PROXY WEB SERVER

PROJECT REPORT

P- 872

Submitted by

ARUN. L. K.	BRIJITH. K. K.	KINGSHUK ROY
9927K0117	9927K0822	9927K0824

Under the guidance of

Ms. S. RAJINI, B.E.

In partial fulfillment of the requirements for the award of the Degree of
Bachelor of Engineering
in Computer Science and Engineering of Bharathiar University,
Coimbatore.

April 2003

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KUMARAGURU COLLEGE OF TECHNOLOGY,
Coimbatore – 641 006.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KUMARAGURU COLLEGE OF TECHNOLOGY,
Coimbatore – 641 006.

PROJECT REPORT
April 2003

CERTIFICATE

This is to certify that the project work entitled **"PROXY WEB SERVER"** submitted by the following students:

ARUN. L. K.	**BRIJITH. K. K.**	**KINGSHUK ROY**
9927K0117	9927K0822	9927K0824

In partial fulfillment of the requirements of the award of the Degree of Bachelor of Engineering in Computer Science and Engineering of Bharathiar University, is a record of bona fide work carried out by them.

Rajini

Ms. S. Rajini
Guide

Prof. S. Thangasamy
Head of the Department

Place : Coimbatore

Certified that the candidate with Register no ____________________

was examined by us in the project viva voice held on 17.03.2003

Rajini
Internal Examiner

[illegible]
External Examiner

ne Possibi

Infosystems

To Whomsoever It May Concern

This is to certify that the following students of final year B.E. (Computer Science and Engineering), Kumaraguru College of Technology, Coimbatore has completed the project entitled "**PROXY WEB SERVER**" from December 2002 to March 2003.

1.	ARUN L.K	9927K0117
2.	BRIJITH.K.K	9927K0822
3.	KINGSHUK ROY	9927K0824

[illegible] /11-03-2003

K.R.DHARANKHANNAN

Director

ACKNOWLEDGEMENTS

We are extremely thankful to our beloved Principal **Dr. K.K Padmanabhan,** Kumaraguru College of Technology, for being our source of inspiration through out my study in our college.

We owe our sincere gratitude to **Prof. S.Thangasamy, Ph.D.,** Professor and Head of the Department, Computer Science and Engineering, Kumaraguru College of Technology, who took keen interest in me and encouraged me on every effort through out this Course.

We are indebted and extremely thankful to my internal guide **Ms.S.Rajini,** B.E for her valuable guidance and help through out the course of the project.

We are grateful to **M/s REGALIA** Infosystems, Coimbatore, for giving me the great opportunity to take up this project and for providing the excellent facilities to execute the project. I also wish to express am sincere thanks to **Mr.K.R.Dharankhannan,** M/s REGALIA Infosystems, Coimbatore, for his valuable guidance and constant encouragement during the course of the project.

We would fail in our duties if we don't thank Lord Almighty, our Family and our Friends for their invaluable support to see through this project.

SYNOPSIS

The project entitled PROXY WEB SERVER is to develop a proxy mail agent, proxy HTTP agent and Proxy DNS for an organization having web space in the remote web server. In a networked environment this agent acts as a gateway between the local network and the Internet Zone.

The system acts as a mail agent for intranet and a mail agent for Internet zone. The system will collect the mail from the mail client and check whether the mail belongs to Intranet or Internet. The mail will be sent to the appropriate recipient immediately, if the mail belongs to the Intranet else the mail will be stored in the user's local mail box. When the system is online the mail stored in the user's mail box will be sent to the remote web server.

The incoming mails which are stored in the remote server are collected and stored in the appropriate user's mail box, when the system is connected to the internet. The mailing system is developed using SMTP and POP3 protocol standards.

The system also acts as a proxy HTTP and proxy DNS agent. The system provides necessary authentication and security to the user. The system also provides remote administration interface for user account.

Introduction part deals with explaining the basic function of this system. This part gives a description of the problem that has to be addressed. The draw back of the existing system and the need for this system were identified.

The analysis part deals with identifying the requirement of the system and the hardware and software required to develop the system. It specifies the scope of the system with in which this system can be used. It also explain the methodology used for developing the system.

The next part design explains the modules of this project and the way they interact with each other. This part specifies the user interfaces of the system. Design representation like architectural diagram and module diagram are specified in this part,

The implementation part deals with testing and testing methodologies and the test cases used to test the system. It describes about the installation of the system. It also presents an overview of the errors and the failures encountered during the implementation process.

Limitation and feature enhancement deals with the constraints of the system and the improvement that can be made to the system.

This project will be developed under the Windows NT 4.0 work station platform using JAVA.

Table of Contents

1.1 PROBLEM DEFINITION

The main aim of this project is to develop a proxy agent for any protocol request to the actual service provider. Client terminal raises the protocol requests to one unified location – this system. These systems then dispatch the request to the internet zone and collect the response. This response is then delivered back to the client terminal. This system maintains a connection table for all the incoming connections via 802.3 protocol and outgoing connections via a PPP (Point to Point Protocol).

Also the system aims to effectively manage and share the resources. This system implements a priority queue for the shared communication resources like a PPP session. Various users fall under one priority level. The effective bandwidth is shared among the various class of users.

This system behaves like an offline mail server for the recipients in the Internet zone. The system collects all the outgoing mails to internet zone and stores it to a local database. Before actually sending the mail to the Internet zone, the mail size destination address and the source address are employed. Request succeeded against the policies are routed to the Internet interface. All the inbound mails are verified and validated against policies before actually retrieving the mail from the remote web server.

The system behaves as a mail server for the intranet zone. This system implements the SMTP protocol for forwarding mails and implements POP3 for retrieving mails. POP3 protocol uses a simple file based authentication mechanisms. A copy of all the incoming and outgoing mail is sent to the administrator's mail id.

The system also has a remote administration tool for effective management of users. The incoming and outgoing mails are encrypted and then are stored in the local database. When the mail is dispatched it is decrypted and dispatched.

The system proxies the DNS request and implements the HTTP protocol for fetching the pages from the remote site. HTTP and DNS protocols use simple files for authentication mechanism and firewall policies.

1.2 ORGANIZATION PROFILE

REGALIA is a premier solution provider started in the year 2002. It has pioneered itself in their products and services that operate in

- Business technology consulting.
- Traditional programming.
- Web application.
- Multimedia graphic solutions.

1.3 EXISTING SYSTEM

- In the existing system the user should always be connected to internet for SMTP and POP3 access.
- No proxy server exists. Hence each and every system has to be connected individually to the Internet for HTTP access.
- No Intranet mailing facilities exist.
- There is virtually no internet security in the existing system.
- The system is expensive.
- Expensive resources like telephone line modem cannot be used in a shared mode.
- Individual user's access to the internet cannot be traced as well as accounted.

1.4 PROPOSED SYSTEM

Proposed system incorporates bulk mail dispatch feature so that users can send mails even when the system is not connected to the internet. Bulk mail dispatch feature is one in which the mailing system collects all the internet mail from the connected system and saves it in a centralized mail box and dispatches all those saved mails when ever the system is connected to the internet there by hiding the actual mailing scenario from the connected systems. The point to be noted is that the intranet mails are dispatched immediately to the local intranet user mail box.

One system is designated to be a proxy HTTP agent and the other systems are connected to it for HTTP access so that a single dialup connection can be shared over the network.

Increased internet security and regulatory features –

- SMTP and POP mail size can be controlled.
- Mail blocking system to regulate mails.
- Authentication of user access to the internet.
- Character based encryption for secure transfer of mails.
- Mail monitoring tool for mail surveillance.
- GUI based remote mail box administration tool for easy maintenance.

With all the above mentioned features the draw backs of the existing system have been overcome effectively.

- Cost of the system is considerably less.
- Resources such as modem and telephone are effectively shared.

2.ANALYSIS

2.1 SYSTEM REQUIREMENTS

The system requirements specify the functional and user interface requirement for the system.

2.1.1 Functional Requirement

The functional requirement of the system includes the following:

- Implementing a mail server in accordance to the SMTP (Simple Mail Transfer Protocol) and POP3 (Post Office Protocol) is one of the fundamental requirements.
- The mail server is customized in such a way that it can handle mails even when the system is not connected to the internet making the actual mailing scenario transparent to the users.
- Effective management and proper authentication of the user mail boxes.
- Restricting the size of the incoming as well as out going mails
- Mails are encrypted to out smart the eavesdroppers.
- Developing a mail blocking system to legalize mails.
- Developing a HTTP (Hyper Text Transfer Protocol) server to handle client HTTP request.
- Proper authentication of the client for HTTP and DNS request.
- Maintaining a log of all the sites visited by the user.
- Developing a remote administration tool for maintaining the user accounts.
- Developing a mail monitoring tool which maintains log showing information namely senders name recipient name, date and time of all the mail sent and received by a user.

2.1.2 User Interface Requirement

- Any mailing client conforming to the internet standards for example MS Outlook Express, Eudora and Netscape mail client etc., can be used to compose, send and receive mails.
- Like wise any Web Browser that can be configured to handle proxy server, example MS Explorer, Netscape Communicator etc., can be used to act as a HTTP client.
- Remote administration tool is a user friendly GUI based tool which has menus to add and remove mail users as well as configure the size of the users mail box.

2.2 SCOPE OF THE SYSTEM

The scope of the system is limited to the following

- The intranet activity when the system is restricted to the IEEE 802.2 standards (Ethernet).
- The public network for the internet access is restricted to the dial up connection through the telephone line.
- The mail server supports only the POP3 protocol for fetching the mails.
- The number of connections to the system is limited in order to achieve better performance levels.

2.3 SYSTEM ENVIRONMENT

It gives the details regarding the environment in which the system is developed. The specifications are given below:

2.3.1 Development Environment

Hardware

Processor	: Pentium III
Hard disk	: 40 GB
RAM	: 128MB
Modem	: Data Fax / Voice 56.6 kbps

Software

Operating system	: Windows 98
Language	: JAVA

2.3.2 Implementation Environment

Hardware

Processor	: Pentium III & above
Hard disk	: 40 GB
RAM	: 128 MB
Modem	: Data Fax / Voice 56.6 kbps

Software

Operating System : Windows NT workstation is the preferable choice and the system can also run under any Windows '98 & above platforms.

2.4 SYSTEM MODELING

2.4.1 PROCESS MODELING

2.4.1.1 Context Flow Diagram

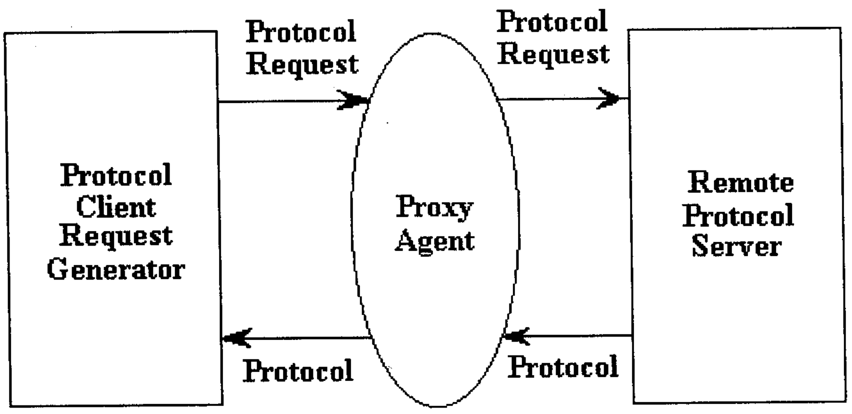


Figure 2.1

2.4.1.2 Level 1 Data Flow Diagram

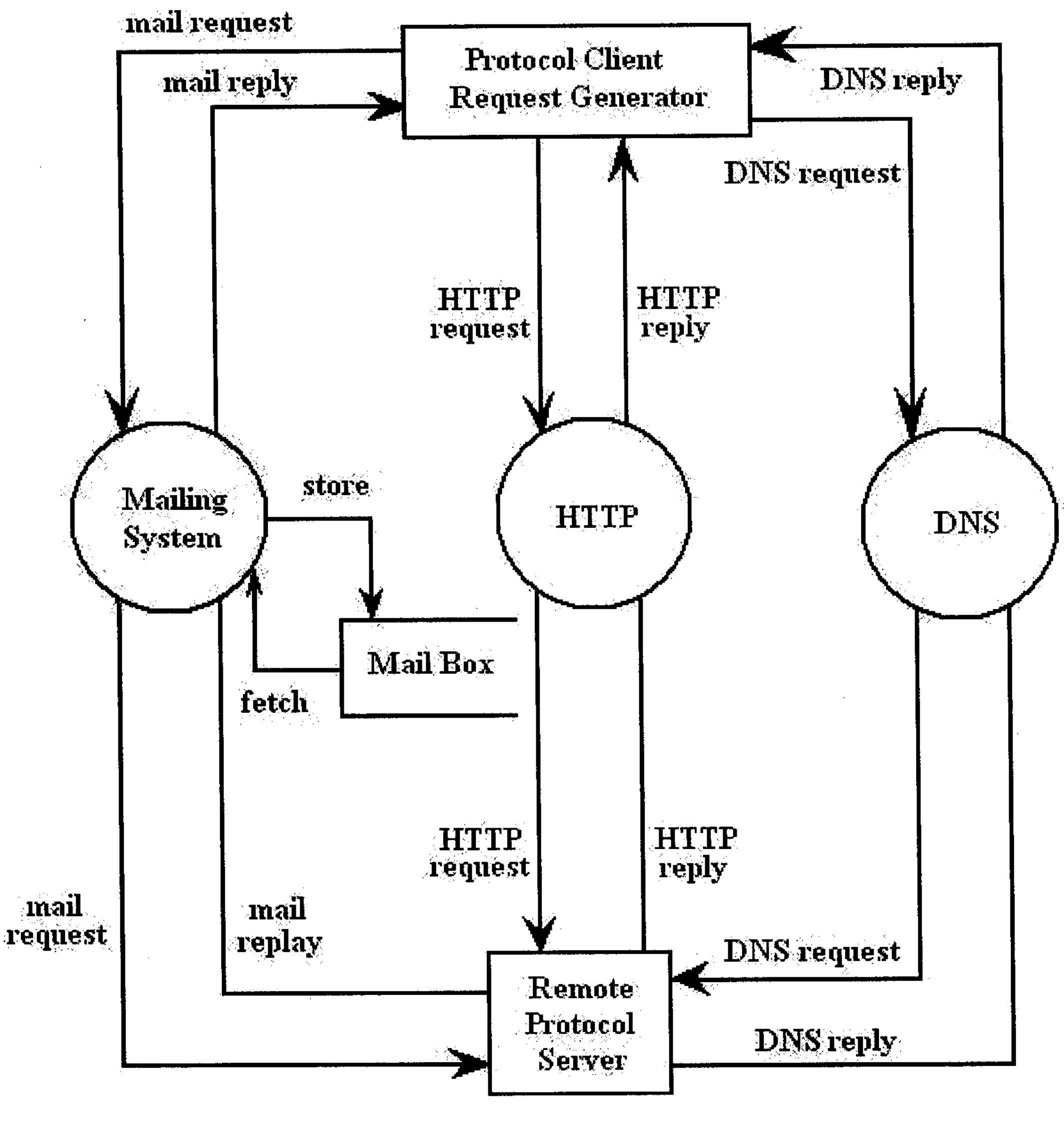


Figure 2.2

2.4.1.3 State Transition Diagram

The following figure 2.3, 2.4, 2.5 and 2.6 show the State Transition for the whole system.

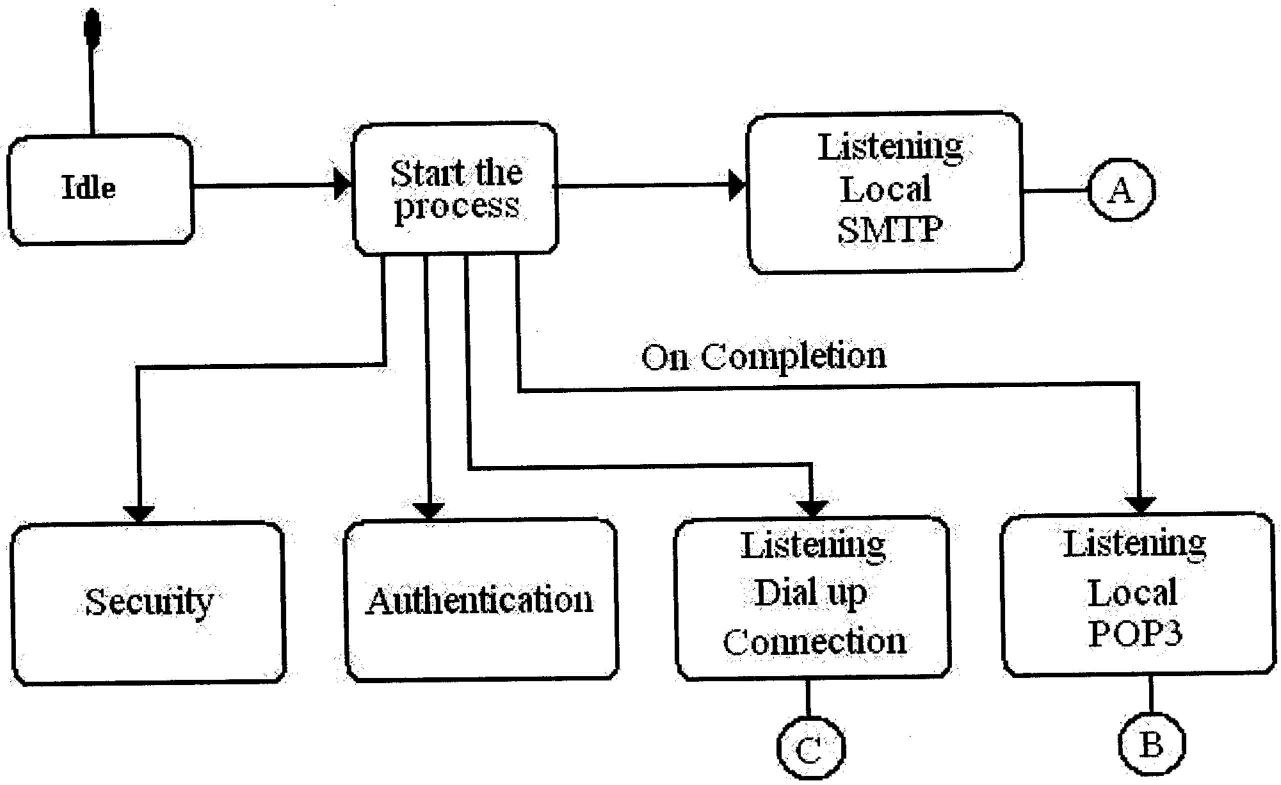


Figure 2.3

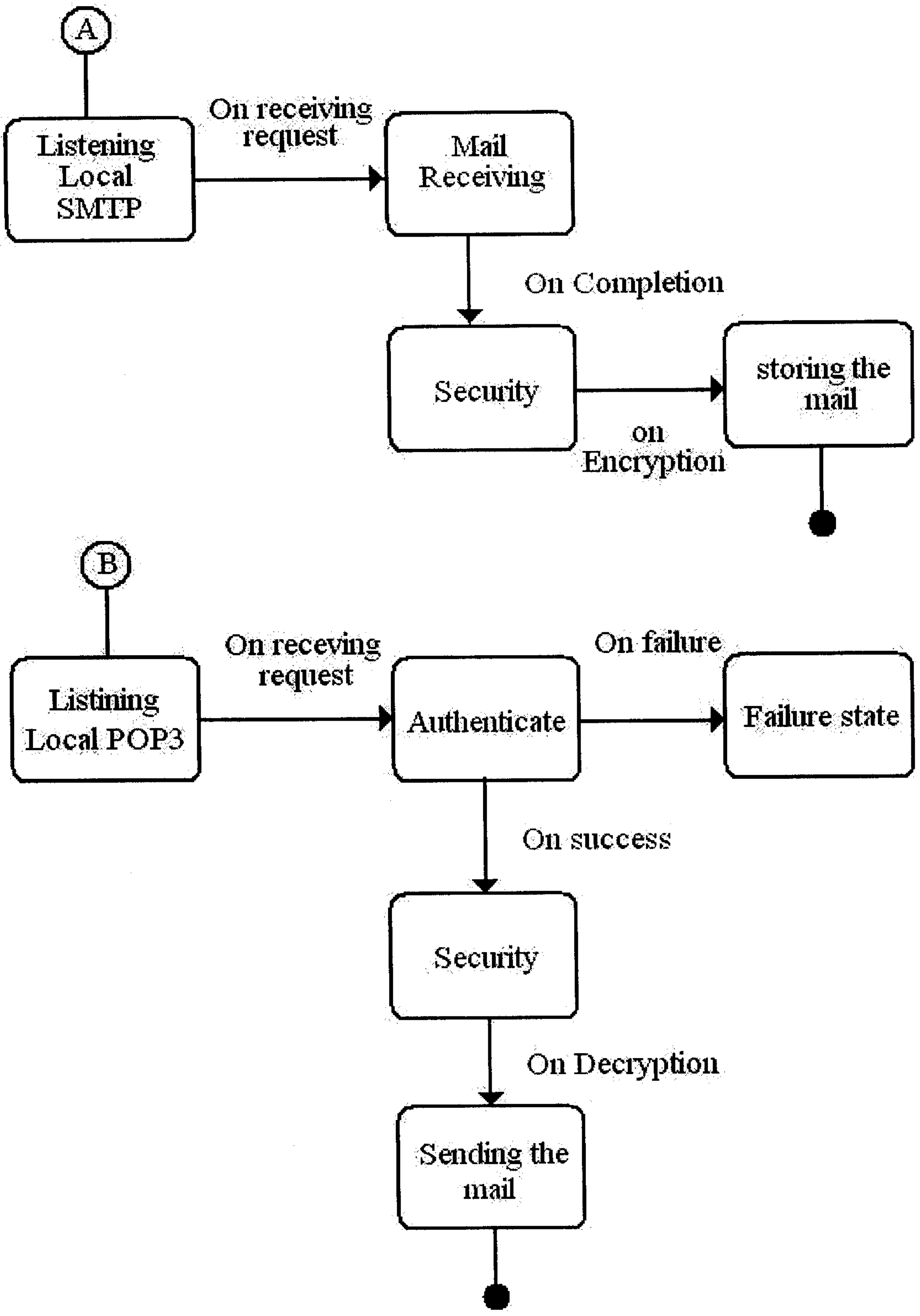

A
Listening Local SMTP
On receiving request
Mail Receiving
On Completion
Security
on Encryption
storing the mail
B
Listining Local POP3
On receving request
Authenticate
On failure
Failure state
On success
Security
On Decryption
Sending the mail

Figure 2.4

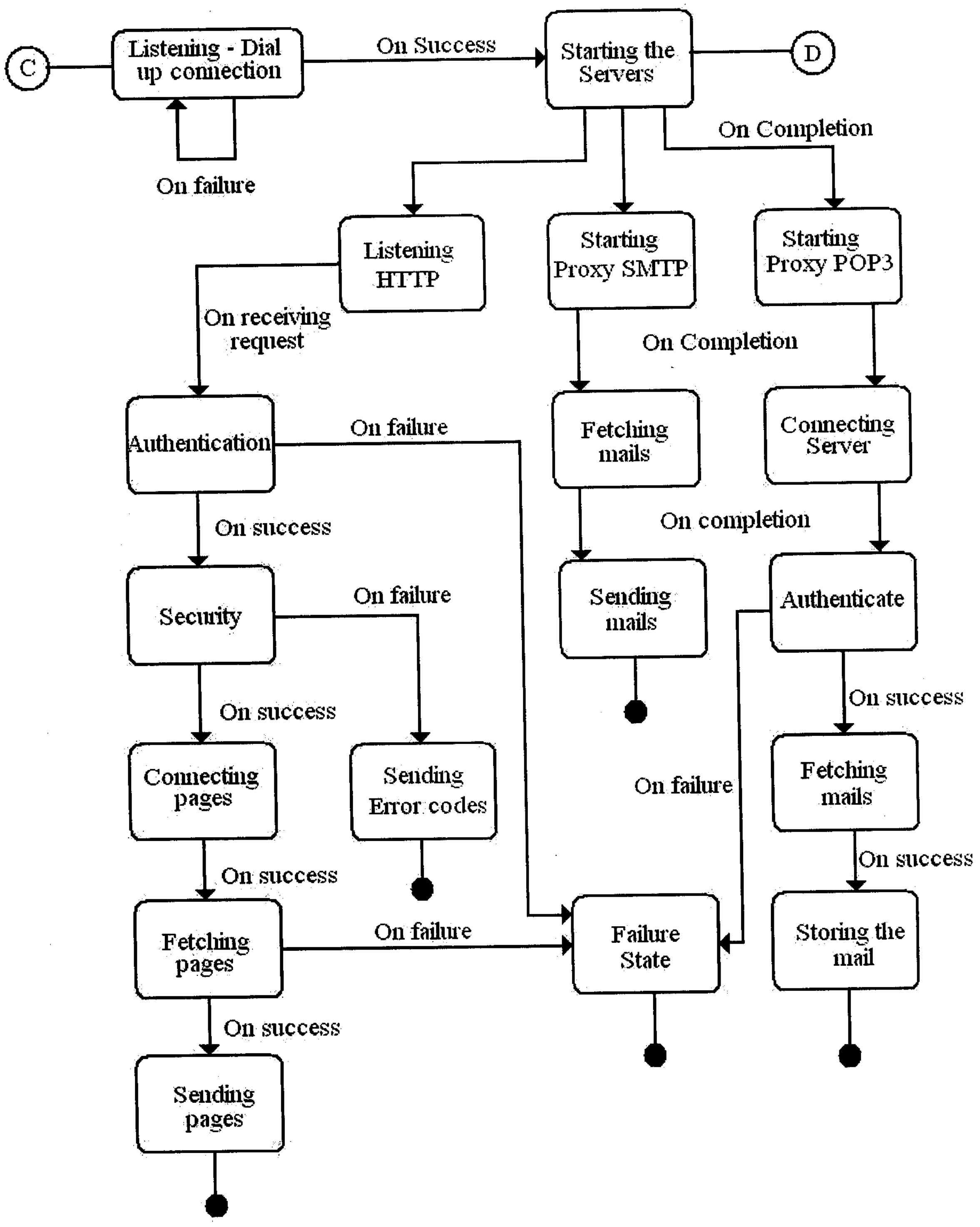
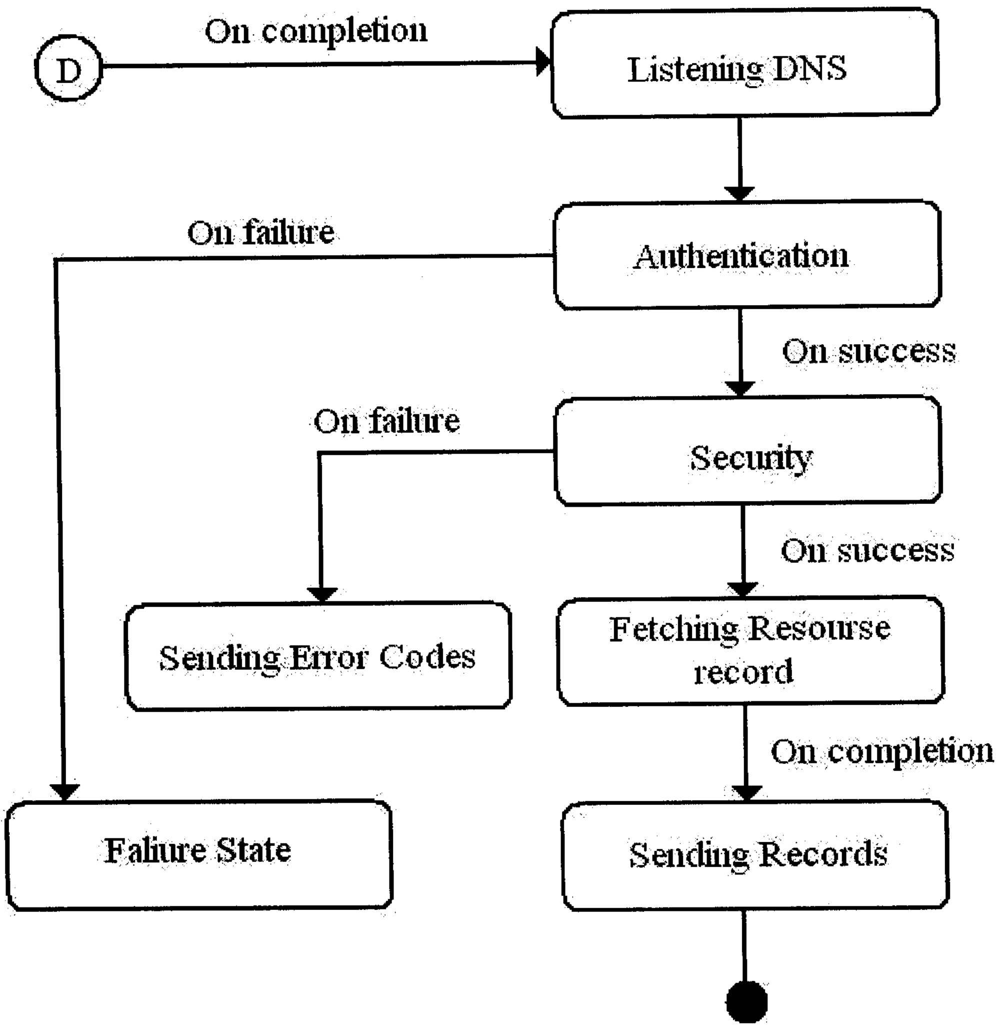

C
Listening - Dial up connection
On failure
On Success
Starting the Servers
D
On Completion
Listening HTTP
Starting Proxy SMTP
Starting Proxy POP3
On receiving request
On Completion
Authentication
On failure
Fetching mails
Connecting Server
On success
On completion
Security
On failure
Sending mails
Authenticate
On success
On success
Connecting pages
Sending Error codes
On failure
Fetching mails
On success
On success
Fetching pages
On failure
Failure State
Storing the mail
On success
Sending pages

Figure 2.5



D
On completion
Listening DNS
On failure
Authentication
On success
On failure
Security
On success
Sending Error Codes
Fetching Resourse record
On completion
Faliure State
Sending Records

Figure 2.6

2.5 JUSTIFICATION OF DEVELOPMENT METHODOLOGY

The proposed system is developed using the incremental modeling. The incremental model combines elements of linear sequential model with iterative philosophy of prototyping. Each linear cycle sequence produces a deliverable "incremental version of software". The process is incremental in the sense that each pass through an analysis cycle leads to gradual refinement of strategic and tactical decisions, ultimately converging upon a solution that meets the end users real requirements, and yet is simple, reliable and adaptable. When an incremental model is used, the first increment is often a core product. That is, basic requirements are addressed, but many supplementary features remain undelivered. The plan addresses the modification of the core product to meet the needs of the customer and the delivery of the additional features and functionality. This process is repeated following the delivery of each increment, until the complete product is produced.

Applying the above model the basic requirement for the system-protocol requirement is developed to produce the core product. The next step is the functional requirement of the system is that addressed and this process continues.

The advantage of using the incremental model is that it has the features of both the linear sequential and the prototype model.

3.1 DATABASE DESIGN

Mail Box Structure

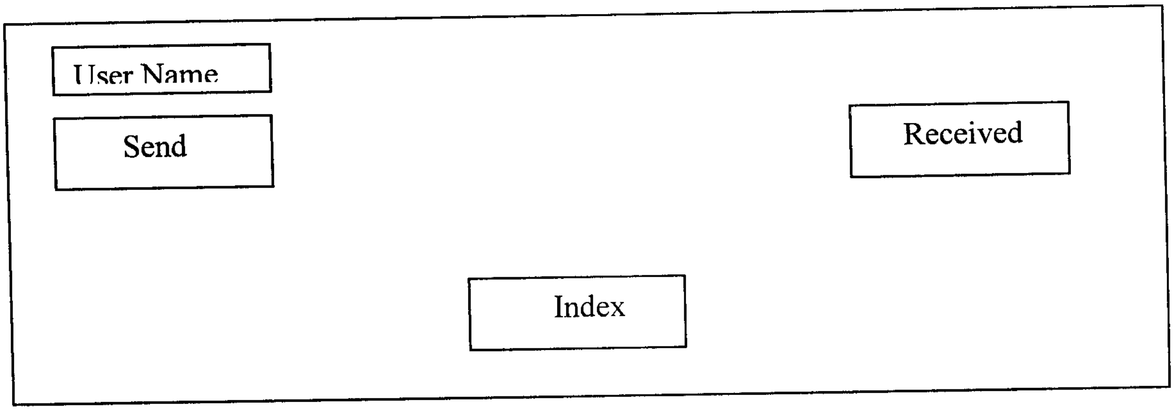


Figure 3.1

The structure of the mailbox is shown in the above figure. The send folder contains the mails that have been sending by the user to the Internet Zone. The received folder contains the incoming mails for the user. The index file maintains the mails that have been present in the send folder. The system will use the index file for its reference.

3.2 MODULE DESIGN

The following figure 3.2 shows the High level Design for the system.

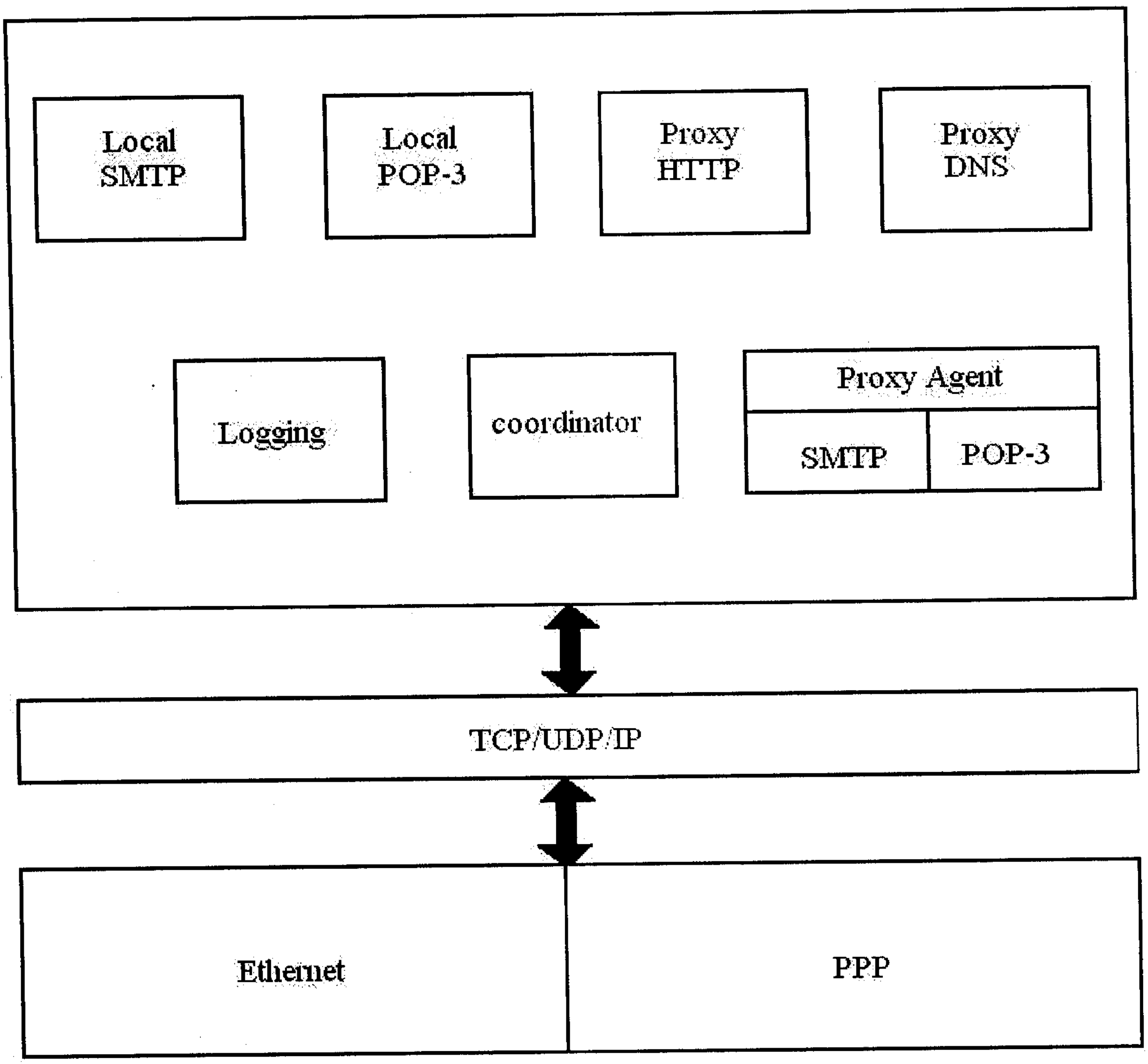


figure 3.2

The system has various modules, they are as follows

- Authentication
- Local SMTP
- Local POP3
- Proxy HTTP
- Proxy DNS
- Proxy SMTP
- Proxy POP3
- Coordinator

Authentication

This module is to authenticate the user while receiving mails from the remote web server, while sending the mails to the mail client and while the HTTP request is raised through the client.

Local SMTP

This module acts as server to the mail client. This module acts as online mail server to the Intranet. It sends the mail immediately if it is an Intranet one. It stores the Internet mail to the local database. The Internet mails are stored in the appropriate user's mailbox.

Local pop3

This module acts as online POP3 server for the Intranet Zone. When the request is raised from the client, the mails are fetched from the appropriate user's mailbox and out bounded. Before this scenario user authentication is done.

Proxy SMTP

When the system is online this module is invoked. This module collects the Internet mails from the user's mailboxes and out bounds to the remote web server. The mails are out bounded according to the priority of the user.

Proxy POP3

This module is also invoked , when the system is online. This module fetches the mail from the remote web server, where the user's mails are actually stored. The mails are collected according to the user's priority level. Before this process starts proper authentication is done with the remote web server.

Proxy HTTP

This proxy agent acts as a forwarding agent. When it receives a URL request from the client it first resolves the DNS by using the proxy DNS module. When it receives the IP from the module, it connects to the appropriate server and fetches the pages for the client. After receiving the pages it sent it to the requested client, it maintains a connection table for the entire request. When the system doesn't receive any further request for an alive connection for a particular time interval, it automatically closes the transmission channel for that connection. The connection table contains the information such as connection number, source IP, destination IP, destination connection number, time since last request received etc.,

Proxy DNS

These module proxies the DNS request for the client. When the request is received it gets connected to the remote DNS server and gets the IP address and sends it to the client.

Coordinator

This module is for remote administration tool. New user creation, deletion of a user etc., is done in the remote web server by this module. Also it coordinates other modules.

3.3 DESIGN DIAGRAM AND CHARTS

3.3.1 Architectural Diagram

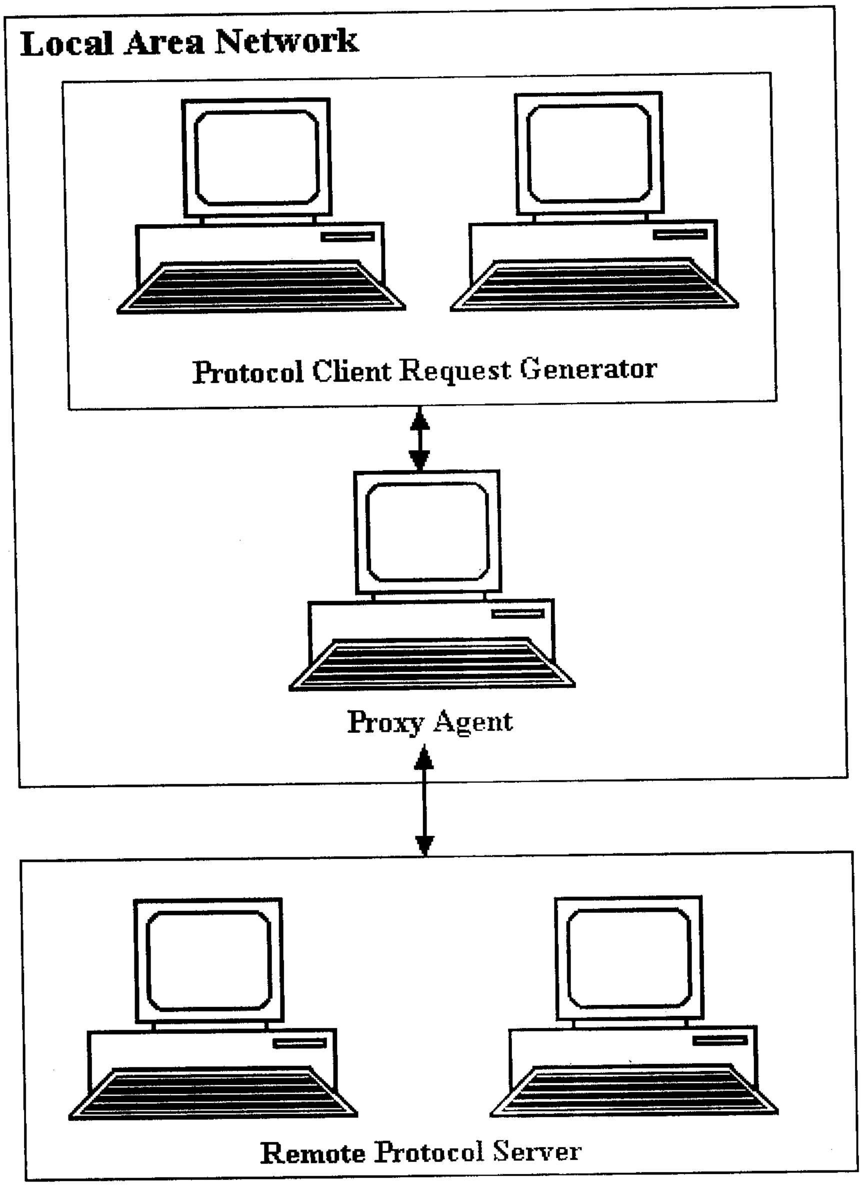


Figure 3.3

3.3.2 Module Diagram

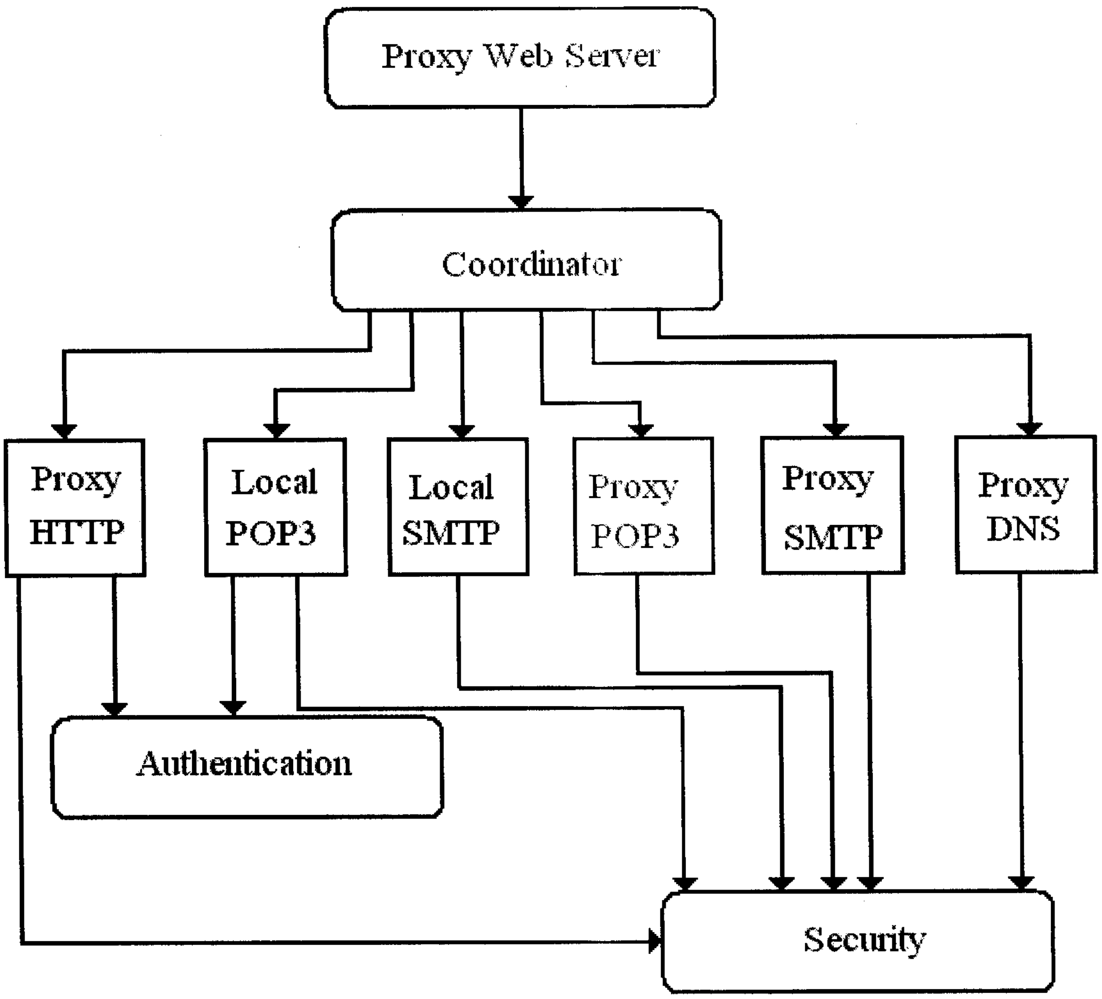


Figure 3.3

IMPLEMENTATION

4.1 TESTING AND TEST PLAN

Testing is a process of executing a program with the intent of finding an error. A good test plan is one that has high probability of finding as – yet – undiscovered error. A successful test is one that uncovers an as – yet – undiscovered error.

The plan for testing was automated test. The two systems client stub and server stub are developed to fix the bug in the system. The client stub acts as a Protocol Request Generator and the server stub acts as a Protocol Server. These two system maintains a table, which consists of the fields protocol reply or protocol request and test result.

The **System Under Test** (SUT) – Proxy Web Server is placed between the client stub and server stub and the testing is done. It is shown below in the figure.

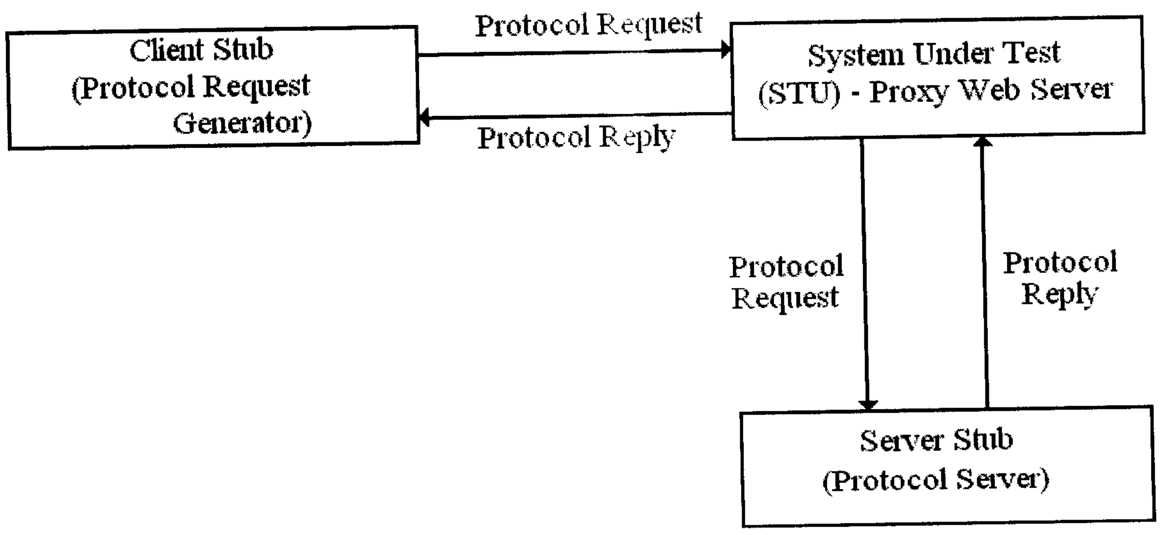

Client Stub (Protocol Request Generator)
Protocol Request
Protocol Reply
System Under Test (STU) - Proxy Web Server
Protocol Request
Protocol Reply
Server Stub (Protocol Server)

Figure 4.1

4.2 TESTING METHODS

4.2.1 Unit Testing

The program unit as a single entity was tested independently of other units. This was done to test the working of each unit as independent entity, the functional fulfillment of the unit, the error handling capabilities of the program and the data integrity aspects

For example when the system receives a mail, the mail data will be ended with a CRLF pair. There was an error in the program unit in identifying the end of the mail. This was corrected.

4.2.2 Integration Testing

The integration testing was performed to ensure that the modules are properly integrated in all levels of the system.

For example when the system receives the HTTP request, it first resolves the URL using the proxy DNS module. This was tested successfully.

4.2.3 Performance Testing

The performance testing was performed to ensure that the whole system performs well to achieve better results.

For example initially the number of maximum connections to each protocol was 40. To achieve better performance it was limited to 20.

4.3 INSTALLATION OF THE SYSTEM

PROXY WEB SERVER is a generalized product, any organization having a network environment and having a web space in remote server can use it. To install this product Windows NT workstation is preferable, though the system can run under any Windows platform and a modem is required as an additional hardware.

4.4 ERRORS AND FAILURES DURING THE SYSTEM DEVELOPMENT AND SOLUTIONS

Initially the user has to start the system. This would lead to poor performance of the system. To over come this, server is implemented as Daemon process.

The system supports only limited number of connections the system can be enhanced in the future. Some of the enhancements are given below:

- Strictly implementing the fire wall policies.
- Implementing Socks and RASP protocol.
- Permitting remote logging for client via TELNET.

Firewall can be strictly implemented in order to enhance security and save the network from hackers while using the Internet.

If the system is implemented using TELNET, then a remote user can login and access the network resources.

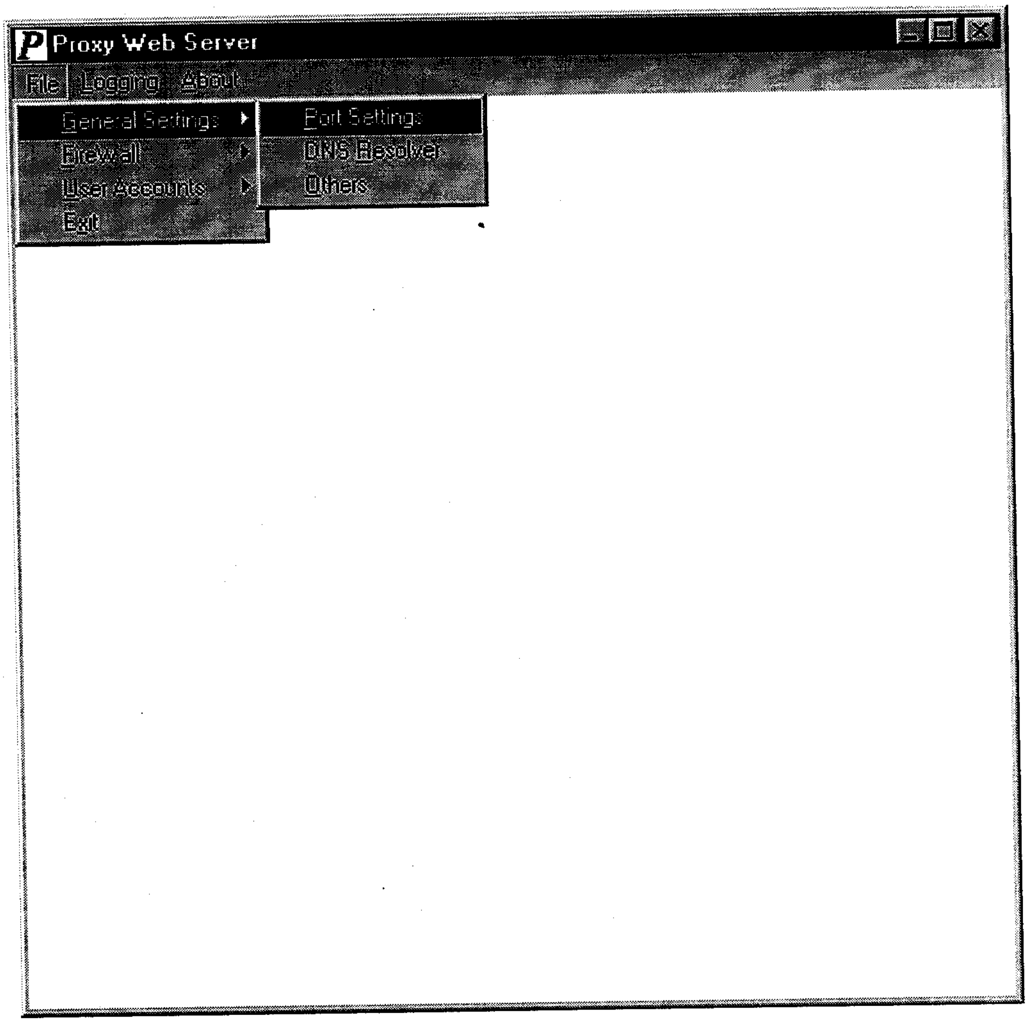
This project was aimed to proxy the request coming from the protocol client request generator and reply back to the client. The user interface screens in the system are easy to use. The system maintains log for each protocol so that error can be easily be fixed and debugged. The system is developed to accommodate changes during the future enhancement.

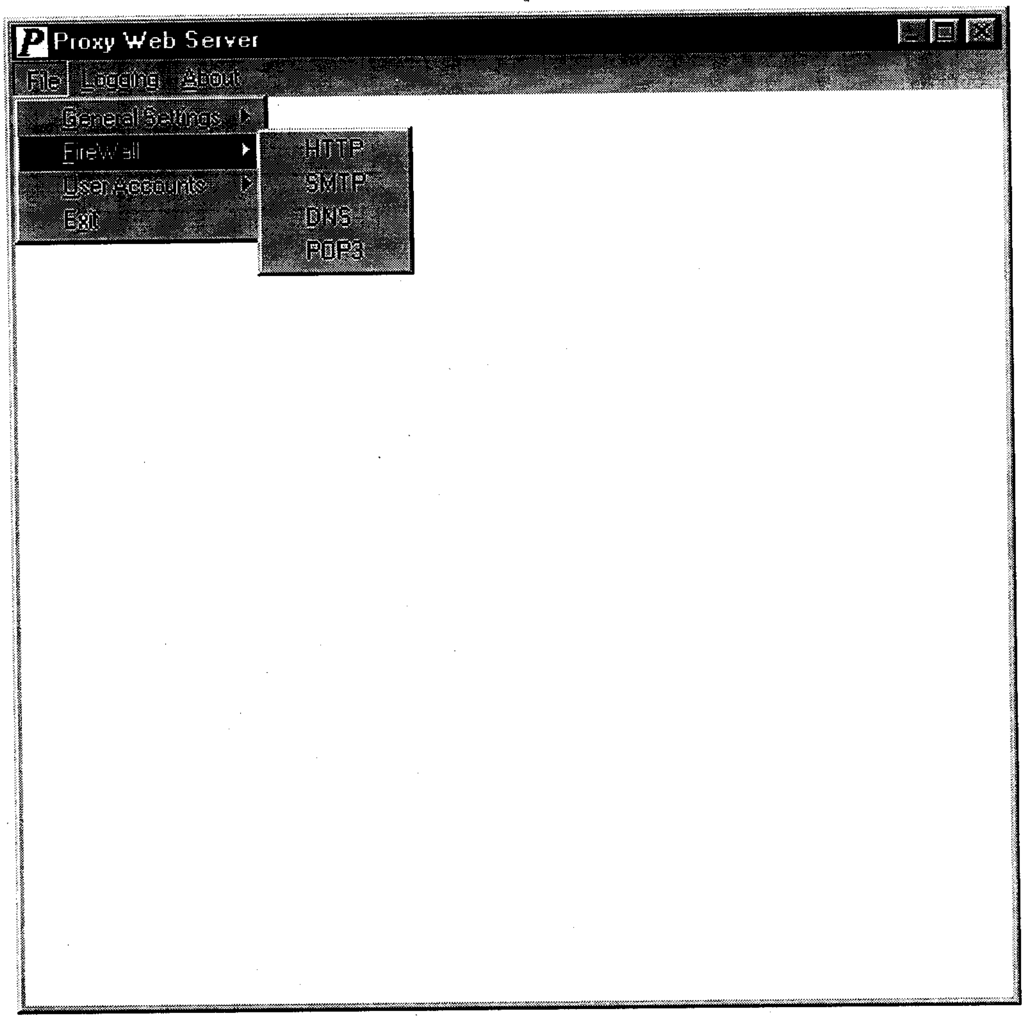
The project has a large scope for future enhancement and is suitable for any Organization having web space in the remote Web server.

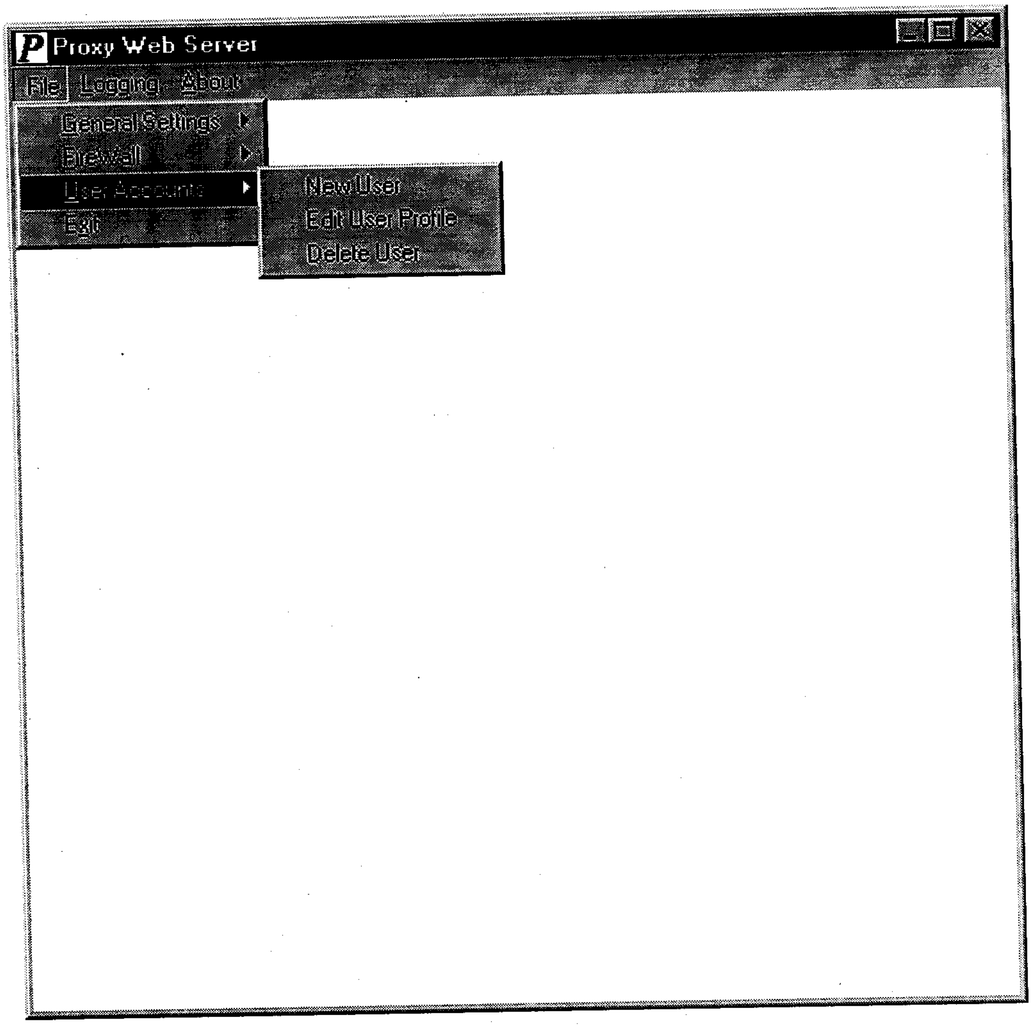
TELNET can be integrated to enable remote login, thus enabling a remote user to access various resources in the network.

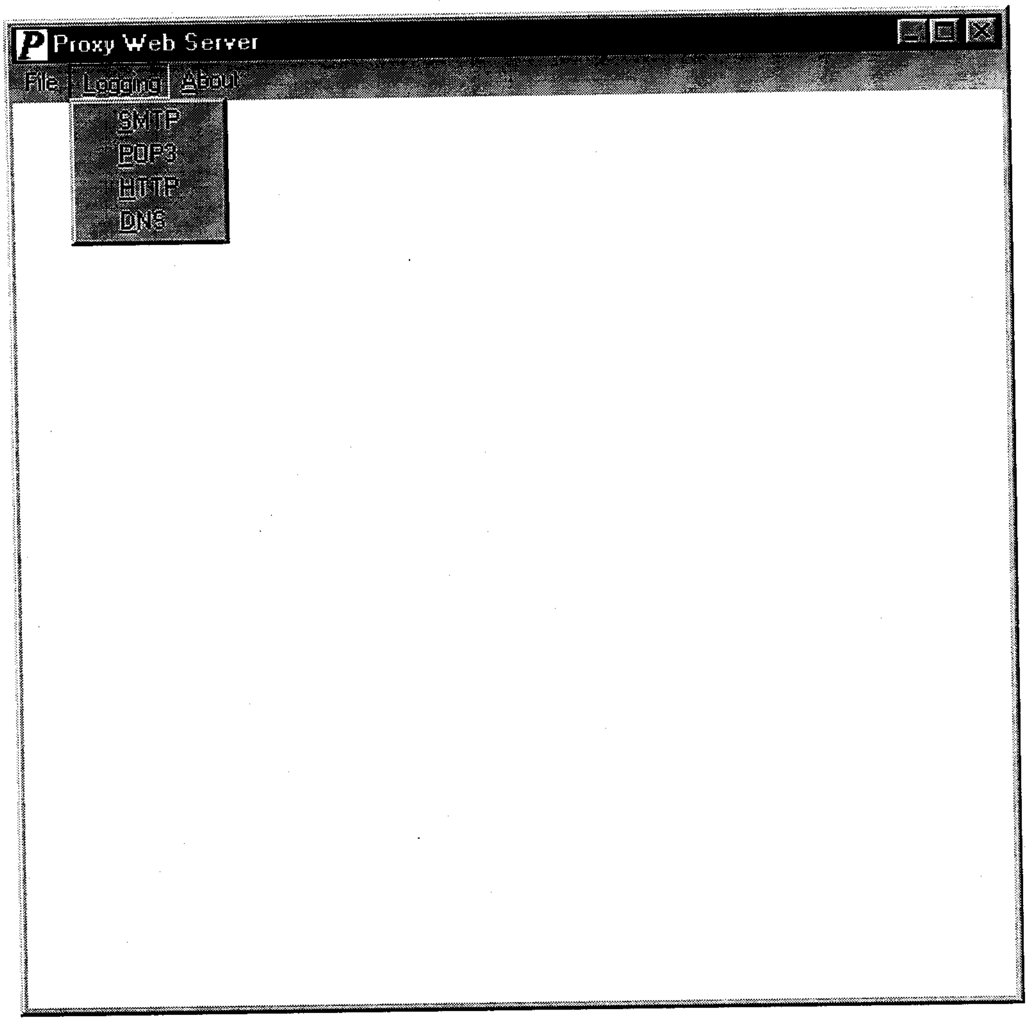
A) PROJECT DICTIONARY

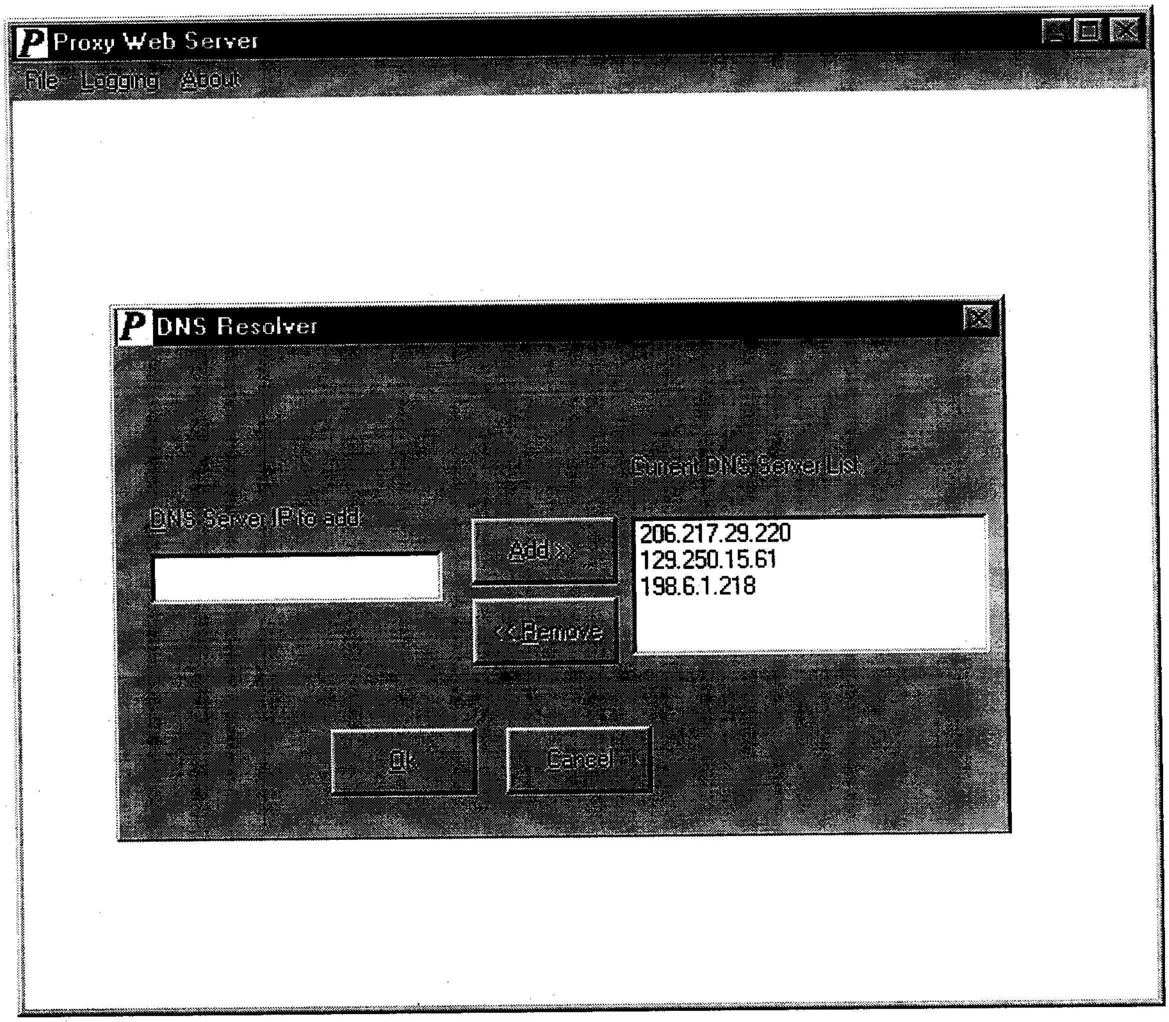
SCREEN DESIGN



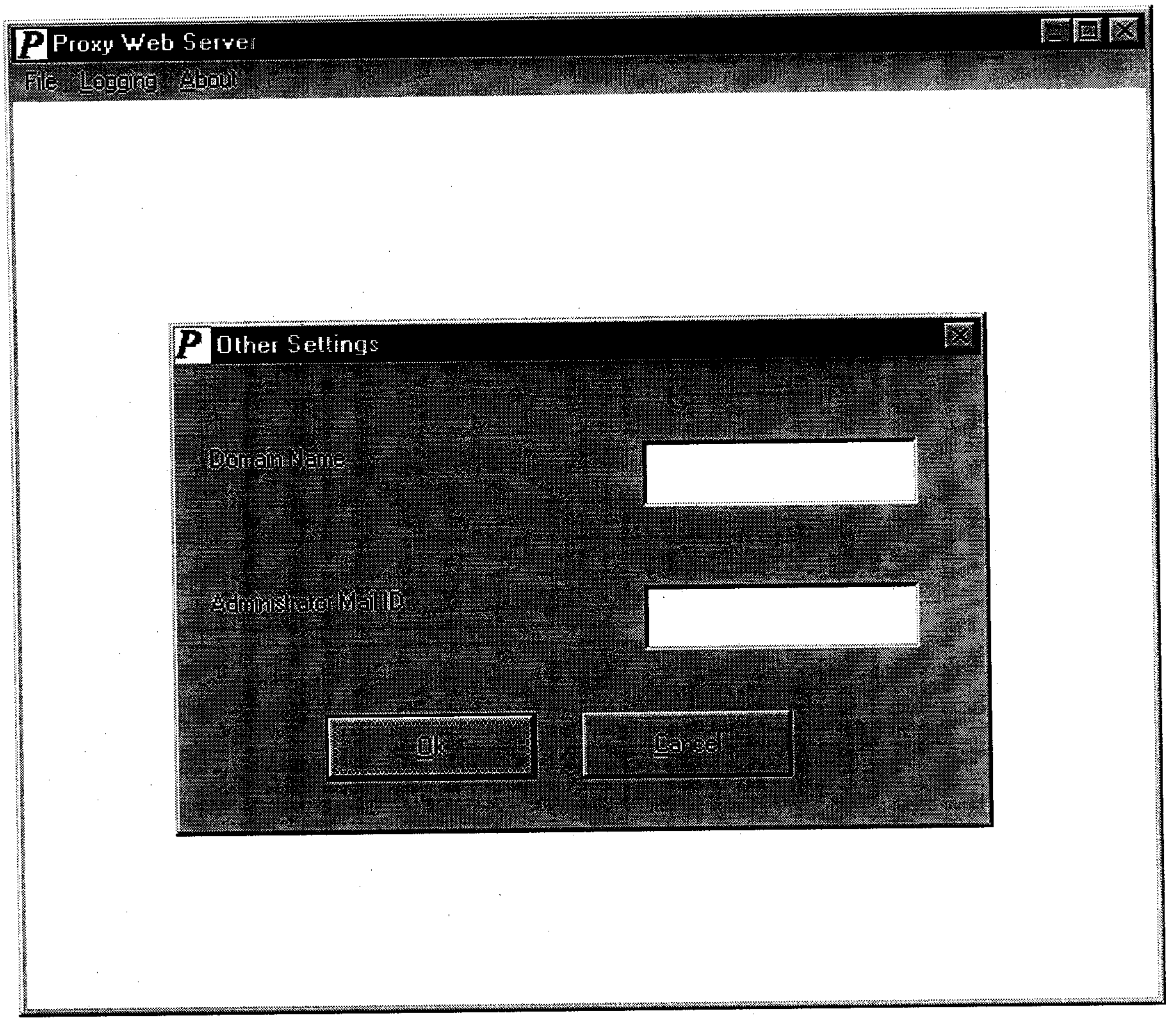

Proxy Web Server
File
Logging
About
General Settings
FireWall
User Accounts
Exit
HTTP
SMTP
DNS
POP3

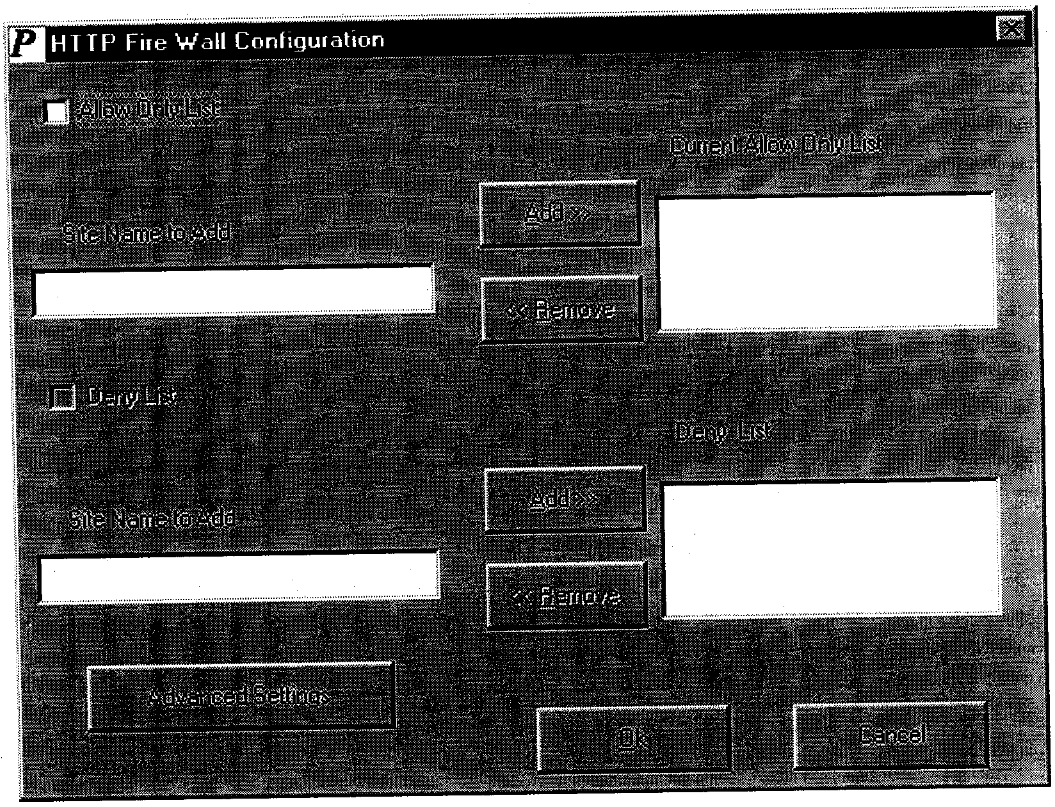

Proxy Web Server
File
Logging
About
General Settings
Firewall
User Accounts
Exit
New User
Edit User Profile
Delete User

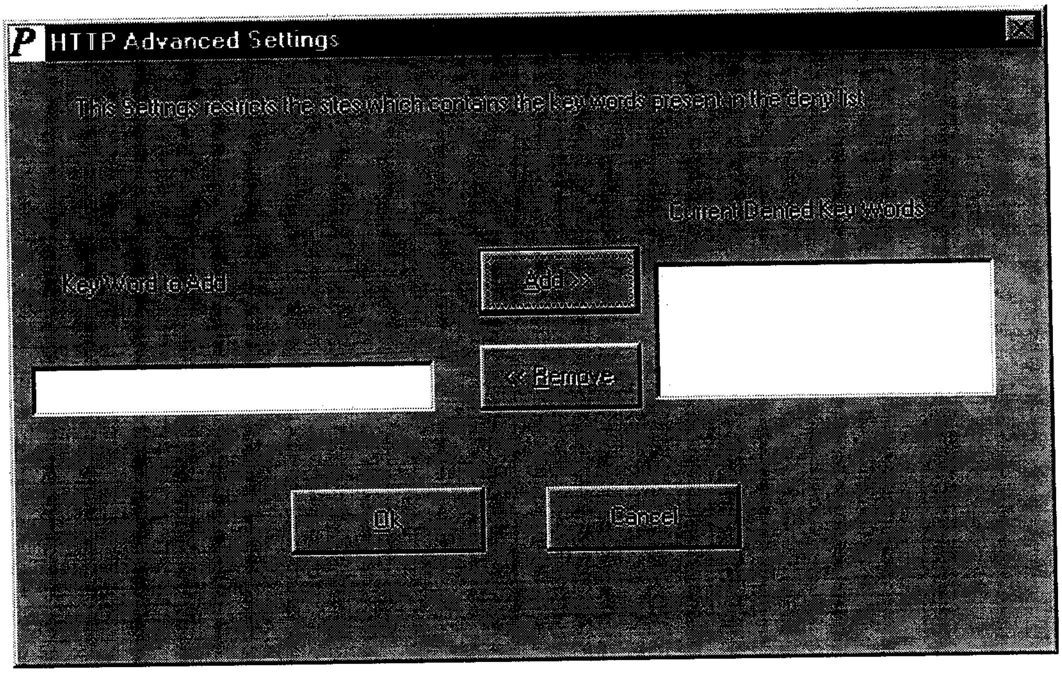

Proxy Web Server
File
Logging
About
SMTP
POP3
HTTP
DNS



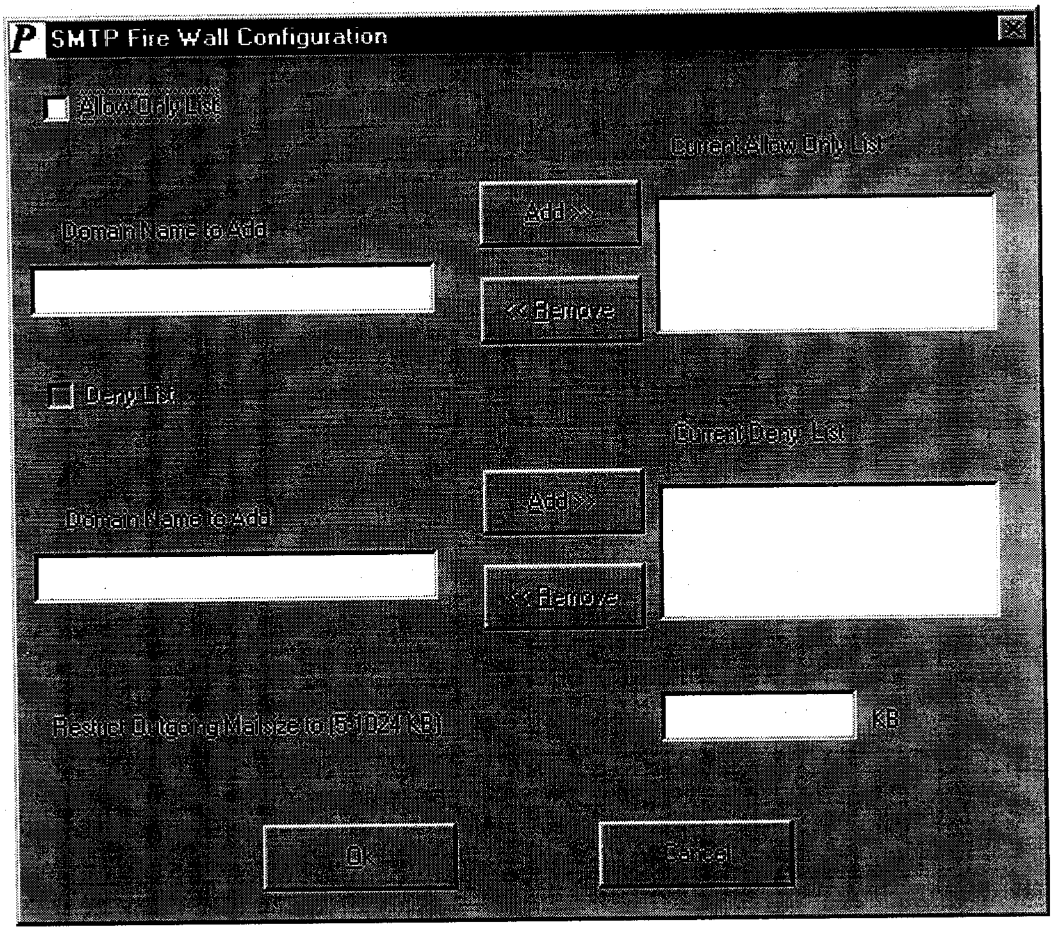
206.217.29.220
129.250.15.61
198.6.1.218

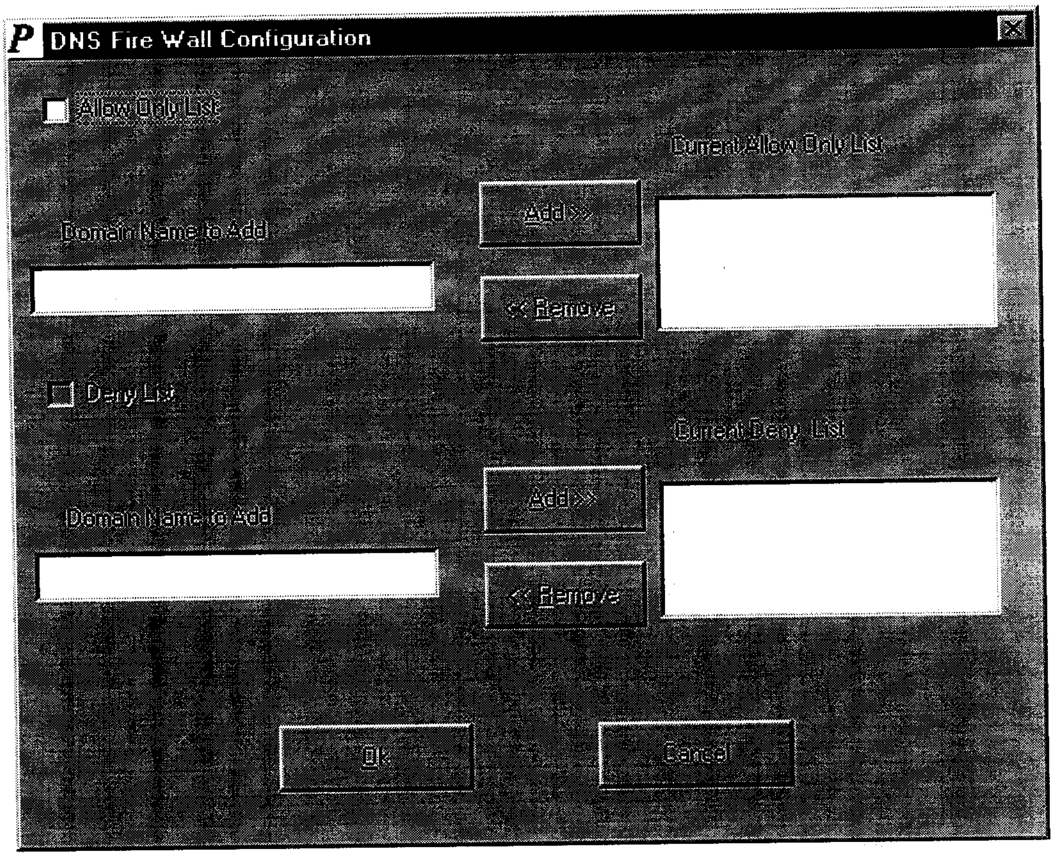

Proxy Web Server
File
Logging
About
Other Settings
Domain Name
Adminstrator Mail ID
Ok
Cancel

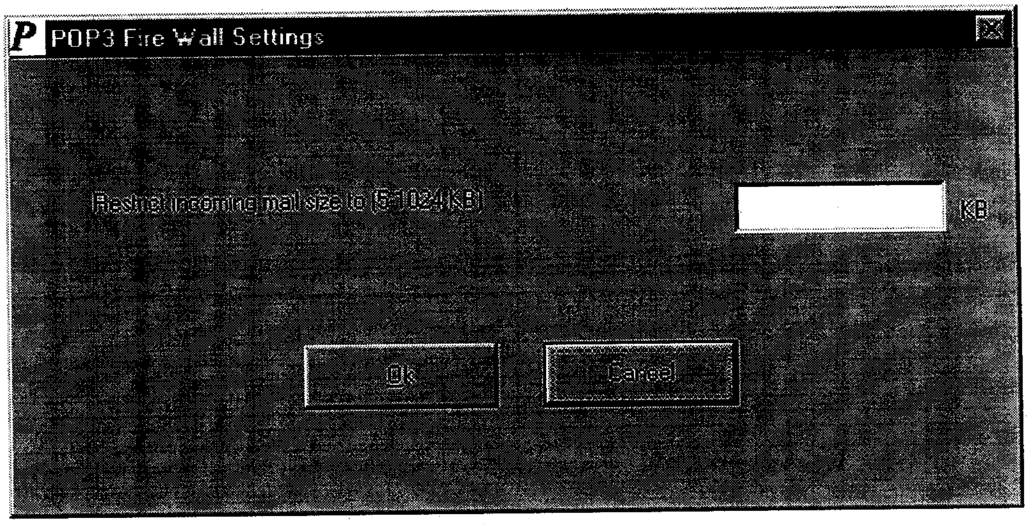


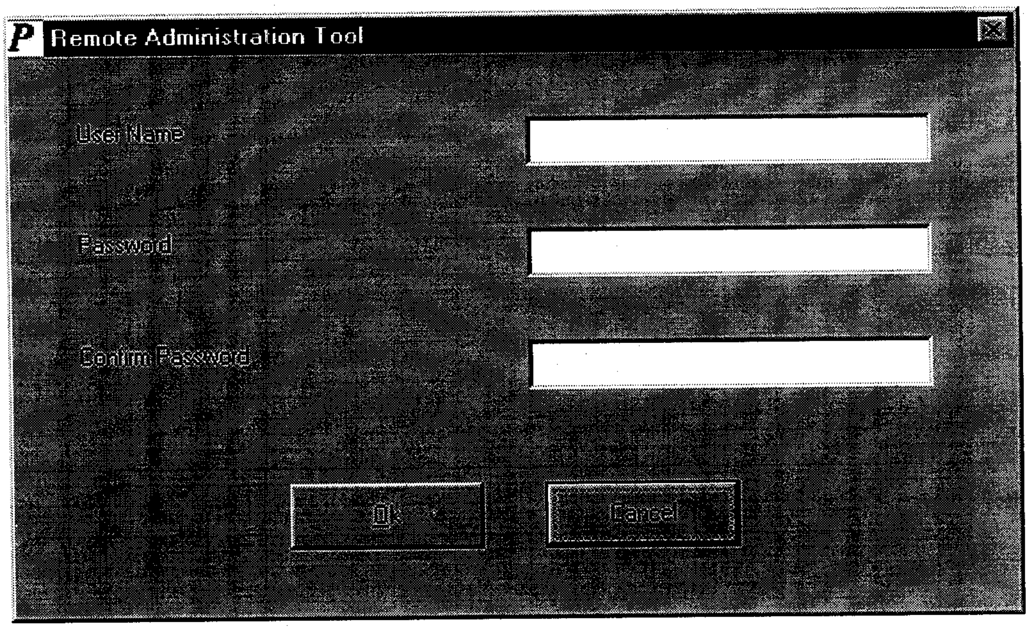


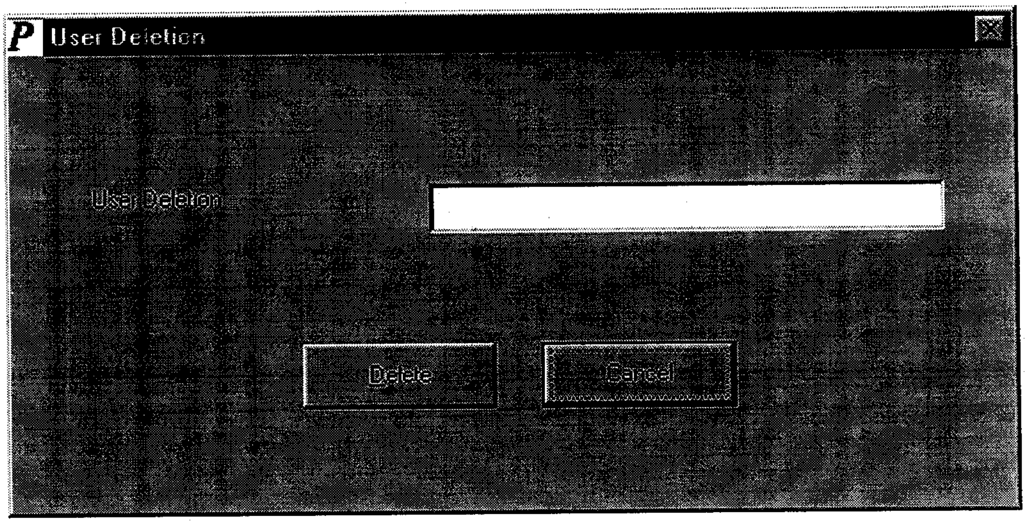
This Settings restricts the sites which contains the key words present in the deny list


SMTP Fire Wall Configuration
Allow Only List
Current Allow Only List
Add >>
Domain Name to Add
<< Remove
Deny List
Current Deny List
Add >>
Domain Name to Add
<< Remove
Restrict Outgoing Mailsize to (5-1024 KB)
KB
Ok
Cancel


DNS Fire Wall Configuration
Allow Only List
Current Allow Only List
Domain Name to Add
Add >>
<< Remove
Deny List
Current Deny List
Domain Name to Add
Add >>
<< Remove
Ok
Cancel


POP3 Fire Wall Settings
Restrict incoming mail size to (5-1024 KB)
KB
Ok
Cancel


Remote Administration Tool
User Name
Password
Confirm Password
Ok
Cancel


User Deletion
User Deletion
Delete
Cancel

B) DESCRIPTION OF TECHNOLOGY

SIMPLE MAIL TRANSFER PROTOCOL (SMTP)

1. Introduction

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.

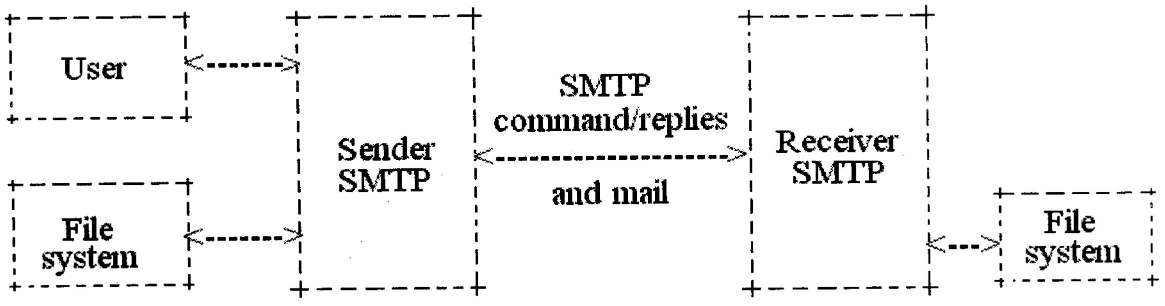
2. The SMTP Model

The SMTP design is based on the following 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 one. 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 command.

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 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 OK reply; if not, it responds with 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 responses with an OK reply. The dialog is purposely lock stepped, one-at-a-time.

The SMTP provides mechanism for the transmission of the mail; directly from the sending user's host to the receiving user's host when the hosts are connected to the same transport service or via one or more reply SMTP-server when the source and destination are not connected to the same transport service.

The model of SMTP is given in the below figure.



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. Here we assume that host Alpha contacts host Beta directly.

S: MAIL FROM:<Smith@Alpha.ARPA>
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: Hello
S: How are you?
S: How is your parent?
S: <CRLF><CRLF
R: 250 OK

3. THE SMTP SPECIFICATIONS

3.1 SMTP COMMANDS

3.1.1 Command Semantics

The SMTP commands define the mail transfer or the mail system function request by the user. SMTP commands are character string terminated by <CRLF>. The command codes themselves are alphabetical character terminated by <SP> if parameters follow and <CRLF> otherwise. The syntax of the mail box must conform to receiver site conventions.

MAIL (MAIL)

This command is to initiate a mail transaction in which the mail data is delivered to one or more mailboxes. The argument field contains a reverse-path.

The reverse-path consists of an optional list of hosts and the sender mailbox. When the list of hosts is present, it is a "reverse" source route and indicates that the mail was relayed to each host and the list (the first host in the list was the most recent relay). This list is used as a source route to return non-delivery notices to the sender.

RECIPIENT (RCPT)

This command is used to identify an individual recipient of the mail data; multiple recipients are specified by multiple use of this command. The forward-path consists of an optional list of host and a required destination mailbox. When the host is present, it is a source route and indicates that the mail must be replayed to the next host on the list.

DATA (DATA)

The receiver treats the lines following the commands as mail data from the sender. This command causes the mail data from this command to be appended to the mail data buffer. The mail data may contain any of the 128 ASCII character codes.

The mail data is terminated by a line containing only a period, which is the character sequence "<CRLF>.<CRLF>". This is the end of the mail data indication.

VERIFY (VRFY)

This command asks the receiver to confirm that the argument identify a user. If it is a user name, the full name of the user (if known) and the fully specified mailbox are returned.

This command has no effect on any of the reverse-path buffer and the forward-path buffer of the mail data buffer.

RESET (RSET)

This command specifies that the current mail transaction is to be aborted. Any stored recipients and mail data must be discarded, and all buffer and state table cleared. The receiver must send an OK reply.

QUIT (QUIT)

This command specifies that the receiver must send an OK reply, and then close the transmission channel.

REPLY CODES BY FUNCTION GROUPS

500 Syntax error, command unrecognized

502 Command not implemented

503 Bad sequence of command

250 Requested mail action okay, completed

451 Requested actions aborted: error in processing

354 Start mail input; end with <CRLF>.<CRLF>

554 Transaction failed

552 Requested mail action aborted: exceeded storage allocation

POST OFFICE PROTOCOL (POP3)

INTRODUCTION

The Post Office Protocol – version 3 (POP3) is intended to permit a workstation to dynamically access a mail drop on a server host in a useful fashion. Usually, this means that the POP3 protocol is used to allow a workstation to retrieve mail that the server is holding for it. 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.

All responses are terminated by a CRLF pair. Responses may be up to 512 characters long, including the terminating CRLF. There are currently two status indicators: positive ("+OK") and negative ("-ERR"). Server 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 mail drop. And the session enters TRANSACTION state. In this state, the client requests action on the part of 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.

The commands involved in a POP3 session are as follows

- APOP
- DELE
- LIST
- NOOP

- PASS
- QUIT
- RETR
- RSET
- STAT
- TOP
- UIDL
- USER

Example of a POP3 session

```
S: <wait for connection on TCP port 110>
C: <open connection>
S: +OK POP3 server ready <1896.697170952@abc.mtview.ca.in>
C: APOP mrose c4c933bac560ecc979e58001b3e22fb
S: +OK mrose's mail drop has 1 message (120octets)
C: STAT
S: +OK 1 120
C: LIST
S: +OK 1 message (120 octets)
S: 1 120
S: .
C: RETR 1
S: +OK 120 octets
S: <the POP3 server sends message 1>
S: .
C: DELE 1
S: +OK message 1 deleted
C: QUIT
S: +OK Dewey POP3 server signing off (mail drop empty)
C: <close connection>
S: <wait for next connection>
```

HYPER TEXT TRANSFER PROTOCOL (HTTP)

The Hyper Text Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information system. HTTP has been in use by the World Wide Web global information initiative since 1990. The HTTP protocol is a request/response protocol. A client sends the request to the server in the form of request method, URL and protocol version, followed by a MIME-like message containing request modifier, client information and possible body content over a connection with a server. The server responds with a status line, including the message protocol version and a success or error code, followed by a MIME-like message containing server information, entity Meta information, and possible entity-body content.

Most HTTP communication is initiated by user agent and consists of a request to be applied to a resource on some origin server. In the simplest case, this may be accomplished via a single connection (V) between the user agents (UA) and the origin server (O).

```
Request chain------------------------→
UA------------------V-----------------O
←--------------------- Response chain
```

A more complicated situation occurs when one or more intermediaries are present in the request/response chain. There are three common forms of intermediary: proxy, gateway and tunnel. A proxy is a forwarding agent, receiving requests for URL in its absolute form, rewriting all or part of the message, and forwarding the reformatted request towards the server identified by the URL.

Request

A request message from a client to a server includes, within the first line of that message, the method to be applied to the resource, the identifier of the resource and the protocol version in use.

```
Request = request-line
                *(general-header | request-header | entity-header)
CRLF
[ message-body]
```

Method

This Method token indicates the method to be performed on the resource identified by the Request-URL. The method is case-sensitive.

```
Method        =       "OPTIONS"
                      | "GET"
                      | "HEAD"
                      | "POST"
                      | "PUT"
                      | "DELETE"
                      | "TRACE"
```

The following HTTP/1.1 headers are hop-by-hop header:

- Connection
- Keep-Alive
- Public
- Proxy-Authenticate
- Transfer-Encoding
- Upgrading

Reply Codes

- Successful 2xx
- Redirection 3xx
- Error 4xx
- Server Error 5xx

DOMAIN NAME SYSTEM (DNS)

The DNS is used to map a name onto an IP address; an application program calls a library procedure called resolver, passing it the name as a parameter. The resolver sends a UDP packet to a local DNS server, which then looks up the name and returns the IP address to resolver, which then returns it to the caller. Armed with the IP address, the program can then establish a TCP connection with the destination, or send it UDP packets.

Every domain, whether it is a single host or a top-level domain, can have a set of resource records associated with it. For a single host the most common resource record is just its IP address, but many other kinds of resource records also exist. When a resolver gives a domain name to DNS, what it gets back are the resource records associated with that name. Thus the real function of DNS is to map domain names onto resource records. A resource record is a five-tuple. The format is

Domain_name Time_tto_live Type Class Value

The type field tells what kind of record this is The most important types are listed below:

Type	**Meaning**	**Value**
SOA	Start Of Authority	Parameters for this zone
A	IP address of a host	32-Bit integer
MX	Mail Exchange	Priority, domain willing to accept email
NS	Name Server	Name of a server for this domain
CNAME	Canonical name	Domain Name
PTR	Pointer	Alias for IP address
HINFO	Host description	CPU and OS in ASCII
TXT	Test	Uninterrupted ASCII text

C) GLOSSARY OF TERMS

HTTP	-	Hyper Text Transfer Protocol
SMTP	-	Simple Mail Transfer Protocol
POP3	-	Post Office Protocol
DNS	-	Domain Name System
PPP	-	Point to Point Protocol
PAM	-	Pluggable Authentication Modules
URL	-	Uniform Resource Locator
TCP	-	Transmission Control Protocol
UDP	-	User Datagram Protocol

Bibliography

1.Patrick Naughton and Herbert Schildt, "*The complete Reference Java 2 Third Edition*", Tata Mcgraw Hill.

2.Michael Morrision, "*Java1.1 Unleashed*", et al Technomedia.

3.E.Balagurusamy, "*Programming with Java 2nd Edition*", Tata Mcgraw Hill.

4."*Computer Based Tutorial on JAVA made easy*", Inndsoft Empowering Technology an SSI Company.

5.Glenn L. Vanderburg, "*Tricks of the Java Programming Gurus*", et al Technomedia.