DESIGN CONSIDERATIONS FOR LARGE DISTRIBUTED ELECTRONIC MAIL SYSTEMS

A Thesis

Presented to

the Faculty of the Department of Computer Science

College of Natural Sciences and Mathematics

University of Houston, University Park

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Hui-Tung Yuen

May, 1987

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and appreciation to my thesis advisor, Dr. Wael Bahaa-El-Din for his valuable guidance and advice. In spite of his heavy work load, he always found time to speak with me when I needed guidance.

I would also like to thank the other members of my thesis committee, Dr. Pen-Nan Lee and Dr. Tiee-Jian Wu, for their interest in my research. They kindly provided comments and suggestions on my thesis.

I am particular thankful to my parents, Wai-Ping and Yuen-Wah Yuen, for their encouragement and support.

Finally, to many friends who have given encouragement and help, I thank each of them most sincerely.

DESIGN CONSIDERATIONS FOR LARGE DISTRIBUTED ELECTRONIC MAIL SYSTEMS

An Abstract of

a Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston, University Park

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Hui-Tung Yuen

May, 1987

ABSTRACT

Electronic mail is one of the most important means for communication and information exchange in internetworking environments. In this thesis, three methodologies for designing large mail systems are investigated, namely, mail systems with syntax-directed naming, mail systems with location-independent access, and attribute-based mail systems. Mail systems with syntax-directed naming identify users by names which are syntactically structured according to user locations. Algorithms for load balancing among mail servers, system reconfiguration, and efficient message delivery are developed and are tested using simulation. Mail systems with location-independent access allow users to access them from different locations. Procedures for keeping track of migrated users and redirecting their mail are presented. The attribute-based mail system provides maximum flexibility to users by allowing them to identify one or more mail recipients by attributes instead of only by precise names. It can also be used in mass distribution of electronic mail. An algorithm for efficient broadcasting and searching using Minimum-weight Spanning Tree (MST) is investigated. Criteria for evaluating electronic mail systems are presented. Simulation experiments are used to test the procedures and algorithms, and to study the performance of the three mail systems.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In order to facilitate communication, information exchange and sharing in todays' complex business environment, immense interconnections of public and private data networks have been established [TER 85]. Among the many services built upon these networks, electronic mail service is one of the most important and widely used facilities. Present mail systems vary from exchange of messages among users on the same computer system, to some large multinetworking distributed systems that involve hundreds of networks and thousands of users residing on different part of the world. For example, DARPA Internet consists of over 300 networks connecting most of the major U.S. universities, military organizations and computer corporations, and is still growing. The industry of electronic mail will increase tremendously in the near

future as systems become easier to use and less expensive.

Since communication is so important in our daily work, its efficiency will directly affect the performance, productivity, and competence of people and organizations in an increasingly complex world. Electronic mail system provides a convenient, efficient and reliable way of communication and information exchange and sharing. It offers the advantages of speedy delivery and rapid reply. Also the messages, which can be in the form of text or graphic information, are replicated and stored as written records. Moreover, the cost of electronic mail becomes less expensive than telephone communication in recent years. In this thesis we investigate the nature of electronic mail systems and explore design methodologies for large distributed mail systems.

1.1 ELECTRONIC MAIL SYSTEMS

Electronic mail system is composed of users, user interfaces, hosts, mail servers, mail forwarders and networks, as shown in Fig. 1.1.

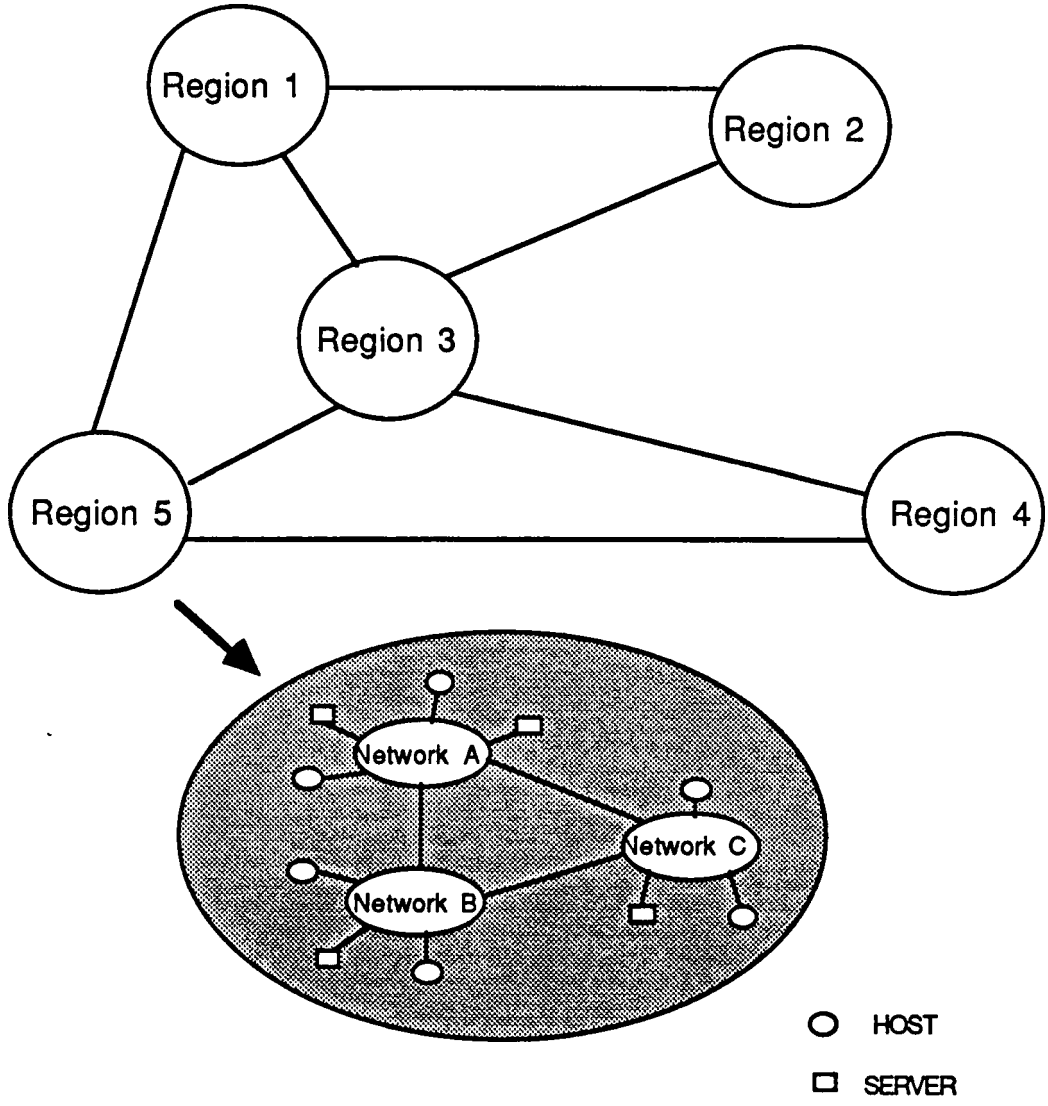

Fig. 1.1 Environment of electronic mail system.

A user or client is identified by a name which is used for the purpose of reading mail, accessing the mail services, and addressing his mail. This name may be different from the user's computer account. A host is a computer connected to the network. The user interface is a software package that interacts with the users and assists users in composing, sending and receiving, reading and filing mail. A mail server is a process responsible for obtaining address of recipients, sending, relaying and delivering messages to the mail recipients. It contains information about users, hosts, networks, and other servers. A mail forwarder is a host computer that can store and forward messages. The networks provide communication facilities for message transportation.

When a user wants to send a message, the message is first composed and formatted by the user interface. The user interface locates an active mail server, to which the message is submitted. The mail server then obtains the address of the recipient using name resolution scheme, and sends the message to the recipient's mail server through the forwarders using the communication services. Upon receiving a message, the server notifies the recipient (if possible). The recipient user retrieves the message from the server through the user interface, and can read and

file the message.

In this internetworking environment, each network may be totally unique in the sense that each has its own network protocol and own administrative and management policy. However, the many different networks have to operate and co-operate concurrently. The environment can be characterized as large and diverse. Every machine in the global network can communicate with other machines using some basic communication facilities such as virtual circuit and datagram services, routing and flow control. Such services can be provided by standard protocols such as the Transmission Control Protocol (TCP/IP). The mail system is built on the top of those facilities. Most of the mail traffic is within the local environments and at a very high speed while a small portion is internetwork traffic at a relatively slower speed.

1.2 MOTIVATION AND OBJECTIVES

Most of the current electronic mail systems are limited to a single corporation (Grapevine) or a group of institutions (ARPANET and CSNET). Aided by increasing uses of computers and powered by new technologies, future electronic mail systems are expected to connect everyone

with everyone else, just like today's telephone systems. The hardware used in electronic mail systems may range from small personal computers through workstations to minicomputers, mainframes, and supercomputers [QUA 86]. Significant work has been done in the area of naming and addressing in multinetwork systems ([COM 85], [TER 85], [PET 85]). However, most of the naming and addressing schemes are limited in flexibility that they have many constraints on system reconfiguration and expansion, the movement of objects. Moreover, algorithms for efficient load balancing and models for mass distribution of mail based on attributes are not investigated in previous work.

In this thesis, we present design methodologies for electronic mail systems in large and diverse distributed environment using three different approaches. The first one is a mail system with syntax-directed naming scheme. It emphasizes the hierarchical partitioning and distribution of the mail services using syntax-directed names. Algorithms for balancing loads among servers, and message delivery are developed. The second methodology is location-independent access to mail systems, which provides flexibility in accessing the mail services and user movement. It allows users to access the mail services through any host in the local region. The last system is

called attribute-based mail systems, which allow selective search of recipients and mass distribution of mail using some defined attributes.

There are many important aspects in designing large electronic mail systems. In this thesis, special considerations are given to the problem of naming and addressing, flexibility in reconfiguration and user movement, and mass distribution of mail using attributes. Criteria for evaluating the performance of mail systems are also developed.

1.3 THESIS OUTLINE

In this thesis, we discuss some important issues in designing large electronic mail systems for multinetwork environment. The major considerations are on naming and addressing of mail system users and mail boxes, balancing the load among mail servers, system reconfiguration and the problem of user migration and system growth.

Chapter 2 provides the background and basic concepts of electronic mail systems. Formal terminologies are defined. The fundamental procedures and mechanisms for large distributed electronic mail systems are listed. Four existing mail systems are examined.

Chapter 3 is the core of the thesis. It presents the three methodologies for designing large electronic systems. Important mechanisms and procedures of naming and addressing, load balancing, reconfiguration, flow control and user migration for each methodology are discussed.

Chapter 4 discusses criteria for evaluating mail systems and compare the performance of different approaches.

Finally chapter 5 is the conclusion of this thesis. Some areas for future research are proposed.

CHAPTER 2

LITERATURE SURVEY

In this chapter, the background for designing large distributed mail systems is given. First we introduce the basic concepts. Then the fundamental procedures and mechanisms required in a distributed mail system are listed. Finally, several existing distributed mail systems are described.

2.1 BACKGROUND

The followings are basic terminology and concepts.

2.1.1 Objects

The distributed electronic systems are conceptually view as a collection of objects [PET 85]. Objects are the physical and logical entities in the system. They can be

either active or passive. In the electronic mail system, the major objects we are referring to are users, mailboxes, and servers.

2.1.2 Names

Each object is associated with a name for the purpose of identifying, locating, and accessing it in distributed environment. It should be noted that names and addresses are closely related, and addresses are also names that indicate the physical location of the objects. Names can be translated into addresses when accessing the objects. The names need not be bound to addresses until mapping occurs.

2.1.3 Group Names

One objective of computer mail system is to share information among the users. Therefore, recipients of mail messages should not be restricted to individuals only. They can be groups or distribution lists. Distribution lists represent named sets of recipients. The grouping may be based upon organizational structure, geographical locations, job responsibilities, projects or interests. When a message is addressed to a distribution list, each

member in the list will receive a copy of the full message or a summary that each member can request the full message t be sent if he is interested. This is a convenient way to distribute messages to groups of people in a distributed environment.

2.1.4 Name Structure

Names are usually structured as a set of alphanumerical symbols separated by delimiters. A name is a string composed of a set of symbols chosen from a finite alphabet [PET 85]. A general form of name is "X.Y.Z.", where X, Y, Z are name tokens and "." is a delimiter. The name tokens X, Y, Z correspond to some characteristics or attributes of the object. Each existing distributed system has its own naming convention. For example, the original ARPANET users are identified by a flat name, and its present names are in the form of "user@host". Grapevine uses a hierarchical name in the form of "F.R", where F is the unique user-id and R is the registry name. Names in UUCP is like "host1!user@host2" which represents a source-route naming convention.

2.1.5 Characteristics Of Names

Since names are used to identify objects in the system, the set of names used should be simple, sharable and easily understandable by human beings. Another concern about names is the degree of ambiguity and uniqueness. A name is unambiguous if it identifies at most one object. A name is unique if it is the only name for an object. The same name cannot be used to refer to several different objects but several non-unique names may identify the same object. Therefore, names must be unambiguous but not necessarily unique. Actually all computer systems enforce the uniqueness rule on the names of user-ids. Therefore, many mail systems use the user-id of the systems as the mailbox addresses. A name is said to be global or absolute if it has consistent meaning for all users in the system regardless of their locations in the environment. On the other hand, a name is said to be relative if its interpretation varies with some state information.

2.1.6 Name Space And Context

In order to have a uniform naming scheme, each system has naming convention which is a set of rules that govern the syntactic representation and semantic interpretation of names used in the system. The set of names complying with a given naming convention is called the name space.

Furthermore, names always exist within some contexts. A context is the domain in which a name is valid. In distributed system, the name space is partitioned into contexts, often along geographical or organizational boundaries. A name can belong to more than one context and contexts may be nested.

2.1.7 Distribution Of Name Space

In a single computer system or single network environment, the size of its name space is very small that the name space can be contained in a single centralized database. However, in a large distributed system, a single centralized database is too inefficient to use and manage. Therefore, the name space is partitioned into some easy manageable contexts and distributed among servers so that no server needs the complete knowledge of the whole space.

The name space can be partitioned into levels or hierarchies. For example, Grapevine divides its name space into registries and distributes them among the registration servers. ARPANET Domain name scheme has a tree structured name space. An example is shown in Fig. 2.1.

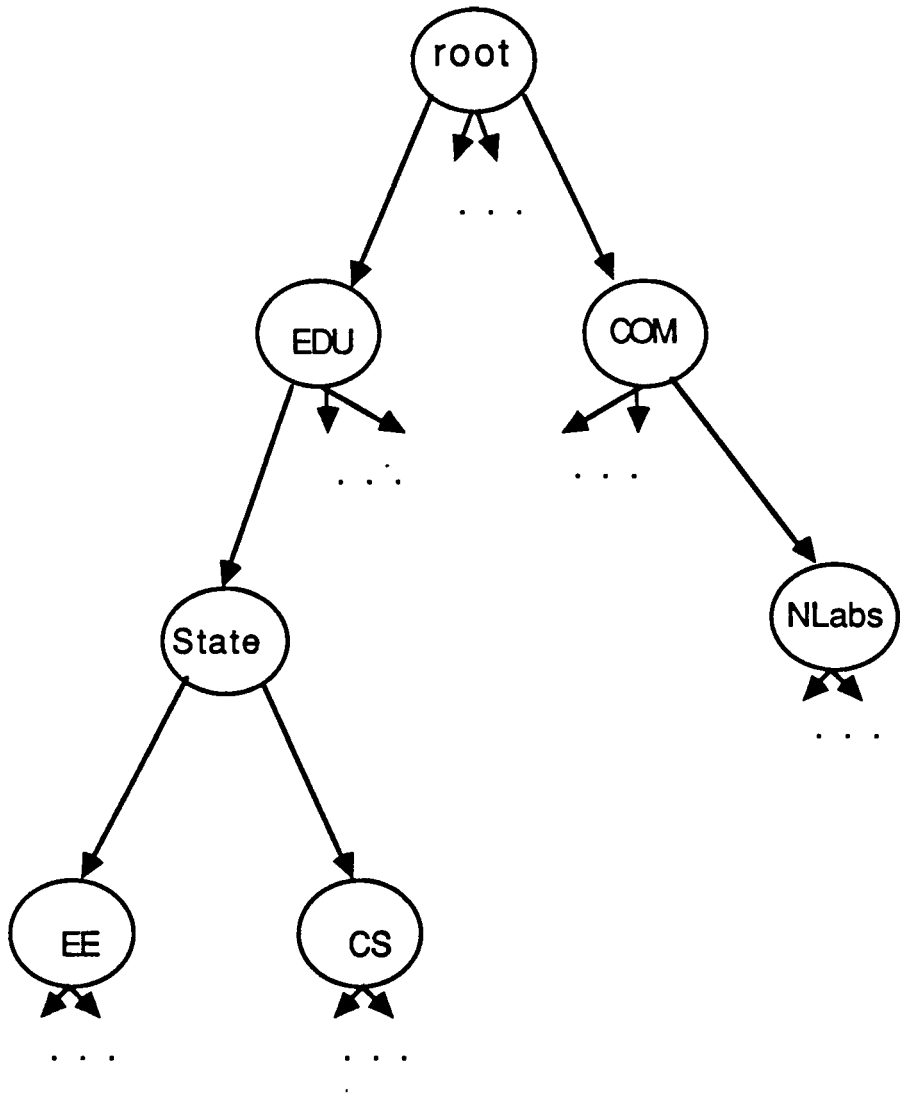

Fig. 2.1 ARPANET tree structured name space.

Terry [TER 85] proposed "Structure-free Name Distribution" which distributes names to authority servers bases on some clustering conditions. This is quite different from the syntax-directed naming discussed in [PET 85], which partitions the name space according to syntactic characteristics of the names.

2.1.8 Name Servers

Name service is a very important component of a distributed system that it enables users to name, locate and access resources and share information about the objects of the system. To manage the name space and to maintain the name service, name servers are needed. A name server is an active process that offers name services to its clients, generally in co-operation with other servers. In computer mail systems, the principal functions of name servers are to facilitate mail services by translating names into addresses at which objects can be located based on the set of bindings between names and addresses, providing directory assistance in locating addresses of mail recipients, and aiding in forwarding mail. The functions are not partitioned but the control and data are decentralized. All name servers play identical roles in the system but each server manage a subset of the name

space. A server usually runs on a single computer. The argument of whether servers should run on dedicated machines will be left to the designers and administrators to consider based on factors such as cost, performance requirement, system environment etc. In this research, the name service is considered as part of mail server functions.

In some systems like ARPANET, CSNET, centralized name services are provided, where a single name server manages the complete service database. In other systems like Grapevine, Clearinghouse, Domain Name System, several name servers collectively manage the name space and support the basic set of operations. Each server does not contain the complete name space database. It only contains part of the name space. All name servers present a common interface and accept requests from any client. If the contacted name server does not contain enough information to process the request locally, it can pass the request to another server(s).

2.1.9 Name Resolution

Name resolution is the process of determining the authority name server for a given object [TER 85]. Given a

name of an object, there must be a mechanism to translate the name into an address in order to access the object. In the next sections we will discuss the name resolution models and mechanisms.

2.1.9.1 Name Resolution Model

Names can be resolved syntactically or algorithmically. Syntactic name resolution depends on the syntax of the names. Basically it is a pattern matching method. Algorithmic name resolution, on the other hand, does not rely on the syntax of names. Some algorithms or functions are used to map the names into addresses.

1. Syntax-directed

   The syntax-directed name resolution model resolves names based on the syntax of the names. In [PET 85], a detailed discussion on the syntax-directed name resolution model is presented. In that model, names are divided into three sets: resolvable, unresolved, and unresolvable. The name resolution process involves the mapping of arbitrary names into resolvable names. The model does not distinguish between "names" and "addresses". They are treated as different forms of names existing in different levels

of the system. Names are viewed as purely syntactic entities, and the name resolution as a syntax-directed operation.

2. Structure-free name management

In most existing distributed name services, the assignment of servers to objects, as well as the mechanisms for name resolution depends heavily on the name structure. A recent research by Terry [TER 85] suggests a structure-free name management model which breaks this dependency between the structure of names and name assignment and resolution.

The differences between Terry's model [TER 85] and other models are that the owner of an object may choose its naming authorities, subject to administrative constraints, and independent of the object's name. This gives flexibility to the assignment and reassignment of authorities. A list of authoritative name servers are maintained for each object. When the authoritative servers for an object need to be changed or reassigned, only this authority server list is changed. Most importantly, the name of the object need not be changed because it is

independent of the assignment of authority. Furthermore, the changes in name resolution procedures also will not require name changes.

The name space is partitioned into contexts using some clustering conditions. A clustering condition is a function that will yield either TRUE or FALSE value when applied to a name. Given a name to be resolved in some context, the particular context is searched for either an authority attribute for that object or a context binding which points to another context where the name might be resolved. The resolution activity will migrate from context to context until the authoritative name servers for the named object are located. This is called the resolution chain. The length of the chain varies for each name.

By separating the authority assignment and name resolution from the structure of the names, the system becomes more flexible. The name service can be easily reconfigured. New servers can be added to the system and assume authority over parts of the existing name space. Object names need not be changed because they do not reflect the underlying configuration.

2.1.9.2 Name Resolution Mechanisms

The name resolution mechanism depends very much on the partition, distribution, and replication of the name space. If the name space is centralized and managed by a single server, the scheme is very simple. All clients will pass the request to a server which can resolve all names. However, this scheme is not very reliable because the server may fail and services become unavailable. Also the response time is relatively long. Another scheme is to replicate the name space information fully in all servers. Then name resolution involves a single database query. It seems to be easy and fast but in very large and diverse environment, the database is undoubtedly too cumbersome to be stored everywhere in its entirety. Also there are problems concerning the storage, updates and consistency of the databases. A more efficient scheme is to partition and distribute the name space among the servers. Distribution will reduce the amount of storage required in each server and the amount of update activity required for adding servers or users to the system. The databases are also replicated to increase their availability and reliability.

2.2 PROCEDURES AND MECHANISMS OF COMPUTER MAIL SYSTEMS

The computer mail system can be functionally divided into the following procedures and mechanisms:

1. Naming and Addressing

   Naming and addressing procedure is a vital part of the system. It specifies how objects in the distributed system are named and addressed. It also defines how name distribution and assignment are done.

2. Message Delivery

   The message delivery procedure defines how a message is transferred from the sender to the recipient. It is further divided into three parts: connection setup, name resolution and forwarding, and delivering. The user must setup a connection with a server in the mail system in order to use the services. After receiving the request from the user, the server is responsible for resolving the name into an address and forwarding the message to the server that is responsible for managing the recipient's mailbox. The server will then deliver the messages to the recipient.

3. Reconfiguration

Reconfiguration mechanism allows the system to make logical and physical changes in order to adjust to the changes in the environment. The basic changes are increases or decreases in users population, and adding, deleting, or moving of hosts and servers. New networks may also be included due to expansion and growth. The reconfiguration mechanism must involve minimum human interference.

4. Flow Control

The traffic pattern in electronic mail system may vary dramatically and must be under appropriate control. Flow control mechanism is a preventive measure that will guard the system from over congestion of messages in the networks and servers. It is an essential part of a reliable mail system. Its function is to monitor the traffic condition of the system and make appropriate adjustment to the system.

5. Clean-up

Clean-up policy is important in preventing overuse of system resources. The main resource we are considering here is the system's message buffers. The buffer management policy defines the allocation of buffers to each users and the maintenance of the buffers. Clean-up procedures also defines how and when old messages should be archived to repository storage.

6. Error and Failure Handling

The ability of the system to handle errors and failures determines the survivability and reliability of the systems. The system should be able to detect errors such as message duplicates, message loses, inconsistencies in the system databases.

7. Migration of Users

Users in the distributed system are not necessarily static. They may move from one location to another. Users should be allowed to move as freely as possible without much inconvenience and overhead. This way the users

can access the mail system from any host in a local region.

8. User Interface

Computer mail is a system service that should not be accessed directly by the users. Users can only access the service through user interface. The user interface provides a large variety of primitives that will assist users in using the mail services. The design of user interface should be friendly, convenient, and powerful.

2.3 EXISTING MAIL SERVICES

Many electronic mail systems already exist. In the following sections, some major identifiable mail systems that have been implemented and documented are examined. We choose four existing systems - Grapevine, ARPANET, CSNET and UUCP, to discuss because they are representatives of the different types of current electronic mail systems.

2.3.1 Grapevine

Grapevine, developed by XEROX, is a distributed and replicated system that provides mail, resources

location, access control and authentication services [BIR 82]. It is a typical single-corporation system that connects the computers and workstations of a company. It is considered one of the most successful electronic mail systems for the office environment. Names in Grapevine are structured as a two-level hierarchy and are of the form "F.R", where "R" is a registry name and "F" is unique within registry "R". Registries correspond to locations, organizations, and applications that exist within the user community. The registration database is distributed and replicated among the many Grapevine computers.

2.3.2 ARPANET

ARPANET, developed by the department of defense, is one of the oldest and largest networks that connects geographically distributed computers and provides full range of services to many institutions. Electronic mail is one of its most popular facilities. Through the years of operations, it experienced rapid growth of user community and host, leading to a slow progression in the name services [TER 85]. At the beginning ARPANET hosts have been identified by flat alphanumeric names. A host table containing the

database of bindings of host names to Internet addresses is maintained by a central authority that oversees changes to the table and distributes copies of the table to every host in the Internet. This centralized host table approach assumes that changes happen infrequently, and that the number of hosts remains relatively small. This flat name space has become impractical when the number of hosts in ARPANET increased [COM 85].

A new scheme called Domain Name System is being developed to decentralize the management of host information [COM 85]. Under the Domain Naming convention, a tree structured name space is used. The original flat name is replaced with a hierarchical one. This allows the single host table to be partitioned and distributed over multiple databases, thus makes the name space more manageable [COM 85].

2.3.3 CSNET

CSNET is a logical network that is build on ARPANET, Telenet, and Phonenet. It is used for communication among computer science researchers from academic and industrial institutions. The CSNET Name

Server, a directory service, is implemented by a central database at University of Wisconsin [SOL 82]. A mail recipient can be unambiguously identified in a location-independent way by supplying a suitable set of keywords, which are mapped by the server to a mailbox address in the form of "user@site". Most mail users use mail facilities directly without consulting the name service.

2.3.4 UUCP

The name "UUCP", for UNIX to UNIX CoPy, originally applied to a transport service used over dial ups between adjacent systems [QUA 86]. It differs from other systems in that it uses a relative naming and addressing scheme. In UUCP, the mail addresses are called source-route addresses because they specify the route through the network from source to destination computer. The addresses, in the form of "host1!host2!site1!user", are only relative because a recipient is identified relative to the sender. Another sender might identify the same recipient differently. The UUCP map and pathalias have made this bearable, but it is still nuisance [QUA 86].

2.4 SUMMARY

This chapter provides the fundamental background and basic concepts of electronic mail systems. Naming and addressing is an important part of the electronic mail systems. Many research have been done to investigate name distribution and management. The syntax-directed naming model provides simple and straight forward solution to the naming problems. However, it has many constraints on the system reconfiguration and user movement. The other approach to separate the name assignment and management from the syntax of the names is more flexible and suitable for very large and diverse computing environments. The discussion of existing mail systems shows that there are still many unresolved problems and limitations in current systems.

CHAPTER 3

DESIGN METHODOLOGIES FOR LARGE ELECTRONIC MAIL SYSTEMS

In this chapter, we develop three design methodologies for electronic mail systems that can be used in large and diverse distributed environments. The three models are mail system with syntax-directed naming, mail system with location-independent access, and attribute-based mail. For the syntax-directed naming model, name assignment, load balancing and message delivery algorithms are developed. For location-independent access model, the main concern is to develop a flexible system which allows users to access the mail service from any location, and a system that can be reconfigured easily without much overhead. The attribute-based mail model is developed to distribute information based on some attributes possessed by the users. An algorithm for efficient broadcasting is explained. For each model, the procedures or mechanisms of

naming and addressing, message delivery, reconfiguration, and user migration are described. Finally the common problems of flow control, storage clean-up and user interface for electronic mail systems are discussed.

3.1 ELECTRONIC MAIL SYSTEM WITH SYNTAX-DIRECTED NAMING

Before messages can be delivered, names, which are used to identify users or mailboxes in a mail system, must be first translated to addresses to indicate the locations of the users or mailboxes. In order to facilitate the mapping between names and addresses, it is natural to incorporate some location attributes into the structure of the names. Syntax-directed naming scheme uses the syntax of a name to identify the location. In this model location dependent hierarchical names are used. This model provides a straight-forward approach to the naming and resolution mechanisms. However, the rigidity of name structure and syntactic pattern matching requirement somehow reduce the system's flexibility and ability to grow.

3.1.1 Naming And Addressing

In large system, the name space is partitioned and distributed in a hierarchical fashion. The number of

hierarchies depends on the environment. The current hierarchical numbering scheme for telephone services is a good example of syntax-directed naming for an environment that covers almost every part of the world. The four level hierarchy of country, state, city, and local switch can be applied to electronic mail system. In this thesis we use a four level hierarchical name to identify users of the computer mail system. The name is in the form of "country.region.host.user". The name components are location dependent. The country and region name is globally unique; the host name is unique within a region; and the user name is locally unique within a host.

The name space consists of all names of the form "country.region.host.user". Each user is assigned an authority server which stores information about him, and assumes responsibility for reliably managing that information. The authority server is responsible for sending and receiving mail on behalf of the user, and is also involved in name resolution, forwarding and delivering of messages. Assigning a single authority server for each user is not reliable enough because if the server goes down, the user will be unable to use the mail services. Therefore, each user is assigned several authority servers, which are ordered in a list such that the first server in

the list is the primary server for the user, and the next is the secondary server, and so on. If one server fails, the user can still access the mail system through other authority servers in the list.

The list of authority servers consists of first the local servers and then some non-local servers in the nearby neighbor regions. The length of the list depends on the probability of server failures and the degree of reliability requirements. Basically, the assignment of server to users is based on some "cost" measure. The "cost" depends on the communication cost between the server and the user, the processing cost and the queuing delays in the server. Next we develop a new algorithm for server assignment. The algorithm has two objectives:

1. To assign a closest server as possible,
2. To balance the load among servers,

so that the total cost (communication + processing + queuing) from each user to all servers in the region is minimized (within a small range). The algorithm first assigns a local server to every host arbitrarily. Then it will move users from a server with higher cost to one with lower cost until the costs are balanced. The algorithm is described as follows:

```
Hi = Host i, 1 <= i <= TotalNumOfHost;
Sj = Server j, 1 <= j <= TotalNumOfServers;
C[i,j] = communication_cost between Hi and Sj;
Pj = processing time of server j;
Lj = current load of server j;
MLj = max. load of server j;
RHOj = the load factor of server j = Lj/MLj;
TC[i,j] = total cost
        = Communication_cost * factor1 +
          (1 + load_factor/(1 - load_factor)) *
          processing_cost * factor2
      = C[i,j] * f1 + (1 + RHOj/(1 - RHOj)) * Pj * f2
Ni = number of users in host i;
A[i,j] = number of users of host i assigned to server j;
factor1, factor2 = some constant factors used to assign
                   different weight to communication cost
                   and the queuing delays
-------------------------------------------------------------
procedure Initialization;
begin
   {** each host is assigned to the nearest server. **}
      for i := 1 to TotalNumOfHosts do
       { A[i,j] := Ni; where (TC[i,j] is min. for all j's)
       }
   {** At this point, some servers may be overloaded. **}
end;
-------------------------------------------------------------
procedure balancing;
begin
 repeat
  change := false;
   for i := 1 to TotalNumOfHosts do
     {Smin := server with min. total cost (TC[i,j]);
      Smax := server with max. total cost (TC[i,j]) among
              the servers with users from this host
              (i.e. A[i,Smax] > 0);
      if (Smin <> Smax) and (TC[i,Smin] < TC[i,Smax]) then
        {move one user of host i from Smax to Smin;
         adjust load of Smin, Smax and the total cost to them;
         if (TC[i,Smin] > TC[i,Smax] ) after adjustment then
           { undo the previous action
             (i.e. move user back from Smin to Smax);
           }
         else
           change := true;
      }
    }
 until no more changes needed;
end;
```

The following is an example to show how the algorithm works. Fig. 3.1 shows the topology and user distribution of our example. The processing powers are the same for all servers. Servers, S1, S2, and S3 are in the same region while S4 belongs to another region. The communication cost is one unit for all local links, and 100 units for inter-region links. After initialization, the server

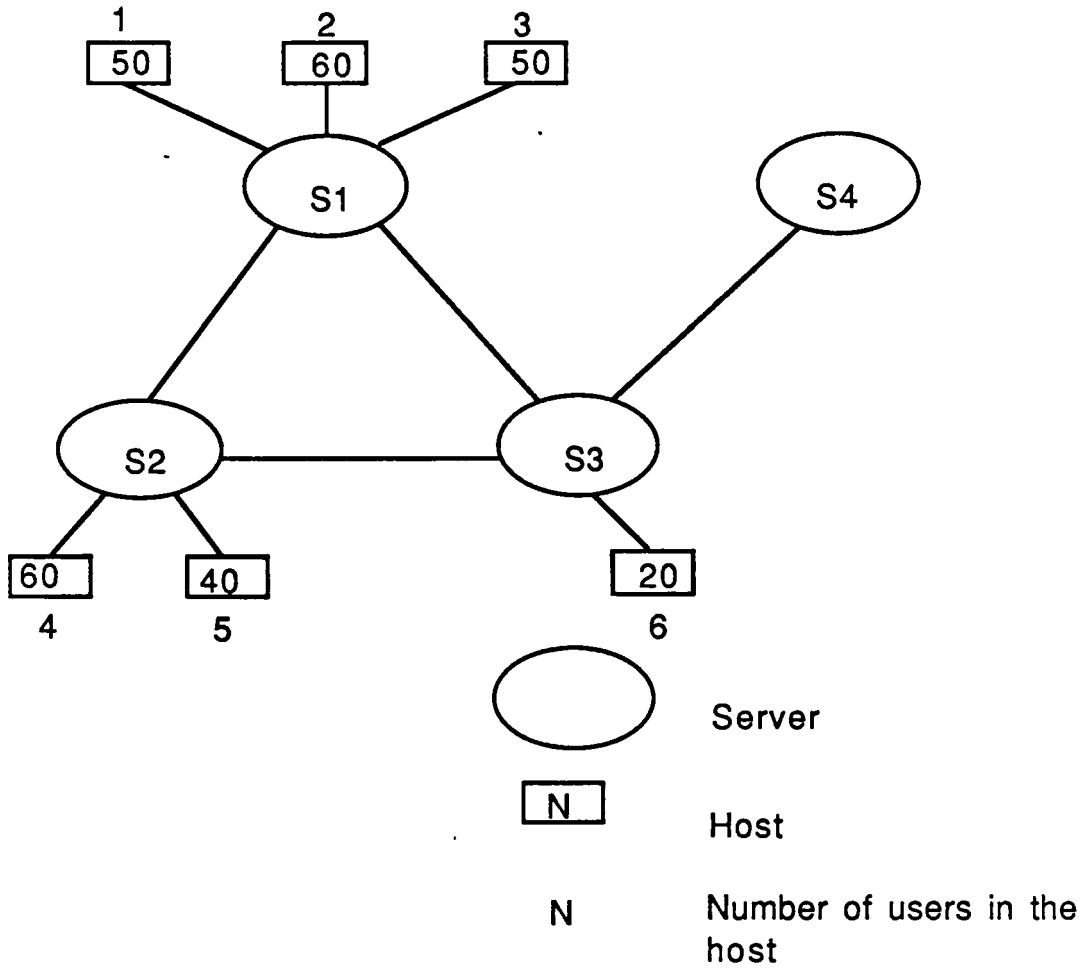

Fig. 3.1 Topology and user distribution.

assignments and load distribution among servers are shown in Fig. 3.2. The final result of balancing the loads is shown in Fig. 3.3. It can be seen that users from some of the hosts are assigned to different servers. Therefore, we must apply some functions to identify the authority server of a particular user, for example using a hashing function.

User Assignment	Servers S1	S2	S3
Host 1	50		
Host 2	60		
Host 3	50		
Host 4		60	
Host 5		40	
Host 6			20
Server load	160	100	20

Fig. 3.2 Initial server assignment and load distribution

User Assignment	Servers S1	S2	S3
Host 1	34	3	13
Host 2	44		16
Host 3	15		35
Host 4		48	12
Host 5		40	
Host 6			20
Server load	93	91	96

Fig. 3.3 Final server assignment.

3.1.2 Message Delivery

The message delivery process starts after the message is presented to the mail server for delivery and ends when the message is delivered to the recipient. The mechanism can be further subdivided into three phases, namely, connection setup, name resolution and forwarding, and delivering.

(a) Connection Setup

The users need to contact a server through the user interface in order to use the mail service. Since each user is assigned a list of authority servers, the user interface will contact the first server from that list, and ask for a mail service. If that server is not available, he will contact the next one and etc.

The problem with this scheme is that it requires large overhead in maintaining the authority server list for each user in the user interface. Some grouping of the users can reduce number of lists, such as keeping a list for each host. However, the lists still need to be updated when there are changes in system configurations, (i.e. adding or deleting a server).

Another way to establish connection between a user and a server is through name servers. A name server maintains complete information about the status of servers in the network and is able to locate an active server for a client. The scheme seems to be simpler than the previous one and offloads the responsibility of locating a mail server to the name server. However, it may introduce another problem - how to contact the name server, which is similar to the problem of locating mail servers. So the problem still exists but in a different level. This scheme is used in Grapevine which uses a primitive and inefficient broadcasting technique to locate name servers.

(b) Name Resolution and forwarding

The name resolution scheme is based on the syntax of names. A name is said to be resolved if an authority server for the name is located. Given a name, the resolution procedure will either return the authority server or a server that may be able to resolve the name. The structure of name resolution tables are shown in Fig. 3.4.

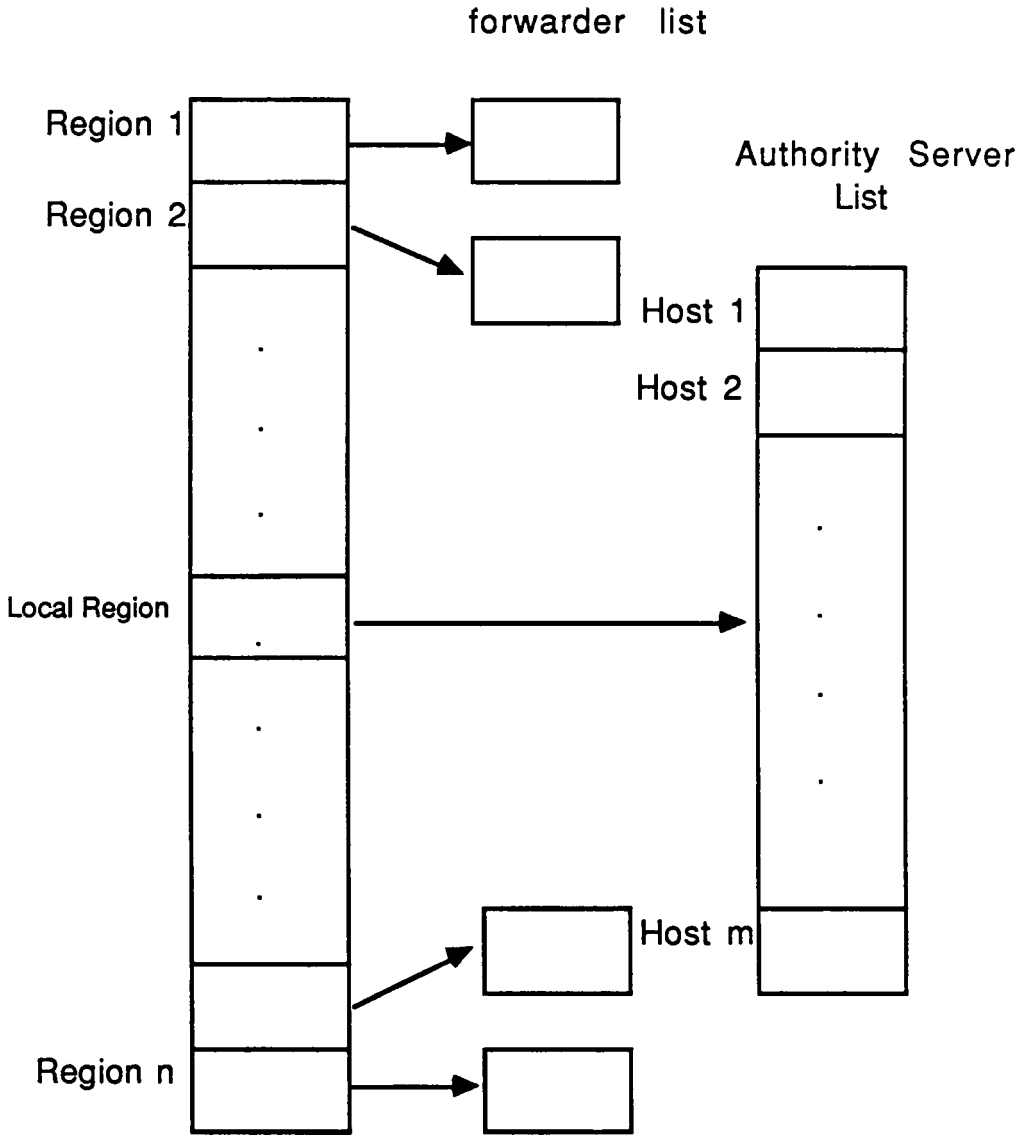

Fig. 3.4 Name resolution table.

A procedure for resolving names can be as follows:
{ In Server }

```
Procedure ResolveName(Name)
   return Server;

BEGIN
   IF RegionName = LOCAL THEN
      BEGIN
         groupId = MappingFunction(HostName);
         Locate a server for that group;
         return the server;
      END
   ELSE
      BEGIN
         Locate a connecting server for the region;
         return the server;
      END
END;
```

(c) Delivering the Message to Recipients

The hosts or computers used by the users are not necessarily large computers. They can be terminals, personal computers, or workstations. The users will not have a lot of storage of their own, and a user's machine will not be turned on all the time (for example a personal computer). Therefore, the received messages are stored in the servers' storage space until the users retrieve them. When a server receives a message on behalf of its recipient, it tries to notify the user immediately by sending an alert signal to him if he is logged on or notify him as soon as he is connected to the system. The user can

choose to save the message in his own storage or delete it after he reads it. Another option can be provided to allow a copy of the message be retained on the server. In that case, some policy of message archiving and clean-up must be implemented to protect the servers' storage from being used up.

Since each user is assigned an ordered list of authority servers {S1,S2,...Sn}, mail will be deposited in the first active server from the list. Servers may become unavailable because of failures or being disconnected from the network. As a result, the user messages may be deposited in more than one server. To retrieve the messages, the most straight-forward method is to poll all the authority servers for that user. However, this is very inefficient and for most times unnecessary. We present an algorithm for retrieving messages that is more efficient than the scheme which polls all servers because it will not check servers when it is sure that they do not store any messages for the user. For each user, the system records the time when the user last checked his mail (LastCheckingTime) and a list of servers (LastUnavailableServers) that were unavailable at that time. Each server records the time (LastStartTime) that it is last recovered from failure or initialized. Whenever a

user wants to check his mail, the interface will check with the first active server in the user's authority server list. If the user's LastCheckingTime is greater than the server's LastStartTime, this means the server has been unavailable for receiving mail for sometime since LastCheckingTime and that some mail might be deposited in other servers. In this case, the user needs to check with other servers in the list. Also the user always checks with active servers who are in the LastUnavailableServers list. The following is the pseudocode for the algorithm.

```
procedure ReceiveMail(user);
   return messages;

BEGIN
 recv_mail_time(user) := current_time;
 while not finished and there are more server to be checked do
  {
   Si := next server in the authority server list;
   if Si is alive then
    {
     Get mail;
     if LastCheckingTime(user) > LastStartTime(server) then
       {
        for all server Sj(j>i) in the LastUnavailableServers do
         {
          if Sj is alive then
           {check_mail;
            remove Sj from the LastUnavailableServer;
           }
         }
        finished := true;
       }
    }
    else
       { if Si is not in the LastUnavailableServers then
           add Si to the LastUnavailableServers list; }
   }
END;
```

3.1.3 Reconfiguration

In large distributed systems, the environment is dynamic and changing. From time to time, users, hosts, and servers are added, deleted, or moved. The system may also expand to include new regions and networks. Reconfiguration is needed to adjust to these changes.

(a) Add/Delete Users

The adding or deleting of users to the mail system is very common operation and usually will not affect the configuration of the system. However, if too many users are added, existing servers will be overloaded. New servers must be added. The load will be redistributed among the servers using the algorithm for server assignment specified before.

(b) Add/Delete hosts

The system can include new hosts. When a new host is added to the system, the new load is distributed among the servers in the region.

On the other hand, if a host is removed, the load balancing state among the servers is upset. The system can reassign the servers among the users by moving some load from servers with high load to servers with low load.

(c) Add/Delete Servers

Adding of new server requires the system to reconfigure. Most changes will be localized to the region where the server is added, although some changes are made to tables in all servers. First, the new server notifies all other servers about its being added and exchanges identification and other information with them. Then the server assignment procedure is performed to redistribute the load so that some users are assigned to the new server. Between the transformation from the old server assignment to the new one, some mechanism of synchronization are needed to co-ordinate the reconfiguration so that old messages are redirected

and new messages are forwarded to the new addresses. Also the databases and tables in the system are updated correctly and consistently.

Deleting a server follows the same procedure as adding a server. The server to be deleted notifies all other servers before it is removed. Those servers then cooperate to share the load of the removed server. Again the server assignment procedure specified before can be used to change the authority servers for the users and to balance the load among the servers.

3.1.4 Migration Of Users

Since the names in this model are location dependent and the resolution mechanism depends on the structure of the names, migrated users have to change their names to indicate their new locations. Also the users are assigned to new servers. Basically the operation involves adding the user to the new location, then deleting the user from the old location. Between the two operations, mail addressed to a migrated user can be redirected to the new user address, and the senders are notified about the name

changes. This is not a very flexible approach.

3.2 LOCATION-INDEPENDENT ACCESS TO MAIL SERVICES

For a system that is changing and growing constantly and dynamically, flexibility in configuring and reconfiguring the system is strongly desired. In syntax-directed naming model, server assignment and name resolution are based on the syntactic characteristics of the names and hence changing server assignment requires changing user names. Furthermore it places some restrictions and constraints on the reconfiguration, growth of the system and migration of the users. On the other hand, complete independence of the name structure will give the system flexibility at the expense of increasing the overhead and response time. Therefore, we must make a compromise between flexibility and cost.

For efficiency purposes, the authority server for a user should be in close vicinity to the user. It is inefficient to allow a user to choose a server that is far away from the user, despite its flexibility. In this model, we divide the name space into regions. Maximum flexibility is allowed within a region - server assignment and name resolution do not depend on the syntax of names;

users can move freely and can send or receive messages from any hosts without having to change names. Also reconfiguration can be localized and has minimum effects on users and on other parts of the system.

We still use the concept of "region" to enhance forwarding of messages so that any reference to a name can be forwarded immediately to the recipient's region and further name resolution and message delivery can be done in the local region.

3.2.1 Naming And Addressing

Although we still use a hierarchical name in the form of "region.host.user", the "host" here indicate the primary location of the user. It does not tell anything about the current location of the user. The difference between this scheme and the previous one is that a user is no longer attached to a fixed host and can access the mail system through any host in the region.

The name space is partitioned into regions. Regions are further divided into small groups of manageable size using some mapping functions. For example, if a region is to be divided into K groups, a function that will map names into an integer in the range of {1..K} can be used. Users

are assigned a list of authority servers using the algorithm described in the previous model. Fig. 3.5 is an example of group mappings.

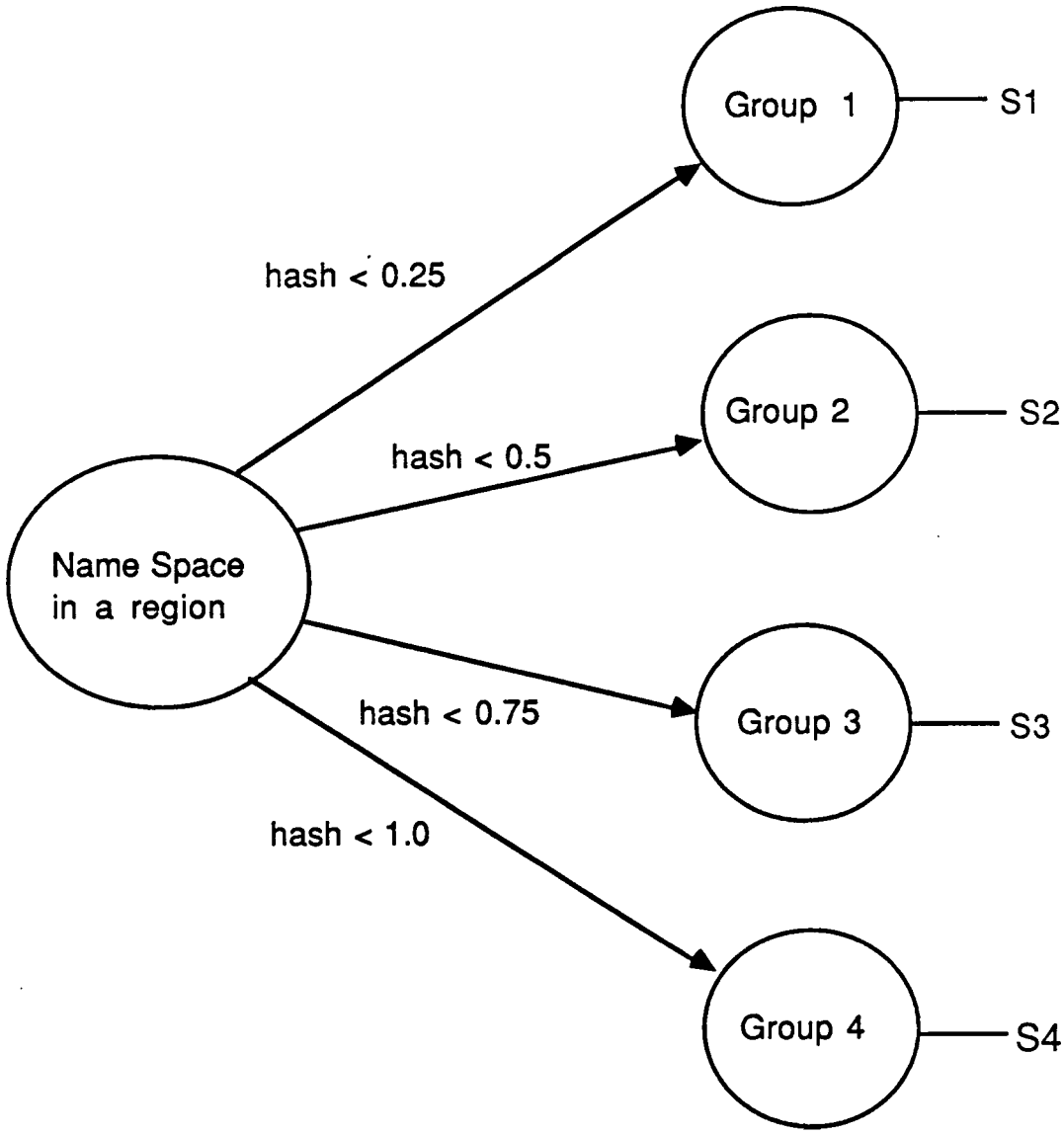

Fig. 3.5 Group mapping using hash function.

3.2.2 Message Delivery

(a) Connection set-up

The users set up a connection to a mail server through the user interface. A user always contacts the nearest active server. All server in a region will co-operate to keep track of the movement of users.

(b) Name resolution and forwarding

Upon receiving a request from the user, the server will try to resolve the name. In this model, all servers can resolve local names within the region. In this case, a hash function is applied to the name to find out in which sub-cluster the name belongs. Then the name can be resolved within the context of that sub-cluster. If the name is not a local name, the server has to contact the corresponding server in the region where the name belongs. The request will be forwarded to that server which will presume the responsibility of resolving the name and delivering the messages.

```
Procedure ResolveName(Name)
   return a server;

BEGIN
   Extract region name;
   IF region = LOCAL THEN
      BEGIN
```

```
            Sub-cluster := Hash(Name);
            Get an active server for that sub-cluster;
            return the server;
         END
      ELSE
         BEGIN
            Locate a connecting server for the region;
            return the server;
         END
   END;
```

(c) Delivering Messages to Recipients

The mechanism for this model is more complicated than the previous scheme because of possible movement of users. Whenever a user logs on to a host, the host will signal the nearest active server to retrieve mail messages for this user. The connecting server keeps the information about the current location of this user. When a server receives a message on behalf of its recipient, it tries to notify the user immediately. From the user name, the primary location of the user can be obtained. The server can send an alert signal to the user if he logs on to his primary location. If the user is not at his primary location, the server has to consult other local servers to find out the current location of the user. This scheme is the same as the previous model if the user does not move. Overhead is only incurred if the users moved to other locations other than his primary location.

3.2.3 Reconfiguration And Expansion

Since reconfiguration mostly involves changes within regions, there is no need to change user names as long as they are inside a region.

Add/Delete Users

Users can be added, deleted or moved within regions without any difficulty. However, if there are too many users in a region that existing servers become overloaded, new servers must be added and reconfiguration is needed.

Add/Delete hosts

In this model host can be added, deleted, or even moved within a region. Since the name assignment and resolution are independent of which host the users belong as in the previous model, adding of a host is treated as addition of a group of users.

Add/Delete Servers

When new servers are added, the system can reconfigure easily because of the independence between server assignment and name structure. Reassignment of servers and rebalancing of load can be done by changing the clustering conditions and the hashing functions. The main advantage of this scheme is that reconfiguration can be done easily without much overhead. Also it does not require changing of names.

3.2.4 Migration Of Users

In this model, since names are not host or location dependent within a region, users can move freely within a region without changing names. The server assignment of the migrated user need not be changed because the communication cost among the servers is very low and does not vary much with the relative location inside the region.

If users moved from one region to another, the overhead of redirecting the mail from old location to new location may be very high. This may also result in long response time for all mail of the migrated users. Therefore, obtaining a new name will in long term be beneficial to the migrated users and place less overhead on

the system, although it might cause temporary inconvenience.

3.3 ATTRIBUTE-BASED MAIL SYSTEM

It has been projected that electronic mail will be a major means of communication for everybody in the future, not only in the office environment, but also at home. It may become as popular as today's telephone services and may replace large part of postal services. It is usually insufficient just to communicate with people you know. Today's business communication requirements go far beyond that. People have to reach out to find potential clients for their markets, services or for information exchange. The attribute-based mail system allows messages to be delivered to recipients who possess certain particular characteristics or attributes even though the senders do not know the complete names of the possible recipients. This model, if properly designed can be a very powerful communication tool for tomorrow's mail systems.

3.3.1 Application Examples

Before we discuss the details of the attribute-based

mail system, we first describe some examples of how the system can be used, so as to give some ideas about the characteristics and possible problems in such a system.

1. Directory Look-up

In our daily communications with other people, we seldom use their full legal name. Instead first names or nicknames are more preferred. In electronic mail system, names are assigned by the system to meet the system naming constraints and to avoid ambiguity. The names usually have very rigid structures. This will impose difficulties on using and remembering those names. People do not always remember the exact spelling of the system names for so many users in such a large and diverse system. Misspelling occurs so often that the system fails to recognize them and services cannot be provided. In attribute-based mail system, the users are allowed to provide aliases, nicknames or some possible misspellings of the names, together with some other information of the intended recipients such as organization and location. The system will try to locate users with the given attributes. There may be more than one user being found possessing same attributes. In that case the user can provide more information to separate them or

resolve them by himself using his intuition, experience or a trial and error method.

2. Information Exchange

A large portion of business communications are between people who do not know each other very well but they have some common interests. They share information to do business. One important task is how to locate people who might be the potential information holders or recipients. The common ways people are using now are checking telephone directories, advertising, recommendations from other people, sending letters, mail lists and etc. These methods are not very convenient and efficient. Using attribute-based mail services, users can easily locate a group of people who share a common set of attributes. For example, a user who wants to collect some information on a special topic can send requests to users who are specialized in the field by using the topic or field name as the attribute for the system to carry out searches.

There will be no doubt that such attribute-based mail services can have many other applications such as job search, marketing, surveying, etc.

3.3.2 Attributes

In this model, a user is identified by a name and a set of attributes. To implement this, there must be a well defined and well designed set of attributes. Attributes can be any characteristics that are associated with a user. The possible types of attributes are too numerous. To name some of them, they can be

- names
- nickname
- alias
- commonly misspelled names
- nationality
- social security number
- job title
- type of job
- organization
- type of organization
- location
- region (city,county,state,country)
- school of graduation
- year of graduation
- college major
- expertise/specialty
- experience
- interests
- hobbies

Each attribute has a type and a value. The "type" indicates the format and the meaning of the value field.

The choice of the attributes must be those in which most mail service users are commonly interested. The values of the attributes should not be ambiguous.

3.3.3 Broadcasting And Searching

In most current mail systems, a single request usually involves one recipient or a group of recipients. The number of recipients and their names and locations are usually known at the time the request is generated. In attribute-based mail systems, however, the names and locations of the recipients may be unknown. The number of the recipients involved can range from zero to all users in the system. If each time, we send messages to all servers in the system to carry out the search, the performance of the system will be very poor. Therefore we need an efficient method for broadcasting and searching for potential recipients.

3.3.3.1 Minimum-Weight Spanning Tree (MST)

One interesting feature of attribute-based mail system is to search a class of customers. We assume that the networks on which the mail system is built, form a

connected undirected graph with computers as nodes and the communication links as the edges. Each edge is assigned a finite weight. Fig. 3.6 is an example of such graph.

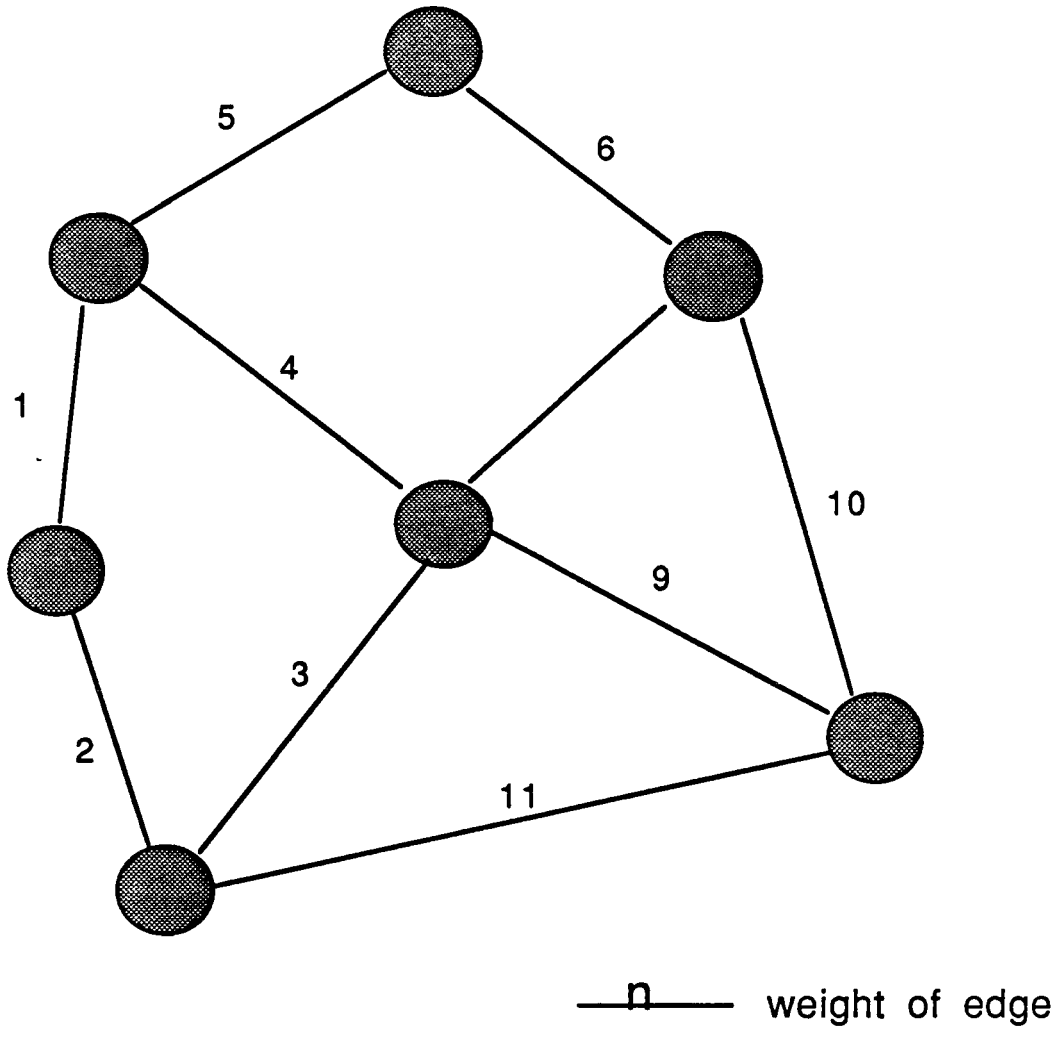

Fig. 3.6 A connected undirected graph

Starting from any node in the graph, we can find a path to go to all other nodes. Such a path is called the Spanning Tree for the graph. The weight of the tree is defined as the total sum of the weights of the edges in the tree [GAL 83]. Minimum-weight Spanning Tree (MST) is the spanning tree of minimum weight among all the possible spanning trees.

(a) A Distributed Algorithm for MST

We first introduce some properties of MST. A fragment of an MST is defined as a subtree of the MST, that is , a connected set of nodes and edges of the MST. The algorithm starts with each individual node as a fragment and ends with the MST as a fragment. An edge is called an outgoing edge of the fragment if one adjacent node is in the fragment and the other is not.

PROPERTY 1. Given a fragment of an MST, let e be a minimum-weight outgoing edge of the fragment. Then joining e and its adjacent nonfragment node to the fragment yields another fragment of an MST.

PROPERTY 2. If all the edges of a connected graph have different weights, then the MST is unique.

The proof of the two properties can be found in [GAL 83]. Using the properties, we can find the MST for a graph with different edge weights. Starting from one or more fragments of single nodes, these fragments can grow in any order based on Property 1. Whenever two fragments have a common node, Property 2 assures that the union of these fragments is also a fragment, allowing fragments to be combined into larger fragments.

In a distributed algorithm for MST [GAL 83], each node performs the same local algorithm, which consists of sending messages over adjoining links, waiting for incoming messages and processing the messages. Messages can be transmitted independently in both directions on an edge and arrive after an unpredictable but finite delay, without error and in sequence. A detailed description of the algorithm and the program codes are presented in [GAL 83].

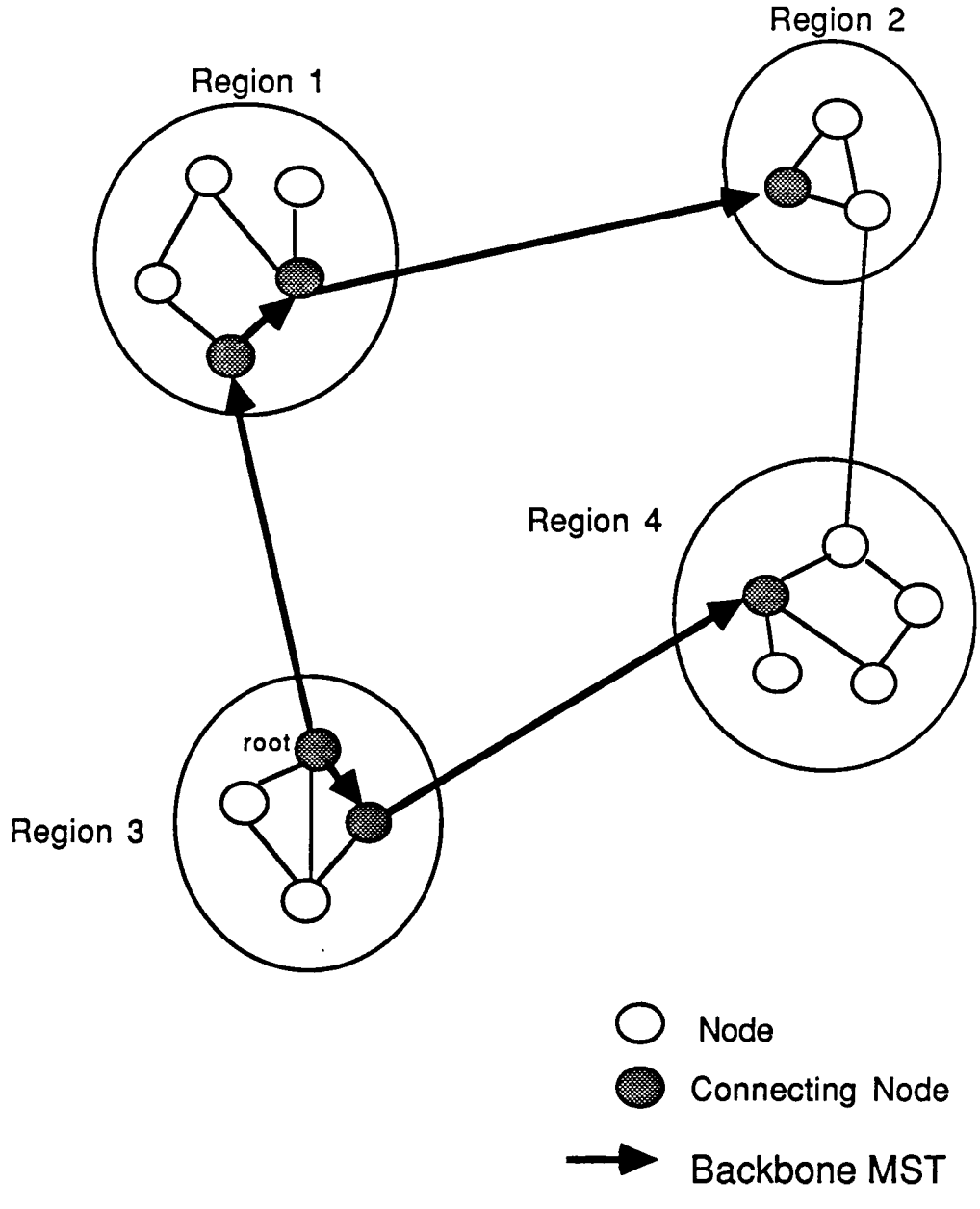

Fig. 3.7 Backbone MST connections of regions.

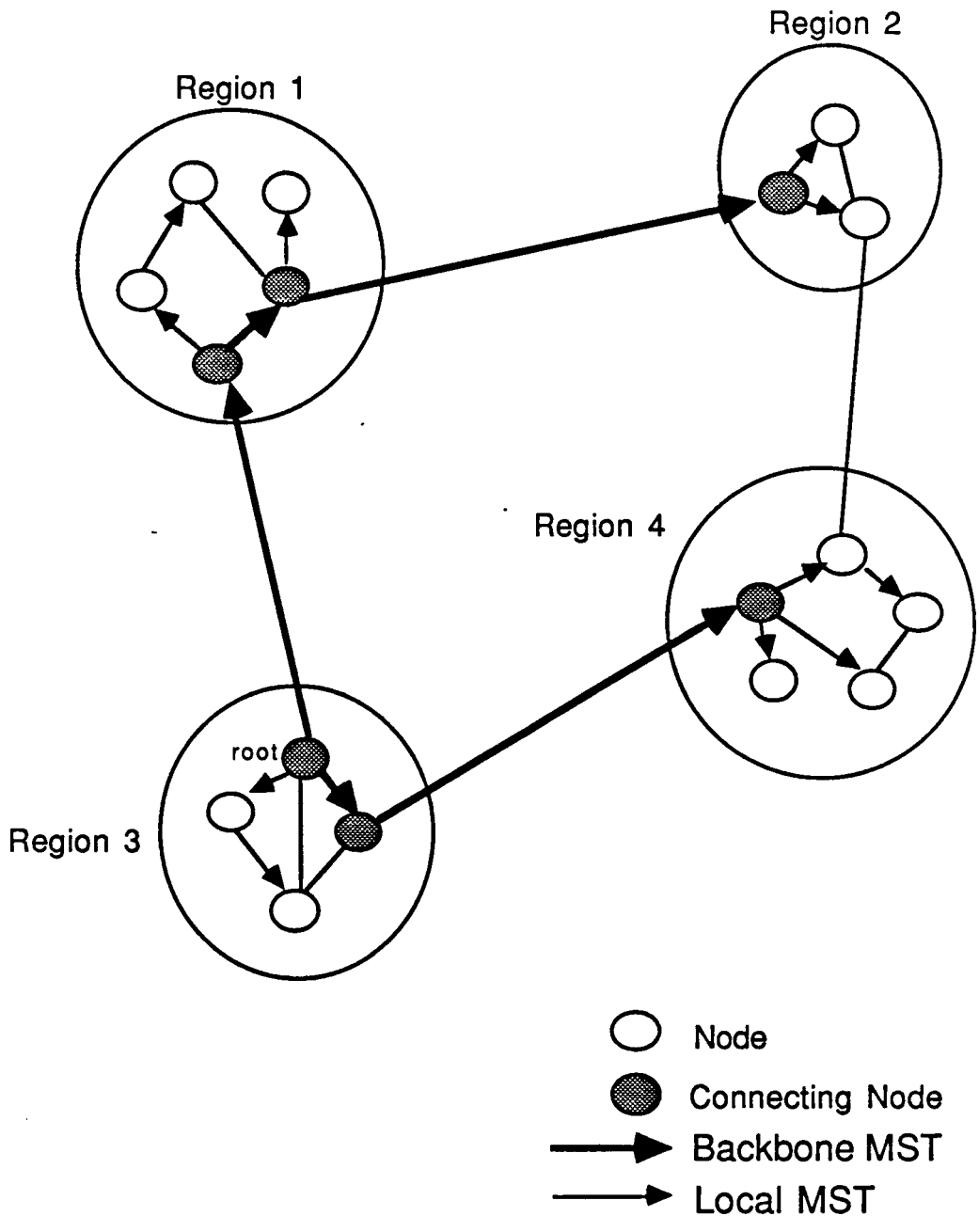

Fig. 3.8 Backbone MST and local MST.

(b) A Modification of the Algorithm

Since our mail system is partitioned into regions, we modify the algorithm to find a back-bone MST to connect all regions. Then the MST algorithm can be performed in each region to span all local nodes. The back-bone MST is formed by nodes which are directly connected to nodes in other regions. Fig. 3.7 shows an example of a back-bone MST. and Fig. 3.8 shows the back-bone MST and the local MSTs.

3.3.3.2 Cost Analysis

The amount of traffic in such a system can be very large as each request may generate many messages. It is very important to estimate and calculate the cost of broadcasting and searching before sending mail to the potential recipients and before getting their responses. A detailed estimate is given to users about the cost for a fee. Based on the detailed estimate of charges and traffic volume, the user can select his recipients and the amount of search he wants to be done. The cost includes the communication cost for broadcasting to the regions to be searched, processing cost for searching the databases in those regions, and delivering messages to users.

Since the weight associated with each edge in the MST is the communication cost between the nodes, the total cost of traversing the MST is the sum of the weights of the MST. When an MST is generated following the previous algorithm, a table listing the costs for delivering to the targeted recipients in each region can be generated. The user who is interested in broadcasting mail then can choose the regions he wants to send his mail to, based on the cost table.

3.3.3.3 Message Control

Another advantage of using the MST for broadcasting is that it can be also used to collect the responses from all other nodes to the source node in a reverse manner. Instead of each node sending a response to the sources, responses can be grouped together to form one summary message as the responses are returning from the leaves of the tree to the root. The scheme is illustrated in Fig. 3.9.

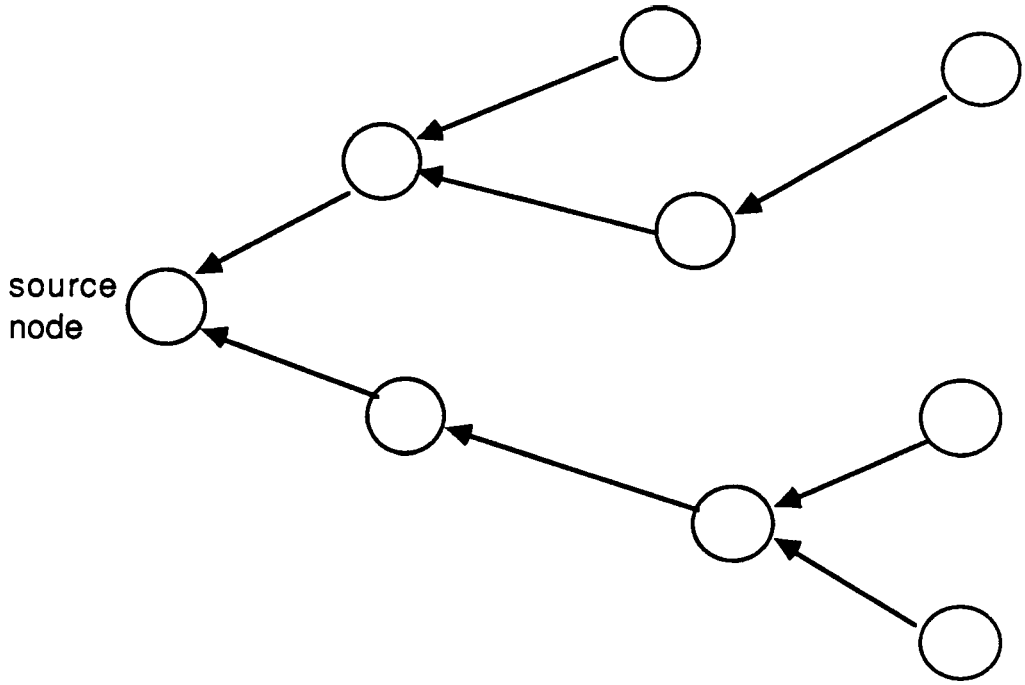

Fig. 3.9 Response collections from other nodes to source nodes.

Upon receiving a request from the father node in the MST, each node sends the message to its son nodes, and wait for the messages to come back from all the son nodes. It then combines them into a single summary message and returns it to its father node. Problem may occur if one of the son nodes goes down while the father node is waiting. Therefore, a father node should time out if it waits for certain period of time. In that case, the estimate for the

failed node can be marked as unavailable.

3.4 PROCEDURES COMMON TO THE THREE MAIL SYSTEMS

The following are the procedures common to the three mail systems.

3.4.1 Flow Control

The function of flow control mechanism is to limit the traffic flow to a rate that can be handled by the receiving servers and the message recipients. We assume that there are already flow control procedures implemented at network level. At the mail service level, flow control involves distribution lists control and buffer management.

1. Distribution List

   When sending a message to a group of users identified by a group name or distribution list, the size of distribution list can be very large such that sending a copy of the message to every member in the list may introduce huge amount of traffic and cause congestion. Therefore it is important to let members of the group share one copy of the message as much as possible. One way

is to expand the distribution list in a hierarchical manner One copy of the message is sent to every primary server of the members of the group. Those members will share the message. Of course, a member can request a copy of the message and store it in his own storage.

2. Buffer Management

a. Buffer allocation

Each user is allocated a specified amount of space for storing the messages. The allocation should be reasonably large that under ordinary situation, it will not be used up. Some buffers are shared among users for storing short system messages such as control or alert messages.

b. Policy of Using Buffers

When a message comes, it is normally stored in the user's allocated buffers. If the user's buffer is full, a short alert message is generated to inform the sender that the receiving user is out of buffers, and messages might not be delivered and should be sent later. An alert

message is also sent to the recipient to inform him that messages have been discarded because buffers are full. The alert message to the recipient should include information about the number of messages discarded, the senders of the messages and the subject of the messages.

The messages can also be categorized according to their importance and characteristics. For example, they can be divided into urgent messages, personal messages, group messages or advertisement, etc. The user can specify the categories in the order of preference. When the buffers are full, less important messages are discarded and important messages are saved.

Buffer clean-up is also important in buffer management and is discussed in the next section.

3.4.2 Message Archiving And Clean-up

Users have responsibilities to clean-up their buffer space. After the user reads his message, he can delete the message, or keep a copy of the message in his own storage. The penalty of not cleaning-up the buffers is that incoming

messages may be discarded when the buffers are full.

The system cannot rely on the self-discipline of users for cleaning-up their buffers. It should have mechanisms to archive the old messages to some repository storage. The archiving of messages should be done periodically, e.g. once a week or two weeks. If resources permit, it is preferable to provide automatic clean-up when the buffers are full.

3.4.3 Error And Failure Handling

The systems we present are very reliable because they use multiple authority servers for each user. When some servers fail, the services can still be provided through other servers. Distribution and replication of data also enhance reliability of the systems that if some server crashes, the data can be recovered from other copies of the data.

To further minimize the chances of loosing messages, priority is assigned to each message. Urgent and important messages can be sent with multiple copies.

In distributed systems, data that are distributed and replicated to improve availability and response time must be kept consistent. Usually updates are propagated by messages. Temporary inconsistency is unavoidable. Synchronization is necessary to co-ordinate the updates and hence to maintain consistent behavior of the system.

For example when a user moves from location A to location B, we cannot delete the user from location A immediately, and then add him to location B because this will result in a time period that the user cannot be accessed at both locations. The user should be added to location B first. All mail for the user addressed to the old address should be redirected to the new location. After a certain period of time, the user can be deleted from location A. This shows that the ordering of events is important.

3.4.4 User Interface

User interface is important from users' point of view because it is the middleman between the users and the mail system. Each user interface may either be implemented as a set of subroutines, that it serves a single client, or can be incorporated in the operating system with system calls

to revoke service operation. In the latter case, A user interface is shared by several clients.

In general, a user interface should provide facilities for preparing, sending, reading, exchanging, removing and organizing messages. Besides the basic functions such as EDIT, SEND, READ, SAVE, and DELETE, the following is a list of useful options. A brief discussion of each service is given.

AUTOREPLY: sends a pre-prepared reply message to all senders in case the user is absent or change his id or location.

APPEND: append file(s) to a message.

MESSAGE SUMMARY: indicates, for each message, whom it was from, when it is mailed, and its subject. The summary can be grouped along sender, time period, or subject.

FOLDER: groups related messages into folders so that they can be reviewed for later references.

REDIRECT: redirects messages automatically to another

recipient specified by the user. This is useful when user moves.

TIMED-DELIVERY: provides the capability of constraining the date and time of message delivery.

PRIORITY-DELIVERY: provides for expedited transfer and delivery of messages.

ADDRESS-LOOKUP: helps to find the mail address of another user by giving information such as name, alias, misspelled name, and location etc.

PERSONAL DIRECTORY: records a list of frequent mail partners. Their addresses are mapped to numbers so that the user need not to remember and type lengthy names and the chances of mis-typing and misspelling are reduced.

3.5 SUMMARY

Syntax-directed naming model provides a straight-forward solution to the distribution and management of name space in very large distributed mail systems. It allows the name space to be partitioned

hierarchically. The name resolution and message delivery mechanisms are based on the syntax of the names. This model is less flexible than the system with location-independent access. In the latter model, names are location-independent within regions. Therefore, names need not be changed when the system reconfigures. Also mail services can be accessed from any host by a user. Attribute-based mail system provides flexibility in addressing recipients by specifying some attributes of the intended recipients, and it can have many applications. For example, most of the mail we receive at home is computer generated letters and advertisements. The attribute-based mail services will be very useful in the future to replace the paper mail because electronic mail will be cheaper, faster and easier to distribute.

CHAPTER 4

COMPARATIVE STUDY OF MAIL SYSTEMS PERFORMANCE

In this chapter, we establish criteria for evaluating the performance of electronic mail systems. The main performance measures are efficiency, reliability, flexibility, and cost. Using these measures and some simulation modeling and analysis, we evaluate and compare the three methodologies we developed in the previous chapter. Using our criteria and the aforementioned performance measures, we evaluate and compare the three methodologies for developing large mail systems.

4.1 EVALUATION CRITERIA

Many computer mail systems already exist. Some of them have been in operation for quite a long time. There are many ways to design a mail system, and each has its

advantages and disadvantages. In order to evaluate them, we establish the following criteria:

4.1.1 Efficiency

Efficiency is always a major concern of any system because it will affect the performance of the system greatly. In electronic mail systems, the efficiency criteria is measured in terms of response time, message delivery, data distribution, and caching capability.

4.1.1.1 Response Time

The response time is the period between the sending of a message and the time the message arrives at its destination. The perceived delay for interactive users should be a few seconds while the actual transport delay should be a few minutes at most. Service connection setup, name resolution, message transportation and delivery are the major components that will affect the response time.

4.1.1.2 Message Delivery

The recipient users should be notified about the arrivals of new messages as soon as possible. When messages arrive, some signals in textual, visual or audio form should be displayed on the recipient's computer. If the recipient does not logon at that time, the signals should be displayed as soon as he logs on the system.

4.1.1.3 Data Distribution

The distribution of data can have a major impact on overall efficiency, in terms of both responsiveness and cost-effective use of the system. It also plays an important role in imposing the storage requirements on the system, and increasing the availability of data.

4.1.1.4 Caching

Caching is another way of enhancing efficiency. Local cache can be maintained to store data that are frequently used, or most recently used so that the information needed can be found in the cache. Care must be taken to maintain a proper balance between level of cache accuracy and the cache hit ratio.

4.1.2 Reliability

The system must be reliable and secure such that users can have confidence in the mail system that their messages, once accepted for delivery, will be made available to the intended recipients or returned with proper error messages. The requirements for reliability is based upon services availability, message flow control, buffer clean-up, and consistency.

4.1.2.1 Service Availability

A major concern is to provide continuous service of the system as a whole in the face of server and network failures. Failures should be localized so that a user can use mail services as long as one of his authority servers is functioning and reachable. The system should also use replication of data to increase reliability and availability.

4.1.2.2 Message Flow Control

As computer mail services become more popular and less expensive, they might become too convenient that people will use them without any reasonable control. In this case, the proper functioning of the system may

be jeopardized. For example, sending lengthy messages might lead to congestion on the communication network, especially during peak traffic period. It might also flood the receivers of the messages.

On the other hand, many messages may be irrelevant or of no interests to the recipients of the messages. A user usually must delete a lot of "junk mail." If not controlled properly, those "electronic junk mail" can be more wasteful than the traditional "paper junk mail", because it is so easy to send or broadcast electronic messages.

4.1.2.3 Buffer Clean-up

Buffer clean-up is very important because mail services cannot continue if the buffers are full. Therefore it is necessary for the system to clean-up buffers periodically. The system should setup rules or policy regarding the priority of buffer allocation when buffers are full.

4.1.2.4 Consistency

In distributed systems, data is distributed and replicated to improve availability and response time. There are consistency constraints that must be maintained. Such constraints apply not only to individual pieces of data, but to distributed sets of data as well. Usually updates of user information and system status are propagated by messages. Temporary inconsistency is unavoidable. Synchronization is necessary to co-ordinate the updates and hence to maintain consistent behavior of the system.

Therefore, during the use of mail services, the user should be informed of the state of availability, accessibility and changes of the system if such actions are needed to justify the unexpected or inconsistent behaviors such as duplication of messages, failure to receive mail or failure to access a user who has just been added to the system, etc. By trying to hide everything from the users, they might not understand the behavior of the system. This problem might raise doubt about the integrity and reliability of the system.

In summary, reliability should not be achieved only by replication of messages and services, but also by reliable software, hardware and communication, appropriate management of system resources.

4.1.3 Flexibility

Flexibility deals with the system's ability to provide wide range of functions, to minimize restrictions and constraints on users, and to adjust to changes in the system. The most important issues concerning flexibility are related to user migration, group naming, system reconfiguration and user interface design.

4.1.3.1 User Migration

It is conceivable that users may move from one organization or area to another, especially in large corporations which tend to restructure their organization charts and move people from one group to another reasonably frequently. If a user who moves is forced to change names, this is considered to be unnecessary impairment of naming freedom. The system should allow a certain degree of user migration without requiring the users to change names. For

temporary migration such as business trips or temporary relocation, mail should be redirected to the temporary location. This way a user can access his mail from any part of the system.

4.1.3.2 Group Naming And Attribute-based Addressing

One objective of electronic mail system is to share information among the users. Therefore, recipients of mail messages should not be restricted to individuals only. They can be groups or distribution lists. A group of users can be identified either by a group name or by a common set of attributes. One copy of the message can be shared among members of the group whenever possible.

4.1.3.3 Reconfiguration

An important feature that a distributed system must have is the ability to expand and reconfigure itself in order to increase processing power, decrease response time, increase availability of data or adjust to changes in the system. To maintain continuous services, reconfiguration must be done dynamically or with little intervention by humans and without

interrupting system operations.

The problems of reconfiguration deal with when to reconfigure, and how to reconfigure. Usually the system needs reconfiguration when a server or host is added to or deleted from the system, or when new networks are included in the system. Reconfiguration procedures include updating entries in the databases and reassigning servers. If the original configuration is not well designed, reconfiguration can lead to large system overhead. For example if each server contains a copy of database for all users, a slight change in a single record will require changes be made in all servers. However if the database is distributed and partially replicated among the servers, only the servers involved need to make updates. System reconfiguration should be transparent to users.

Internet computing environment are continuously evolving and expanding in size, either by the participating organization acquiring new computing equipments or by their interconnecting to other computing environments. The initial design of the system must consider the potential for the system to

grow and expand. Basically, the system has to deal with the adding of new users, hosts, servers, networks and regions. In any cases, the size of the components of the mail service at individual sites and the number of interactions between components should not be directly proportional to the size of the environment. Also the size of the databases should not be directly proportional to the size of the user community.

4.1.3.4 User Interface

The objective of electronic mail is to increase people's efficiency in communications. Beside providing basic facilities for interpersonal communications, electronic mail system must be convenient, easy and friendly to use in order to attract people away from traditional ways of communications. The user interface plays an important role. A variety of facilities and functions should be provided to aid the users in dealing with the messages. The requirements for a user interface are that it should be friendly, easy to use, and intelligent.

4.1.4 Cost

The system must be cost-effective. Cost can be measured in terms of response time, storage space used, implementation overhead, and many other factors.

These are the criteria we use to evaluate electronic mail systems. Actually some of these criteria are related to each other. The effectiveness of flow control and clean-up will affect both the efficiency and reliability of the system. On the other hand, some of them may have conflicting requirements. For example it is very difficult to have a very efficient system and a very flexible and reliable system at the same time. Shorter response time may require more storage for additional data in the databases. Therefore, it is necessary for designers and administrators to weight different alternatives and make a balance between the benefit and cost.

4.2 THE SIMULATION MODEL

Simulation programs have been developed to simulate the procedures and algorithm described in previous chapter. Simulation means driving a model of a system with suitable

inputs and observing the corresponding outputs. It is widely used to study systems that have not been implemented or cannot be accessed directly. We use simulation experiments to test our procedures and algorithms, and to compare the performance of the three methodologies. Details that are not relevant to the operation and traffic flow of the system, such as user interface, users, and host computers, are not represented. Only the relevant parts of the model such as messages, servers, the links are incorporated in the simulation experiments. Therefore it is reasonable to use simulation instead of implementation of the whole model.

In our simulation model, we use the topology of Grapevine system as our backbone structure because Grapevine is a real distributed system that has been in operation for some time. Fig. 4.1 shows the topology we used. The size of the network under study represents a compromise between the large computer time needed for simulation experiments and a fair representation of a distributed network topology.

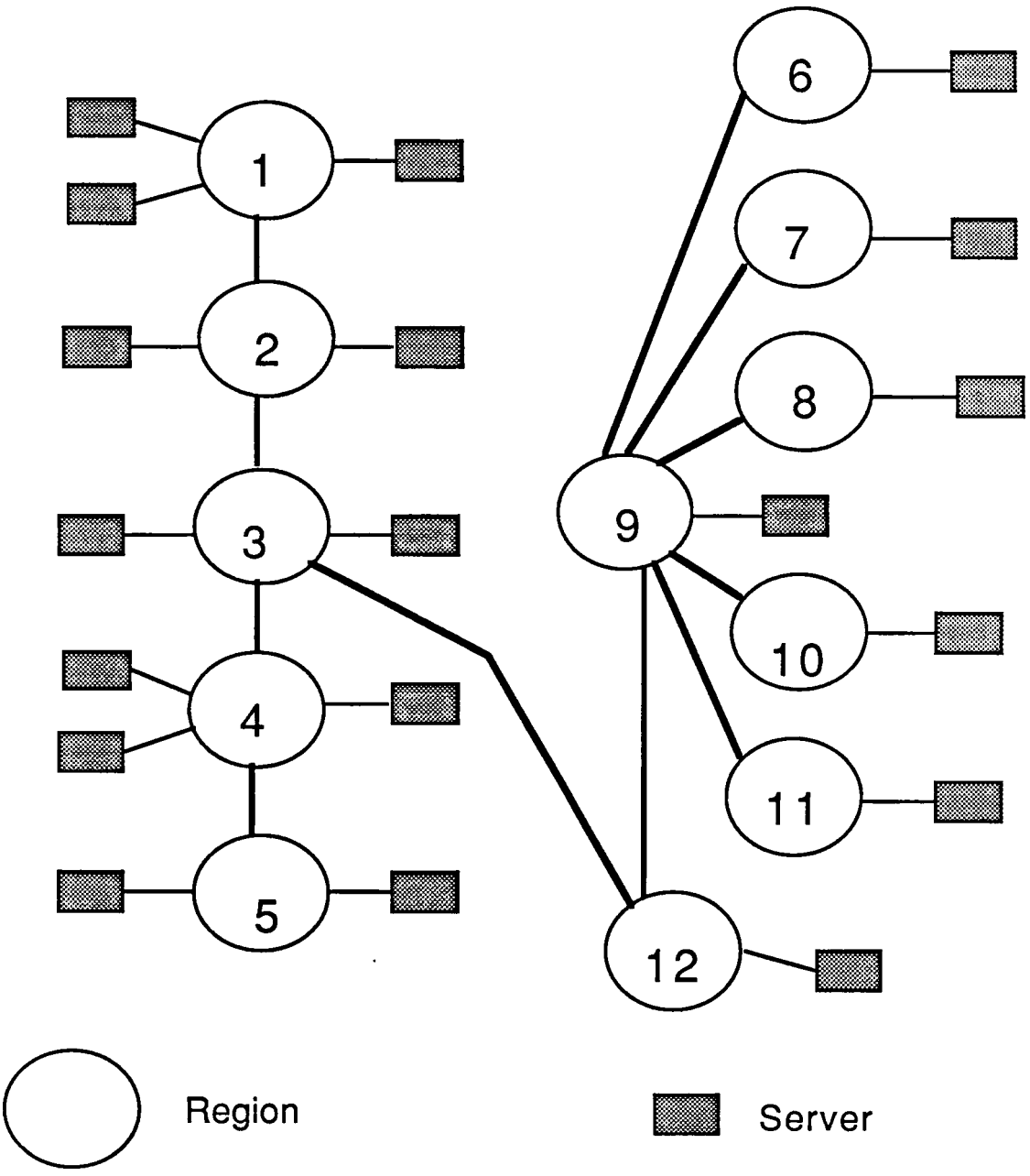

Fig. 4.1 Topology of the simulation model.

The simulation program is written in SIMULA, a language specially designed for simulation experiment purpose. The SIMULA provides queue manipulation, process scheduling and a set of random number generators to ease the task of writing simulation programs. The simulation uses an event driven approach.

4.3 ASSUMPTIONS

The following are the assumptions about regions, servers, users, messages and traffic rate of our simulation models.

1. The system consists of twelve regions connected by long distant links. Each region is treated as a collective unit of local networks. The region consists of hosts and servers. Fixed routing is used to route messages. Traffic flow control is part of the network communication services but is not simulated in our experiment because we assume the traffic load is not heavy and congestion will seldom happen.

2. There are at least two servers in each region for reliability reasons because we want to provide continuous services to users even when some servers are down. Since the chance of one server going down is very small, the chance of two servers in the same region going down is negligible. Even if the later case does happen, the services can still be available using servers of nearby regions.

3. There are two systems. In one system, users are attached to hosts and in the other one, each user has a primary host but can move to other hosts. The selection of sender is random. Once the sender is selected, the recipient is selected either locally or externally. We assume that 80% of the messages are addressed to a recipient within the same region while 20% is destined to users in remote region. We further assume that all remote users have equal chance of being the recipient of the message.

4. All messages generated are single-packet messages. The packet length is assumed to have a truncated exponential distribution with mean of 6500 bits.

5. The number of messages generated by each host is proportional to the number of users in that host. Messages are generated according to a Poisson distribution. The generated traffic is found to have bursty nature of real systems.

6. Messages are removed from the system when they arrive at the buffer areas where they can be accessed.

7. A server is said to be overloaded when the number of users under its management exceeds a certain limit. In this case a new server is added to the region of the overloaded system and reconfiguration is performed.

4.4 COMPARATIVE EVALUATION

The following is the comparative evaluation of the three methodologies using our criteria. Since the performance measures depend on the implementation of the mail system, our comparisons are based on the design issues rather than particular implementations. The importance of the criteria is discussed in previous sections. In the following section, we present some results from our

simulation experiments.

(a) Response Time

The response time of a message is defined to be the elapsed time since the message was generated until the message is stored in one of the recipient's authority servers. The main components of response time are shown below.

1. Connection Time (CT) - the time used to locate an active server and setup the connection for service. It is measured as the elapsed time since the message is generated until it reaches one of the servers from the sender's computer.

2. Name Resolution Time (RT) - the time used to resolve a name to a destination address.

3. Processing Time (PT) - the time used to process each message by the server or host.

4. Transmission Delay (TD) - the time used to transmit a message from one node to another.

The evaluation of response time hinges on the analysis of the individual components. However, sometimes the boundaries between the different times are not clear cut. There is some overlapping. This makes it difficult to measure the different kind of delays. For example, the name resolution procedure may involve more than a single step. When a server does not have enough information to resolve a name, it will find a more knowledgeable server and forward the request to it. That server will repeat the name resolution procedure until the name is resolved in some server.

	Mean Response Time (in seconds)		
Model	Local traffic	Outbound Traffic	Total
I	0.2	1.1	0.4
II	0.3	1.3	0.5
III	0.4	1.5	0.6

I: Syntax-directed naming model
II: Location-independent access model (no user migration)
III: Location-independent access model (10% user migration)

* Assume the local to outbound traffic ratio is 80:20

Table 1. The mean response time for model I and II.

Table 1 shows the mean response time for syntax-directed naming model and location independent access model. The response time is measured separately for local traffic and outbound traffic. The total mean depends

on the ratio of local and outbound traffic. It can be seen that the response time for the second model is very close to that for the first model when there is no user migration. However, when users are allowed to move, the response time increases moderately due to the increase in connect time as users move farther away from their primary server. This shows that flexibility of the system to handle user migration may affect the efficiency of the system.

The attribute-based mail model has a different approach. The response time for broadcasting includes the time to traverse the Minimum Weight Spanning Tree and the time to search the database at each node.

(b) Message Retrieval

Since the messages received for a recipient may be stored in more than one server, there must be a way to retrieve them. One way of retrieving the messages is to poll all servers who may store some messages for a user. We use a message retrieval algorithm which does not need to poll all servers. The algorithm is described in chapter 3, and is used for both model I and II. We assume that each user is assigned a list of 5 authority servers.

Using polling policy, each time the user poll all his authority servers. This is unnecessary and wastes a lot of time. Using our algorithm, the user only needs to check with the primary server if the probability of server failure is small. As the probability of server failure increases, the mean number of polls also increases. As technology advances, the server failure probability in mail system is getting smaller. A user needs to poll two servers at most under ordinary situation.

Another factor that affects the mean number of polls is migration of users. When a user moves away from his primary host, he has to contact a nearby server which will in turn contact the primary server for user so the number of polls increases when the user migrates

The polling method is simple and easy to implement. Our algorithm has larger overhead because it needs to record some information about the time a user last retrieves his mail, the servers that are unavailable at that time and the restarting time (after failures) for each server.

(c) Cost

The syntax-directed naming model is straight-forward and easy to implement. It also saves a lot of space because location information is embedded in the syntax of names. The location-independent access model is more flexible but has larger overhead because it needs to keep track of the current location and movement of users. Also there are some delays due to longer connect time when users move. The attribute-based mail system needs large databases to store the attributes for users.

4.5 SUMMARY

We use efficiency, reliability, and flexibility as our criteria for comparing and evaluating electronic mail systems. Simulation models and experiments are developed to gain some quantitative measures about the performance of the systems in term of response time, efficiency of message retrieval, and cost of implementation.

CHAPTER 5

CONCLUSION

5.1 SUMMARY

Electronic mail will become one of the major means of communication and information exchange in the future. In this thesis, we develop three methodologies for designing electronic mail system in large and distributed environments.

In syntax-direct naming model, names of users are hierarchical and location-dependent. This model can serve as a framework of large systems because it fits the current geographical division of communities. Hierarchical name in the form of "country.region.host.user" can be used. We develop a load balancing algorithm to distribute loads among servers systematically using some cost measure. This algorithm can be applied to the initial server assignments

and server reassignments when adding or deleting hosts and servers.

Using the first model as the backbone structure, we develop location-independent access model to improve the flexibility of the system by reducing the dependence between the syntax of names and the locations of users. The system includes an algorithm for keeping track of user movements so that users can move and access the system from different hosts in a region without the need to reassign names and servers. Another advantage of this model is that the reconfiguration is simpler and with less overhead.

We further enhance the mail system by using attribute-based naming scheme. Mail recipients can be identified using attributes which are usually more meaningful than just the simple user names. It can be used for mass distribution of mail and information exchange. Its usage in address lookup, information exchange, marketing and advertising can be very powerful.

The requirements of large distributed electronic mail systems are efficiency, reliability, flexibility and cost. These are the criteria we use for comparing and evaluating electronic mail system design. The response times for the three mail systems are very reasonable. There might be

longer delays due to broadcasting and searching when using the attribute-based naming but we use the Minimum-weight Spanning Tree algorithm to minimize the delay. The reliability of the systems is enhanced through the use of multiple authority servers and distribution and replication of data. Also our algorithm for retrieving messages guarantees that no messages will be lost even when some servers fail. Moreover, the algorithm is more efficient than a simple polling scheme and the number of polls per retrieval request is at most two under most conditions. Also user migration, group naming, and flexible reconfiguration make the systems very flexible.

5.2 AREAS FOR FUTURE RESEARCH

Several interesting areas for future work related to electronic mail systems are discussed herein.

Integration of voice, video and text data: Electronic mail systems should be able to transfer messages that consist of different forms of data, such as voice, video, graphs, and facsimile. The systems should be able to convert the information into some standard formats and transmit it through the networks. Before presenting the messages to the users, data should be

transferred back to its original form.

Message Filtering: The volume of electronic mail will be very huge as it become easier and less expensive to send electronic mail. Some process with artificial intelligence should be developed to categorize and distinguish the importance and relevancy and priority of messages so that unwanted mail can be filtered out.

REFERENCES

[BIR 82] A. Birrell, R. Levin, R. M. Needham, and M. D. Schroeder. "Grapevine: An Exercise in Distributed Computing," Communications of ACM, Vol. 24, No. 4, April 1982, pp. 260-274.

[CHE 84] R. F. Cheng. "Naming and Addressing in Interconnected Computer Networks," Ph. D. Thesis, University of Illinois at Urbana-Champaign, May 1984.

[CHO 83] W. Chou, A. A. Nilsson and S. C. Chang. "Distributed Directories in Internetworking Environment: Strategy and Performance," Proceedings IEEE INFOCOM 83, April 1983, pp. 563-571.

[COM 83] D. Comer. "The Computer Science Research Network CSNET: A History and Status Report," Communications of ACM, Vol. 26, No. 10, October 1983, pp. 747-753.

[COM 85] D. Comer and L. Peterson. "Issues in Using DARPA Domain Names for Computer Mail," Proceedings 9th Data Communication Symposium, September 1985, pp. 158-164.

[CRO 79] David H. Crocker, Edward S. Szurkowski, and David J. Farber. "An Internetwork Memo Distribution Capability - MMDF," Proceedings of the Sixth Data Communication Symposium, November 1979, pp. 18-25.

[GAL 83] R. G. Gallager, P. A. Humblet, and P. M. Spira. "A Distributed Algorithm for Minimum-Weight Spanning Trees," ACM Transactions on Programming Languages and Systems, Vol. 5, No. 1, January 1983, pp. 66-77.

[HUM 83] Pierre A. Humblet. "A Distributed Algorithm for Minimum Weight Directed Spanning Trees," IEEE Transactions on Communications, Vol. 31, No. 6, June 1983, pp. 756-762.

[LAN 85] K. Lantz, J. Edighoffer, and B. Hitson. "Towards a Universal Directory Service," Proceedings Fourth Symposium on the Principals of Distributed Computing, Minaki, Canada, August 1985, pp. 250-260.

[PET 85] Larry L. Peterson. "Defining and Naming the Fundamental Objects in a Distributed Message System," Ph. D. Thesis, Purdue University, May 1985.

[QUA 86] John S. Quarterman and Josiah C. Hoskins. "Notable Computer Networks," Communications of ACM, Vol. 29, No. 10, October 1986, pp. 932-971.

[SCH 84] M. D. Schroeder, A. D. Birrell, and R. M. Needham. "Experience with Grapevine: The Growth of a distributed System," ACM Transactions on Computer Systems, Vol. 2, No. 1, February 1984, pp. 3-23.

[SHO 78] J. F. Shoch. "Internetwork Naming, Addressing, and Routing," Proceedings 17th IEEE COMPCON, September 1978, pp. 72-79.

[SOL 82] M. Solomon, L. H. Landweber, and D. Neuhengen. "The CSNET Name Server," Computer Networks, Vol. 6, No. 3, July 1982, pp. 161-172.

[SUN 82] C. A. Sunshine. "Addressing Problems in Multi-network Systems," Proceedings INFOCOM 82, March 1982, pp. 12-18.

[TER 85] D. B. Terry. "Distributed Name Servers: Naming and Caching in Large Distributed Computing Environments," Ph. D. Thesis, University of California at Berkeley, February 1985.