing any kind of data, of arbitrary length. MIME messages can point to files or other data outside the mail message. The only functional limitation is that the MUA on each end must know how to handle the particular MIME type.

Formatted: Space After: 0 pt

Shortly after MIME began being used, the Web became popular, and suddenly, people needed to send URLs via email. Sending the URL to a file instead of the file itself is popular. Instead of sending the file as an attachment, the user sends a pointer to the file. Large mail attachments can be problematic. Many ISPs still use only POP service and implement it in such a way that forces users to download all new email without picking and choosing particular messages. Messages with large attachments make downloading POP mail painfully time consuming. SMTP servers often have size limits on messages they'll accept (typically 10–20 MB per message).

If a message has a large attachment, it could be rejected by the SMTP server. Sending the URL instead of the file it gets around those problems. It's no wonder URLs are a popular way of conveying information stored in large files. A common use for MIME nowadays is to send two versions of the message: a *text/plain* version and a *text/html* version with more formatting. The Multipurpose Internet Mail Extensions, or MIME, redefines the format of messages to allow for- textual message bodies in character sets other than US-ASCII, an extensible set of different formats for non-textual message bodies, multi-part message bodies, and textual header information in character sets other than US-ASCII.

~~1.6.3.1.1~~2.1.4.1 M

IME Header Fields

MIME defines a number of new header fields that are used to describe the content of a MIME entity. These header fields occur in at least two contexts i.e. as part of a regular RFC 822 message header or in a MIME body part header within a multipart construct.

A header field, "MIME-Version", is used to declare the version of the Internet message body format standard in use. The presence of this header field is an assertion that the message has been composed in compliance with the standard for MIME attached messages. Many media types which could be usefully transported via email are represented, in their "natural" format, as 8bit character or binary data. Such data cannot be transmitted over some transfer protocols. It is necessary, therefore, to define a standard mechanism for re-encoding such data into a 7-bit short-line format. The Content-Transfer-Encoding field is used to indicate the type of transformation that has been used in order to represent the body in an acceptable manner for transport.

~~1.6.3.2~~2.1.4.2 C

ontent Transfer Encoding

Proper labeling of uuencoded material in less restrictive formats for direct use over less restrictive transports is also desirable. Such encodings are indicated by a "Content-Transfer-Encoding" header field. These values are not case sensitive -- Base64, BASE64 and bAsE64 are all equivalent. An encoding type of 7BIT requires that the body is already in a 7bit mail-ready representation. This is the default value i.e. "Content-Transfer-Encoding: 7BIT" is assumed if the Content-Transfer-Encoding header field is not present.

Base64 is a data encoding scheme whereby binary-encoded data is converted to printable ASCII characters. It is defined as a MIME content transfer encoding for use in internet e-mail. The only characters used are the upper- and lower-case Roman alphabet characters (A-Z, a-z), the numerals (0-9), and the "+" and "/" symbols, with the "=" symbol as a special suffix code. The scheme is defined only for data whose original length is a multiple of 8 bits, a requirement met by most computer file formats. The resultant base64-encoded data has a length that is approximately 33% greater than the original data, and typically appears as seemingly random characters.

Formatted: Font: Not Italic
Formatted: Heading 4
Formatted: Bullets and Numbering
Formatted: Space After: 0 pt
Formatted: Space After: 0 pt

The Quoted-Printable encoding is intended to represent data that largely consists of octets that correspond to printable characters in the ASCII character set. It encodes the data ~~in such~~in ~~a~~such a way that the resulting octets are unlikely to be modified by mail transport. If the data ~~being encoded~~being ~~are~~encoded are ~~mostly ASCII~~mostly ~~text~~ASCII text~~, the~~, the ~~encoded~~ form ~~of~~ the data remains largely recognizable by humans. A body which is entirely ASCII may also be encoded in Quoted-Printable to ensure the integrity of the data should the message pass through a character-translating, and/or line-wrapping gateway.

The purpose of the Content-Type field is to describe the data contained in the body fully enough that the receiving user agent can pick an appropriate agent or mechanism to present the data to the user, or otherwise deal with the data in an appropriate manner. The value in this field is called a media type. The Content-Type header field specifies the nature of the data in the body of an entity by giving media type and subtype identifiers, and by providing auxiliary information that may be required for certain media types. After the media type and subtype names, the remainder of the header field is simply a set of parameters, specified in an attribute=value notation. The ordering of parameters is not significant.

2.1.4.3 Discrete Media Types

~~Textual information.~~ The simplest and most important subtype of "text" is "plain". This indicates plain text that does not contain any formatting commands or directives. Plain text is intended to be displayed "as-is", that is, no interpretation of embedded formatting commands, font attribute specifications, processing instructions, interpretation directives, or content markup should be necessary for proper display. Unrecognized subtypes of "text" should be treated as subtype "plain".

~~Image data.~~ "Image" requires a display device to view the information. An initial subtype is defined for the widely-used image format JPEG. The two widely used subtypes are jpeg and gif. Unrecognized subtypes of "image" should at a ~~miniumum~~minimum be treated as “application/octet-stream”.

Formatted: Heading 4

Formatted: Font: Not Italic

Formatted: Space After: 0 pt

~~Audio data.~~ "Audio" requires an audio output device to "display" the contents. An initial subtype "basic" is defined. Unrecognized subtypes of "audio" should at a ~~miniumum~~minimum be treated as "application/octet-stream".

A media type of "video" indicates that the body contains a time-varying-picture image, possibly with color and coordinated sound. "Video" requires the capability to display moving images, typically including specialized hardware and software. The subtype "mpeg" refers to video coded according to the MPEG standard (MPEG). Unrecognized subtypes of "video" should at a min~~i~~mum be treated as "application/octet-stream".

Another media type is some other kind of data, typically either un-interpreted binary data or information to be processed by an application. The subtype "octet-stream" is to be used in the case of un-interpreted binary data, in which case the simplest recommended action is to offer to write the information into a file for the user. The "PostScript" subtype is also defined for the transport of PostScript material.

2.1.4.4 Composite Media Types

They consist of multipart and message. Multipart is the data consisting of multiple entities of independent data types. In the case of multipart entities, in which one or more different sets of data are combined in a single body, a "multipart" media type field must appear in the entity's header. The body must then contain one or more body parts, each preceded by a boundary delimiter line, and the last one followed by a closing boundary delimiter line. After its boundary delimiter line, each body part then consists of a header area, a blank line, and a body area.

The primary subtype for multipart, "mixed", is intended for use when the body parts are independent and intended to be displayed serially. A body of media type "message" is itself all or a portion of some kind of message object. Such objects may or may not in turn contain other entities.

2.2 Database

~~MYSQL Database~~

Formatted: Font: Not Italic
Formatted: Heading 4
Formatted: Space After: 0 pt
Formatted: Heading 2
Formatted: Space After: 0 pt

A database organizes information in a logical way, so that it can be accessed and maintained easily. There are two types of databases: flat file and relational. A flat file database contains all the information one might need in one datasheet or table. A relational database, however, consists of many different tables linked together by 'key' (common) fields. The data can be manipulated in many different ways to produce different results. The MySQL database package consists of MySQL server, MySQL client programs and MySQL client library. MySQL server is the heart of MySQL and can be considered as a program that stores and manages the databases. MySQL comes with many client programs. The one which is mostly used is called mysql (note: ~~smallcaps~~small caps). This provides an interface through which SQL statements can be issued and have the results displayed. MySQL client library can help in writing client programs in C.

MySQL is classified as a Relational ~~DataBase~~Database Management System. The collection of data in a database is organized into tables. Each table is organized into rows and columns.

Each row in a table is a record.

Formatted: Heading 3

2.2.1 MySQL Tables

Formatted: Space After: 0 pt

MySQL supports two different kinds of tables: transaction-safe tables (InnoDB and BDB) and not transaction-safe tables (HEAP, ISAM, MERGE, and MyISAM).

In transaction-safe tables (TST); many statements can be combined and these all can be accepted in one go with the COMMIT command. ROLLBACK can be executed to ignore the changes (if it is not the auto-commit mode). If an update fails, all of the changes will be restored. (With NTST tables all changes that have taken place are permanent). Not transaction-safe tables (NTST) are

~~Safer.~~ Faster than TST as there is no transaction overhead, they use less disk space and require less memory to do updates.

As the default table type in MySQL, MyISAM tables probably persist more than 95% of all MySQL data worldwide. MyISAM tables have very little storage overhead. The underlying

storage for MyISAM tables is quite transparent, too. Every table is composed of three files. This makes it very easy to see what's going on i.e. how quickly tables are growing, which ones have changed recently, and so on. They also have some handy features that make building web applications easier. Full-text search is also available only in MyISAM tables. Adding efficient keyword searches to the MySQL applications has never been easier. MySQL spends very little time shuffling bytes around or reading unnecessary data when a row is required in a MyISAM table. As a result, MyISAM tables are blazingly fast compared to most other disk-based relational database engines and even some commercial in-memory databases.

Two of the downsides of MyISAM tables (which result in a move to InnoDB for some applications) are locking and the lack of transactions. MyISAM tables use table-level locking. SELECT queries use shared locks, while all write queries (INSERT, UPDATE, and DELETE) set exclusive locks. While this may sound like a disaster waiting to happen, it is a fact that MyISAM tables are fast. That means most locks are held only for very short periods of time, unless a very poorly optimized query is executed. Lock contention really doesn't become an issue until the load on the server gets quite high.

2.3 Passive Network Systems

Passive network monitors interoperate with the live network at the link level. They are considered a vendor independent means of gathering data as they do not depend on features that would otherwise have to be provided by active network equipment operating as routers, switches, hubs or end systems. Being non intrusive in nature they place no burden on underlying network. Operating at the medium dependent physical layer, these systems re-implement all layer one and two functions, such as de-serialization, packetization and various packet encapsulations, and then pass the data on to analysis-specific functions, such as arrival time stamping, selective filtering and payload discard or flow state analysis. Such functions are executed by reconstructing and accessing information that is specific to network layers three to seven.

Formatted: Heading 2

Formatted: Font: (Default) Times New Roman, 12 pt

Formatted: Body Text,Body Text Char Char, Adjust space between Latin and Asian text, Adjust space between Asian text and numbers

2.3.1 Components of Passive Network Systems

Operating at data link layer to capture raw traffic flowing over the network and being able to perform application layer protocol analysis requires that network systems be equipped with sniffing, filtering, re-assembly and protocol analysis features.

2.3.1.1 Packet Sniffer

Packet sniffing is listening (with software) to the raw network device for packets that are of interest. When software sees a packet that fits certain criteria, it passes them on for further analysis to user programmed analysis stack.

Raw sockets let us read and write ICMPv4, IGMPv4 and ICMPv6 packets. This capability also allows applications that are built using ICMP or IGMP to be handled entirely as user processes, instead of putting more code into the kernel. Where raw sockets provide us with the functionality of writing raw IP packets directly to network and data-link layers bypassing the kernel, their packet capture capabilities are limited. No TCP or UDP packets are ever passed to raw sockets by the kernel. Thus they are limited only to capture of ICMP and IGMP packets.

Packet sniffing at link layer is dependant on type of network interface supported by the operating system. Network interfaces include LLI, NIT (Network Interface Tap), Ultrix Packet Filter, DLPI (Data Link Provider Interface), BPF (Berkeley Packet Filter).

LLI was a network interface used by SCO, which has been augmented with DLPI support as of SCO OpenServer Release V. NIT was a network interface used by Sun, but has been replaced in later releases of SunOS/Solaris with DLPI. Ultrix supported the Ultrix Packet Filter before Digital implemented support for BPF. DLPI is supported under current releases of System V Release 4, SunOS/Solaris, AIX, HP/UX, UnixWare, Irix, and MacOS. DLPI is partially supported under Digital Unix. Sun DLPI version 2 supports Ethernet, X.25 LAPB, SDLC, ISDN LAPD, CSMA/CD, FDDI, token ring, token bus, and Bisync as data-link protocols. BPF is supported under current releases of BSD and Digital Unix, and has been ported to SunOS and Solaris. AIX supports BPF reads, but not writes. A BPF library is available for Linux.

The newer versions of Linux provide very powerful interface to data link level packet access called Packet Sockets. Packet socket interface is a high level clone BPF. Packet

Formatted: Heading 3, No bullets or numbering, Adjust space between Latin and Asian text, Adjust space between Asian text and numbers

Formatted: Body Text,Body Text Char Char, Adjust space between Latin and Asian text, Adjust space between Asian text and numbers

Formatted: Font: (Default) Times New Roman, 12 pt

Formatted: Font: (Default) Times New Roman, 12 pt, Not Italic

Formatted: Font: (Default) Times New Roman, 12 pt, Not Italic

Formatted: Font: (Default) Times New Roman, 12 pt, Not Bold

Formatted: Font: 12 pt

Formatted: Body Text,Body Text Char Char

sockets are used to receive or send raw packets at the device driver (OSI Layer 2) level. They allow the user to implement protocol modules in user space on top to physical layer. Packet socket is either raw socket for raw packets including the link level header or datagram socket for cooked packet with the link level header removed.

In raw packet socket interface, packets are passed to and from device driver without any changes in packet data. When receiving a packet the network address information is still parsed by kernel. When transmitting a packet the user supplied buffer should contain the physical layer header. The packet is then queued unmodified to the network driver of the interface defined by the destination address.

Datagram packet sockets operate on a slightly higher level. The physical header is removed before the packet is passed to the user. Packets sent through a datagram packet socket get a suitable physical layer header based on the information with kernel for destination address before they are queued at the network interface. Promiscuous mode is a packet capture mode for Ethernet cards which enables the card to capture even those packets arriving at the network interface that are not destined for it and pass them on to packet sniffer. Ethernet card is set into promiscuous mode before sniffing.

2.3.1.2 Packet Filter

Packet filtering refers to the technique of conditionally allowing or denying packets entering or exiting a network or host based on the characteristics of that packet. The network layer portion of a firewall solution, packet filtering is one part of a good security stance. As the embodiment and manifestation of an organizational security policy for network layer traffic, the packet filter restricts traffic flows between networks and hosts. There is tremendous value from a security perspective in enforcing these traffic flows instead of allowing arbitrary traffic flows.

There are two fundamental types of packet filters static and dynamic. A dynamic packet filter keeps track of the connections currently passing the firewall. This is usually described as a stateful or dynamic packet filtering engine. Netfilter provides the capability for Linux (2.4+) to operate as a stateful packet filtering device. Netfilters are a set of hooks in the Linux Kernel that allow to catch packets and filter, change (mangle) or transform (NAT) them. With Netfilters one can select packets based on many parameters like

Formatted: Font: 12 pt, Not Bold
Formatted: Font: 12 pt
Formatted: Font: 12 pt, Not Bold
Formatted: Font: 12 pt
Formatted: Body Text,Body Text Char Char, Adjust space between Latin and Asian text, Adjust space between Asian text and numbers
Formatted: Font: Not Italic
Formatted: Font: Not Italic

source/destination IP address or port, state of the packet, owner of the packet or flag; drop, accept or reject packets; mangle packets, that is mark them or change some of their flags; NAT (Network Address Translation) which means changing their IP address (source or destination) and classify packets.

A static packet filter is a set of rules against which every packet is checked and allowed or denied. The Berkeley Packet Filter [5] provides a raw interface to data link layers in a protocol independent fashion. All packets on the network, even destined for other hosts, are accessible through this mechanism. Besides packet capture they provide extensive and efficient packet filtering mechanism. BPF is depicted in Figure 2-2. BPF use a pseudo filter language used to install filters. BPF are only supported on Berkeley derived UNIX kernels.

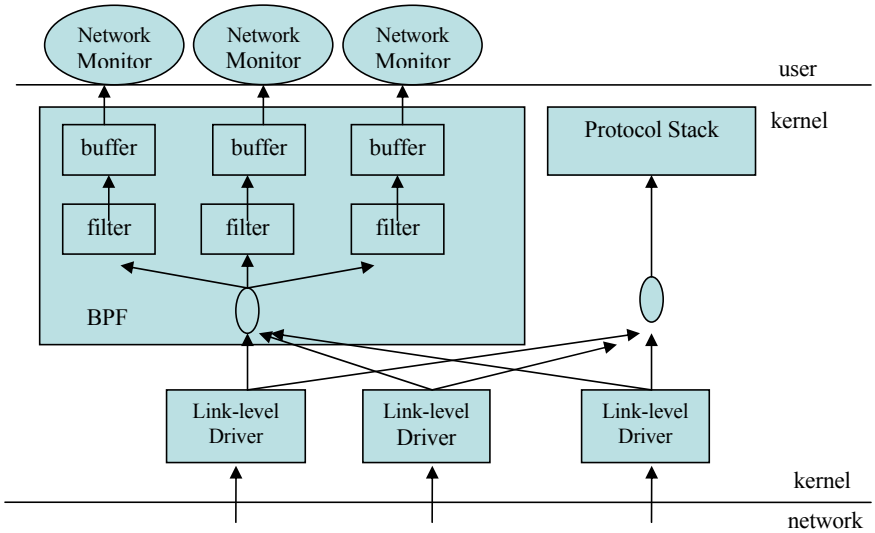


Figure 2-2: Berkeley Packet Filter (BPF)

Since Linux version 2.2, there has been a nice little feature called LSF, Linux Socket Filter. Linux Socket Filter is actually an extra layer to the networking code of Linux, as it sits on top of the device driver, but is still independent of other layers. It provides means of applying filters within kernel directly above device driver layer. Linux Socket Filters work with Linux Packet Sockets for effective packet filtering. Figure 2-3 shows the Linux Socket Filter.

Formatted: Font: (Default) Times New Roman, 12 pt

Formatted: Font: (Default) Times New Roman, 12 pt

Formatted: Font: Not Bold, Font color: Auto

Formatted: Font: Not Bold, Font color: Auto

Formatted: Caption, Centered, Line spacing: 1.5 lines, Adjust space between Latin and Asian text, Adjust space between Asian text and numbers

Formatted: Body Text,Body Text Char Char

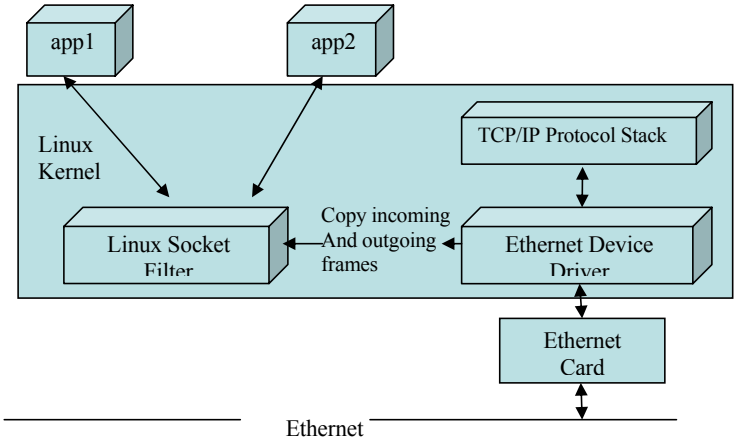


Figure 2-3: Linux Socket Filter (LSF)

Formatted: Caption, Centered, Line spacing: 1.5 lines

2.3.1.3 Traffic Re-Assembly Engine

Formatted: Font: Not Italic

Formatted: Font: Not Italic

Formatted: Heading 4

Formatted: Body Text,Body Text Char Char, Adjust space between Latin and Asian text, Adjust space between Asian text and numbers

In internet communication user data is not transmitted as a single chunk but is divided into smaller fragments to meet the limitation of the communication system. In OSI and TCP / IP models user data is divided at almost each layer to meet the transmission requirements of lower layer or the underlying network. Therefore in order to understand the application level communication, fragmented traffic must be assembled into application level protocol streams. Thus during passive analysis of application layer protocols like Email protocols, it becomes necessary to perform traffic re-assembly particularly at two levels: network layer; IP de-fragmentation and transport layer; TCP stream reassembly.

The internet datagram is fragmented when it is originated in a local net that allows a large packet size and must traverse a local net that limits packets to a smaller size to reach its destination. The internet fragmentation and reassembly procedure provides to break a datagram into an almost arbitrary number of pieces that are de-fragmented at the destination host. The internet modules use fields in the internet header to fragment and reassemble internet datagrams when necessary for transmission. The internet protocol standard [6] describes the detailed structure of internet header. The fields in internet header such as *Identification field*, Fragment offset field and more fragment field are used in fragmentation and reassembly procedure.

Formatted: Font: Not Bold, Font color: Auto

Formatted: Font: Not Bold, Font color: Auto

To fragment a long internet datagram, an internet protocol module creates two new internet datagrams and copies the contents to the internet header fields from the long datagram into both new internet headers. The data of the long datagram is divided into two portions on an 8 octet boundary. To assemble the fragments of an internet datagram, an internet protocol module combines internet datagrams that all have the same value for four fields: identification, source, destination and protocol. The combination is done by placing the data portion of each fragment in the relative position indicated by the fragment offset in that fragment's header. The first fragment will have the fragment offset zero and the last one will have the more-fragments flag reset to zero. Internet Header is shown in Figure 2-4.

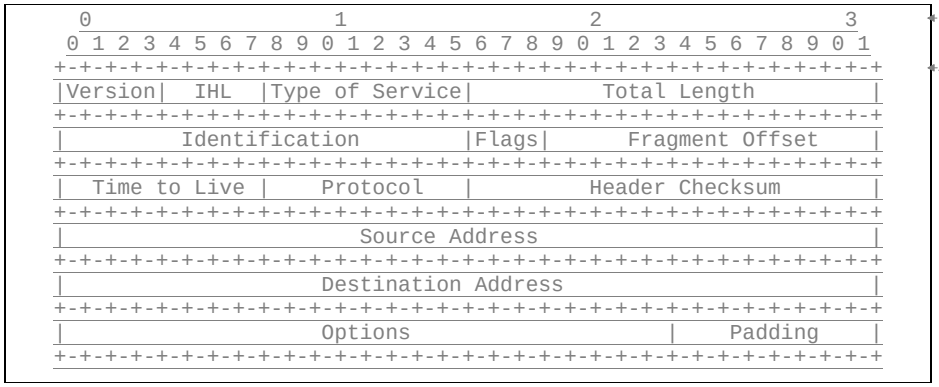


Figure 2-4: Internet Header

Transmission Control Protocol [7] is a connection oriented protocol which was designed for reliable communication on the top of unreliable, connectionless Internet Protocol. Reliable data transfer, effective flow control, multiplexing and connection oriented communication is the essence of TCP.TCP provides ports and socket mechanism and sequence numbers for reliable connection oriented communication. A pair of sockets uniquely identifies each connection on communicating entities. Sequence numbers are used to correctly order segments that may be received out of order.

Processes transmit data by calling on the TCP and passing buffers of data as arguments. The TCP packages the data from these buffers into segments and calls on the internet module to transmit each segment to the destination TCP. The receiving TCP places the data from a segment into receiving user's buffer and notifies the receiving user. The TCPs

Formatted: Centered, Border: Top: (Single solid line, Auto, 0.5 pt Line width), Bottom: (Single solid line, Auto, 0.5 pt Line width, From text: 6 pt Border spacing:), Left: (Single solid line, Auto, 0.5 pt Line width), Right: (Single solid line, Auto, 0.5 pt Line width)

Formatted: Centered, Border: Top: (Single solid line, Auto, 0.5 pt Line width), Bottom: (Single solid line, Auto, 0.5 pt Line width, From text: 6 pt Border spacing:), Left: (Single solid line, Auto, 0.5 pt Line width), Right: (Single solid line, Auto, 0.5 pt Line width)

Formatted: Font: 12 pt, Not Italic

Formatted: Caption, Centered, Line spacing: 1.5 lines, Adjust space between Latin and Asian text, Adjust space between Asian text and numbers

Formatted: Body Text,Body Text Char Char, Adjust space between Latin and Asian text, Adjust space between Asian text and numbers

Formatted: Font: Not Bold, Font color: Auto

Formatted: Font: Not Bold, Font color: Auto

include control information in segments which they use to ensure reliable ordered data transmission. TCP Header format is given in the Figure 2-5.

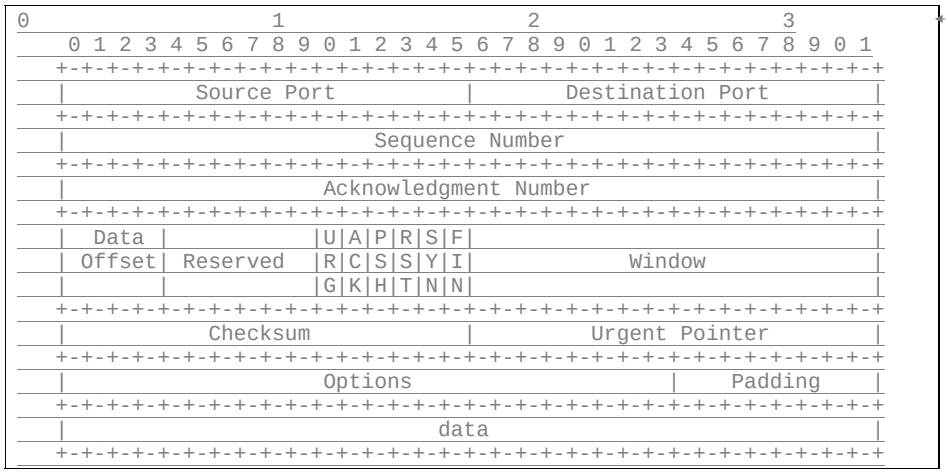


Figure 2-5: TCP Header Format

2.3.1.4 Protocol Analysis

Understanding the specific protocol's request/reply messages, and learning how to recognize related headers and payload information is the key requirement for analysis of application layer protocols. Application layer protocols are usually ASCII based, with human readable request/reply commands. Client and server applications exchange these commands on the top of some Transport layer protocol. The analysis of protocol streams involve carefully analyzing and verifying protocols commands to extract the required information.

2.4 Data Mining

Data Mining is the process of extracting knowledge hidden from large volumes of raw data. The importance of collecting data that reflect the business or scientific activities to achieve competitive advantage is widely recognized now. Powerful systems for collecting data and managing it in large databases are in place in all large and mid-range companies. However, the bottleneck of turning this data into success is the difficulty of extracting knowledge about the system which is studied from the collected data. Human analysts with no special tools can no longer make sense of enormous volumes of data that require processing in order to make informed business decisions. Data mining automates the process of finding

Formatted: Border: Top: (Single solid line, Auto, 0.5 pt Line width), Bottom: (Single solid line, Auto, 0.5 pt Line width), Left: (Single solid line, Auto, 0.5 pt Line width), Right: (Single solid line, Auto, 0.5 pt Line width)

Formatted: Caption, Centered, Line spacing: 1.5 lines

Formatted: Font: Not Italic

Formatted: Heading 4, Adjust space between Latin and Asian text, Adjust space between Asian text and numbers

Formatted: Font: Not Italic

Formatted: Body Text,Body Text Char Char

Formatted: Heading 2, Left

relationships and patterns in raw data and delivers results that can be either utilized in an automated decision support system or assessed by a human analyst.

2.4.1 Data Mining Domains

They consist of predicting which is a task of learning a pattern from examples and using the developed model to predict future values of the target variable; classification meaning finding a function that maps records into one of several discrete classes; clustering which is basically a task of identifying groups of records that is similar between themselves but different from the rest of the data (Often, the variables providing the best clustering should be identified as well) and market basket analysis which involves processing transactional data in order to find those groups of products that are sold together well. One also searches for directed association rules identifying the best product to be offered with a current selection of purchased products.

2.4.2 Applications of Data Mining

Data might be one of the most valuable assets of a corporation - but only one knows how to reveal valuable knowledge hidden in raw data. Data mining allows extracting diamonds of knowledge from the historical data and predicting outcomes of future situations. It will help in optimizing the business decisions, increasing the value of each customer and communication, and improving the satisfaction of customer with the services. Data that require analysis differ for companies in different industries. Examples include: Sales and contacts histories, Call support data, Demographic data on the customers and prospects, Patient diagnoses and prescribed drugs data, Clickstream and transactional data from a website. In all these cases data mining can help in revealing knowledge hidden in data and turn this knowledge into a crucial competitive advantage. Today increasingly more companies acknowledge the value of this new opportunity and turn to Megaputer for leading edge data mining tools and solutions that help optimizing their operations and increase the bottom line.

2.4.3 Text Mining

Text Mining is the discovery by computer of new, previously unknown information, by automatically extracting information from different written resources. A key element is the linking together of the extracted information together to form new facts or new hypotheses

Formatted: Body Text,Body Text Char Char

Formatted: Heading 3, Left

Formatted: Body Text,Body Text Char Char, Space Before: 0 pt, After: 0 pt

Formatted: Heading 3

to be explored further by more conventional means of experimentation. It is different from the web search. In search, the user is typically looking for something that is already known and has been written by someone else. The problem is pushing aside all the material that currently isn't relevant to the needs in order to find the relevant information.

2.4.4 Feature Extraction

Formatted: Heading 3, Left

Formatted: Body Text,Body Text Char Char, Indent: Left: 0"

Feature Extraction is a sub branch of text classification which transforms a text document into feature vectors that can serve as input to the classifier .When a document is given as input it identifies text patterns that helps in classification of the document to some predefined category. The stream of words is passed through a series of filters before text analysis can be performed on it.

Formatted: Body Text,Body Text Char Char

Feature Extraction is the process of extracting useful information from raw text documents which can help classify it to some predefined category. Most machine learning methods treat text documents as a feature vector. A simple way to transform a text document into a feature vector is using a "bag-of-words" representation, where each feature is a single token. There are two problems associated with this representation.

The first problem to be raised when using a feature vector representation is to answer the question, "what is a feature?" In general, a feature can be either local or global. In text categorization, local features are always used but in different-length scales of locality. A feature can be as simple as a single token, or a linguistic phrase, or a much more complicated syntax template. A feature can be a characteristic quantity at linguistic levels. To transform a document, which can be regarded as a string of tokens, into another set of tokens will lose some linguistic information such as word sequence. Word sequence is crucial for a human being to understand a document and should be also crucial for a computer. Using phrases as features is a partial solution for incorporating word sequence information into text categorization.

A feature weight should show the degree of information represented by local feature occurrences in a document, at a minimum. A slightly more complicated feature weight scheme may also represent statistical information of the feature's occurrence within the whole training set or in a pre-existing knowledge base (taxonomy or ontology). A yet more

complicated feature weight may also include information about feature distribution among different classes.

2.4.4.1 From Text to Features

Formatted: Font: Not Italic

In order to transform a document into a feature vector, preprocessing is needed. This includes feature formation (tokenization, phrase formation, or higher level feature extraction), feature selection, and feature score calculations. Tokenization is a trivial problem for white-spaced languages like English. Feature formation must be performed with reference to the definition of the features. Different linguistic components of a document can form different types of features. Features such as single tokens or single stemmed tokens are most frequently used in text categorization.

In this bag-of-words representation, information about dependencies and the relative positions of different tokens are not used. Phrasal features consisting of more than one token are one possible way to make use of the dependencies and relative positions of component tokens. Previous experiments show that introducing some degree of term dependence in the Bayesian network method will achieve undoubtedly higher accuracy in text categorization compared to the independence assumption in the Naive Bayesian method. However, whether the introduction of phrases will improve the accuracy of text categorization has been debated for a long time. In this Feature Extractor single tokens are used and evaluated as most effective.

Some applications of feature extraction are latent semantic analysis, data compression, data decomposition and projection, and pattern recognition. Feature extraction can also be used to enhance the speed and effectiveness of supervised learning. For example, feature extraction can be used to extract the themes of a document collection, where documents are represented by a set of key words and their frequencies. Each theme (feature) is represented by a combination of keywords.

Formatted: Body Text,Body Text Char Char

Motivated by applications like spam filtering, e-mail routing, Web directory maintenance, and news filtering, text classification has been researched extensively in recent years. Text categorization is problem of text classification which assigns text content to predefined Categories. Feature extraction is a must to optimal performance of any standard classifier and its various techniques affect the classifier's performance to a thousand folds.

2.4.4.2 Stemming

Formatted: Font: Not Italic

Formatted: Heading 4, Left

In most cases, morphological variants of words have similar semantic interpretations and can be considered as equivalent for the purpose of IR applications. For this reason, a number of so-called *stemming Algorithms*, or *stemmers*, have been developed, which attempt to reduce a word to its *stem* or root form. Thus, the key terms of a query or document are represented by stems rather than by the original words. This not only means that different variants of a term can be *conflated* to a single representative form – it also reduces the *dictionary size*, that is, the number of distinct terms needed for representing a set of documents. A smaller dictionary size results in a saving of storage space and processing time.

The Porter Stemmer is a conflation Stemmer developed by Martin Porter at the University of Cambridge in 1980. The Stemmer is based on the idea that the suffixes in the English language (approximately 1200) are mostly made up of a combination of smaller and simpler suffixes. This Stemmer is a linear step Stemmer. Specifically it has five steps applying rules within each step.

The Krovetz Stemmer was developed by Bob Krovetz, at the University of Massachusetts, in 1993. It is quite a 'light' stemmer, as it makes use of inflectional linguistic morphology. The Krovetz Stemmer effectively and accurately removes inflectional suffixes in three steps, the conversion of a plural to its single form (e.g. '-ies', '-es', '-s'), the conversion of past to present tense (e.g. '-ed'), and the removal of '-ing'. The conversion process firstly removes the suffix, and then though a process of checking in a dictionary for any recoding (also being aware of exceptions to the normal recoding rules), returns the stem to a word.

———**Email**

Formatted: Heading 3

2.4.5 Classification

Email classification comes in the domain of text classification. Normally emails are classified in the broad categories of spam and ham messages. An important tradeoff in spam filtering is between false positive (ham messages flagged as spam) and false negative (*misses*, spam messages flagged as ham) rates. A detector that always says "ham", after all, will never experience a false positive. False positives, however, are generally much more

expensive than false negatives, so it may be desirable to operate the filter outside of its optimal range.

The accuracy of the corpus is also a concern. It is to be expected that a certain amount of misclassification of messages and misrecognition of features will be present in the data. Different learners may cope with different types of features and classifications. Ultimately, spam filtering tends to concern itself with a binary classification: ham vs. spam. More sophisticated document classifiers can provide both more detailed classification outputs (e.g. "Nigerian spam", "message from Mom") and more precise estimates of their classification confidence.

Some machine learners (notably neural nets) can handle continuous feature values. The fact that binary feature data can be handled by essentially any inductive learner permits the comparison of a wide range of approaches. The binary-feature version of a typical learning algorithm is easier to explain: the mathematical notation is complicated enough without worrying about many-valued features. Perhaps most importantly, the common types of binary feature detectors are more difficult for a spammer to manipulate.

2.4.5.1 Neural Networks

Neural networks are systems composed of a large number of highly interconnected processing units. Connections between units have weights which can be modified. They are inspired by the physiology of biological nervous systems. Units in a neural network may be organized into layers (structured networks). A net can have one input layer, one output layer and any number of hidden layers. There are two basic types of networks: feed forward networks and feed backward networks. In feed forward network there are no cycles. However in feed backward networks, cycles are allowed. A typical feed-forward Artificial Neural Network is shown in the Figure 2-6.

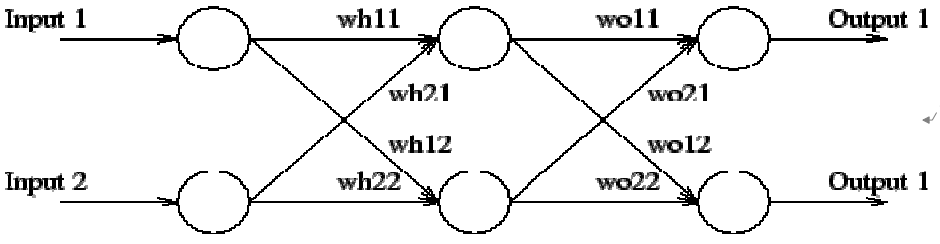


Figure 2-6~~ 2-2~~: An Artificial Neural Network

Formatted: Font: Not Bold

Formatted: Font: Not Italic, Font color: Red

Formatted: Font: Not Italic

Formatted: Font: Bold, Not Italic

Formatted: Space After: 0 pt

Formatted: Centered, Space After: 0 pt

Designating (I_1, I_2), (H_1, H_2) and (O_1, O_2) as the inputs, hidden-layer outputs and output-layer outputs respectively, the outputs of Hidden Node 1 and 2 are given in the Equations

$$H_1 = sgm\left(\sum_{i=1}^{2} I_i W_{i1}^h\right)$$

2-1 to 2-5.-

$H_1 = sgm\ (I_1wh11 + I_2wh21)$ Equation 2-1

and

$H_2 = sgm\ (I_1wh12 + I_2wh22)$ Equation 2-2

$$H_2 = sgm\left(\sum_{i=1}^{2} I_i W_{i2}^h\right)$$

where

$Sgm(x) = 1 / 1+e^{(-x)}$ Equation 2-3

$$sgm(x) = \frac{1}{1+e^{-x}}$$

The output-layer outputs are given by

$O_1 = sgm\ (H_1w011 + H_2w021)$ Equation 2-4

and

$O_2 = sgm\ (H_1w012 + H_2w022)$ Equation 2-5

$$O_1 = sgm\left(\sum_{m=1}^{2} H_m W_{m1}^o\right)$$

~~and~~

$$O_2 = sgm\left(\sum_{m=1}^{2} H_m W_{m2}^o\right)$$

Error is calculated as the difference between the actual output and the target. Error is propagated backward and the weights are changed.

A neural network is useless if it only sees one example of a matching input/output pair. It cannot infer the characteristics of the input data from only one example; rather, many examples are required. [8] This is analogous to a child learning the difference between

Formatted: Space After: 0 pt
Formatted: Font: 12 pt
Formatted: Font: 12 pt, Subscript
Formatted: Font: 12 pt
Formatted: Font: 12 pt, Subscript
Formatted: Font: 12 pt
Formatted: Font: 12 pt
Formatted: Font: 12 pt
Formatted: Font: 12 pt, Subscript
Formatted: Font: 12 pt
Formatted: Font: 12 pt
Formatted: Font: 12 pt
Formatted: Font: 12 pt
Formatted: Font: 12 pt
Formatted: Space After: 0 pt
Formatted: Font: Times New Roman, 12 pt
Formatted: Font: Times New Roman, 12 pt
Formatted: Font: Times New Roman, 12 pt
Formatted: Font: Times New Roman, 12 pt, Superscript
Formatted: Font: Times New Roman, 12 pt
Formatted: Caption, Line spacing: 1.5 lines
Formatted: Space After: 0 pt
Formatted: Font: 12 pt
Formatted: Font: 12 pt, Subscript
Formatted: Font: 12 pt
Formatted: Font: 12 pt
Formatted: Font: 12 pt
Formatted: Font: 12 pt
Formatted: Space After: 0 pt

(say) different types of animals - the child will need to see several examples of each to be able to classify an arbitrary animal. If they are to successfully classify birds (as distinct from fish, reptiles etc.) they will need to see examples of sparrows, ducks, pelicans and others so that they can work out the common characteristics which distinguish a bird from other animals (such as feathers, beaks and so forth). It is also unlikely that a child would remember these differences after seeing them only once - many repetitions may be required until the information `sinks in'.

It is the same with neural networks. The best training procedure is to compile a wide range of examples (for more complex problems, more examples are required) which exhibit all the different characteristics one is interested in. It is important to select examples which do not have major dominant features which are of no interest, but are common to the input data anyway. One famous example is of the US Army `Artificial Intelligence' tank classifier. It was shown examples of Soviet tanks from many different distances and angles on a bright sunny day, and examples of US tanks on a cloudy day. Needless to say it was great at classifying weather, but not so good at picking out enemy tanks. If possible, prior to training, some noise or other randomness to the example is added (such as a random scaling factor). This helps to account for noise and natural variability in real data, and tends to produce a more reliable network.

~~If~~If ~~you~~one is ~~are~~ using a standard un-scaled sigmoid node transfer function, ~~please note that the~~the desired output must never be set to exactly 0 or 1 [9]. The reason is simple: whatever the inputs, the outputs of the nodes in the hidden layer are restricted to between 0 and 1 (these values are the asymptotes of the function. To approach these values would require enormous weights and/or input values, and most importantly, they cannot be exceeded. By contrast, setting a desired output of (say) 0.9 allows the network to approach and ultimately reach this value from either side, or indeed to overshoot. This allows the network to converge relatively quickly. It is unlikely to ever converge if the desired outputs are set too high or too low. Once again, it cannot be overemphasized: a neural network is only as good as the training data! Poor training data inevitably leads to an unreliable and unpredictable network. Having selected an example, it is presented to the network and an output is generated.

A consequence of the back-propagation algorithm is that there are situations where it can get stuck [10]. Think of it as a marble dropped onto a steep road full of potholes. The potholes are local minima - they can trap the algorithm and prevent it from descending further. In the event that this happens, the network can be resized (add extra hidden-layer nodes or even remove some) or a different starting point can be tried (i.e. randomize the network again). Some enhancements to the BP algorithm have been developed to get around this - for example one approach adds a momentum term, which essentially makes the marble heavier - so it can escape from small potholes. Other approaches may use alternatives to the Mean Squared Error as a measure of how well the network is performing.

2.4.5.2 Bayesian Classification

A Naive Bayesian classifier is simply a Bayesian network applied to a classification task that assumes feature independence. Naive Bayesian Classifiers look at an amount of previous (or training) data. From this data, probabilities are determined about the likelihood of the event occurring in the future. The classifier learns from the data. Naive Bayesian Classifiers classify data by computing the a-posteriori probabilities. The class with the maximum a-posteriori probability is chosen as the class of the data. The a-posteriori probabilities are calculated using the Bayes Theorem given in Equation 2-6.

$$P(H|X) = P(X|H)\,P(H) / P(X) \qquad \text{Equation 2-6}$$

Naive Bayes classifier assumes that every attribute is independent from the other attributes. The Naive Bayes classifier proves an unexpected accuracy especially in text mining applications. However, the independence assumption is too strict and maybe relaxing this assumption could lead to more accurate classification. ~~Naïve Bayes models joint distribution of class and features and derives class probability by applying Bayes rule.~~

Formatted: Font: Not Italic
Formatted: Heading 4
Formatted: Font: 12 pt
Formatted: Font: 12 pt
Formatted: Font: 12 pt
Formatted: Font: 12 pt
Formatted: Space After: 0 pt

Chapter 3

3 Architectural Design

3.1 Design Overview

RUMMAGE has been designed to be a simple and modular architecture keeping the flow of control across the software. This modular architecture provides the capability to RUMMAGE to be extended easily in future for further enhancements. Each module is designed to be standalone in nature that provides a smooth interface for accessing its services. Any module can be extended or changed according to requirements without having significant effect on rest of the software.

3.2 Conceptual Model

This model of project provides an overview of the conceptually conceived functional requirements of the project and discusses communication interfaces between various modules of project. The Figure 3-1 represents top level conceptual model of RUMMAGE. Interfaces between various components are also shown.

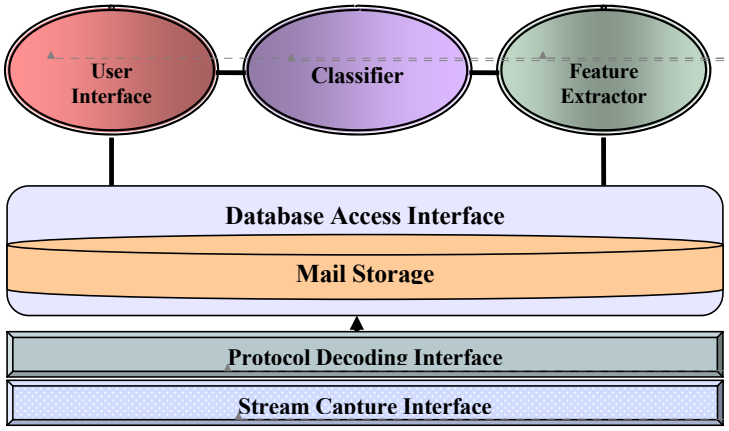


Figure 3-1-: Conceptual Design

At the heart of the model lies the mail storage layer. The database access Interface represents a high level interface to the database functionality implemented on the top of database programming API. This layer provides customized access to underlying database by restricting the individual modules to communicate to database directly. Three ovals on

Formatted: Heading 1
Formatted: Body Text,Body Text Char Char, Indent: Left: 0.25"
Formatted: Font: 10 pt
Formatted: Font: 10 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 12 pt
Formatted: Caption, Centered
Field Code Changed
Formatted: Body Text,Body Text Char Char

the top represent three processes running on the top of database access layer. The user interface provides the view of database tables by accessing the database access interface. It uses the classification mechanism to for analysis purpose. The classification module performs analysis of data by accessing the data through feature extractor which transforms the text documents into feature vectors before passing then on to classifier. The stream capture module communicates with the protocol decoding engine through a consistent interface after it has reassembled some stream. Protocol decoding module communicates with the database access layer through its own interface which facilitates storage of disintegrated email components in database.

The conceptual model only depicts a functional model but does not specify underlying implementation details for the project. In the model described in Figure 3-1, all of the project may be deployed on single station or all of the modules on separate computers to meet the functional requirements. A more reasonable approach would be to run traffic capture engine on a separate computer for optimized data sniffing.

3.3 Distinct Architectural Components

These consist of the email capture engine which captures email data packets from internet traffic and reconstructs email sessions; email decoding engine which decodes email protocol streams to parse captured emails (MIME decoding is done to obtain attachments); database and access interface where captured emails are logged and are accessed for further analysis; feature extractor whose function is to convert the email text into a weighted feature vector that can be used by the classifier; and the classifier which intelligently classifies email data based on training carried out~~provided~~ by using the information provided by the feature extractor.

3.4 Detailed Architectural Design Model

This model of project provides a detailed overview of functionality of project, identifying the distinct components and specifying interaction patterns between them. The Figure 3-2 represents a detailed architecture of RUMMAGE. It provides an insight into the implementation model of the project.

Formatted: Font: Times New Roman
Formatted: Font: Times New Roman
Formatted: Font: Times New Roman
Formatted: Font: Times New Roman
Formatted: Font: Times New Roman
Formatted: Font: Times New Roman
Formatted: Font: Times New Roman
Formatted: Font: Times New Roman
Formatted: Font: Times New Roman
Formatted: Font: Times New Roman
Formatted: Font: Times New Roman

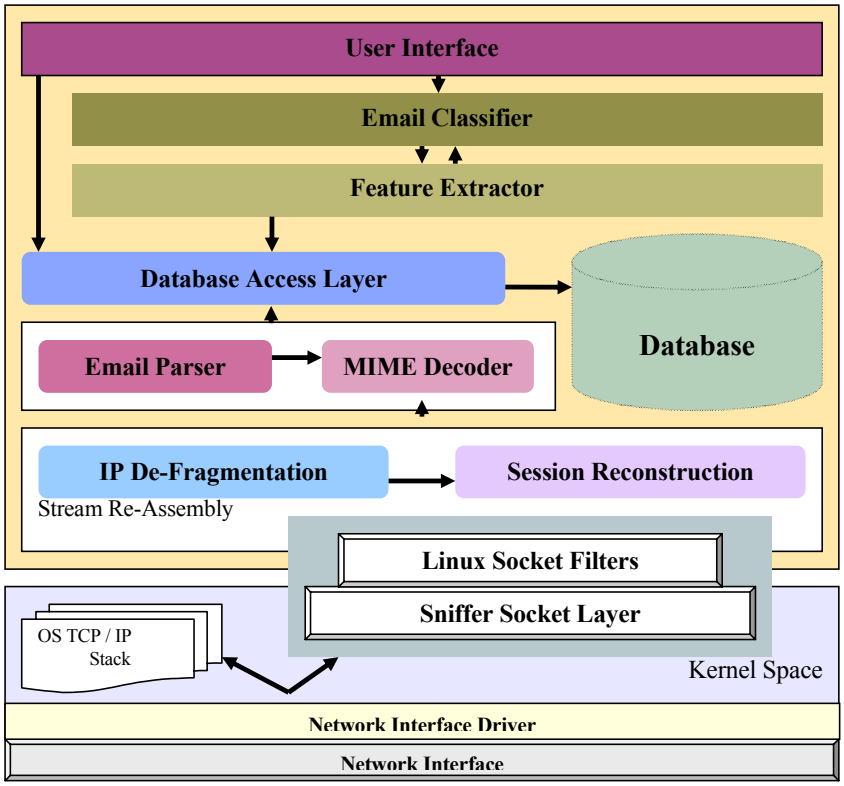


Figure 3-2-: Detailed Architectural Design

As obvious from the Figure 3-2 RUMMAGE represents a highly modular and a simple architecture. This modularity leads to a neat implementation and enables the project to be extended in future easily. As soon as network interface card driver gets data from the interface it passes it on to TCP/IP protocol stack. The Linux systems provide a functionality to retrieve a copy of this data from NIC driver through Linux Packet Sockets. Packet sockets are mechanism provided by Linux kernel that copies packets directly from NIC driver to user space without the intervention of kernel layer. Each module has a clearly defined interface through which its functionality can be accessed.

3.5 Implementation platform

RUMMAGE is developed in C and C++ on Linux OS keeping in view efficiency constraints. Linux OS provides extensive networking capabilities. It provides low level

Formatted: Body Text,Body Text Char Char, Centered, Indent: Left: 0.25"

Formatted: Caption, Centered, Line spacing: 1.5 lines

Field Code Changed

Formatted: Body Text,Body Text Char Char

network access through its rich network APIs. The Linux inter-process communication mechanism has been proved to be the most efficient. More over Linux is an open source operating system which provides enormous configuration capabilities. Also the source code of Linux is available for reference purposes. C being the most efficient language in use today is extensively used in low level hardware and network access. There are rich C libraries available for network manipulation and inter process communication. In order to provide modular architecture to RUMMAGE C++ proved to be a great help.

Chapter 4

4 Email Capture Engine

4.1 Introduction

Traffic capture lies at the core functionality of passive network traffic monitoring systems. Therefore Email Capture Engine enables RUMMAGE to capture email traffic for analysis. The Email Capture Engine is provided with a copy of network traffic coming from ***mirror port*** and it separates out the email data discarding rest of the traffic. It then reconstructs email sessions from fragmented data to obtain email streams. Figure 4-1 shows the position of email capture engine in the design hierarchy. The email capture engine gets its input from the backbone in the form of mirrored Ethernet frames and sends protocol streams to email parser as its output.

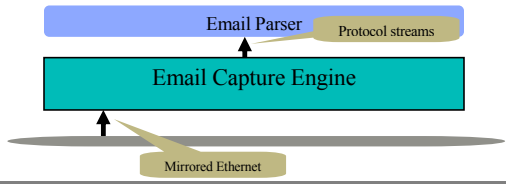


Figure 4-1-: Email Capture Engine Stand Point

4.2 Objectives

Email Capture Engine provides email protocol streams to RUMMAGE. The key objectives for the engine consist of capturing of network traffic at data link layer, discarding non IP traffic; reassembly of fragmented IP datagrams; protocol based filtering of assembled IP packets to keep only email traffic and TCP session reconstruction to get email protocol streams. The reconstructed email sessions are in the form of client/server request/reply commands. By parsing and verifying these commands, email protocols are decoded.

4.3 Design Considerations

Email Capture Engine assumes distinct importance in the context of RUMMAGE. Being the source of data for RUMMAGE, much of the performance of the project depends on it. The design requirement for the module places prime importance on use of efficient and effective techniques in order to minimize packet loss and fruitful stream reassembly. Being

Formatted: Space After: 0 pt
Formatted: Heading 1
Formatted: Font: Times New Roman
Formatted: Line spacing: 1.5 lines
Formatted: Font: Times New Roman
Formatted: Font: Times New Roman
Formatted: Font: Times New Roman
Formatted: Font: Times New Roman
Formatted: Keep with next
Formatted: Caption
Formatted: Font: 15 pt
Formatted: Heading 2
Formatted: Line spacing: 1.5 lines
Formatted: Heading 2
Formatted: Line spacing: 1.5 lines

implemented in user space, it also raises consideration for efficient use of memory. Moreover the module needs an architecture that can be extended easily. To make the design simple and modular is a issue in the implementation of Email Capture Engine.

4.4 Architectural Breakdown and Implementation

The functionality of Email Capture Engine can be divided into to major sub-modules; Packet capture sub-module and Reassembly sub-module. Packet capture sub-module has two components, Packet Sniffer and Packet Filter. While Reassembly sub-module comprises of three components that are Internet Datagram Reassembly, Datagram Filtering and TCP Session Reassembly. Figure 4-2 shows the architecture breakdown of Email Capture Engine.

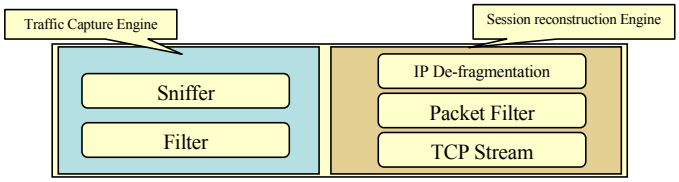


Figure 4-2: Email Capture Engine Architecture Breakdown

4.4.1 Packet Capture Engine

This module gets copied traffic from internet/network backbone through ***mirror port***. Only data packets that meet filter test are passed on to ***Reassembly Module***; others are discarded. Figure 4-3 shows the position of Packet Capture Engine in design hierarchy.

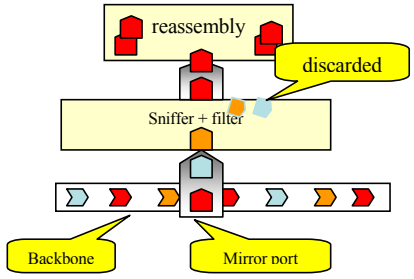


Figure 4-3: Packet Capture Engine

Formatted: Heading 2, No bullets or numbering
Formatted: Line spacing: 1.5 lines
Formatted: Caption
Formatted: Font: 15 pt
Formatted: Heading 3
Formatted: Line spacing: 1.5 lines
Formatted: Font: Times New Roman
Formatted: Font: Times New Roman
Formatted: Caption, No bullets or numbering

Packet sniffer is a piece of software that captures raw data packets from network interface driver before they are passed on to protocol stack in operating system, by kernel. Packet sniffers use packet filtering mechanism to discard the unwanted packets before handing them over to users. Packet sniffers and filters are implemented using special kind of sockets provided by networking API of operating system.

Formatted: Line spacing: 1.5 lines

4.4.1.1 Linux Networking APIs

Formatted: Heading 4, No bullets or numbering

Formatted: Line spacing: 1.5 lines

Linux Network APIs provide with sufficient functionality for low level network access. In order to implement packet sniffers and filters they provide Packet sockets and Linux Socket filters.

Linux Socket filters (LSF) are mechanism provided by modern Linux kernels to filter packets at socket level by installing the packet filter code into the socket. Due to their deployment at network interface driver level, these socket filters are able to effectively filter only those Internet datagrams that arrive at un-fragmented. In fragmented datagrams no information of higher layer headers like TCP is available. This information can only be extracted once the fragments are reassembled to construct the complete internet datagram. Therefore installing application layer protocol based filter code will discard all fragmented packets except first fragment. While performing application layer protocol analysis Linux Socket Filters are of significance only for filtering non-IP traffic. For application based filtering effective internet datagram reassembly must be done.

Passive monitoring of applications layer protocols requires that traffic flowing on network be captured for analysis. There is an inherent feature of Ethernet NICs that they, by default, discard all the network traffic that is not addressed to their MAC address. This default behavior must be overridden for effective capture of traffic. Linux kernel provides the functionality to set network interface cards in promiscuous mode through socket option provided with networking APIs.

Linux Packet Sockets are much detailed in their implementation and require comprehensive knowledge of underlying functionality for effective usage. There may be portability issues when migrating from one kind of OS architecture to other. Packet

Capture Library provides a portable and easy to handle interface to sniffing and filtering capabilities of UNIX and Linux systems. On Berkeley derived Linux distributions LibPcap uses Berkeley packet filter mechanism, a more efficient cousin of LSF. Berkeley packet filters, although ported to many UNIX architectures, do have operational limitations on Linux systems. LibPcap uses LSF where BPF do not work.

4.4.2 Reassembly Engine

The reassembly sub-module gets partially filtered IP traffic from Packet Capture Module and performs three tasks; it performs Internet datagram de-fragmentation if necessary, it performs packet filtering to keep only Email traffic (other packets are discarded) and performs TCP session reassembly. Figure 4-4 shows the architecture of Reassembly Engine.

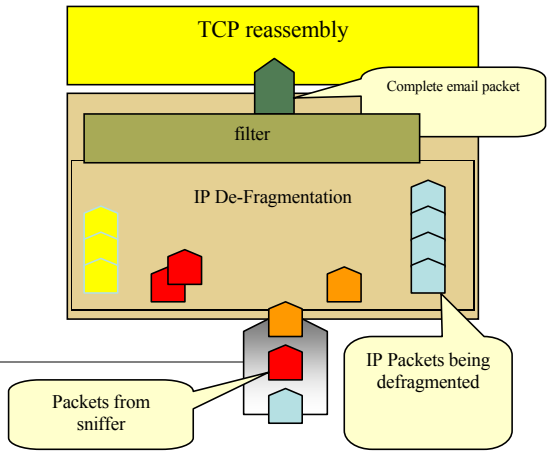


Figure 4-4: Reassembly Engine

4.4.2.1 Internet Datagram Reassembly

Internet datagram reassembly is a procedure of concatenating fragmented datagrams to obtain complete IP packets. Internet header provides the necessary information to identify the siblings and reconstruct the internet datagram. IETF standard for Internet Protocol [6]

Formatted: Heading 3
Formatted: Font: Not Italic
Formatted: Line spacing: 1.5 lines
Formatted: Line spacing: 1.5 lines
Formatted: Border: Box: (Single solid line, Auto, 0.5 pt Line width)
Formatted: Caption, Line spacing: 1.5 lines
Formatted: Font: Not Italic
Formatted: Heading 4
Formatted: Line spacing: 1.5 lines

and IETF draft for IP datagram reassembly algorithms [11] discuss in sufficient detail, the Internet datagram reassembly mechanism.

~~IP Reassembly algorithm~~

Presented in Figure 4-5 is the IP reassembly algorithm used in Reassembly sub-module along with the notation used in the algorithm.

Notation:

——FO - Fragment Offset
——IHL - Internet Header Length
——MF - More Fragments flag
——TTL - Time To Live
——NFB - Number of Fragment Blocks
——TL - Total Length
——TDL - Total Data Length
——BUFID - Buffer Identifier
——RCVBT - Fragment Received Bit Table
——TLB - Timer Lower Bound

Procedure:

```
(1)  BUFID <- source|destination|protocol|identification;
(2)  IF FO = 0 AND MF = 0
(3)     THEN IF buffer with BUFID is allocated
(4)             THEN flush all reassembly for this BUFID;
(5)          Submit datagram to next step; DONE.
(6)     ELSE IF no buffer with BUFID is allocated
(7)             THEN allocate reassembly resources
                     with BUFID;
                     TIMER <- TLB; TDL <- 0;
(8)          put data from fragment into data buffer with
               BUFID from octet FO*8 to
                           octet (TL-(IHL*4))+FO*8;
(9)          set RCVBT bits from FO
                           to FO+((TL-(IHL*4)+7)/8);
(10)         IF MF = 0 THEN TDL <- TL-(IHL*4)+(FO*8)
(11)         IF FO = 0 THEN put header in header buffer
(12)         IF TDL # 0
(13)           AND all RCVBT bits from 0
```

Formatted: Font: (Default) Times New Roman, 12 pt, Bold
Formatted: Indent: First line: 0"
Formatted: Font: (Default) Times New Roman, 12 pt
Formatted: Font: (Default) Times New Roman, 12 pt
Formatted: Line spacing: 1.5 lines, Border: Right: (Single solid line, Auto, 0.5 pt Line width, From text: 2 pt Border spacing:)
Formatted: Font: (Default) Times New Roman, 12 pt
Formatted: Font: (Default) Times New Roman, 12 pt, Bold
Formatted: Font: (Default) Times New Roman, 12 pt
Formatted: Border: Top: (Single solid line, Auto, 0.5 pt Line width), Bottom: (Single solid line, Auto, 0.5 pt Line width), Left: (Single solid line, Auto, 0.5 pt Line width), Right: (Single solid line, Auto, 0.5 pt Line width, From text: 2 pt Border spacing:)

```
                                to (TDL+7)/8 are set
(14)          THEN TL <- TDL+(IHL*4)
(15)               Submit datagram to next step;
(16)               free all reassembly resources
                  for this BUFID; DONE.
(17)        TIMER <- MAX(TIMER,TTL);
(18)        give up until next fragment or timer expires;
(19) timer expires: flush all reassembly with this BUFID; DONE.
```

Figure 4-5: IP Reassembly Algorithm

Formatted: Caption

4.4.2.2 TCP Stream Reassembly

Formatted: Font: Not Italic

Formatted: Line spacing: 1.5 lines

TCP stream reassembly is a procedure of converting IP datagram into effective connection oriented communication. TCP connections are uniquely identified on the basis of a data structure called transmission control block TCB. IP and TCP headers provide the necessary fields that compose a TCB. TCB is comprised of source and destination IP address and source and destination ports. The combination of these four fields remains unique throughout the connection lifetime. Sequence numbers play a distinct role in arranging the fragments in correct order. IETF draft for Transmission Control Protocol [7] discusses in sufficient details the connection specific details which provide basis for reassembly of TCP streams.

5 Email Parsing and Storage

5.1 Introduction

Most email servers conduct email services by running two separate processes on the same machine. One process is the POP3 server, which holds emails in a queue and delivers emails to the client when they are requested. There is the SMTP server that receives outgoing emails from clients and sends and receives emails from other SMTP servers. These two processes are linked by an internal mail delivery mechanism that moves mail between the POP3 and SMTP servers.

5.2 Protocol Decoding

5.2.1 SMTP

The SMTP design is based on a simple model of communication. As the result of a user mail request, the sender-SMTP establishes a two-way transmission channel to a receiver SMTP. The receiver-SMTP may be either the ultimate destination or an intermediate. SMTP commands are generated by the sender-SMTP and sent to the receiver-SMTP. SMTP replies are sent from the receiver-SMTP to the sender-SMTP in response to the commands.

Once the transmission channel is established, the SMTP-sender sends a MAIL command indicating the sender of the mail. If the SMTP-receiver can accept mail it responds with an OK reply. The SMTP-sender then sends a RCPT command identifying a recipient of the mail. If the SMTP-receiver can accept mail for that recipient it responds with an OK reply; if not, it responds with a reply rejecting that recipient (but not the whole mail transaction). The SMTP-sender and SMTP-receiver may negotiate several recipients. When the recipients have been negotiated the SMTP-sender sends the mail data, terminating with a special sequence. If the SMTP-receiver successfully processes the mail data it responds with an OK reply. The dialog is purposely lock-step, one-at-a-time.

5.2.1.1 Example of the SMTP Procedure

This SMTP example shows mail sent by Smith at host Alpha.ARPA, to Jones, Green, and Brown at host Beta.ARPA. It is assumed that host Alpha contacts host Beta directly.

S: MAIL FROM:<Smith@Alpha.ARPA>

Formatted: Heading 1, Widow/Orphan control, Adjust space between Latin and Asian text, Adjust space between Asian text and numbers

Formatted: Body Text,Body Text Char Char, Indent: First line: 0", Widow/Orphan control, Adjust space between Latin and Asian text, Adjust space between Asian text and numbers

Formatted: Font: Not Italic

Formatted: Heading 4, Widow/Orphan control, Adjust space between Latin and Asian text, Adjust space between Asian text and numbers

Formatted: Body Text,Body Text Char Char, Indent: First line: 0", Widow/Orphan control, Adjust space between Latin and Asian text, Adjust space between Asian text and numbers

R: 250 OK
S: RCPT TO:<Jones@Beta.ARPA>
R: 250 OK
S: RCPT TO:<Green@Beta.ARPA>
R: 550 No such user here
S: RCPT TO:<Brown@Beta.ARPA>
R: 250 OK
S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Blah blah blah...
S: ...etc. etc. etc.
S: <CRLF>.<CRLF>
R: 250 OK

The mail has now been accepted for Jones and Brown. Green did not have a mailbox at host Beta.

5.2.2 POP

Initially, the server host starts the POP3 service by listening on TCP port 110. When a client host wishes to make use of the service, it establishes a TCP connection with the server host. When the connection is established, the POP3 server sends a greeting. The client and POP3 server then exchange commands and responses (respectively) until the connection is closed or aborted. Responses in the POP3 consist of a status indicator and a keyword possibly followed by additional information. There are currently two status indicators: positive ("+OK") and negative ("-ERR"). Servers MUST send the "+OK" and "-ERR" in upper case.

A POP3 session progresses through a number of states during its lifetime. Once the TCP connection has been opened and the POP3 server has sent the greeting, the session enters the AUTHORIZATION state. In this state, the client must identify itself to the POP3 server. Once the client has successfully done this, the server acquires resources associated with the client's maildrop, and the session enters the TRANSACTION state. In this state, the client requests actions on the part of the POP3 server. When the client has issued the QUIT command, the session enters the UPDATE state. In this state, the POP3 server releases any resources acquired during the TRANSACTION state and says goodbye. The TCP connection is then closed.

5.2.3 MIME

5.2.3.1 Message Parsing

The email message parser receives its input form the protocol decoder. The decoder splits the various mails according to the protocol employed i.e. SMPT, POP etc. Each mail is scanned line by line to recognize the parts namely the email header, email text and the email attachments. Email header contains information like the sender and recipient addresses, subject of the message and date on which the message is sent. The content type field determines the type of information contained in the message. Values of all these header fields are extracted by scanning every line of the message and finding the values assigned to the various parameters. If the content type is text/plain, it is simply written to the text file containing the message body. However, if content-type=”multipart/mixed”, then further checks need to be performed. If it is a multipart body and the content disposition field has the value: “inline”, no further processing of the message is required. If content-disposition is: “attachment”, the attached message is further analyzed to determine the type of encoding used. Each part in mail message starts with specific boundary information and attributes which help to identify each part and their encoding. The attached message is then decoded accordingly by the MIME decoder.

5.2.3.2 MIME Decoder

It decodes the encoded message received from the parsing module. Decoding techniques may be used to decode base64 and quoted-printables. Figure 5-1 shows the mime decoder.

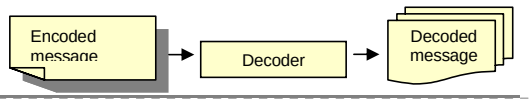


Figure 5-1: MIME Decoder

Base64 encoding takes three bytes, each consisting of eight bits, and represents them as four printable characters in the ASCII standard. It does that in essentially two steps. The first step is to convert three bytes to four numbers of six bits. Each character in the ASCII standard consists of seven bits. Base64 only uses 6 bits (corresponding to 2^6 = 64 characters) to ensure encoded data is printable and humanly readable. None of the special characters available in ASCII are used. The 64 characters (hence the name Base64) are 10 digits, 26 lowercase characters, 26 uppercase characters as well as '+' and '/'. If, for

Formatted: Heading 3, Widow/Orphan control, Adjust space between Latin and Asian text, Adjust space between Asian text and numbers
Formatted: Font: Not Italic
Formatted: Font: Not Italic
Formatted: Font: Not Italic
Formatted: Hyperlink, Left, Line spacing: 1.5 lines
Formatted: Font: Bold,
Formatted: Caption, Centered
Formatted: Hyperlink, Left, Line spacing: 1.5 lines
Formatted: Default Paragraph Font, Font: (Default) Times New Roman
Formatted: Default Paragraph Font, Font: (Default) Times New Roman

example, the three bytes are 155, 162 and 233, the corresponding (and frightening) bit stream is 100110111010001011101001, which in turn corresponds to the 6-bit values 38, 58, 11 and 41. These numbers are converted to ASCII characters in the second step using the Base64 encoding table. The 6-bit values of the example translate to the ASCII sequence "m6Lp".

1

55 -> 10011011

1

62 -> 10100010

2

33 -> 11101001

1

00110 -> 38

1

11010 -> 58

0

01011 -> 11

1

01001 -> 41

3

8 -> m

5

8 -> 6

1

1 -> L

4

1 -> p

This two-step process is applied to the whole sequence of bytes that are encoded. To ensure the encoded data can be properly printed and does not exceed any mail server's line length

Formatted: Default Paragraph Font, Font: (Default) Times New Roman

Formatted: Default Paragraph Font, Font: (Default) Times New Roman

Formatted: Hyperlink, Left, Line spacing: 1.5 lines, No bullets or numbering

Formatted: Hyperlink, Left, Line spacing: 1.5 lines

limit, newline characters are inserted to keep line lengths below 76 characters. The newline characters are encoded like all other data.

At the end of the encoding process a problem may arise. If the size of the original data in bytes is a multiple of three, everything works fine. If it is not, one may end up with one or two 8-bit bytes. For proper encoding, exactly three bytes are required. The solution is to append enough bytes with a value of '0' to create a 3-byte group. Two such values are appended if there is one extra byte of data; one is appended for two extra bytes. Of course, these artificial trailing '0's cannot be encoded using the encoding table. They must be represented by a 65th character. The Base64 padding character is '='. Naturally, it can only ever appear at the end of encoded data. For example, the slogan,

"Man is distinguished, not only by his reason, but by this singular passion from other animals, which is a lust of the mind, that by a perseverance of delight in the continued and indefatigable generation of knowledge, exceeds the short vehemence of any carnal pleasure."

Encoded in base64 is as follows:

TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0aGlzIHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1c3Qgb2YgdGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udGludWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRoZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS4=

The Quoted-Printable encoding is intended to represent data that largely consists of octets that correspond to printable characters in the ASCII character set. It encodes the data in such a way that the resulting octets are unlikely to be modified by mail transport. If the data being encoded are mostly ASCII text, the encoded form of the data mains largely recognizable by humans. A body which is entirely ASCII may also be encoded in Quoted-Printable to ensure the integrity of the data should the message pass through a character-translating, and/or line-wrapping gateway.

Formatted: Default Paragraph Font, Font: (Default) Times New Roman

5.3 Implementation

Any 8-bit byte value may be encoded with 3 characters, an "=" followed by two hexadecimal digits (0-9 or A-F) representing the byte's numeric value. For example, a US-ASCII form feed character (decimal value 12) can be represented by "=0C", and a US-ASCII equal sign (decimal value 61) is represented by "=3D". All characters except printable ASCII characters or end of line characters must be encoded in this fashion. Printable ASCII characters except "=", i.e. those with decimal values between 33 and 126 excepting decimal value 61, may be represented by themselves.

ASCII tab and space characters, decimal values 9 and 32, may be represented by themselves except if these characters appear at the end of a line. If one of these characters appears at the end of a line it must be encoded as "=09" (tab) or "=20" (space). End of line characters, a carriage return (decimal value 13) followed by a line feed (decimal value 10), receive special treatment. If this pair of characters is used in the input format to mean end of line, as is common for many text formats, this pair of characters must be represented by themselves. If the input format is not text, or uses a different end of line indicator these characters must be encoded as "=0D" and "=0A" respectively. Lines of quoted-printable encoded data must not be longer than 76 characters. To satisfy this requirement *soft line breaks* may be added as desired. A soft line break consists of an "=" at the end of an encoded line.

5.3.1 Database

There are two types of databases: flat file and relational. A flat file database contains all the information one might need in one datasheet or table. A relational database, however, consists of many different tables linked together by 'key' (common) fields. The data can be manipulated in many different ways to produce different results. MySQL is classified as a Relational Database Management System. The reliability and longevity of the C programming language coupled with the stability of MySQL make them a strong pair for system database management. The C APIs are distributed with the MySQL source and included in the *mysqlclient* library. They are used to connect to a database and execute a query. There are some examples in the *clients* directory of the MySQL source code. The MySQL API uses a MYSQL data structure (defined in mysql.h) to establish a connection

Formatted: Heading 2, Left

Formatted: Hyperlink, Left, Line spacing: 1.5 lines

Formatted: Default Paragraph Font, Font: (Default) Times New Roman

Formatted: Default Paragraph Font, Font: (Default) Times New Roman

Formatted: Font: (Default) Times New Roman

Formatted: Heading 3

Formatted: Font: (Default) Times New Roman

Formatted: Body Text,Body Text Char Char, Left

Formatted: Font: (Default) Times New Roman

with the database engine. The first step is to connect to the database. Either of *mysql_connect* or m*ysql_real_connect* could be used. Commands for creating the database and tables are issued if they don't exist; else data is inserted into the table. Some queries of most common use are also used. These include the returning all the records in the table, deleting a specific record from the table and accessing the attached messages.

After a successful query, if data is to be returned to the client, the result set must be transferred via either the mysql_store_result or mysql_use_result functions. Both of these functions store the result set in a MYSQL_RES structure. The difference is that mysql_store_result reads the entire result set into memory on the client, where mysql_use_result instructs the client to retrieve a row dynamically from the server with each call to mysql_fetch_row. However mysql_use_result ties up server resources and thus should not be used for interactive applications where user actions are often unpredictable and could result in extended delays.

Data retrieved from the result set with mysql_fetch_row is placed into a MYSQL_ROW structure, which is simply an array of pointers to the beginning of each field. A query is an object that provides a custom view of data from one or more tables. Queries are a way of asking questions of the data stored in the tables. RUMMAGE defines queries to select, update insert or delete data.

Formatted: Hyperlink, Left, Line spacing: 1.5 lines

Formatted: Hyperlink, Line spacing: 1.5 lines

Chapter 6

6 Feature Extractor~~ion Implementation~~

——Introduction

6.1

Feature extractor accesses data base and retrieves text messages. It takes raw text documents from the ~~database which are~~database which is emails messages in this case. At the end of the phase of feature extractor, feature vectors are ~~generadted~~generated which become the input of the classifier as ~~shwon~~shown in the Figure 6-1. Feature vectors are saved in the weight files.

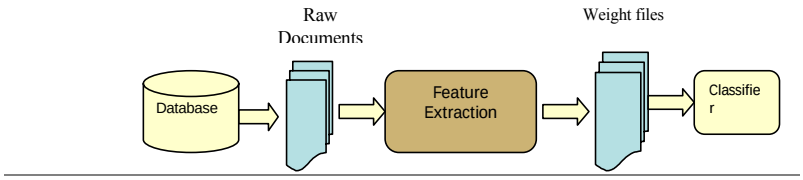
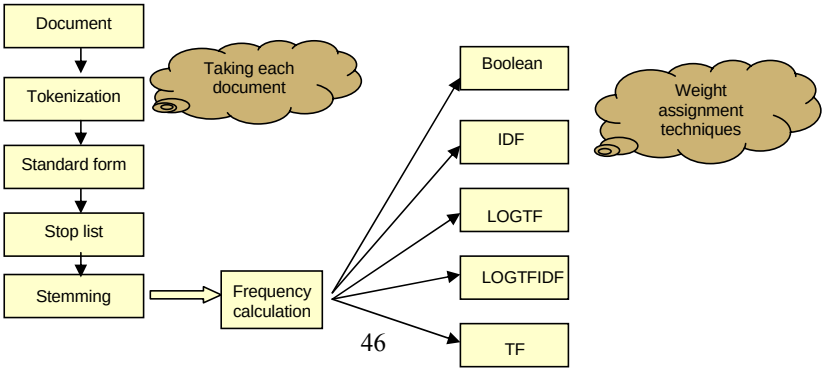


Figure 6-1: Architecture Breakdown

Figure 6-2 gives a detailed overview of feature extractor components and its pipelining. A text document is first passed through a tokenizer that generates appropriate tokens. The tokens are brought back to the standard form. A stop list of words is applied to remove unimportant dictionary words. The next filter is a stemmer that reduces many redundant words to root prefixes and reduces dictionary size. The most important function of weight assignment is frequency calculation that is performed before weight formulae as shown in Figure 6-2.



Formatted: Heading 1

Formatted: Body Text,Body Text Char Char, Left

Formatted: Caption

Formatted: Caption, Line spacing: 1.5 lines

~~Architecture Breakdown~~

~~Pre Processing in Feature extraction~~

Figure 6-2: Preprocessing in Feature Extractor

6.2 Aim of Filters

The aim of passing the words of a given document through these filters is to bring the words into a program readable form ,remove all those words of English language that are irrelevant to text analysis and classification, remove punctuation that can confuse the program. Words are brought back to the root forms so that many words with same meaning can be processed only once, removed redundant strings at various stages. These filters increases the efficiency of the Feature Extractor during the analysis phase by reducing the words strings to a set that holds high information content for being selected as a features on which analysis will be performed. The filters are pipe lined in the same order as described to achieve the optimum results for targeted efficiency and correctness of the feature extractor [12]. The words then will be transferred to the weight vectors during weight assignment phase and weights assigned to them according to amount of information held by them for classification phase.

6.3 Standard Form Filter

After the words are loaded into the buffer the are passed through this filter to bring all the words into the standard form remove capitalization so that words like 'That' and 'that' can be treated the same. This is what a simple reader can see but digging deep into the problems of data mining the problem is not that simple. There are not just capital and small letters; in the computers ASCII codes and set of available symbols are also used to represent a character. The possibilities are diverse and can confuse the classifier by representing a single English language word by a hundred ways. The module is

Formatted: Heading 2

Formatted: Font: 12 pt, Not Bold

Formatted: Font: Not Bold

Formatted: Body Text,Body Text Char Char, Centered

Formatted: Heading 2

implemented to replaces 32 possible ways of writing alphabet 'a' like '@','á', 'â', 'ä', 'å', 'ã', 'Ă', 'ă', 'Ą', 'Ǻ', 'ǻ', using different symbols provided by ASCII set called Latin-I, Latin extension-A, Latin extension-B etc. similarly 12 substitutes for 'o' ,16 for 'n' and so on. A two dimensional 26 X n matrix is formed to cater for 26 alphabets with its numerous possible substitutes. To emphasize the importance of the filter the number of ways by which a simple word "apple" can be represented is calculated. It gives a figure of 32x7x7x9x17=239904 ways. It is a huge figure and each of these words will appear to be a different string to the analyzer confusing the text analysis process. But as mentioned it is a possibility that a word can be represented by this many representation giving the worst case. This type of character substitutions are intentionally carried out by spammers so this particular filter is targeted at spammer's activity. The filter brings all of these to one standard form—making it possible to be analyzed by text analyzer in the next phase.

6.4 Stop List Filter

The aim of passing the words through this filter is to remove all those words of English language that are irrelevant to text analysis and classification e.g., very common English language words like 'this', 'that', 'was', 'were', 'a', 'the' etc do not play any role in distinguishing a document as belonging to some category. It is because the frequency of these words is same in all documents. First of all a data base of these most common English words based on language research organizations are developed. Many such data bases have been integrated to form the stop list dictionary.

6.5 Removal of Redundant Characters

This particular filter is targeted at spammers activity of putting redundant characters to confuse the classifier. Such redundancy of characters is removed by automaton implementation e.g., a word "looooooser" will be converted into loser. Different techniques of adding redundant characters were studied and all such characters were removed. Research have shown that one of the spammers activity is to degrade the performance of spam filters to induce such redundancy which results in total failure of text analysis phase.

6.6 Stemming to the Root Forms

In most cases, morphological variants of words have similar semantic interpretations and ~~e~~can be considered as equivalent for the purpose of IR applications. For this reason, a number of so-called *stemming Algorithms*, or *stemmers*, have been developed, which attempt to reduce a word to its *stem* or root form. Thus, the key terms of a query or document are represented by stems rather than by the original words. This not only means that different variants of a term can be *conflated* to a single representative form – it also reduces the *dictionary size*, that is, the number of distinct terms needed for representing a set of documents. A smaller dictionary size results in a saving of storage space and processing time. Well known stemming algorithms are used and integrated as filters in the module as theses Krovetz Stemmer Paice /Husk Stemmer Porter Stemmer.

6.7 Post Processing-Weight Assignment

6.7.1 ~~Frequecy~~Frequency Calculation

Weight assignment ~~techniques~~techniques uses different variables among which one basic variable is frequency of individual tokens ~~occuring~~occurring in the whole data set. This frequency calculation is done by implementing an efficient algorithm described in the Figure 6-3 that utilizes least amount of memory and processing time by giving out exact word count in a single scan of the document. LnkLst is a 2D array of pointers, where each of its index value is specified by an alphabet from 'a' to 'z' dimension of LnkLst is therefore 26x26. ~~e~~Each element in LnkLst points to a unique linked list, i.e., there are 676 (26x26) linked lists, ~~intially~~initially NULL, and pointed to by a particular combination of two alphabets.

~~Algorithm~~

loop - till end of file

Formatted: Body Text,Body Text Char Char, Left

Formatted: Heading 2

Formatted: Body Text,Body Text Char Char, Border: Right: (Single solid line, Auto, 0.5 pt Line width)

Formatted: Body Text,Body Text Char Char, Border: Top: (Single solid line, Auto, 0.5 pt Line width), Bottom: (Single solid line, Auto, 0.5 pt Line width), Left: (Single solid line, Auto, 0.5 pt Line width), Right: (Single solid line, Auto, 0.5 pt Line width)

Figure 6-3: Weight Assignment Algorithm

Formatted: Caption, Centered, Line spacing: 1.5 lines

6.7.2 Weight Assignment Formulae

Formatted: Heading 3

Selected features must be associated with a numerical value to evaluate the impact of the feature to the classification problem. Most types of feature weighting schemes in text categorization are borrowed from the field of information retrieval. The most frequently used weight techniques are given in Equation 6-1 through Equation 6-6.

-Weight assignment with- document frequency and terminal document frequency

$$IDF = \log(D/dff) \qquad \text{Equation 6-1}$$

Weight through calculation of terminal frequency local and global occurrence

Formatted: Body Text,Body Text Char Char

$$TF = tffd \qquad \text{Equation 6-2}$$

Formatted: Body Text,Body Text Char Char, Left

Weight assignment integrating previous two techniques

Formatted: Body Text,Body Text Char Char

$$TFIDF = tffd\log(D/dff) \qquad \text{Equation 6-3}$$

Weight assignment technique with standard equation

Formatted: Body Text,Body Text Char Char

LOGTFIDF = log(tffd +0.5)log (D/dff) Equation 6-4

Weight assignment with Normalized terminal frequency

LOGTF = log(tffd) Equation 6-5

BINARY weight assignment technique, Weight assignment through normalization of frequency

NF = F/Wc*Nf Equation 6-6

where 'fd' is the weight of feature 'f' in document 'd', 'tffd' the occurrence frequency of feature 'f' in document 'd', 'D' the total number of documents in the training set, and 'dff' is the number of documents containing the feature 'f'. Different types of feature and weighting schemes are used. The feature types include single tokens, single stemmed tokens, and phrases. Weighting schemes will include binary feature, term frequency (TF), TFIDF, log TFIDF, etc.

Figure 6-4 shows the flow chart of featur

~~Flow Chart Feature extractor~~

e extractor. First of all, the document is tokenized i.e. converted to tokens. This is followed by conversion of the tokens in the standard form unless they are already in this form. All such tokens are removed which also appear in the stop list. A check is made that whether stemming is required. In case it is required, stemming is carried out otherwise weight assignment phase is carried out.

Formatted: Body Text,Body Text Char Char, Left

Formatted: Body Text,Body Text Char Char

Formatted: Body Text,Body Text Char Char

Formatted: Body Text,Body Text Char Char

Formatted: Font: Not Bold

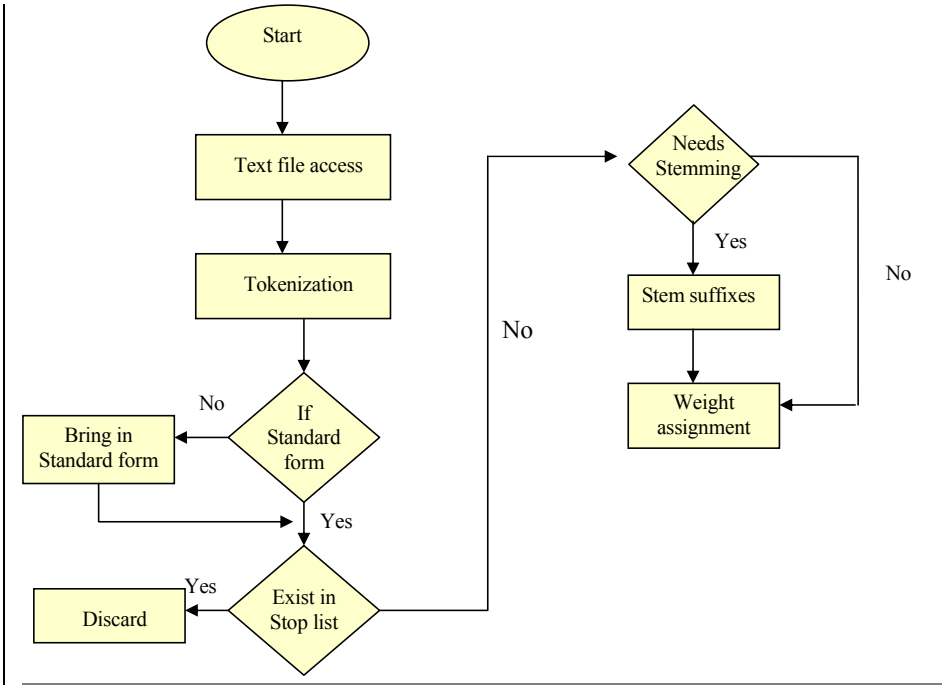


Figure 6-4: Flow Chart of Feature Extractor

Formatted: Caption, Centered, Line spacing: 1.5 lines

Formatted: Font: 15 pt

~~Analysis and results~~

~~Precision~~

~~Recall~~

~~Formula used~~

	~~Correct as Ham~~	~~Correct as Spam~~
~~Classified as Ham~~	~~a~~	~~b~~
~~Classified as Spam~~	~~c~~	~~d~~

~~Spam Precision = d / c+d~~

~~Spam Recall = d / b+d~~

~~Ham Precision = a / a+b~~

~~Ham Recall = a / a+c~~

~~Accuracy = a+d / N~~

~~Error = 1 - Accuracy~~

~~Naïve Bayesian used with LOGTF~~

~~Train Data: 1257 Emails (723 Ham & 534 Spam)~~

~~Test (Unseen) Data: 244 Emails (149 Ham & 95 Spam)~~

~~Spam Precision = 100%~~

~~Spam Recall = 87.4% (No of false -ve = 12)~~

~~Ham Precision = 92.54%~~

~~Ham Recall = 100% (No of false +ve = 0)~~

~~Accuracy = 95.08%~~

Formatted: Heading 2

Formatted: Font: 12 pt, Bold

Formatted: Heading 2, No bullets or numbering, Widow/Orphan control, Adjust space between Latin and Asian text, Adjust space between Asian text and numbers

Formatted: Heading 2

~~Naïve-Bayesian used with TF~~

~~Train Data: 1257 Emails (723 Ham & 534 Spam)~~

~~Test (Unseen) Data: 245 Emails (149 Ham & 96 Spam)~~

~~Spam Precision = 100%~~

~~Spam Recall = 90.6% (No of false -ve = 9)~~

~~Ham Precision = 94.3%~~

~~Ham Recall = 100% (No of false +ve = 0)~~

~~Accuracy = 96.33%~~

~~Naïve-Bayesian used with IDF~~

~~Train Data: 1257 Emails (723 Ham & 534 Spam)~~

~~Test (Unseen) Data: 245 Emails (149 Ham & 96 Spam)~~

~~Spam Precision = 67.22%~~

~~Spam Recall = 83.33% (No of false -ve = 39)~~

~~Ham Precision = 87.30%~~

~~Ham Recall = 73.82% (No of false +ve = 16)~~

~~Accuracy = 77.55%~~

~~Naïve-Bayesian used with LOGTFIDF~~

~~Train Data: 1257 Emails (723 Ham & 534 Spam)~~

~~Test (Unseen) Data: 245 Emails (149 Ham & 96 Spam)~~

~~Spam Precision = 80.01 %~~

~~Spam Recall = 83.33% (No of false -ve =19)~~

Formatted: Heading 2, No bullets or numbering, Widow/Orphan control, Adjust space between Latin and Asian text, Adjust space between Asian text and numbers

Formatted: Heading 2

Formatted: Heading 2, No bullets or numbering, Widow/Orphan control, Adjust space between Latin and Asian text, Adjust space between Asian text and numbers

Formatted: Heading 2

Formatted: Heading 2, No bullets or numbering, Widow/Orphan control, Adjust space between Latin and Asian text, Adjust space between Asian text and numbers

~~Ham Precision = 89.04%~~

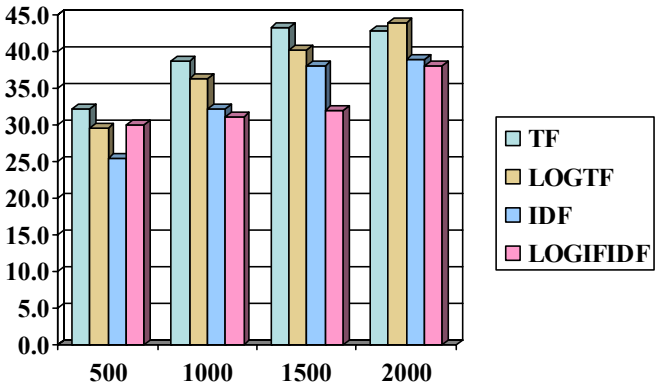
~~Ham Recall = 87.24% (No of false +ve =16)~~

~~Accuracy = 85.71%~~

Formatted: Heading 2

~~Graphical Analysis~~

Formatted: Font: 12 pt, Bold



Formatted: Heading 2, Indent: Left: 0"

Formatted: Heading 2

6.8 Conclusions

Formatted: Heading 2, Indent: Left: 0"

Formatted: Font: Not Bold

The stop lists' best adjusted length was 400 words. LOGTF and TF ~~performs~~performed best among the weight assignment techniques implemented with all data sets. The classification accuracy for these techniques was greater than 90%. Dropping of tokens with single occurrence improved the performance of classifier, both in terms of classification time and accuracy by almost 100%. Almost all weight assignment techniques gave an improved performance with increase in size of training data set.

7 Email Classifier

7.1 Introduction

While electronic mail has undoubtedly become a success over the last ten years, it is not without its problems. The problems range from an enormous increase of unsolicited e-mail messages, better known as "spam" to viruses, worms and Trojans. It all started 10 years ago, on the 5 March1994, as the first spam message was sent to the Usenet newsgroups. A law firm Canter and Siegel advertised their services for the U.S. Green Card lottery. This was the beginning of a flood of spam messages which effectively destroyed the usefulness of newsgroups. The next step for spammers was the individual e-mail addresses. Collecting these has become much easier with the rise of the Internet where people offer their contact details. Today's amount of spam is estimated to almost 50 percent of all electronic messages, and is about to grow to 70 percent in the year 2007. The amount that unwanted commercial email cost U.S. corporations in 2002 (Ferris Research) estimates to 8,900,000,000 U.S. dollars.

7.2 Objectives

The major objectives of email classifier consisted of the automated classification of emails into specified categories, implementation of best available Artificial Intelligence (AI) techniques, comparison of the implemented techniques and to choose the best one and achievement of best possible accuracy. Another objective was to use minimum amount of memory so that there is less overhead.

7.3 Conceptual Design

The email classifier gets weight files as input from the feature extractor. They are preprocessed (sorted) and normalized. Only the most important features are selected for training. Back-propagation algorithm was used for training. During the classification phase, the email classifier gets a single document (or multiple documents) for classification. It creates tokens for the document if it is not already tokenized. This is followed by searching of tokens in the list of features used during training. The classifier classifies the document

in one of the categories defined during training. The design of email classifier is shown in the Figure 7-1.

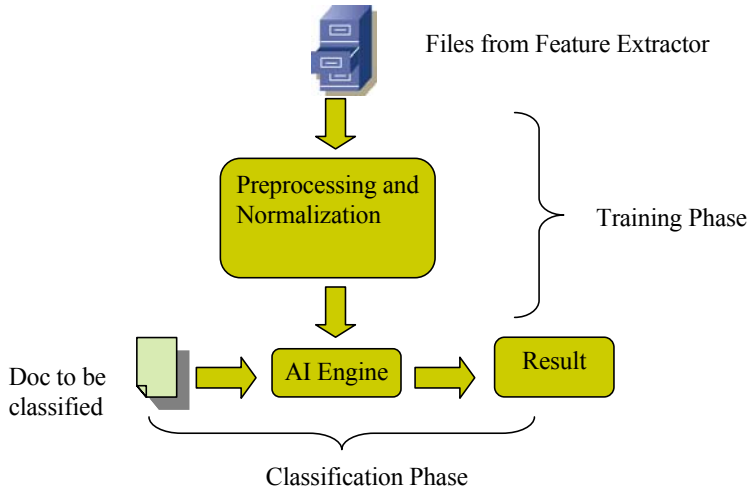


Figure 7-1 ~~13~~ 1: Conceptual Design of Email Classifier

7.4 Classifiers Implemented

Two approaches were employed for the intelligent classification of emails namely Artificial Neural Networks (ANN) and Naive Bayesian algorithm. ANN was chosen because the classification efficiency is very good, classification is faster and it can be used to solve many non-linear problems such as pattern recognition and text classification. Naive Bayesian has the advantage of having best accuracy among other classifiers. Besides this, it is comparatively easier to implement. However the time required for classification is much greater as compared to other possible techniques.

7.4.1 Design Considerations for ANN

ANN was used along with many different variations such as varying the number of hidden neurons, momentum, learning rate, number of input neurons (features), activation functions and initial weights [13]. All of the parameters were varied and the classification accuracy and the overhead were noted. Those values were used for which the accuracy was the best and the overhead was minimum.

Formatted: Space After: 0 pt

Formatted: Heading 3

Formatted: Space After: 0 pt

7.4.1.1 Number of Neurons in Hidden Layer

The neurons in the hidden layer were increased from 2 to 20. Figure 7-2 shows that as the number of hidden units (neurons) is increased from 2 to 10, there is a sharp increase in the performance of the ANN but if this is increased further, there is hardly any decrease in the number of iterations. Increasing the number of hidden neurons increases the overhead because of increase in the number of weights. There is no definite number of hidden units with which the number of iterations would be minimum.

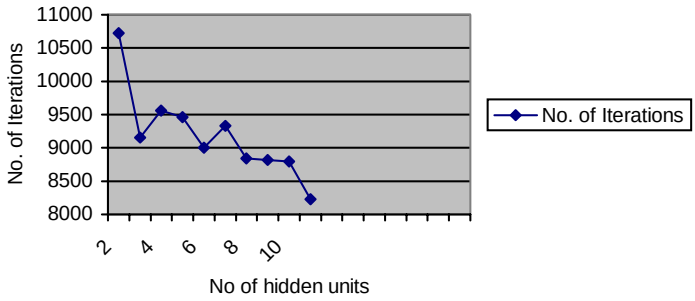


Figure 7-2 ~~23 2~~: Effect of Varying the Number of Hidden Units on the Performance of ANN

7.4.1.2 Momentum (α)

Momentum was varied from 0.05 to 1.0. Figure 7-3 shows the relationship between the momentum (α) of an ANN and the number of iterations required to train the network. As α was increased from 0.05, the required numbers of iterations were reduced to a large extent. Gradual change in the number of iterations was observed in the beginning which was followed by an abrupt change. When momentum was 0.05, the number of iterations was 13000. But increasing momentum to 0.9 reduced the number of iterations below 3000.

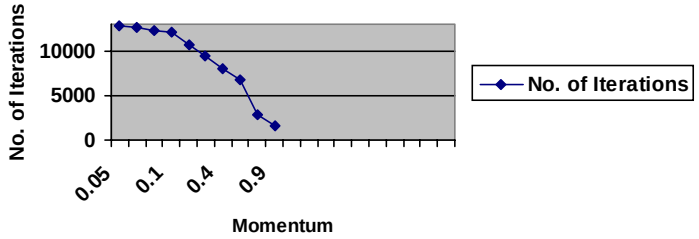


Figure 7-3 ~~33 3~~: Effect of Changing Momentum on the Performance of ANN

Formatted: Font: Not Italic
Formatted: Heading 4
Formatted: Font: Not Italic
Formatted: Indent: Left: 0", Space After: 0 pt
Formatted: Space After: 0 pt
Formatted: Caption, Indent: Left: 0.81", Hanging: 0.81"
Formatted: Font: Not Italic
Formatted: Heading 4
Formatted: Font: Not Bold, Not Italic
Formatted: Font: Not Italic
Formatted: Font: Not Bold, Not Italic
Formatted: Body Text,Body Text Char Char, Indent: Left: 0"
Formatted: Font: Times New Roman, 12 pt
Formatted: Font: Times New Roman, 12 pt
Formatted: Font: Times New Roman, 12 pt
Formatted: Font: Times New Roman, 12 pt
Formatted: Font: Times New Roman, 12 pt
Formatted: Font: Times New Roman, 12 pt
Formatted: Font: Times New Roman, 12 pt
Formatted: Font: Times New Roman, 12 pt
Formatted: Caption, Indent: Left: 1.26", Hanging: 0.81"

Comparing the momentum only with the number of iterations doesn't give the true picture. The number of misclassifications must also be taken to account. Figure 7-4 shows that as the momentum is varied from 0.05 to 0.9, the no. of misclassifications increase and there is a rapid increase beyond momentum of 0.5. If a very small value is chosen for the momentum then the number of iterations required for training becomes very large. Hence the best range for momentum was found to be 0.3 to 0.5.

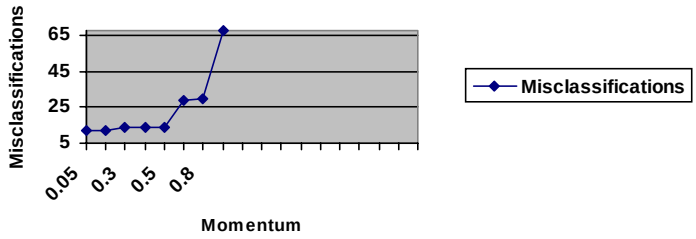


Figure 7-4~~3 4~~: Number of Misclassifications vs. Momentum

7.4.1.3 Learning Rate (η)

Figure 7-5 shows that as the learning rate is increased from 0.2 onwards, the numbers of iterations are reduced considerably. However, increasing the learning rate beyond a certain limit deteriorates the performance of the ANN and the required number of iterations increases in a quick manner. If the learning rate is made too large, the weights are updated by large amounts and the algorithm may never converge. This is evident from the Figure 7-5 which shows that increasing the learning rate beyond the value of 29 rapidly increases the required number of iterations for training of ANN.

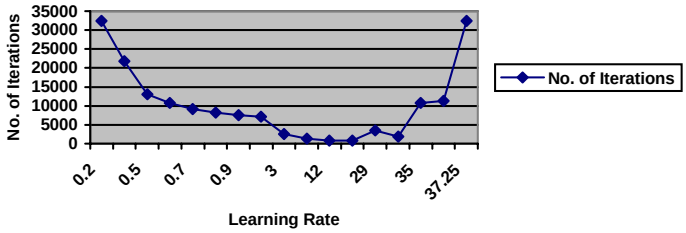


Figure 7-5~~3 5~~: Effect of Changing Learning Rate on the Performance of ANN

Formatted: Indent: Left: 0", Space After: 0 pt

Formatted: Space After: 0 pt

Formatted: Caption, Centered, Indent: Left: 0.69", Hanging: 0.67"

Formatted: Font: Not Italic

Formatted: Heading 4

Formatted: Space After: 0 pt

Formatted: Caption, Indent: Left: 1.04", Hanging: 0.8"

7.4.1.4 No. of Input Neurons (Features)

The number of input neurons which was equal to number of input features was varied from 20 to 400. The best results i.e. maximum accuracy of 90% was observed when the number of input neurons was set to 100. If too many features are used, a lot of features from the different categories match and the classifier cannot work properly. Reducing the number of features is also called feature pruning or dimension reduction [14]. Similarly selecting too few features proved to be of no use.

7.4.1.5 No. of Hidden Layers

The number of hidden layers was increased from 1 to 3. There was not much increase in the classification accuracy of the email classifier when hidden layers were increased. However the time taken to classify increased a lot due to the increase in the number of weights and other related overhead. ANN with 1 hidden layer was almost 33% faster as compared to ANN with 2 hidden layers. The best results were obtained when only one hidden layer was used.

7.4.1.6 Choice of Activation Function

Two activations functions were tested for artificial neural network namely tanh(x) and $1/1+e^{(-x)}$ (Also known as sigmoid or logistic function). They are differentiable through out the range of input values and their derivatives can be expressed in terms of their output. Both these functions performed equally well. Other non-sigmoid functions may be used as long as they are differentiable.

7.4.1.7 Weight Initialization

The performance of ANN greatly depends upon the weight initialization. If proper and small values are used, it is trained quickly and the classification accuracy is high. Otherwise it may not even reach the global minimum error and training does not converge. If the synaptic weights are assigned large initial values, it is highly likely that the neurons in the network will be driven into saturation. The weights were initialized with small random values.

Formatted: Font: Not Italic
Formatted: Heading 4
Formatted: Space After: 0 pt
Formatted: Font: Not Italic
Formatted: Heading 4
Formatted: Font: Not Italic
Formatted: Font: Not Italic
Formatted: Space After: 0 pt
Formatted: Font: Not Italic
Formatted: Heading 4
Formatted: Font: Not Italic
Formatted: Font: Not Italic
Formatted: Space After: 0 pt
Formatted: Superscript
Formatted: Font: Not Italic
Formatted: Heading 4
Formatted: Space After: 0 pt

~~1.6.4~~7.4.2 Flowchart of ANN

Figure 7-6 shows the flowchart for ANN which is composed of two phases i.e. training phase and the classification phase. The training phase begins with the decision about the number of input and hidden neurons, and the number of hidden layers. Basic parameters are initialized and input feature vectors are preprocessed (sorted). Only the most important features are used for training. This is followed by the application of back-propagation algorithm. The final weights are saved at the end of the training phase. A single document (or multiple documents) is fed to ANN which is classified using the already saved weights.

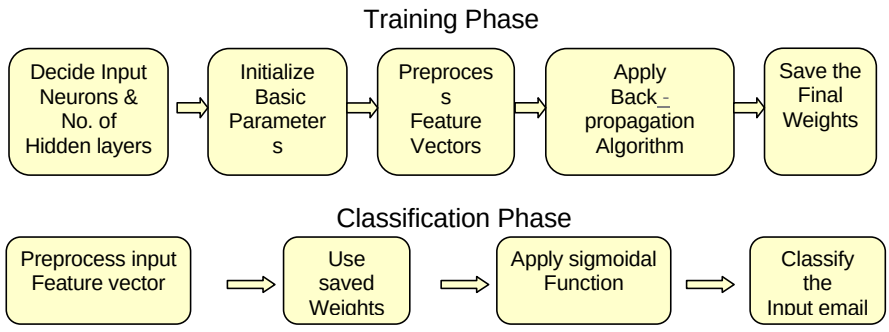


Figure 7-6~~Figure 63-6~~: Flowchart for Artificial Neural Network

~~1.6.5~~7.4.3 Implementation of ANN

When the feature vectors are input to the ANN for training, it sorts them and uses only the most important of them which is indicated by their weights. The features are also normalized in the range of 0-1 because by doing so the efficiency of the classifier is greatly increased. The algorithm for ANN is shown in Figure 7-7. It is composed of two procedures i.e. one for training and one for classification. The language used for the implementation of ANN was C and the C compiler used was gcc. The code was run successfully on both MS Windows and Linux (Slackware and RedHat) operating systems.

Initialize basic parameters of ANN namely learning rate, momentum, number of hidden layers, number of neurons in input and hidden layers. Initialize the weights with small values

Procedure Train_ANN (Feature Vectors)
Sort the feature vectors, normalize and choose the best one
Save the features chosen
Employ Error-Back propagation learning rule
For (number of epochs)
Forward Computation
Compute the output of every unit in the network using the sigmoid function:
$f(x) = 1 / 1+\exp^{(-x)}$
Calculate the error i.e. the difference b/w the actual output and the target:
$E(w) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$ (LMS rule)
Backward Computation
Back-propagate the errors through the network and update the weights
End
End Procedure
Procedure Classify_ANN (Feature Vector of a single email)
Use the important features used during training
Use the weights learned during training
For (Each feature of input email)
Search in the file of important features
If there is a match, set that input
End
Apply sigmoidal function at each unit to get the final output
Classify the email
End Procedure

Figure 7-7: Algorithm for Training and Classification Phases in ANN

~~1.6.6~~7.4.4 mplementation of Naive Bayes Algorithm

To implement a learning email filter using the Naives Bayesian classifier, a chosen tokenizer setting and some sort of feature selection model is executed on a repository of email pre-classified as "junk" or "legitimate." Probabilities or weights of tokens being in a legitimate or a junk email are calculated. Different settings for tokenizing email messages and selecting features for classification are what most often vary from filter to filter. The Naive Bayesian algorithm is given in the Figure 7-8. It compares all of the tokens for a given email with the list of tokens created after the training phase. If there is a match, it multiplies the weights to get resultant weight. Finally this weight is multiplied with the probability of email within this classification. A token can consist of an alphanumeric word, non-alphanumeric characters, phrases, etc. contained in the subject, body, header,

Formatted: Heading 3
Formatted: Bullets and Numbering
Formatted: Font: (Default) Times New Roman
Formatted: Space After: 0 pt
Formatted: Font: (Default) Times New Roman
Formatted: Font: (Default) Times New Roman
Formatted: Font: (Default) Times New Roman
Formatted: Font: (Default) Times New Roman

etc. The trick is to find the combination of tokenize settings that produce the most desired results. For email features, it is inherent that a feature set of independent, unordered tokens will be huge. Due to this, some sort of feature selection cutoff is employed and also differs from application to application as well. Efficient searching techniques were implemented for the search of email tokens in the weight files. The language used was C and the C compiler used was gcc. The code was run successfully on both MS Windows and Linux (Slackware and RedHat) operating systems.

Procedure Classify (Feature Vector of a single email)
Get tokens from the feature vector
For (Each target classification)
Search the tokens in the weights' file (for this target classification)
If there is a match
Multiply the weight with resultant weight
Multiply the resultant weight with the probability of document within this classification
End
Assign that category to the input email which has the largest final weight
End Procedure

~~Figure~~Figure 7-8: Naive Bayes Algorithm

~~**Analysis and Table 2-1: Classification types**~~

~~:~~

~~**The training set consisted of 1257 emails downloaded from LingSpam; of which 723 were ham and the remaining 534 were spam. The test set consisted of 245 emails composed of 149 ham and 96 spam emails.**~~

~~**Table 2-2 shows the results for ANN when one hidden layer was used and learning rate was set to 0.22 and momentum to 0.3. As the number of input features was increased from 20 to 100, a sharp increase in the accuracy was**~~

Formatted: Font: (Default) Times New Roman

Formatted: Space After: 0 pt, Border: Left: (Single solid line, Auto, 0.5 pt Line width), Right: (Single solid line, Auto, 0.5 pt Line width)

Formatted: Heading 3, Don't keep with next

Formatted: Heading 3

Formatted: Heading 3, Don't keep with next

seen. However increasing that further deteriorated the performance.Table 2-2: Results for ANN when 1 hidden layer was used

* means

Table 2-3 shows the results for ANN when the number of hidden layers was increased to 2 and learning rate was still set to 0.22 and momentum to 0.3. As the number of input features was increased from 100 to 200, a sharp decrease in the classification accuracy was seen. Table 2-3: Results for ANN when 2 hidden layers were usedThe classification accuracy of Naive Bayes, used with TF technique of feature extractor was observed to be 96% including spam precision and ham recall of 100%. The number of false negatives (number of spam emails classified as ham) was just 9 and hence the spam recall was 90%.

Figure 2-7 and figure 2-8 show that the Naive Bayes algorithm is almost 30 to 100 times slower than ANN. ANN having 1 hidden layer is faster than ANN having 2 hidden layers.

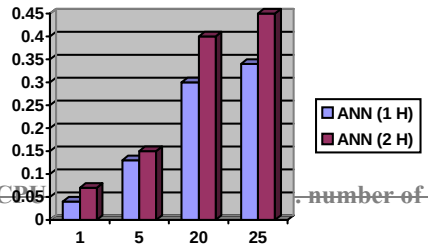


Figure 73-9: CPU [illegible]. number of examples for 1 and 2 layers ANN

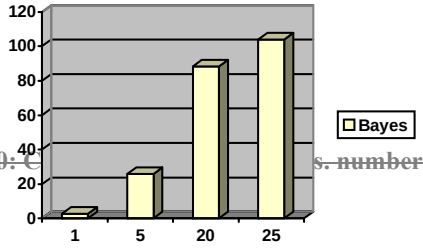


Figure 83-10: C[illegible]s. number of examples for Naive Bayes

Formatted: Heading 3

7.4.5 Testing of ANN

Boundary conditions were applied to the neural network in boundary testing. It was successfully carried out in two phases. In the first phase, all the inputs meant for spam features were set to the maximum value of '1' while the inputs for ham features were made '0'. The output indicated the input as being spam. This was followed by the second phase in which all the inputs for ham features were made '1' and other inputs remained at '0'. The output in this case was ham.

Chapter 8

8 Performance Analysis

The classification rate of Naive Bayes algorithm was much better than ANN. However it took much longer to classify emails using Naive Bayes algorithm. ANN classified the emails in a matter of milliseconds. The various parameters for analysis are shown in the Table 8-1. Equations 8-1 through 8-6 give different formulae used for analysis of Email Classifiers [15].

Table 8-1: Parameters for Analysis

	Correct as Ham	Correct as Spam
Classified as Ham	a	b
Classified as Spam	c	d

Spam Precision = d / c+d Equation 8-1

Spam Precision is the ratio of spam emails correctly classified and all of the emails classified as spam.

Spam Recall = d / b+d Equation 8-2

Spam Recall is the ratio of spam emails correctly classified and all of actual spam emails.

Ham Precision = a / a+b Equation 8-3

Ham Recall = a / a+c Equation 8-4

Ham Precision is the ratio of ham emails correctly classified and all of the emails classified as ham. Similarly Ham Recall is the ratio of ham emails correctly classified and all of the actual ham emails.

Accuracy = a+d / N Equation 8-5

Error = 1 – Accuracy Equation 8-6

Accuracy is defined as the number of emails correctly classified.

8.1 Artificial Neural Network

The training set consisted of 1257 emails downloaded from LingSpam; of which 723 were ham and the remaining 534 were spam. The test set only for ANN consisted of 197 emails composed of 149 ham and 48 spam emails. Table 8-2 shows the results for ANN when one

Formatted: Space After: 0 pt
Formatted: Heading 1
Formatted: Centered, Line spacing: 1.5 lines, Don't keep with next
Formatted: Centered, Line spacing: single
Formatted: Centered, Line spacing: single
Formatted: Centered, Line spacing: single
Formatted: Heading 2

hidden layer was used and learning rate was set to 0.22 and momentum to 0.3. As the number of input features was increased from 20 to 100, a sharp increase in the accuracy was seen. For all networks, all of the original spam was classified correctly (no false negative). Hence spam recall and ham precision is 100%. The best network consisted of 100 input neurons and 40 hidden neurons and the classification accuracy was 90.86%. However increasing that further deteriorated the performance.

Table 8-2: Results for ANN with 1 Hidden Layer

Formatted: Left, Line spacing: 1.5 lines, Don't keep with next

Input Neurons	Hidden Neurons	Spam (Total 48)	Ham (Total 149)	Spam Precision	Accuracy
20	10	48	120	62.34%	85.3%
100	40	48	131	72.73%	90.86%
200	10	48	93	46.15%	71.57%
200	20	48	96	47.52%	73.10%
200	30	48	114	57.83%	82.23%
200	40	48	100	49.48%	75.13%

Table 8-3 shows the results for ANN when the number of hidden layers was increased to 2 and learning rate was still set to 0.22 and momentum to 0.3. As the number of input features was increased from 100 to 200, a sharp decrease in the classification accuracy was seen i.e. from 90% to 66%. The overall accuracy figures were not as good as was in the case when 1 hidden layer was used.

Table 8-3: Results for ANN with 2 Hidden Layers

Formatted: Left, Indent: Left: 1.28", Line spacing: 1.5 lines

Input Neurons	Hidden Neurons (Layer 1)	Hidden Neurons (Layer 2)	Spam (Total 48)	Ham (Total 149)	Accuracy
100	25	35	33	106	71%
100	40	80	27	101	65%
100	50	100	48	131	90%
200	50	94	28	102	66%
200	50	95	44	111	79%

8.2 Naive Bayes

Formatted: Heading 2

The training set consisted of 1257 emails downloaded from LingSpam [16]; of which 723 were ham and the remaining 534 were spam. The test set was composed of 245 emails of which 149 were ham and 95 were spam emails. It was tested with the TF, LOGTF, IDF and LOGTFIDF. The percentages for spam and ham precision and recall along with the overall accuracy are shown in the Table 8-4.

Table 8-4: Different Weight Assignment Techniques and their Classification Accuracies

Weight Assignment Techniques	Spam Precision	Spam Recall	Ham Precision	Ham Recall	Accuracy
LOGTF	100%	87.4%	92.54%	100%	95.08%
TF	100%	90.6%	94.3%	100%	96.33%
IDF	67.22%	83.33%	87.30%	73.82%	77.55%
LOGTFIDF	80.01%	83.33%	89.04%	87.24%	85.71%

ANN with 2 hidden layers was found to much slower as compared to ANN with 1 hidden layer. This behavior is shown in the Figure 8-1. The number of emails for classification was increased from 1 to 25. For 20 emails; ANN with 1 hidden layer took about 0.3 seconds to classify and ANN with 2 hidden layers took almost 0.4 seconds to classify which is about 33% higher. This trend prevailed for all variations in the number of emails.

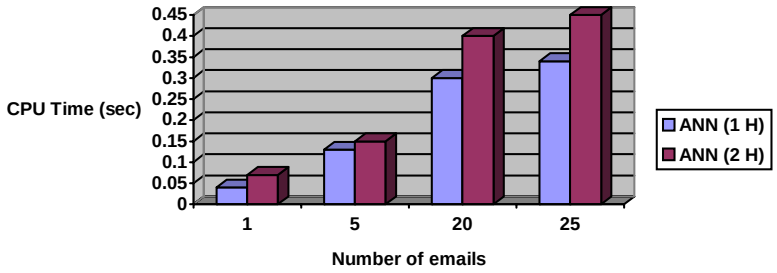


Figure 8-1: CPU User time (seconds) vs. Number of Examples for 1 and 2 Layers ANN

Naive Bayes has got good classification accuracy as compared to ANN but is much slower. Figure 8-2 shows that the Naive Bayes algorithm is almost 30 to 100 times slower than ANN. The number of emails for classification was increased from 1 to 25 as was the case for ANN. Naive Bayes algorithm classified 5 emails in about 20 seconds which ANN with 1 hidden layer had classified in only 0.3 seconds. Hence Naive Bayes was 60 times slower as compared to ANN with 1 hidden layer. The number of emails was increased subsequently to 25. Naive Bayes took about 100 seconds to classify these emails.

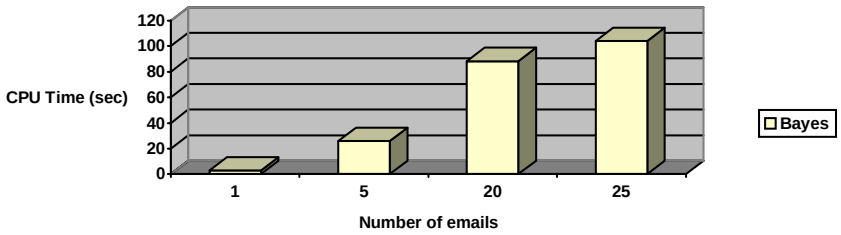


Figure 8-2: CPU User time (seconds) vs. Number of Examples for Naive Bayes

Formatted: Left, Indent: Hanging: 0.83", Line spacing: 1.5 lines

8.3 Comparison of Weight Assignment Techniques

Formatted: Heading 2

The classification accuracy of Naive Bayes, used with TF technique of feature extractor was observed to be 96% including spam precision and ham recall of 100%. The number of false negatives (number of spam emails classified as ham) was just 9 and hence the spam recall was 90%. However the classification accuracy of Naive Bayes with other techniques namely LOGTF, IDF and LOGIFIDF was not much high as shown in Figure 8-3.

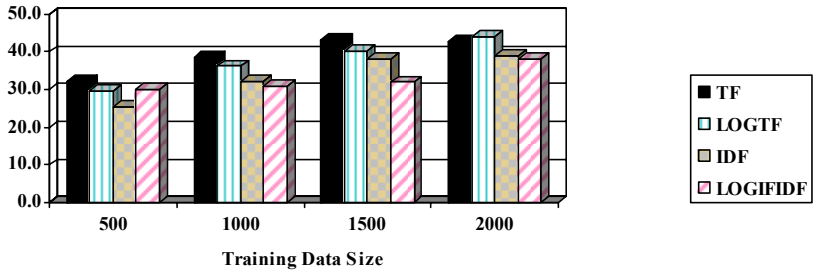


Figure 8-3: Comparison of Different Techniques

Formatted: Caption, Centered

It also shows that as the training size increases (from 500 documents to 2000 documents), the accuracy also increases. In overall, the classification accuracy of Naive Bayes (96%) was found to be better as compared to ANN (90%).

Formatted: Centered

Chapter 9

Formatted: Space After: 0 pt

~~2~~9 Summary and Future Work

~~2.1~~9.1 Summary

Passive systems have been used widely by network administrators to monitor their networks over the years to ensure stability and efficient working of their networks. As network attacks and security breaches have increased in recent years as well as due to growth in the use of internet user preferences have changed along with network traffic, these systems have come to embed AI techniques into them, to automate the network monitoring. In RUMMAGE this idea is extended to email systems. The system passively captures email traffic and performs intelligent analysis on it to produce useful results. Being intelligent system it can be configured and trained by administrators according to their preferences.

RUMMAGE can be considered to be comprised of three major portions which are distinguished on the basis of type of development, they deal with. These are: Email data retrieval, Email parsing and storage; and Analysis of captured email data.

Email Data Retrieval portion captures email traffic from network and reconstructs email sessions. As RUMMAGE is a passive system; it mirrors the internet traffic from network possibly through mirror port. This traffic is sniffed while putting network card into Promiscuous Mode; which enables the NIC to capture even those packets not destined for it. Captured data is then filtered to get IP packets only; other data is discarded. To achieve efficiency, filtering must be done at lowest possible level in operating system. Linux provides a mechanism to achieve this task in the form of Linux Socket Filters (LSF). This filtering mechanism is embedded into Packet Sockets provided by Linux kernel, which discards non IP traffic without copying it into kernel space of operating system.

The IP packets are usually fragmented into smaller ones to cope with packet size limitations on local networks as well as the TCP fragments. The filtered IP traffic is then passed through an IP de-fragmentation process followed by TCP session reassembly. Reconstructed IP packets are again filtered to get email packets only before TCP reassembly process; which provides with complete email messages in original form. Once

email messages are retrieved, they must be parsed to be disintegrated into basic elements of an email message in order to get useful information from them; thereafter message parts must be saved in permanent storage for further analysis to be done on it. This is precisely done during Email Parsing and Storage.

An email message can be divided into four basic parts; email envelope, email header, message text and the attachment (if any). Envelope holds the routing information of email whereas header has the information like sender, receiver, CC/BCC, subject etc. The attachment can be of various forms as supported by MIME standard. As email protocols are character based, the mail attachments must be encoded in some suitable format in order to be converted into ASCII characters. To retrieve the original mail attachment it must be decoded into original form. This is achieved during parsing. MySQL database system is used to store and manage email parts obtained during parsing.

Now that email data is ready to be processed, intelligent data mining techniques are applied to it. The objective of analysis phase is to classify the email message under consideration according to criterion set during training phase, using email/text classification techniques. As the result of whole of the effort depends on correct classification training phase is of particular importance in this analysis as well as the classification.

The analysis phase includes the process of feature extraction; the process during which useful information is retrieved from an email message which is used for training and for classification of a message. The features (possibly words) which are crucial for a message to belong to a certain category are extracted during training phase and weighted accordingly. These weighted features are then used to train the classifier for that category and during the classification are searched through the message to be classified. The information gathered during feature extraction is input to a classifier which rates the document accordingly. Classification engine is implemented using Neural Networks and Naive Bayesian classification. The performance of both techniques is compared.

Analysis of communication domains is a complex task and absorbs a lot of technological details. Communication domains are diverse in nature and much effort is needed for information retrieval from such complex systems. RUMMAGE has given a humble beginning with many milestones to be achieved in this domain.

2.29.2 Future Work

With the advent of information technology the Information Retrieval system are becoming to be more and more absorbing. They have come a long way beyond traditional data analysis. Where RUMMAGE currently target non-encrypted text, the system can be enhanced to handle much more complex domains of multimedia messaging and VOIP. Here is presented some overview of the things that can be done in future to make RUMMAGE a much more effective system.

2.2.19.2.1 Web Mail Analysis

Web mail is email access system which where email communication is done over HTTP. This represents an indirect model of email access. The web based email clients communicate with mail servers through Web Mail. In Web Mail system clients do not access mail servers directly. But they communicate with the web server over HTTP which in turn communicates with mail server to service the request. Web Mail monitoring capability can be added to RUMMAGE by decoding HTTP and identifying the email communication over HTTP.

2.2.29.2.2 Automated Analysis of Formatted Text Document Attachments

Currently RUMMAGE performs intelligent analysis only on text part of email. Attached documents that have formatted text for example PostScript, Microsoft office and html documents can be converted into simple text document by decoding and removing formatting. That can be in turn treated as simple text documents for classification.

2.2.39.2.3 Optimization to Work on Gigabit Speed

RUMMAGE currently works well on network speed of 100 Mbps. As many of the networks run on Gigabit speed, RUMMAGE must be enhanced in its packet capture capabilities at higher speeds. For sniffing and filtering at gigabit speeds, some hardware support may have to be incorporated along with packet processing capabilities of software in user space.

~~2.2.4~~9.2.4 Handling Encrypted Traffic

RUMMAGE currently analyses non-encrypted traffic due to limitation of its scope and resources. Handling encrypted traffic is not an easy task and involves a lot of effort. Extremely high performance resources are required for task.

~~2.2.5~~9.2.5 Enhancements in Classification

Currently mail analysis is done using statistical data mining techniques. There are slots of enhancement in the domain of semantic based data mining. More sophisticated mail classification techniques can be incorporated in future.

~~2.2.6~~9.2.6 Porting to Unified Messaging Systems

Unified Messaging systems are becoming widespread, handling emails, messaging and fax like facilities through single interface. Much can be done in this domain.

~~2.2.7~~9.2.7 VOIP Analysis

VOIP is widely used across the globe and more and more communication systems are being transformed to be VOIP based. VOIP analysis includes decoding and examining of voice codecs. This is a complex domain and needs lot of effort. RUMMAGE provides suitable grounds for analysis of VOIP based systems.

Formatted: Space After: 0 pt

Formatted: Caption

BIBLIOGRAPHY

[1] RFC-1939, Post Office Protocol.

[2] RFC-3501, Internet Mail Access Protocol.

[3] RFC-821, Simple Mail Transfer Protocol.

[4] RFC- MIME

[5] S. McCanne _ and V. Jacobson, "The BSD Packet Filter: A New Architecture for User-level Packet Capture," *Lawrence Berkeley Laboratory*, December 19, 1992.

[6] RFC-791, Internet Protocol, 1981.

[7] RFC-793, Transmission Control Protocol.

[8] T. Sivanadyan, "Spam? Not Any More! Detecting Spam emails using neural networks", 2004

[9] M. Vinther, "Intelligent junk mail detection using neural networks", 2002

[10] J. S. Breese, D. Heckerman, C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering", 2003

[11] RFC-815 IP datagram reassembly algorithm.

[12] Y. Yang and Pederson, "A comparative study of feature selection in text categorization," in *Proc. ICML-97, 14TH International conference on Machine Learning, San Francisco*, pp. 412-420, 1997.

[13] S. Haykin, *Neural Networks: A comprehensive foundation*, 2nd ed. Prentice Hall, 1999

[14] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys 34*, 1, 1-47, 2002

[15] R. Goetschi, "Spam-Filtering using Artificial Neural Networks", 2004

[16] "LingSpam," *http://www.iit.demokritos.gr/skel/i-config/downloads/-*

Formatted: Space After: 0 pt

Formatted: Font: Not Italic

Formatted: Font: Not Italic

Formatted: Dutch (Netherlands)