M. I. T. PROJECT ATHENA

The Athena File Storage Model

by J. H. Saltzer

DRAFT 4, Athena Technical Plan Section C.6, February 17, 1987

Introduction

Probably the single most difficult technical problem in the design of the Athena workstation/server model is the choice among various models of file storage. The tools available consist of modest-sized hard disks on both private and public workstations, large hard disks on central servers, at least three quite different remote file systems, a remote virtual disk system, two or more kinds of removable storage media, and a base starting point of the UNIX[1] file system. The combination of rapidly changing hardware technology and uneven advances in remote storage system technology[2] makes discussion of the alternatives extremely confusing; a requirement that hardware and operating costs be kept within bounds affordable by an educational institution dramatically constrains the possibilities.

Adding to the complexity is that there are several different uses of storage: private student storage, which quantity is multiplied by the number of students (5 to 10 thousand), storage for libraries of programs and data used by classes, storage for libraries used by everyone, and storage for users who work together intensely in various-sized groups, ranging from two students working on a laboratory project, through application development groups with half a dozen faculty and teaching assistants, to the headquarters development staff of 70.

This note discusses the following topics:

- Assumptions and requirements of the M.I.T. environment that affect the choices of a file system model.
- The properties of three overlapping styles of deployment of a remote file system: a single, uniform, centrally managed file hierarchy containing all files, a system consisting of a large number of independently managed file servers, and finally the use of lockers to provide a temporary emulation of private workstations during the time when public workstations are still the norm.

---

[1]UNIX is a trademark of AT&T Bell Laboratories

[2]This document uses the term *remote storage system* to encompass both remote file and remote disk systems. A *remote file system* provides network access to named files; a *remote disk system* provides network access to numbered disk tracks.

- An analysis of the problem of file interchange on removable media.
- A comparison of the properties of four available remote storage systems.
- A plan for action, based primarily on use of the Sun Network File System.

Assumptions and Requirements

Assumptions

- The long-term goal is that most workstations are privately owned by the student; public workstations will always be available but their current dominance is a temporary phenomenon related to their current high cost.
- There are immediately some needs for privately operated workstations, among faculty developers, certain living group experimental installations, and the staff. A subsidized switchover to privately managed workstations will probably begin in Fall, 1988.
- The number of student users will grow to about 10,000, as Athena expands to encompass graduate students as well as undergraduates.
- To be acceptable to M.I.T., the entire cost of educational computing must be less than about 10% of tuition, currently about $1200/year or $4800 over the four-year undergraduate experience. A very rough allocation of this cost might be as follows:

	per-student		aggregate	
cost category	$/year	total	$/year	%
Workstation purchase	$600	$2400	$6.0M	50%
Workstation sales cost	48	192	0.5M	4%
Workstation maintenance	120	480	1.2M	10%
Central hardware & its maintenance	110	440	1.1M	9%
Central administration & operations	110	440	1.1M	9%
Telecommunications	140	560	1.4M	12%
Printing and documentation (avg)	72	288	0.7M	6%
totals	$1200	$4800	$12.0M	100%

The amount budgeted above per year for central hardware and maintenance corresponds to a capital investment of about $3.7M, assuming a five-year average hardware lifetime and a contract maintenance cost of about 10% of list price per year. Note also that this item includes the cost of maintaining and replacing residual public workstations.

Central administration and operations includes software licensing, software development, system engineering, and software distribution for workstations, as well as general administration and service operation.

- There are four easily available remote storage systems for use with Berkeley UNIX: "Vast Integrated Computing Environment" (Vice) from the CMU Information Technology Center, the Network File System (NFS) from Sun

Microsystems, Inc., the Remote File System (RFS), a public domain system developed by Todd Brunhoff, and the Remote Virtual Disk (RVD) system from the M.I.T. Laboratory of Computer Science. Other remote storage systems, such as Apollo Domain, that provide similar function but are not yet available for the current crop of Athena workstations, are not discussed here.

Detachability Requirement

The file storage model must allow privately-owned workstations to be useful when not attached to the M.I.T. network. There are three distinct reasons for this requirement:

1. About 15% of undergraduate and 80% of graduate students live in apartments off the M.I.T. campus. The only hope for communications from those apartments to an on-line M.I.T. network in the next several years is dial-up access at no more than 2400 bits/second. Although graduate students generally have offices, when light-weight portable configurations become available, a common mode of use will be to carry a personal computer back and forth, leaving the less-movable display in the office and having a second, perhaps lower-performance display at home.

2. Students will want to take a personally-owned computer home in the summer. If we try to suggest that the student purchase a computer system that is not useful when detached, the student will ignore our suggestion and purchase one that is.

3. When the student graduates, the computer follows, to a home in a new city, a job, or graduate school. Some students may attempt to sell their used computers to incoming undergraduates, but the market for four-year old personal computer technology will probably continue to be relatively weak. Although there is often discussion of potential future provisions for graduates to remain somehow connected to M.I.T. (or for workplace environments to provide an equivalent place to plug in), such possibilities are uncertain, expensive, and not universal. The one certainty is that a large percentage of students will want their computer to continue to be useful after departure from Cambridge.

Usefulness when not attached to the M.I.T. network means that the basic operating system and most software packages are capable of operating without help from network services and network-accessible libraries. Most important to the discussion of the file storage model, the user's personal files must be available without network attachment. The attachment to the M.I.T. network must have the property that it is optional, augmenting the set of software, services, and data that are available.

Thus the detachability requirement eliminates as unworkable (except in special applications or as a temporary measure) any architecture or storage model that does not provide persistent files on the workstation.

Attachability Requirement

Despite the importance of the detachability requirement, it is equally important that network attachment significantly augment that workstation's usefulness. The augmented value includes

1. Ability to keep software and data stored on the workstation up to date.

2. Ability to draw information and programs from large on-campus libraries.

3. Ability to obtain current class assignment information.
4. Ability to turn in assignments electronically.
5. Access to electronic mail and electronic bulletin boards.
6. Ability to export private files to and import private files of other users.

In other words, the purpose of the network attachment is to provide various forms of information sharing that are otherwise absent when using a set of isolated workstations.

Interchangeability Requirement

Although they are almost identical, a UNIX file system created, for example, on an RT PC is not directly readable by a VAX, and vice-versa. The underlying reason is that the two architectures store objects other than character strings in different byte orders, and this difference is reflected in the stored structures and internal indexes of the file system. (In the future, several other hardware architectures are potential candidates as Athena workstations; although other implementations of UNIX might choose yet different byte orders, all of the examples encountered so far are the same as one of those two.) The difference in byte order becomes apparent in two cases in which the potential exists for one machine to read a file system created by another:

1. A remote storage system exported by a machine using one byte order is to be imported by one using the other byte order.
2. A diskette written at one kind of machine is moved to the other kind. Although 1.2 Megabyte diskettes used by the RT and the VAX are physically identical and are soft-formatted with identical sector layouts, standard UNIX file system software operating at the next higher level makes them non-interchangeable.

The primary interchangeability requirement surrounds personal files of students—in a public workstation environment it is essential that a student be able to write personal files with a workstation of one architecture, and read them later from other machines. This requirement applies both to removable media and also to files read and written using a remote storage service.

Even though a private workstation is of only one architecture, and therefore the interchangeability requirement may seem less important once private workstations predominate, the owner of a private workstation will use public workstations in laboratories and seminar rooms and will also want to export files that can be imported by other students. Thus interchangeability, though perhaps used more often initially, is a permanent requirement. In addition, the owner of a detached private workstation may find that removable media are the primary method of moving files from campus libraries to the workstation, in which case removable media interchangeability is an essential property.

System libraries contain primarily binary programs, which are not at all interchangeable among different machine architectures for reasons far deeper than byte order. Since there must be multiple copies of those libraries in any case, lack of interchange is not a serious limitation in that application. The same thing is true for some, but not all, class libraries. Maintaining multiple copies of a library is a significant additional burden when the library contains mostly text material, and is thus potentially useful to clients of several architectures.

Storage Quantity Requirement

The amount of storage that a student needs for personal files is a complex tradeoff among cost, ability to find things, effort required to keep it in order, speed of access, housecleaning, reliability, and archiveability. Different students will have different opinions on how they would like to make those tradeoffs.

The best way to allow students to make their own tradeoffs is for them to own the storage medium that holds their personal files. In a privately-owned workstation, the student owns the disk and can decide, for example, whether to purchase a large one, to get along with a small one by being more aggressive about housecleaning, or perhaps even to postpone disk purchase completely until departure from M.I.T. becomes imminent. In the (temporary) public workstation model, it is harder to see how to simulate this effect. One way is to allocate a small, fixed amount of storage for each student on a central storage system, and insist that removable media owned by the student absorb all overflow. Note that fixing the amount of storage has the side effect of minimizing the cost of administration.

Operation Cost Requirement

The storage model must keep the cost of central operations within a budget that students can afford. This requirement places strong limits on all central expenses that are proportional to the number of student users, to the number of personal files, to the amount of personal file reading/writing activity, or to the number of deployed workstations. These expenses include

1. Any centrally owned disk storage space used for private files.

2. Labor to manage centrally owned disk storage facilities. (E.g., allocation of space to new students or to people who need more, and housecleaning of storage of departed students.)

3. Labor to operate centrally owned disk storage facilities. (E.g., disk hardware maintenance and storage backup and recovery.)

4. Communications—gateway capacity to allow access to both private and centrally stored storage from any workstation.

Network Considerations

Network loading

The network is a high-performance pipeline that can get a large volume of data transmitted to a workstation quite quickly. The usability of this high performance is different for local transfers, between computers on the same local area network, as compared with long-distance transfers, that cross one or more gateways. The aggregate load that gateways of current technology can handle is quite limited, with the consequence that individual workstations must be sparing in the intensity with which they use long-distance connections.

The best way of expressing the loading limit of a gateway is in packets per second; current gateway technology can handle about 200 packets per second, a limit that is largely independent of the packet size. This limit compares with a raw Ethernet capability of 800 maximum-sized packets per second and over 10,000 minimum-sized packets per second. An RT PC or VS-II workstation running UNIX and using a remote storage system can with

ease generate about 40 packets per second of two-way traffic.

An important aspect of gateway and network loading is that overload leads to avalanche effects. Just as a city gridlock makes itself worse as cars caught in the gridlock run out of gas, when a network overloads and begins to introduce substantially more delay than usual, protocols designed for robustness begin retransmitting, which adds to the load. At the present level of understanding of network, gateway, and protocol design, the system architecture must be quite conservative in the area of overload probability.

Vulnerability to Network Outages

Experience so far with both local and cross-campus network connections is that one should design in anticipation of brief network outages. A conservative planning position is to assume that outages of one second may happen many times a day, of 20 seconds may happen once or twice during any given workstation session, and of an hour or more are rare but they do happen. Both the detailed design of a remote file access method and scenarios of use of remote files must take this anticipation into account.

Potential Models for File Access

There are several styles of deployment of remote storage systems. In each style, a server *exports* the contents of some directory, or a subdirectory hierarchy, making those contents available to clients. A client *imports* one or more such subdirectory hierarchies, perhaps from several different servers at the same time, and integrates those subdirectories into its own file system in such a way that applications programs need not know whether a particular file, directory, or link is local or remote.

The Single, Uniform Storage Hierarchy

One style of deployment of a remote file system is to create what appears to be a single giant hierarchy that contains all of the shared files of the community of users, classes, and libraries. This organization is very appealing for its uniformity and simplicity from the point of view of the user. The Vice file system designed by the Information Technology Center of Carnegie-Mellon University supports such a view, by allowing multiple, cooperating servers to export what appears to be a single mount point that every client attaches into a standard position high in its own file system hierarchy. The user has two choices for placing a newly-created file, either in the part of the hierarchy that is completely private to the workstation, or in the part of the hierarchy that holds the community file system. If placed in the community area, the file is protected by an access control list system.

Of the four remote storage systems under consideration, Vice is the only one that supports this model. Other file systems appear to allow such a model of usage, but as the scale grows they run into one of two limits: the amount of file storage that can be handled by a single server, or the number of cycles that a single server has available for file access activity. Either way, the only expansion method is to go to a second server, then a third, and a fourth, etc. But since there are no provisions in the other remote file systems for cooperation among servers, creating the illusion of a single giant hierarachy must be done by the client. One way would be to place different pieces of the file system hierarchy on different servers. Each client would then separately import the file systems of each of the multiple servers. The problem with this approach is that reallocation—for example, moving a library from an overflowing server to another that has more space—requires

changing the user-visible name of the reallocated information. As the scale grows to where ten or more servers are involved, it is apparent that the identity of the server, and thus the path name of any given library or user's directory, would have to be learned through the help of a nameserver, and the uniformity benefit of the single giant hierarchy would be largely lost.

There is some question as to the value of a single naming hierarchy that includes a region under which one finds 10,000 entries, one per user. In particular, the mode of browsing that consists of listing a directory to see what lies below would not be particularly useful in passing through that region of the naming hierarchy. Instead, the user would require mechanisms outside the naming hierarchy to find his or her way around. Once those outside mechanisms are brought into play, they could also create on demand a private subset or variation of the main naming hierarchy that the user might find easier to deal with.

The Multiple, Independent Server Model

An alternative to the giant central name space is is a model based on many private servers. The Network File System of Sun Microsystems provides a design that supports this model quite effectively. Suppose that every workstation operates an NFS server, exporting a part of its file system under a name listed in a nameserver. In addition, centrally managed library servers would export NFS file systems containing system libraries, and departmentally managed servers would export class libraries. A workstation acting as a client would identify just which things it wished to import and make private arrangements with each of the servers, with the aid of the nameserver. In doing so, it would build up a private naming hierarchy containing just those items of interest.

This model has several interesting properties:

- It can scale up very far, because the only centrally managed piece of information is the name server record of the exported file systems.
- Because the storage for private, exported data is privately owned, it permits the workstation owner unlimited ability to decide how much file space is devoted to shared information. Similarly, the instructor of a class has the ability to purchase an extra disk with department funds if the export of a larger data base is important educationally.
- It permits import on a completely private basis, unknown even to the name server.
- Access control is a private matter that can be determined entirely by the exporter.
- For information exported from a private workstation, since the central administration is not involved in the exportation it cannot be held liable for what is exported. All liability and responsibility for the proper operation of the bits, continued availability, reliability, certification of licensability, legality of sharing, etc., lies completely with the exporter. This decentralization of responsibility seems quite natural, especially in a university community.
- Because the user must explicitly import each different set of files or library, the user is explicitly aware of what remote things he or she is depending on.

It also has some disadvantages

- for user A to obtain user B's files, B must leave his or her workstation operating.

- Network traffic patterns are hard to predict, and if patterns surrounding privately exported file systems are undesirable they are hard to change.
- There is an explicit importation step required for each different user's files or library that a client wants to use.

Lockers

Evolution: Public Workstations and Lockers

Dominance of public workstations is a temporary phenomenon, until advanced-function workstations drop in price (or private ownership is temporarily subsidized) to the point that most students can afford them. Privately owned workstations will begin replacing public workstations gradually, and will eventually dominate the environment. The storage model should thus be designed for privately owned workstations, and the model used for public workstations should simulate as closely as possible the future environment.

The main mechanism that can be used to accomplish this simulation is the "locker," a per-user allocation of storage on a central file or disk server. A locker contains a student's personal files, making them available for that student's use from any public workstation that is located on the same local area network as the server containing the locker. Because overflow can be handled by removable media (e.g., 1.2 Mbyte diskette) the space allocated to a locker need be no more than that required to handle the student's currently active files. (Even a modest allocation of 2 Megabytes of storage per student will require 20 Gigabytes, about half the total storage currently available at the initial introduction of public workstations to undergraduates only. It is critical that lockers not still be required at the time that graduate students begin using the system.)

A major limitation in the concept of the locker is that it generates a substantial amount of network traffic. At the scale that Athena would deploy lockers it would be necessary to arrange that most use of lockers is by workstations on the same local area network as the server that holds the locker. That necessity in turn requires restrictions on where students find workstations, and careful placement of locker servers. (The alternative of developing server software that automatically moves locker contents to a server near the user is not very attractive; such movement itself generates network traffic and lockers are a temporary measure anyway.)

Note that the view that a locker is a simulation of the storage of a private workstation, rather than an allocation of storage in a community file system, simplifies the requirements on its function. (Simplification is especially interesting for a temporary mechanism.) In particular, backup can be assumed to be done by the owner, and it is not essential that anyone but the owner be able to obtain access to the locker's contents.

However, lack of access to the locker by anyone but the owner raises the question of how that owner can export files and data. A separate mechanism is required for the purpose. One mechanism that might be used in a private workstation, running a private file server on the workstation, is not available for the case of a centrally provided locker. There are at least four alternatives:

1. Declare that export of private files is a feature not available from lockers. This alternative has the virtue that it adds to the attractiveness of private ownership of the workstation.

2. Provide a separate mechanism for file export, e.g., a small allocation of space in a community-wide shared file system. That space allocation would have to come

at the expense of the locker allocation, it increases the work required to do space allocations, and any one allocation strategy would not be satisfactory for all students.

3. Implement lockers as directories in a community-wide shared file system. This approach has the drawback that it does not simulate the future workstation environment; it provides a level of sharing that will disappear when the student moves to a private workstation.

4. Implement lockers as separately exportable subhierarchies on central servers. This approach comes closest to simulating the future environment; the key difference is that (without development) a user cannot export only part of his or her files; the entire namespace of the locker is visible to any importer. (There are two effects of this visibility: the user must be careful to set protection bits correctly throughout the entire subhierarchy in the locker, and importers must use pathnames describing the entire personal hierarchy of the exporter, not just the part the exporter intended the importer to use.)

The first and last alternatives are the most attractive; evaluation of the performance and administrability of the available remote file systems is required to determine whether or not the last alternative is actually feasible.

Locker Space Allocation

One interesting question is whether or not one can overallocate locker space. There are three reasons why overallocation is probably a bad idea. First, overallocation requires that there be a some procedure available to deal with the case when a server runs out of real space. That procedure should not involve human intervention (e.g., to decide where to move users from the crowded server); the economic and operational constraints require that we design this kind of human intervention out of the system wherever possible. The alternative of automated coordination of allocation on multiple servers is a feature not currently available on any remote file system, apparently because it is very difficult to do. Of course, one can overallocate only a little, and thereby hope to make the occurrence of overuse so rare that it does not present a problem. But in that case there is little point in overallocating at all.

The second problem with overallocation of lockers is that it presents the student user with a storage model that is quite different from that of private workstation storage. On a private workstation, one can depend on the physical space actually being there whenever it is demanded. One need not be prepared for the possibility of running out of space because of the activity of other users. From a different perspective, the overallocation paradigm works well on time-sharing systems because one often can find someone else's temporary storage to trim back, either by oneself or with the help of an administrator. The entire model is quite different from that of private workstation storage, in which the only files one can reclaim are one's own, and when a file is discarded, the space isn't snapped up by someone else.

The third problem with overallocation is the following: A property of a locker is that it cannot hold all the student's files; overflow is onto removable media. In consequence, it will be the normal case that many students are near the limit of their space allocations and are frequently doing housecleaning to keep from running into their limit. Thus the amount of overallocation that one can safely accomplish is less than one might expect.

With these three considerations in mind, it seems clear that if lockers are used as a

simulation of private workstation storage, they should be allocated on a 100% usage basis.

Long-term Use of Lockers

An open question is whether or not some continued use of lockers might be appropriate even after private workstations dominate the scene. Since there will continue to be public workstations, for example in laboratory settings and libraries, some provision is required for a student working in one of those locations to store a few files in a place where they can also be accessible from the private workstation. The locker could meet this need, though a distinctly different implementation might suffice. The reason is that the locker would not carry the burden of being the primary storage medium for most personal files. As a result, both server and network loading would probably not be significant, and the locker allocation could be small.

An alternative approach would be for the private workstation to run a remote file system service, exporting all or a portion of its file system for use by the owner, but from a public workstation in a laboratory or library. That approach has the advantage that it minimizes the need for the student to plan what might be needed and it eliminates the small, fixed limit that the locker enforces on file storage.

File Interchange and Removable Media

As mentioned earlier, moving data among workstations of different vendors is an important goal. Removable media are one method of moving data, useful primarily in cases where the user is forced to work in isolation from the campus network, for example in a private apartment. Removable media add an additional interchangeability complication, beyond byte-order and binary architecture dependence: removable media themselves come in many forms, and imposing a standard form across several different vendors is unlikely possibility, especially over any substantial length of time.

Temporarily, however, there is an opportunity that might be of some use: both of Athena's primary vendors can supply diskette drives that read identical 1.2 Mbyte diskettes. This opportunity will probably vanish at about the same time that private workstations become viable, because new, lower-priced workstations are likely to come with various new removable media options.

Short-Term Solutions for Removable Media Interchange

Even while the opportunity exists, workstations with different byte orders can make these hardware-compatible diskettes software incompatible. This section offers several different solutions to the problem of overcoming byte-order differences on hardware-compatible diskettes, during the period that public workstations dominate the scene. It is assumed that in a private workstation environment interchange reduces to providing a few public examples of every popular removable medium. Non-resident students would use these public facilities to move files to and from their on-campus locker.

Solution A. Use standard UNIX file systems on the diskettes and have students use separate diskettes for the two byte orders.

Advantages. . .

1. Procedure is simple to describe and understand.
2. No development effort required.

Disadvantages. . .

1. This solution ducks the problem, rather than solving it.

2. To move files from one kind of file system to another one must find two free workstations of different architectures, log in to both, mount the two file systems, and copy files back and forth. (It may be possible to cascade NFS and RVD to achieve the same effect more elegantly.)

3. The student ends up managing two sets of diskettes, and will often find that a file is on the wrong kind of diskette, or if a copy is kept on both kinds of diskettes, a change made to one of the copies didn't get made in the other copy.

4. Applies only to the public workstation environment.

Solution B. Use standard UNIX file systems on the diskettes and arrange that every public workstation cluster, living group, and class or seminar room have a 50-50 mix of the two kinds of workstations, and tell students to always use the same kind of workstation.

Advantages. . .

1. No development effort required.

2. Procedure is simple to describe and understand.

Disadvantages. . .

1. This solution ducks the problem, rather than solving it.

2. Requires replanning present clusters.

3. Breakage: when students don't happen to split evenly, there will be a shortage of one kind of workstation; the numbers deployed in each cluster would have to be increased slightly to compensate.

4. Some application programs run only on one kind of workstation, leaving students who have diskettes of the other format out in the cold.

5. Applies only to the public workstation environment.

Solution C. Use a PC/DOS-format file system on diskettes

Advantages. . .

1. Permits complete interchange of text files among machines of different byte order.

2. Interchange with AT-class PC-compatibles becomes available.

3. Format is very robust—if the user changes diskettes without informing the operating system, the contents of the diskette probably won't be damaged.

4. Code is available off-the-shelf with RT PC.

Disadvantages. . .

1. Information stored on a diskette is not part of a mounted file system, so files must be explicitly moved from the diskette to the fixed disk of the workstation before use.

2. The DOS file system format limits file names to 8 upper-case characters plus a three-character name-extension; other UNIX file properties such as permission

bits are lost.

3. Permission is needed to use the RT PC code on the VAX.

4. May require development of a file system salvage and repair program that runs under UNIX. (A version of DOS program CHKDSK.COM.)

Solution D. Modify the DOS file system code that came with the RT PC to allow it to handle UNIX standard file names.

Advantages (as compared with solution C). . .

1. The name of a file on the diskette can be identical to the name it has when it is moved onto the workstation.

Disadvantages as compared with solution C. . .

1. Probably requires substantial development effort to allow variable-length names, and substantial checkout effort to verify that robustness is still maintained.

2. Ability to exchange files with AT-class machines is lost.

Solution E. Develop a program that does in-place conversion of an unmounted RT-format UNIX file system into a VAX-format UNIX file system and vice-versa. The UNIX mount command can be modified to detect attempts to mount a file system that has the wrong format and suggest that the user run the conversion program.

Advantages. . .

1. The resulting file system can be mounted in the storage hierarchy of the workstation and files on the diskette can be read and written directly by application programs. Complete exchange of text files is possible.

Disadvantages. . .

1. The resulting file system is quite fragile; the user must be careful to shut down and "umount" the file system before removing a diskette.

2. The available tools for file system repair require a high level of wizardry.

3. Significant development effort is required.

4. Start-up overhead: all directory blocks of a file system must be rewritten before the file system can be used.

5. If a failure occurs during conversion, the resulting file system is probably irretreivably damaged, and all the files in it are lost.

Solution F. Modify the UNIX file system to allow disk-stored file systems that are of either format.

Advantages. . .

1. The resulting file system can be mounted in the storage hierarchy of the workstation, and the diskette can be read and written directly by application programs. Complete exchange of text files is possible.

Disadvantages. . .

1. Major development effort, and has to be integrated into at least two different

kernels.

2. The Athena kernel becomes significantly different from the standard UNIX kernels of the vendors, leaving Athena with a long-term maintenance problem unless they can be convinced to pick up this change.

3. The resulting file system is quite fragile; the user must be careful to shut down and "umount" the file system before removing the diskette.

4. The available tools for file system repair require a high level of wizardry.

Because this solution requires duplicating the interchange code found in some remote file systems, it is possible that this solution could be simplified by adapting that code.

Solution G. Ignore the interchange problem for removable media, on the basis that students can accomplish interchange by copying files in and out of a locker in a network-based remote file system that provides interchange.

Advantages. . .

1. No implementation effort beyond that already required to provide some kind of central storage service.

2. Relatively easy to document and explain.

Disadvantages. . .

1. Meeting the detachability requirement is awkward. A student with a private workstation located in an apartment must go through an extra step to bring home a file from a classroom that uses workstations with incompatible media or wrong byte-order: locate a public workstation of the same architecture and media as the one at home, attach the locker to it, and copy the file onto a compatible diskette.

A Comparison of Available Remote File Systems

This section compares the four remote storage systems already mentioned: the Carnegie-Mellon storage system (Vice), the Sun Network File System (NFS), Brunhoff's Remote File System (RFS), and the M.I.T. Remote Virtual Disk System (RVD).

Since "performance" is often mentioned as a point of difference among different designs and different implementations, it is important to recognize that a remote storage system can be engineered to "maximize performance" in two distinctly different ways:

1. To minimize the load on the server of each client request.

2. To minimize file access delays for the client.

The first approach allows a large number of clients to make use of each server; the design of RVD is optimized to minimize the server load. The client sees a remote virtual disk pack as having about half the performance (both seek time and data transfer rate) of a local disk. However, an RVD server can handle a lot of simultaneous clients.

The second approach allows the client to be indifferent as to whether any given file is local or remote. The design of Sun NFS is optimized to minimize impact on the client. Sun reports that the client can expect an overall system performance degradation of under 5% when using remote files as compared with a local disk. However, the NFS server can handle only a few simultaneously active clients.

Properties of the Available Remote Storage Systems

- VICE: When a file is touched for the first time a complete copy moves to the workstation. Vice allows transparent cooperation of many servers; it is designed to scale up to many thousands of clients. It requires all servers in a cooperating set to trust one another. There is a small limit (3) on the number of independent Vice services that can be imported simultaneously. (Implication: all storage under vice files must be centrally owned). Vice is owned by IBM; licensing is presumed to be available, although terms and conditions have not yet been announced. In addition to Athena standard workstations, a client implementation is available for the IBM PC. Support is currently by CMU/ITC, and is anticipated to be by IBM/ACIS. Tools are available for monitoring and tuning performance by moving files and directories among cooperating servers. An evaluation copy is available. Performance from the point of view of the client is not known; the server load per client is moderately high. (CMU claims 20 to 50 clients per server.) Management and operation effort required to run it are not yet known. Requires integration with the Project Athena authentication system, Kerberos. (CMU/ITC has already designed an integration strategy for a similar key distribution system.) Provides an access control list system.

- NFS: Sun's Network File System is supported both by Sun and by Digital in Ultrix 2.0[3]. It has been adopted by about 100 vendors. A port for the RT is available from Brown University. A client implementation is available for the IBM PC. It does not provide for cooperating servers; multiple servers require multiple importation actions. A private workstation can be an NFS server and can export a subset of its files. Its license is owned by SUN; a binary education license is available with Ultrix 2.0, but the source license has proprietary information hassles. Performance is high from the point of view of the client, but the server load per client is also high. NFS requires some work to integrate with the Kerberos authentication system; Sun has already designed an integration strategy for a similar key distribution system. NFS depends on Unix permission bits for access control.

- RFS(Brunhoff): This is an unsupported public domain package that requires no license. Most of its technical properties are quite similar to NFS. It is easily installable in new system releases. Integration with the Kerberos authentication system has already been accomplished, and an experimental version is in use at Athena. Performance is weak both from point of view of client and in load on server; better data is needed. RFS depends on Unix permission bits for access control after the user is Kerberos-authenticated.

- Common properties: All three systems provide a byte-order-independent network representation of data, so that any machine for which an implementation exists can use files and directories created by any other. All are available for VAX, RT PC, and Sun Architectures under 4.2/4.3 BSD Unix, Ultrix, and ACIS 4.2a.

---

[3]Ultrix is a trademark of Digital Equipment Corporation

The Remote Virtual Disk System

The Remote Virtual Disk system (RVD) provides a remote access system at the disk driver level of the operating system, rather than at the file system level. It places a very low load on servers and a moderately low load on the network. It allows sharing only of read-only information, not writeable information. When used to hold UNIX file systems it does not permit interchange of information between machines with different byte orders. (However, because it provides a raw disk interface, it can be used with any storage-organizing system, for example a Unix tar stream, or a DOS file system.) Its combination of low server load and ability to share read-only information make it an effective technique for mass distribution of widely-used libraries. Its lack of ability to share writeable information makes it less useful for other applications. Lack of interchange across machine types impedes its usefulness for student lockers.

Athena Requirements Compared with Capabilities

Table I summarizes the requirements that Athena places on remote storage systems, and for each requirement, those systems that currently meet it.

In that table, note that every available alternative misses meeting several of these requirements. Of the four systems, NFS comes closest, and if an acceptable source language agreement could be negotiated, most of the remaining requirements (e.g., Kerberos, SMS integration, and perhaps access control on export points) could be added locally.

Recommended directions

In light of all these considerations, this section makes a concrete set of proposals, for purposes of getting at least one consistent set of plans on the table for discussion. The proposal is based on the assumption that an NFS source license can be negotiated, and that the modifications suggested at the end of the previous section are made. The effort required to do all the things suggested here is substantial—in some cases unknown—and other development projects may have to be deferred in order to proceed in this area. In addition, although some transition suggestions appear here, much more transition planning is required.

System Libraries

Use the remote virtual disk system to deliver central system libraries, and as a way of extending the scope of a single NFS server, as outlined in the section on Staff Storage, below. Its properties are well understood, it provides low server load per client, minimum administration is involved because the shape and number of system libraries changes slowly, and there is little extra burden caused by the need to maintain different library versions for machines of different byte order because different binary program copies are needed for each workstation architecture, anyway.

In the future, if a remote file system proves to have adequately small server loading, it would be appropriate to switch over to use that system also for the system libraries, so as to reduce the number of different systems deployed. But the urgency of such a change is low because the payoff is small.

Table I: Comparison of Remote Storage System Properties

Requirement	*Vice*	*NFS*	*RFS*	*RVD*
Allows directory and text file interchange across machine architectures.	•	•	•	
Allows 10,000 clients (not required to be all on one server)	•	•	•	•
Allows many, mutually untrusted exporters (up to 10,000).		•	•	•
Allows import from many exporters at the same time.		•	•	•
Allows private ownership of disk space containing files to be exported.		•	•	•
Source license available on terms acceptable to M.I.T.	?	•	•	•
Provides Kerberos integration.		†	•	•
Controls export authorization with access control lists.	•	†		•
Integrates with Athena Service Management System for setup of exported file system configuration tables and allocation assignments on centrally managed servers.	?	†	†	•
Operable with minimal labor.		•	•	•
High server performance for centrally managed servers used for system and class libraries.				•
Client performance loss for library access is small.		•		•
Basic system has vendor support.	?	•		

• Meets requirement

† Not too hard to add

? Not yet determined

Class Libraries

Convert three of the present time-sharing systems to be NFS servers, and place all low-traffic class libraries on one of those three systems. Each class library would be a separate export point. Use the name server (Hesiod, described in detail elsewhere in the Athena Technical Plan) to discover which server has which library. Allocation and management is by hand.

Faculty development projects would also be assigned storage space on those same servers used for export of class libraries; typically an exported class library would actually be a subdirectory of the associated development project. Whether or not backup should be run on these NFS servers is an open question. In the longer run, when Athena is no longer subsidizing faculty development projects, both development and export of class libraries would more appropriately be done using resources owned by individual academic deparments.

Place high-traffic libraries (in multiple copies, one for each architecture) on one (or more) of the RVD servers. Management is by hand.

Student Locker Storage

The plan for providing interim lockers is based on availability of a large number (to get enough server cycles) of the Phase I VAX 11/750 time-sharing systems. Convert about 20 of these systems to NFS service for lockers. Since each system has about 500 Mbytes of disk storage, each would hold about 250 2-Mbyte student lockers, for a total of 5000 lockers, about the right starting number.

Make lockers individually exported file systems. Rely on a combination of a small locker and removable media to get us through the public workstation period; when private workstations become available the local storage of those workstations can reduce the removable media from a prime storage place to its more appropriate function as an archive and backup mechanism.

By using RVD as the storage mechanism for system libraries and having separate NFS and RVD servers for class libraries, much of the potential remote access traffic that student workstations generate can be absorbed. But in order for NFS to be usable as a locker storage medium, it will also be necessary to develop a style of use that minimizes traffic to lockers, so as to prevent overloading both servers and gateways. Coming up with such a style of workstation use is a major challenge, for which few ideas are apparent. Here are some to start with:

- Switch to a mail system that is less intense in its use of the file system than is RAND mh. Use one that brings current mail into virtual memory and leaves it there for the duration of the mail-reading session.
- Make the user's home working directory a directory on the workstation rather than a directory on the remote storage system. Encourage use of the home working directory as an initial home for all files created and modified during the working session. Treat the remote directory as a place to get previous work and a place to save current work at the end of the session.
- Provide an automatic feature at logout time that checks for files in the temporary directory created for the user and suggests saving them.
- Encourage application developers to use the home working directory, rather than the current working directory, as a place to create temporary files.

Staff Storage

The development staff requires a very large community file system that includes automatic backup. Because there is available in the staff cluster both a high-performance VAX 11/785 processor and a second Ethernet, it is possible to simulate a very large community file system using NFS cascaded with RVD. The plan is to convert the VAX 11/785 system to be an NFS server that is attached to both Ethernets, and convert the remaining staff time-sharing machines on the second Ethernet to be RVD servers that have as their only client the staff NFS server. (Under this plan, the second Ethernet is used almost exclusively for the secondary, high-intensity RVD traffic between the NFS server and its supporting RVD servers.) Create a single hierarchy on this NFS server that contains all the storage of the staff 750 cluster, and export that entire hierarchy as a single file system. Place all sources and all project files there. Allocate for each staff member a modest amount of space for personal files. Run backup on the NFS server, which has the effect of running backup on the entire set of machines. Export of the staff community file

system is restricted to E40, so as to honor source license agreements.

This model of a cascaded NFS/RVD server may also be useful in other specialized contexts; the prerequisites are a high-performance processor with enough cycles to act both as NFS server and RVD client, and a private local area network for the secondary network traffic that the cascaded remote system architecture produces. Use in the staff cluster should be considered a trial intended to verify the feasibility of such a cascaded architecture. The main objection to this organization is that the exporting machine becomes a critical resource; when it is down all files in the community hierarchy are inaccessible.

Removable Media

Now that 1.2 Mbyte diskette drives are available on all Athena workstations, standardize on 1.2 Mbyte diskettes as the preferred removable storage medium, and retrofit machines located in public places that currently have other removable media, leaving one or two streaming tape drives available in each cluster for larger-volume activities.

Encourage users of public workstations to identify one kind of workstation architecture as preferred, based, for example, on availability in their living areas, and to format their own diskettes for this architecture. Depend on the primary burden of file interchange to be carried by the locker system. Provide enough public workstations of both byte orders that an apartment dweller can easily find a place to mount diskettes for the purpose of moving files to and from a centrally-provided locker.

Occasional conversion of files from one kind of diskette to another can be accomplished by finding two workstations within walking distance of each other, one of each byte order. The procedure is to place the diskette on the machine for which it is intended, and mount that diskette inside a file system that is being exported via NFS. The workstation of the other byte order can then import the file system and copy the files from the remote diskette onto a locally mounted diskette.

During the initial period when lockers in a remote file system that provides interchange are not yet available, use DOS format file systems on diskettes, obtaining permission from IBM to use their dosread/doswrite programs throughout Athena.

Roadblocks and Unknowns

The primary show-stopper that might lie in the way of executing the plan outlined above is that the server performance level is unknown for any of the remote storage systems except RVD; if a typical workstation user places too great a load on the gateways and the remote file server then it may simply be impractical to proceed with this plan. On the other hand, if a typical user only lightly loads the gateways and a server, it may be possible to relax some of the constraints. For this reason, the first step in following this plan must be to do some performance measurements. Now that evaluation versions of all the candidate remote storage systems are available those measurements are a high priority.

For the purpose of estimating the number of servers required, the required measurements are not sophisticated. There are two parts to the measurement.

1. Define three or four standard "typical transactions", such as open/write/close a one Kbyte file, open/read/close 50 Kbyte file, and write a 2K block. Set up an NFS and an RVD server, each with enough clients continually running one of these standard transactions as to load the server beyond its capacity. When fully

loaded, count the number of such transactions per second that the server can handle. A realistic load on the server, one that does not introduce excessive queueing delays and variability on the clients, is then probably somewhere in the vicinity of half that number. Also, count the number of packets that each standard typical transaction generates.

2. Set up a workstation environment roughly like the one envisioned (that is, a home directory with a quota of one or two megabytes), have several students use this environment for real homework, and measure the read and write traffic they generate to that home directory, to class libraries, and to system libraries. This measurement should characterizing each file system interaction in terms of one of the "typical transactions". From this establish a range on the number of typical transactions per second that a student generates during a typical login session.

Although any such measurement activity would be crude, would not characterize all possible modes of use, and would require a lot of interpretation, it should provide enough guidance for a reality check on the overall plan. If it appears feasible to proceed, better approximations can be determined later after some real deployment experience is gained.

Dissenting Views

Initial discussion of this proposal has identified two significant dissenting positions, both of which would build the same architecture as in the proposal above, but which would operate that architecture with quite different policies:

Locker allocation

This view holds that a much larger per-student central storage allocation is required to adequately simulate the private workstation, which would be expected to have 15 Mbytes or more of free space. One way to obtain this larger allocation is to purchase more centrally-managed disk space; a second is to overallocate the already available central disk space; a third is to allow more flexibility in allocation, giving students with a larger need a larger allocation, those without the need a smaller allocation. A combination of the three approaches could be used. All three methods would require a budget reallocation to cover the increased cost of providing more storage.

A related position argues that a large locker should continue to be a standard feature even during and after the transition to private workstations. That position is primarily a prelude to the next dissenting position, on automatic backup.

Backup

This view argues that the budget should be rearranged to include the cost of automatic backup of central file storage, on the basis that users would value this use of the budget more than other possible uses. The proposal outlined in this document does not require any change to run backup on the central NFS servers, but doing so would probably imply that lockers would become a permanent fixture rather than a temporary way of emulating private workstations in a public workstation environment.

Acknowledgements

This document benefited from extensive discussions, comments, and observations contributed by S. R. Lerman, W. D. Cattey, W. E. Sommerfeld, J. I. Schiller, N. G. Mendelsohn, D. E. Geer, and G. A. Champine, among others.