Mail-system differences can hinder the use of electronic networks. The key to easy communication is support for heterogeneous environments.

Many electronic mail systems are now available. Interconnecting these largely independent systems broadens the communities of users that can communicate electronically. When the systems are at different sites, the existing style of loose interconnection is sufficient. But when a local site has multiple mail systems, a user cannot easily perform tasks such as reading mail from all the systems during a single session. An integrated local mail environment that lets a user perform such tasks easily is desirable.

Such an integrated environment should have three properties. First, it should be inexpensive to implement the environment. Replacing the existing mail systems with a single, new system is too expensive. Second, it should be inexpensive to integrate a new mail system into the environment. As new systems are acquired, it should be easy to use their mail services effectively. Third, it should be possible to integrate diverse classes of mail systems into the environment.

We have designed and implemented the Heterogeneous Mail System as an environment with exactly these properties. Our implementation integrates our local mail environment, which includes Unix and Xerox mail systems.

Today's systems. Existing mail systems are best described in the context of a basic model of electronic mail systems such as that in Figure 1. A user sends and retrieves messages with a user agent, which typically provides message preparation and archiving facilities. A message is transferred from the originator to a mailbox for the recipient with the message-transfer system, which has several message-transfer agents. The message-transfer agents provide a store-and-forward path between the originator and the recipient's mailbox. A repository, which is a set of mailboxes, stores messages that have been delivered to their final destination but have not yet
been retrieved by the recipient’s user agent.

Generally, the interconnection of such mail systems is limited: Message-transfer agents in different mail systems communicate via gateways that share a standard protocol. Individually, these mail systems perform their tasks reasonably well. In concert, however, their independence introduces some difficulties, especially for local mail processing. These difficulties include poor addressability, mail unavailability, and mail inaccessibility.

Poor addressability arises because most electronic-mail addresses explicitly indicate the host on which the repository resides. Knowledge of different mail-system syntaxes and the specification of routing information may also be required. As additional mail systems are introduced, these problems are compounded. This problem is eased — but not eliminated — through the management-intensive use of aliases and forwarding addresses.

Mail unavailability arises because a user generally has a single repository that resides on a specific host — a host that is a single point of failure for mail delivery and receipt. With commonly used forwarding chains, all hosts in the chain must be available before mail can be transferred.

Mail inaccessibility arises when a user’s repository is not on the machine that the user is working on, forcing the user to connect explicitly to the host on which the repository resides.

Most mail systems are based on one of two models: the host-based model and the server-based model. The host-based model (including the services on Unix) is derived from the early messaging services on time-shared systems. A user agent submits its delivery requests to a message-transfer agent on the local host. The message-transfer agent delivers messages to a repository, which is a file on the local host. A user agent retrieves mail directly by reading and updating the file.

The difficulties of poor addressability, mail unavailability, and mail inaccessibility are particularly evident in host-based systems. Efforts like the Sendmail internetwork mail router⁹ have tried to reduce this burden by encapsulating information about heterogeneity in a configuration file. For example, a Sendmail configuration file includes a set of address-rewriting rules to resolve addresses with different syntax for use when sending to different hosts. Despite some success, many of the difficulties described earlier remain.

The server-based model (implementations of which include Xerox’s Grapevine⁵) lessens many of the problems that arise in host-based systems. Remote message-transfer agents, called mail servers, transfer messages but also act as a repository. Each user is associated with a set of mail servers rather than with a specific host. With only one name throughout the system, it is much easier to address mail to each user. Mail is delivered to any available server in the user’s set, and mail is retrieved from all available servers in this set. This increases the availability of mail: Only when a mail server receives a message and then crashes will a message be unavailable. Accessibility is also increased because users can retrieve messages while working on any machine that can communicate with the remote mail servers.

The server-based approach alleviates the problems of poor addressability, mail unavailability, and mail inaccessibility, but — until our Heterogeneous Mail System — it has been designed for and implemented in only a homogeneous environment. The Heterogeneous Mail System integrates diverse local mail systems, including host-based ones, using the basic structure of the server-based approach.
Our approach. Figure 2 shows the structure of our Heterogeneous Mail System. Each individual mail system in the environment is left essentially unchanged. However, three new interconnections are inserted between the mail systems to integrate the local activities of mail submission, delivery, and retrieval. These interconnections complement external gateways, providing improved mail services among the diverse local systems.

The first interconnection is inserted between the user agents and the message-transfer agents, letting a single user agent send mail directly to each local mail system, without routing through gateways. With the naming support service described later, this interconnection lets users be addressed by a single name throughout the entire local environment.

The second interconnection is inserted between the user agents and the repositories, letting a single user agent retrieve mail from several repositories, perhaps on different mail systems. This interconnection increases accessibility for users who have established addresses in different mail systems.

The third interconnection is inserted between the message-transfer agents and the repositories, allowing delivery to any available repository maintained for the user. This interconnection increases availability of mail messages.

Callahan and Weiser’s Norman is similar to the Heterogeneous Mail System in that it provides retrieval from and submission to distinct mail systems by providing a user agent that speaks to the message-transfer agents of the different mail systems. When a new mail system is added to the local environment, each Norman user agent must be modified to communicate with the new system. By introducing an additional level of indirection, the Heterogeneous Mail System achieves the same degree of interconnection while eliminating the cost of modifying user agents when you introduce a new mail system.

Standardization efforts like the Comité Consultatif International Téléphonique et Télégraphique’s X.400 recommendation are rightfully moving toward a common set of interfaces between mail systems. But our solution is practical now and is robust enough to last until X.400 is widely available.

In any case, the cost of implementing X.400 on every existing system is great. Also, as Redell and White have convincingly argued, there will always be old systems that are too entrenched to be modified and new systems that provide features unforeseen by a standard, which is itself too entrenched to be modified. In other words, standards are often superseded as needs and technology change.

The approach of the Heterogeneous Mail System is to accommodate multiple standards easily. As new standards are introduced, the Heterogeneous Mail System can inexpensively incorporate the systems that implement these standards. Thus, another way of looking at the Heterogeneous Mail System is as a standard that is exceptionally easy to satisfy.

System model

The added interconnections of our approach provide the foundation of the Heterogeneous Mail System model. The basic structure of our model is taken from the server-based model, in which mail is delivered to any available server in the user’s set, and mail is retrieved from all available servers in this set. Our contribution is a practical solution that realizes the server-based model in a heterogeneous environment composed of existing mail systems, including host- and server-based systems.

Our solution has a nice additional property: The internal interactions of individual mail systems are not changed, so users who prefer to can continue to use the system as is. Hence, the integration process is evolutionary, not revolutionary.

The resulting model is based on several key mechanisms:

- a global name space, which is needed to eliminate host-specific addresses within a mail system and to facilitate various mail-system naming conventions,
- a heterogeneous communication structure that supports connections between user agents and diverse message-transfer agents, and
- a uniform mechanism to access both host- and server-based repositories.

The model has three layers, each of which addresses specific aspects of heterogeneity. The top layer represents a single, logical mail system; the bottom layer represents the local systems being integrated; and the middle layer represents the mappings between the two. The key principle of the model is that each layer views the components in the next lower layer through uniform interfaces. Each component then implements the common interface on the basis of the mail system’s available facilities.

Figure 3 shows an instantiation of the model. The clients layer (top) is not part of the model. The components of this layer represent user agents invoking message submissions or retrieval operations and message-transfer agents invoking delivery operations. These clients are agents that have been modified or configured to use the interface provided by the Heterogeneous Mail System.

The top layer of our model represents the Heterogeneous Mail system server, which is responsible for providing this submission, retrieval, and delivery interface to Heterogeneous Mail System clients. Because the system-server interface is stable, its clients need not change when new underlying mail systems are introduced into the local environment. An additional responsibility of the system server is to manage the idiosyncrasies of naming across the different native mail systems, providing the illusion of a single mail system.
The next layer represents a set of mail semantic managers, one for each system. Each mail semantic manager is responsible for transforming operations in the underlying mail systems to and from operations that are defined by the system-server interface. The mail-semantic-manager layer insulates the implementation of the system server from concern about the implementation of the underlying mail systems. Mail semantic managers provide the illusion that each underlying mail system, whether host- or server-based, has the same semantics. This style of encapsulating heterogeneity is similar to that used in the Heterogeneous Computer Systems name service.

The bottom layer represents the existing local mail systems.

The components of the Heterogeneous Mail System model typically run as remote servers, which may be distributed and replicated. The Heterogeneous Computer Systems remote procedure-call facility is the basic communication mechanism between layers of the Heterogeneous Mail System model. Because the facility emulates various remote procedure-call mechanisms, the system server and mail semantic managers can be written on any machine that provides a remote procedure-call mechanism. Existing protocols, such as SMTP, continue to provide communication between distinct mail systems.

**Mail semantic managers.** The abstract interface exported by the mail semantic managers to the system server supports the interconnection of individual mail systems. All mail systems — no matter how primitive or sophisticated — provide basic submission, delivery, and retrieval operations. All mail semantic managers export an interface that has a generic version of these three operations:

- GetMessages returns the messages stored in repositories managed by the mail semantic manager.
- DeleteMessages removes the specified list of messages.
- SendMessage delivers a message to the specified recipient list.

Each mail semantic manager implements this abstract interface using the underlying operations of the mail system it manages.

Although the interface is identical for each mail semantic manager, the semantics come in two styles. The first style, used for Unix mail semantic managers, applies the operations to a single, local mailbox maintained for the user; thus, there is an instance of the Unix mail semantic manager for each host. The second style, used for Xerox mail semantic managers, instead applies the operations to multiple user mailboxes; thus, only one (logical) Xerox mail semantic manager is needed.

There are two styles of mail semantic managers because they reflect the implementation approaches of the two mail-system models: host-based and server-based.

### There are two styles of mail semantic managers because they reflect the implementation approaches of the two mail-system models: host-based and server-based.

The Unix mail semantic manager was implemented by carving out some of the code from the standard Unix 4.3 Berkeley Standard Distribution mail program and encapsulating the program as a Heterogeneous Computer System remote procedure-call server. Because existing code implemented the exported operations, only slight modification was required to match the syntax and semantics of the mail-semantic-manager interface.

Xerox user agents already existed to access multiple mail servers, so it was simpler to construct a Xerox mail semantic manager that accessed all Xerox mailboxes in a single invocation.

**Heterogeneous Mail System server.** The system server exports a simple interface to Heterogeneous Mail System clients. The system-server interface is nearly identical to the mail semantic manager interface and is implemented largely in terms of the underlying mail semantic managers. The primary difference is that each operation applies not to a particular subset of a user’s mailboxes but to all the mailboxes maintained for a user throughout the local environment. Thus, each client of the system server can simply invoke these operations regardless of which mail semantic managers (and which mail systems) are used to implement the interface. By providing the view of a single mail system to Heterogeneous Mail System clients, no further modifications are needed when new underlying mail systems are introduced into the local environment.

A user or message-transfer agent becomes a client of the Heterogeneous Mail System by replacing calls to its system-specific submission, retrieval, and delivery operations with functionally equivalent calls to the system server. In many cases — especially with local message-transfer agents these changes can be made by altering configuration files. In any case, the required modifications are simple.

The system server distinguishes between the two mail-semantic-manager styles. For the first style, which handles only a single repository, the system server iterates through a list of mail semantic managers. For the second style, which handles multiple repositories, the server invokes only a single mail semantic manager. The style of each mail semantic manager is registered in the Heterogeneous Computer Systems name service so the system server can select the proper implementation structure at runtime.

The system server also handles differences in naming conventions. Our model does not replace the user-level naming schemes; users continue to use the naming scheme defined by the underlying local mail systems. Instead, it defines a scheme for mapping names in the local environments to a global Heterogeneous Mail System name space. This name space and the associated data helps control mail delivery and retrieval, as well as helping translate names across different local mail systems.

The name space is contained in several databases maintained by the system server. To take advantage of existing name services, like the Unix Bind and Xerox Clearinghouse, each database has a two-
level, indirect structure.

The top level maps Heterogeneous Mail System names to a (name service, local name) pair. The first entry designates the local name service that contains the desired data; the second entry designates the name by which that data can be found in the local name service.

This approach avoids deregistration, avoids problems with consistency, and provides independent administration and maintenance of mail-specific data by letting each mail system manage its own databases. Uniform access to the existing databases is provided by the Heterogeneous Computer Systems name service.

For existing mail systems that do not register information in a name service, we register the necessary information with a local name service. Each Heterogeneous Mail System database provides a user interface comprised of three simple functions: query, add to, and modify.

The mailbox database maintains mailbox information for each Heterogeneous Mail System user, some of whom have mailboxes in multiple mail systems. A Heterogeneous Mail System user is a local user who has registered a master mailbox list with the Heterogeneous Mail System and uses a Heterogeneous Mail System user agent. The user’s master mailbox list, which is associated with the user’s Heterogeneous Mail System name, contains the list of mail systems where the user receives mail. Our model guarantees unique Heterogeneous Mail System names because these names are internal and therefore not seen by users.

The bottom level of the database contains the individual mailbox lists in each mail system comprising the Heterogeneous Mail System. Figure 4 illustrates the entries in the Heterogeneous Mail System mailbox database for a user with the Heterogeneous Mail System name “squillante.” This user has both Unix and Xerox mailboxes registered with the Heterogeneous Mail System, with the Unix name “ms” and the Xerox name “Mark S. Squillante:CS:UWash” (spaces are both legitimate and common in Xerox names).

Because a user may invoke the Heterogeneous Mail System from any local system, the Heterogeneous Mail System provides a mapping from each local name for the user to the user’s Heterogeneous Mail System name. It guarantees a unique mapping by prefixing the user’s name in one mail system with the name of that mail system.

This many-to-one mapping is contained in the Heterogeneous Mail System’s global alias database. The global alias relation maps, directly or indirectly, any proper local name for a Heterogeneous Mail System user to the user’s Heterogeneous Mail System name.

Given this name, the mailbox database can be queried to obtain the user’s master mailbox list; this information is then used to query the individual name services specified in the master mailbox list and retrieve the user’s various mailbox lists. Thus, squillante’s Unix mailbox list is obtained by querying Bind, the name service associated with Unix mail, with the key “ms.” Similarly, his Xerox mailbox list is obtained by querying the Clearinghouse, the name service associated with Xerox mail, with the key “Mark S. Squillante:CS:UWash.”

The system server also modifies the headers of messages that pass across local mail systems’ boundaries. Consider a message that is sent from the Xerox mail system to someone on a Unix host. If the system server passes the request unchanged to a Unix mail semantic manager, replies to the original message by a non-Heterogeneous Mail System user agent could fail because the return address may not be understood by the remote Unix host.

To prevent this, the system server modifies the reply-to address to use a corresponding Unix name before passing the request to the Unix mail semantic manager. Rewriting rules for each mail system, similar to those used by Unix’s Sendmail are registered in a system-server database. By registering appropriate rules, new mail systems can be added to the Heterogeneous Mail System environment without modifying the system server.

The Heterogeneous Mail System makes no assumptions about message content. All fields are passed through Heterogeneous Mail System clients so that those that can take advantage of new features do so. Other clients can simply ignore these fields. One problem is that passing a message from one mail system to another may require conversion between formats. Our model solved this problem by using an approach like that in the Heterogeneous Computer Systems remote procedure-call facility, where multiple representations are handled by selecting translation routines dynamically when a server becomes associated with a client.

Mail submission and delivery. The interaction of Heterogeneous Mail System components follows the server-based model. Using a Heterogeneous Mail System user agent, a Heterogeneous Mail System originator prepares the message, specifies the list of recipients, and submits the message to the system server. The system server authenticates the caller, converts the message into the Heterogeneous Mail System format, obtains the originator’s Heterogeneous Mail System name by using the global alias database, and translates each recipient’s name by using the originator’s alias list and the global list of aliases. During alias translation, distribution lists are expanded and Heterogeneous Mail System names are obtained, if they exist.

Based on name syntax, the system server partitions the aliased recipient list into a

---

November 1989 63
set of sublists, one per mail-system type plus one for Heterogeneous Mail System users. For each non-Heterogeneous Mail System recipient sublist, the system server locates an available mail semantic manager corresponding to the sublist type and passes the request to this mail semantic manager. The mail semantic manager authenticates the caller, converts the message into the format required by the local message-transfer agent (delivery agent), and invokes the agent.

For recipients who are Heterogeneous Mail System users, the process is more complicated. Consider a Heterogeneous Mail System recipient sublist containing a single Heterogeneous Mail System user named "squillante."

The system server queries the mailbox database to obtain the user’s master mailbox list, which in this case contains both a Unix and a Xerox entry (see Figure 4). The system server queries the Heterogeneous Computer Systems name service for the list of Unix mail semantic managers. The result indicates that each mailbox is handled by a separate mail semantic manager, so the system server queries Bind for the list of Unix mailboxes associated with "ms", his Unix name. The system server then selects any one of the designated Unix mail semantic managers and invokes SendMessage, passing it "ms" and the message.

The mail semantic manager authenticates the caller, converts the message into the Unix format, and completes delivery by appending the message to the local file /usr/spool/mail/ms. If the mail semantic manager cannot deliver the message, the system server tries the other designated Unix mail semantic managers until one succeeds or until the list is exhausted.

If none of these calls is successful, the system server queries the Heterogeneous Computer Systems name service for the list of Xerox mail semantic managers. The result indicates that each mail semantic manager handles all Xerox mailboxes, so the system server selects a Xerox mail semantic manager and invokes SendMessage, passing it "Mark S. Squillante:CS:UWash" and the message. The mail semantic manager authenticates the caller, converts the message into the Xerox format, and completes delivery by calling one of the Xerox mailing-stub routines. If the mail semantic manager crashes, the system server tries another Xerox mail semantic manager.

If the Heterogeneous Mail System sublist contains more than one user, the system server obtains each user’s master mailbox list, choosing a mail semantic manager for each on the basis of mailboxlist ordering and its opinion of the accessibility of the various mail semantic managers. The individual recipient names — corresponding to the recipient’s name in the mail system managed by the chosen mail semantic manager — are accumulated in steering lists, one for each mail semantic manager to which the message should be forwarded. The system server selects each of these mail semantic managers and invokes SendMessage, passing it the message and appropriate steering list; the delivery algorithm proceeds as described earlier.

In the case of submission by a non-Heterogeneous Mail System originator, the originator’s user agent passes the request to its corresponding delivery agent, which has been configured or modified to relay message delivery requests to a system server. The system server then functions as described earlier.

If the delivery agents of a system cannot be configured or modified to use a system server, Heterogeneous Mail System addresses must not be passed to these agents. Otherwise, these agents might consider a legitimate Heterogeneous Mail System recipient name to be erroneous.

This should happen rarely because most mail-transfer agents let you specify the location of mail gateways.

However, the Xerox mail servers cannot be modified because the source code is unavailable and the servers are not configurable. In our prototype, we modified the single Xerox user agent and installed it on all our Xerox workstations. Replies to messages that originated in other mail systems by a non-Heterogeneous Mail System user are handled properly because the Xerox user agent submits all delivery requests to a system server. Error returns, such as returning a message to its originator because of an erroneous recipient name, are also handled properly since such errors are detected by Xerox mail semantic managers before the delivery request is passed to a Xerox delivery agent.

Mail retrieval. A Heterogeneous Mail System recipient uses a Heterogeneous Mail System user agent to retrieve messages from an available system server. The structure of mail retrieval is identical to mail submission, except that every mail semantic manager (and thus every mailbox) is checked, since messages might be stored in every mail system. Consider retrieving mail for the Heterogeneous Mail System user "squillante."

As in mail submission, the system server authenticates the caller, obtains squillante’s master mailbox list, obtains the list of Unix mail semantic managers corresponding to ms’s list of Unix mailboxes, and obtains the list of Xerox mail semantic managers.

The GetMessages command, with argument "ms", is then passed to each of the Unix mail semantic managers; similarly, GetMessages, with argument "Mark S. Squillante:CS:UWash," is passed to the Xerox mail semantic manager. Each mail semantic manager authenticates the caller, acquires the specified messages from the associated repository, converts each into the Heterogeneous Mail System format, and returns them to the calling system server. The system server merges the retrieved messages and returns them to the calling user agent.

The Heterogeneous Mail System user agent then provides a familiar local session, where messages are read, replied to,
sent, and deleted. Messages that the user deletes are only deleted in the context of the user agent; they still remain in the mailboxes scattered throughout the system. When the user’s session is completed, the Heterogeneous Mail System user agent constructs a delete list corresponding to all the messages deleted during the session. This list is passed to the system server with a request to delete them. The delete list has an internal control field that includes the message’s mail-system type, mailbox name, and index in this mailbox — this approach relieves the Heterogeneous Mail System and mail semantic manager servers from having to retain state information about each message.

If a set of messages cannot be deleted (perhaps the host on which they are stored is down), the system server informs the caller of the problem and functions as if the corresponding delete sublist was empty. Thus, in our prototype, a user could retrieve a message that had been previously “deleted.” Experience with the prototype indicates that this is not a serious problem, as long as it does not occur often and the user is notified when it does.

Accommodating new systems. A primary objective of the Heterogeneous Mail System was to reduce the cost of integrating new mail systems into the existing local environment. There are four steps in adding a new mail system to the Heterogeneous Mail System.

First, mail semantic managers for the new mail system must be built. Our experience is that this task is straightforward, since somewhere in the mail system being integrated there is code to perform these operations. It is simply a case of finding it, encapsulating it, and massaging it slightly to fit the desired syntax and semantics.

Second, a Heterogeneous Mail System user agent must be made available, either by porting an existing Heterogeneous Mail System user agent or by modifying an existing user agent of the new mail system. Our experience shows that modifying existing mail-user-agent code is easy because it consists of replacing calls to system-specific submission, retrieval, and delivery operations with functionally equivalent calls to the system server.

Third, the new mail system’s delivery agent must be configured or modified to use a system server as its delivery mechanism. This is usually a minor task because most mail systems have a configuration mechanism to specify the location of gateways. If such a change is impossible, all user agents in the new mail system must be modified to use a system server.

Fourth, information about the new mail system must be registered with the Heterogeneous Computer Systems naming service. This information is used by the system server and includes data like the locations of mail semantic managers managing the system.

None of these steps requires modifying the system server. Heterogeneous Mail System clients, or mail semantic managers of other mail systems — the changes are restricted to the new mail system’s environment. As soon as these changes are made, the new mail system is fully integrated into the Heterogeneous Mail System.

Security and authentication. Authentication in mail systems prevents recipients from accessing other users’ mail and prevents an originator from posing as another user. Because the Heterogeneous Mail System employs various existing mail systems, it cannot provide greater security than those systems do. But the Heterogeneous Mail System model does not weaken the security provided by any constituent mail system, since the interfaces exported by each component of the Heterogeneous Mail System model include a parameter for authentication data. This information, along with the authentication mechanisms of the mail systems comprising the Heterogeneous Mail System, is used to maintain the security provided within a single mail system and to prevent breaches in security when an operation requires crossing mail-system boundaries.

Before executing a request, each mail semantic manager uses the supplied authentication data to authenticate the caller in the system managed by the mail semantic manager. If successful, the mail semantic manager carries out the caller’s request; otherwise, the mail semantic manager does nothing and notifies the caller that there is an authentication problem. Similarly, the system server authenticates its caller in the originating system. The system server facilitates authentication across different mail systems by including authentication data for each system in the caller’s master mailbox list. All data added by the Heterogeneous Mail System to local name services — including the master mailbox lists — are stored in an encrypted form.

Because the Heterogeneous Mail System performs a request only if the caller is authenticated, recipients are prevented from accessing other users’ mail messages and local originators are prevented from posing as others. However, the delivery agent of a local mail system may make a request on behalf of a user outside the local environment. In this case, we partly rely on the authentication mechanisms of the individual systems to authenticate the originator. The call to the system server, however, will require authentication of the delivery agent before the delivery request is performed.

Scalability. The issue of scale in distributed systems has three dimensions: the number of users and components, the distance between components, and the number of separate yet cooperating administrative domains. The general techniques of distribution, replication, and caching are typically used to address these three dimensions. Our model uses the well-known techniques to resolve the first and third dimensions. Because our work focuses on the local mail environment, we do not directly address the second dimen-
sion; we believe the issue of component distance should be resolved by the individual mail systems.

The first dimension requires that the work to deliver, retrieve, or delete a message be bounded as the numbers of local mail systems and users grow. Also, a single component should not become a bottleneck. In our model, most of the required processing resides with the individual mail systems; additional processing is distributed across the Heterogeneous Mail System server and the mail semantic managers. These components are replicated, and any replica can handle requests. We use information caching to further reduce the load on these components. Furthermore, the number of operations required by each Heterogeneous Mail System component to perform a user request is fixed to the number of mail systems associated with the request.

Most of the data required for each user resides with the individual mail systems. Additional data about the users and about individual mail systems are stored in a distributed and replicated name service. Although data is maintained for each mail system and for each user, the amount of information is relatively small.

For administration, the Heterogeneous Mail System Model scales naturally because, like their operation, individual mail systems continue to be administered independently. The additional data required by the Heterogeneous Mail System is small and easy to administer locally.

Experience

Our experimental prototype, which integrates our local Unix and Xerox mail services, demonstrates our approach’s utility. A primary goal was conserving development effort; we wanted to demonstrate that the Heterogeneous Mail System model actually eases the integration of existing mail systems. A secondary, but still important goal, was making the resulting system efficient.

Prototype. The prototype has Unix and Xerox mail semantic managers, a system server, and two Heterogeneous Mail System user agents (one running under Unix and the other running under Xerox XDE). The mail semantic managers are replicated and distributed throughout the corresponding mail environments. We modified the existing user agents to use a system server for submission and retrieval, and we configured Sendmail to use a system server for delivery. We implemented, replicated, and distributed a single system server throughout the Unix environment. The system server is accessed via Courier RPC by the Xerox/Heterogeneous Mail System user agent and is linked directly with the Unix/Heterogeneous Mail System user agent.

Our prototype also provided mail service to Tektronix 4404/4405 hosts not reachable over the network.

Much work in a distributed mail system can be performed in the background, potentially providing significant improvements in performance. We used such techniques effectively in our prototype. As an example, after the system server passes a delivery request to the appropriate mail semantic managers, the completion of delivery is handled in the background by the mail semantic managers and the mail systems they manage.

Many of the tasks performed by the system server in carrying out a request are independent and can be executed in parallel. Our prototype uses multiple threads of control to reduce the waiting time introduced by the synchronous nature of remote procedure call. For example, the system server uses multiple threads to access different mail semantic managers concurrently. Because Heterogeneous Mail System users have multiple mailboxes, this approach improves the performance of Heterogeneous Mail System user agents. As another example, functions performed by the system server on recipients of a delivery request, such as alias translation, execute concurrently.

We used caching to reduce the number of calls made by the system server and mail semantic managers to access their data. Work on the Heterogeneous Computer Systems name service has shown this to be valuable, and our system provides similar gains because its structure is similar.
Our effort had two primary objectives. First, we wanted to improve the quality of the mail service within a local environment made up of several largely independent, heterogeneous mail systems. Second, we wanted to ensure that integrating new and different mail systems would be a relatively inexpensive operation. We met these objectives by adopting the server-based model, which increases the quality of mail service, and augmenting it with a level of indirection that helped accommodate mail systems' heterogeneous aspects.

The Heterogeneous Mail System represents a general approach to sharing a service across diverse systems while decreasing the cost of integrating new systems. The Heterogeneous Computer Systems project, of which this work is a part, uses closely related software structures to permit the addition of new services and systems. Taken together, these structures let us share service among systems while avoiding the definition of new standards that would be costly to implement on current and future systems.

References

Acknowledgments
We thank the other Heterogeneous Computer Systems project participants: Brian Bershad, Jon Bertoni, Andrew Black, Fran Brunner, Dennis Ching, Bjorn Freeman-Benson, Kimi Goetsch, Ed Lazowska, Hank Levy, John Maloney, Cliff Neuman, Brian Pinkerton, Jan Sanislo, Mike Schwartz, James Syrge, Doug Wiebe, and John Zahorjan.

Doug Wiebe contributed to fruitful discussions about the Xerox Mail Service throughout the Heterogeneous Mail System design process and helped implement the Xerox mail semantic managers. Bill Grisold, Ed Lazowska, Cliff Neuman, Brian Pinkerton, Jan Sanislo, and Doug Wiebe provided helpful comments on earlier drafts.

This material is based on work supported in part by the National Science Foundation under grants DCR-8420945, CCR-8611390, and CCR-8855804, by the Xerox Corp. University Grants Program, and by the Digital Equipment Corp. External Research Program.

Mark S. Squillante is a doctoral candidate in the Computer Science Department at the University of Washington. He has spent two summers in the systems theory and analysis group at the IBM T.J. Watkins Research Center. From 1982 to 1985, he was a member of the technical staff at AT&T Bell Laboratories in Murray Hill, N.J. His research interests include modeling and analysis, design and implementation, and distributed and parallel systems.

Squillante received an MS in computer science from Columbia University and a BA with first honors in mathematics from Washington College in Chestertown, Md. He is a member of IEEE and ACM.

David Notkin is an assistant professor of computer science at the University of Washington. He was a charter member of the Gaiselh project at Carnegie Mellon University. His research interests include software-development environments for both parallel and sequential machines, extensible software systems, software engineering, and heterogeneous computer systems.

Notkin received a PhD in computer science from Carnegie Mellon University and a BS in computer science from Brown University. He is a member of IEEE, ACM, and Sigma Xi. He received a National Science Foundation Presidential Young Investigator award in 1988.

Address questions about this article to Notkin at Computer Science Dept., University of Washington, Seattle, WA 98195

November 1989