A study of computer-based message system

Chun Shan Cheung

University of Wollongong

UNIVERSITY OF WOLLONGONG
COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

This work is copyright. Apart from any use permitted under the Copyright Act 1968, no part of this work may be reproduced by any process, nor may any other exclusive right be exercised, without the permission of the author.

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

Recommended Citation

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au
A Study of Computer-Based Message System

A thesis submitted in partial fulfilment
of the requirements for the award of the degree of

MASTER OF SCIENCE (HONOURS)
(Computing Science)

from

THE UNIVERSITY OF WOLLONGONG

by

CHUN SHAN CHEUNG  B.S.Sc.(Hons) C.U.H.K.

Department of Computing Science
1985
ABSTRACT

A Computer-Based Message System (CBMS) is a form of communication by using computers as mediators between users. It provides a non-interactive form of communication which has proved to be its major advantage. A CBMS also provides a convenient and efficient means of communication for users who are geographically separated.

This thesis is intended to be a general study on CBMS's. References will be made to two such message systems, namely, "EAN" and "ACSNET". Emphasis will be placed on how the various aspects of a CBMS are designed and implemented in these two message systems.
ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my supervisor, Prof. J. Peck, for his constant encouragement and advice throughout the course of this research. Prof. Peck untiringly read through this thesis many times and commented on the text of it. His comments were of great value.

My thanks go also to Prof. Juris Reinfelds, chairman of the Department of Computing Science, and to Mr. G. Stafford for their comments.

I want also to thank Dr. N. Gray and Dr. F. Hille, coordinators of postgraduate students, for advising me on my research direction during my stay at the University of Wollongong and all the teaching and support staffs of the Department of Computing Science for providing an environment that was conducive to research.

I am also indebted to the Department of Computer Science of the University of British Columbia for allowing me to work on the EAN software and advising me on the testing of it.

Finally, not the least acknowledgement goes to my wife, Man-Chee, for her support and consideration throughout the course of my research.
# TABLE OF CONTENTS

1. Introduction ....................................... 1
   1.1. A Case For Computer-Based Message Systems .... 1
   1.2. The EAN Message System ........................ 3
   1.3. The ACSNET .................................... 4

2. A Functional Model for Computer Based Message Systems 5
   2.1. Organization of a CBMS ........................ 6
      2.1.1. The User Agent ............................ 6
         2.1.1.1. Composition ............................ 7
         2.1.1.2. Filing ................................. 7
         2.1.1.3. Submission ............................. 8
         2.1.1.4. Retrieval .............................. 8
         2.1.1.5. Management ............................. 8
      2.1.2. Mail Handler ............................... 9
      2.1.3. Gateway Mail Handler ....................... 10
   2.2. System Operation ............................... 11
      2.2.1. Message Transfer ........................... 11
      2.2.2. Message Storage ........................... 12
   2.3. Protocols and Interfaces ........................ 14
   2.4. EAN ........................................... 16
      2.4.1. The User Service of EAN .................... 16
         2.4.1.1. Composition ............................ 17
         2.4.1.2. Filing ................................. 18
         2.4.1.3. Submission ............................. 18
2.4.1.4. Retrieval ....................................... 19
2.4.1.5. Management ................................... 19
2.4.1.6. The Profile File ................................ 19
2.4.2. The Message Transfer Service of EAN ........... 21
2.5. ACSNET ............................................ 22
2.5.1. The User Service of ACSNET ...................... 22
2.5.2. The Message Transfer Service of ACSNET ....... 23

3. Mapping to the Reference Model for Open System Interconnection (RM/OSI) ....................... 27
3.1. Reference Model for Open System Interconnection .................................................. 27
3.1.1. Application Layer ............................... 28
3.1.2. Presentation Layer ............................. 29
3.1.3. Session Layer ................................ 29
3.1.4. Transport Layer ............................... 29
3.1.5. Network Layer ................................. 30
3.1.6. Data Link Layer ................................ 30
3.1.7. Physical Layer ................................. 30
3.2. Layering the Functional Model .................... 30
3.3. Layered Architecture of EAN ....................... 33
3.4. Layering in ACSNET .............................. 35

4. Naming and Addressing ............................... 40
4.1. Naming ............................................ 41
4.2. Addressing ....................................... 41
4.3. Addressing Schemes .............................. 43
4.3.1. Address Logic Schemes ......................... 43
4.3.2. Message Logic Scheme ........................ 44
4.4. Naming and Addressing Schemes in EAN .......... 45
4.4.1. Naming Scheme in EAN .......................... 45
4.4.2. Addressing Scheme in EAN ....................... 46
4.5. Naming and Addressing in ACSNET .................. 47
4.5.1. Naming Scheme in ACSNET ....................... 47
4.5.2. Addressing Scheme in ACSNET .................... 47
4.5.3. Node and Domain ................................. 48

5. Routing ............................................. 51
5.1. Dimension one: By whom and where are the routes being decided ......................... 51
5.1.1. Source Routing ................................ 52
5.1.2. Incremental or Hop-by-hop Routing ............... 52
5.1.3. Hybrid Routing ................................ 53
5.2. Dimension Two: How often is the routing information changed .......................... 53
5.2.1. Non-adaptive algorithm ......................... 53
5.2.1.1. Flooding ................................ 54
5.2.1.2. Fixed Routing ............................... 54
5.2.1.3. Split Traffic Routing ...................... 55
5.2.2. Adaptive Routing ............................... 55
5.2.2.1. Centralized Adaptive Routing ............... 56
5.2.2.2. Isolated or Local Adaptive Routing ....... 56
5.2.2.3. Distributed Adaptive Routing ............... 57
5.3. Design Considerations ............................. 57
5.4. Routing algorithm in EAN ........................ 58
5.5. Routing Algorithm in ACSNET ..................... 61

6.1. Functions rendered by the Name Server .......... 67
6.2. Architecture of the Network Name Server .......... 69
  6.2.1. Variation One .................................. 69
  6.2.2. Variation Two .................................... 70
  6.2.3. Variation Three .................................. 70
  6.2.4. Variation Four ................................... 72
  6.2.5. Variation Five ................................... 73

6.3. Information in the Network Identification Data Base .............................................. 74

6.4. Security and Privacy ................................. 74

6.5. Personal Identifier .................................. 75

6.6. Network Name Server in EAN .......................... 75

6.7. Directory Service in ACSNET ......................... 77

7. Multi-Media Messages Systems ........................ 78
  7.1. Major Problems of Multi-Media Messages .......... 79
    7.1.1. Encoding Scheme ................................ 79
    7.1.2. Disparity Between Recipients ................... 80
    7.1.3. Constraint on Protocols ........................ 80
  7.2. Current Status of Multi-Media Messages Systems . 81

7.3. Multi-Media Messages in EAN and ACSNET ............. 81

8. A Lesson From EAN and ACSNET ......................... 83
  8.1. Local Configuration of EAN .......................... 83
  8.2. General System Architecture ......................... 84
    8.2.1. Expansibility ................................... 84
    8.2.2. Connectivity .................................... 85
  8.3. Naming Scheme and Directory Service ............... 86
  8.4. Addressing and Routing ............................... 87
  8.5. End User Functionalities ............................. 88
1. Introduction

A Computer-Based Message System (CBMS) is a form of communication by using a computer as the mediator between users. It initially evolved to meet the needs for interpersonal communications. Subsequent improvement in networking technology makes it possible for users on different computers and in different geographical locations to exchange messages with each other via a network of interconnected computers. "ARPANET" is one of these early CBMS's.

One of the major features of a CBMS is that it provides a form of non-interactive communication between users. This implies that when the sender is sending a message, the recipient need not be present and ready to accept the message. The message is simply delivered to the recipient's computer. It can then be picked up at a later time. On the other hand the sender need not be present when a reply is sent from the recipient.

This thesis is intended to be a general study on CBMS's. References will be made to two such message systems, namely "EAN" and "ACSNET". Emphasis will be placed on how the various aspects of a CBMS are designed and implemented in the two message systems.

1.1. A Case For Computer-Based Message Systems

A CBMS does have some advantages to make it popular. One major advantage comes from the non-interactive nature of
communication in a CBMS. The telephone provides a handy form of communication between geographically separated users. Immediate reply can be obtained on the telephone. However the pre-requisite condition of telephone communication is that both the caller and the receiver must be present at the two ends of the line. They need to be ready to communicate. This condition may not always be satisfied especially when the users are in different geographical locations. The sender may not know exactly where the receiver is located. The receiver may not be ready or want to accept any telephone call. This is especially the case when users are located in different time zones. Another point is that an immediate reply may not always be essential. As a result the non-interactive nature of a CBMS becomes one of its most important features.

A CBMS also provides a cost-effective means of communication. It is found that communication via a CBMS is cheaper than using the telephone or the postal service ([24]).

In the business environment, a record of what has been said is important. A record is obviously not readily available when a telephone is used. A CBMS, on the other hand, readily provides a record of what has been communicated.

The advantages of a CBMS are summarized by Holden as follows: "Computer Message Systems are a means to communicate and record communication in a timely manner without locating or interrupting the recipient and without undue
1.2. The EAN Message System

The EAN message system was developed by the Distributed Systems Research Group in the Department of Computer Science at the University of British Columbia in Canada. It is a distributed message system for exchanging interpersonal mail. It is based on the CCITT X.400 protocols for message handling services. The EAN message system was designed to meet the following goals.

a. "... message handling is critical to the efficient functioning of any enterprise, a message handling system must have sufficient functionality to support messaging throughout and between enterprises." ;

b. "Messages may be much more than simple text, since they can consist of data of all kinds, including graphics and voice encoding. File transfer may be regarded as an application of messaging." ;

c. "The system will have to function in a wide range of ... configurations." ;

d. "The network links available over which messages can be sent ... must include public networks, and telephone lines, but also include local area networks of many different kinds." ;

e. "Interconnection to other message handling systems"
should be possible ([21]).

The EAN message system was originally implemented to run on the UNIX* 4.2BSD operating system. It has been ported to other systems running different versions of UNIX and also other operating systems such as IBM VM/CMS or VAX VMS. At the University of Wollongong, it has been ported to two Perkin Elmer 3230 computers running UNIX version 7.

1.3. The ACSNET

The ACSNET is a network developed at the University of Sydney in Australia. It is designed as a "loosely coupled network of heterogeneous machines and has a purpose and function similar to that provided by the UUCP network ..." ([31]). ACSNET provides a store and forward message passing service. It is designed to run on various versions of UNIX.

*UNIX is a Trademark of Bell Laboratories.
2. A Functional Model for Computer Based Message Systems

The definition of a functional model for a Computer Based Message System (CBMS) is a very important step in its evolution. It serves to define the various functionalities of a CBMS and at the same time helps to improve our understanding of it. It also benefits the inter-connection of different CBMS's.

Several functional models for a CBMS have been proposed ([18,19,46]). It is based on these papers that we try to discuss a general model in this chapter. The model that we are going to discuss will define the "organization of the system", the "operation of the system" and the "protocols and interfaces needed" ([18] p37).

Before we discuss the model itself, it should be noted that it is a model for a distributed CBMS rather than a centralized one. Currently a large number of commercially available CBMS's are centralized and usually operate on a single machine or allow only connection of similar machines. But it is expected that these CBMS's will evolve to become distributed and offer wider connectivity. There are several factors contributing to this trend. One important factor is that "the success of CBMS hinges on broad connectivity" ([46] p8). No matter what type of user is using the CBMS there is always a need and desire to send or receive messages from users on another CBMS. It is therefore desirable to connect to other varied CBMS's which may be
geographically separated and also functionally and operationally dissimilar. A centralized system may not be able to satisfy this requirement. The distributed nature of a CBMS means that the users can have the privacy and security of a small system and at the same time broad connectivity. It also enjoys the typical advantages of a distributed system, for example work loads are shared between nodes in the system.

2.1. Organization of a CBMS

In this section we try to decompose a CBMS into functional entities and discuss the repertoire of their functions.

A typical CBMS is made up of at least three functional components or entities, namely the "User Agent" (UA), the "Mail Handler" (MH) and the "Gateway Mail Handler" (GMH). These components co-operate with each other to form the message system.

2.1.1. The User Agent

The User Agent (UA) is the user interface process. It is through the UA that the user can interact with the message system. The functions of the UA can be classified into 5 categories. They are "MANAGEMENT, SUBMISSION, RETRIEVAL, COMPOSITION, FILING" ([42] p137).
2.1.1.1. Composition

Composition refers to the creation of new messages. The user will be able to create both the "envelope" and the "body" of the message. A user should be able to create messages in either free format or some pre-defined format. By pre-defined format we mean that the user can invoke "a library of standard forms" either on a personal basis or a system basis ([42] p137). In this case the user needs only to fill in some variable fields in order to create the message. These standard forms may be announcements, reports etc., and are pre-defined by the users or the system administrator. This library of standard forms is quite common in word processing packages. The composition process should include some preliminarily checking of the message format, e.g., the presence of a recipient address. Format checking becomes more important in the case of multi-media messages. This will be discussed in a later chapter.

2.1.1.2. Filing

Filing refers to the keeping of the messages for the users. The concept of a "folder" provides a handy way for a user to manage and group his messages. A user should be able to move a message between folders or simply delete it from a folder. Deletion may be done as a two-stage process. The message is firstly logically deleted. Logically deleted messages are redeemable. These logically deleted messages can be physically removed from the system by some commands
or on a time basis.

2.1.1.3. Submission

When a user wants to send a message, the UA will negotiate with the serving MH about the submission procedure. The UA will also make sure that the message is submitted for sending. It is also the function of the UA to take delivery of any message from the MH and then to inform the user of newly arrived messages.

2.1.1.4. Retrieval

A user should be able to retrieve any messages that are composed or received. One should also be able to print the messages or save them to some other files outside the message system.

Messages should be retrieved by some characteristics, such as sender, recipient, subject, date or even keywords. In this way a user can retrieve messages without a concern for message identification.

2.1.1.5. Management

Management refers to the functions such as registration with the "Name Server" (refer to the chapter on Directory Services), changing a password for a mailbox, maintaining private distribution lists or aliases. Another important management function is to provide the user with the ability to set or select some default or automatic actions. These
actions may include automatic reply to incoming messages or forwarding of incoming messages to another mail box. If encryption is available in the system, the UA may have to negotiate with the key management authority for assignment of an encryption key in the case of a public key system.

2.1.2. Mail Handler

A Mail Handler (MH) is "responsible for the transfer of messages to and from a specific set of user agents and for the identification of system addresses of the recipients of the messages" ([19] p28). In general the functions provided by the MH can be classified into "handling submission and delivery of messages", "address identification" "route selection" and "relaying messages with other MH's".

The MH handles submission and delivery of messages in conjunction with the UA. It takes the message submitted by the UA and makes it ready for transfer. On the other hand, when a message arrives whose destination is a user served by the local MH, the MH should then deliver it to the corresponding mail box and inform the UA that the message has arrived. The MH should also be responsible for informing the sender of the message whether it has been successfully delivered.

The second function of an MH is "address identification". When sending a message, the user should not be required to supply the exact system address of the reci-
pient. The user will simply supply the address in a user-oriented standard. It is the responsibility of the MH to map the address in a user-oriented format to the actual system address which may be in a system-oriented format. The identification process may involve the "Network Name Server" and the "Network Identification Data Base". This will be further discussed in later chapters.

The next function of an MH is "route selection". Once the system address of the recipient is identified the MH should then select the appropriate route to transfer the message to its destination MH. This selection may be the identification of the actual absolute path or just the next MH on the path. It depends on the type of routing algorithm used (see chapter on Routing).

The final function of an MH is to relay messages to other MH's. In a CBMS there are usually a number of MH's. Each MH serves a specific group of users. The usual case is that on one computer there is an MH which serves all the users. These MH's co-operate with each other in order to transfer messages from one user served by an MH to another user served by another MH. Thus "a mail network is formed by the union of logically connected mail handlers ..." ([19] p29)

2.1.3. Gateway Mail Handler

"In each mailing network, there exists at least one
gateway mail handler, which represents half of the gateway between a mail network and any other mail network" ([19] p29). This gateway mail handler (GMH) serves as the interface between different "mail networks". It converts the internal message format of a mail network to an inter-mail network format. This inter-mail network format is agreed with the mail network to which the GMH is interfacing to or is agreed upon on a system-wide basis. The GMH also "supports addressing and message routing at the inter-net level" ([18] p38). As compared to the normal MH, the GMH needs to know about the GMH's of other mail networks while the normal MH needs only to know about the MH's in the same mail network.

As mentioned before a mail network is the union of logically connected MH's. Now with GMH's connecting different mail networks, we say that "a computer mail [message] system consists of a set of inter-connected mail networks" ([19] p29). A computer message system can thus be represented graphically as in fig. 2.1.

2.2. System Operation

Having discussed the functions of the various components of a CBMS, we now proceed to the operation of the system.

2.2.1. Message Transfer

In order to send a message from one UA to another, a
three-phase operation is involved. The first phase is the message submission or posting phase. The user invokes the UA to compose a message or nominate an existing message for submission. The UA then passes the message to the serving MH. The MH starts the second phase of the operation, the message transfer. This is also a two-step process. The first step is the address identification. As mentioned before it maps the destination address in a user-oriented format supplied by the user to the actual system address. The next step is the message transfer. This involves route selection and the actual routing of the message to its destination MH. The third phase begins when the message arrives at its destination MH. The MH will firstly deposit the message to the "mail box" of the user. A mail box is a place where messages are stored first before they are "picked up" by the user. A mail box is usually allocated on a per user basis. The MH should then inform the UA of the arrival of the message.

2.2.2. Message Storage

Before we leave the subject of system operation, we will discuss two different implementations of the storing of messages. The first implementation is that the messages received or sent by a user are stored physically in the storage under the user's account. When a message is being submitted, it is copied physically to the storage under the MH for further processing. The second implementation is that
all messages are kept in the system's storage. This can be viewed as a data base of messages. What is stored in the user's storage is in fact keys or pointers into the message data base. When the user submits a message, only the key or pointer is passed to the MH. There is no need to copy the message physically.

In the first implementation more storage space may be required as a single message addressed to several recipients is physically copied to each recipient's storage. However, the management of a user's messages is less complex because it is localized to the user's own storage.

Storage space usage is more efficient in the second implementation, but the management of messages is more complicated. It is necessary to keep a data base of all the messages in the system. The first implementation is more suitable for a general purpose inter-personal messages system. For other special purpose applications such as banking or a military application, the second implementation is more appropriate. It provides a better management and control of messages. Messages can be kept for an unlimited period of time with less overhead. The message data base can be periodically transferred to off-line secondary storage, e.g., magnetic tape or by changing a disk pack. Retrieval of any past message is possible although operator intervention may be needed.
2.3. **Protocols and Interfaces**

Before we go on to discuss the protocols and interfaces needed in a CBMS, we try to define these two terms briefly. In order to have meaningful communication between two entities, a set of rules must be defined to govern the ways in which these two entities communicate. They must adhere to this set of rules. This set of rules is called the protocol. "Protocols are defined for peer-entity communication" ([39] p17). By peer-entities, we mean the entities that are performing similar functions in the system, e.g., two mail handlers are peer-entities.

The interface is the set of rules that govern the way in which adjacent entities communicate. Two entities are said to be adjacent if they request the service of each other directly. For example, a UA requests the service of an MH to transfer a message. The UA and the MH are said to be adjacent.

The Protocols and Interfaces defined in a CBMS should at least cover two areas. The first area is "the form in which the dialogue is established between communicating parties" ([19] p29). In other words, it defines how messages are exchanged between parties, e.g., how to establish a session between a UA and an MH before messages are exchanged. The second area is "the structure of the information exchanged between processes" ([19] p29). This describes the format of the messages exchanged.
Protocols and interfaces should be defined for at least three levels in a CBMS. For the end user there should be a set of service commands defined to compose, process, submit and retrieve messages. These define the dialogue between the end users and the UA. A set of protocols should also be defined for message submission and delivery. These describe how the UA can set up a session with the serving MH in order to submit and take delivery of messages. Finally the protocol for inter-network and intra-network message transfer is needed. This defines how MH's can transfer messages between each other and how GMH's can deliver or receive messages from other mail networks.

Besides defining how messages can be exchanged, it is necessary to define in what format the messages are structured. This format may be different at different points in the CBMS. A common approach is that a message consists of two parts, the envelope and the body. The message body contains the information that is to be exchanged between two communicating parties. The message envelope contains the control information such as addresses, delivery date, priority etc.

When the end user first composes the message, he needs to compose the body and fill in details on the envelope. Both the envelope and the body are presented to the UA for posting. The UA will then add another envelope to the message. This envelope contains control information for the
destination UA. The message is then passed on to the MH. The MH will also add its own envelope to the message. When the message arrives at the destination MH, the MH envelope will be removed and the remaining part of the message is passed on to the destination UA. The destination UA will then further remove the UA envelope before presenting the message to the end user. This process is shown in fig. 2.2.

2.4. EAN

In the architecture of EAN, "the format of the mail and its supporting protocols follows the international CCITT recommendations for Message Handling Service (MHS)" ([44] p1). In this Message Handling Service Model there is a collection of "User Agents" (UA's) and the "Message Transfer Service (MTS)" ([44] p1). This Message Transfer Service is provided by a number of Message Transfer Agents (MTA's) cooperating with each other. The MTA's relay messages and deliver messages to the intended recipients.

Functionally, EAN can be viewed as consisting of 3 components, namely "The Message Transfer Service", "The User Service" and the "Directory Service" ([44] p1). We shall confine our discussion to the Message Transfer Service and the User Service in this chapter while the Directory Service will be discussed in a later chapter.

2.4.1. The User Service of EAN

User Service is provided by "User Agents". These supply
the facilities for "composing, editing, filing, sending and receiving interpersonal messages" ([44] p2).

There are about 30 commands available for the end user. We shall not go into detail concerning these commands here. Interested readers can refer to the user manual of EAN ([44]). In this section we try to group the commands into the 5 categories of functions of a UA which are mentioned in previous sections. There will also be a brief review of the facilities provided under each category. The grouping of the commands can be summarized as follows:

Composition - compose, edit, exchange, reply;
Filing - accept, delete, file, folders, move, remove tidy, untidy, where;
Submission - forward, send;
Retrieval - close, get, open, print, list;
Management - drop, exit, find, help, install, quit, register, set, show, stop.

(It should be noted that this grouping is by no means definitive. It only serves to provide a convenient way to discuss the facilities available.)

2.4.1.1. Composition

Under EAN, a user can compose a message only in a free text format. There is not a "library of standard forms". A user has to fill in the fields on the envelope, such as "To"
or "Cc" etc., when composing a new message. If the command "reply" is used then the UA will fill in the envelope of the user according to that of the message to which the user is replying. In order to edit a message, a user can make use of the "exchange" command or the "edit" command. The latter command invokes a text editor as nominated by the user and the message can be edited by using that text editor.

2.4.1.2. Filing

Messages are filed into folders. Each user can have any number of folders. "Inbox" is the default folder into which all incoming messages will be filed. Messages can also be moved between folders.

Messages can be deleted from a folder. Deletion is a two-step process. The "delete" command only logically deletes the message while the "tidy" command will remove messages physically.

2.4.1.3. Submission

Two commands are available for submitting messages, namely "send" and "forward". "Send" refers to the submission of the current draft message while "forward" will submit the specified message.

Several options are available to the user during submission to condition the transfer of message. These include timed deliver, status report and filing.
2.4.1.4. Retrieval

All commands under this category refer to the currently opened folder. For instance, "list" will list all the messages in the current folder. The "list" and "print" command are for scanning and printing of existing messages. Messages can also be saved to files outside EAN.

2.4.1.5. Management

Commands under this category can be further divided into 3 groups. The first group deals with the interaction with the "Network Name Server". It includes "install", "drop" and "find". The second group manipulates the user's "profile file", including "set" and "show". The "profile file" is the place where a user can set some options to tailor his interaction with the UA. This will be discussed in the next section. The remaining commands either terminate the user's session with EAN, i.e., "stop", "quit" and "exit", or provide the online user manual, i.e., "help".

2.4.1.6. The Profile File

Each EAN user has a personal "profile" file. This file contains the default settings for some commands. When a user issues these commands without giving specific details, EAN will look into the profile file for the default options.

These options describe the default settings of 3 areas. The first area is the system default features e.g., editor,
pager and printer. The second area is the personal details and references. Aliases used to stand for system addresses of other users are kept in the profile file. Personal details such as postal address, telephone number are also kept here and they are used when the user registers with the "Name Server". Users can specify the fields to appear on the envelope in this profile file.

The third area concerns some default actions. These include "auto-forward" and "auto-reply" of all incoming messages; "folder" for automatic or selective filing of incoming messages.

Some interesting options will be discussed in more detail here. The first one is the "folder" option. This specifies the "automatic filing of incoming messages in folders depending on message header content" ([44] p23). For example the "folder" option could be set as follows:

```
folder secret_mail importance = high & from = mary
```

In this case all incoming messages with the header field "importance" equal to "high" and "from" field equal to "mary" will automatically be filed into the folder "secret_mail". Other messages will be filed to the default folder "inbox" or other folder as indicated by other "folder" options. This provides a very convenient way for archiving related messages.

The other more useful options are "auto-forward" and
"auto-reply". The "auto-forward" option, when set, will automatically forward all incoming messages to another mail box. The "auto-reply" will cause a pre-defined message to be sent back to the senders of all incoming messages. Although these two options may seldom be used, they do provide valuable services. They are particularly important when a user moves from one system to another. This user can then set the option of "auto-forward" in his original profile file such that all messages addressed to his original mail box will be forwarded to his new mail box. As an alternative, he can set the "auto-reply" option, so that a message describing his change of address will be generated in reply to all incoming messages.

2.4.2. The Message Transfer Service of EAN

The Message Transfer Service of EAN is the co-operative effort of a group of Message Transfer Agents (MTA's). An MTA is basically the MH that we have discussed in the previous sections. It "accepts messages and delivers them to the intended recipients" ([59] p1). When a message is received by an MTA it will try to locate the computer system on which the recipient resides. It also determines which data network reaches the destination computer system. The MTA then transfers the message across the appropriate data network. If the recipient's system is not directly connected with the local MTA, it will try to find a route to it and send the message on to that route (see chapter on Routing for the
routing algorithm of EAN).

A gateway can also be provided by an MTA. If the computer system on which an MTA is running can set up a connection with another mail network, then this MTA will be able to provide a gateway service into that other mail network. Messages addressed to a user on that particular network will be sent first to this MTA. It will then pass the message on to the gateway process. The gateway process will translate the message envelope to a form understandable by the delivery service in the destination mail network.

2.5. ACSNET

In this section we try to describe the working of the ACSNET according to the functional model discussed earlier. Basically, the ACSNET provides "a message passing service, from one host to another, possibly utilising intermediate hosts in a store and forward manner" ([31] p2). The functions which ACSNET provides are "similar to that currently served by the UUCP network" ([31] p2). In fact it is aimed as an alternative to the UUCP network.

2.5.1. The User Service of ACSNET

ACSNET provides a minimal set of services to its users. These are message submission and delivery. The "sendfile" primitive is for submitting a message for transfer while the "getfile" primitive is for taking delivery of messages. They all work on a file basis. User messages are created as
a file and are passed to ACSNET for mailing. Message delivery is also in the form of getting a file from the ACSNET.

No facilities are provided by ACSNET for the filing, management, retrieval and composition of user messages. The absence of these services reflects the difference in the design goals. ACSNET does not aim at providing a self-contained mailing system. Rather it aims at an open ended message or file passing service. It is up to the sending and receiving end to interpret the file transferred.

2.5.2. The Message Transfer Service of ACSNET

A message transferred between hosts is not addressed directly to the intended recipient. Rather it is addressed to a "handler" at the destination host. For example all mail is addressed to the "mail handler" and print files are addressed to the "printer handler". When the handler receives the message, it will interpret the envelope and take the assigned action. The mail handler for instance will accept mail, look in the envelope for the recipient's account and send a UNIX mail telling the user that mail has arrived for him.

During the submission process, a user in fact can specify the destination handler besides the recipient's address. The default handler is the mail handler.

Gateway services can also be added to the ACSNET. It is
indicated in the routing table that "a non-standard spooling program should be used to send messages over a designated link or to deliver a message to a particular domain [other mail network]" ([31] p6).
Fig. 2.1: A typical mail system

A Typical Mail System
**Fig. 2.2: The mailing process**

**The Mailing Process**

- **En**: Envelope added by the process "n"
- **eu**: end user
- **ua**: user agent
- **mh**: mail handler
- **B**: message body
3. Mapping to the Reference Model for Open System Interconnection (RM/OSI)

In the previous chapter we have discussed a functional model for a CBMS in general. In this chapter we see how this functional model can be mapped onto the Reference Model for Open System Interconnection (RM/OSI) as suggested by the International Standard Organization (ISO). This mapping to the RM/OSI not only facilitates the definition of various components in a CBMS but most important, the interconnection of mail networks. It provides a layered approach to the architecture of a CBMS.

The RM/OSI will not be discussed in detail in this thesis. Only a brief review will be given before we proceed to relate the functional model to the RM/OSI. An interested reader can refer to other documents ([11,25,34,39,60,62]) for a detail discussion on RM/OSI.

3.1. Reference Model for Open System Interconnection

The RM/OSI provides a layered architecture for computer networks. This layering technique has been in fact recognized in the discipline of computer science for quite a long time. It is also applied in other areas such as structured programming. In the RM/OSI, this layering concept allows "networks of open systems to be viewed as logically composed of a succession layers, each wrapping the lower layers and isolating them from the higher layers" ([62] p426). "The
basic idea of layering is that each layer adds value to services provided by the set of lower layers in such a way that the highest layer is offered the set of services needed to run a distributed application" ([62] p426). In other words, all the services needed by the highest layer, the application layer, are broken into smaller pieces and distributed among the lower layers. These services are partitioned and distributed in such a way that each layer is responsible for a specific and well defined set of tasks. The services needed for a particular layer to accomplish its assigned tasks are provided by the lower layers. Consequently this particular layer is able to provide the services needed by its upper layer.

There are seven layers defined in the RM/OSI. They are:

(1) Application Layer;
(2) Presentation Layer;
(3) Session Layer;
(4) Transport Layer;
(5) Network Layer;
(6) Data Link Layer;
(7) Physical Layer.

The Application Layer is said to be highest layer while the Physical Layer is the lowest one.

3.1.1. Application Layer

The Application Layer which is at the top of this layered architecture is the layer that directly interfaces with the application or end user. It serves the end user. "The other layers exist only to support this layer" ([62] p230).
It is from this layer that data exchanged in the network originates and exits.

3.1.2. Presentation Layer

The second layer is the Presentation Layer. This "provides the set of services which may be selected by the application layer to enable it to interpret the meaning of the data exchanged" ([62] p230). It is through the use of this layer that applications in different networks or systems can communicate with each other without giving rise to much difficulties or cost.

3.1.3. Session Layer

The Session Layer "coordinates the dialogue between the communicating presentation entities" ([39] p24). This coordination includes "Session Administration Services" and "Session Dialogue Services" ([39] p230). Session administration services are the binding and unbinding of the communicating entities. Session dialogue services are the management of data exchange between entities.

3.1.4. Transport Layer

The Transport Layer is the fourth layer. This "provides transparent transfer of data between session entities" ([62] p230). This transparent service relieves the session layer from having to ensure that data are transfer reliably and cost effectively.
3.1.5. **Network Layer**

The Network Layer provides a "transparent transfer of all data submitted by the transport layer" ([39] p22). It enables the transport layer to be completely independent of the way in which networks or systems are interconnected. It takes care of the routing and switching considerations.

3.1.6. **Data Link Layer**

"The purpose of the Data Link Layer is to provide the functions and procedural means to establish, maintain and release data links between network entities" ([62] p230). It is responsible for the reliable transmission of data between network entities. Mechanisms for error detection, and if possible, error correction, should be provided.

3.1.7. **Physical Layer**

"The Physical Layer is responsible for transparent transmission of bit streams across the physical interconnection of systems" ([39] p19). It deals with the mechanical, electrical and other characteristics of the physical connection between data link entities.

3.2. **Layering the Functional Model**

It is obvious that the services provided by all the seven layers in RM/OSI are essential to the proper functioning of a CBMs. However, it is not necessary that all the seven layers are supported in the CBMS itself. It is
possible that a CBMS resides on a private or public data network that serves not only the CBMS but also other applications. The CBMS is then built with the assumption that certain services are provided by the data network and therefore relieves itself from the provision of these services. It is generally acceptable to assume the existence of certain services and based on these services the functional model is defined. For example in other models ([18]), it is assumed that session services already exist and therefore the functional model only defines the top two layers of the total seven layers. However, in this thesis we do not rely on such a high level of abstraction. We assume that only minimal services, up to the data link layer is provided by entities outside our model and on which our model can rely. We then proceed to discuss how the functions defined in our model are distributed among the remaining five layers, namely the application layer, presentation layer, session layer, transport layer and the network layer.

"The Application Layer contains all functions dealing with the processing of messages by the users and the management of the services" ([43] pl66). These services are those provided by the UA to the end users and cover the message processing and management functions.

The "presentation layer contains all functions of structuring of message content" ([43] pl66). It structures the message into a standard form which is to be recognised
by the MH's or converts it back to the original format when the message is delivered to the end user. This is particularly important in case of multi-media messages (this will be discussed in a later chapter).

The "session layer contains the functions of establishment of a session between users and message processing ends [mail handlers]" ([43] p166). These functions include the setting up of a logical session between two end users and the handling of message transfer over the session. In order to establish a logical session with the destination MH it may be necessary to perform the "name to address" mapping. This maps the name of the recipient supplied by the user to the system address of the recipient. It may involve a query to the "Name Server" and access to the "Network Identification Data Base" (refer to the chapter on Directory Services). It is also necessary for this layer to take care of the mode of operation of the session established, e.g., whether it is "two way simultaneous" (TWS) or "two way alternate" (TWA) or one way interaction.

The Transport Layer and Network Layer will only come into the scene when the communicating MH's are not in the same machine. If the MH's concerned are located in the same machine or in fact the same MH then a local session can be established without passing through the transport and network layers (as shown in fig. 3.1).

"The Transport Layer is only concerned with the
transfer of data between session entities [Mail Handlers]" ([39] p23). It is not concerned with or aware of the actual network topology. The Network Layer takes care of the actual interconnection between the systems of the communicating MH's. The communicating MH's are not necessarily directly connected and therefore the network layer may need to perform some routing functions. Given the recipient's network address, the network layer will determine the route to transfer message to the recipient.

3.3. Layered Architecture of EAN

The design of EAN is based on a layered architecture. There are five layers in EAN. They correspond to the top five layers of the RM/OSI, i.e, the application layer to the network layer. However the first two layers of EAN are modified. They are called the "interpersonal message layer" and the "message transfer layer" instead of the application and presentation layer in RM/OSI.

The interpersonal layer is basically the application layer in the RM/OSI. The UA's in EAN are operating at this level. The services provided by the UA's at this layer are the interpersonal messaging services. Details of the services have been discussed in the previous chapter. The UA's communicate with each other by using the CCITT X.420 protocol. This protocol "specifies the format of the message envelope and the message content" ([59] p11).
The next layer is the message transfer layer. This is the layer at which the MTA's operate. This layer, however, is not merely the presentation layer. Besides the taking and delivering of messages to a UA, it does perform some other functions. It maps the destination address to the actual computer system on which the recipient can be found. If it turns out that the recipient is also residing on the same computer system and most likely served by the same MTA, then there is no need to pass on the message to other layers. The MTA will deliver the message locally without going through the other layers, i.e., the session, transport and network layers. However if the recipient is on a remote computer system, then the MTA will also determine which of the data networks will reach the destination computer system. After these mapping processes the MTA will pass on the message and the network information to the "Reliable Transfer Services" which is made up of the next three layers, namely, the session, transport and network layers.

As mentioned before the UA's communicate by using the X.420 protocol. That is to say the message, both the envelope and the body, submitted by the UA are in this format. The MTA's communicate with each other by using the CCITT X.411 protocol. If a message is destined for a remote system, the serving MTA will make up another envelope in X.411 format to encapsulate the whole message from the UA. As a result the message now consists of two envelopes which are used by the MTA and UA respectively. These envelopes are
independent, as "one protocol does not rely on the information known by the other in order to function properly" ([59] pl1).

The next three layers, the session, transport and the network layers, are called the "Reliable Transfer Service" in EAN. Each of these 3 layers do exist in EAN and their functions are basically that described in the RM/OSI. The protocols used by these layers in EAN are shown in fig. 3.2.

3.4. Layering in ACSNET

It seems that the design of ACSNET is not based mainly on the RM/OSI. As a result not all the functional layers in RM/OSI can be identified in it.

The top layer that can be identified in ACSNET is the application layer. It is called a "handler" in ACSNET terminology. There is one handler for each application. Existing handlers and their applications are as follows:

(1) Filer - file transfer ;
(2) Mailer - mail transfer ;
(3) Printer - remote print facilities ;
(4) Stater - network status handler ;
(5) Reporter - news handler ;
(6) Eanmailer - sending mail into EAN

Handlers (1) to (5) are supported by the original ACSNET software while (6) is specially written for interfacing to EAN.

There is a process called "receiver" in ACSNET which preforms the function of name to address mapping and
routing. The setting up of the actual network connection is the function of another process called "NNcall". These two processes constitute the session and network layer. After the connection has been set up the process "NNdaemon" will be invoked to transfer the data which functions essentially as the data link layer. The layered architecture of ACSNET is shown in fig. 3.3.
Fig. 3.1: Local and remote sessions
(adapted from [43])

Local Session

Non-Local Session
Fig. 3.2: Protocols in EAN
(adapted from [59])

Protocols in EAN

* - Protocols supported in the original EAN software
+ - Enhancement at the University of Wollongong
Fig. 3.3: Layered Architecture of the ACSNET

Layered Architecture of ACSNET
4. Naming and Addressing

For every message in a CBMS there should be at least three attributes associated with it. These attributes are the "name" of the recipient, the "address" of the recipient and the "route" along which the message can be sent to the intended recipient. The understanding of the subtle difference between these three attributes is important to our understanding and designing of a CBMS.

Shoch defines "name", "address" and "route" as follows:

"The name of a resource indicates what we seek, an address indicates where it is, and a route tells us how to get there" ([52] p72).

In other words the name is used to identify for whom the message is destined. The address identifies the location of the intended recipient. Finally the route identifies the path to travel in order to reach the recipient.

However, in quite a number of existing message systems, these three attributes are not distinctively separated. In some cases all three of them are combined. While in other cases the name and address are combined. An example is the old but still popular telex system. The telex number serves both as the name and the address of the recipient.

In this chapter we only discuss the naming and addressing issues while routing will be discussed in the next
4.1. Naming

A name is a symbol that "identifies some resource or set of resources" ([52] p72). One of the main reasons to use a name is that the sender of a message is more interested in for whom the message is destined rather than the actual location of the destination. The format of a name should therefore be oriented towards ease of use by the user instead of ease of handling by the CBMS. It should be in a user oriented format and preferably in a human-readable form.

A name, since it is user oriented, need not be meaningful to all users. It can be used to "identify processes, places, people, machines and functions or anything else that the user chooses" ([52] p72). In order for the name to be useful, some mechanism must be available to map the name to the address(es). The existence of such a mechanism does not necessarily mean that the name is bound to the address. "The name needs not be bound to the address until this mapping takes place" ([52] p72).

4.2. Addressing

An address is a data structure which identifies the location of the recipient. One of the most important properties of an address is that it must be unique. In order that an address can be unique, it must be drawn from a "uniform
An address space is the set of all the addresses. This address space can either be "flat" or "hierarchical". The social security number of a particular country is an example of a flat address space. It spans over the entire domain of addresses. Addresses drawn from such an address space do not indicate any relation between each other. The uniqueness of the address is system wide. An address space is hierarchical if addresses are assigned to certain regions of the network and sub-addresses to regions or subscribers within the region. In this case an address does indicate that it is within a certain hierarchy. The full address should still be unique, but a sub-address in this case need only be unique within its hierarchy. The telephone number with area code is an example of hierarchical address.

As mentioned before, a flat address does not indicate any relation between any two addresses and therefore carries no implication of the actual route. An hierarchical address may carry some routing implications. In general there should be some mechanism to map an address into an appropriate route. It is important that "the address needs not be bound to the route until this mapping takes place" ([52] p72). The association of an address and its appropriate route may change over time.
4.3. Addressing Schemes

By addressing schemes we mean the mechanisms of "specifying and interpreting information on messages as to finally bringing them to the attention of the the proper recipient" ([58] p58).

Tsichritzis distinguished two basic addressing schemes, namely "address logic" and "message logic" ([58]). The major difference between these two schemes is the role that is played by the message in the process of determining the recipient.

4.3.1. Address Logic Schemes

"In address logic schemes messages are completely passive" ([58] p61). The recipient of a message is solely determined according to the name or address supplied together with the message. In other words a message is simply a data structure that is passed from its originator to the destination.

Under the address logic schemes we can further identify three logic schemes, namely "routing logic", "sender logic" and "receiver logic" ([58] p62).

In the routing logic scheme the decision of forwarding and delivery of messages to the recipients rests with the routing stations. The sender only provides data about the recipients but "the receiver specifications are not neces-
sarily binding on the routing stations" ([58] p62). In the sender logic scheme, the address of the recipient is supplied by the sender. All addressing logic is under the control of the sender. The message system has no influence on the final destination. In the receiver logic, all addressing decisions rest with the receivers. Messages are broadcasted to all addresses. The receiving addresses will decide whether they want to accept the messages.

The three logic schemes are by no means mutually exclusive. Mixed schemes can be used in a single CBMS.

Most currently available message systems, whether computer based or not, make use of address logic schemes and in particular the sender logic.

4.3.2. Message Logic Scheme

Under message logic schemes, the message plays an active role in destination determination. "When a message arrives at an address, it [the message] obtains the address identification and has access to the information present in the address. The message, as an object has its own procedures to decide whether it should leave a copy of itself, to what addresses it should go next, and whether it should retain some information available to it" ([58] p62). In other words, the message has complete control on the destination and its processing at the destination.

Message logic schemes are rarely used, one example is
the intelligent message system "Imail" developed at the University of Toronto ([58] p76).

4.4. Naming and Addressing Schemes in EAN

Basically name and address are distinct entities in EAN. A user can use either of them in specifying the destination of a message.

4.4.1. Naming Scheme in EAN

Names in EAN can be classified into two categories, namely "Network User Name" (NUN) and "Alias". Among these two, only an alias can be used in a message as a specification of the recipient. An NUN is used only "by the Directory Service to help locate a user's mailbox [address]" ([59] p3). An NUN "consists of the user's full name and the name of the Organization to which he belongs" ([59] p3).

Aliases and their corresponding addresses, called "Network User Addresses" (NUA), are kept in the "profile" file of each user and maintained by the user himself. A user can use an alias in the envelope of a message wherever an NUA is expected. EAN will search for the alias in the profile file and perform the necessary mapping and replacement of the alias by the actual NUA.

The mapping of an alias to an NUA is not necessarily one to one. An alias can be mapped to several NUA's. This provides a form of private distribution list.
The NUN consists of the user's full name and the name of the organization of the user. It is used only when querying the Directory Service. The Directory Service will return the corresponding NUA to the user if the NUA is registered.

4.4.2. Addressing Scheme in EAN

The addressing scheme in EAN, like that of most existing CBMS's, is a sender addressing logic scheme. The sender supplies the address, called the "Network User Address" (NUA), to identify the recipient. The address space under EAN is an hierarchically partitioned address space.

The top level of the hierarchy in the address space is called a "domain" or "private domain". A "private domain can be thought of as an organization that renders particular services to its member" ([59] p3). It consists of a group of subdomains. A subdomain is a group of mailboxes and other subdomains. The number of levels of hierarchy is not fixed. The format of an NUA is as follows:

mailbox@subdomain_list.domain

where subdomain_list is the hierarchy of subdomains in which the mailbox resides. The elements in the subdomain_list are separated by a period '.' and are listed with the closest enclosing subdomain to the outermost enclosing subdomain from left to right. Examples of NUA's are shown in fig. 4.1.

A mailbox may be within many subdomains, but it is said to belong to only one subdomain, the smallest enclosing one.
A mailbox can not at the same time belong to 2 subdomains. 
The configuration shown in fig. 4.2 is not allowed in EAN.

4.5. Naming and Addressing in ACSNET

4.5.1. Naming Scheme in ACSNET

When addressing a message in ACSNET, the sender can either supply the actual address or an alias of the recipient. The setting up of aliases is, however, on a system basis rather than on a personal basis. In other words, it is usually the system administrator's responsibility to set up an alias. A user generally will not set up an alias that is catered specifically for his own use.

4.5.2. Addressing Scheme in ACSNET

The addressing scheme in ACSNET is again a sender logic addressing scheme. Addresses are drawn from an hierarchical address space. The concept of domain is used in ACSNET. The syntax of an address is specified below.

"The full syntax for an address is:-

address ::= userlist[@:]destlist
userlist ::= <user>[,<user>...]
destlist ::= explicit|broadcast|multicast
explicit ::= multicast!multicast[!multicast...]
multicast ::= {dest|broadcast}[,{dest|broadcast}...]
dest ::= <alias>|{<node>|<domain>}[.<domain>...]
broadcast ::= *[.{*|<domain>}]" ([14]).
4.5.3. **Node and Domain**

A "domain is a grouping of computers to support a common purpose" ([27]). Domains in ACSNET may or may not be hierarchical, but this hierarchical relation usually exists for practical purposes.

A "node" is the name of a computer in the network. ACSNET does distinguish between node and domain. Nodes are viewed slightly differently from domains. It is "a special case of domain" ([30]). A node, in fact, can be viewed as the lowest level domain though a node need not belong to any domain. In this case the node is visible to the whole network. A local node is one that belongs to a certain domain and it may or may not be visible to the other domains. In order to address these local nodes from an outside domain, the appropriate domain name must be appended to the address.

In ACSNET a node can at the same time belong to several domains. One of these domains is said to be the "primary domain" of the node. A "primary domain of a node is the smallest domain that encompasses all the nodes to which it is directly connected" ([27]). In other words, all the links to a node should be members of its primary domain or a domain below the primary domain in the local hierarchy. Examples of primary domains and nodes are shown in fig 4.2.
Fig. 4.1: Map of a Hypothetical Private Domain "pd"

Network User Address for:
- mailbox m1 is m1@s1.s2.s4.pd
- mailbox m2 is m2@s5.s6.pd
Fig. 4.2: Primary Domains and Nodes in The ACSNET

D1 may be the Primary Domain of node Na.
D2 may be the Primary Domain of node Nb.
D3 may be the Primary Domain of node Nc.
Node Nb belongs to two domains, i.e., D1 and D2.
5. **Routing**

The discussion of routing algorithms alone can be sufficient for a separate thesis. In this thesis we only attempt to give an overview of some aspects of the routing algorithms.

"A route is the specific information needed to forward a piece of information to its specified address" ([52] p73). A routing algorithm is an algorithm to identify the path along which a message should go in order to reach its destination. The actual routing or transfer action may be a one step process if the destination is directly connected. Otherwise, it may require a series of steps in order to route the message to its destination, e.g., in a store and forward system. In the former case, the route needs only to identify the direct connection to the destination. In the latter case, the route may need to define the path which consists of several intermediate switch points.

We try to discuss various routing algorithms through two dimensions. The first dimension is the place at which the intermediate routing decision takes place. The second dimension is "the time constant of the information upon which the routing decisions are based" ([52] p73).

5.1. **Dimension one**: By whom and where are the routes being decided

The actual path to transfer a message from its
originator to its destination may involve several interme-
diate "switching points" or mail handlers. Three categories
of Routing algorithms may be distinguished according to "the
place at which each intermediate routing decision is speci-
fied", namely "source routing", "incremental routing" and
"hybrid routing" ([52] p73).

5.1.1. Source Routing

The first category is called "source routing". The ori-
ginator has to "specify all the intermediate routing deci-
sions and include this information along with the data being
sent" ([52] p73). In other words, the absolute path of the
message is specified by the originator of the message. This
implies that the originator possesses comprehensive informa-
tion about the whole network or at least about the environ-
ment of the path. Intermediate nodes need only to follow the
routing information attached to the message and send it away
to the next point.

5.1.2. Incremental or Hop-by-hop Routing

Under this category, the originator of the message
needs only to specify the destination address and the next
intermediate node to transfer the message. Each of the
intermediate nodes then decides on the subsequent node of
the path. In this case the originator needs only to have
sufficient knowledge to transfer the message to the first
intermediate node. No complete knowledge of the whole net-
work is needed by the originator.

5.1.3. Hybrid Routing

This category is a combination of the previous two categories. The originator can specify some of the major intermediate nodes of the path. It is the responsibility of the intermediate nodes to decide on how to route the message through to the specified intermediate node.

5.2. Dimension Two: How often is the routing information changed

Now we turn to the second dimension of when and how frequently the routing tables are changed. These tables provide information for making routing decisions. Within this dimension, routing algorithms can be classified into two categories, the non-adaptive algorithm and the adaptive algorithm.

5.2.1. Non-adaptive algorithm

The non-adaptive algorithm is a deterministic routing policy. The routing decision is based on a pre-defined set of rules or routing tables. Routing tables are rather static. They usually remain unchanged for quite a long period and are only changed to reflect modifications in the network topology. This type of routing algorithm does not respond to dynamic network conditions. "No routing information is exchanged by the nodes, and no observations or meas-
measurements are made at individual nodes" ([38] p1802). The non-adaptive routing algorithm provides "the optimal routing for a network in steady state (no load fluctuations, or failures)" ([20] pl11). It is time invariant and most important of all, easy to implement and analyse.

Basically, we can have at least three techniques to implement non-adaptive routing, namely, "flooding", "fixed routing" and "split traffic routing" ([22] p17).

5.2.1.1. Flooding

According to the flooding technique, a node on receipt of a message will transmit the message over all the connected links, except the one that the message came from. It is hoped that the message will eventually reach its destination. Usually acknowledgement is necessary to confirm the delivery of the message. This flooding technique is in fact a sort of broadcasting method. However, in large and distributed computer networks this technique is not very practical and feasible due to the large volume of traffic generated.

5.2.1.2. Fixed Routing

This algorithm assumes a fixed network topology and a known traffic loading. The optimal routes for each possible destination are derived beforehand and stored as static tables in the nodes. The routing decision is basically a table look up procedure. One obvious deficiency of this
technique is its inflexibility, especially in the case of link failures. However it does provide a simple, efficient and easy to understand routing technique.

5.2.1.3. Split Traffic Routing

This technique may be employed when more than one path exists between a given source and destination. If there are \( n \) paths available for transfer of messages from a node \( A \) to another node \( B \), then the probability of using the path \( i \) \( (1 \leq i \leq n) \) is \( P_i \) such the sum of all \( P_i \)'s is 1. Nodes that utilized this method have to keep a table for each possible path, the probability for each path and possibly the past choice in order to determine the current choice. This technique may be more complicated than the previous two but it provides a better balance of traffic between paths and therefore reduces message delay.

5.2.2. Adaptive Routing

Adaptive routing algorithm is a stochastic or non-deterministic routing policy. A routing decision is based on a dynamic routing table. The routing table is changed dynamically to reflect the changes in the routing environment, e.g., traffic pattern or momentary link failure.

Based on the control mechanism of the adaptive routing we can further distinguish three different classes of adaptive routing, namely "centralized", "isolated" and "distributed" adaptive routing ([38] p1802).
5.2.2.1. **Centralized Adaptive Routing**

A central authority exists in the network. Changes in network connectivity and traffic flow are reported to this central authority. This central authority then "dictates the routing decision to the individual node in response to these changes" ([38] p1802). Routing computation is performed by the central authority. An individual node does not make any changes in routing decision in response to changed environment. It only updates its routing table according to that supplied by the central authority. This technique is appropriate when the central authority is a large and powerful processor with relatively less powerful nodes. The individual nodes are then relieved from the load of route computation and this leads to a better message transfer response. The central authority is most likely able to have a global picture of the network and therefore it may be possible to attain global optimization when performing the routing computation.

5.2.2.2. **Isolated or Local Adaptive Routing**

Nodes utilizing this technique operate independently. The nodes make exclusive use of locally available data to adapt to changing routing environment. Routing computation is performed at each node and based on local optimization decisions.
5.2.2.3. Distributed Adaptive Routing

This technique relies on the cooperation of nodes and the exchange of routing information between them. Routing tables are changed according to the information exchanged between nodes.

It is obvious that under this approach, overhead is involved in exchanging routing information between nodes. This technique, therefore, may not be very practical for very large networks if this exchange is frequent and large in volume.

5.3. Design Considerations

After presenting the different approaches to routing algorithms, we now look at the major factors that are to be considered when designing or choosing routing algorithms for a particular network. McQuillan identified several design considerations for routing algorithms ([35]). In this section the most important factors are discussed.

The first factor to be considered is simplicity of the routing algorithm. As the routing algorithm performs more functions or the network grows bigger, this factor tends to become more important. The routing algorithm should be simple enough to be understood. This is essential to tackle problems that arise or further improvement of the network or the algorithm itself. As the network grows larger, complicated or non-deterministic algorithms become undesirable.
The second factor is reliability. Under extreme conditions, for instance all links of a node are down, it is difficult to require the routing algorithms to be reliable. However, the routing algorithm should be able to adjust itself to momentary or prolonged malfunction of certain links. It should be able to redirect the traffic to any existing alternate route.

The third factor is a steady state solution. This requirement is that the routing algorithms should have a predictable behaviour given the same input and environment. The output or the routing decision should not fluctuate under the steady input and environment. This is again a very important factor for large networks.

5.4. Routing algorithm in EAN

The routing algorithm used in EAN is basically a non-adaptive, fixed and incremental algorithm. It is incremental because the sender only supplies the destination address and the source MTA does not need to know the exact path to transfer the message to the destination. The source MTA only knows which the next MTA is. It is then the next MTA's responsibility to send the message further on its path. The routing algorithm also does not respond or adapt to any dynamic changes in the network. No routing information is exchanged between MTA's. It is the system administrator's responsibility to change the various routing tables in order to take care of failures or changes in network topology.
In EAN there are altogether five tables for storing routing information in each MTA. These five tables are the "user table", "private domain table", "subdomain table", "connection table" and the "local identification table". There is one entry for each local user in the user table. Each entry contains information such as the mailbox name and the name of the process to invoke whenever a message arrives. "The private domain table contains one entry for each MTA that provides private domain gateway service" ([59] p8). There is also one entry for the local private domain. The subdomain table contains entries for all the directly connected subdomains and should also contain "entries for all the top level subdomains in the local private domain" ([59] p9). "The connection table contains information on how to establish a connection with another MTA" ([59] p7). Finally the local identification table, a single entry table, contains identification of the local MTA. For the hypothetical domain map shown in fig. 5.1, the content of various tables of the subdomain "uowcsa" are as follow:
The route identification process can be explained by using an example. Assume that a message is submitted to the MTA uowcsa with the address M2@uowcsb.uow. The steps taken by the MTA uowcsa are shown in fig. 5.2(i).

Step 1: Consult the private domain table to check whether the message is destined for a mailbox within the private domain.

Step 2: Consult the subdomain table to get the connection.
Step 3: Consult connection table to get the queue file and the connection information.

The message is then put into the queue file. The MTA will then invoke the corresponding transfer process described by the connection information to set up a connection with MTA uowcsb.

When the message reaches MTA uowcsb, the following steps are taken by the MTA to deliver the message. These steps are shown in fig. 5.2(ii).

Step 1: Ensure that the message is for the local private domain.

Step 2: Consult the subdomain table and find out that the message is for the local subdomain.

Step 3: Consult the user table to get the queue file and the notification program. The message will then be placed in the queue file and the notification program invoked to notify the user of the arrival of the message.

5.5. Routing Algorithm in ACSNET

An adaptive and distributed routing algorithm is employed in ACSNET. Link status information is exchanged between nodes. The routing tables of each node may then be changed accordingly. As for the specification of the abso-
lute path, the ACSNET routing algorithm is a hybrid routing algorithm. This means that the originator can specify some major intermediate nodes of the path. For instance such a route can be specified by an address as follow:

host-A!host-B!host-C

This address will cause the message to pass to host-A, host-B and then to host-C. However the route for host-A to pass the message to host-B and that from host-B to host-C are not under the control of the originator. It is solely determined by host-A and host-B respectively.

Basically there are two tables in each node to store the network information. They are the "network state table" and the "routing table". The network state table "contains for each known host, a list of the hosts to which it is directly linked, and the cost and current status of each of those links" ([31]). This table may be sent to another host on request or "broadcast to all hosts whenever a new link is added" ([31]).

The routing table is constructed from the network state table. "It indicates which links should be used to transmit a message bound for any host or domain on the net" ([31]). The routing algorithm based on these two tables can then choose the shortest and fastest path between nodes.
Fig. 5.1 : A hypothetical map of domain "uow".
Fig. 5.2(1): Routing procedure in subdomain "uowcsa".

subdomain table

<table>
<thead>
<tr>
<th>Subdomain</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>uowcsa</td>
<td>@local</td>
</tr>
<tr>
<td>uowcsb</td>
<td>uowcsb</td>
</tr>
</tbody>
</table>

connection table

<table>
<thead>
<tr>
<th>Subdomain</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>uowcsb</td>
<td>uowcsb</td>
</tr>
<tr>
<td>uowcsa</td>
<td>acs</td>
</tr>
<tr>
<td></td>
<td>nsg,...</td>
</tr>
</tbody>
</table>

recipient's address: m2 @ uowcsb . uow

private domain table

<table>
<thead>
<tr>
<th>Subdomain</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>uow</td>
<td>-</td>
</tr>
<tr>
<td>AcS</td>
<td>uowcsa</td>
</tr>
</tbody>
</table>
Fig. 5.2(ii) : Routing procedure in subdomain "uowcsb".

user table

<table>
<thead>
<tr>
<th>m2</th>
<th>m2</th>
<th>auto m2 &amp;</th>
<th>m2</th>
</tr>
</thead>
<tbody>
<tr>
<td>m4</td>
<td>m4</td>
<td>auto m4 &amp;</td>
<td>m4</td>
</tr>
</tbody>
</table>

recipient's address: m2 @ uowcsb . uow

private domain table

<table>
<thead>
<tr>
<th>uow</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcS</td>
<td>uowcsa</td>
</tr>
</tbody>
</table>

subdomain table

<table>
<thead>
<tr>
<th>uowcsa</th>
<th>uowcsa</th>
</tr>
</thead>
<tbody>
<tr>
<td>uowcsb</td>
<td>@local</td>
</tr>
</tbody>
</table>

1

2

3
6. **A Case for the Directory Service**

It is obvious that the most important function of a message system is the reliable delivery of a message to the intended recipient. This reliable transfer or delivery of a message can be ensured by protocols at various levels. But before all these protocols come into the scene, the message system must first be instructed to send the message to a certain address. On many message systems it is solely the sender's responsibility to supply the actual and correct address of the recipient. In other words the sender must know beforehand to whom the message should be sent and the address of the recipient. In most cases the only way for the sender to know all this is to contact the recipient by some means other than the message system. This is not always possible. Therefore it is much better if the message system itself can provide a "Directory Service" to its users, such as that of the telephone services. The user can simply look up (enquire) the directory for the address of the intended recipient. This is one of the reasons for having a server, which is called the Name Server (NS), in the message system to provide the directory services. It helps to locate a named object in the message system of the network.

Even if the sender can manage to get hold of the address of the recipient in some other way, it is still necessary for the sender to remember the address for future reference. Unfortunately, in most cases the addresses are
not so easily handled by the user not to mention the memorizing of it, e.g., 8489381@376.258. Internetwork connection tends to complicate the subject. This is why people keep their small pocket telephone book. This small pocket book is in fact providing directory services to its owner, though its service is limited but relevant in most cases. The keeping of this private directory is far from a good way to solve the problem. It just puts the responsibility on the user. One obvious deficiency is that it may not be sensitive enough to reflect changes of addresses. Therefore, another important reason for having an NS to provide the directory service is to "shield users from the idiosyncrasies of individual addressing and naming schemes" ([28] p316).

The importance of the NS can easily be overlooked. The absence of it may not have any visible effect on the throughput or delay time of the system. But in fact it may prevent the system from realizing its potential. The case is similar to a very big library without any cataloguing system. If every user knows exactly the location of the books that he wants then the library can function smoothly. But if this is not the case or the location of books is changed because of some reasons then it will be in total confusion.

6.1. Functions rendered by the Name Server

In order to provide the directory service, the NS should keep the minimum information on "Who-is-Where" ([15]
p21:3-2). It keeps a data base, or the "Network Identification Data Base" (NIDB), of any named object and its corresponding network address ([15] p21:3-1). The term "named object" is used here instead of "user" to emphasize that not all users of the system or the network are necessarily human users. They can be service programs, such as database management systems, or peripheral devices, such as printers. Therefore additional information may also be included to describe the property of the named object. It may include some personal details such as telephone number, if the named object is a human user. If the named object is a server process then some information about the service it renders should be included. This information may help the other users to locate certain resources in the network.

So far it seems that the NS serves only to answer a query from a user. This is only part of the picture. The NS can play a more active role in the message system. In some message systems or networks, e.g. the CSNET, the sender of a message need not supply the full network address of the recipient. Instead the sender need only specify "enough keywords to produce a unique match with a directory database entry" ([28] p316). The NS will then supply the actual network address during the mailing process.

Another important function of the NS is "Group Mail Distribution" ([15] p21:3-2). A distribution list is kept by the NS in the NIDB. This binds a group identifier to a set
of addresses. A "user could address a message to a group (such as an organization, committee or special interest group) without knowing who all the specific members are" ([15] p21:3-2). The mailing process working in co-operation with the NS "would find the group identifier in the data base and send a copy of the message to each individual who is listed as a member of the group" ([15] p21:3-2).

The NS can also provide "Mail Forwarding" services for a user whose address has changed. The NS keeps a "trace of past network addresses for users, so that messages with an old address can be forwarded to the new address. If a user is unknown on the host toward which a message is initially directed, the recipient's name can be checked against the central Identification Data Base for a forwarding address" ([15] p21:3-1).

6.2. Architecture of the Network Name Server

The architecture of an NS can be discussed according to the distribution of the NS's and the distribution of the NIDB's. Our discussion starts with a centralized version of the NS and then goes on to various variations by decentralizing the components of the NS.

6.2.1. Variation One

The first variation is a single NS with a centralized copy of the NIDB. All enquiries, registrations and updates are sent directly to this NS. The NS then processes the
enquiry and sends the answer back to the user or updates the NIDB accordingly. This is similar to a centralized data base management system.

6.2.2. Variation Two

A second variation is that there exist a central NS and a number of local NS's. The NIDB is still kept by the central NS and there is only one copy of it. It can be viewed that the NS's are organized in an hierarchical way. The function of the local NS is to perform some processing on the user's enquiry before sending it to the central NS. This processing may include changing the format of the enquiry so that it is compatible with that of the central NS. This is especially important in a heterogeneous network. The local NS also helps to release the "User Agent" or the user from the burden of having to know where and how to relay the request to the central NS.

It should be pointed out that it may not be necessary to have an NS on each node or host machine. It is more desirable to have nodes grouped together to form organizations. There is a single NS, the organization NS, to serve the organization and interface with the central NS. This arrangement may be more efficient and practical.

6.2.3. Variation Three

A third variation may be that there is still a central NS, a number of local organization NS's and a central copy
of the NIDB. But now each local organization NS also maintains a copy of its local organization NIDB. An analogy can be drawn here with the "Replicated Distributed Data Base System" ([6]).

The maintenance and distribution of these copies of the NIDB can be achieved in two ways. The first way is that the local NS sends all the registration data of its users to the central NS and receives back a copy of the NIDB of its local organization users. In this way the central NS will have a copy of the NIDB which contains the information of all the users of all the organizations in the network. The second way is that each local organization NS maintains its own local NIDB and sends a copy of it to the central NS.

There is a slight but not insignificant difference between these two approaches. In the former case the local NIDB can at most be equal to the part of the central NIDB as kept by the central NS about the local organization. Of course the local NIDB can be only a subset of that part of the central NIDB for the local may decide not to keep as much detail information as that in the central NIDB. In the second approach, the local NIDB may contain more information than the copy that will be sent to the central NS. That is to say, the local organization NS may decide to store more detailed information about the local user than that required by the central NS. The local organization NS can then provide more detailed information to its local users than that
provided by the central NS to users in other organizations. But in this case the minimum information that must be kept by the local organization NS must not be less than that required by the central NS.

It is difficult to point out which is a better approach as under different requirements they tend to show different degrees of suitability. The local organization NS which resides on a large and powerful computer may be able and may wish to store more information in the local NIDB. While for the central NS, the storing of information in the central NIDB in such detail may not be considered feasible. In this case the second approach may seem appropriate. A local organization NS on a small computer due to limitation on storage space and processing power may have to provide less thorough information to its users and prefer to keep less information in the local NIDB. For this case the first approach should be considered.

The keeping of a local organization NIDB by the local NS can help to resolve an enquiry about a local user without the help of the central NS. This reduces the work load of the central NS and reduces traffic. However enquiries that cannot be solved by the local NS will still need to be sent to the central NS.

6.2.4. Variation Four

The fourth variation is characterised by the absence of
a central NS and central NIDB. There is only a group of cooperating local organization NS's. Each of the NS's keeps only that part of the NIDB which pertains to the local users. There is no need to send a copy of this local NIDB to another NS. This is a completely "Partitioned Data Base System" ([6]). Whenever an enquiry cannot be solved locally it is necessary to pass it on to other NS's. The problem is how to single out the target NS to which this enquiry should be sent. One simple solution is to send this enquiry to the other NS's one by one until an answer is returned. Another way is to broadcast the enquiry to all the NS's. It is obvious that unnecessary traffic is created. But if in the real situation inter-NS enquiry is rare, or at least not frequent, this approach may be considered acceptable. A more detailed discussion of the actual NS looking up algorithm can be found in [[45]].

6.2.5. Variation Five

The last variation is an extreme case at the end of decentralization. Every local organization NS keeps a copy of the total NIDB of the network. There may or may not be a central NS. If there is one, its function is to receive copies of local NIDB, make up an NIDB of the whole network and send it to each local organization NS. If there is no such central NS, each local organization NS just broadcasts its local NIDB to all other NS's. Since every NS in the network has a copy of the network NIDB there is no need to send
any enquiry to another NS. This situation is in fact a completely "Replicated Distributed Data Base". ([6]).

6.3. Information in the Network Identification Data Base

So far it seems that the information stored in the NIDB serves only to bind a name to its network address and some of its properties. In fact other information may be stored such as "accounting and billing data", "access control to resources", encryption keys ([45]). In this case the NIDB serves not only the NS but also other network utilities.

6.4. Security and Privacy

Security involves two issues, namely "authentication" and "access control" ([45] p250). Authentication is concerned with the validation of the origin of the request to the NS. It should check to see whether the actual origin of the request is really as claimed in the request. Access control is concerned with the checking of whether the origin of the request really has the right to raise that particular request.

The keeping of a data base of personal information will inevitably give rise to the problem of privacy. It is important that a user can choose to be listed in the NIDB or unlisted. One should also be able to choose to supply all or only part of the information as required in the NIDB.
6.5. **Personal Identifier**

Some implementations of the NS and NIDB may require each entry in the NIDB to be uniquely identified by an identifier. As discussed in [[45]], a user's full name i.e., firstname-middlename-lastname, plus the domain name and the organization name may be used. However this does not completely eliminate the possibility of ambiguity. Therefore a "birthmark" is used whenever necessary. "A birthmark is any string that together with the user name, the domain name and the organization name, unambiguously identifies the user" ([45] p241).

Some other implementations may allow the existence of ambiguous cases. In this case, a query to the NIDB may result in more than one entry and it is up to the user to check for its validity.

6.6. **Network Name Server in EAN**

The functions provided by the EAN NS include the directory service and the keeping of distribution lists. Users can query the NS to find out the network user address and some other information of another user.

In EAN "the directory service is implemented by the Central Name Server and a number of Organization Name Servers" ([59] p10). An organization is a group of Message Transfer Agents. The organization NS need only be running on a single Message Transfer Agent in the organization. In the
current implementation, there is only a central copy of the
NIDB and it is managed and kept by the Central NS. An EAN
user issues the registration or enquiry request through the
User Agent which then "packages the request in a message and
sends it to the Message Transfer Agent supporting the Organ-
ization Name Server. The Organization Name Server adds the
name of the Organization to the message and forwards it to
the Central Name Server." ([59] p10)

In EAN, a user can use an alias in addressing a mes-
sage. It should be noted that the mapping of the alias to
the actual network user address is not part of the function
of the NS but rather it is done by the User Agent. A list of
aliases can be kept in the "profile" file of the user. When-
ever an alias is used in addressing a message, the User
Agent will look it up in the list of aliases kept by the
user and fill in the corresponding network user address. It
is the user's responsibility to set up the list of aliases
for the network user addresses that are used most fre-
quently. The user is also responsible for updating that
list.

There is basically no control over access to the direc-
tory services. Any user can request information about other
users from the NS as long as the other users have registered
with the NS. As for authentication of the request, this is
done by the normal message submission procedure.
6.7. **Directory Service in ACSNET**

The NS is not implemented in ACSNET. There is no service provided to the user to enquire information about another user in the ACSNET, although each host will keep a list of other hosts to which this local host can send mail. It is the user's responsibility to find out the actual network address of another user in the ACSNET.
7. Multi-Media Messages Systems

Traditionally, message systems are designed to cater for textual messages only, e.g., telegraph and telex. In general the character sets that are used by these systems are limited. For instance in the telex system only upper case alphabetic characters and a limited number of symbols can be used in the messages. This is obviously not adequate when users want to include graphs or pictures in the message. This is why facsimile is of interest for message systems. Using a facsimile machine both characters and pictures can be sent electronically. However it should be noted that facsimile does not distinguish between characters or pictures. All it does is to send the black or white characteristics of each point on the source document to the destination. This has been a fairly convenient way of transferring messages that contain both text and graphics. However it would be more efficient if characters could be encoded as characters by some coding schemes, e.g., ASCII, and graphics encoded by some graphics encoding schemes.

As technology improves, there is a need for message systems to be able to handle multi-media messages, i.e., messages that consist of not only text but also other media such as graphics or digitized voice. At present multi-media messages systems are still in an experimental stage. In this chapter we try to highlight some of the problems that are encountered when handling multi-media messages.
7.1. Major Problems of Multi-Media Messages

The major problems encountered by a CBMS when handling multi-media messages concern mainly the presentation level. Once a message is encoded, there should not be much difference between a multi-media or a single medium message except for the character set used. Problems arise at the level which is above the transmission level.

7.1.1. Encoding Scheme

One major difficulty is the lack of general tools for composition or generation of multi-media messages not to mention the editing of them. In order to create a message in a certain medium, an encoding scheme for that particular medium is needed. For text messages there are the ASCII or other coding schemes. For graphics and speech there are already some commonly acceptable encoding schemes ([12]). Basically there should not be any problem in creating messages in a single medium regardless of what the medium is. However problems arise when creating messages that consist of more than one medium. In general there is no efficient and standardized encoding scheme that can handle all possible media. If the media are to be encoded individually then another problem arises. The relation between the different media in the dimension of time is lost. There is generally no efficient and simple mechanism or encoding scheme that can specify the relation between the different media in the time domain. In other words, "the originator of a message
can not control or suggest how a message should appear to its recipient ..." ([13] p191). It is not easy to reconstruct the message once we separate the different media in the message.

7.1.2. Disparity Between Recipients

The second problem of multi-media messages is due to the disparity of the hardware and software between different recipients. This is especially important in the case of a distributed message system that connects heterogeneous machines. It is difficult to define what should happen when a multi-media message is sent to a recipient whose hardware cannot handle all the media. This is very likely to happen in a CBMS. This difficulty arises because users in a CBMS are not communicating interactively in real time. There is in general a time lag between the sending and receiving of messages. The sender and the receiver have no opportunity to negotiate the media parameter that should be used.

7.1.3. Constraint on Protocols

Multi-media messages also impose constraints on the protocols that can be employed by the message system. Since multi-media messages are most probably encoded in binary format, the message transfer protocol cannot simply assume that data consists solely of 7-bits characters or with special character sequences to mark the start or end of a message.
Another constraint is due to the size of the message. As multi-media messages can consist of different media, the size of a message may be several times greater than the usual text message. This is especially important when digitized voice is involved because the message can be very large. The protocols employed by the message system should be able to handle messages of this size efficiently. The routing algorithm should try to minimize the path of the message as intermediate nodes may be loaded by large messages. This is very important in store and forward systems. The protocols should also try to minimize the replication of messages. Error correction should also be provided to reduce the retransmission of messages.

7.2. Current Status of Multi-Media Messages Systems

As mentioned before, multi-media message systems are still in an experimental stage, e.g. the "DIAMOND" system ([16]) and the "AGORA" system ([41]). It is generally accepted that further research is needed in this area before any simple and effective solution to the problems can be devised.

7.3. Multi-Media Messages in EAN and ACSNET

The X.400 protocols used in EAN were designed to accommodate messages combining different media. This form of conversion needs further study. ACSNET was not designed specifically to handle multi-media messages. However, mes-
sages of any medium can be transferred by EAN and ACSNET. This is because the transfer processes in these systems do not examine the content of the messages. Both of these two message systems can therefore be used as a vehicle for transferring multi-media messages as long as the sender and receiver is responsible for the encoding and decoding of the messages. The sender and receiver should also know in advance what media are to be used in the messages.
8. A Lesson From EAN and ACSNET

In this chapter the two message systems, EAN and ACSNET will be discussed from the point of view of the lessons that can be learned from them. The discussion will be focused on the following aspects:

- General system architecture;
- Naming scheme and directory service;
- Address and routing;
- End user functionalities.

8.1. Local Configuration of EAN

In this section the local configuration of EAN at the University of Wollongong will be described. EAN was originally implemented on UNIX 4.2BSD. Since then it has been ported to different versions of UNIX and different machines.

At the University of Wollongong, EAN is running on UNIX version 7 on two Perkin Elmer 3230 computers. Most of the network interfaces that are supported in the original EAN software are not applicable with the exception of the ttxp connections. An additional network interface to the ACSNET is implemented.

For the ttxp connection, an Apple 1200 auto-dial modem is used. At the time of writing of this thesis the ttxp connection is running locally. The testing with the mother site, the University of British Columbia, was successful.

The ACSNET network interface is added to EAN such that
EAN, running on different machines, can relay messages via the ACSNET. A gateway into the ACSNET is also implemented. Making use of this gateway, an EAN user can send messages to other users in the ACSNET. (Detailed discussion of the porting process of EAN and the ACSNET interface are included in the appendix of this thesis.)

8.2. General System Architecture

In this section the general system architecture is discussed in terms of "expansibility" and "connectivity".

8.2.1. Expansibility

For a CBMS, expansibility can be expressed in terms of adding new end-users or mailboxes, and the adding of new nodes in the network. End-users can be added in EAN and ACSNET without affecting the normal functioning of the systems and other users.

The effort needed to add new nodes in EAN or ACSNET is also minimal. In the case of EAN, most probably the routing tables of the directly connected MTA will need to be updated. In ACSNET this update of routing tables can be automatic. The network will be informed automatically and the software will take care of updating the corresponding tables. It can be seen that a smooth growth of the network is ensured in both EAN and ACSNET.
8.2.2. Connectivity

One important factor affecting the success of a CBMS is wide connectivity. Connectivity refers to the connection with different message systems and the connection to different network interfaces.

Both EAN and ACSNET are not designed to work on a particular network interface. They are capable of interfacing to different physical networks. Several network interfaces are currently supported in EAN, e.g., X.25, ttxp, tcp/ip etc. The network interfaces between different MTA's are stored in the "connection table". The interface can be changed simply by changing the corresponding entry in the table. This means that adding or changing of a network interface will not affect the normal functioning of the interfaces to other MTA's.

In ACSNET the network connection information is parameterized. The connection program is passed as a parameter to the driving program "NNcall". Therefore the effort involved in changing or adding new network interface between nodes is small.

In our experience with the interfacing of ACSNET to EAN, we find that the two systems offer connection to other message systems quite readily. It should be noted that different protocols are used in the two systems. The CCITT X.400 protocols are used in EAN while ACSNET is using a pro-
tocol that is designed specifically for it. In the current interface an intermediate protocol is used. Messages exchanged between EAN and ACSNET are first converted to the ARPA A822 format before transporting into the other system. This format is chosen because it is readily accepted by both systems.

The X.400 protocol used by EAN "is ratified by the CCITT as a standard for ... Distributed Message Handling Systems" ([54]). This protocol is receiving international recognition and also recognition from different computer manufacturers ([1]). Therefore other message systems that are using the same protocol can be connected without much difficulty.

8.3. Naming Scheme and Directory Service

The ability of the sender of a message to use names is an important step towards the ease of use of a message system. In general it can be said that a naming scheme is particularly important and beneficial in the case of a large and distributed CBMS which serves a wide variety of geographically distributed users.

If a naming scheme is used in a CBMS, there is an obvious need to have a network name server to provide directory service. The role of this network name server should not be limited to the passive role of handling enquiries from users. Rather, it should play an active part in the name to
address mapping. This mapping algorithm should be built into the CBMS to form part of the address identification process. In this way the users need not first query the network name server to obtain the actual addresses of the recipients. Users can simply supply the names in the messages. The system is responsible for the interaction with the network name server so as to map the names to the addresses.

In the case of a large and distributed CBMS, a decentralized version of a network name server can serve the purpose better than a centralized one. This is because a large volume of enquiry traffic can be reduced by a decentralized version. Unnecessary delay of messages can also be avoided.

Currently a centralized version of a network name server is adopted in EAN. However it is indicated that a decentralized version will be implemented at some later stage. ([59])

8.4. Addressing and Routing

An hierarchically partitioned address space is used in both EAN and ACSNET. This is not a coincidence. The main reason is that such an address space is more suitable to distributed CBMS's than a flat address space. The use of domain and subdomain reduces the number of addresses that have to be kept by individual nodes. In general, if a partitioned address space is used, the addresses that have to be kept by a node are those directly connected nodes and the
other top level domains that can be reached via one of those directly connected nodes. This is important in a distributed CBMS. The whole address space can be very large, and in fact it can grow without limit as more nodes are connected. It is therefore not feasible for each node or even just one node to keep all the addresses.

As for routing algorithms, it is most likely that an incremental routing algorithm is more acceptable to a large and distributed CBMS. The reason is to reduce the volume of routing information needed by each node. In case of an incremental routing algorithm, each node need only to know how to pass the message onto the next node on its path.

For a large and distributed CBMS, a simple routing algorithm is preferred. However this does not necessarily imply that a non-adaptive algorithm should be adopted. A simple adaptive algorithm that can adjust to avoid a faulty node is desirable. In other words, the routing algorithm should have the adaptability to reroute messages to alternate routes in case one of the routes is blocked due to a faulty node.

8.5. **End User Functionalities**

The repertoire of end user functionalities of a CBMS should depend on the aim of the CBMS and the target users it is going to serve. In EAN there is a sophisticated set of commands available to manage, send and receive messages. On
the other hand, in ACSNET there are only send and receive primitives available. Even though there is such a difference in the end user functionalities available, it is not appropriate to conclude that every CBMS should provide a full set of commands for its users. Moreover, it is difficult to define what a full set is.

In the case of a distributed CBMS which serves different groups of users, it is difficult to have a single set of end user functionalities that can satisfy all these groups. An alternative solution is to have different end user subsystems built on top of the CBMS. Each of these subsystems is to provide a specific set of functionalities and serves a specific group of users. In this way each of these subsystems can be built to utilize the resources available. This partitioning of the end user functionalities of a CBMS into subsystems is a good way to satisfy different groups of users. However, it is essential that the CBMS is designed in such a way the subsystems can be built on top of it.

In EAN the set of commands is more or less fixed, but some of these commands can be personalized or be system dependent. They can be changed via the "profile" file. In ACSNET, because of the simplicity of the end user interface, subsystems can be implemented without much difficulty.

8.6. Research Directions and Conclusion

Currently, single medium messages can well be handled
by a CBMS. However, a multi-media CBMS is still very primitive and in an experimental stage. Research in this area is essentially the next important step in the evolution of CBMS's. An extension of the protocol for CBMS's is also necessary in order to accommodate multi-media messages.

Another interesting research direction is concerned with intelligent message systems. How a "message addressing logic" can be implemented such that the message itself becomes an active object interacting with message systems to determine its destination(s). This kind of message system could be very useful in some situations.

Early CBMS's were limited to research purposes and served only the academic community. There is obviously a demand for such services in the business sector or even amongst individuals. Consequently, public or commercial message systems have come into existence. These message systems can be accessed simply via the common telephone line. With the proliferation and rapid drop in price of microcomputers, more and more business companies and even individuals can afford to have the necessary equipment to access these message systems. This helps to increase the popularity of CBMS's.

Although CBMS's are becoming more and more popular, users of different systems may not always be able to communicate with each other. Different systems usually adopt different protocols and interfaces. Gateways are always
necessary to interconnect different CBMS's. However, looking at other forms of telecommunications such as telephone and telex, it is not difficult to realize that efficiency can be greatly increased if a standardized protocol is adopted by different CBMS's. Currently, there is still not a commonly accepted standard for CBMS's. Several organizations are working towards this direction. For example, the X.400 protocol for distributed message handling systems, which is recommended by the CCITT, is receiving international recognition. It would be beneficial if this could be a step closer to standardization of protocols for CBMS's.
APPENDIX A - Porting of EAN

A. Porting EAN to Version 7 UNIX

The original EAN software was implemented to run on UNIX 4.2BSD. The machines that are available at the University of Wollongong are two Perkin Elmer computers running Version 7 UNIX. EAN was therefore ported to Version 7 UNIX. In this appendix the problems encountered during the porting process are discussed.

A.1 Disparity between C compilers

The first task necessary in porting EAN to Version 7 UNIX is compilation of the whole software. It was found that the C compiler used to compile the original EAN software is different from the C compiler at the University of Wollongong. Though the language C is claimed to be highly portable, problems can still arise because of disparities between different versions of C compilers. As a result of this problem it was necessary to modify about 1000 lines of source code.

A.1.1 The #define Identifier

Some of the #define identifiers used in the EAN software have more than eight significant characters. The local C compiler does not distinguish such identifiers whose first eight characters are identical. In order to minimize the changes, if two identifiers clashed, then one of them is
modified such that the first eight characters are different from the other one. As a result, it was necessary to change about forty two identifiers. The header files and the #define identifiers which were confusing are listed in table A.1.

A.1.2 Name of Structure Member

The treatment of names of structure members by the local C compiler is strictly the Kernighan and Ritchie standard\(^*\) for "portable" C ([26] p.197). Names of members of different structures can only be the same if they are of the same offset within the structures. Each structure does not have its own table of member names. In other words, a member name of any structure can be placed on the right of the operator "." or "->". The compiler accepts this. These two features turned out to be quite devastating constraints, because many of the different structures in EAN have the same name for their members. It was therefore necessary to examine all member names and to change the source code that referenced the names.

Two options were considered. The first option was to write a program that resembled a parser to scan through all the source files to change the names accordingly. The second was to do the changes manually. The second option was adopted by the author since it seemed that the first option

\(^*\)According to the author's understanding, a number of more recent C compilers do not impose this restriction.
might take a longer time. Unfortunately, this turned out to be the wrong choice. Two factors accounted for this. The first one was that human fallability is not infrequent. There were a number of cases where the names were changed wrongly to the names of members of other structures. The second factor, the limitations of the local C compiler, made the first one worse. As mentioned before, the local C compiler does not maintain a table of member names for each structure. Therefore it will not complain about the use of the wrong member, as long as the name used is a member of some structure. So these human errors are left unnoticed even after the compilation. In some cases, a core dump resulted. With a core dump it was relatively easy to find out what was wrong. In other cases, the software just behaved strangely. These cases were the most difficult to tackle. In fact even in later stages of the porting the modified software was still haunted by these errors. Names of the structures and the members which were confusing are listed in table A.2.

A.1.3 External Variable Name and Function Name

As mentioned before, the local C compiler only allows the first eight characters of an identifier to be significant. Also, it does not distinguish between an external variable name and a function name. There is, for example, an external variable "Term_interrupt" which is defined in the file ~Ean/src/rts/term/external.c. (~Ean is assumed to
be the path name of the source files of EAN.) At the same time there is a function name "Term_interact" in the file "Ean/src/rts/term/interact.c. These names were therefore treated as the same by the compiler. Somehow the loader did not complain of the multiple definitions. It simply picked the wrong one during relocation. As a result every time the function Term_interact was called the software behaved strangely. It took considerable time for the author to realize that the problem was because of the two names.

A.2 Disparity between UNIX 4.2BSD and Version 7

The other problem during the porting came from the difference in system calls of the two versions of UNIX. The following system calls that are used in EAN are absent in Version 7 UNIX.

1. fcntl
2. flock
3. fsync
4. ftruncate
5. gethostname
6. setregid
7. setreuid

These system calls were either replaced by the appropriate calls in Version 7 or emulated for the purpose of EAN. The two system calls fsync and ftruncate are the most difficult ones. No good ways were conceived to replace or to emulate these two calls without going into the operating system itself. For fsync it seems that the only way is to use the system call sync in Version 7. Sync will cause
all information in core memory that should be on disk to be written out, not only a specific file. This is obviously not very appropriate. As for ftruncate, the best way seems to be to write a similar routine to truncate a file. This routine requires the file name to be the input parameter instead of the file pointer as in ftruncate. This implies some changes to the logic of the original EAN software, which is also not desirable. Fsync is used to ensure that the disk copy of the file is the same as the core copy. This reduces the risk of losing the updated core copy due to machine failure. The system call ftruncate is used in EAN to release some of the unused disk space in a file to the the system. In a mailbox file, if a message is physically removed, then the space occupied by this message can be returned to the system. On the other hand, if the space is not returned to the system, it can be used by future messages. Since good solutions for these two system calls have not been found and they are not critical to the normal functioning of EAN, only two stub routines are used currently. More research is needed to find out better ways to get around these problems.
<table>
<thead>
<tr>
<th>Header file</th>
<th>#define identifier changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>util/blk.h</td>
<td>BLK_LOCK_MASK</td>
</tr>
<tr>
<td>util/std/ascii.h</td>
<td>ASC_ISALPHANUM</td>
</tr>
<tr>
<td>util/io.h</td>
<td>IO_CTL_PTR</td>
</tr>
<tr>
<td></td>
<td>IO_CTL_PTR</td>
</tr>
<tr>
<td>rts/sess/interface.h</td>
<td>CTL_SYNC_MINOR_ACK</td>
</tr>
<tr>
<td></td>
<td>RC_ACTIVITY_INT</td>
</tr>
<tr>
<td></td>
<td>RC_ACTIVITY_DISCARDED</td>
</tr>
<tr>
<td></td>
<td>RC_ACTIVITY_REJECTED</td>
</tr>
<tr>
<td>rts/sess/local.h</td>
<td>RTS_REFUSE_BUSY</td>
</tr>
<tr>
<td></td>
<td>RTS_REFUSE_CANTRECOVER</td>
</tr>
<tr>
<td></td>
<td>RTS_REFUSE_VALIDATION</td>
</tr>
<tr>
<td></td>
<td>RTS_REFUSE_DEMODE</td>
</tr>
<tr>
<td></td>
<td>RTS_DEFAULT_CKPT_SIZE</td>
</tr>
<tr>
<td></td>
<td>RTS_DEFAULT_WINDOW</td>
</tr>
<tr>
<td></td>
<td>RTS_ABORT_LOCAL</td>
</tr>
<tr>
<td></td>
<td>RTS_ABORT_PARAMETER</td>
</tr>
<tr>
<td></td>
<td>RTS_ABORT_UNRECOGNIZED</td>
</tr>
<tr>
<td></td>
<td>RTS_ABORT_TEMPORARY</td>
</tr>
<tr>
<td></td>
<td>RTS_ABORT_PROTOCOL</td>
</tr>
<tr>
<td>rts/tran/local.h</td>
<td>READY_TO_SEND</td>
</tr>
<tr>
<td></td>
<td>READY_TO_RECV</td>
</tr>
<tr>
<td></td>
<td>READY_TO_ACK</td>
</tr>
<tr>
<td></td>
<td>CALLING_ID</td>
</tr>
<tr>
<td>mta/P1.h</td>
<td>P1_DEFERTIME</td>
</tr>
<tr>
<td></td>
<td>P1_DEFERDLIV</td>
</tr>
<tr>
<td></td>
<td>P1_REPORTMPDU</td>
</tr>
<tr>
<td>ua/language.h</td>
<td>PR_AUTO-FWRD</td>
</tr>
<tr>
<td></td>
<td>PR_PRINTER</td>
</tr>
<tr>
<td>ua/P3.h</td>
<td>P3_MESSAGE_TRANSFERRED</td>
</tr>
<tr>
<td>ua/UAL.h</td>
<td>RECEPTION_ENVELOPE</td>
</tr>
</tbody>
</table>
Table A.2: List of Structure Members Changed

<table>
<thead>
<tr>
<th>File name</th>
<th>Structure</th>
<th>member</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccitt/element.h</td>
<td>ENODE</td>
<td>id</td>
</tr>
<tr>
<td>ccitt/rw.h</td>
<td>EDESC</td>
<td>id</td>
</tr>
<tr>
<td>host/host/mta.h</td>
<td>SYS_WORK</td>
<td>next, work</td>
</tr>
<tr>
<td>mta/Plinterf.c</td>
<td>PLID</td>
<td>status, type, qname, st</td>
</tr>
<tr>
<td>mta/P3interf.c</td>
<td>P3ID</td>
<td>state, qname</td>
</tr>
<tr>
<td>mta/dist.c</td>
<td>CONNECTION</td>
<td>next, state, rtinfo</td>
</tr>
<tr>
<td>mta/maint/check.h</td>
<td>CHECK_LIST</td>
<td>file, links, status, next</td>
</tr>
<tr>
<td>mta/maint/pmpt.h</td>
<td>PMPT_CMND</td>
<td>name, cmd, doc</td>
</tr>
<tr>
<td>mta/rec.h</td>
<td>GENERIC_LIST</td>
<td>next, num</td>
</tr>
<tr>
<td>mta/rec.h</td>
<td>REC_FLDDESC</td>
<td>size</td>
</tr>
<tr>
<td>mta/route.c</td>
<td>CACHE</td>
<td>next, type, connect</td>
</tr>
<tr>
<td>mta/route.c</td>
<td>RTE_CONNECT</td>
<td>next, name</td>
</tr>
<tr>
<td>mta/store.h</td>
<td>STORE</td>
<td>fid, type, state, statelen</td>
</tr>
<tr>
<td>mta/store.h</td>
<td>STORE</td>
<td>file, end</td>
</tr>
<tr>
<td>mta/store.h</td>
<td>ST_QLINK</td>
<td>time, state, statelen, file</td>
</tr>
<tr>
<td>nsg/A822/A822.h</td>
<td>A822_DATA</td>
<td>name</td>
</tr>
<tr>
<td>rts/interface.h</td>
<td>RTS_DESC</td>
<td>io, state</td>
</tr>
<tr>
<td>rts(sess/local.h</td>
<td>SESS_DESC</td>
<td>io</td>
</tr>
<tr>
<td>rts tran/local.h</td>
<td>TRAN_DESC</td>
<td>io</td>
</tr>
<tr>
<td>ua/alias.h</td>
<td>ALIAS_DESC</td>
<td>next, alias, name</td>
</tr>
<tr>
<td>ua/command.c</td>
<td>SORTED</td>
<td>next, key, data</td>
</tr>
<tr>
<td>ua/folder.h</td>
<td>FOLDER</td>
<td>next, name, current, file</td>
</tr>
<tr>
<td>ua/folder.h</td>
<td>FSYS</td>
<td>content, current, list</td>
</tr>
<tr>
<td>ua/message.h</td>
<td>MESSAGE</td>
<td>content, envelope, reports</td>
</tr>
<tr>
<td>ua/message.h</td>
<td>MESSAGE</td>
<td>status, filed, headerloc</td>
</tr>
<tr>
<td>ua/msgset.h</td>
<td>MSG_SET</td>
<td>next, folder, list</td>
</tr>
<tr>
<td>ua/sequence.h</td>
<td>LIST</td>
<td>next, value</td>
</tr>
<tr>
<td>util/host/blk.h</td>
<td>BLK_DESC</td>
<td>status</td>
</tr>
<tr>
<td>util/host/key.h</td>
<td>database</td>
<td>status</td>
</tr>
<tr>
<td>util/std/bf.h</td>
<td>BF_REC</td>
<td>ptr</td>
</tr>
<tr>
<td>util/std/byte.h</td>
<td>BDESC</td>
<td>io</td>
</tr>
<tr>
<td>util/std/io.h</td>
<td>IO_DESC</td>
<td>status, ptr</td>
</tr>
</tbody>
</table>
APPENDIX B - Enhancements of EAN Network Interfaces

B. The Ttxp Connection and the ACSNET interfaces

The two EAN network interfaces used at the University of Wollongong are the ttxp connection and the ACSNET interfaces. The ttxp connection is supported by the original EAN software. A new modem driver was needed to handle the Apple 1200 auto-dial modem used locally. Notes on adding a new modem driver to EAN are given in Appendix C. The ACSNET interfaces are local enhancements to the EAN software.

B.1 ACSNET Network Interface

The ACSNET network interface is implemented such that EAN running on different machines can relay messages via ACSNET. A configuration of this interface is shown in fig. B.1. This interface is implemented as a non-standard gateway in EAN. Messages from EAN are firstly converted to ARPA A822 format. They are then passed to an intermediate process called "ACS.intrf" together with the destination addresses. Then the ACSNET "sendfile" primitive is used to transport messages into the ACSNET. The destination ACSNET handler is specified as a program called "eannmailer". When the messages arrive at the destination host, the "eannmailer" will be invoked by the ACSNET "receiver". Eannmailer will accept messages in A822 format from the ACSNET. It then converts the messages into X.400 format and sends them into EAN. It is in fact a modified version of the "mailer" in ACSNET.
If there are two MTA's, uowcsa and uowcsb, and they are connected via ACSNET. The corresponding "connection table" entry in MTA uowcsa for the MTA uowcsb should be as follow:

Connection Table

<table>
<thead>
<tr>
<th>Connection</th>
<th>queue file</th>
<th>network info</th>
</tr>
</thead>
<tbody>
<tr>
<td>uowcsb</td>
<td>_uowcsb</td>
<td>nsg,a822,acsmail</td>
</tr>
</tbody>
</table>

An additional element should be added to the array "A822_tbl" in the file ~Ean/src/nsg/A822/config.c as follow:

```c
#define ACSNET
	{ "acsmail", "/usr/lib/ean/ACS.intrf %R" },
#endif
```

B.2 ACSNET Gateway

A gateway into ACSNET is also implemented such that EAN users can send messages to ACSNET users. This is again implemented as a non-standard gateway in EAN. Messages are first converted to ARPA A822 format before being sent into the ACSNET. An intermediate process is written to accept the A822 messages from EAN and kickoff ACSNET "sendfile" primitive to send messages into ACSNET.

If a gateway into ACSNET exists in an MTA, the corresponding entries in the "private domain table" and the "connection table" are as follow:
Private Domain Table

<table>
<thead>
<tr>
<th>Private Domain</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>acsgate</td>
<td>acsgate</td>
</tr>
</tbody>
</table>

Connection Table

<table>
<thead>
<tr>
<th>Connection</th>
<th>queue file</th>
<th>network info</th>
</tr>
</thead>
<tbody>
<tr>
<td>acsgate</td>
<td>_acsgate</td>
<td>nsg,a822,acsgate</td>
</tr>
</tbody>
</table>

An additional element should also be added to the array "A822_tbl" in the file ~Ean/src/nsg/A822/config.c as follow:

```c
#ifdef ACSGATE
    { "acsgate", "/usr/lib/ean/ean/acsgate %R"},
#endif
```
Fig. B.1: Configuration of the ACSNET interface

Configuration of the ACSNET Interface
APPENDIX C - Notes on Adding a New Auto-Dialing Modem for the Ttxp Connection in EAN

C. Adding a New Auto-Dialing Modem for the Ttxp Connection in EAN

There appears to be no documentation available on how to add a new auto-dialing modem for the ttxp connection in EAN. This appendix aims at giving a brief description of what should be done in order that a new type of modem can be used for the ttxp connection.

We use the driving routines for the Apple 1200 modem as an example. First, an entry must be added to the array "acutable" in the file ~Ean/src/rts/term/acutab.c to indicate that this new type of modem is now supported in EAN. The entry should be as follow:

```
#ifndef APPLE_1200
   "apl200", ap_dialer, ap_disconnect, ap_abort,
#endif
```

The define tag APPLE_1200 should be defined as 1 in the file ~Ean/src/rts/term/term.h to indicate that this type of modem can be used. The first field "apl200" is the name of the modem. It is the name used in the field "at" in the file /etc/remote. The remaining three fields are the names of the three functions that are needed to handle the modem. They are used for dialing, disconnecting and aborting respectively. They should also be defined as external variables in the same file.
The dialing function "ap_dialer" is called as follow:

```c
ap_dialer(conn, num, acu)
CONN* conn ;
char* num ;
char* acu ;
```

The first parameter "conn" is a pointer to the structure CONN. The structure CONN contains status, file descriptor and various flags of the UNIX device used to set up the tttxp connections. It is passed to this dialing routine for logging purpose only, i.e., it can be used as a parameter when the logging routine Term_log is called by this dialing routine. The second parameter "num" is a character pointer which points to the telephone number that is to be dialed. The last parameter "acu" is the name of the dialing device to be opened for dialing. If the auto-dialing modem used is such that the dialing device is the same as the line that should be used to talk to the remote host, i.e., a straight through modem, such as the Apple 1200 or the BIZCOMP 1302, then this last parameter is redundant. When the dialing routine is called, the corresponding UNIX device has already been opened. There is no need to open the dialing device again in the dialing routine. The corresponding file descriptor is in the external variable "FD". This file descriptor "FD" should be used in the dialing procedure. It is important to note that in order to have the correct file descriptor placed in "FD", the corresponding entry in the file /etc/remote should contain the field "hw".
If the auto-dialing modem is not the straight through type, then the corresponding UNIX device should be opened within the dialing routine. In general the dialing routine should return the value of 1 to indicate successful dialing, otherwise 0 should be returned.

The routine ap_disconnect is used to hang up the modem in case of a normal termination of the call to the remote host. For any abnormal termination the routine ap_abort will be invoked. Both of these two routines are called with no parameters. The file descriptor of the dialing device should be in the external variable "FD" if the device is of the straight through type. Otherwise, the file descriptor should be stored by the dialing routine into a variable known to the other two.
Glossary of Abbreviations

ACSNET - A message transfer system developed in the University of Sydney.

ARPA - Advanced Research Project Agency.

ASCII - American National Standard for Coded Information Interchange.

CBMS - Computer-Based Message System.

CCITT - International Telegraph & Telephone Consultative Committee.

EAN - A distributed message system developed in the University of British Columbia.

GMH - Gateway Mail Handler.

ISO - International Standard Organization.

MH - Mail Handler.

MTA - Message Transfer Agent.

NIDB - Network Identification Data Base.

NS - Name Server.

NUA - Network User Address.

NUN - Network User Name.
RM/OSI - Reference Model for Open System Interconnection.

TWA - Two Way Alternate.

TWS - Two Way simultaneous.

UA - User Agent.
References


Proceedings of the Sixth Data Communications Symposium, Nov. 27-29, 1979, p18-25.


[27] KUMMERFELD, Bob and LAUDER, Piers, "Domain Addressing in SUN III", ACSnet software distribution document, University of Sydney.


