Project Athena as a Distributed Computer System

George A. Champine, Digital Equipment Corporation
Daniel E. Geer, Jr., Massachusetts Institute of Technology
William N. Ruh, IBM

Project Athena was established in 1983 to improve the quality of education at Massachusetts Institute of Technology by providing campuswide, high-quality computing based on a large network of workstations. The major sponsors of Athena in addition to MIT are Digital Equipment Corporation and IBM.

Although the approximate outline of the system was understood early in the project, the design evolved considerably over the first few years as the requirements became clearer. Today, Athena has met its design goals and is a fully operational production system. It consists of 1,000 workstations, in 40 clusters of 10 to 120 workstations each, for use by students 24 hours a day. One of the clusters is an electronic classroom. In addition to the student clusters, there are nine instruction development clusters and two facilities equipped to project images from workstation display screens. Workstation clusters have been installed in five student housing facilities, some with workstations in bedrooms and others with workstations in common areas.

Currently, the system provides 10 Network File System file servers, two Andrew File System file servers, 24 Remote Virtual Disk file servers, 79 Postscript printers, three each of name servers and post office servers, and two authentication servers. It supports the Digital VAXstation and the IBM RT/PC. In addition to the standard Athena monochrome workstations, there are 15 multimedia workstations that support full-motion color video. The system has 40 gigabytes of disk storage in the workstations and an additional 50 gigabytes of disk storage in network file servers. Each workstation has from 50 megabytes to 100 megabytes of disk.

Now providing 10,000 students and faculty with a variety of network services, MIT's educational workstation system is designed to grow to 10 times its present size.

The Athena system is one of the largest centrally managed educational workstation networks. It presently serves about 10,000 active user accounts, which generate about 4,000 logins and 9,000 mail messages per day. The average student uses the system eight hours per week. In aggregate, users generate 12,000 questions per semester for the on-line consulting system, and print 3 million pages per year. About 90 percent of undergraduate students and 50 percent of graduate students use the system. Usage is increasing about 15 percent per year, as more students use the system and use it for more hours per day.

A major benefit of Athena is that students need learn only one system for educational computing. Athena presents a coherent model of computing in which all applications can run on all supported workstations, independent of architecture. Because of the strong level of coherence, the human interface to the system is independent of the type of workstation being used. Thus, only one training program and one set of documentation are needed.

The initial equipment for Athena was a time-sharing environment installed in late 1983, utilizing about 63 VAX 11/750 and associated ASCII terminals running Berkeley Unix. About 160 PC ATs also were used. Although the PCs did not run Unix, they provided the Athena designers valuable experience in deploying networked

0018-9162/90/0900-0040$01.00 © 1990 IEEE

workstations and in managing the distribution of software to them.

The first workstations were introduced in late 1984 for staff use. The first public cluster of workstations for student use opened in March 1985. Deployment has continued aggressively since then. As workstations became more plentiful, the time-sharing system was used less and less. It was phased out entirely in September 1987 in favor of a uniform workstation environment, and the time-sharing minicomputers were converted to file servers.

This article describes the design of Athena's distributed workstation system. Because the Athena project is a large undertaking, we include here only the distributed aspects of the system. Sources of additional information are listed in "Further reading" at the end of the article.

Requirements

The first work in the Athena project was to decide on the requirements of the system. Some of the requirements established are characteristic of campus computing in general; others are unique to MIT. The following requirements of a distributed system were identified as necessary for meeting the needs of instructional computing on campus:

- *Scalability*. The system must scale to support 10,000 workstations or more.
- *Reliability*. The system must be available continuously 24 hours a day, even though equipment failures occur frequently in a system this size.
- *Public workstations*. Any user must be able to use any workstation.
- *Security*. System services must be secure even though workstations are not.
- *Heterogeneity*. The system must support a variety of hardware platforms.
- *Coherency*. All system applications software must run on all workstations.
- *Affordability*. The cost to own and operate must not exceed 10 percent of tuition on a sustaining basis.

System support of public workstations is necessary because workstations are presently too expensive to be purchased by individuals. The plan is to allow (but not require) individuals to purchase workstations when they become affordable.

Concerned about the complexity that might result from the system's size, the designers have followed a policy of eliminating needless complexity throughout the project.

The mainframe model would have serious drawbacks in the Athena environment.

Definitions

We define a *user* as a person who uses a subsystem (such as a workstation), program, or service. A *client* is a user or a program acting for a user. A *server* is a provider of services or resources. A *service* is a set of actions to be performed.

An object is referenced by its *name*. The object is located by its address or (more generally) by its path. A *binding* of the name to the object occurs when the name is associated with the address. Often part of the address is contained in a context of the name. Some systems use an address for a name. Although this may simplify system development, it can cause problems in binding a name to a different object. *Resolving* a name means identifying the address related to a name.

Coherence refers to the ability of two distinct hardware architectures to compile and run the same software. *Interoperability* is the ability of two subsystems to cooperate in the execution of a single task. For example, two subsystems supporting the X Window System (described later in the article) can cooperate on a single task as client and server because they support the same network protocol.

Security has two major aspects: authentication and authorization. *Authentication* is determining the identity of the user. *Authorization* is determining that the user has legitimate access to the requested resources.

Fail-soft refers to the ability of a system to continue to operate in spite of the failure of a subsystem, possibly with degraded performance.

Models of distributed systems

Many different models of distributed systems have been proposed. One extreme model might be called the *mainframe model*. In this model each workstation or other node in the distributed system can exchange files with other nodes subject to security restrictions, but nodes cannot work together in any other manner. All resource allocation, security access, and functional access are handled on a per-node basis.

If a user in a distributed system based on the mainframe model wants access to a printer on another node, the user must log in to that node, gain security authorization on that node, move the file to be printed to that node, and then command the print function. A major benefit of the mainframe model is that presently available time-sharing operating systems (such as Unix) support it and can be used to implement it with minimal development.

Another extreme model could be called the *unified model*. In this model all nodes in the distributed system are considered part of one logically unified system. Resource allocation, security, and access to function are handled at the system or network level, not at the node level.

If a user in a distributed system based on the unified model wanted to do the same print function, the user would simply issue the print command, logical printer name (if not the default), and file name. By logging in on the local workstation, the user becomes logged in to all services provided anywhere in the system, and all system resources are accessible transparently.

Athena is a single unified model, similar to a single time-sharing system, although it consists of more than 1,000 individual computers. The major benefit of the unified model is that it automates much of the user control that must be supplied manually in the mainframe model. Indeed, the unified model has most of the characteristics of the single-mainframe model.

Although the mainframe model would have been easier to implement than the unified model, it would have serious drawbacks in the Athena distributed-system environment. The mainframe model maintains system integrity by preventing users from obtaining access to the kernel of the operating system. In the Athena environment users can get access to the workstation kernel either by obtaining the superuser password or by booting their own operating system. Thus, workstation integrity cannot be assured. Since users can corrupt the kernel of the operating system, they can masquerade as others or as a system service. The user can also "infect" the workstation with a Trojan horse, a virus, a worm, or other undesirable code. Of course, the user can monitor all Ethernet messages on the local net.

The mainframe model also assumes that users are attached to a particular machine. User files exist on that machine, and mail is sent to the user on that machine. Because one of the Athena objectives was to let any student use any workstation, this restriction was undesirable. Instead, the system should provide mail and file access as network services, accessible from any workstation, independent of location.

Access to these network services (described later) has security implications. If user files or system software were stored on a public workstation, a user could damage (or delete) files. A user could corrupt the operating system, perhaps by inserting a Trojan horse that would capture passwords of subsequent users. Since any user can use any workstation, sending mail to a machine is not suitable for the Athena environment. A preferable alternative is to send mail to "username" through a network service, and to allow mail to be read, written, and filed from any workstation.

The mainframe model also had undesirable support implications. The classical support model for mainframes is a system manager per system. At a scale of the order of 10,000 systems, this was clearly not appropriate. The ultimate objective for Athena is to have a system manager per 1,000 workstations, requiring a three-order-of-magnitude improvement over conventional approaches. (Presently, five operations programmers support 1,000 workstations.)

Other distributed operating systems

Several other distributed operating system projects have requirements similar in some respects to Athena's. Two of them, Amoeba and Grapevine, are designed for larger environments than a single campus, but they have addressed many of the same issues as Athena.

Amoeba[1] is designed to run in both a local and extended network environment. A typical local environment consists of perhaps 16 processors connected to a pool of several tens of workstations. Beyond that, multiple Amoeba sites can be interconnected through a wide area network to form a single system. Amoeba research focuses on the use and management of the processor set in addition to communications and protection. An extended system interconnects local Amoeba systems in a wide area network through several countries in western Europe.

The Athena and Andrew projects had similar objectives but approached them with different priorities.

Andrew[2] is a system for support of instructional and research computing at Carnegie Mellon University, with objectives very similar to Athena's. Both systems support on the order of 1,000 midrange workstations in a Unix and Ethernet environment, with plans to scale to 10,000 workstations. Both assume that the user has complete control over workstation functions, with a central mechanism controlling and supporting network services.

A distributed file system called the Andrew File System (AFS), developed as part of Andrew, is now being used as the basis of a nationwide file system experiment. The AFS file space is separated into two parts: local and shared space. The local space is accessible to the user (generally on a local disk) but not publicly shared. The shared space can exist anywhere in the network and is publicly accessible.

In contrast to the systems just described, Dash[3] is intended for very large hardware and systems of the future. It is designed for far faster and cheaper processors and networks, implemented in numbers of thousands to millions, with worldwide dimensions.

Eden[4] is an object-oriented operating system based on a single remote procedure call mechanism. It has a single, uniform, systemwide name space spanning multiple machines. An object is a set of processes that is referenced by capabilities and can migrate freely among systems. All objects have a data part, which includes short- and long-term data. Objects can checkpoint autonomously. According to the definition given earlier, Eden is a unified distributed system.

Grapevine[5] is an older system, designed for much larger geographic distribution. It supports message delivery, resource location (naming), authentication, and access control services. Grapevine's primary use is delivering mail. It has rapidly come to be used throughout the Xerox network, with nodes located in clusters around the world. Resources are accessed within the local cluster, but access is allowed to any other Grapevine system in the Xerox network. Grapevine consists of about 1,500 users and 17 servers in 50 local area networks.

The objective of HCS (Heterogeneous Computer Systems)[6] is to integrate different hardware/software combinations into a single system. Based on heterogeneous communications protocols including TCP/IP (transmission control protocol/Internet protocol), it uses a single global name space for the entire heterogeneous environment along with remote procedure calls. The network services supported are remote computation, mailing, and filing.

Locus[7] is a distributed operating system that supports transparent access to data through a networkwide file system. It permits automatic replication of storage and transparent distributed process execution. Locus also provides automatic replication of resources to meet reliability requirements and can degrade gracefully with network and node failures. Locus is another example of a unified distributed system.

Mach,[8] supporting both tightly coupled and loosely coupled general-purpose multiprocessors, also supports transparent remote file access between autonomous systems. It has large, sparse, virtual address spaces, copy-on-write virtual copy operations, and memory-mapped files. Multiple threads of control are provided within a single address space. Large amounts of information can be transferred by the interprocess communications facility by means of copy-on-write techniques. Mach has a binary applications program interface compatible with Berkeley Unix 4.3.

Sprite[9] is an operating system that supports multiprocessing and distributed files. Its user-level facilities are identical to BSD Unix. All files and I/O devices are uniformly accessible to all systems, and they appear as a single shared hierarchy. There are no private partitions to manage, and devices and files can be accessed remotely. Complete Unix file semantics are provided, including locking and consistent access.

V[10] is a testbed for distributed-system research. Its four logical parts are a distributed Unix kernel, service modules, runtime support libraries, and added user-level commands. V manages a cluster of workstations and servers, providing the resources and information-sharing facilities of a conventional single machine.

At the beginning of the Athena project, these and other distributed operating systems were reviewed. None were found to be suitable for Athena, often because they were in the early stages of research.

System comparisons. Athena differs from the distributed systems just described in various ways. The following are some of the more important differences and similarities.

Amoeba supports processor server back ends supporting workstation front ends over a network using a capabilities-based operating system. The operating system supports distributed processing by network processors assigned to tasks dynamically as the processing load requires. In contrast, Athena performs all processing on the local workstation as a default. Remote execution can be initiated manually through the X Window System.

The Athena and Andrew projects are similar in scope and objectives, but they approached the problem with different priorities. The Andrew designers felt that the issue of a distributed file system was so important that they needed to develop their own (AFS). They also developed a number of end-user application packages, such as compound document editors. Athena's developers also thought that the distributed-file problem was very important and, therefore, that industry would solve the problem—thus the use of Sun Microsystems' Network File System. Project Athena concentrated on distributed services and did little with end-user services. The Andrew and Athena implementations complement each other quite well: Andrew uses the Athena windowing system (X) and authentication service (Kerberos), while Athena uses the Andrew compound document editor (EZ) and instructional authoring environment (cT).

Eden is object oriented and allows processes to migrate freely among processors; Athena uses conventional subroutines and does not support process migration.

Athena and Grapevine have relatively similar architectures. Both build on existing operating systems, and both provide distributed services of mail, naming, and authentication on dedicated servers. One difference is that Grapevine provides access control, while Athena uses Unix access control. Athena supports the additional distributed services of file access, notification, printing, and service management.

Like HCS, Athena supports a systemwide name space and provides distributed services. In contrast, Athena uses a single operating system programming and user interface.

The Athena designers studied the Locus architecture carefully early in the project. They used a number of its design concepts, and the fact that it operated on 17 mainframes interconnected by a local area network established confidence that the approach was viable. In several areas Athena and Locus are very similar: Locus provides a high degree of network transparency. Files and programs can be moved dynamically with no effect on correct operation or naming. Local and remote resources are accessed uniformly. Locus also provides reliable operation. Reliable facilities include replicated file storage and automatic recovery from various subsystem and network failures. A major difference between the two systems is that the Locus distributed file system puts all files in a single tree, while Athena has about 10,000 file roots, for flexibility and reliability.

Athena's designers studied the Locus architecture and used a number of its design concepts.

Whereas Mach is the kernel of a distributed operating system on which a variety of programming interfaces can be built, Athena's designers took the Berkeley programming and user interface as a standard and developed a relatively clean layer of distributed services on it.

The V system provides a software backplane that supports network-transparent address spaces, lightweight processes, and interprocess communication. Athena supports none of these, using standard Unix instead. Both systems support distributed file servers and print servers.

Issues in distributed systems

Several issues arise immediately in the design of a distributed system. Among the most important are naming, scalability, security, and compatibility.

Naming. A name service converts a logical name into a physical address by an algorithm, a table lookup, or a combination of the two. The purpose of a name service is to decouple the logical support of a function (such as printing) from the physical implementation of that function. This makes changing the configuration of a system far easier. It is only necessary to change the logical-to-physical mapping, not all the code that references the function.

System objects that can be named include hosts, printers, services, files, and users.

Factors affecting the design of name servers include the number of object types, number of objects, frequency of queries, and update frequency.

The simplest implementation of a name server is by means of a centralized service. If that is inappropriate for reasons of network bandwidth, distance, reliability, or load, the service can be distributed. If the implementation is by means of a database, the usual methods of data distribution can be used, including replication and partitioning.

A replicated database is created in multiple copies. Replication has the advantage of providing good performance and reliability, but update is difficult because of synchronization requirements. Replication also uses considerable storage.

In partitioning, the database is split into multiple, disjoint databases. Some means must be provided to identify the correct partition, the one holding the data of interest. This can be done through a directory or a broadcast search. The advantage of a partitioned database is that only one copy of the data exists, thus saving storage and simplifying update. One drawback is that the correct partition must be found for the access, and in some cases multiple partitions must be accessed to resolve the name. Another drawback is that a partitioned database is less reliable than a replicated database.

Most of the systems mentioned here use a combination of replication and partitioning to achieve a balance between performance and storage requirements. Most name servers also use caching of frequently used names.

Scalability. Design approaches that work well in a distributed system with a few nodes may not be usable in one with many nodes. For example, broadcast protocols work well for small numbers of nodes but do not scale well to large numbers. More generally, any design approach in which a scarce resource such as storage or manpower scales linearly with the number of nodes is likely to be too costly to implement. Thus, techniques such as putting complete system configuration lists on every workstation (storage intensive) or making each workstation's software configuration different (labor intensive) are not usable for even moderately large configurations (on the order of 100 workstations).

Security. Two issues that must be considered in security design for distributed systems are authentication and authorization. In a system of more than a few dozen workstations, the root password cannot be different on each workstation because the support task quickly becomes unmanageable. Therefore, Athena has made all root passwords the same and known to all users.

For user passwords, Unix systems store encrypted passwords for all users on each workstation. Maintaining user passwords for 100 or more workstations is very difficult, and providing complete password and group files for each machine in a large network is highly impractical. This (replicated database) approach to authentication and authorization does not scale well and suffers from the usual problems of replicated databases. A natural alternative is to provide a centralized password-checking service. To access remote services, passwords must be sent over the net. This method makes the system vulnerable to eavesdroppers, who can steal passwords. It also makes the system vulnerable to availablity failures.

A solution to the problems associated with sending passwords across the net is to encrypt the passwords. Two encryption-based authentication approaches have been developed: public-key cryptography and key distribution servers.

With public-key cryptography, encryption keys come in pairs. Each key in the pair is used to decrypt messages encrypted in the other key. Therefore, one key of the pair can be given to the servers and the other (secret) key can be given to the user, who uses the secret key to become authenticated.

A key distribution server assigns secure keys to be used between a user and a server. This approach scales well but requires a third party. The method generally used employs private keys and a trusted third party.

Authorization can be handled in many ways. One way is to delegate the function to the servers and implement it by access control lists showing membership in groups.[5] Another way is to use capabilities implemented as a bit pattern passed to the server, showing that the user is authorized to perform a particular operation.[11] Capabilities are usually encrypted to prevent forgery.

Authentication and authorization are two issues that must be considered in security design for distributed systems.

Compatibility. Various levels of system compatibility are possible, including the binary, execution, and protocol levels.

In binary compatibility all processors execute the same binary instruction repertoire and are compatible at the binary level, with differences only in performance and input/output. The Emerald distributed operating system[12] and VAX systems exhibit this level of compatibility. Although it greatly simplifies system development, binary compatibility severely restricts the system source and architecture and is not often used in large systems for this reason.

It is more common to provide system compatibility at the next level up: the execution abstraction (called coherence in Athena). A common execution-level abstraction exists if the same source code can be compiled and executed properly on the two systems. Andrew and Athena both support the Unix execution abstraction for the hardware platforms (VAXstation and RT for Athena).

The least restrictive form of compatibility achieves interoperability by requiring all system components to support a common set of protocols. The X Window System in Athena is an example. Protocol-level compatibility allows a distributed system to be based on common protocols for essential system services such as file access, naming, and authentication. Because Athena supports this level of abstraction, the execution abstraction could be given up and diverse operating systems would still be interoperable.

Athena does not address the issue of application data coherence except through the X Window System, which enables each type of workstation to access its own type of data representation (for example, byte order) separately.

Athena distributed-system model

The distributed-system model for Athena has three major components:

- workstations,
- network, and
- servers.

The approach taken by the Athena developers was to implement a set of network services to replace equivalent time-sharing services, in essence converting the time-sharing Unix model into a distributed operating system.

The network is invisible to the user. All services appear to be local and are available to the user with only a single submission of a password at login time. The actual delivery of the services is physically distributed over the system and communicated to the user transparently over the network.

In concept, any operating system can be used on components (workstations or servers) in Athena. To achieve interoperability with the other system components (and to gain access to the distributed services), the components need only support the Athena protocols for authentication service, name services, file access, and print service.

To minimize development cost, the protocols have been implemented only in Berkeley Unix on all hardware platforms. A side benefit of supporting only Berkeley Unix is the minimization of training and support cost. The computational model seen by the users, therefore, is Berkeley Unix.

Scalability. The design approaches used in Athena have improved its scalability by minimizing demands on scarce resources in three areas: network bandwidth, mass storage, and labor.

The campus network was designed with a backbone using optical fiber and a token ring protocol running at 10 megabits per second. Network routers are attached to the backbone, with each router supporting one Ethernet configured as an Internet Protocol subnet. Because of the routers, traffic local to a subnet does not load the backbone. This approach gives a good measure of growth capability because as more workstations are added, more Ethernet subnets can be added to the backbone.

Athena workstations are dataless nodes; that is, all workstations have local hard disks, which reduce paging traffic on the net. System software file service is replicated on all subnets, so communications traffic for the loading of system software remains local to a subnet.

To minimize mass storage, files that would normally be replicated on a per-workstation basis are centralized. These include password files and most configuration files.

To minimize labor requirements, all workstations are anonymous and interchangeable. They all have the same root password and the same software configuration. The only unique items are the workstation name and net address. The initial loading of software is done over the net.

Reliability. To improve reliability, no systems other than network routers are attached to the backbone. This limits the number and types of problems that can bring the entire net down. All other systems are attached to the Ethernet subnets. As a general rule, a hardware failure may prevent the updating of master files for, say, adding new user accounts or changing the hardware configuration. However, system operation can still continue. Subsystems whose failure would stop system operation, such as authentication service, name service, or access to system software, are replicated with automatic cutover.

Public workstations. Clusters of workstations are provided at about 40 sites around campus; students do not have to walk more than about 10 minutes from any location to get to a workstation. In addition to the clusters for student use, several clusters are dedicated to the development of departmental instructional software. Any user can log in to any workstation and gain immediate access to private files and environments.

Because public workstations have no retained state between sessions, the operations staff can reload the system at any time with no adverse consequences. By default the workstations refuse remote services such as login, to eliminate the impact of response time on the workstation user by others who might log in. Private workstations, generally used by faculty and staff, retain state between sessions.

Security. The Athena designers assumed that workstation security cannot be maintained. To obtain system security with insecure workstations after login, they developed an authentication service called Kerberos.

The Kerberos server is a trusted-third-party, private-key, network authentication system that validates the identity of individuals to network servers. It is based on the model developed by Needham and Schroeder.[13] Time stamps have been added to that model to assist in the detection and prevention of replay. Kerberos operates without imposing any additional burden on the user as a part of the normal login process.

To minimize labor requirements, all workstations are anonymous and interchangeable.

Kerberos establishes authentication by securely making and distributing a session key for each server-client instance. The particular server then has the responsibility to use the key in a manner consistent with its own authentication policy. The server may elect to (1) not use authentication, (2) use authentication on the first request only, (3) use authentication on all transactions, or (4) authenticate and encrypt all transmissions.

Heterogeneity. Since DEC and IBM, the two major sponsors of the Athena project, have incompatible workstation products, the system must support at least two incompatible workstation platforms. In addition, multiple generations of hardware from each vendor are not always compatible. The following minimum workstation requirements were imposed to bound heterogeneity and thereby bound the range of hardware for which instructional courseware must be developed:

- Ethernet interface,
- one million instructions per second processing speed,
- one million pixel display (monochrome),
- local hard disk of at least 40 megabytes,
- four megabytes of main memory, and
- pointer device (mouse).

At this time Athena does not support personal computers and other 300,000-pixel devices. However, some level of support for such devices is planned for the future, as described in the section on future plans.

Interoperability and coherency. Athena supports both interoperability and coherency. As with the X Window System, the subsystems in Athena interoperate by using a number of common network protocols. Unix and C provide a high level of coherence so that the same application source code, and most of the system source code, can be compiled and executed on all workstations regardless of architecture.

Coherence in Athena includes standardization of three lower-level interfaces:

- network service—the system interface between the applications and the operating system;
- the applications programming interface—the interface between applications and the system;
- the user interface—the interface presented by applications to users.

To obtain system coherence, the same operating system and the same communications protocols were used on both hardware platforms. Berkeley Unix was selected for its functional power and multitasking capability. It provides the same virtual machine interface to applications on all workstations. From 85 to 90 percent of the source code for the operating system is common to the two workstation types. Essentially all the source code is common for applications. Separate binaries (compiled from the same sources) are supported for the two workstation types. For communications coherence, TCP/IP was selected to implement the same virtual network interface to all applications.

Applications coherence is a much more difficult problem. It has been handled in part by supporting a small number of languages on Athena. Presently supported languages are C, Lisp, and Fortran.

Another part of the problem has been solved by a standard windowing system, the X Window System, developed by the MIT Laboratory for Computer Science and by Project Athena. X provides a network-transparent, device-independent, vendor-neutral windowing environment.

The user interfaces for the two types of workstations are as identical as the keyboards and mouses will allow. X was developed to provide applications coherence to the human interface. However, X solved only part of this difficult problem. Recently the Motif human interface development environment from Open Software Foundation has been installed to provide a more powerful level of abstraction in obtaining user interface coherence.

In a similar manner, standard applica-

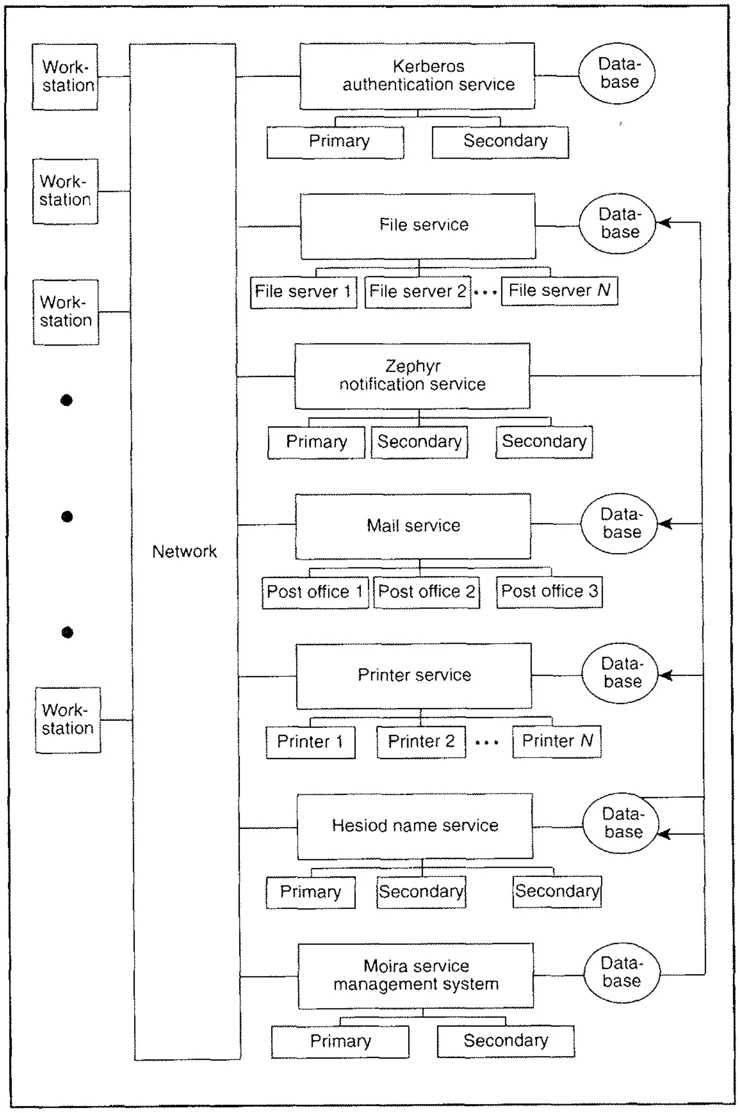

Figure 1. Athena system model.

tions including a 2D drawing package (GKS), a text editor (Gnu EMACS), a spreadsheet, a laboratory data management system, and a document formatter have been adopted.

Since the two different workstation architectures support two different floating-point and byte orders, the system must handle these differences in a method transparent to users. Each workstation knows its own type (VAX or IBM RT). Separate sets of binaries are stored for each system type. Text file formats are the same for both hardware platforms. When binary data are accessed, each workstation type accesses its own type based on data from the name server.

Affordability. To achieve affordability, the designers decided to use centralized management and support of the system even though the hardware was distributed. Experience with Athena has indicated that centralized management and support provide significant benefits in reduced support cost and improved quality of services compared to distributed management and support. (It is true, of course, that distributed management can solve political and organizational problems that the centralized approach cannot solve as easily).

Athena system design

The Athena system design views the entire hardware complex of (up to 10,000) workstations, file servers, communications servers, and printers as a single, unified distributed system (as described earlier). The system is diagrammed in Figure 1. It is network oriented; workstations attached to the network are identical in software configuration. They communicate with a variety of network servers, which provide services to users. Because all services provided are network services, they are available to all users without regard to location. They include

- name service,
- file service,
- printing,
- mail,
- real-time notification,
- service management,
- authentication, and
- installation and update.

Files that would normally be provided at every workstation are centralized, with a small number of copies to provide fail-soft capability and improved performance.

Service management. The configuration management of large numbers of workstations and servers is very labor intensive and cannot be done effectively by manual means. Therefore, Project Athena has developed a service management system called Moira to automate much of the routine management task. Moira includes a central database of system control and configuration information, tools for manipulating that information, and tools for updating the services to conform to the database. Examples of types of information stored in Moira include disk quotas, all hardware configuration files, allocation of individuals to post offices, and access control lists.

The use of a centralized database has proved both effective and economical because each piece of data is stored only once, is easily updated, and is subject to fine-grained access control.

Name service. In time-sharing systems names are resolved (typically) into addresses by means of configuration files. These files, which reside in each node, tend to be relatively static. This static approach is not appropriate for meeting system needs for dynamic reconfiguration in workstation networks. Also, it wastes storage if replicated on all workstations.

The purpose of the Athena name server, called Hesiod, is to allow centralized, dynamic linkage between names and objects. For example, Postscript printing for the third floor of building E40 might initially be provided by a printer named Nil at address "ps.mit.edu." If, at some later time, this service were moved to a different printer at address "lqp.mit.edu," it would not be feasible to update a configuration file on each of 10,000 workstations to reflect the change. Instead, the name server provides a single, centralized location for the configuration file for all workstations, thus providing very significant savings in storage and making changes in system configuration manageable with minimal labor.

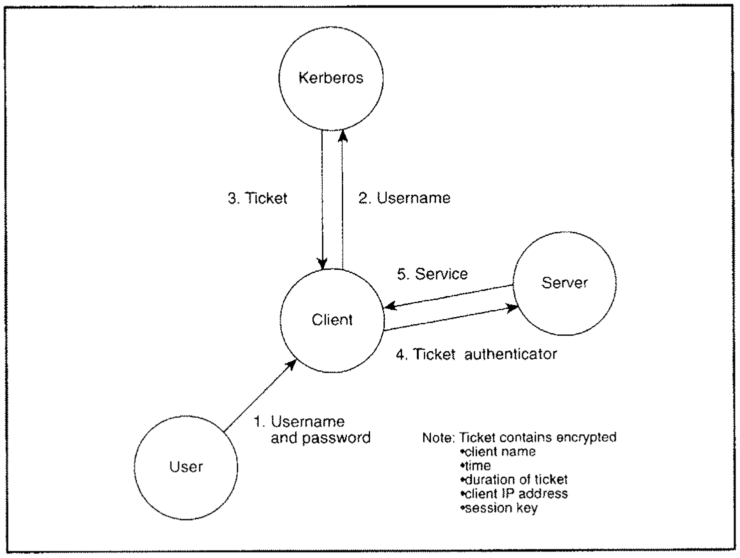

Figure 2. Kerberos information flow.

Hesiod includes the Berkeley *bind* name server with some extensions. Hesiod provides information on users, location of user private files, mail delivery addresses, and locations of network services such as authentication and printing.

Hesiod is updated by the central system database (maintained by the Moira service management system) every few hours and in effect represents a fast front end to the Moira database.

Authentication service. System software (except for the kernel) is not kept on the workstations; they function as dataless nodes. When a user logs in, new system software is down-loaded from a secure network file server, thus assuring a validated initial operating system.

When the user logs out, a program called *toehold* deactivates the workstation. All attached file systems are detached. All temporary storage areas are cleaned out and the window system is terminated. No network connections (for example, to external file servers) are retained. Several other housekeeping activities are performed, such as the destruction of any tokens of authenticity created for the previous user. Finally, the next login is solicited by a "Press any key" prompt. If a new system library is loaded onto a server while the workstation is deactivated, it will be used at the next login; this deactivation-activation cycle provides an en masse software update of the nonresident portion of the system.

When the key depression is received, *toehold* executes a shell script that attaches the system libraries from RVD (described in the next section) and starts the X Window System. A login window then is activated for the user and login is solicited.

Login proceeds with the user submitting username and password. The user is authenticated transparently by the Kerberos authentication server, which obtains information about the user from the Hesiod name server. The user home directory is then attached from the appropriate file server. The Zephyr real-time notification service (described later) is activated and a Zephyr windowgram client is started. Login continues with the usual execution of the login files.

Subsequently, Kerberos authenticates the identity of the user to the desired service in a fully transparent manner through the use of encrypted "electronic credentials." The general information flow in Kerberos is shown in Figure 2. Kerberos only establishes identity (authentication) and does not become involved in the decision of whether an individual is allowed to use a specific service (authorization). The normal Unix mechanisms are used for authorization.

Two types of credentials are generated by Kerberos: tickets and authenticators. A *ticket* is used to securely transmit the identity of an individual to a server and to securely exchange a temporary session key. The *authenticator* contains additional information that a server can compare to the ticket to prove that the user presenting the ticket is the same one to whom the ticket was issued.

File service. Athena workstations are dataless nodes supported by extensive file service. Athena has three different file services and in principle supports an arbitrary number of other file services. One service, Remote Virtual Disk (RVD), holds read-only files such as system software libraries and some applications software. Another, Sun NFS, holds read/write files. The third, AFS, also holds read/write files but is especially useful for files used for cooperative work.

On user request additional file systems can be mounted via the *attach* command. File systems are named objects. Hesiod resolves the names to location, type, and mount point. User customization of the name space is an important capability.

RVD. Workstation software includes a considerable amount of library software. Storing this software on each individual workstation might improve performance but would lead to excessive cost and would greatly complicate the software update and

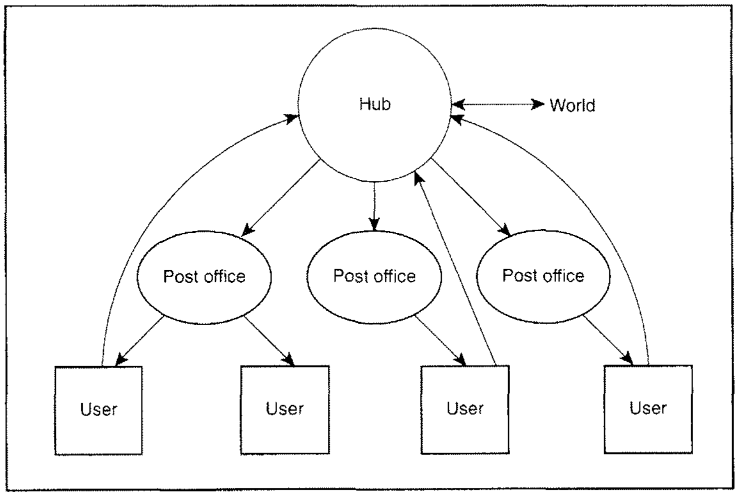

Figure 3. Athena mail.

software integrity problems. Providing the library software in its entirety would require large local disk resources. Outboard file service, however, must be able to support high enough client-to-server ratios that a high degree of redundancy is practical.

To solve these problems, Athena uses RVD[14] to store system software libraries. RVD was originally developed at the MIT Laboratory for Computer Science and subsequently enhanced in Project Athena. RVD supports logical "packs" on network server machines that appear to a workstation to be the same as local physical disk packs. All file system information is maintained by the workstation, and the RVD servers simply deliver requested disk blocks. An RVD pack may be used by many workstations in a read-only mode or by a single workstation in an exclusive read/write mode. Because of the efficiency of the block-level service, an RVD server can support about 10 times as many users as a generalized file server. Athena provides this system library service redundantly on a per-subnet basis.

NFS. An Athena design requirement is location independence, so that any user can use any workstation. NFS provides the capability for private user files to be available at any workstation. Private files are allocated as "lockers." Lockers are stored on network file servers distributed across the system, but they appear to the user in a single, local file directory. The files must be explicitly attached to the workstation file hierarchy, but this is done transparently by *.login* for home directories. Normal NFS security and privacy facilities are provided.

Printing. Conventional time-sharing systems provide local printing. The use of this model in a workstation environment creates two problems: print queuing and printer configuration.

Unix normally queues the print file on the workstation, with no indication of whether the requested printer is accepting jobs. If the user logs off, the next user may trash the spooled file before it gets printed. The possibility also exists that the spooled file will never get printed because the printer is not accepting jobs. Finally, sending a print request to a particular named printer is an arbitrary limitation on the manner in which the print request is eventually serviced.

In Athena the *lpr* Berkeley print spooler has been modified to queue the print file at the remote print server. If the print server is not available, an immediate error message is generated.

A second problem is the configuration problem, which is similar to the password problem. The *printcap* file would normally be stored on every workstation in a static form. Athena has moved the *printcap* function to the Hesiod name server, permitting configuration control to be separated (logically) from the print service location.

Electronic mail. In a time-sharing system, users send mail to remote individuals by using a mail address of "username@absolute-path-name" or to local individuals by using an address of simply "username" with the remainder of the address as a default. An Athena requirement is that any user can use any workstation. Therefore, absolute path names are not appropriate because individuals are not limited to one system.

The approach developed for Athena, shown in Figure 3, combines the best of local and remote approaches. All individuals are addressed as though they were local, without using a machine name. The mail is sent to a central mail hub and then to a "post office," where the mail is queued until the addressee decides to pick it up. The addressee picks up the mail by transferring it from the addressee's post office (based on Hesiod information) to the addressee's private (NFS) mail file (also based on Hesiod information). The addressee then can read, write, and edit mail from any workstation just as if everything were actually local.

To accomplish this, Athena uses the Mail Handler (MH) mail system. An improved graphical front end (called XMH) based on the X Toolkit has been added to MH so that all commands are entered through mouse selection of active objects instead of the command line interface. The Post Office Protocol utilized by MH has been retained and modified to require the use of the Kerberos authentication server. Multiple post office servers are provided as required for load management and reliability. MH uses information in the Hesiod name server to select the proper post office for each individual. All changes to MH to support a distributed system are transparent to the user, and the user uses the software as if it were on a single, centralized time-sharing system.

Users outside of Athena can also send mail to individuals within Athena as if it were a single, centralized system. They send mail to the address "username@athena.mit.edu." This mail goes to the mail hub and is distributed by the same mechanism as internal mail.

The mail hub also handles distribution lists within Athena. At this time there are 400 mailing lists, and a total of 9,000 mail messages per day cross the hub. These 9,000 messages are handled by three post office servers supporting the 10,000 users.

Real-time notification. There are occasions when real-time notification is useful—for example, when new mail has arrived or when a file server is going to be taken out of service temporarily. Real-time notification is easily accomplished on a centralized time-sharing system because the identities of all individuals logged in are available in a single location. In a distributed system it is much more difficult because there is no central file holding this information, nor can the workstation's report of this information be naively trusted.

To meet this need, the Athena project has developed a real-time notification service called Zephyr. With this system a user can send a message immediately by using a single command, without knowing the location of the intended recipient or even if the recipient is logged in. Zephyr searches the active workstations to locate the recipient, and if the recipient is logged in, immediately creates a popup window with the message. The recipient can select classes of Zephyr messages of interest and suppress the rest. This capability operates within the constraints of privacy and permission.

On-line consulting. An on-line consulting system, *olc*, which permits users to direct questions to consultants or other expert users logged in elsewhere on the network, has been developed for Athena. A set of stock answers is maintained for quick response to common questions. The student can access the answers through a menu system, and the consultants can return a standard answer electronically if appropriate. If the consultant is not available, questions are automatically queued. If a question is difficult or remains unanswered when the user logs out, the answer may be returned by electronic mail.

Discuss. *Discuss* is an electronic conferencing system that allows Athena users to communicate easily and effectively on specific subjects. At present about 100 concurrent electronic meetings are in progress. The *discuss* program is authenticated by Kerberos and uses Zephyr to send real-time new-transaction announcements. The system evolved from an earlier one called *forum*, supported on Multics.

System software development

To support coherence at the application programming interface, most of the source code for Athena's operating environments is the same on the two different workstation platforms. The source code is partitioned into a machine-independent section (the applications and most of the operating system) and a machine-dependent section to simplify the system-building process.

The basic operating system software used on all Athena workstations and servers at this time is BSD 4.3, with machine-dependent software (such as device drivers) supplied by the manufacturers. The standard Berkeley distribution is augmented by several third-party software packages and local Athena modifications and additions.

To maintain control of the software configuration, all software is completely built from source code about once a year. Modules are updated three or four times a year.

The new source code for the Athena distributed services is about 20 Mbytes. The total amount of source code for the Athena system, including Unix, the Athena distributed services, and applications, amounts to 400 Mbytes. The source code is structured into a hierarchy of about 1,300 makefiles and generates 12,000 binary modules.

To control the labor content and quality of the development, management, and support of this large amount of source code, the Athena project developed or imported a number of software tools, including

- *prot_sources*—to manage access privileges;
- *track* (like *rdist*)—to manage distributed files;
- *maksrv*—a script set to convert workstation software to servers.

Prot_sources gives a measure of protection against aborts of system builds due to lack of access privileges. It allows software developers to start a build at any time without using *root* access and its lack of protection. *Prot_sources* provides four categories of protection: writable, executable, locked, and unlocked. It traverses the 400 Mbytes of source code in about 1.5 hours, reporting on its actions as they occur.

Track creates a local file containing the update status of all the distributed files used for the system build. After the local file is complete, the system build process can compare local copies to the update status contained in *track* instead of comparing against remote files, thus saving thousands of remote file accesses.

Ideally, all software update should be entirely automatic and should propagate to all workstations essentially simultaneously. Tools to do this in Athena are available for nonkernel code but not for kernel code. Presently, changing the kernel requires that each individual workstation be visited and the kernel loaded from a floppy disk (the "roller skate" approach). Work on automating the update of workstation kernel software is proceeding.

The *makesrv* tool builds server systems (such as printers and name servers) as modifications of a standard workstation system. The workstation software is initially installed on servers. Building a server from a workstation then becomes a standard operation. *Makesrv* can build the server types of NFS, Hesiod, Kerberos, Moira, *olc*, RVD, Zephyr, *discuss*, and Printer (the server portion of *lpr*).

Future plans

Because of the significant value of Project Athena to its two industrial sponsors, funding has been committed through June 1991. The following new initiatives are planned for the distributed system during the remainder of the project:

- *Move to private ownership.* To date, all Athena workstations are owned by MIT and are made available to students and faculty at no cost. When the price of a workstation meeting Athena standards drops to an appropriate level (currently seen as \$3,000), MIT may sell workstations to students and faculty through the Microcomputer Store. Eventually the number of workstations supported (primarily privately owned) will approach 10,000. The network, servers, and infrastructure must scale transparently in accordance with the number of workstations.
- *Shift to supported software.* The strategy of Athena is to move away from Berkeley Unix to a vendor-supported Unix. Since both major sponsors belong to the Open Software Foundation, the OSF/1 operating system looks attractive. (Project Athena also belongs to the Open Software Foundation and has a research grant relationship with it).
- *Automate software update and integrity checking.* Installation of a new version of the system software is a major undertaking because it requires a visit to every workstation. Tools to install a new version of the system software over the network and to verify its integrity after installation are now being developed.
- *Provide dynamic network reconfiguration.* One of the most common problems in Athena workstation installation is the installation of an incorrect Internet address.

Experiments are underway to assign this address at boot time. In addition, when Athena begins to allow privately owned workstations, it must be possible for a user to unplug from one network connection and reconnect at a new location without notifying the network administration. Work on dynamic reconfiguration is also in process.

• *Support personal computers.* The widespread availability of personal computers with screen resolutions of 300,000 pixels makes them attractive. The degree of support and methods of support are now being investigated.

• *Deploy to other universities.* Part of the original vision of Project Athena was that the system be available at no cost to other research labs and universities once it is fully operational. In late 1988 part of the system was installed at DEC's Cambridge Research Laboratory. In early 1989 the entire distributed system was installed at Bond University, a new, private university in Australia. Subsequently, parts of the system have been installed and are in use at Iowa State University, the University of Massachusetts at Amherst, New York University, the University of Texas at Austin, North Carolina State University, and the University of California at Santa Cruz. Other installations are now being considered. ■

Acknowledgments

The design of the Athena system was led by Jerry Saltzer of MIT in association with the Athena staff, which includes among others Ed Balkovich, Tony Della Fera, Steve Dyer, Dan Geer, Jim Gettys, John Kohl, Steve Lerman, Steve Miller, Cliff Neuman, Ken Raeburn, Mark Rosenstein, Jeff Schiller, Bill Sommerfeld, Ralph Swick, and Win Treese.

References

1. A.S. Tanenbaum, "Distributed Operating System," *Computing Surveys*, Dec. 1985, pp. 419-470.

2. J. Morris, "Make or Take Decisions in Andrew," *Usenix Conf. Proc.*, Usenix, Sunset Beach, Calif., Winter 1988, pp. 1-8.

3. D. Anderson et al., "The Dash Project: Issues in the Design of a Very Large Distributed System," Technical Report 87/338, Computer Science Division, Dept. of Electrical Engineering and Computer Science, Univ. of California, Berkeley, Jan. 1987.

4. A. Black, "Supporting Distributed Applications: Experience with Eden," *Proc. 10th ACM Symp. Operating Systems Principles*, Dec. 1985, pp. 181-193.

5. A.D. Birrell et al., "Grapevine: An Exercise in Distributed Computing," *Comm. ACM*, Apr. 1982, pp. 260-274.

6. D. Notkin et al., "Interconnecting Heterogeneous Computer Systems," *Comm. ACM*, Mar. 1989, pp. 258-273.

7. B. Walker et al., "The Locus Distributed Operating System," *Proc. Ninth ACM Symp. Operating Systems Principles*, Oct. 1983, pp. 49-70.

8. R.F. Rashid et al., "Mach: A New Kernel Foundation for Unix Development," *Usenix Conf. Proc.*, Usenix, Sunset Beach, Calif., Summer 1986, pp. 93-112.

9. J.K. Ousterhout et al., "The Sprite Network Operating System," *Computer*, Vol. 21, No. 2, Feb. 1988, pp. 23-36.

10. David Cheriton, "The V Distributed System," *Comm. ACM*, Vol. 31, No. 3, Mar. 1988, pp. 314-333.

11. S.J. Mullender, "Amoeba as a Workstation Operating System," *Proc. IEEE Workshop on Workstation Operating Systems*, Cambridge, Mass., Nov. 1987.

12. E. Jul et al., "Fine-Grained Mobility in the Emerald System," *ACM Trans. Computer Systems*, Feb. 1988, pp. 109-133.

13. Roger Needham and Michael Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Comm. ACM*, Vol. 21, No. 12, Dec. 1978, pp. 993-999.

14. M. Greenwald and J. Sciver, "Remote Virtual Disk Protocol Specification," MIT, Cambridge, Mass., 1986.

Further reading

Athena is a very large project and cannot be fully covered in this article, which is limited to the distributed aspects of the system. The following bibliography lists sources of information on other aspects of the system, as well as further reading on its distributed aspects.

Cohen, K.C., "Project Athena—Year 5 Student Survey Findings, 1988," MIT Report, June 1988, 33 pp. [pedagogical aspects of the project]

Coppeto, T., et al., "OLC: An On-line Consulting System for Unix," *Usenix Conf. Proc.*, Usenix, Sunset Beach, Calif., Summer 1989, pp. 83-94.

Davis, Don, "Project Athena's Release Engineering Tricks," *Proc. Usenix Workshop on Software Management*, Usenix, Sunset Beach, Calif., Apr. 1989, pp. 101-106. [specifics of the system software development process]

Della Fera, T., et al., "The Zephyr Notification System," *Usenix Conf. Proc.*, Usenix, Sunset Beach, Calif., Winter 1988, pp. 213-220.

Dyer, S.P., "Hesiod," *Usenix Conf. Proc.*, Usenix, Sunset Beach, Calif., Winter 1988, pp. 183-190.

Hodges, M., R. Sasnet, and M. Ackerman, "A Construction Set for Multimedia Applications," *IEEE Software*, Jan. 1989, pp. 37-42. [multimedia workstation and Muse interactive video authoring environment]

Raeburn, K., et al., "Discuss: An Electronic Conferencing System for a Distributed Computing Environment," *Usenix Conf. Proc.*, Usenix, Sunset Beach, Calif., Winter 1989, pp. 331-342.

Rosenstein, M., D. Geer, and P. Levine, "The Athena Service Management System," *Usenix Conf. Proc.*, Usenix, Sunset Beach, Calif., Winter 1988, pp. 203-212. [Moira system]

Scheifler, R., and J. Gettys, "The X Window System," *ACM Trans. Graphics*, Apr. 1987, pp. 79-109.

Steiner, J.G., B.C. Neuman, and J.I. Schiller, "Kerberos: An Authentication Service for Open Network Systems," *Usenix Conf. Proc.*, Usenix, Sunset Beach, Calif., Winter 1988, pp. 191-202.

Swick, Ralph, "The X Toolkit: More Bricks for Building Widgets," *Usenix Conf. Proc.*, Usenix, Sunset Beach, Calif., Winter 1988, pp. 221-228.

Treese, G.W., "Berkeley Unix on 1000 Workstations: Athena Changes to 4.3 BSD," *Usenix Conf. Proc.*, Usenix, Sunset Beach, Calif., Winter 1988, pp. 175-182.

George A. Champine is associate director at Project Athena for Digital Equipment Corporation, responsible for directing and managing DEC's participation in the project. His prior assignment was with the Advanced Computer Architecture project for DEC at MCC in Austin, Texas, with special emphasis on system architecture and the human interface definition. Earlier he was technical director of workstation development for DEC.

Champine received his BS in physics in 1956, his MS in physics in 1959, and his PhD in information systems in 1975, all at the University of Minnesota.

Daniel E. Geer, Jr., is the manager of systems development for Project Athena. He is responsible for all technical infrastructure, including X, Kerberos, Hesiod, Zephyr, Moira, and all other aspects of the network services system. Previously he was manager of systems for the Health Sciences Computing Facility, School of Public Health, Harvard University.

Geer holds a BS degree in electrical engineering and computer science from MIT, and a Doctor of Science degree in biostatistics and epidemiology from Harvard University. A member of several professional societies, he was recently on the program committee for the Washington, DC, Usenix conference.

William N. Ruh is a staff programmer with IBM in Cambridge, Massachusetts. He is currently working on highly parallel fault-tolerant systems. His previous work includes Project Athena and automatic wiring algorithms at IBM in East Fishkill, New York.

Ruh graduated summa cum laude from Vanderbilt University in 1983 with bachelor's degrees in electrical engineering and mathematics.

Readers may write the authors c/o George Champine, Digital Equipment Corp., 146 Main St., Maynard, MA 01754-2571.