Designing Large Electronic Mail Systems

By

Wael Bahaa-El-Din Hidal
Digital Equipment Corporation

Hsi-Tung Yuen
AT&T Information Systems

Abstract

Electronic mail is one of the most important means for communication and information exchange in internetworking environments. In this paper, three methods for designing large mail systems are investigated with varying degrees of flexibilities, namely, mail systems with syntax-directed naming, mail systems with limited 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 tested using simulation. Mail systems with limited location-independent access allow users to access them from one primary location and a number of secondary locations. Procedures that keep track of users who migrate from their primary location and redirect 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. Finally, criteria for evaluating electronic mail systems are presented.

1 Introduction

Electronic mail systems provide a convenient, efficient, economical, and reliable way of communication and information exchange and sharing. It offers the advantages of speedy delivery, low cost, and rapid reply. Moreover, the cost of electronic mail became less expensive than telephone communication in recent years. Aided by the decreasing cost of computers and powered by new technologies, future electronic mail systems are expected to include thousands of users just like today's telephone system.

An electronic mail system is composed of users, user interfaces, hosts, mail servers, and networks. A user or client is identified by a name which is used for the purpose of reading mail, accessing the mail services, and addressing 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, receiving, reading, and deleting mail and doing other mail-related functions. A mail server is a process responsible for obtaining addresses of recipients, sending, buffering, relaying and delivering messages to the mail recipients. It contains information about users, hosts, networks, and other mail servers. The electronic mail system is usually built on top of networks that 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 a name resolution scheme, and sends the message to the recipient's mail server through other hosts and servers using the communication service. Upon receiving a message, the server notifies the recipient (if possible). The user can retrieve his messages from the server through the user interface, and can read and buffer these messages.

In this research, we are concerned with some of the problems in designing large electronic mail systems, namely, naming and addressing of users and mailboxes, flexibility in reconfiguration, user migration, and mass distribution of mail using attributes. We present three different naming and addressing approaches that can be used for electronic mail systems in large and diverse distributed environments. The first approach is a mail system with syntax-directed naming scheme. It emphasizes the hierarchical partitioning and distribution of mail services using syntax-directed names. New algorithm for balancing loads among servers, and a new procedure for efficient message delivery are presented. Next, we consider a more flexible approach for naming and addressing which permit location-independent access to mail systems. It allows users to access the mail services through any host in their local geographical area. A mechanism to keep track of users who move from their primary access location and redirect their mail is presented. To enhance the flexibility of a mail system even further, we consider an attribute-based mail system, which allows selective search of recipients and mass distribution of mail using some defined attributes about users.

A procedure that uses efficient broadcasting is discussed to perform the two previously mentioned functions.

In this paper, we discuss three approaches to design large electronic mail systems and present new procedures and algorithms to perform some of the important mail system functions. Section 2 presents the basic operation of large electronic mail systems and definitions that are needed to understand the rest of this paper. In Section 3, some interesting procedures and mechanisms to perform functions such as naming and addressing, message delivery, reconfiguration, and migration of users are presented for the three mail systems with different degrees of flexibilities. In Section 4, criteria for evaluating mail systems are presented. Finally, conclusions are given in Section 5.

2 The Operation Of Electronic Mail Systems

A distributed electronic mail system is conceptually viewed as a collection of passive and active objects such as users, mailboxes and servers [PET85]. Each object is associated with a name for the purpose of identifying, locating, and accessing it. Names are usually structured as a set of alphanumeric strings chosen from a finite alphabet [PET85] and separated by delimiters. A general form of a 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. In order to have a uniform naming scheme, each object has a naming convention which is a set of rules that governs 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. In a single computer system or in a single network environment, where the number of objects is reasonable, the names can be stored in a single centralized database. However, in a large distributed sys-
ter, a single centralized database is too inefficient to use and manage. Therefore, the name space is partitioned into some easily manageable subspaces referred to as constants and distributed among servers so that no server needs the complete knowledge of all names.

Name service is a very important component of a distributed system in that it enables users to name, locate and access resources and share information about the objects of the system. In computer mail systems, the principal functions of name servers are to facilitate mail services by translating names into addresses and providing directory assistance in locating addresses of mail recipients. In this research, the name service concerning mail is considered as part of the mail server functions. This part of the mail server will be referred to here as the name server.

In some systems like CSNET [ROL82], centralized name services are provided, where a single name server provides several directory services. In other systems like Grapevine [BIR82, SCH82], Clearinghouse, and ARPANET Domain Name System [COM85], several name servers collectively manage the name space and support the basic set of name-resolution operations. In the systems under study, we assume that each server does not contain the complete name database. It only contains a subset of the user names. All name servers present a common interface and accept requests from all users. If the contacted name server does not contain enough information to process the request locally, it can pass the request to another server; until a server is found that has complete information about the user and has a mailbox for him. This server is said to be the authority server [TER85] for the user.

Name resolution is the mechanism to translate a name into an address in order to access a user mailbox. 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, then the scheme is very simple. All user interfaces will pass in the requests to a server which can resolve all names. However, this scheme is not very reliable because the server may fail and services become unavailable. Another scheme is to replicate the name space information fully in all servers. Then name resolution involves a single database query. It might appear as an attractive method but in very large and diverse environments, 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 or removing users to the system. Also, the databases are partially replicated to increase the availability and the reliability of the system.

Names can be resolved syntactically (i.e., syntax-directed naming resolution) [PET85]. Syntax-directed naming resolution depends on the syntax of names. Basically it is a pattern matching method. Algorithmic name resolution, on the other hand, does not rely on the syntax of names but rather on some algorithms or functions that are used to map the names into addresses. Structure-free name resolution [TER85] breaks the dependency between the structure of names and name assignment and resolution. The differences between the model in [TER85] and other models are that the owner of an object may choose its naming authority server, subject to administrative constraints, and independent of the object's name. This gives flexibility to the assignment and reassignment of authority servers. A list of authoritative servers is maintained for each user or a group of users. 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 will not require name changes in this case.

3 Design Alternatives for Large Electronic Mail Systems

We discuss three alternative methods for designing electronic mail systems that can be used in large and diverse distributed environments.

The three approaches are mail systems with syntax-directed naming, mail systems with location-independent access, and attribute-based mail systems. For the first two mail systems, the procedures of mechanisms for naming and addressing, message delivery, reconfiguration, and user migration are described. For the third mail system, procedures unique to attribute-based mail system are described. There are other important procedures common to most mail systems such as flowcontrol, message archiving and clean-up, error and failure handling, and user-interface. These procedures are not considered in this paper.

3.1 Electronic Mail System with Syntax-directed Naming Resolution

In this system, location-dependent hierarchical names are used. This approach provides a straightforward approach to the naming and resolution mechanisms. However, the rigidity of name structure and syntactic pattern matching requirements somehow reduce the system's flexibility. Below, we discuss some of the algorithms that can be used with a large mail system that uses syntax-directed name resolution.

3.1.1 Naming and Addressing

In a large system, the name space must be 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. A three or four hierarchy system can be applied to electronic mail. In this paper, we use a three level hierarchical name in the form of "region.host.user" to identify users of the computer mail systems. The name components are location dependent. The region name is globally unique, the host name is unique within a region, and the user name is locally unique within a host.

Each user is assigned one or more authority servers which store information about him, and assumes responsibility for reliable 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, and forwarding and delivering of messages. Assigning a single authority server for each user is not a reliable way to handle mail because if the server goes down, then the user is prevented from using 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 first secondary server, and so on. If one server fails, the user can still access the mail system through the next authority server in the list.

The list of authority servers consists of local servers and may contain non-local servers from the nearby neighboring regions. The length of the list depends on the probability of server failures and the degree of reliability required. The assignment of mail servers to users is based on user connection cost. The connection cost depends on the communication time between the server and the user, and the processing and the queuing time in the server. Below, we develop a new algorithm for assigning users to authority servers when the system is either initialized or reconfigured. The algorithm assigns users to servers such that two objectives are satisfied:

i. To minimize the user connection cost which is a function of communication time, processing time, and queuing time,

ii. To balance the expected load level among servers.

The algorithm first assigns the nearest local server to all users. Then it will move users from a server with a higher connection cost to one with lower connection cost until the connection costs are minimized. The algorithm and its variables are described as follows:

$$H_i = \text{Host } i, 1 \leq i \leq \text{Total Number Of Hosts};$$

$$S_j = \text{Server } j, 1 \leq j \leq \text{Total Number Of Servers};$$

$$C_{ij} = \text{Average communication time for a message between } H_i \text{ and } S_j \text{ (i.e. transmission time, propagation time , etc.)};$$
\[ P_j = \text{Average message processing time on server } j; \]
\[ L_j = \text{Current load on server } j \text{ measured by the current number of users assigned to server } j; \]
\[ M_j = \text{Maximum load on server } j \text{ determined by maximum number of users that can be assigned to server } j; \]
\[ \rho_j = \text{Estimate for system utilization of server } j = L_j/M_j; \]
\[ \beta = \text{Average message processing time of a message}; \]
\[ N_j = \text{Number of users on host } i; \]
\[ A_j = \text{Number of users of host } i \text{ assigned to server } j; \]
\[ W_j = \text{Weight assigned to communication time in connection cost formula}; \]
\[ \beta = \text{A very large constant}; \]
\[ W_j = \text{Weight assigned to server processing and service times in connection cost formula}; \]
\[ Q(P_j) = \text{Estimate for average waiting time at server } j; \]
\[ \beta = \rho_j/(1 - \rho_j) \text{ if } \rho_j < 0.99 \]
\[ TC_{ij} = \text{Total connection cost between host } H_i \text{ to Server } S_j \]
\[ = \text{Average Communication and Processing Costs} \]
\[ = \text{Average Communication time} \times \text{Communication Weight} \]
\[ + (Q(P_j) + \text{Average Processing Time for a request}) \times \text{Processing Weight} \]
\[ = C_{ij} \times W_j + (Q(P_j) + z) \times W_j \]

**procedure Initialization:**

\{'**Initialize TC_{ij}, connection cost is computed as a function of the communication time alone using the shortest-path zero-load (i.e., no traffic) algorithm between hosts and servers with all the other cost components ignored for now.**\}'

\{'**All users on a host are assigned to the nearest server.**\}'

for \( i := 1 \) to \( \text{TotalNumberOfHosts} \) do

\[ A_{ij} := N_j \text{ where } j \text{ is the column} \]
\[ \text{of the minimum } TC_{ij}, \forall k; \]

\{'**Obtain better cost estimates**\}'

for \( i := 1 \) to \( \text{TotalNumberOfHosts} \) do

begin

Compute \( TC_{ij}, \forall j \) by using the server loads in \( A_{ij} \) and the formula for total connection cost shown above;

end;

**procedure balancing:**

begin

repeat

\[ \text{change} := \text{false}; \]

for \( i := 1 \) to \( \text{TotalNumberOfHosts} \) do

begin

\[ S_{\text{min}} := \text{Select } S_{\text{min}} \text{ such that } TC_{ij} = \text{minimum } TC_{ij}, \forall k; \]
\[ S_{\text{max}} := \text{Select } S_{\text{max}} \text{ such that } TC_{ij} \text{ is maximum } TC_{ij}, \forall k \text{ such that } A_{ik} > 0; \]

if \( S_{\text{min}} \neq S_{\text{max}} \) and \( TC_{ij, S_{\text{min}}} < TC_{ij, S_{\text{max}}} \) then

assign one user on host \( i \) to \( S_{\text{min}} \), instead of \( S_{\text{max}} \), recompute connection costs from host \( H_i \) to servers \( S_{\text{min}} \) and \( S_{\text{max}} \) and the total cost to these servers;

end if

undo the previous action

(i.e. assign user back to \( S_{\text{max}} \) and adjust the connection cost appropriately);

end if

else

\[ \text{change} := \text{true}; \]

end if

end for loop;

until no more changes are needed;

\text{check if some of the servers are still overloaded;}

End;

The average waiting time on a specific server is approximated by the average waiting time of an \( M/M/1 \) queue. The following is an example to show how the algorithm works: Fig. 1 shows the topology and user distribution of our example. Servers \( S_1, S_2, \) and \( S_3 \) are in the same region. The average communication time is one time unit for all communication links. When the shortest-path zero-load routes are used then the minimum communication time between \( H_2 \) and \( S_1 \) for example, is 2 time units (one-way trip). The values of \( W_1 \) and \( W_2 \) are 4 and 1 respectively to force the algorithm to select the closest servers to the hosts (whenever possible). \( W_3 \) taken into consideration the round-trip communication delay of messages. The message processing time is considered 0.5 time units. The values of \( M_1 - M_2 = M_3 \) are equal to 100. After initialization, the server assignments and load distribution among servers are shown in Table 1. The final result of balancing the loads is shown in Table 2. It can be seen that users on one host may be assigned to different servers.

It is obvious that the algorithm can be made much faster if in each iteration more than one user is moved from the server with the highest connection cost to the server with lowest connection cost. Also, the speed of convergence of the algorithm depends on how close it is to the optimal value at the initialization stage. Another simple modification is needed to extend the algorithm to handle adding a server, a host, or a user starting from a specified configuration. The algorithm can be extended to assign the primary server instead of only the primary server.

The weights assigned to communication and processing costs give the designer the option of minimizing a cost function in which the cost of communication is different from the cost of processing. A final modification can be done to include variable communication delays by having approximate queuing delays that is a function of the channel utilization (in the above algorithm, we assume constant communication delays which are valid in the case of light loads on the channel).

3.1.2 Message Delivery

The message delivery process begins 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 message delivery.

a. Connection Setup: The user needs 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, it will contact the next one and will keep attempting to contact a server until it succeeds. One problem with this scheme is that it requires large overhead in maintaining the authority server list for each user in the user interface. Keeping a list for every group of users at each host can reduce the number of lists. However, the lists still need to be updated when there are changes in system configurations, i.e., adding or deleting a server. Another method to establish connection between a user and a server is through a name server. In this case, the problem is shifted to locating a name server.

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 properly. If the recipient is located within the local region then his server can be located directly from other servers in the region. Otherwise, the message is transmitted to one of the servers in the recipient region where the name resolution process continues.

c. Delivering the Message to Recipients: The hosts used by the users are not necessarily large computers. They can be personal computers, or workstations. The user may not be turned on all the time. Therefore,
the received messages are stored in the servers' storage space until the
users retrieve them. When a sewer receives a message on behalf of its
recipient, it tries to notify the nser 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 received message in
his own storage or delete it after he reads it. Another option can be
provided to allow a copy of the message to 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 \( S_1, S_2, \ldots, S_n \), mail will be deposited in the first active server from the
list. Servers may become unavailable because of failure or being
disconnected from the network. As a result, user messages may be
deposited in more than one server. To retrieve 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. This scheme will not check
servers when it is sure that they do not store any messages for the
user. The algorithm assumes that all servers and hosts synchronize
their clocks together which is a simple requirement if the synchroniza-
tion is done on the level of a coarse time unit (i.e., a "second" or even
a slower unit). For each user, the user interface records the time when
the user last checked his mail in the variable LastCheckingTime\[user\].
It also records the list of servers that were unavailable at the time of
checking and may have buffered some mail (the user's list is called
PreviouslyUnavailableServers). Each server records the time in which
the server had last recovered from failure or been initialized in
the variable LastStartTime\[server\]. 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\[user\] is
greater than the server's LastStartTime, this means that the server
has been unavailable for receiving mail for sometime since the user's
LastCheckingTime and that some mail might have been deposited on
other servers. In this case, the user needs to check with other servers in
the list. Also the user interface always checks with active servers that
are in the PreviouslyUnavailableServers\[user\] list. The following is
the pseudocode for the algorithm.

```plaintext
procedure GetMail(user);
{ this procedure will get all the user received mail from
his authority servers to his local host }
Begin
CurrentCheckingTime := CurrentTime;
Finished := False;
While not Finished and there are more Servers in the
authority list do
begin
\( S_i := \) next server in the authority server list;
if \( S_i \) is alive then
begin
get mail from server;
remove \( S_i \) from PreviouslyUnavailableServers\[user\]
list if it is on it;
if LastCheckingTime\[user\] >
LastStartTime\[server\] then
Finished := True;
end
else
if \( S_i \) not in PreviouslyUnavailableServers\[user\]
list then
add \( S_i \) to the PreviouslyUnavailableServers\[user\] list;
end {while};
{ Get old mail in servers that might have it but
were unavailable }
for all alive servers \( S_i \) in the
PreviouslyUnavailableServers\[user\] list do
begin
get all mail from server;
remove \( S_i \) from the PreviouslyUnavailableServers\[user\]
list;
end {for};
LastCheckingTime\[user\] := CurrentCheckingTime;
End;
```

3.1.3 Reconfiguration

Large distributed systems are constantly 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 can be done using a simple procedure that does not have to
balance the load on the servers. However, if many users are added,
and existing servers are overloaded, then new servers should be added.
The load, then, should be redistributed among the servers using the
algorithm for server assignment specified before.

b. Add/Delete 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 and our load balancing algorithm should be applied.

c. Add/Delete Servers: Adding a new server requires the system
to be reconfigured. Most changes will be localized to the region where
the server is added, although some changes are made to tables in all

Table 1: Initial server assignment and load distribution

<table>
<thead>
<tr>
<th>User</th>
<th>Servers</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host 1</td>
<td>S1</td>
<td>50</td>
</tr>
<tr>
<td>Host 2</td>
<td>S2</td>
<td>60</td>
</tr>
<tr>
<td>Host 3</td>
<td>S1</td>
<td>50</td>
</tr>
<tr>
<td>Host 4</td>
<td>S2</td>
<td>50</td>
</tr>
<tr>
<td>Host 5</td>
<td>S2</td>
<td>40</td>
</tr>
<tr>
<td>Host 6</td>
<td>S3</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 2: Final load distribution among servers

<table>
<thead>
<tr>
<th>User</th>
<th>Servers</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host 1</td>
<td>S1</td>
<td>28</td>
</tr>
<tr>
<td>Host 2</td>
<td>S2</td>
<td>28</td>
</tr>
<tr>
<td>Host 3</td>
<td>S1</td>
<td>29</td>
</tr>
<tr>
<td>Host 4</td>
<td>S2</td>
<td>59</td>
</tr>
<tr>
<td>Host 5</td>
<td>S2</td>
<td>35</td>
</tr>
<tr>
<td>Host 6</td>
<td>S3</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 3: Initial server assignment and load distribution

<table>
<thead>
<tr>
<th>User</th>
<th>Servers</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host 1</td>
<td>S1</td>
<td>100</td>
</tr>
<tr>
<td>Host 2</td>
<td>S2</td>
<td>100</td>
</tr>
<tr>
<td>Host 3</td>
<td>S3</td>
<td>20</td>
</tr>
</tbody>
</table>

Figure 1: Topology and user distribution used in the example
servers. First, the new server notifies all other servers about its being added and exchanges its identity 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. 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.

3.1.4 Migration of Users

Since the names in this system 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. A more flexible approach is presented next.

3.2 Electronic Mail Systems with Limited Location-independent Access

For a system that is changing and growing constantly, flexibility in configuring and reconfiguring the system is strongly desired. In a syntax-directed naming system, server assignment and name resolution are based on the syntactic characteristics of the name 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 more flexibility at the expense of increasing the overhead of name resolution and response time. Therefore, we need to make a compromise between flexibility and cost.

For efficiency purposes, the authority server for a user should be in close vicinity to the user's possible access locations. It is inefficient to allow a user to choose a server that is far away from him, despite its flexibility. In this model, we divide the name space into regions with maximum flexibility being allowed within a region. Server assignment and name resolution are less dependent on the syntax of names; users can move freely and can send or receive messages from any host inside a region 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 expedite 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 indicates the primary location of the user. It does not determine the current access point 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. Regions are divided into small groups of manageable size using some mapping functions. Users are assigned a list of authority servers using the algorithm described in the previous model.

3.2.2 Message Delivery

a. Connection Setup: Users set up a connection to a mail server through the user interface. A user always contacts the nearest active server. All servers in a region will cooperate 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. All servers can resolve local names within the region. A hash function is applied to the name to find out in which sub-group the name belongs. Then the name can be resolved within the context of that sub-group. 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 assume the responsibility of resolving the name and delivering the messages.

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 inform 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 with other local servers to find out the current location of the user. This scheme is the same as the previous system if the user does not move. Overhead is only incurred if a user moves 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. A region can be as large as a country or a state.

a. 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 such that existing servers are overloaded, new servers must be added and reconfiguration is needed.

b. Add/Delete hosts: In this design, a host can be added, deleted, or even moved within a region with ease. Since the name assignment and resolution are independent of which host the user belongs to, then adding a host has very little impact on the system. This host can assign unique addresses to users on it and it becomes their primary access location.

c. Add/ Delete Servers: When new servers are added, the system can be reconfigured easily because of the independence between server assignment and name structure. Reallocation of servers and reallocation of load can be done by changing the hashing functions. The main advantage of this mail system is that reconfiguration can be done easily without much overhead. Also it does not require changing of user names.

3.2.4 Migration of Users

In this design, since names are not host-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 low and does not change with the relative location inside the region. If a user moves from one region to another, the overhead of redirecting the mail from old location to new location may be high. This may also result in long response time for mail of the migrated users. Therefore, obtaining a new name for a user who plan to move for a long time may place less overhead on the system, although it might cause temporary inconvenience. Of course, a user can remotely access his old region and access his mail but remote access is usually slow and imposes large overhead on the network (i.e., very few characters are packed in every remote-access packet).

3.3 Mail Systems with Attribute-Based Naming

It has been projected that electronic mail will be a major means of communication in the near future, not only in the office environment, but also at home. It will become as popular as today's telephonic services and replace large part of the existing 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 full electronic mail addresses of possible recipients. This model, if properly designed can be a very powerful communication tool for tomorrow’s mail systems. Before we discuss the details of the attribute-based mail system, we first describe some applications of how the system can be used, so as to give some ideas about the characteristics and possible problems in such a system.

i. Directory Look-up

In our daily communications with other people, we seldom use their full legal name. Instead first names, last names, or nicknames are more preferred. In electronic mail system, names are assigned by the system. The names usually have very rigid structures. This imposes difficulties on using and remembering those names. People do not always remember the exact spelling of the full electronic mail addresses 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, 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 the same set of attributes. In this 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.

ii. Information Exchange

A large portion of business communications is between people who do not know each other 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, advertisements, recommendations from other people, mail lists and etc. These methods are not very convenient and efficient. By 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.

3.3.1 Issues in Attribute-based Mail Systems

In this design, a user can be identified by a name or by 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, nicknames, aliases, commonly misspelled names, nationality, social security number, job title, type of job, organization, type of organization, location, information concerning location such as city, state, country, education, expertise/specialty, experience, interests, and hobbies. One important issue that has to be dealt with in this scheme is related to revealing personal information. Users must have the option to limit the access to their personal information to specific groups or organizations.

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. In most current mail systems, a single request usually involves one recipient or a group of recipients. The number of recipients, their electronic addresses, and their locations are usually specified when mail is sent. In attribute-based mail systems, however, the names and locations of the recipients may not be known. 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 poor. Next, we discuss two important issues related to broadcasting and searching of mail.

A. Efficient Distribution of Attribute-based Mail

One interesting feature of attribute-based mail system is how to efficiently search for a class of customers in a large network. We assume that the networks on which the mail system is built, form a connected undirected graph with computers (i.e., hosts, servers, mail-forwarders, etc.) as nodes and the communication links as the edges. Each edge is assigned a finite weight cost. Starting from any node in the graph, we can find a path that goes through 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 [GAL83]. The Minimum-weight Spanning Tree (MST) is the spanning tree of minimum weight among all the possible spanning trees. A modified version for MST algorithm can be used for efficient broadcasting and searching as explained below.

i. A Distributed Algorithm for MST:

In a distributed algorithm for MST by Gallager [GAL83], we can find the MST for a graph with different edge weights. Each node performs the same local algorithm, which consists of sending messages over attached links and waiting for incoming messages from other nodes and processing these 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 is presented in [GAL83].

ii. 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. 2 shows an example of a back-bone MST and the local MSTs. The details of our modifications can be found in [VUEN97]. The algorithm was implemented and tested using simulation.

B. Cost Analysis and Flow-control

Attribute-based mail systems can generate a large amount of traffic due to its flexibility of addressing a group of people with a simple mechanism (i.e., no distribution list has to be available). This has the risk of flooding the network erroneously or intentionally without being aware of the cost or being able to pay the cost. It is very important to estimate the cost of broadcasting and searching before sending mail to the potential recipients and before getting their responses. Estimated cost can be used as a flow-control mechanism and/or for guaranteeing that the users can pay the costs. A detailed estimate should be available 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 level of search he wants to be done. The cost includes the communication cost for broadcasting to the targeted regions, 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 (processing costs are ignored here). When an MST is generated following the previous algorithm, a table
listing the costs for delivery 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.

Another advantage of using the MST for broadcasting is that it can
also be 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.

Upon receiving a request from the parent node in the MST, each
node sends the message to its children nodes, and waits for the messages
to come back from all the children nodes. It then combines them into
a single summary message and returns it to its parent node. Problem
may occur if one of the children nodes goes down while the parent
node is waiting. Therefore, a parent node should time out if it waits
for a certain period of time and the unavailable estimates can be marked
so.

4 Performance Criteria for Evaluating
Mail Systems

In this section, we establish a criteria for evaluating the performance of
electronic mail systems in general and large mail systems in particular.
The main performance measures are efficiency, reliability, flexibility,
and cost. Actually some of these performance measures may have con-
flicting requirements with each other. For example it is very difficult
to have a very efficient system which is a very flexible and a reliable
system at the same time. Therefore, it is necessary for designers and
administrators to weigh different alternatives and strike a balance be-
tween the benefits and the costs. Below, we discuss each of the above
performance measures in more detail.

4.1 Efficiency:

Efficiency is always a major concern in designing frequently-used ap-
lication software. In electronic mail systems, the efficiency criteria is
measured in terms of connection set-up time, message transportation,
message delivery, name resolution, message storage, caching capability
(i.e., the capability of maintaining a list of both frequently and re-
cently used names and addresses), and receiving server notification for
existence of mail.

4.2 Reliability:

A mail system must be reliable and secure such that users can have confi-
dence that their messages, once accepted for delivery, will be made
available to the intended recipient or returned with proper error mes-
sages. Reliability has to be achieved by replication of messages and
services, by reliable software, hardware and communication, and by
appropriate management of system resources. The requirements for
reliability is based upon mail-service availability, message flow control,
buffer clean-up, and consistency of information concerning users and
messages.

4.3 Flexibility:

A flexible system should have the ability to provide wide range of func-
tions, 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.

4.4 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.

5 Conclusions

Electronic mail has become one of the major means of communication
and information exchange. In this paper, we discuss three alterna-
tive designs for electronic mail systems in large and distributed en-
vironments. First, we consider mail systems with syntax-directed
naming, names of users are hierarchically structured and are location-
dependent. It provides a straightforward solution to the distribution
and management of name space in very large distributed mail systems.
Hierarchical names are in the form of "region@host@user." We also
developed an algorithm for distribute loads among servers system-
tically using some cost function. This new algorithm can be applied
to the initial server assignments and server reassignments when adding
or deleting hosts and servers. We also developed an algorithm for ac-
cessing user mail in such a system. The reliability of the system is
enhanced through the use of multiple authority servers and distribu-
tion 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 approximately one un-
der normal conditions.

Secondly, we improve the flexibility of mail systems with syntax-
directed naming by incorporating limited location-independent access
to improve the flexibility of mail systems 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 a
user can access the system from a host different from its primary host
in a region without the need to reassign names and servers. Another
advantage of this model is that the reconfiguration is simpler and is
performed with less overhead.

We further enhance the mail system flexibility by introducing an
attribute-based naming scheme. Mail recipients can be identified using
attributes which are more meaningful than just electronic addresses.
It can be used for mass distribution of mail as well as for information
exchange. Its usage in address lookup, information exchange, market-
ing and advertising can be very powerful. It can provide an excellent
match for users, and jobs, needs, knowledge, etc. The attribute-based
mail services will be very useful in the future to replace the paper mail
because electronic mail is cheaper, faster and easier to distribute.

In the near future, electronic mail systems should be able to transfer
messages that consist of different forms of data such as voice, video,
graphs, and facsimile. The integration of different types of data already
exists in some systems and has presented challenges to the research
community.

References

"Grapevine: An Exercise in Distributed Computing." Communications

[COM85] D. Comer and L. Peterson. "Issues in Using DARPA Do-
main Names for Computer Mail." Proceedings of the 9th Data Com-
unication Symposium, September 1985, pp. 158-164.

Distribution Algorithm for Minimum-Weight Spanning Trees," ACM
Transactions on Programming Languages and Systems, Vol. 5, No. 1,

[LAT85] Larry L. Peterson. "Defining and Naming the Fundamental
University, May 1985.

Computer Networks," Communications of ACM, Vol. 29, No. 10,

[SCH84] M. D. Schroeder, A. D. Birrell, and R. M. Needham. "Ex-
perience with Grapevine: The Growth of a Distributed System," ACM
1-23.

CSNET Name Server," Computer Networks, Vol. 6, No. 3, July 1982,