Network Services in the Athena Environment

Jennifer G. Steiner
Daniel E. Geer, Jr.

Project Athena
Massachusetts Institute of Technology

ABSTRACT

As a prerequisite for achieving its mission of fostering educational innovation at M.I.T., Project Athena must support a large network of independently owned and controlled workstations. At the Project's inception, systems software to support such a configuration did not exist. As a result, a large part of the systems development effort during Athena's first five years has been devoted to the design and implementation of network services to fill this need. This paper describes the use of network services in the Athena environment, including three new systems level services developed at Athena: the Kerberos authentication service, the Service Management System, and the Hesiod name service.

Introduction

Project Athena[1] is a joint effort between M.I.T. and a number of external sponsors, principally Digital Equipment Corporation and International Business Machines. Funded originally for five years and recently extended for an additional three, Project Athena is an experiment in computer-aided education; its charter is to create a computing environment that fosters educational innovation. Specific goals are software coherence in a heterogeneous context, unlimited upward scalability of the service environment, and providing M.I.T. faculty with a powerful teaching tool. Athena's most complete product to date is its systems base – a distributed workstation system based on the network services model. This model furthers the goals of coherence and scalability, and provides an environment which facilitates the building of teaching tools.

In this paper, we give an overview of the use of network services at Athena. We begin with a description of the hardware and software environment at Project Athena. We then give the motivations for using a network services model in this environment. Next we describe three network service systems developed at Athena that perform functions required by the network services model itself: authentication, service management, and name service. We then describe how we have modified some more traditional services for use in our environment, including file, mail, and print services. Finally, we mention some newer services in use at Athena, which were developed at M.I.T., and discuss some services which are absent from the Athena environment.

1. The Athena Environment

The campus network, run by the networking group of M.I.T. Information Systems, is built on a 10Mb/s Proteon token ring hosting Level 3 Internet Protocol routers. Behind these gateways lie individual 10Mb/s Ethernet segments, each defined as an IP subnet. The M.I.T. network connects some 30 subnets, totalling about 1300 nodes, over a campus area of approximately 700 acres. Project Athena is the network's biggest customer.

The Athena plant currently consists of 850 hosts, totalling over 1000 MIPS of processing power and over 40 gigabytes of storage. Of these hosts, 450 are ‘‘public workstations’’ – centrally located workstations that are available to any member of the M.I.T. community. Another 100 machines are employed as dedicated servers. The remaining hosts are ‘‘private workstations,’’ assigned to or owned by particular individuals, such as faculty or Athena staff members. Approximately half of the workstations are DEC Vaxstation II's and 2000's; the other half are IBM RT/PCs. The average Athena workstation has a 1 MIPS processor, a 1000 by 1000 pixel graphics display, a three or four megabyte memory, and a 30-70 megabyte local disk. A variety of peripheral devices, notably laser printers at the ratio of one per eight workstations, complete the hardware environment.

Athena's software environment is based on 4.3BSD Unix, plus many imported and domestic components. Athena is a source code shop, building software from source for every distribution. Currently, the pool of source code is over 800 megabytes. Imported software is generally covered by site licenses, though often limited to educational use (as opposed to research or administrative).

2. Why Do Business Across the Network?

Network services are, as the term would suggest, a set of services delivered to the workstation user over the network, without specificity nor concern as to where the services are located. A network services model of computation is inherently consistent with Athena's goals. It promotes a coherent environment – one that delivers service to any network-capable node in the same manner – within a heterogeneous workstation plant. A network services model supports information and resource sharing. It provides a degree of connectivity that represents a significant if intangible benefit to its users. Careful design of the service environment allows an advantageous allocation of the computational burden, minimizing unnecessary demands on the server plant, and relying instead on the more abundant, less heavily loaded client workstations. As a form of distributed system, a network services model also enjoys the potential of higher reliability and availability, extensibility, and flexibility as compared to a set of wholly autonomous computers.

One interesting feature of the Athena plant is the public workstation – much like the public telephone in concept. The serially reusable public workstation represents the most consistent model of the combination of coherence, shared resources and information, and connectivity. Such workstations allow the workstation plant to be treated as a commodity market, increasing supply to meet demand without imposing arbitrary geographic or temporal constraints on users. Such fiscal advantages as standardization of the repair infrastructure, interchangeability, and interoperability are maximized. A commodity workstation, for example, permits a software maintenance strategy of simply reloading software at the first sign of trouble – an option unavailable in the face of idiosyncratic installations, each maintained by its own wizard. Scalability requires such strategies, and scalability is our principal metric of the software environment's maturity.

Finally, the network services model allows for central management of data and resources. If a network service manages a set of data, with client programs requesting information of the network servers, then when a change to the data needs to be made, it can be done in one central location – on the network server. The update will be handed to the clients on subsequent requests. In this way, it is not necessary to update data files on every individual workstation, an important consideration with respect to scale.

3. New Systems Level Network Services

Some of the network services in use at Athena have been developed because the network service environment itself required them. This section gives an overview of three of them: Kerberos, SMS, and Hesiod.

3.1. Kerberos

In the Athena environment, the user has complete control over the workstation. Since a user can boot the workstation in single user mode, or boot off removable media, the root password provides no useful barrier to privileged functions, and so we publish it. Workstation operating systems therefore cannot be trusted to accurately identify their users. Some other method of authentication was needed, and this motivated the design and implementation of the Kerberos authentication service.[2,3] In the case of Kerberos, a network service proves not only useful in the central management of data (user names and passwords) but also necessary as a trusted authority for authentication.

Kerberos is based on the Needham and Schroeder third-party authentication model,[4] using private-key encryption. Each user and network server has a key (like a password) known only to it and the Kerberos database. When a client program wishes to make an authenticated request to a network server, it first applies to Kerberos for a *ticket* for that server. Kerberos creates the ticket, encrypted in the server's key. The ticket contains the client's identity, along with a temporary encryption key, or *session key*. Kerberos encrypts both the ticket and a copy of the session key in the client's key, and returns them to the client.

The client decrypts the returned message, stores the ticket and session key away for future use, and passes the ticket on to the network server. The network server decrypts the ticket with its key and retrieves the session key and the identity of the client. At this point the server is certain of the identity of the client, and the client and server share the session key, which they can use for any appropriate purpose, *e.g.*, to encrypt subsequent communication.

The above is a simplified description of the Kerberos model. Kerberos also addresses issues which are not described here, such as authentication of the server to the user, re-use of tickets, prevention of ticket replay by imposters, expiration of tickets, and others. It should also be noted that the authentication mechanism is not visible to the user. Users choose their own passwords, and the only interaction required of a user is the password at the beginning of a login session, as in a timesharing system.

The implementation consists of a database, one or more authentication servers, an administration server, a Kerberos library, and an encryption library.

Database. Kerberos requires a simple database management system; Athena's implementation uses *ndbm*, which is distributed with 4.3BSD Unix. The database resides on the master Kerberos machine, which is physically secure and connected to an uninterruptable power supply.

Authentication Server. The authentication server is a continuously-running program which services authentication requests sent over the network by creating server tickets and session keys. It runs on the master Kerberos machine and accesses the database in read-only mode. In addition, there is *slave* authentication server, which also runs on a physically secure machine, and periodically obtains a read-only copy of the database from the master Kerberos machine. The slave server is a backup authentication server, in case the master authentication server is down or busy.

Administration Server. The Kerberos administration server accepts requests for write-mode access to the database, such as changing passwords and adding new entries. It runs on the master Kerberos machine and uses Kerberos authentication to verify its clients' identities.

Libraries. The Kerberos library implements the protocol used for sending requests to the Kerberos server. The encryption library implements routines used in encrypting and decrypting tickets. It is based on the Data Encryption Standard,[5] but could be replaced with another private key encryption scheme.

3.2. The Service Management System

In order to keep management cost down in the face of rising numbers of workstations and servers, we have whenever possible collected information in a central location rather than keeping any configuration information on workstations, or manually managing copies of server configuration files. In this way, when data must be modified, it can be done in one place rather than on hundreds of machines. Project Athena's Service Management System[6,7] (SMS) provides central management of a database of information on users, machines, lists, and network servers. This information in turn is used to automatically manage network services in the Athena environment.

SMS has six components: the database itself; the SMS server, which executes requests received over the network to manipulate the database; the SMS library, which implements the protocol used to communicate with the SMS server; client programs, which users run to obtain or update SMS information; the Data Control Manager (DCM), which extracts information from SMS for specific network servers and automatically updates them with it; and the Update Servers, which run on machines with servers managed by SMS and use the information passed over by the DCM to update those servers. SMS currently maintains and updates three name servers, 49 file servers of two different types, one mail hub, and three mail repositories.

Database. SMS requires an underlying relational database system. The current implementation uses RTI Ingres.[8] The database resides on a secure machine, and currently takes up about 13 megabytes of storage.

Server. The SMS server is a continuously-running program which accepts requests across the network and performs manipulations on the database according to those requests. It performs simple queries, plus any more complex queries which must be performed atomically. The server controls access to the database using fine-grained access control lists, one for each query and object in the database.

Library. The SMS library implements the protocol for communication with SMS and is used by programs updating or extracting SMS database information. It is based on a (homegrown) remote procedure call system,[9] which isolates the protocol from the underlying database system. This RPC is implemented on

TCP/IP streams. The library also performs validity checks on data sent in SMS database queries. It supports three kinds of requests: authentication, database query and update, and access checks.

Client Programs. Client SMS programs are run by users to effect changes in the SMS database. They send requests over the network to the SMS server. There are client programs for maintaining user information (including modified versions of *chfn* and *chsh*), groups, mail and other lists, post office boxes, file systems, workstation clusters, network servers, machine information, printer capabilities, port locations, and DCM control information. Information maintained about network services includes their location, obviating the need for this information on each workstation; and the information necessary for server configuration files, enabling service configuration to be managed centrally and then propagated to the individual servers.

Data Control Manager. The Data Control Manager (DCM) is a program run on the SMS machine periodically by *cron*. It looks in the database to see which servers need to be updated, converts the necessary information from the database into a format suitable for the server's consumption, and propagates it to the server's machine with the cooperation of an Update Server running on that machine.

Update Servers. There is one continuously-running Update Server on each machine hosting a network server maintained by SMS. The Update Server is contacted by the DCM when its server needs to be updated. Typically, the DCM authenticates itself (using Kerberos) to the Update Server, transfers the data files which drive the server, transfers a set of instructions for updating the server, sends a command to execute those instructions, and finally, verifies that the update procedure has been successfully completed. The status of the update is stored in the database for future reference.

3.3. Hesiod

The Service Management System is our central store of information, but the Hesiod name server[10, 11] provides a fast, highly available front end for retrieving that information. The Hesiod name servers receive data from SMS, and use those data to reply to name resolution requests posed over the network. Hesiod data are in the form of ASCII files, so Hesiod may be used without SMS, an occasional convenience.

Hesiod takes a request consisting of the name (HesiodName) of an object, and the type (HesiodNameType) of information desired about that object, and returns all the information it has associated with those two keys. For example, a request might consist of the HesiodName ‘‘geer’’ and the HesiodNameType ‘‘passwd’’; and Hesiod might return a line in the format of */etc/passwd* associated with user ‘‘geer’’. At Athena, *login* has been modified to make this request and to store the return value in */etc/passwd*.† In this way, programs looking for information in the password file can find it there, yet we need not maintain a ten thousand line *passwd* file on each of several hundred workstations. When a new user is added, only one change – to SMS – has to be made, rather than adding an entry to the */etc/passwd* files on every machine.

Hesiod is based on the Berkeley Internet Domain Server (BIND).[12, 13] Hesiod takes the given HesiodName and HesiodNameType, expands them into a BIND name according to well-defined rules (some locally configurable), and passes it on to BIND. It then returns the result of the BIND call to the Hesiod client. We have added a new class to BIND, the HS (for Hesiod) class, and a new query type, TXT (for text).

Servers. Hesiod servers are replicated, read-only servers, managed by SMS. They are fed about 3 megabytes of data, which are held in core, resulting in an in-core image of about 10 megabytes. Resolution requests take on the order of milliseconds. The servers receive SMS updates about once every four hours.

Library. The Hesiod library implements the routine for making a request to Hesiod. The routine takes the HesiodName and HesiodNameType as arguments, and returns an array of strings containing the information found. HesiodNameTypes are application specific. Athena has defined HesiodNameTypes for the following kinds of information: workstation service cluster, file system, post office box, user id, group, group list, group id, printer capability, server, and service location.

Clients. Examples of programs which use Hesiod include the following. *Attach:* given the name of a filesystem to mount, return the type (RVD, NFS, *etc.*), the host, the pathname exported from the host, the default local mount point on the workstation, and the default access mode for the given filesystem. *Login:*

† The actual password is not retrieved from Hesiod; password administration is done by Kerberos.

given a user's name, return the password file entry and group information. These are inserted into the */etc/passwd* and */etc/group* files. *Mail:* given the user's name, return the post office server for that user. *Lpr:* given a printer name, return the printer capabilities for it, *i.e.*, the */etc/printcap* information.

4. Traditional Services

Some of the network services in use at Athena consist of older programs which have been modified for use in our environment. In particular, we have modified several programs to use Kerberos and Hesiod.

4.1. File Service

File service is an obvious candidate for a network service. A file service can either consist of a global filesystem visible to all users at all times, or a large set of separately exported filesystems advertised through some intermediate service such as a name service. Athena conforms to the latter model; the filesystem is not a glue that binds everything else together, rather it is a decentralizable resource. In addition, we make no requirement that the form of the remote file service be of any particular type. At this time, we are using two, may use three, and have used others now discarded. This level of detail is opaque to the user community, however.

RVD. The Remote Virtual Disk[14] (RVD) service is an apparently-local disk device that is, in fact, a disk-like local interface connected to physical media through the network. At Athena, its principal advantage is its low-level functionality (block service as opposed to file service) and consequent high performance. It also allows read-only filesystems to be shared, and provides local caching of disk pages. We expect to serve the whole of */usr* to 75 workstations from a single VAX750-class machine, which compares quite favorably to other alternatives.

NFS. The Network File System[15] (NFS) of Sun Microsystems, Inc., is a familiar service to the UNIX community. NFS, like RVD, provides the illusion of local files within the context of a remote file service. Athena's use of NFS is largely conventional, with the exception of strengthening its security by requiring the client to authenticate itself before permitting the client to assume a particular UID on the server. This modification, built on the Kerberos authentication system, has been provided to the original authors of NFS for inclusion in their future releases.

AFS. The Andrew File System[16] (AFS) of CMU's Information Technology Center is currently being evaluated at Athena. The AFS design may offer substantial advantages over NFS and RVD: seamless volume migration, access control lists on a per directory basis, quota on a per volume basis, and tolerance of off-speed networks, to name those apparent to us at this time.

4.2. Mail Service

There are two components of mail service − the routing function and the repository function. Athena routes all mail through a single hub, and uses as many repositories, or *post offices*, as load considerations require. Mail delivery must be reliable. Therefore, no mail may be queued on the workstation for delayed delivery; it must leave the workstation immediately. If an error is to be generated, it must be presented to the user at that time, not later when the workstation might be under the control of another person.

All Athena mail is routed through its mail hub; mail is routed to the post offices, rather than to individual hosts on the network. The post office in turn delivers mail to the recipient upon request. Since the hub delivers mail to post office servers rather than workstations, it is better protected against potential problems such as a mail recipient that is not responding.

From an implementation point of view, Athena has chosen to use the Rand MH system,[17] the *xmh*[18] and EMACS *rmail*[19] client interfaces, and the Post-Office-Protocol[20] (POP) repository capability of MH. The POP subsystem has been modified to require that Kerberos authentication be presented when a user requests mail pickup. As with all such introductions of an authentication requirement, in normal usage this is invisible to the user.

4.3. Print Service

Printing is, of course, a requirement. The Athena model of printing is in many ways comparable to the way in which mail is handled. Print is not queued on the workstation, rather it is immediately dispatched to a print server. Print is directed to printers by logical name, a name that is resolved by the Hesiod name service into an actual server and printer name. The traditional file, */etc/printcap*, has no role on the local workstation. Printing is provided on printers of varied quality, throughput, cost, and features.

At the time of this writing, we are evaluating participation in the development of a standard print service system as part of an industry consortium in this area. The features of this new print systems standard include the expectable benefits of standardization, rational assignment of function between servers and clients, steering of print by its contents, network logging of accounting information (with hooks for authenticating this process), and a substantial improvement in management utilities. We will report on this in subsequent papers.

5. Other Services

This section contains descriptions of additional network services in use at Athena, many of which use the lower-level network services of Kerberos and Hesiod.

5.1. The X Window System

The X Window System[21, 22] was one of the first network services in use at Athena. X is a network transparent, device independent, policy-free, non-proprietary, Unix compatible window system. It encourages consistency of applications by presenting a common library interface independent of the particular display hardware. It permits the display (server) and the application program (client) to be geographically distant or to coexist on a single workstation with no code change required. It is defined by its network protocol, and represents a readily available, top quality interconnection tool for users of multiple systems.

X serves as a fundamental coherence tool with respect to both the user's and application writer's views. A specialized processor needs only network capability and an X client to become available to the Athena user community. Only a distinct portion of the server code is device dependent, which makes X attractive for use in a heterogeneous environment.

5.2. Zephyr

Having access to the same computing environment whether in a dorm room, library, laboratory, or office is a desirable feature of the Athena environment, but it makes locating an Athena user (who may be logged in to any one of hundreds of workstations) very difficult. How, for example, can a member of the Operations staff send a message to users whose filesystems reside on a given server, to warn them of an impending shutdown?

The Zephyr notification service[23, 24] was designed to address these problems. Zephyr servers collect information about which users are logged in where, and use this information to deliver messages upon request. This delivery is based on ‘‘subscriptions’’ whereby users may on the one hand register to receive notices of a given class, and on the other hand send a notice to anyone subscribed to that class, without having to know who they or where they are. Zephyr messages usually appear as windows popped up on an X display.

Some examples of Zephyr may be useful. The *write* program has been revised to use Zephyr to locate the user specified, and to pop up a window on the user's screen, with the name of the sender (and whether the message is authenticated) and displaying the message contents. Another program requests notification when a given user logs in or out. Users may choose whether or not they wish such information be made available about themselves. (Our former head of Systems Engineering, for example, decided to make his logins invisible to other users, after several days of having a queue form at his door within minutes of his logging in for the day.)

Zephyr notice types can be created on the fly. ‘‘Late-night-food’’ is our favorite example – if someone decides to order out for pizza in the middle of the night, he or she can send out a ‘‘late-night-food’’ notice to find out if anyone else is interested.

Zephyr is organized as several distinct components: the server, hostmanager, windowgram client, Zephyr library, and client programs.

Servers. The Zephyr server or servers maintain information about users who are logged in, where they are, whether hosts on the system are up or down, who is subscribing to what types of messages, and also handle the routing and delivery of messages. Hosts are dynamically assigned to one of the Zephyr servers and the servers keep one another informed of their hosts' and one another's status asynchronously and cooperatively.

Hostmanager. One Zephyr host manager runs on each host. It registers the machine with a Zephyr server and transmits messages on behalf of local clients.

Windowgram client. One Zephyr windowgram client runs for each user login session. It registers the user with a Zephyr server, controls the information to be made available about its user, and filters and displays the messages received for its user.

Library. The Zephyr library implements the Zephyr network protocol. It is based on UDP/IP.

Clients. Both senders and receivers of Zephyr notices are clients of Zephyr. Some examples of programs which send messages on behalf of users are On-Line Consulting, to send messages between a user and a consultant; emergency notification, used by Operations staff; and *zwrite*, the Zephyr modified form of */usr/ucb/write*. Some examples of programs which send Zephyr notices not initiated by users are */etc/shutdown*, for warning users of a shutdown; post office servers, to inform users of newly arrived mail; and the *discuss* conferencing system server, to notify users of a new transaction. Zephyr could also be used by print and other servers to notify users of service completion. Some programs that rely on *syslog* have been modified to use Zephyr when possible. With */etc/shutdown*, for example, a Zephyr windowgram duplicating the shutdown syslog message is transmitted to all users who have registered a subscription for the particular message type.

5.3. Time Service

Computers are poor timekeepers; there tend to be as many different opinions of what time it is as there are machines. It is important that computers in a distributed system maintain a certain degree of synchrony for many reasons; at Athena, in particular, the Kerberos authentication system requires that clock variance be no greater than a given amount of time (currently set at five minutes) in order to determine ticket expirations and prevent replay of authentication information by imposters.

In order to synchronize the Athena systems, we have a time service which workstations consult when rebooting. The time server is connected to a radio clock for accuracy, and an uninterruptable power supply for availability. It also participates in a time synchronization service on the ARPANET. The Kerberos master server and the time server run on the same host. Critical network servers use the Network Time Protocol[25] to synchronize.

Setting a workstation's time at boot time seems to be an adequate strategy; it is rare that authentication fails because of clock skew. However, if the allowed time discrepancy were shortened, the workstations booted much less frequently, or machines with less accurate clocks were used, the frequency of time service requests would have to be increased.

5.4. Bulletin Board Service

The *discuss*[26] system, developed by the M.I.T. Student Information Processing Board, provides an electronic bulletin board system in which people can hold ongoing ''meetings'' on topics of interest by reading and posting ''transactions.'' The *discuss* client provides a user interface, and contacts a *discuss* server with requests to read or post transactions. *Discuss* uses Kerberos to authenticate its users, in order to control access to and administration of the meetings, and authenticate transactions. The *discuss* server is a client of Zephyr, and users may subscribe to notification of new transactions in meetings of interest. Because of this, discussions can actually happen in real-time.

5.5. On-Line Consulting Service

The Athena On-Line Consulting service[27] (OLC) provides a means for users with questions and people with expertise to find one another and exchange information. It uses Zephyr to send notices between users and consultants (falling back on */bin/write* if Zephyr is not available). The server remembers which users have asked questions and which consultants have announced themselves available, and matches requests with consultants. If a user goes away (perhaps the workstation crashes or the user logs out), any orphaned consultant message to the user is sent through electronic mail.

5.6. NIP Service

We have reduced the site-specific information needed for an Athena public workstation to one file, namely */etc/rc.conf*, which contains the host name and network address of the workstation, and a series of switches (in the form of environment variables) which specify the range of functionality the workstation permits, such as whether it is an NFS client or not. We would like to remove even this local dependency, and to this end, the Network Information Protocol[28] (NIP) has been written, as a means for booting workstations to communicate with a network server which will dynamically assign them an available Internet address.

6. Absent Services

In this section, we mention two services which might be used in a distributed workstation environment which are not present in the Athena environment.

6.1. Compute Service

Although the Athena development community might be able to make good use of a service that provides remote computation on an idle workstation such as the Butler service at CMU,[29] or a compute service such as the Amoeba Processor Pool,[30] we have not yet determined that this type of service is a high priority for educational computing, and we therefore have neither acquired nor created one.

6.2. Update Service

Currently an update of software residing on the local disk requires personal attention to each workstation. Specifically, a member of the Operations staff must visit each workstation during a period in which it is unused and perform a manual action to initiate update. Although that action is minimal, the requirement to visit the workstation becomes impractical as we make the transition from a thousand workstations to several thousands. An automatic update procedure has been designed and implemented, and is being tested at this time. After a short period during which no user attempts to log in, the workstation comes under the control of the update process. In comparing the workstation to a reference standard (over the network), the local workstation is brought into congruence with that reference standard, and a log message is appended to a standard location to indicate that fact. An added benefit of such a facility is the ability to make minor modifications to the workstation – modifications that were heretofore of too little value to warrant visiting each workstation for manual update.

7. Status of Athena’s Network Services

Except for the NIP and Update services which are still being tested, the services described above are in *broad* use. The areas of future investigation mentioned will be reported on in subsequent papers. To the extent that designed software and exposure testing at our scale is indicative, the software works.

In addition, it is available. All software developed at Project Athena is governed by the same licensing issues familiar to those who have taken the X Window System. Specifically, M.I.T. retains copyright but grants free license without warranty and for any purpose to any party who will maintain that copyright. Inquiries are welcome, either by hardcopy mail or electronically to *info-athena@athena.mit.edu*.

8. Future

Project Athena is entering a three-year extension period. In this period, the focus of development activities will be on maintainability, reliability, and educational productivity. The first two involve a wide range of incremental improvements, such as improvements to boot load installation of workstations. The latter is of crucial importance for the success of the Project. The value added at Athena is to utilize the high degree of connectivity that a network services model of computation provides to foster educational productivity of lasting value and long-term affordability. We have laid the foundation for such accomplishments, and turn now toward fulfillment of these educational goals.

Acknowledgements

The work reported in this paper is the joint effort of more than 30 Athena staff members and students who contributed to the design and implementation of the system described. Project Athena wishes to acknowledge the contributions of the principal backers, DEC and IBM. Through their generosity, we have had the privilege of having these problems first. We also thank the users of Athena, whose experiences have been the metric by which we measure our progress. The authors gratefully acknowledge improvements to this paper suggested by Steve Dyer, John Kohl, Kathy Lieben, Jon Rochlis, Mark Rosenstein, Jerry Saltzer, and Win Treese.

References

1. E. Balkovich, S. R. Lerman, and R. P. Parmelee, ‘‘Computing in Higher Education: The Athena Experience,’’ *Communications of the ACM* **28**(11), pp. 1214-1224, ACM (November, 1985).
2. S. P. Miller, B. C. Neuman, J. I. Schiller, and J. H. Saltzer, *Section E.2.1: Kerberos Authentication and Authorization System,* M.I.T. Project Athena, Cambridge, Massachusetts (December 21, 1987).
3. J. G. Steiner, B. C. Neuman, and J. I. Schiller, ‘‘Kerberos: An Authentication Service for Open Network Systems,’’ pp. 191-202 in *Usenix Conference Proceedings*, Dallas, Texas (February, 1988).
4. R. M. Needham and M. D. Schroeder, ‘‘Using Encryption for Authentication in Large Networks of Computers,’’ *Communications of the ACM* **21**(12), pp. 993-999 (December, 1978).
5. National Bureau of Standards, ‘‘Data Encryption Standard,’’ Federal Information Processing Standards Publication 46, Government Printing Office, Washington, D.C. (1977).
6. P. J. Levine, M. R. Gretzinger, J. M. Diaz, W. E. Sommerfeld, and K. Raeburn, *Section E.1: Service Management System,* M.I.T. Project Athena, Cambridge, Massachusetts (1987).
7. M. A. Rosenstein, D. E. Geer, and P. J. Levine, ‘‘The Athena Service Management System,’’ pp. 203-212 in *Usenix Conference Proceedings*, Dallas, Texas (February, 1988).
8. Relational Technology, Inc., *Ingres Reference Manual*, Release 5.0, Unix, 1986.
9. N. Mendelsohn, *A Guide to Using GDB,* M.I.T. Project Athena (1987). Version 0.1
10. S. P. Dyer and F. S. Hsu, *Section E.2.3: Hesiod Name Service,* M.I.T. Project Athena, Cambridge, Massachusetts (1987).
11. S. P. Dyer, ‘‘Hesiod,’’ pp. 183-190 in *Usenix Conference Proceedings*, Dallas, Texas (February, 1988).
12. P. Mockapetris, *RFC 1034 – Domain Names – Concepts and Facilities,* USC/Information Sciences Institute (November, 1987).
13. P. Mockapetris, *RFC 1035 – Domain Implementation and Specification,* USC/Information Sciences Institute (November, 1987).
14. M. Greenwald and J. V. Sciver, *Remote Virtual Disk Protocol Specification,* Massachusetts Institue of Technology (1986).
15. Sun Microsystems, *NFS Protocol Specification and Services Manual*, Revision A, 1987.
16. M. Satyanarayanan, John H. Howard, David A. Nichols, Robert N. Sidebotham, Alfred Z. Spector, and Michael J. West, ‘‘The ITC Distributed File System: Principles and Design,’’ *Proceedings of the Tenth ACM Symposium on Distributed Systems Principles* **19**(5), pp. 35-50 (December, 1985).

17. Rand Corp., *The Rand Message Handling System: User's Manual,* U.C.I. Dept. of Information & Computer Science, Irvine, California (November, 1985).
18. ‘‘Guide to the xmh Mail Handler,’’ in *Ultrix Worksystem Software Version 1.1*, Digital Equipment Corporation.
19. Richard Stallman, *GNU Emacs Manual*, March 1987.
20. M. Butler, J. Postel, D. Chase, J. Goldberger, and J. K. Reynolds, ‘‘Post Office Protocol Version 2,’’ RFC 937, ISI (February 1985).
21. R. W. Scheifler and J. Gettys, ‘‘The X Window System,’’ *ACM Transactions On Graphics* **5**(2), pp. 79-109 (April, 1987).
22. *X Window System Protocol,* M.I.T. Software Distribution Center (September, 1987).
23. C. A. DellaFera, M. W. Eichin, R. S. French, D. C. Jedlinsky, J. T. Kohl, and W. E. Sommerfeld, *Section E.4.1: Zephyr Notification Service,* M.I.T. Project Athena, Cambridge, Massachusetts (December 21, 1987).
24. C. A. DellaFera, M. W. Eichin, R. S. French, D. C. Jedlinsky, J. T. Kohl, and W. E. Sommerfeld, ‘‘The Zephyr Notification System,’’ pp. 213-220 in *Usenix Conference Proceedings*, Dallas, Texas (February, 1988).
25. D. L. Mills, ‘‘Network Time Protocol,’’ RFC 958, M/A-COM Linkabit (September 1985).
26. Stan Zanarotti, Ken Raeburn, and Nancy Gilman, *Using Discuss,* M.I.T. Student Information Processing Board (April 1988).
27. D. E. Geer, W. G. Treese, B. Anderson, and T. Coppetta, *On-Line Consulting in a Distributed Educational Environment (in preparation),* M.I.T. Project Athena.
28. Mark A. Rosenstein and Jeffrey I. Schiller, ‘‘Network Information Protocol,’’ RFC (in preparation).
29. David A. Nichols, ‘‘Using Idle Workstations in a Shared Computing Environment,’’ in *Proceedings of the 11'th ACM Symposium on Operating Systems Principles* (November 1987).
30. R. van Renesse, A. S. Tanenbaum, and G. J. Sharp, ‘‘The Workstation: Computing Resource or Just a Terminal?,’’ in *Proceedings: Workshop on Workstation Operating Systems*, The Computer Society of the IEEE (November 1987).