INFORMATION PROCESSING 86, H.-J. Kugler (ed.)
Elsevier Science Publishers B.V. (North-Holland)
© IFIP, 1986

ANDREW: CARNEGIE MELLON'S COMPUTING SYSTEM

James H. MORRIS, John LEONG, David NICHOLS, Michael J. WEST, Mahadev SATYANARAYANAN
Information Technology Center, Carnegie Mellon
Pittsburgh, Pennsylvania 15217, U.S.A.

The Information Technology Center, a collaborative effort between IBM and Carnegie Mellon, is in the process of creating Andrew, a prototype computing and communication system for universities. It will be one of the largest distributed personal computing systems of its kind. An earlier paper [11] describes the system in general. This one focuses on our experience with the network and file system.

1. Background

Carnegie Mellon is a relatively small university located on a physically compact campus, with a long-standing record of excellence in computer science research. In October 1982, Carnegie Mellon and IBM signed a contract that created the Information Technology Center. The ITC was to consist of about 30 people, ten of whom would be IBM employees on assignment. This organization was to design and develop a system to support the university's needs, based partly upon IBM hardware. This system has been named *Andrew*, after two benefactors of the university, Andrew Carnegie and Andrew Mellon.

How will this project affect university education? Here are four things that we are working to facilitate:

Computer Aided Instruction.
For many years some pioneering institutions have used computers to deliver instruction. Perhaps the largest effort is the Plato Project [15] at the University of Illinois where over 10,000 hours of course material have been developed. We expect the presence of ubiquitous and powerful graphics workstations to greatly stimulate efforts to develop computer-aided instruction. Producing computer-animated demonstrations is a new art, and the university is well-poised to develop it.

Creation and use of new tools.
A research university like Carnegie Mellon is devoted as much to changing the methods of professional work as to teaching them. Virtually every department pursues work in developing computer tools that will alter their professional areas. Beyond the expected activity in the engineering and scientific areas, there are developments in writing, historical research, social science, music, painting, psychology, and many other fields. A professor's research on a new tool can be aided by teaching it to students. They form a good set of users who have few built-in biases about how particular problems are to be approached. Sometimes they even help create the new tools.

Communication.
The use of electronic mail can have a profound effect upon a community that uses it extensively [5, 16]. With every member of a university community plugged into a smoothly operating communication system one can expect far-reaching effects. Class discussions held on a computer bulletin board will last longer, involve more participants, and allow time for more reflection and analysis. The use of graphics, formula manipulation, and automatic typesetting can improve the form and, indirectly, the content of much of our communication. We don't expect computer-mediated communication to supplant the more traditional sorts but to broaden and deepen the community's ability to communicate.

Information access
A mark of tomorrow's professional will be the ability to navigate in large information repositories available for problem solving. A university's primary data base is its library of books and journals; our communication system will provide better access to it. In addition, access to the growing number of world-wide data bases will be provided. Finally, we expect universities to develop extensive new data bases devoted to particular courses and areas of specialization.

A recent paper from M.I.T.'s Project Athena [2] presents a more extensive catalog of possibilities.

2. Basic Decisions

The Andrew system is an attempt to accelerate the delivery of computing technology to a large university community. It is based upon four key components of that technology: personal computers, raster graphics, local area networks (LANs), and time-sharing file systems. The Xerox Alto System [6] has been a powerful inspiration for our system.

The computing paradigm envisioned in Andrew is a marriage between personal computing and timesharing. It incorporates the flexibility and visually rich user-machine interface made possible by the former, with the the ease of communication and information sharing characteristic of the latter. This model is depicted in Figure 1. The large amoeba-like structure in the middle, called VICE, is a collection of communication and computational resources serving as the information sharing

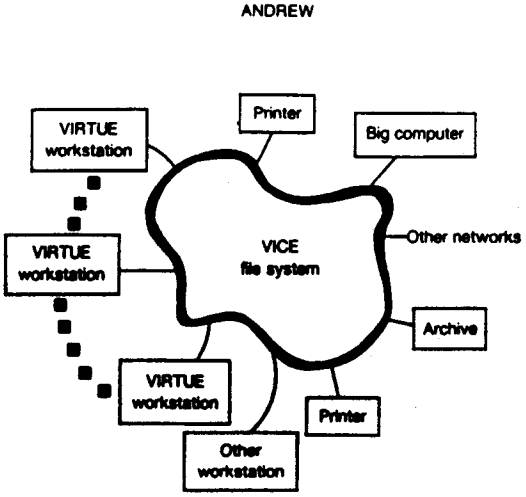

Figure 1: Andrew = VICE + VIRTUE

backbone of a user community. Individual workstations, called VIRTUEs, are attached to VICE and provide users with the computational cycles needed for actual work as well the overhead of a sophisticated user-machine interface.[1]

2.1. Workstation Hardware

The model we chose for our personal computer was much more powerful than any existing ones. The Carnegie Mellon Computer Science Department had created a carefully thought-out machine specification for its Spice Project [3]. It included such advanced requirements as a million-instruction per second (MIP) processor, a bit-map display with a million pixels, virtual memory, a megabyte of real memory, a mouse, and a LAN connection. We also decided to depend upon a dedicated, high-speed disk memory for each machine, although it is possible to use network-based disk servers to supply such memory. The Spice proposal predicted, accurately, that such machines, now called engineering workstations, could be had for $10,000 in the mid-eighties; so it seemed that subsequent price reductions would make it an affordable machine for students a few years later.

The primary machine that Andrew runs on is the IBM RT PC, a 32-bit RISC architecture machine inspired by the 801 project done in IBM's research division. It exceeds the original Spice machine specifications in all dimensions except the display which is slightly smaller, 800K pixels. Andrew also runs on SUN workstations and DEC VAXstations.

2.2. Operating System

We chose to use the Berkeley UNIX[2] operating system [13] for our development. It was a well-defined standard, had several advanced features, and was well-liked by many of developers in the university environment. Most important it was portable; this made it possible to develop Andrew on one machine and move it to others. In retrospect, this decision has been good in that it has allowed rapid development and gives our system access to a good variety of workstations. However, it has made our system rather large and expensive relative to the goal of providing it on inexpensive student workstations. Our hope is that the rising tide of hardware technology will solve this problem if we can avoid raising our ambitions along with it.

2.3. Communications Paradigm

An academic environment requires a large amount of information sharing. A high degree of user mobility between dormitories, faculty offices, libraries, and laboratories is also essential. Thus it was important to choose the central communications model carefully.

The simplest paradigm is the telephone: point-to-point, real-time connections. While this is obviously necessary, we felt that we should be more ambitious and support asynchronous communication among users. The simplest form of asynchronous communication is electronic mail. We felt that the mail paradigm was also too restrictive in that it didn't support browsing of passive data bases by students and researchers.

We chose a time-sharing file system as our communications paradigm. Each workstation appears to share a single large file system. This is easy to grasp and has clear advantages over the less ambitious models. We considered technically more ambitious paradigms such as network operating systems and distributed data base system, but rejected them. We reasoned that the pace and content of communication at a university does not require the complex structure they imply. Most of the information used by a university is in the form of papers, memo, and computer programs. Thus, it seemed acceptable to make the unit of sharable information a file and to record changes centrally whenever a file changes.

VICE provides a common name space for files. Users may thus access files in a uniform manner regardless of the specific workstations at which they are logged in, or at which the files were created originally. Most other shared facilities, such as mail, bulletin boards, and printing, can be built on top of VICE, obviating the need for machine or location-specific information. Software in each VIRTUE workstation makes these facilities of VICE appear as a transparent extension of that workstation's operating system.

[1] VICE stands for "Vast, Integrated Communications Environment." VIRTUE, for Virtue is reached through UNIX and EMACS."

[2] UNIX is a trademark of AT&T.

2.4. High-Bandwidth Communications

Traditionally, most communication traffic is explicitly generated by users. The most common application is host access from terminals, followed by file transfer and mail. These types of applications can normally be satisfied to a large extent by the standard serial line network especially if it operates at the relatively high speed of 9.6K or 19.2K bits per second. Even when a user requests the transfer of a large file, the delay is generally tolerable since the transfer is explicitly requested and the user is psychologically primed to wait. For those infrequent occasions when the files to be transferred are huge, the traditional approach of physical transfer by magnetic tapes has proved to be reasonably acceptable in most cases.

In the past few years, however, new applications profiles have emerged. The most noticeable area is in the development of network-based shared resource systems. In that mode of operation, a number of hosts or workstations share a set of common resources -- primarily disk space. In this profile, the remote access of the resource is system initiated and as such, is done behind the back of the user. Any delay will show up to the user as irritating fluctuation in performance. Carnegie Mellon's relatively small campus and localized student residences allowed us to chose high-bandwidth communications via local area networks (LANs) as the normal mode of operation. Depending on the type of coupling between the workstation and the resource provider, the communication requirement can range from very severe to moderate. We have applications falling into both categories: diskless workstations paging across the Ethernet to IBM PC's accessing remote file servers

2.5. System Components

Figure 2 shows the major components of Andrew. This system structure was chosen to allow multiple options to be tried for each component, either simultaneously or through time.

> Many workstations can support UNIX.
>
> There are no hard dependencies between the file system and the user interface; each has been used without the other.
>
> Multiple network implementations are possible.

The ITC development effort has focussed on three of these components: network communication, the shared file system, and the user interface. We examine the first two of these in the remainder of this paper. The user interface is discussed elsewhere [11, 4].

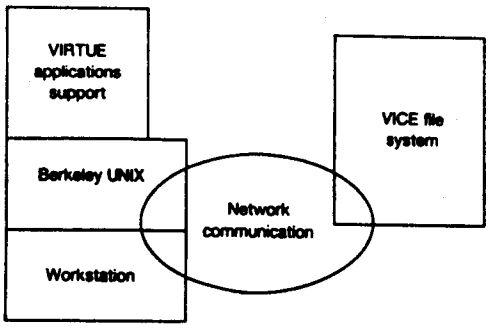

Figure 2: The components of Andrew

3. Network Communication

3.1. Local Area Networks

Carnegie Mellon has been using a number of LAN technologies during recent years. The Ethernet [9] is a relatively mature technology and has established itself in the scientific and engineering community. Among the IEEE802 standards, it has the strongest followings today and has been adopted by many companies. Various departments, at their own initiative and funding, have installed Ethernets. Most of this work was not co-ordinated centrally. Currently, there are 15 separate Ethernets with over 20 segments on the campus. More than 600 high performance work stations and hosts of various types are attached to those networks.

In general, we have had very favorable experience with Ethernet technology and have developed a significant amount of operational expertise across the campus [7]. One thing we have learned along the way is the need for good cable plant design. The earlier networks were installed and expanded on an *ad hoc* basis with little long term planning. Maintainability was usually overlooked in favor of expedient installation. Stations were attached to the most convenient or nearest location of the Ethernet trunk -- often without any logging of the event. When problems arise, trouble shooting can be a tedious process. The simple problem of complete net outage due to shorting caused by improper transceiver installation can be determined relatively easily with the help of a time domain reflectometer (TDR). However, unless the trunk cable is carefully labeled, trying to determine the location where a certain distance from the TDR point for a given cable that meanders all over a building can be very difficult. A more common and trickier problem is the substantial performance degradation of a network caused by faulty Ethernet controllers. Typically, the controller will fail to listen before talking or transmit garbage packets frequently -- but not continuously enough for the jabber controller of a transceiver to

shut it off. Problem determination involves detaching stations in a trial and error manner until the culprit is found. This is sometimes difficult when the stations are locked up in offices. The alternative is to detach the station from the transceiver. The difficulty, in this case, is a function of the number of transceivers involved and where they are located. For a network with over 50 stations, this can be very time consuming. We have come to the conclusion that while the bus topology has the advantage of ease of installation and reduced drop cable length, the star topology is substantially better from an operational and maintenance point of view. Problem determination, in that case, can be carried out at the small number of star hub locations. Given the typical lifetime of a network, the star configuration is well worth the small initial investment of longer cable runs. As a matter of fact, we have recently reconfigured a very high traffic Ethernet with over 100 stations from bus to star topology using DELNI multiport transceivers from DEC. This has reduced our typical trouble isolation and recovery time from hours to minutes.

IBM recently introduced a star-shaped token ring local area network [17]. While this network has a data rate of only 4 megabits compared with Ethernet's 10, this is not a big disadvantage. From the measurement we have done on the Ethernets, we have rarely seen network utilization in excess of 15%-- even for networks with a lot of stations relying on network file servers as well as disk servers for paging. Another potential problem of the token ring is the fact that it requires active elements in every station. This increases the potential number of points of failure. On the other hand, the token ring design has extensive operational and maintenance features built into its hardware components. On balance, it is a good networking technology particularly for large scale LAN deployment. Furthermore, because it is topologically easier to configure than Ethernet, which is subject to more severe distance constraints, the token ring is more suitable for use as the backbone inter-building network. Two rings with roughly 50 RT PC's and PC's have been operational since the beginning of 1986. It has been very reliable so far. We will be increasing the deployment of this technology in the coming years.

In summary, from an operational and maintenance point of view, we would like to see only one LAN technology on campus; in practice, we will be required to support multiple types of LAN's. Currently, this means Ethernet, IBM token ring and AppleTalk.

3.2. Communication Protocols

Assuming we have a comprehensive physical network in place, meaningful communication between machines is still not assured since a variety of machines exists on campus and they use different protocols.

At Carnegie Mellon, the most popular protocol family in use is the Internet Protocol (IP). It is the protocol supported by DARPA and is available under Berkeley UNIX. There are over 400 stations on campus that use IP as the native protocol. These stations are attached to a variety of Ethernets, ProNets and IBM token rings. The campus IP internet is also connected to the ARPANET through an IMP operated by the Computer Science Department. The second most popular network on campus is DECNET. Over 100 VAXes, Pro-350 and DEC-20 machines are run DECNET on Ethernet, ProNet and high speed point-to-point links. The Carnegie Mellon DECNET is, in turn, connected to DECNETs at Columbia, Case Western, University of Pittsburgh and Westinghouse Research Laboratory. Other protocols in use, to a lesser degree, are PUP and XNS.

There are three approaches to solving the protocol problem: (a) provide every machine with the capability to handle all other protocols in use beside its native set; (b) provide protocol translation machines; and (c) select a standard protocol and ensure all machines can handle this protocol in addition to the native set. The first approach is impractical. Protocol translation can be achieved either by implementing a set of any-to-any protocol translators or switching through an intermediate protocol. It is simple to see that the latter is preferable. Approach (c) has the advantage that, if we can find a standard protocol which has implementation on all the machines, no additional work is required. We focused our attention on approach (c) making IP the campus standard, using (b) as the fall back.

3.3. Interconnection of LAN's

Physically, LAN's in different buildings can be interconnected using the large fiber optic plant that we have in place. Hence for Ethernet, we can use fiber repeaters supplied by DEC or Ungermann-Bass. However, physically connecting networks together is not very desirable since we will end up summing the traffic of the networks and make the whole network less reliable and secure. The ideal approach is to connect the networks together logically. We have achieved that by using the LANbridge from DEC as well as with locally developed routers.

The LANbridge is an Ethernet MAC layer (ISO level 1.5) selective relaying entity. It sits between 2 Ethernets, examines every packet on the networks and decides whether to relay it to the other side or not based on a route table. The route table is generated dynamically through observing the source address of all the packet traffic. Because of the fact the bridge has to handle and examine every packet on both networks, very high speed processing, probably with hardware assist is required -- particularly in the area of table look up. The big advantage of this device is that it is higher level protocol independent. Hence it can be used to

interconnect networks supporting DECNET, IP, XNS and other protocols. There are a number of small disadvantages. Currently at least, the bridge can only be used for the interconnection of Ethernets. Furthermore, each bridge can only be used to interconnect two networks. This makes the cost per connection quite high.

Another device we used for network interconnection is a router developed locally by the Computer Science Department [1]. A router is a IP layer (ISO level 3) relaying entity that attaches to two or more LANs. For each LAN it masquerades as all the workstations not on that LAN by the following trick: Every workstation is assumed to use Plummer's algorithm [12] to discover the physical (e.g. Ethernet) addresses corresponding to IP addresses on it local net; i.e. it sends and receives broadcast address resolution packets (ARPs) that ask who has IP address Z". A router on nets A and B, upon seeing an ARP from station X on net A will broadcast it on net B. If it receives a response on net B, it will then answer X, substituting its own physical address as the answer and remembering the real physical address from net B. Note that if there is one machine somewhere with the given IP address, X will get an answer from either the router or some station on net A. When X sends a packet to the physical address of the router addressed to the given IP address, the router looks up the physical address for net B and passes the packet on.

The original implementation was done for PDP-11. Most of the routers currently deployed use lower cost, higher performance 68000 multibus or PC/AT based hardware. Since each router can support the interconnection of 3 to 4 networks, the per net cost is significantly lower than that of the LANbridge. Furthermore, the router can support the interconnection of a variety of LAN types including Ethernet, ProNet, IBM token ring, 56K and 9.6K synchronous lines. We have been using the routers for almost two years and they have been very reliable. There are three shortcomings, however. First, it depends on ARP which is native only to the DARPA protocol set. Hence it will not support the interconnection of DECNET or other non-IP stations. Since all machines on campus will support DARPA protocol as described earlier, it is not a serious problem. The second shortcoming is that ARP request is currently relayed as a broadcast. Since broadcast has to be handled by every attached station in the net, this can become quite expensive for a large inter-connected set of nets with thousands of stations. We have a modified version of the router under test which will heuristically relay some of the ARP requests to specific hosts instead of general broadcast. The third problem is that our current algorithm does not allow loops in the internet. This means no alternative paths for either redundancy or load sharing. While it has not been a problem for us since the reliability of the 12 deployed routers has been very high, having multiple path support is still desirable.

We are in the process of developing a second generation router that is sub-net based. It will address both the ARP problem and provide redundant paths with best path selection capability. It is similar in concept to those used by MIT and Stanford [10].

The Carnegie Mellon internet as of January, 1986 is shown in Figure 3. It is our goal to provide the minimum number of hops between any two nets in our interconnection topology. The UCC Ethernet fulfills the role of a backbone switch net. Most departmental networks are connected directly to this net through only one relaying element. Hence the number of hops between most networks is two. In our topology, the relaying elements are attached to the backbone rather than the departmental networks. This has the advantage of allowing the routers to be shared and hence lowering the overall cost. However, it also means that the additional inter-building cable lengths are added to the departmental Ethernets. This adds to the planning complexity for the managers of those networks. Currently, the UCC Ethernet serves both the function of a switch net as well as the main resource sharing network. All the VICE file servers are also attached to this net. When our topology is changed to one that support multiple paths, a separate server net will be installed.

3.4. The Cable Plant

While the current Ethernet, ProNet and IBM token ring cable plants have provided a large campus wide integrated high speed communication path way, expansion is still very much on an *ad hoc* basis and service is by no means universal. It is clear that this type of *ad hoc* installation of Ethernet drop cables or token ring lobes will, sooner or later, lead to mass confusion. It was decided that we should comprehensively re-wire the campus -- both for data communications as well as for future phone services -- using the IBM cabling system. While this may not be the most exciting aspect of data communication technology, cable plant planning and installation, especially in case of retrofitting existing buildings, can be extremely costly. Furthermore, its impact spans a few generations of networking technologies. It is, therefore, most important that the cable plant selection, design and implementation be handled with proper care.

Approximately 50 buildings of various size, shape and vintages will be affected. Over 10,000 outlets will be installed. The current estimated cost of the project runs over $5 million. Most of this will go into labor. The design of the cable plant is well under way. Installation began in the fall of 1985. So far, 9 buildings have been wired. Completion is expected by end of 1986. Details of our

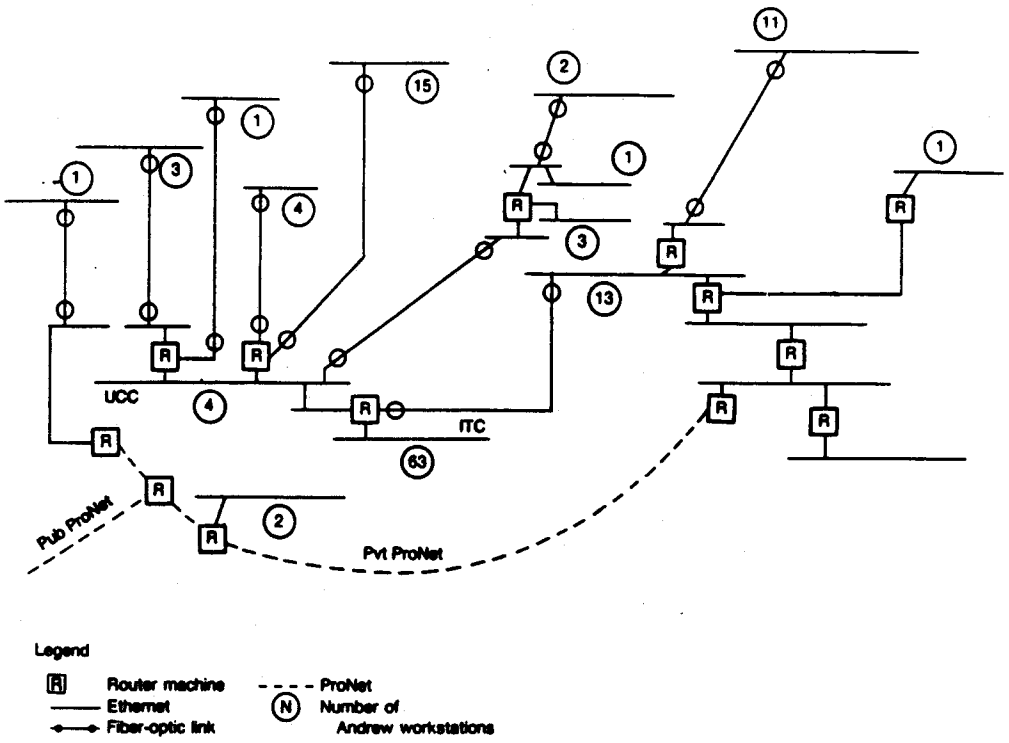

Figure 3: The current Carnegie Mellon Internet

design can be found in [8], and a paper summarizing our experience will be produced upon the completion of the project.

4. The Shared File System

In this paper we present our experience in building the first version of VICE. A separate paper [14] describes the design in detail, including justification for the decisions made and a survey of related work. Here we concentrate on implementation and experience, including only enough design information to make the description self-contained.

VICE is intended to be shared by thousands of workstations. It provides an integrated UNIX-like file naming hierarchy, using a network of cooperating file servers; users' workstations attach to it by the network described above. The first version was a prototype, intended primarily to verify the following fundamental design concepts:

Whole File Transfer
: Workstations read and write entire files from file servers rather than pages or records. What effect does this have on performance? Are large files sufficiently rare?

File Caching
: Workstations cache files on their local disks. Are these disks large enough to cache a typical working set of files? How well do our cache management algorithms work?

UNIX Semantics
: How well can we emulate UNIX and retain the benefits of centralized timesharing systems in a distributed environment? Does whole file transfer conflict with our desire to run UNIX application programs without modification?

Server Load
: VICE is implemented with multiple cooperating servers in order to grow with the user community. How many workstations can a single server support? Can the load be balanced across servers?

Access Control
: VICE augments the UNIX protection system to allow access lists on directories. Will this addition prove useful?

The major components of the implementation include the file server, VICE proper, and the workstation cache manager, *Venus*. Figure 4 shows the two components and their process structure. They communicate using a remote procedure call package (RPC) that is linked into each program. Since our primary goal in the prototype was to test our design, we traded performance for function whenever it would affect development time. Often this meant using off-the-shelf software. For example, the file server itself is based on UNIX and depends heavily on its file system; the RPC package uses sockets, a unique feature of Berkeley UNIX.

Since VICE is based on whole file transfer, the interface between the servers and Venus is very simple. It has two basic types of

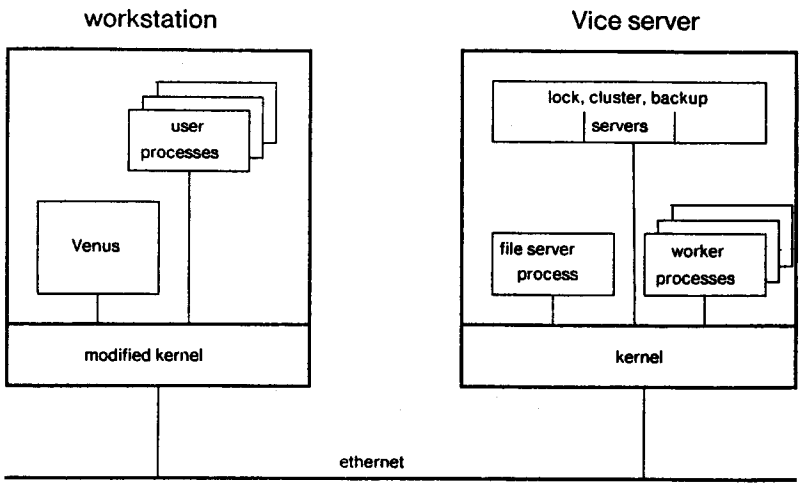

Figure 4: VICE Server/Venus Process Structure

objects: *files* are uninterpreted byte streams, and *directories* are lists of files and other directories.

The operations provided by the server allow Venus to:

Fetch, store and remove files. The entire file is shipped on each fetch or store.

Make and remove directories. Venus cannot directly store a directory; directories are updated as side effects of other operations.

Get and set status information. For both files and directories.

Check access rights and file currency. Access rights are checked for a particular user relative to a particular file or directory. The same operation also returns the last change date for the file or directory.

Acquire and release locks. To implement advisory locking.

4.1. The Servers

VICE is distributed over several server machines. On each of these machines, there are a number of independent processes that implement the server function by fielding requests and performing updates on the local file system. The directory tree is partitioned among the file servers so that each server is the *custodian* of a set of subtrees. Certain important, infrequently changing subtrees are replicated on many servers. For example, the root of the file system and a few levels below it are replicated everywhere as are commonly run programs

When a server machine is brought up several processes are started to handle the requests to the server. The primary process is the *file server process*, which listens for connecting workstations. When a workstation connects, the file server process forks a *worker process* which handles all requests for a particular user from that workstation. All requests from that user on that workstation are routed to this single worker process.

The *lock process* is used to handle any lock requests. Because the operating system does not allow memory to be shared between processes, we use a separate process in order to keep the locks in memory. Lock requests must be directed to the custodian of the file being locked.

4.2. Venus

Venus is implemented as a user-level process running on a workstation with a modified kernel. In response to file system calls made by client programs, Venus caches files from the servers and makes the files in its cache available to the client programs. In order to do this, Venus must locate the files among the servers, connecting to servers as needed on behalf of the client.

From the point of view of an application running on the workstation, the VICE file system appears as a subtree in the name space. The workstation kernel intercepts file system calls directed to files in that subtree and routes them to Venus. When it has performed the system call, Venus sends a reply via the kernel to the original calling process. The cache is maintained with an LRU replacement algorithm. The prototype implementation limits the cache size by the number of files in the cache.

Whenever Venus sends a request to the wrong server, it receives in the reply the name of a subtree that contains the file it needs and the custodian for that subtree. Venus uses the custodian name to continue its search for the file. Venus remembers the names of the subtrees and uses this information to make first guesses as to

where to find files in subsequent requests. When Venus needs to direct a file operation to some server, it looks in the location table for the longest prefix of the path name named in the operation. This prefix gives the deepest spot in the file hierarchy along the path to this file for which a server is known. This server is used as the first guess for the real location of the file.

4.3. Measurements

In this section, we discuss a number of performance-related issues pertinent to our implementation. One of the most important of these issues is the ratio of remote to local file access times. Also important are the effects of whole-file transfer and caching on performance. We are also interested in knowing how many users can be reasonably assigned to a server, whether we are balancing the load on our servers evenly, and on the factors which currently limit performance.

The prototype servers run on SUN 170s and VAX 11/750s with four megabytes of memory and two 400 megabyte disk drives each. The workstations are SUN 120s and SUN 100s with two megabytes of memory and 70 megabyte local disks. The SUNs are based on the Motorola 68010 processor chip.

In of March 1985 we had two file systems running. One was our internal system and consists of two servers with about 50 workstations and 75 users. The other was used outside the ITC and consists of four servers, with 50 workstations attached exclusively to it in addition to the internal workstations. About 250 users were authorized to use the external system. The internal system had about a gigabyte of data stored in it, and the external system had about 1.5 gigabytes of data.

A user who is accustomed to a stand-alone workstation perceives some qualitative performance degradation when he uses a Virtue workstation. Although command execution is noticeably slower in Virtue than on a stand-alone workstation, the performance is often better than on the timesharing systems used by the general campus user community at Carnegie Mellon. Performance degradation is not uniform across all operations. Some operations, like the compilation of a large program, proceed at almost stand-alone speed. Other operations, such as a recursive search of a subtree of files, take much longer when the subtree is in VICE. We find that performance is usually acceptable up to a limit of about 25 active users per server. However, there have been occasions when even a few users intensely using the file system have caused performance to degrade intolerably.

Certain programs run much slower than we had originally expected, even when all relevant files are in the local cache. This is because such programs obtain status information about files (using the *stat* primitive), before actually opening them. Since each stat call either involves a cache miss or a cache validity check, the total number of client-server interactions is significantly higher than the number of file opens. This increases both the total running time of these programs and the load on the servers.

The whole-file transfer approach contributes significantly to good performance during many frequent user operations such as program compilation. Execution proceeds at the same speed as on a stand-alone system except for an initial delay to fetch the compiler binary and the file being compiled (or to validate their cache entries), and to store the generated code back into VICE. Another common user operation is the editing of files. The editors in use in our environment read the entire file being edited into virtual memory prior to using them. Whole-file transfer neither improves nor hurts performance in this case,

An obvious quantity of interest in a caching file system is the hit ratio observed during actual use. Venus uses two caches: one for files and the other for status information about files. A snapshot of the caches of 12 machines in our environment shows an average file cache hit ratio of 81% with a standard deviation of 9.8% and an average status cache hit ratio of 82% with a standard deviation of 12.9%

Also of interest is the relative distribution of individual file operation calls. Such a profile is valuable in improving server performance, since attention can be focussed on the most frequent calls. Table 1 shows the observed distribution of each call which accounts for more than one percent of the total. This data was gathered over a one-month period on five cluster servers. The distribution is dramatically skewed, with two calls accounting for nearly 90%of the total. The *TestAuth* call is used to validate check entries, while *GetFileStat* is used to obtain status information about files absent from the cache. The table also shows that only 6% of the calls to VICE (*Fetch* and *Store*) actually involve file transfer, and that the ratio of *Fetch* calls to *Store* calls is approximately 2:1.

In order to investigate the performance penalty caused by VICE and Venus, we performed a series of controlled experiments using a benchmark. This benchmark stresses the file system far more intensely than a typical user and involves a series of file copy operations, directory and file scans, and a large compilation. Table 2 presents the total running time for the benchmark as a function of the number of clients simultaneously executing that benchmark. The table also shows the average response time for the most frequent VICE operation, *TestAuth*, during each of the

Server	Total Calls	Call Distribution						
		TestAuth	GetFileStat	Fetch	Store	SetFileStat	GetMembers	All Others
cluster0	1625954	64.2%	28.7%	3.4%	1.4%	0.8%	0.6%	0.9%
cluster1	564981	64.5%	22.7%	3.1%	3.5%	2.8%	1.3%	2.1%
cmu-0	281482	50.7%	33.5%	6.6%	1.9%	1.5%	3.6%	2.2%
cmu-1	1527960	61.1%	29.6%	3.8%	1.1%	1.4%	1.8%	1.2%
cmu-2	318610	68.2%	19.7%	3.3%	2.7%	2.3%	1.6%	2.2%
Mean		61.7% (6.7)	26.8% (5.6)	4.0% (1.5)	2.1% (1.0)	1.8% (0.8)	1.8% (1.1)	1.7% (0.6)

NOTE:

Figures in parentheses are standard deviations.
The data shown here was gathered over a one-month period.

Table 1: Observed Distribution of Vice Calls

Configuration	Overall Benchmark Time		Time per TestAuth Call	
	Absolute (s)	Relative	Absolute (ms)	Relative
Stand-Alone	998 (9)	100%	NA	NA
1 client/server	1789 (3)	179%	87 (0)	100%
2 clients/server	1894 (4)	190%	118 (1)	136%
5 clients/server	2747 (48)	275%	259 (16)	298%
8 clients/server	5129 (177)	514%	670 (23)	770%
10 clients/server	7326 (69)	734%	1050 (13)	1207%

NOTE:

Figures in parentheses are standard deviations.
Each client had a 300-entry cache.
Each data point is the mean of 3 independent replications.

Table 2: Stand-Alone versus Remote Access

Server	Samples	CPU Utilization			Disk 1			Disk 2		
		total	user	system	util	KBytes	xfers	util	KBytes	xfers
cluster0	13	37.8% (12.5)	9.6% (4.4)	28.2% (8.4)	12.0% (3.3)	380058 (84330)	132804 (35796)	6.8% (4.2)	186017 (104682)	75212 (42972)
cluster1	14	12.6% (4.0)	2.5% (1.1)	10.1% (3.4)	4.1% (1.3)	159336 (41503)	45127 (21262)	4.4% (2.1)	168137 (63927)	49034 (32168)
cmu-0	15	7.0% (2.5)	1.8% (0.7)	5.1% (1.8)	2.5% (0.9)	106820 (41048)	28177 (10289)			
cmu-1	14	43.2% (10.0)	7.2% (1.8)	36.0% (8.7)	13.9% (4.5)	478059 (151755)	126257 (42409)	15.1 (5.4)	373526 (105846)	140516 (40464)

NOTE:

Figures in parentheses are standard deviations
Peak period is defined as 9am to 5pm on weekdays.
The data shown here was gathered over a two-week period.

Table 3: Server Usage During Peak Period

experiments. One important observation from this table is that the benchmark takes about 80%longer in the 1 client/server case than in the stand alone case. A second observation is that the time for *TestAuth* rises rapidly beyond a load of 5 clients/server, indicating server saturation. For this benchmark, therefore, a client-server ratio between 5 and 10 is the maximum feasible.

For measuring server usage, we have installed software on servers to maintain statistics about CPU and disk utilization, and about data transfers to and from the disks. Table 3 presents this data for four servers over a two-week period. The data is restricted to observations made during 9am to 5pm on weekdays, since this is the period of most intense system use. As the CPU utilizations in the table show, the servers are not evenly balanced. This fact is independently confirmed by Table 1, which shows a spread of about 5:1 in the total number of VICE calls presented to each server. This imbalance is due to a flaw in the implementation of the file location algorithm that encouraged workstations to gang up on one server and also the difficulty of reassigning users to servers in the prototype.

Table 3 also reveals that the two most heavily used servers show an average CPU utilization of about 40% This is a very high figure, considering that it is an average over an 8-hour period. Closer examination of the raw data shows much higher short-term CPU utilization: figures in the neighborhood of 75% over a 5-minute averaging period are common. Disk utilizations, however, are much lower. The 8-hour average is less than 15% and the short-term peaks are rarely above 20% We conclude from these figures, and from server utilization data obtained during the benchmarks, that the current performance bottleneck is the server CPU. Based on profiling of the servers, we are led to believe that the two factors chiefly responsible for this high CPU utilization are the frequency of context switches between the many server processes, and the time spent by the servers in traversing full path names presented by workstations.

To summarize, the measurements presented in this sec indicate that significant performance improvements are possibl we reduce the frequency of cache validity checks, reduce the number of server processes, require workstations rather than the servers to do path name traversals, and balance server usage by reassigning users.

4.4. Qualitative Evaluation

Transferring entire files to and from the workstation contributes greatly to the success of our prototype. Despite the relatively slow performance of our servers, the overall system is quite usable because reads and writes are performed locally. With the current system's performance, we feel quite comfortable handing files up to about a megabyte. We have rarely encountered larger files in day-to-day usage.

The default file cache for Venus is 300 files. With this cache size, we achieve an average cache hit rate of over 80% The cache normally fits into about ten megabytes The cache manager provides an interface to the file system that is highly compatible with UNIX. We run standard applications without modification. The high-level design decisions do not interfere with our ability to emulate UNIX. The difference in file-sharing granularity has not proved to be a problem in practice.

We have found the access list mechanism to be very useful. It is superior to UNIX group protection because the user is easily able to create lists of users for protection purposes without interacting with the system administrator. While it has many advantages, it makes emulation of the UNIX protection scheme difficult. We are occasionally inconvenienced by not being able to set protections separately on individual files. Also, since the *chmod* system call does not change the access lists, UNIX programs cannot change the protection of files. This has been a minor inconvenience.

4.5. VICE II

Based on the measurements and experience, we set out to design and build a more efficient and easily operable implementation of our basic architecture. The result of this effort is VICE II.

In VICE II, a single process on each server services all file server requests from clients to that cluster server. This process uses a lightweight process package with non-preemptable scheduling to concurrently service many client requests. The RPC package is integrated with the lightweight process package, allowing the file server to be making or servicing one remote procedure call per lightweight process. We stopped using the TCP layer of the IP protocol implementation because it limited the number of connections a machine could handle; instead the RPC package handles the functions of multiplexing and flow control.

The use of a single server process makes it possible for us to maintain virtual-memory caches of many data structures which were kept in the file system in the prototype. This improves performance and avoids the resource limitation problems, excessive paging and context-switching we encountered.

In VICE II, we use the UNIX file system on servers only to provide access to disk blocks, to manage storage allocation for files, and to maintain in-memory buffers of recently used disk blocks. The VICE directory structure is built on top of this low-level interface and does not appear as a UNIX directory structure on the server.

We have introduced the notion of a *Volume* as the basic abstraction for administrative and operational purposes. A volume is a collection of VICE files comprising a partial subtree of the file system hierarchy and is typically small; each user in our system currently has a volume allocated to him. Tape backup and restoration, application of disk space quotas, and read-only replication are all done on individual volumes.

The VICE-Venus interface in VICE II uses unique *File Identifiers* (fids) rather than full path names. A fid contains a volume number, a key into the volume index, and an additional field to ensure uniqueness within the volume. Fids remain invariant across renames and are therefore the key to making the renaming of directories possible. The translation of full path names into fids is done by Venus, which caches each directory encountered during translation. For robustness, modifications to directories can still only be done by servers. Symbolic links are also interpreted by Venus.

Cache management is an area where VICE II differs conceptually from VICE I. In VICE II, servers maintain a record of what each workstation has in its cache. If the file or directory held in a workstation's cache is ever modified by anyone else, the server will inform the workstation that its cache entry has been invalidated. Thus a workstation can use cache entries without any validation, thereby cutting down significantly on client-server traffic. Servers are free to invalidate cache entries for their own convenience (e.g. to reduce space used on a server), even if the corresponding files are unchanged. Caches are still write-through in VICE II, but the cache replacement algorithm is based on the total space used by cached files.

VICE II was deployed at Carnegie Mellon in January of 1986. Some of the functions, such as authentication, are still being integrated into the system. We have not yet performed measurements corresponding to the ones reported above. However, even our limited experience with this system confirms its superiority. The ability to have symbolic links in Vice and to rename directories has enhanced the usability of the system. The cache invalidation mechanism and the use of a single UNIX process per server have resulted in marked performance improvement. We have not encountered any UNIX resource limitation problems with VICE II, even though we have hundreds of workstations connected to a sever. However, actual file transfer times were initially worse, especially through routers, because of our replacement of TCP and its flow control algorithms. The internal structure of the server, RPC and Venus is considerably more complex and has made debugging more difficult. A server process crash is now no longer a matter of a single user being inconvenienced.

5. Summary and Acknowledgements

Andrew is still being developed so the experience related here should not be read as the final story. We have presented this rather detailed and candid account of our network and file system developments so that others involved in similar efforts might benefit.

The work reported on here is the result of a large number of people at Carnegie Mellon. The network was created through the cooperation of several departments and the Computation Center coordinated by John Leong and Bob Cape. The file system was designed and programmed by an ITC team consisting of John Howard, Mike Kazar, David Nichols, Bob Sidebotham, Mahadev Satyanarayanan, and Mike West.

Carnegie Mellon and IBM deserve considerable credit for the foresight to initiate this project and the determination to carry it through.

References

1. Accetta, M. A network router. Department of Computer Science, Carnegie-Mellon University, September, 1983.

2. Balkovitch, E., Lerman, S., and Parmalee, R.P. "Computering in Higher Education: the Athena Experience." *Communications of the ACM 28*, 11 (November 1985).

3. Ball, J.E., Barbacci, M.R., Fahlman, S.E., Harbison, S.P., Hibbard, P.G., Rashid, R.F., Robertson, G.G., Steele, G.L. . The Spice Project. Computer Science Research Review, Carnegie-Mellon University, 1981.

4. Gosling, J.A. and Rosenthal, D.S.H. A Network Window Manager. Proceedings of the 1984 Uniforum Conference, Washington, D.C., January, 1984.

5. Hiltz, J., Starr, R., and Turrof, M.. *The Network Nation.* Addison-Wesley, 1979.

6. Lampson, B. and Taft, E. (eds). *Alto User's Handbook.* September 1979.

7. Leong, J. "Nuts-and-bolts guide to Ethernet installation and interconnection." *Data Communications 14*, 10 (September 1985).

8. Leong, J. Data Communication at CMU. Tech. Rept. CMU-ITC-85-043, Information Technology Center and Computer Center, Carnegie-Mellon University, July, 1985.

9. Metcalfe, R.M. and Boggs, D.R. Ethernet: Distributed Packet Switching for Local Computer Networks. Tech. Rept. CSL-75-7, Xerox Palo Alto Research Center, May 1975, reprinted February 1980.

10. Mogul, J. and Postel. *RFC950: Internet Standard Subnetting Procedure.* Defense Advanced Research Projects Agency, Information Processing Techniques Office, August, 1985.

11. Morris, J. H., et al. "Andrew: A Distributed Personal Computing Environment." *Communications of the ACM 29*, 3 (March 1986).

12. Plummer, W. *RFC826: An Ethernet Address Resolution Protocol.* Defense Advanced Research Projects Agency, Information Processing Techniques Office, November, 1982.

13. Ritchie, D.M. and Thompson, K. "The UNIX Time-Sharing System." *Bell System Technical Journal 57*, 6 (July-August 1978).

14. Satyanarayanan, M., Howard, J.H., Nichols, D.N., Sidebotham, R.N., Spector, A.Z. and West, M.J. The ITC Distributed File System: Principles and Design. Proceedings of the 10th ACM Symposium on Operating System Principles, December, 1985.

15. Smith, S., and Sfferwood, B.A. "Educational uses of the PLATO computer system." *Science 192* (1976).

16. Sproull, L. and Kiesler, S. Reducing social context information: The effects of electronic mail on organizational communication. Department of Social Science, Carnegie-Mellon University, October, 1985.

17. Strole, N. "A Local Communications Network Based on Interconnected Token-Access Rings: A Tutorial." *IBM Journal of Research and Development 27*, 5 (September 1983).