Section C

Project Athena's Model of Computation

by E. E. Balkovich, R. P. Parmelee, and J. H. Saltzer

This document discusses and outlines the model of computing to be developed by Project Athena. The computing model concentrates on academic needs and does not attempt to encompass research and administrative computing. The description focuses on features that are important to developers and users of applications. Software to be developed by project staff is identified and a discussion of coherence with respect to the model is included. Also, the evolution of existing computing facilities to implement the model is outlined.

1. introduction

The scope of this document is the model of computing to be developed by Project Athena. The computing model concentrates on academic needs and does not try to encompass research and administrative computing.

The lasting technical contributions of this project will most likely deal with coherence and discipline-specific software used in education (applications). Thus, a portion of the document is devoted to a discussion of coherence and how it is reflected in the computing model presented to the users and developers of applications.

Project Athena will adopt and use externally developed software wherever feasible. In particular, base system software and applications with a broad commercial market (e.g., word processing) can be better developed and supported outside of the project. Project Athena will seek to influence potential vendors of such software to provide products which suit project needs. The project will concentrate its own development efforts on making applications and system software that enable coherence.

This document has two objectives. The first objective is to describe the computing model that Project Athena expects to implement by the end of its experiment. If the experiment is successful, then this computing model will be refined and carried forward for future use. The description of the computing model focuses on features that are important to developers and users of applications. It also identifies software to be developed by the project staff. The second objective of the document is to outline how existing computing facilities will evolve to implement the computing model.

2. The nature of academic computing

From the point of view of the student, the dominant use of computers will be to interact with existing programs rather than to write new ones. The use of computing to develop or modify programs will be a component of use, but a minor one. Program development will be

Copyright © 1985 by the Massachusetts Institute of Technology

required when it is an integral part of an academic subject (as in some computer science subjects) or when the material being taught has a strong algorithmic component. Athena will provide an open system that permits anyone to add new software. However, academic usage will emphasize the execution of existing programs.

Most student users will view computing as a tool or a service. It will deliver services such as electronic mail and will provide tools that support activities such as word processing, sketching, simulation, graphic illustration, laboratory data acquisition, and financial models. Students will use computing in many of their subjects. It will routinely be used in laboratories, for homework, and to review and explore lecture materials. The computers and programs that students use will be well integrated. For example, tools such as word processing will accept input generated by the application software. Furthermore, a coherent computing environment will be available throughout the campus.

From the point of view of faculty, the dominant use of computing will be as a presentation medium, and to develop new course materials. Faculty will use Athena computing to distribute course materials (e.g., programs and data) to students, and to communicate with students. Athena will also provide a variety of standard approaches and techniques for developing new course materials (e.g., libraries for generating 3-d graphics, standard data representations and programming interfaces, and standard user interfaces). These supporting materials will make it possible for faculty to develop new course materials with minimal effort.

3. Hardware

The academic experiment of Project Athena is based on the premise that low-cost, powerful workstations will be feasible in a few years. The sections that follow describe the assumptions made about the hardware components and offer the rationale for the system configuration.

3.1. Components

The computing system envisioned includes public and privately owned workstations, a campus computer network, and "backend" computers that offer general services via the network. The following subsections discuss each of these system components.

3.1.1. Workstations

Workstations may be owned by an individual or may be part of a public facility owned by M.I.T. Workstations will be located in private residences, offices and public areas (e.g., a classroom or laboratory). The owner of a private workstation will assume that the workstation is never used by anyone else. A public workstation will be dedicated to a single individual for a session. However, users of public workstations will need to assume that others use it between sessions. Public workstations are "serially reusable" resources. These assumptions result in differences in the way public and private workstations will be used. These differences are elaborated in later sections of this document.

Workstations will be manufactured by different vendors. However, they will have the following common characteristics:

- A memory of 1-16 Mbytes with memory management features that can be used to communicate among independent address spaces and to effect dynamic linking in a large address space.

- A high performance processor capable of executing at least one million instructions per second. It may have optional floating point hardware. The architectural features of the processor may vary.
- Local mass storage in the form of a hard disk and some removable media (e.g., floppy disk or cartridge tape). The capacity and format of these devices may vary. The capacity of the hard disk is sufficient to store a single-user operating system, a temporary file area and swapping and/or paging space.
- A high bandwidth, bit-mapped display. Its resolution is at least that needed for NTSC input or output. Color and resolution may vary. Low bandwidth, character-only displays are not supported.
- A keyboard. The layout and coding of key strokes may vary. The processor must be able to detect up-down key transitions.
- A pointing device that can identify individual bit positions in the display. The nature and implementation of the device may vary.
- A network interface for one of a small number of network attachment methods. At present there are two methods: Ethernet local area networks and RS-232 asynchronous ports. The network interface supports a data rate of at least 64K bits per second. (See the discussion of networks that follows.)
- A (optional) printer or hardcopy graphics device. The nature and implementation of the device may vary. The range of functions and characteristics is not currently specified.
- An "open" system that can be extended to include peripherals such as keyboards, sound generation devices, video input/output, data acquisition and control devices. In general this will take the form of a "standard" bus that can be used to incorporate peripheral devices.

3.1.2. Network

M.I.T. will own and operate a campus computer network. All users (owners) of workstations in the M.I.T. academic community will be able to access the network.

The network will be implemented with multiple technologies. It will be based on a high-speed, backbone network. Many campus facilities will have access to high-speed, local area networks connected to the backbone network. Other locations will have access to lower-speed, wide area networks connected to the backbone network.

Although multiple technologies will be used to implement the network, there is no commitment to make every network technology available at every workstation site. Initially, Ethernet and RS-232 asynchronous ports will be used. Other network attachments methods will be considered, if a need for other technology is identified. However, the overall number of network attachment methods is expected to be small.

All Athena computers will communicate using a standard protocol family. The protocol family will be based on TCP/IP. TCP/IP will be used because it is implemented for the systems selected for use in Project Athena. The important principle it implements is that of a standard set of protocols. In principle, TCP/IP could be replaced by another suite of protocols later in the project (e.g., ISO). Future replacement of TCP/IP would assume that applications are written to anticipate a protocol change and that they are isolated from protocol-specific features of the system.

Project Athena regards protocol specifications as published information accessible to the entire community. Anyone willing to implement the protocols for a specific machine and operating system could achieve a degree of inter-operability with Athena computers. However, there is no commitment to support the development and testing of such implementations beyond the set of supported Athena workstations. Furthermore, inter-operability will never assure that arbitrary applications can be made to execute on a computer that is not a supported Athena workstation.

3.1.3. Service computers

M.I.T. will own and operate a number of general purpose computers and peripherals. These computers will be connected to the network and will be dedicated to specific functions such as electronic mail, file storage and database support.

The computers and peripherals will be supplied by different vendors. Their nature and implementation may vary. They need not have the same characteristics as workstations. Although their internal structure may vary, they will all implement the standard protocol suite.

3.2. Rationale

Athena's model of computing provides users with a predictable response that is independent of system load. Timesharing cannot satisfy this requirement. The use of workstations to deliver computing services will also provide users with a great degree of autonomy. Users will operate their workstations independent of other system components and will be able to customize their environment.

Unlike personal computing, timesharing provides information sharing. In Athena's model, the network and service computers will provide shared resources and information. Users of workstations will have discretionary access to traditional timesharing services such as electronic mail and shared files. Some shared resources will be inherently limited (e.g., storage, printing, special purpose computers). The network will be used to administer shared resources centrally. Services provided by the network will be made to operate at specific levels of reliability.

The system will be flexible. It will be possible to purchase computers and networks from multiple vendors. It will be possible to locate computers in existing work places and on-campus residences with minimal renovation. Network technology will make it possible to site service computers without imposing major constraints on the location of workstations.

Finally, the system will decentralize the costs of acquisition and operation. Individuals will be able to purchase computers for use as private workstations. M.I.T. will provide service computers, the network infrastructure and some number of public workstations.

Users will be responsible for the operation (and maintenance) of workstations. Functions which require a support organization or which consume resources will be delivered via the network. These services will be accounted for. Their costs will be identified and recovered on a "fee-for-service" basis, applied against a quota, charged as university educational overhead or billed to the individual or their department, as appropriate.

The distribution and maintenance of privately (and publicly) owned workstations present a number of difficult and important operational problems. In addition, there are difficult and important problems associated with the distribution and maintenance of licensed software that is executed or stored at a workstation. The requirements for distribution and

maintenance of hardware and software are not yet specified and will be the subject of another planning document.

4. Coherence

Project Athena's educational experiment will stimulate and evaluate the use of computing in education at M.I.T. The sustained use and development of educational computing requires that applications minimize unnecessary differences and that training required to use the system be managable. It must be possible for applications to share common code and to build on one another. It must also be possible to make a one-time investment in training that is applicable to all uses of the system, or to make the use of applications and the system self-evident. Modularity and coherent module interfaces are necessary to achieve these goals.

A key module is the operating system. It must insulate the user and applications from differences in hardware supplied by multiple vendors and technological change. Other major modules include discipline-specific software (e.g., software that illustrates electro-magnetic fields), and general purpose software (e.g., word processing). These modules define three interfaces that must provide coherence:

- *System Interface:* the interface between the operating system and applications
- *Application Interface:* the interface between applications
- *User Interface:* the interface between the user, and the applications and operating system

Coherence is the principal technical objective of Project Athena. The following sections discuss how coherence will be realized.

4.1. System interface

The interface which the operating system presents to application software will be the same on all machines. This will reduce the amount of training needed to use computing and will establish a framework for writing (ex)portable applications. This will be achieved by using a single operating system and a single communication protocol family. The operating system will be based on Unix™.[1] The protocol family will be based on TCP/IP.

Unix is available (and supported by vendors) for a variety of existing and planned computers. The ability to port Unix to new hardware is established and makes Unix a desirable operating system for new hardware. Some versions of Unix provide functions to exploit the hardware technology assumed by this project (e.g., paged memory).

Unix will be used to implement the same "virtual machine" on all workstations. The virtual machine will be defined by a set of kernel functions and the libraries used to invoke them. This virtual machine will minimize hardware specific features and provide a uniform, lowest-level programming interface for applications. This interface will be documented for supported hardware and software systems. No attempt will be made to verify that other systems provide the functionality of the virtual machine assumed by Athena. The goal of exporting M.I.T.-developed applications will depend on extensive

[1]Unix is a trademark of Bell Laboratories.

university and industry adoption of Unix implementations with features similar to those used by Athena.

Implementations of the TCP/IP protocols are available from several sources, including M.I.T. Some versions of Unix include an implementation. Implementations that function as a "layered product" will be available and can be ported to Unix. The use of the TCP/IP protocols with a range of network technologies (spanning both local and wide area networks) and applications is established.

The TCP/IP protocols are used to implement the same "virtual network" on all network technologies used in this project. The virtual network will hide technology specific features. A uniform virtual network will be needed to provide common communication functions that span multiple vendor's computers. The goal of exporting M.I.T.-developed applications will depend, to some extent, on extensive university adoption of the TCP/IP protocols.

A coherent system interface is *not* guaranteed just by adopting a uniform operating system and protocol family. Industry and universities support multiple versions of Unix. These versions may diverge with time. Unix was originally designed to provide mechanisms that support a system interface based on character streams and a small address space. New technology (e.g., large address spaces and bit-mapped displays) will require extensions to the system interface to fully exploit its potential. There are no widely accepted Unix extensions that provide interfaces to this technology. Thus, Project Athena will experiment with such extensions. The experimental approach will track emerging "standard" approaches and will seek to have successful ideas adopted and supported by commercial vendors of Unix implementations. The extensions to be experimented with include:

- Window management, integrated with graphics
- Concurrent programming in a single address space
- Dynamic linking
- Remote procedure calls

The system interface will also be defined by a subset of existing Unix functions. These functions will be ones that are supported in common variants of Unix, or ones which can be developed as extensions to common variants of Unix (e.g., implementing selected 4.2 bsd kernel functions as a program library). These extensions will also introduce mechanisms needed to support new technology. In addition, the project will establish guidelines for the use of Unix, and its extensions, which help application developers to create (ex)portable software.

There are a number of criteria which can be used to judge whether Project Athena successfully develops a coherent system interface. The most important criterion is whether or not applications run on different types of computers using different types of networks. This judgement must consider whether extensions to Unix and guidelines for application developers successfully manage the differences in removable media, keyboards, pointing devices and displays, including printers. Finally, in order to export applications developed at M.I.T., the Unix system in use at M.I.T. must be compatible with "industry standard" Unix implementations.

Project Athena will jeopardize most of its objectives if it fails to define and implement a coherent system interface. The system interface is the basis for higher levels of coherence. If the project fails to achieve a coherent system interface, then it will not achieve coherent

higher level interfaces and should expect the effectiveness of computing as a medium for education to be significantly diminished. It will be difficult to share software for academic applications within M.I.T., let alone export software. Thus, greater investments will be required to develop a new application. Applications will be system dependent and will require more user training to understand the effects of hardware differences. Even if M.I.T. ultimately judges the academic value of computing to be worth continued investment, M.I.T.'s future campus computing solutions will be more constrained.

4.2. Application interface

The investment required to develop new applications can be reduced in several ways. A high risk, high payoff approach is to look for "vehicles" other than one-of-a-kind programs for delivering educational materials. Examples of "vehicles" include spread sheets or application generators for drill and practice materials. A lower risk, lower payoff approach is to support methods and techniques that reduce the effort required to create new software. An example would be a library of routines for 3-d graphics. The former approach requires considerable innovation and is best left to exploration by faculty and students. The latter approach can be systematically attacked by defining and implementing the interface between applications.

The interface between applications is defined by data and mechanisms for manipulating them. A working hypothesis of Project Athena is that most scientific and engineering applications can usefully interact using only a small number (20-30) data types (e.g., graphs, arrays, tables, etc.). The application interface can be defined by implementing common representations of these data types and mechanisms for manipulating them.[2]

If the number of ideas in the application interface is small and concise, then the programming interface of an application is easier to understand. If an application's use of the concepts in the interface is complete, then program's can be made to combine in standard ways. A program which takes advantage of such an interface is more likely to be used with other programs. For example, a word processing program should be able to incorporate the tables or graphs produced by a program performing an analysis. The ability to combine applications should also lead to interdisciplinary efforts. For example, programs which model thermodynamic systems could be used with a variety of discipline-specific applications.

The data types that define the application interface are probably "large grained". A typical data object is probably larger than a complex number and is probably smaller than a database. Data objects are probably manipulated only a few times per execution of an application (e.g., at the start and end of the program).

The mechanisms for manipulating data objects are probably based on several programming paradigms. These paradigms include concepts such as operations on abstract data types and streams.[3]

---

[2]This is one of the motivations for keeping the number of "supported" programming languages small. It is practical to provide such an implementation for only a small number of languages.

[3]A more complete list of candidate paradigms can be developed from sources such as Abelson, H., et al., *The Structure and Interpretation of Computer Programs*, McGraw-Hill, (1985), or Bobrow, D., "If Prolog is the Answer, What is the Question?", *Proceedings of the International Conference on Fifth Generation Computer Systems*, Tokyo, (November 6-9, 1984).

Project Athena will identify and implement the application interface. The implementation will allow an application, written in a supported programming language, to create and manipulate instances of interface data types as it executes. The implementation will also allow the same interface data object to be manipulated from applications written in different supported programming languages. The supported programming languages will include Lisp, Fortran, C and Pascal. The implementation will also incorporate standards (e.g., IGES, DIF, etc.) which would allow externally developed software to be included in the system.

The implementation of the application interface will use the "virtual machine" provided by Unix. In the worst case, it should be possible to convert the representation of interface data objects to ASCII characters and use existing Unix functions (e.g., files) to manipulate them. There is no requirement for an implementation of the Unix kernel to provide special support for such data types. The implementation of these data types will take the form of a library which can be linked with applications.[4] The implementation of the application interface will be (ex)portable. It will not necessary to solve the general information interchange problem. The problem will be simplified by limiting solutions to the supported hardware, by minimizing extensions of the "virtual machine" provided by existing Unix systems and by providing implementations for only a small number of data types oriented towards specific applications.

There are several ways the project can fail to achieve a coherent application interface. An implementation is practical only if the product of the number of supported languages and the number of data types is modest. There may be pressure to expand the number of languages. The conjecture that the number of needed data types is small may be flawed. It may also be the case that a Unix-based implementation of the application interface may be inefficient or ponderous to use. If the performance of the implementation is perceived to be poor, its use will be resisted.

There are several criteria that can be used to determine if an implementation of the application interface is successful. Clearly, its acceptance and use by application developers is the principal criterion. The value of an application interface is demonstrated only when applications are combined. Finally, the application interface should be (ex)portable to other Unix systems.

If the project fails to produce an application interface, then the effectiveness of computing in education in probably reduced, but not completely jeopardized. Faculty and student efforts will undoubtly produce important individual applications, independent of the application interface. However, failure to define and implement an interface between applications implies that programs will be poorly integrated (e.g., it will be difficult, if not impossible, to use the output of a program illustrating fluid flows as input to a word processing programs used to generate a report).

If existing applications cannot be effectively combined, then the cost of producing or using applications may be prohibitively expensive. The interdisciplinary use of software would be non-existent. Also, there would be little chance for a common understanding of of how to exploit computing in education. This would influence M.I.T.'s decisions about future

---

[4]For example, one possible implementation of the application interface would allow multiple programs, written in different programming languages, to execute as a single Unix process. Alternatively, Unix features (e.g., pipes, sockets and files) could be used to pass objects between programs executed as separate Unix processes.

computer acquisitions. Finally, if the application interface requires Unix to be modified, then it probably jeopardizes the use of M.I.T. produced applications outside of M.I.T.

4.3. User interface

New technology (i.e., bit mapped displays and computers dedicated to a single user) and the potential for more comprehensive use of computing motivate the need to refine user interfaces. A "standard" framework for the user interface, common to all applications, can reduce the amount of training needed to use applications. New applications will emphasize interaction and graphics. If code for user interfaces can be generalized, it will reduce the amount of new code required to develop an application.

There are two ways to think about a "standard" framework for user interfaces. The first is to require all applications to use a single framework. The interface may be complex enough to require training. However, because there is only one, training is a one-time requirement. The alternative is to allow user interfaces to differ, but to require that any interface be obvious to use and self-explanatory.

In either case, the user interface of an application should be based on established principles, libraries of routines and tools. Tools and libraries might include programs for creating and managing "help" commands, heuristics for displaying information (e.g., heuristics for tiling windows), support for graphics, or input/output packages which isolate an application from a user's preferred mode of interaction (e.g., the user's choice of menu-based or command-based interaction with a program should be transparent to the application).[5] Libraries of routines should utilize features of the system interface that make the user interface device independent (i.e., it should work with different types of bit mapped displays). When the user needs to interact with the operating system, the same principles and techniques should be applied.

The major risk of a "standard" framework for user interfaces is that it suppresses innovation if it is applied prematurely or blindly. Concepts used to design a user interface span diverse areas of research (e.g., human factors and graphics) and there is little agreement on what contributes to a good user interface.

The principles for a user interface framework require considerable research. It is unlikely that Project Athena's efforts will be an unqualified success. Since user interfaces are, in part, driven by new hardware technology, the software developed by the project may diverge from emerging "industry standards" in areas such as window management.

In order to successfully implement a coherent user interface, Project Athena must identify a body of principles and implement software support for those principles. The principles and their implementation can be judged to be successful if they are adopted for use in applications. Presumably, they are adopted, in part, because they reduce the effort required to train users and/or the effort required to develop the application. The resulting applications should be device independent. Finally, the implementation of the user interface should be consistent with emerging "industry standards".

If the project fails to define and implement a coherent user interface, the consequences are tolerable. Failure would not preclude individual applications that have significant

---

[5]Coutaz, J., *A Paradigm for User Interface Architecture*, CMU-CS-84-124, Carnegie Mellon University, (1984).

educational impact. The ability to (ex)port applications should only be affected if the effort completely fails to identify principles for applying system interface features which hide differences in display and input devices. The sharing of data and integration of applications would not be prevented. The most significant consequence is that a failure will not reduce the overhead of training users to apply computing in education. It may ultimately limit the extent to which the educational process can afford to apply computers.

5. Using the system

Users of Athena will see a system of individual computers linked by a network. Public and private workstations may be the same types of hardware. Differences will result from the differences in vendor's products and the need to provide special purpose support (e.g., a laboratory computer may differ significantly from most other workstations). However, differences will be minimized by a coherent set of interfaces. The network will provide workstations with access to centrally operated computers and peripherals that deliver specific services. Unlike timesharing or batch systems, the workstations will be dedicated to one individual per session.

It will be assumed that workstations are not designed to be physically portable. A user will not be expected to bring a private workstation to a classroom or laboratory. Any member of the campus community will be able to use any public workstation as well as privately owned equipment. A user will not limited to using just one computer. The system will provide facilities that make it possible to share information and access data and programs from any computer.

5.1. Using the system

Users will have access to more than one filing system. Every filing system will present the user with a hierarchy of directories and files. Each file will provide byte addressing of 8-bit bytes of data. This will essentially be the lowest-level, common view provided by existing Unix filing systems.

A filing system will be implemented on one storage device (e.g., a disk). A storage device may support one or more filing systems. Storage devices used by filing systems may be part of a workstation, or may be accessed via the network. Filing systems will be named objects. Users will be able to copy files between filing systems with little regard for the location of the filing system.

There will be several types of filing systems. Each type of filing system will have different properties. These properties will be apparent and important to users. The types of filing systems will include:

- General purpose libraries
- Class libraries
- User lockers
- Local filing systems
- Removable filing systems

General purpose libraries will be filing systems that can be accessed from any workstation via the network. They will be used to store files that define the operating system and other general purpose programs and data. The contents of general purpose libraries will be

established and administered by the organization(s) responsible for operating Athena. Any user will be able to read the contents of a general purpose library. These libraries will be used to distribute the operating system and general purpose applications and data.[6]

Class libraries will be filing systems that can be accessed from any workstation via the network. They will be used to store files that contain class specific programs and data. The contents of class libraries will be established and administered by instructors. Any member of the class will be able to read the contents of a class library. Class libraries will be used to distribute course specific materials to students.[7]

User lockers will be filing systems that can be accessed from any workstation via the network. They will be used to store files for one user. The contents of a locker will be established and administered by a single user. Only that user will be able to read and write the contents of the locker. User lockers will be used to retain information between sessions at workstations and to move data between workstations. For example, a student might collect data using a laboratory workstation, save it in a locker, and then access and analyze the data from any workstation.

Local filing systems will be unique to a single workstation. They will be used to store files copied from other filing systems (e.g., general purpose or class libraries, or a locker) and to store files created by the workstation. Any user of a workstation will be able to read or write the contents of its local filing system. The local filing system of a workstation cannot be accessed by any other computer.

Local filing systems will be used to store copies of standard system software as well as user data and programs. A centrally administered organization will maintain and distribute standard system software. This distribution will be used to maintain and restore the integrity of that portion of local filing systems used for standard system software.

Removable filing systems will be used to store files copied from other filing systems (e.g., general purpose or class libraries, lockers, and local filing systems). They will be used to back up or archive data and programs. A removable filing system need not always be on line. It will be possible to physically remove a removable filing system from one workstation and install it in another workstation of the same type. Removable filing systems will be machine specific and may be used only with one type of workstation (e.g., media used for the filing system, such as a floppy disk, may work with only one vendor's products and even then with only one type of product). Any user of a workstation of the requisite type will be able to read or write the contents of a removable filing system that he or she possesses. The removable filing system of a workstation cannot be accessed by any other computer.

The contents of general purpose and class libraries, and user lockers will be backed up. Their contents will not be archived. The back up procedures will be initiated by centrally administered organizations responsible for delivering network-provided services. Their

---

[6]There are a number of open questions that must be addressed by the development efforts of the project. Such questions include: are file always copied from a general purpose library to a local filing system before use? If so, how are large public databases handled? How is access to licensed software controlled? How is unauthorized distribution restricted?

[7]The concept of a Remote Virtual Disk appears to be a good model for the implementation of a class library.

procedures will be designed to guard against disasters (e.g., a head crash) and will not provide users with a means of recovering files that they have been accidentally deleted. There will be some probability that information can be lost after a system failure. However, it will be possible to bound the amount of information lost (e.g., at most one day's input). The contents of these filing systems may also be inaccessible for periods of time. In general, the operating characteristics of these filing systems will meet or exceed those of of a "typical" Unix timesharing system.

Users will be expected to augment these backup procedures. The user will be able to initiate a backup or archive of data using removable media. It is also conceivable that existing computer service organizations (not Athena) may offer a data archival service on a fee-for-service basis. It will be the responsibility of the user to explicitly invoke such a service.

The administration of that portion of local and removable filing systems devoted to user data and programs will be the sole responsibility of users. Their contents will not be backed up or archived by a centrally administered organization. However, there will be general purpose applications that make it convenient for a user to save and restore selected parts of a local filing system from a locker or a removable filing system. These functions will allow users to implement a personal strategy for backing up information (e.g., such a function might be part of a ".logout" file, or might be periodically invoked as a background process).

The hard disk of a workstation will be used to implement the local filing system. In addition, it will be used for paging and the storage of temporary files. In the case of public workstations, the hard disk and the local filing system stored there will be shared resources with no access controls. In particular, it will not be possible to guarantee the integrity or contents of the local filing system between sessions.

This storage model can be implemented using existing extensions to Unix (e.g., the Remote Virtual Disk software) and/or explicit invocations of existing remote file copy utilities. The model does have limitations. For example, it is not possible to share write-access to a filing system. Also, there is no mechanism, other than mail, that would allow students to share information needed for a joint project. However, the storage model does address most of the basic information sharing needs of academic computing. This basic model of file sharing will be extended to reduce these limitations as resources permit.

5.2. Computation

The user will view all applications as running on the workstation being used for the session. If other sources of computation are required (e.g., access to a large database or a high-performance computer), then a program run on the user's workstation will initiate and control access to these computations in a way that is transparent to the user. In general, a user will not be concerned or aware of where a program is executed. It will always appear to execute locally.

5.3. Network services

The network will be a source of discretionary services. There will be no requirement for a workstation to be connected to the network at all times. In particular, the resources of a workstation will not be available to another computer via the network (i.e., local files or computational resources cannot be involuntarily accessed from another workstation via the network).

The network will be a "universal" medium for exchanging machine readable information. It will be possible to use it (in conjunction with network provided services) to move information between workstations. It will be the only medium guaranteed to be vendor independent. The same guarantee cannot be made for removable media, even for different products supplied by the same vendor.

A workstation may be used to execute an application that accesses network supplied services. Such applications are run at the discretion of the user. Network supplied services will include:

- Electronic mail
- Hardcopy output of text and graphics
- Centrally administered file storage (e.g., general purpose and class libraries, and user lockers)
- Alternative removable media (e.g., magnetic tape)
- Name resolution (e.g., translation of a person's name to an address for electronic mail)
- Special purpose computations (e.g., a Macsyma server or database server)

Some of these services will be accounted for and some will require authentication of the user.

In general, network supplied services will be centrally administered and the responsibility of the organization(s) supporting Athena. Computers, owned and operated by M.I.T., will be used to provide concurrent service to multiple workstations. These service computers will be accessible to all users of the system and their location will be transparent. However, the performance of the network connecting them to specific workstations may vary.

5.4. Session characteristics

An user will run a small number of applications in a typical session. Some of these applications will be needed for classes and will be used frequently. These programs will be used for homework or to review and explore lecture materials. They might also be general purpose programs such as word processing. Other applications will be used for reference or remedial instruction and will be used less frequently. Such programs might include library services or a calculus drill.

Application will require the Unix kernel and a small number of key files (e.g., the "/dev" directory) for run time support. Much of Unix will *not* be used. The command interface will *not* be a Unix shell. It will be an application(s) that conforms to the principles and implementation techniques of the system, application and user interfaces.

The resources of a workstation will not be implicitly accessible to other workstations or service computers. For example, the contents of the local filing system will not be shared. This will not be true for centrally operated computers attached to the network. Users will be able to share information by placing it in centrally administered filing systems, copying removable filing systems or by explicitly mailing it to other users.

Users of a workstation will be able to explicitly offer a service (e.g., a faculty member might wish to make a multi-player game available as part of a class assignment)

implemented with the resources of their workstation. It will be the user's responsibility to locally administer such a service. Athena will provide the tools necessary to do so (e.g., name resolution and authentication services).

Authentication will not be required to use a public or private workstation. A public workstation may be used by anyone in the M.I.T. academic community. A private workstation may be used by anyone with access to it. A single individual may find it convenient to use any one of several computers. For example, a student may own a workstation and may also use a public workstation that is part of a classroom facility or laboratory. Use of a public or private workstation does not automatically permit the use of network supplied services. Network services may require user authentication for accounting and protection.

The hard disk of a workstation will be used in several ways. It will support swapping and/or paging operations which the operating system may perform. It will also provide storage for a local filing system used to store operating system code and data. This code will essentially be the Unix kernel. The hard disk will also be used to "boot" the computer. Finally, the hard disk will also be used as a local filing system for applications and data.

There will be applications that allow the user to quickly test the integrity of the local copy of the operating system. This test will verify that the local filing system is intact and that all components of the operating system are present. It will be possible to load this test application from a number of different media. These media will include the hard disk, removable media and the network. Users will run such applications at the start of a session on a public workstation or when they have reason to believe that local storage has been corrupted.

If a user cannot establish the integrity of the operating system, then the user will be able to efficiently rebuild the operating system from a "standard" distribution.[8] It will be possible to rebuild the operating system using a number of different media. They will include removable media (kept with the individual computer) and the network.

The user will be responsible for the integrity of programs and data stored in the local filing system. There will be applications which can be used to test the integrity of this filing system. However, only the user can verify if the local filing system contains all of the needed applications and data. There will also be applications that allow a user to repair a damaged local or remote filing system. The applications used to repair filing system may lose data.

The local filing system of a workstation defines the domain of search for applications. Only those applications copied into the local filing system will be executable. In general, local filing systems will be used as an explicitly managed file cache used to improve workstation performance. Programs and data stored locally can be obtained from removable filing systems or from filing systems accessible via the network.

Once executing, it will be possible for an application to use the local and removable filing systems. It will also be possible for an application to interact (using a protocol) with a program executed elsewhere in the network to access data stored in remote filing systems (e.g., querying a database).

---

[8]Are there multiple versions of the operating system which the user can chose from? If so, does the test for integrity have to establish which version is in use?

5.5. Differences

There will be inherent differences between public and private workstations that are apparent to users. These differences will stem from the assumption that multiple individuals will use a public workstation.

The hard disk of a public workstation will be viewed as a cache for programs and data referenced during a single session. Programs and data written in a local filing system by one user may be overwritten by another. Furthermore, programs and data written there by one user will be readable by another user unless explicit steps are taken to remove them. There will be applications which can be used to remove information from local filing systems. However, it will be the user's responsiblity to implement a strategy for using them. The privacy of information left in the local filing system will not be guaranteed.

A public workstation will operate in one of three states:

- Standard state
- Unauthorized state
- Authorized state

It will be possible to a user to place the system in the standard state at any time during a session. The standard state will remove any local filing system containing private data and will verify the integrity of and restore, if necessary, the standard system software.

Any use of the workstation (e.g., executing a standard system function, or loading a program from a removable filing system) will place the workstation in the unauthorized state. This state allows the user access to any resource available at the workstation, but does not provide the user with access to network supplied resources that may require authorization. The user may return the the standard state from the unauthorized state at any time.

The user will perform an operation similar to "login" to enter the authorized state of the workstation. This operation will verify the identity of the user and allow the user to access network resources which require authorization. In general, a user will need to enter the authorized state only once per session. The authorized state can be entered from either the standard or unauthorized state of the workstation. The user can return to the standard state of the system from the authorized state at any time. If a user leaves a workstation in an authorized state, then the next user of the workstation may exercise any of the authorized privledges.

The hard disk of a private workstation will be viewed as providing viable long-term local file store for programs and data. Access to that information will be controlled by access to the private workstation.

A private workstation's local filing system will store data and programs that change less frequently than releases of the operating system. Thus, it will be possible to install a new version of the operating system on a personal computer without disturbing the contents of the local filing system.