01

A
R
M
S

The Early Years of Academic Computing:

A Memoir by William Y. Arms

This is from a collection of reflections/memoirs concerning the early years of academic computing, emphasizing the period from the 1950s to the 1990s when universities developed their own computing environments.

Published by The Internet-First University Press
Copy editor and proofreader: Dianne Ferriss
The entire incremental book is openly available at http://hdl.handle.net/1813/36810

1.0

A. Memoir by William Y. Arms

Preface

The past fifty years have seen university computing move from a fringe activity to a central part of academic life. Today's college seniors never knew a world without personal computers, networks, and the web. Sometimes I ask Cornell students, "When my wife and I were undergraduates at Oxford, there were separate men's and women's colleges. If we wanted to meet in the evening, how did we communicate?" They never knew a world without telephones and email, let alone smartphones, Google, and Facebook.

They are also unaware that much of modern computing was developed by universities. Universities were the pioneers in end-user computing. The idea that everybody is a computer user is quite recent. Many organizations still do not allow their staff to choose their own computers, decide how to use them, and select the applications to run. But end-user computing has been established in universities since the 1960s. When the computer industry was slow in seeing the potential, universities took the initiative. This narrative describes a thirty-year period when academic computing diverged from the mainstream. Universities built their own state-of-the-art systems and ran them successfully. They led the development of timesharing and local area (campus) networks. They were major contributors to distributed computing, email, the national networks, and the web.

This is not a history. It is a memoir. As a student, faculty member, and administrator, I lived through many of these developments. From 1978 to 1995 I was in charge of computing at two of the leading universities, Dartmouth College and Carnegie Mellon. After 1995 I was no longer personally involved in the developments, but in the final section I describe how academic computing and the mainstream have now come together again, to the benefit of both.

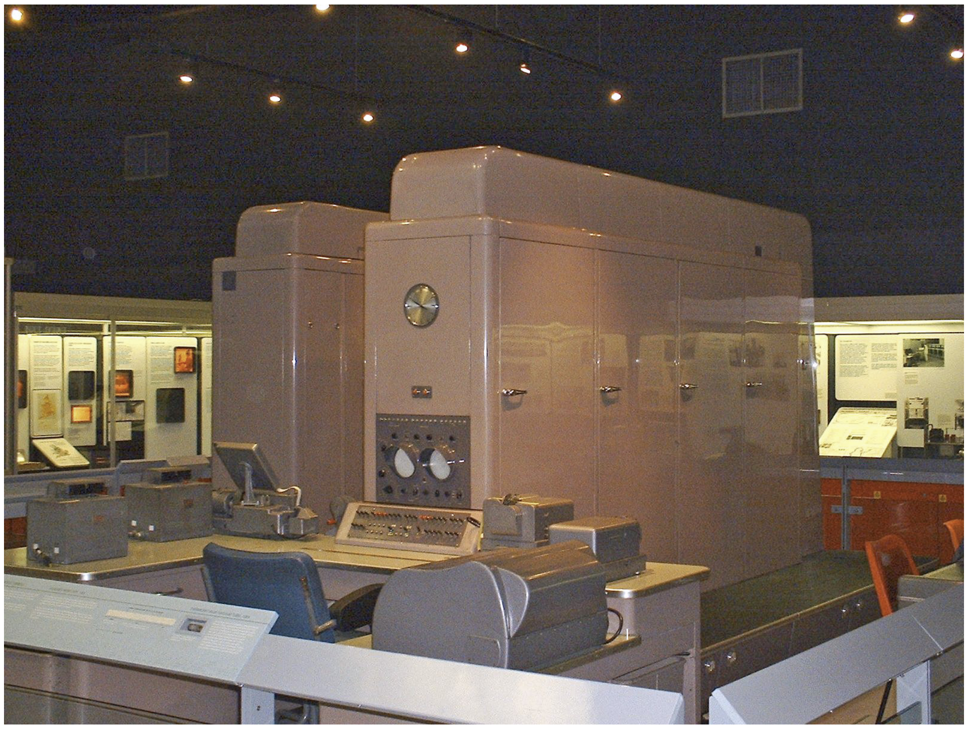
There is a glaring gap that runs throughout this narrative. It says little about the impact of computing on academic life, on teaching, research, and libraries. The final section has a few thoughts about educational computing, but it is a huge topic that deserves a much fuller treatment.

Disruptive change is an underlying theme of this period. I have tried to give some indication of what it was like to be a computing director, the opportunities and the excitement, but also the pressures and uncertainties. I was fortunate in working for two remarkable presidents, John Kemeny at Dartmouth and Richard Cyert at Carnegie Mellon. Each was a visionary in seeing how computing could be used in higher education. When I remember my colleagues at other universities wrestling with less enlightened leaders, I know how lucky I was.

Contents

Chapter 1

The Early Days



A second generation computer

This photograph is of a Ferranti Pegasus computer in the Science Museum in London. The museum claims that it is the oldest computer in the world that is still operational.

Wikipedia

Background: The 1960s

The 1960s

The mid-1960s were a pivotal time for computing. In 1965, as semiconductors replaced vacuum tubes, Gordon Moore of Intel observed that the number of transistors on integrated circuits was doubling every two years. A year earlier, IBM had announced the 360 family of computers. Timesharing was born that same year, when two people at Dartmouth College using separate terminals typed "PRINT 2+2" and (after some bugs were fixed) both got the answer 4.

Moore's Law is not a law of nature, but for fifty years hardware engineers have used it as a target. On aggregate they have actually done better than Moore predicted. This exponential growth has created enormous opportunities, but it has also created intense challenges. Many years later, Allen Newell pointed out that collectively we do well in predicting the performance of components such as processors and disks; we do quite well in predicting the next generation of hardware, such as networks and personal computers; but we have consistently failed to predict the breakthrough applications enabled by this hardware. He might have added that we also fail to anticipate the organizational and social impact.

A Pegasus computer

My early experience provides some typical examples of computing in the 1960s, before the emergence of timesharing. It was characteristic of the period that as a mathematics undergraduate at Oxford, graduating in 1966, I never went near a computer.

I wrote one small program in 1962 while at school in England. A local engineering company had a Ferranti Pegasus computer and invited a group of us (all boys) to visit it. There is a Pegasus in the Science Museum in London. It is an imposing collection of shining boxes, which occupy a space the size of a small room, but the computer we saw had most of its covers removed exposing racks of vacuum tubes and electrical circuits. The covers were off because the components were highly unreliable. The processor was an array of modules, each about the size of a shoe box and containing several vacuum tubes. Individual modules could be slid out of the rack and replaced, even while the machine was powered up.

We each had one attempt to run a small program. The programming language was Pegasus Autocode, a dialect of the family that later became Algol. My program solved a simple mathematical puzzle. We wrote our program instructions on paper. A clerk keyed them on to 5-track paper tape and an operator fed the tape into the computer. After a substantial pause, the results were printed out on a teletype terminal. My program gave the correct result. Other people were not so successful.

Fortran programming with punched cards

After my brief experience at school, I did not use a computer again until 1966 when I was a master's student at the London School of Economics. The school had an IBM 1440 computer.

The programming language was Fortran II, which was easy to use for the mathematical programs that we wrote. Nobody gave us instructions. Programming was assumed to be a simple skill that any student could pick up from the manual.

Before class we left our coding sheets at the reception desk and an hour later we picked up the results. We got back a deck of cards, a listing of our program, its compilation, and results. If we wanted to make changes, we wrote them on a new coding sheet and handed back the deck of cards. Apparently the school had decided that it was less expensive to pay somebody to key our programs than to waste expensive computer time trying to run badly keyed programs that we had punched ourselves.

English Electric - Leo - Marconi

My first job after graduation was in the operational research group of English Electric - Leo - Marconi. As the name implied, this was a merger of three separate companies, each of which manufactured computers during the 1960s. By the time that I actually started work the company had changed its name to English Electric - Leo, and soon afterwards it merged with International Computers and Tabulators Ltd. to become International Computers Limited (ICL).

A Leo computer from the early 1960s

Leo was one of the pioneers of computer-based data processing. It began as an in-house venture of J. Lyons, a bakery and food company. Lyons built the original machines for its own use. Leo was perhaps the first company to use multiprogramming successfully. The Leo III was able to run up to three programs simultaneously. The company also developed a high-level programming language, Cleo, similar to IBM's Cobol.

LEO Computers Society

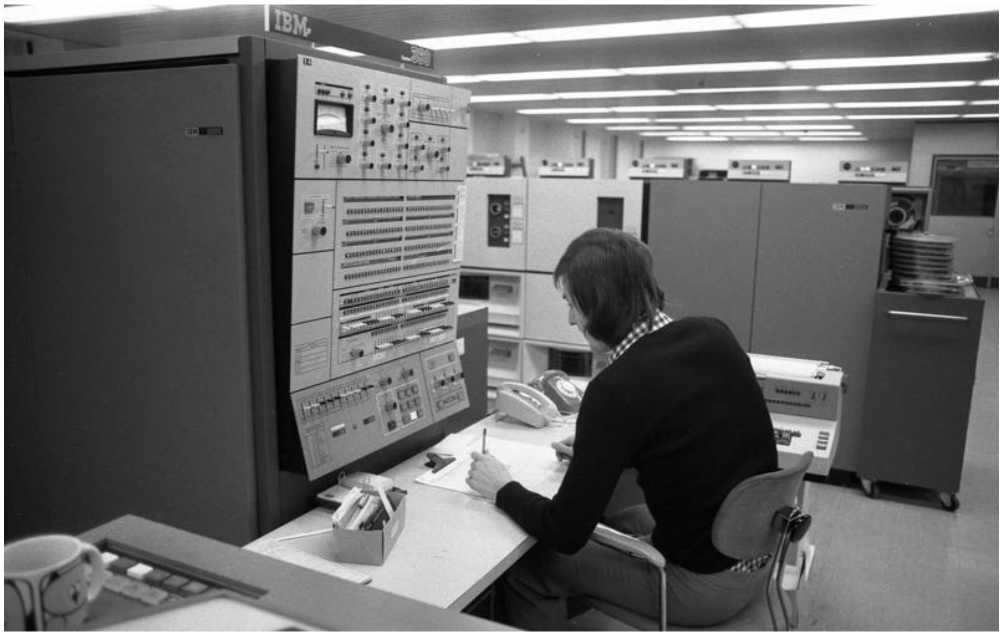
In 1967, English Electric - Leo introduced a new range of computers, known as System 4, which was a derivative of the RCA Spectra series. These machines used the same instruction set as the recently introduced IBM 360 series. The company was proud of the semiconductor technology, claiming to be the first commercial computer to have more than one gate on a silicon chip. Building on experience with the Leo 3, the central processor had hardware support for multiprogramming.

For the first three months on my new job, I was sent on an intensive training course to learn about these new machines. The course included assembly level programming and went into considerable detail about the hardware. This was a fortunate time to learn about computing as I was trained in the computer architecture that was to dominate mainframe computers for many years.

The IBM 360 and its Clones

Third generation computers

The computers introduced in the mid-1960s, such as the IBM 360 family, were called "third generation" computers. Third generation computers were important because each computer in a family had the same architecture. Previously, every new range of computers had had a different architecture and machines within a range were often incompatible. Whenever customers bought a new machine they had to convert all their old programs. The huge suites of software that we run today would not be possible without stable architectures and standardized programming languages. Over the fifty years since its first release, the IBM 360 architecture has been steadily enhanced, such as when virtual memory was added, but the enhancements have generally been backwards compatible.



An IBM 360

An IBM System/360 in use at Volkswagen.

German Federal Archives

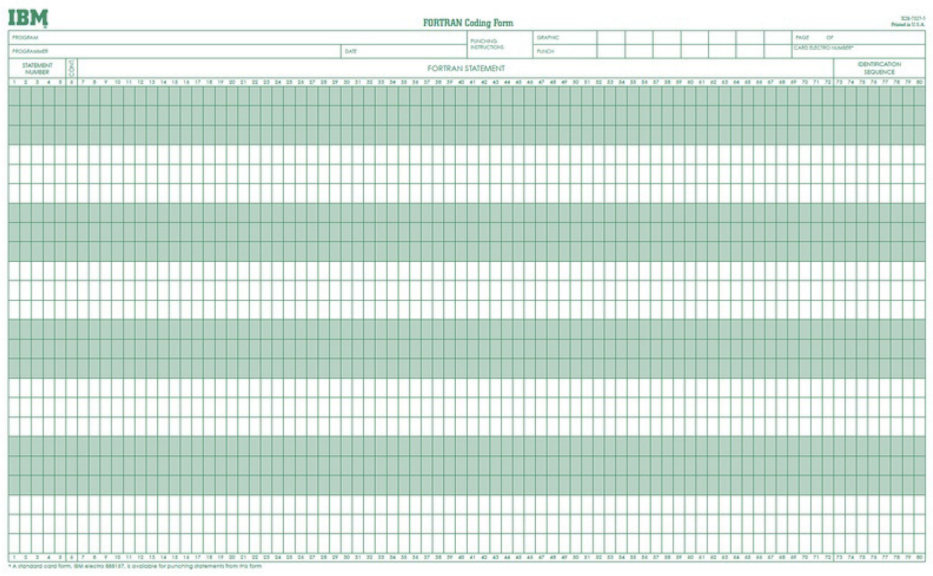
A third generation computer consisted of a group of large metal boxes. The cabinets were laid out in an air-conditioned machine room and connected with heavy-duty cables under a raised floor. Photographs show machine rooms that were models of tidiness, but in practice a machine room was a busy place, full of boxes of punched cards, magnetic tapes, and stacks of paper for the line printers. They were noisy, with card readers and line printers chattering away, and the continuous rush of air-conditioning.

Solid-state components had replaced vacuum tubes, but the logic boards were still large and the central processor was several tall racks of circuit boards with interconnecting cables. Memory used magnetic cores, which were bulky and expensive. In 1968, the price of memory was about $1 per byte.

Most of the boxes in the machine room were peripherals and their controllers. Rotating disks were beginning to come on the market, but most backup storage was on open-reel magnetic tape. The tape held nine bits of data in a row across the tape, representing one byte plus a digital check. Early drives wrote 800 rows per inch, but higher densities followed steadily. To sort data that is on magnetic tape requires at least three tape units and most systems had many more. The controllers that connected the tape and disk units to the central processor were imposing pieces of expensive hardware.

The architecture of the IBM 360 supported both scientific computing and data processing. It firmly established the eight-bit byte as the unit of storage for a character and four bytes were organized as a word. Earlier computers had used a variety of word sizes and characters had often been stored as six bits. In the days before virtual memory, memory was organized into 4096-byte modules and I vividly remember the challenges of memory management when writing assembly code programs.

The instruction set was designed to support high-level languages, such as Cobol for data processing, and Fortran for scientific work. Arithmetic could be decimal, operating on character strings, or binary, operating on words that could represent either integer or floating-point numbers. By using variable-length instruction formats, complex operations could be carried out in a single instruction. For example, only one instruction was needed to move a string of characters to another location in memory. Apart from the memory management, I found the assembly code straightforward and easy to program in.


IBM
FORTRAN Coding Form
PROGRAM
PROGRAMMER
DATE
PUNCHING INSTRUCTIONS
GRAPHIC
PUNCH
PAGE OF
CARD ELECTRO NUMBER*
STATEMENT NUMBER
CONT.
FORTRAN STATEMENT
IDENTIFICATION SEQUENCE

A Fortran coding sheet

Programmers wrote their code on coding sheets. A clerk keyed the program onto punched cards, one card for each line of the program.

Wikipedia

In the days before terminals and interactive computing, a programmer wrote the instructions on coding sheets, with one 80-character line for each instruction. The instructions were punched onto cards, one card per instruction. A colored job control card was placed at the head of the card deck and another at the end, followed by data cards. The complete deck was placed in a tray and handed to the computer operators. Sometime later the programmer would receive a printout with the results of the compilation and the output of any test runs. Computer time was always scarce and one test run per day was considered good service. This placed a premium on checking programs thoroughly before submission, as even a simple syntax error could waste a day's run.

Commercial data processing on third generation computers such as the IBM 360 used batch processing to automate tasks such as billing, payroll, stock control, etc. The algorithms resembled the modern "map/reduce" methods that are used to build indexes for Internet search engines. These methods are used to process huge volumes of data, such as when building indexes for Internet search engines. These algorithms are used when the amount of data is so large relative to the memory of the computer that there is no possibility of holding comprehensive data structures in memory and random access processing would be impossibly slow. Therefore the processing methods use sequential processing. The basic processing steps are sorting and merging, and the computation is done on small groups of records that have the same key.

In earlier years, some data input had used paper tape, but by the late 1960s punched cards were standard. An advantage of punched cards was that an error could be corrected by replacing a card, though it was easy to make further mistakes in doing so. Every computer center had a data preparation staff of young women who punched the transactions onto punched cards in fixed format. The cards were verified by having a second person key the same data. Data preparation was such a boring job that staff turnover was a perpetual problem. The British driving license system, which we developed at English Electric - Leo, was always understaffed in spring before the school year came to an end and the next wave of school leavers could be hired.

The first step in processing a batch of cards was to feed them into a card reader. A "data vet" program checked the cards for syntax errors and wrote the card images onto magnetic tape. This was then sorted so that the records were in the same sequence as the master file.

Records were stored in a master file on magnetic tape and sorted by an identifying number, such as a customer number. Sorting records on magnetic tape by repeated merging was time consuming. The performance depended on the number of tape drives and the amount of memory available. Time was saved by having tape drives that could be read backwards, so that no time was wasted rewinding them.

The master file update program would typically be run at the end of the day. The previous version of the file would be merged with the incoming transactions and an updated version of the master file written on a new magnetic tape. The update programs generated copious output to be printed, including business transactions such as bills or checks, management reports, and data errors. This output was written to another tape, which was then sorted by the category of printing, so that the operators could load the various types of paper into the printers as required.

The standard printer was a line printer, usually uppercase only. The paper was fan-folded with sprocket holes on the side and was advanced one line at a time. Most manufacturers used drum printers, with a complete set of characters around the drum, but IBM used a chain printer in which the characters were on a moving chain. I never understood how it worked. With either type of printer the actual printing was caused by a flying hammer that pressed the paper against the metal character.

Every part of the operation was prone to failure. Most computer centers halted work for several hours every night for maintenance, but even so hardware and software failures were common. An advantage of batch processing was that the system could write a checkpoint at the end of each batch. If the card reader jammed or a magnetic tape failed, the operators went back to the previous checkpoint, and reloaded the cards or the previous copy of the tape and restarted the job.

University Computing before Timesharing

End-user computing

During the 1960s, academic computing diverged from the mainstream. Most commercial applications, whether data processing or scientific, were large production jobs that ran for several hours and used the entire computer. Companies hired professionals to write the programs, punch the input, and run the jobs. In universities, the faculty, students, and researchers wrote their own programs and often ran them themselves. They would spend long periods developing programs, hoping for fast turnaround of compilations and small tests, followed by a few large computations. Since many of the programs were run only a few times, priority was given to convenient program development.

Computer hardware was improving rapidly, both in performance and reliability, but hardware was a scarce resource. Processor cycles could not be wasted. Because there were very great economies of scale, the best strategy for a university was to buy a single large computer and find a way to share it among the users. This led to the growth of university computer centers.

The University of Sussex

When we were at the University of Sussex from 1969 to 1972, we had computing facilities that were typical of the better universities at the time. The university computer was an ICL 1904A. The ICL 1900 series was an early third generation system that competed quite successfully with the IBM 360. The architecture used a 24-bit word. I think that the computer at the University of Sussex had 32K words, equivalent to less than 100K bytes.

ICL developed a sequence of operating systems called George. While waiting for ICL to deliver George III, the University of Sussex developed a simple monitor for running small Fortran jobs. It used a circular buffer on magnetic disk that held jobs waiting to be processed. The jobs were card decks of Fortran programs to be compiled and run. Since card reading was slow, the aim was always to have several jobs that had been read into the buffer, waiting to be run, so that the central processor was never idle. Output to the line printer was also buffered.

The Fortran compiler and the monitor used almost all the memory, but about 4K words were spare. My wife and I reached an agreement with the computing center that we could use this small amount of memory for our experiments with online catalogs. The computer had provision for a few terminals and we used one of them. This sounds an absurdly small amount of memory but we were able to use the Fortran IV overlay mechanism. This was a primitive form of paging by which the programmer specified that certain subroutines and data structures could overlay each other in memory. Since the basic Fortran I/O package itself used more than 4K, we wrote a physical I/O routine to control the terminal and never loaded the I/O package.

The university ran the computer for two shifts per day. The third shift, from midnight to 8 a.m., was available for researchers. Several of us went through the operator training course and for one night shift per week we had sole use of the machine. The procedure for rebooting the computer was typical of the era. Nowadays computers hold their boot program in some form of persistent storage, but the boot program for the 1904A was on paper tape. The first step in booting the machine was to set a few instructions using hand switches on the central processor. These instructions were executed when the computer was powered up. They instructed the paper tape reader to read the boot program into memory, and the boot program then read the operating system from magnetic tape.

Because the machine room was noisy, we would work on our programs in the reception area. We could tell what the machine was doing by listening to the audio monitor. This device, which was common on machines of that era, made a distinctive tone for each category of instruction that was being executed. Since each program had a distinctive pattern of sounds we could tell when a job, such as a tape sort or the Fortran compiler, came to an end.

Punched card equipment

The central computer was used almost entirely for academic work. Administrative data processing used punched card equipment. For example, the library's circulation system used nothing but punched cards and made no use of the computer. For one of my analyses I had more than seventy trays each containing 2,000 cards.

The punched card machines were direct descendents of the Hollerith machines that were built in the early 1900s to tabulate census data. IBM took over the Hollerith company and much of the company's data processing expertise came from its experience with punched card equipment.

Punched card equipment

In this IBM publicity photograph the operators are men, but in my experience most of them were women. The tidiness is also misleading. In practice, trays of cards and boxes of printer paper were stacked everywhere.

IBM Archives

The data processing room at the University of Sussex had about six large devices, each with its specialized function: a card sorter, copier, collator, tabulator, printer, etc. The collator was particularly important as it could merge data from two stacks of cards and punch out a new card, which combined the information from them. Each device was controlled by cables that were inserted into a plug board, thus creating a very simple program.

As an example, the card sorter had one input hopper and ten output hoppers. Sorting was one column at a time. The operator would use the plug board to specify a column of the card and other parameters. She would load the cards into the input hopper, one tray at a time, and the sorter would send each card to the output hopper that corresponded to the number punched in the appropriate column. To sort by a three-digit number, the cards would be passed through the sorter three times, sorting first by the least significant digit. Complex data processing operations, such as a master file update, were carried out by passing trays of cards repeatedly through the various devices.

Chapter 2

Timesharing

Two pioneers of educational computing

Tom Kurtz (left) and John Kemeny were pioneers in using interactive computing for undergraduate education. They developed the Basic programming language and the Dartmouth Time Sharing System.

Dartmouth College Library

Early Timesharing

Multiprograming, multitasking, and timesharing

Mainframe computers were expensive. The central computer that Dartmouth bought in 1975 cost more than four million dollars, a great deal of money for a small university. Moreover, these computers had huge economies of scale. Several years later, when Cornell studied the options for replacing its mainframe computer, an IBM 370/65 cost twice as much as a smaller IBM 370/55, yet was four times as powerful. With such economies of scale, every organization followed the same strategy: buy the largest computer you could afford and find ways to maximize the amount of work done on it.

Computer hardware improved rapidly during the 1960s, both in performance and reliability, but processor cycles were a scarce resource not to be wasted. Computers of the previous generation had run one program at a time and the central processor might spend much of its time idling. For example, a program to process a tray of punched cards would spend most of its time waiting for the next card to be read.

By the middle of the decade, computers were powerful enough to run more than one program at a time. The first technique that allowed several tasks to be run simultaneously was called multiprogramming. Each program was given a priority. At any given moment the operating system would offer the central processor to the highest priority program. Suppose that the highest priority program copied punched cards onto tape. The program would start to read a card and then wait while the card was read. This caused an interrupt and the scheduler would start up the next program in priority. This might send data to a line printer. This program too would spend much of its time waiting and the central processor could be assigned to another program. Rather unintuitively, the highest priority was given to the program that used the central processor least. This primitive form of multiprogramming required the operators to understand the characteristics of each program. It evolved into multitasking, where the user specifies the characteristics of a job, such as memory requirements, and the operating system schedules the execution. Modern operating systems use preemptive multitasking, where the operating system manages the entire scheduling, interrupting (or preempting) tasks after a predetermined time interval.

IBM's OS/360, released in 1966, is often considered to be the first modern operating system. Its focus was on efficient batch processing for commercial data processing. The painful lessons learned during its development were the subject of the first book on software engineering, "*The Mythical Man Month*" by Fred Brooks.

The beginnings of timesharing

Batch processing operating systems, with their emphasis on long-running data processing jobs, were unsuitable for academic computing. Universities, therefore, developed timesharing to give their faculty and students direct access to a central computer. The early systems included Multics at MIT, Titan at Cambridge University, and the Dartmouth Time Sharing System (DTSS). MIT and Cambridge emphasized the needs of researchers, while Dartmouth's focus was on undergraduate education. My first glimpse at timesharing was in the summer of 1965 when I visited Dartmouth and sat in on a lecture by John Kemeny on Basic programming. A year later, when my wife was a graduate student at MIT, Multics was in production and she was able to write Fortran programs for a research project.

Titan

This picture is of the operators' area in the Titan machine room at Cambridge University. Notice the hardcopy terminal with paper coming out.

University of Cambridge

Timesharing allows a central computer to be shared by a large number of users sitting at terminals. Each program in turn is given use of the central processor for a fixed period of time. When the time is up, the program is interrupted and the next program resumes execution. This is called "time slicing." A program can also be interrupted before the end of the time slice, for example it might have to wait for input. Timesharing is particularly effective when each user wants to use the computer intermittently, such as for program development, which has long periods of editing followed by short test runs. It is not suitable for the batch processing that is typical of commercial data processing

Dartmouth's DTSS and MIT's Multics had different objectives, and the systems reflected these differences, but the architectures had many features in common. Multics was closely linked to the emerging discipline of computer science. It is often described as the precursor to Unix. I recall a visit from Brian Kernigham of Bell Labs to Dartmouth in about 1982 when he was intrigued to discover that features of Unix, such as pipes, had equivalents in both architectures. Most of Multics was written in a high-level language, PL/1, unlike other operating systems that were written in assembly code and tightly linked to the hardware architecture.

DTSS had a direct commercial impact. Since large computers were so expensive, commercial timesharing bureaus sold computer time to remote customers. The market leader was General Electric's GEIS, which was originally developed in a joint project at Dartmouth. Concepts from DTSS were adopted by the early minicomputer systems from companies such as Hewlett-Packard and Digital Equipment Corporation. Even the computer HAL in the film "2001: A Space Odyssey" had commands taken from DTSS.

Timesharing dominated academic computing until the late 1980s, when it was replaced by personal computers. Even then, the differences between academic and commercial computing remained and led universities to build the first large networks of personal computers. As a result academic computing and the computing mainstream followed separate paths for about thirty years, from the mid-1960s to the 1990s.

Basic programming

Basic was developed at Dartmouth as a straightforward programming language for students and faculty. It became the language of choice in educational computing. Basic was never suitable for large programs, and for this reason it is often dismissed by computer scientists, but it was ideally suited for its purpose. The simple syntax meant that it was easy for beginners. For example, no distinction was made between integers and floating point; they were just numbers. As editing was always a problem on hardcopy terminals, each Basic statement had a line number; to make a change, the user retyped the line. Finally, the syntax was carefully designed for very fast compilation. Dartmouth specialized in fast compilers but almost every other version of Basic was interpreted.

Because of its efficient use of hardware, the first commercial timesharing systems were based on Basic. Later, for the same reasons, Basic was the dominant language on the early personal computers. Microsoft's first product was a version of Basic. Unfortunately for Basic's reputation, Microsoft Basic had numerous crude extensions to give the programmer control of the hardware.

As computers became more powerful, Dartmouth steadily extended the language. The final version was an elegant structured language, with an effective choice of procedures, and good vector graphics. The design choices always emphasized simplicity. In the early 1980s we used Basic for the introductory computer science course at Dartmouth and PL/1 for the second. For the equivalent courses at Cornell today, we use Python and Java.

Commercial timesharing systems

Some of the first minicomputers ran small timesharing systems with a dedicated Basic interpreter. In 1972 I moved to the British Open University, which was the pioneer in distance education providing degree programs for students across Britain. The university had a national timeshared network using Hewlett-Packard's HP 2000 Time-Shared Basic. The university had teletype terminals in study centers around the country. They were connected to the computer system by placing an ordinary telephone handset into an acoustic coupler. The transmission speed was 110 bits per second. The Hewlett-Packard computers could each support 32 simultaneous users. There was one computer at the Open University's headquarters in Milton Keynes, and I think that there was a second system at a center in the north of England. The computer provided only one service, a Basic editor and interpreter. The version of Basic and the command language were essentially the same as Dartmouth's first version.

The first two computing courses that we introduced at the Open University were based on this system, which was a great success. The main difficulty was not technical. Many of the university's students lived long distances from the study centers and we had to design the courses so that the students did not have to visit the centers very often. My wife was one of the first people to have a terminal at home, in the kitchen. In those days, it was the ultimate status symbol of the working wife.

Digital Equipment Corporation (often known as Digital or DEC) created three very different timesharing systems, each of which was widely used in universities: RSTS for the PDP 11, TOPS-20 for the DEC-20, and VAX/VMS. The original RSTS was another Basic system, rather like Hewlett-Packard's, but the final version, for the PDP 11/70, was a general-purpose timesharing system. It supported up to 63 users and was used by smaller colleges into the 1990s. I never used RSTS, but both the Open University and Carnegie Mellon had DEC-20s. Carnegie Mellon had six of them in the central machine room and the computer science department had several more. TOPS-20 had a flexible command language, with a good balance between simplicity and generality. Unfortunately the unreliability of the hardware often let down the excellence of the software. The DEC-20s had an extended version of Basic but Digital's particular strength was its Fortran 77 language. The VAX built its reputation on the fast computation for science and engineering, but the VMS operating system was also good for timesharing. Many liberal arts colleges used VAX/VMS for their academic computing, and in the 1980s it was widely used for departmental computing centers.

Finally, Unix began as a timesharing system at Bell Labs, where it ran on a variety of Digital minicomputers. The version that became a central part of academic computing was the Berkeley Software Distribution (BSD), from the University of California at Berkeley. The definitive version was BSD 4.2 for Digital VAX computers, released in 1983.

Timesharing at Dartmouth

The Dartmouth Time Sharing System

The founders of computing at Dartmouth were two mathematicians, Tom Kurtz and John Kemeny. With help from a team of undergraduates, they created the Basic programming language and the Dartmouth Time Sharing System (DTSS). Kemeny later became President of Dartmouth and Kurtz was the first director of the Kiewit computing center. Under his leadership, Kiewit gained a reputation for outstanding service to the academic community.

I was on sabbatical at Dartmouth in 1976 to 77, and from 1978 to 1985 I was head of computing, with various job titles. When I came to Dartmouth, the Kiewit technical group was very strong. It is hard to single out individuals, but Stan Dunten (networking) and Phil Koch (operating systems and programming languages) were outstanding. I was fortunate to inherit a smooth-running computer center. The key people were Tom Byrne, who ran the business side, and Punch Taylor, who had overall responsibility for the technical work.

DTSS reached its maturity during these years and it is interesting to compare the services that it provided with the networks of personal computers that swept universities soon afterwards. From a modern viewpoint the most surprising feature was that the entire software was developed at Dartmouth. From the device drivers to the applications programs everything was written locally, either at Dartmouth or by DTSS, Inc., the commercial spin off. Most of the original system was written by undergraduates, and much of the maintenance was still done by student system programmers.

A teletype terminal

Early timesharing used hardcopy terminals of the type that were used for sending telegrams. They are often called "teletype" terminals, the name of one popular model, but they actually came from several vendors. They were replaced by the much superior DECwriters in the mid-1970s. Terminals cost between $1,000 and $1,500.

Wikipedia

DTSS began at a time when almost all academic computing used programs written by faculty and students for their own work. For this reason, the operating environment was tailored to compile and run small programs very quickly. Dartmouth provided excellent compilers for PL/1 and for Basic, which reached maturity during the late 1970s. To support up to 200 simultaneous users, the timesharing monitor enforced strict limits on each user's CPU usage, memory, and disk storage, and the system used these resources very efficiently. For example, in the days before virtual memory, the runtime environment solved the problem of running large programs by automatically swapping procedures within a user's memory allocation.

In 1978, DTSS was running on a Honeywell 66/40 computer, the successor to the GE 625 and 635 computers on which it was developed. The hardware was unusual for a third generation mainframe computer in that it had two processors, each consisting of the central processing unit, the memory, and a multiplexor. Each of these six units had a large cabinet, filled with racks of logic boards. Hardware engineers were in constant attendance.

The performance of each processor was about 1 million instructions per second, and the total memory was 2 megabytes. There were about ten disk drives, each the size of a small washing machine. The total disk capacity eventually reached about one gigabyte, which served the entire university. The disk drives were unreliable and backup copies of the data were stored on magnetic tape. A full backup was made once a week with a daily incremental backup.

The communications architecture was an important reason that DTSS was able to support so many users. All terminal traffic was handled by two Honeywell 716 minicomputers that acted as terminal concentrators. To minimize the number of interruptions, they collected keystrokes and sent them to the central computers in batches, usually one line at a time.

By 1978, the standard terminal was a DECwriter. This was a hard copy device that ran at 300 bits per second over an ordinary telephone line. For graphics, we used the Tektronix 4010 family of terminals. These terminals painted a sequence of vectors on the screen that could be deleted only by wiping the entire screen blank. During my sabbatical year I wrote a graphical extension to the Basic system to display vector graphics. A variant of the syntax was incorporated into later versions of the Basic compiler.

Editing is a problem on a hard-copy terminal. The Basic programming language originally solved this problem by giving a line number to each statement and encouraging users to simply retype a line if it needed to be changed. More advanced users were provided with context editors, which made substitutions based on pattern matching.

Video terminals such as Digital's VT 52 and VT 100 came into widespread use about 1980, usually running at 1,200 bits per second. They were dumb terminals with no internal processing.

When they were used for full screen editing, each keystroke had to be processed before writing on the screen. If this processing was on the central computer, as was done by most commercial companies, the transmission and processing times could lead to a frustrating delay after each keystroke. At Dartmouth, we solved the problem of screen editing by building our own terminal, the Avatar, by adding a Z80 microprocessor and cache memory to a standard video terminal. In combination with a special editor on the central computer, the Reactor, this provided an excellent screen editor. In typical Dartmouth style, the hardware and software were both developed locally with Jim Perry as the leader. After I left Dartmouth, the Avatar software was ported to Macintosh and IBM personal computers, and the Redactor was ported to Digital's VAX/VMS.

As I write this description thirty-five years later, the system sounds primitive in the extreme, but many people considered that Dartmouth provided the best academic computing of any university in the world. The university was justifiably proud of its achievements. The technical staff was outstanding and Dartmouth was the benchmark for supporting users who had no interest in technicalities.

Computers in education

DTSS and Basic were simple and easy to use. Basic was never intended as a language for large programs, but was excellent for writing programs of a few hundred lines. Long before I arrived, the mathematics department passed a resolution that every student who took any mathematics course must learn to write simple programs. The department explicitly excused the faculty from this requirement, but many faculty members enjoyed writing programs to support their teaching and research.

These programs were made available through public libraries. There were libraries for specific courses and for disciplines such as statistics. I used to teach a course in computer graphics and had a course library with demonstration programs on topics such as splines and perspective. Other libraries included utilities such as text formatting and Dartmouth had a fine collection of games. Kemeny wrote a popular baseball simulation based on the 1955 World Series triumph of the Brooklyn Dodgers. In anticipation of the modern “open source” movement, these programs were always stored as source code and continually upgraded by colleagues. Whenever we taught a course, it was a matter of pride to extend and improve the programs in the course library.

Dartmouth was unusual in that computing was seen primarily as a service to education and made no charges for the use of the computer. Funding followed what was called “the library model”. Faculty and students were encouraged to use the computer. The center had an annual budget from the university and supplied a range of services at no cost to the user. I spent seven years successfully postponing demands from the central administration that we should introduce a charging scheme.

Kemeny and Kurtz’s great achievement was to allow everybody to be a programmer. Modern computers provide a magnificent set of features, but modern user interfaces and networked communication are much more difficult to program. Many of today’s faculty are skilled computer users, but it is much less usual to write a simple program the evening before a class.

The limitations of timesharing

Every computer system is a compromise. Flexibility and generality come at the price of simplicity. General purpose systems, such as IBM’s MVS and Microsoft’s Windows, are inevitably cumbersome. Timesharing’s aim was to give users the illusion that they each had sole use of a powerful computer and for most users DTSS did this well. It was responsive and easy to use. Even under heavy load it allocated resources so efficiently that we never had to ration computing resources or charge users. But these virtues came at a price. Two important groups of academic users were poorly served.

Firstly, there was little support for the number crunching that is so important in science and engineering. DTSS’s Fortran compiler was an afterthought, and there was no way to allocate large amounts of computer time to individual researchers, even if they had grant funds to pay for it. At a liberal arts university this was a serious problem and at a research university it would have been a fatal flaw. Many universities, where researchers and administrative computing shared an IBM mainframe, never adopted central timesharing.

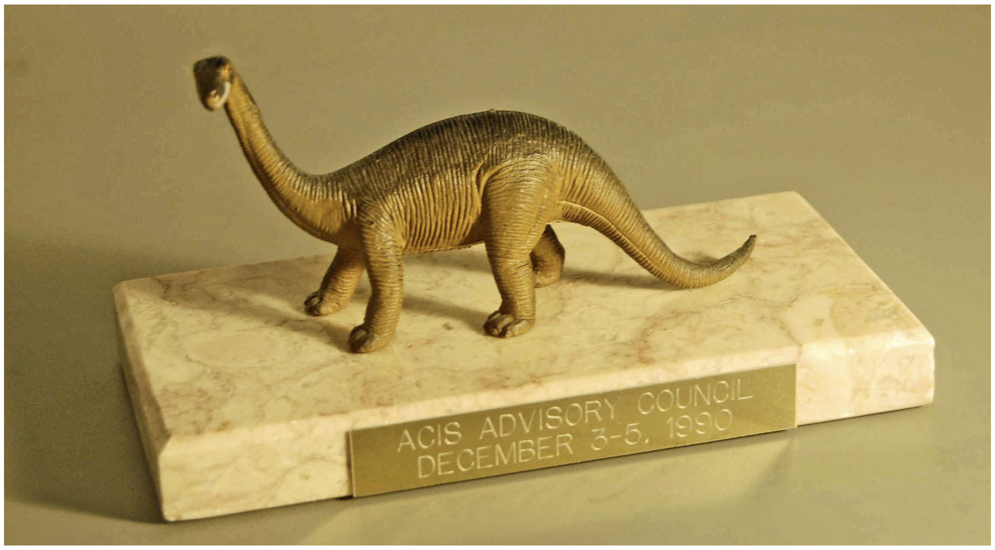
Secondly, every program had to be written locally. Dartmouth built up an impressive public library, but academic computing was steadily moving away from program development to large commercial packages. The inability to run packages such as SPSS for statistics was one of the forces that led Dartmouth away from its own system.

For a while, Dartmouth addressed these two problems by augmenting the central timesharing system with minicomputers, which were used for number crunching and to run software packages, but minicomputers could not solve the fundamental weakness of timesharing, which was capacity. Timesharing was swamped by its own success. Every year more and more users wanted to run more complicated programs that required more terminals, more processor cycles, more memory, and more disk space. The central service could never keep up. Even the best systems slowed down when heavily loaded and made mockery of the illusion that the user had sole use of the computer.

When personal computers arrived there was no illusion and once powerful workstations became available timesharing was doomed. The capacity problems were solved by everybody buying their own computers. Timesharing systems, such as DTSS, became file servers and email hubs. Eventually they were replaced by server farms. But for twenty years timesharing was the leading edge of academic computing

Chapter 3

The Organization of Computing in Universities



A dinosaur from IBM

As personal computers replaced mainframes, the mainframes were colloquially referred to as "dinosaurs". This dinosaur was a gift from IBM after an advisory board meeting.

Photograph by William Arms

Departmental Computing Centers

Minicomputers

During the 1970s, the economies of scale in computer hardware were gradually reversed. I do not understand the engineering reasons that caused this, but it coincided with the development of large-scale integrated (LSI) semiconductors for processors and the phasing out of magnetic core memory. A new type of computer emerged, known as minicomputers, and a new group of computer companies. The market leader was Digital Equipment Corporation. By the late 1970s, minicomputers such as Digital's PDP 11/70, for timesharing, and VAX 11/780, for number crunching, were as cost-effective as the large central computers, and for some tasks were clearly superior

From the 1960s IBM had dominated mainframe computing in the United States, though other companies did well in certain markets, such as Burroughs in banking, and Univac and Control Data Corporation in scientific computing. The mainframe companies were popularly known as "IBM and the Seven Dwarfs" (Burroughs, Univac, NCR, Control Data Corporation, General Electric, RCA, and Honeywell). None of them foresaw the emergence of minicomputers and a new group of companies replaced them. For universities, Digital was the most important minicomputer company, but other companies such as Data General, Hewlett-Packard, Wang, and Prime were also successful. Data General's Nova was popular for computer graphics, and Wang was the leader in word processing and office automation. Many of these companies were spin-offs from MIT and scattered around Route 128 outside Boston.

For most of the 1970s the most widely used family of minicomputers was the Digital PDP 11. The smaller members of the family were used to control laboratory equipment and the larger ones could run a substantial time-shared system. At the end of the decade, 32-bit minicomputers such as Digital's VAX 11/780, the Data General Eclipse, and the Prime 750 had completely reversed the economies of scale. At Dartmouth, we had a Prime 750 for medium-scale number crunching and a variety of VAX and Prime computers for specific applications, such as the library's online catalog and the alumni database. At larger universities, research groups set up their own departmental computing centers. The first book to popularize the high-paced culture of computing was *The Soul of a New Machine* by Tracy Kidder, which described the rush to bring the Eclipse to market in 1980.

These minicomputers had excellent operating systems. While IBM's MVS operating system grew bigger and bigger until it became a nightmare to install and upgrade, the Digital and Prime systems were comparatively straightforward to manage. At Dartmouth we chose Prime because they supported the PL/1 programming language and the X.25 networking protocol, both of which we used for DTSS, but Digital's VAX with its VMS operating system proved more successful in the long term. Several members of the VMS group later moved to Microsoft where they were influential in developing the Windows operating systems.

Departmental computers

Minicomputers had a profound impact on the organization of university computing. For years, the management of central computers had assumed that hardware was a critical resource, particularly central processing cycles. To waste a single cycle was a crime. Only gradually did we come to realize that the time of the faculty and students was much more valuable. When I was on sabbatical at Dartmouth, I watched a group of mathematics professors who kept trying the same iteration, with ever-longer run times. Eventually they realized that they were iterating over a singular matrix and it would never converge. I was horrified at the waste of computer time until I realized that they had gained mathematical insights that were worth much more than the computer time. Not many years later I myself ran an integer programming problem for an entire weekend on a Prime 750. It would have been irresponsible to do so on a shared computer, but not on a machine that would have otherwise been idle.

When a university had a single central computer, the hardware determined the organization. A large central computer needed a team to manage it and to share its resources among departments and users. Almost every university created a computing center with its director. Most of these centers served both academic and administrative users. The center would have a system programming staff to support the central machine and applications programmers for administrative computing.

The centers never fully solved the problem that different groups of users have different computing needs. Most academic users run large numbers of small jobs, but some people want to run big computations or process very large sets of data. Administrative computing has an entirely different set of needs. In aggregate the capacity of the central computer was never enough to satisfy everybody. With varying success, computing centers attempted to balance priorities by technical, administrative, and financial mechanisms, but they could not make everybody happy.

For researchers with research grants, this unhappiness was aggravated by a peculiarity of how many universities charged for computer time. Research universities wanted to recover the cost of research computing from funding agencies, such as the National Science Foundation. These universities charged all users for machine time. Researchers used their grants to cover the costs, while other departments paid from their own budgets. To recoup as much money as possible from grants, the universities set their computing charges at the highest rate that the government would allow, including full overhead recovery.

Frustrated by these high charges and the inflexibility of central computing, the richer departments used their research grants to buy minicomputers and set up their own computing centers. Staff costs were largely eliminated by having graduate students look after the systems. When I arrived at Carnegie Mellon in 1985 the university claimed to have more VAXes than classrooms. There were well-run centers in computer science, electrical engineering, physics, statistics, and several other departments. Later, as personal computers became widely available, schools and colleges, such as fine arts and the business school, set up personal computing centers.

Many computer directors felt threatened by these developments. Some of these feelings were justified. The cost comparisons were often dubious, since departments did not include the indirect costs of running a computer center, which are substantial. The typical departmental computer was run by an assistant professor and a graduate student. Only too often the student never graduated and the assistant professor did not get tenure, but the researchers clearly saw departmental centers as an effective way to spend their resources. An engineering professor at Dartmouth explained to me the advantages of controlling his own computing, free from the juggling act that is inevitable with a central computer that serves the entire university. Recently, for my research at Cornell, I was in a similar position. Our group had its own large computer cluster and we made all the decisions about how to use it.

Computer centers were also worried about the costs of supporting large numbers of different types of computers. Undoubtedly such proliferation did cause difficulties and universities tried various approaches to limit the variety. When I arrived at Carnegie Mellon there were 101 different makes of computer on campus. One of my first acts was to abolish an ineffective policy where I, as vice president for computing, had to approve all purchases of departmental computers. In later years we developed a short list of personal computers that we supported centrally. Most members of the university bought these types of computer, relying on us for support, but there were always a few mavericks who chose differently. These mavericks were invaluable in evaluating new options.

At the universities that I know best, Dartmouth, Carnegie Mellon, and Cornell, the various organizations eventually learned how to work together and support each other, but even today there remains an underlying tension between centralization and decentralization at almost every research university.

The Computer Industry

Motives and motivation

The interactions between universities and computing companies were beneficial for both. The universities' basic motive was straightforward: money. As a politician once said to me, "I know why you are here. Universities always want something for nothing." Over the years we bought computers at deep discount from Honeywell, Digital, Prime, IBM, Apple, Sun, Dell, and many more. We received gifts of millions of dollars of equipment, and cash grants of millions of dollars for specific projects.

What have the companies received in return? First there are sales. Universities are a major customer. IBM, Digital, and Apple were among the companies that cultivated this market with deep discounts and donations of equipment. In the early days of personal computers, IBM gave universities huge grants of equipment in the hope that the universities would standardize on the IBM PC and write leading-edge software for it. But sales were not the only motivation. Many people in the computing industry are genuinely interested in education and are strong believers in technology for education.

Universities were early adopters of many types of computers and the first organizations to take networking seriously. As such, we were important for companies trying to introduce novel products. When Sun was a start-up, Carnegie Mellon was its biggest customer. Today it is hard to remember that the Apple Macintosh was a commercial flop when introduced in 1984. Without orders from a small number of universities, it would probably have died within the first year.

An apple from Apple

This Steuben Apple was given to the president of Dartmouth in 1984. He passed it on to me. I think that he was embarrassed to be buying computers from Apple, which was seen as a slightly improper upstart company.

Photograph by William Arms

Companies valued our insights. By briefing us on new ideas before they became products, companies could get early feedback. They presented their plans and we were not shy in telling them what we needed. When Bell Atlantic (now Verizon) released their first cell phone service, it was a multi-million-dollar gamble. The vice president in charge told me that the only market research that he trusted was from an experimental installation at Carnegie Mellon. When NCR developed Wi-Fi networking, they came to us for an alpha test. When Steve Jobs showed his NeXT computer to a group of university leaders, we told him that universities would not buy a Unix workstation without the X window manager. He did not like it, but we needed it, and he believed us.

The companies also supported university research. Digital was particularly well organized in supporting external research. Most of their support was equipment grants, but for the right project they were willing to provide large sums of money. Intellectual property can be a problem with corporate research, but Digital allowed universities to keep the rights. Their benefits came from exchanges of ideas with researchers, and from leading-edge software being developed on their systems. One of my major disappointments when at Carnegie Mellon was a large grant from Apple to port Unix and the entire Andrew environment to the Macintosh II, and to integrate it with the Macintosh user interface. This was brilliantly done, but when we came to deliver it, Apple had disbanded the engineering group that had sponsored us and the work was wasted.

Finally, companies are always looking for good recruits. Ph.D. and master students from the best universities are in enormous demand. Later in my career, at Cornell, I felt as though I was running a farm system for the Internet industry, teaching upper-level courses on information retrieval and software engineering, with teams of students doing independent research on an Hadoop cluster. Even in the depth of the recent recession companies were hungry for these graduates. Many times each year a student will ask my advice on which job offer to accept. One Carnegie Mellon student had to choose between a faculty position at Stanford and a central position at a fascinating start-up. He chose the start-up.

I think that the corporate people enjoyed visiting the universities. They would often ask to meet students, see demonstrations of research, or attend events on campus. A colleague at Penn State organized his requests for donations around the football season. Carnegie Mellon was at one time a part-owner of the Pittsburgh Pirates baseball team and we would entertain visitors in the owners' box. A visitor from Stanford University even went back to California and persuaded Apple to take a box at Candlestick Park.

Exchanges of information

Visits to the corporations were enjoyable and productive. We would sign a non-disclosure agreement, the company's engineers would brief us on their plans, and we would provide our feedback. Our local salesmen came with us and often learned facts about their companies' plans that were not usually known to the sales force. Each year a group of us from Carnegie Mellon went to California for a week, spending a day each at Sun, Apple, Hewlett-Packard, NeXT, Adobe, and other companies. Other visits were to Digital's research labs outside Boston and the various IBM locations.

Several companies had formal university groups. The first meeting of the Apple University Consortium was a very special occasion. It was in San Jose in December 1983, before the official announcement of the Macintosh. We were given early information about future products and shown the famous 1984 Super Bowl commercial. The consortium later grew too large and lost its effectiveness, but IBM's university advisory group was always small. Each year it concentrated on a single topic. One year the topic was networking. IBM was throwing its main effort into OSI networking, while also contributing large sums to build the NSFnet, which became the backbone for the Internet. The IBM vice president for networking spent much of the day failing to convince us of the virtues of OSI and not understanding why we preferred the Internet.

Only a few universities had this privileged access to the companies. To reach a wider audience, the companies had booths at the EDUCOM conference and hospitality suites where we could talk informally. These grew increasingly lavish until, after consulting with his lawyer, the IBM head of university marketing sent a letter to his competitors, "Let's not have food fights. We should compete with our products and services, not with the lavishness of our hospitality." In contrast to this sensible attitude, I attended a couple of corporate seminars and was appalled by the lack of serious content and the extravagant hospitality. The worst was a so-called seminar by Bell Atlantic. It was a disgrace: minimal content, but expensive food, drink, golf, and sea fishing.

Salesmen

While at English Electric - Leo I had a first-hand look at how computer systems are sold and learned some useful lessons. One lesson was never to trust demonstrations. Multiprogramming was a major selling point for the System 4 computers but the operating system was incomplete. The company regularly ran demonstrations that showed several programs running at the same time. Card readers chattered away, magnetic tapes whirled, and the line printers churned out paper, but these demonstrations were faked. Actually these three programs were the only ones that could be run together.

A second lesson was that salesmen offer the configuration that will get the order, not the one that will do the job. We were asked to bid on a system for a company that ran off-track betting. They had difficulty finding people to operate their telephone center on public holidays, when there are many horse races. Consultants had recommended a computer system and provided a price estimate. Our salesman configured a bid based on that price and I was asked to write a section about performance for the proposal. My calculations showed that the configuration was hopelessly too small, but the salesman submitted the bid anyhow. Fortunately the contract went to a competitor, Control Data Corporation, who ran into all sorts of problems. Years later, another consultant solved the staffing problem by the simple expedient of moving the telephone center from central London to an area with high unemployment.

Years later, as a potential customer, we received a bid from Hewlett-Packard that revolved around a suite of programs for business data processing. Not only was the memory in the configuration too little for the programs that were offered, the machine that was quoted could not be extended. All this information was carefully buried in the tender.

The most important lesson was always to know how the salesmen were rewarded. At Dartmouth we used Honeywell mainframes. Honeywell may have had a price list, but they paid their salesmen by the dollar amount that they booked. To buy a computer we first agreed on the price and then negotiated on what would be delivered. On one occasion, a salesman booked an option to buy a second machine and Honeywell recorded it as an order. Several years later a truck from Arkansas arrived at the Dartmouth computing center in New Hampshire with a large mainframe computer. Fortunately the operator on duty refused delivery. On another occasion Honeywell delivered the wrong central processor. On a happier note, we completed a complex negotiation with Honeywell just before Christmas one year. As soon as the contract was signed, but not before, the salesman delivered a Christmas present. It was a clock. Like many of Honeywell's products, it was made in Japan.

The best time to buy equipment was the week before the company closed its books, as they were always trying to bolster their sales figures. As a computing director the most fundamental rule is that you stay within budget, but budgets are developed a long time before the funds are spent. Private universities have the flexibility to take advantage of opportunities. Twice at Dartmouth, when everybody else had left for the Christmas break, I sat in President Kemeny's office asking for extra funds to take advantage of a year-end offer. At most state universities every line in the budget is fixed and the universities do not have this flexibility.

If the Digital salesmen met their sales target they received rewards. One year our salesman came to me with a hard luck story. He was one computer short of having met his target for five years in a row. Please would I order a VAX system from him? I could cancel the order a week later. Naturally I refused, but he was an excellent salesman and I called his manager. He got his reward. As I remember it was a trip to Jamaica with his wife.

Aggressive marketing

The marketing strategy perfected by IBM for selling mainframe computers was highly centralized. The salesmen cultivated relationships with the computing directors and the customers' higher executives. In many organizations the purchasing of computers remained centralized, even for minicomputers and personal computers, and the IBM method of selling directly to senior executives continued to be successful. In universities, however, as computing became decentralized, purchasing decisions were made by departments. The minicomputer salesmen understood this and developed relationships with the departments. As Barbara Morgan of the University of California at Berkeley quipped, "When the IBM salesman visits me he asks, 'What is happening at Berkeley?' When the Digital salesman visits me, I ask him, 'What is happening at Berkeley?' "

IBM had a reputation for bypassing the computing professionals and selling directly to the president of an organization. This was called an "end run". I could not understand why a chief executive would listen to somebody on a golf course rather than to the experts in his own organization, but it was very effective. When I worked for a strong president, such as John Kemeny at Dartmouth or Dick Cyert at Carnegie Mellon, this was no problem, but their successors were weaker. Kemeny's successor at Dartmouth had an IBM vice president as a buddy who caused enormous difficulties. He appeared to be ashamed that we did business with less prestigious companies such as Digital and Apple. Cyert's successor cost Carnegie Mellon a large sum of money by throwing out a carefully negotiated contract with NEC for a telephone switch because of an end run by the president of Bell Atlantic. He was a Carnegie Mellon alumnus who hinted that there might be grants to the university.

IBM and Digital were aggressive but they were good companies to do business with. The telecommunications companies were frequently dishonorable. The computing companies honored the spirit of an agreement, even if it was only a manager's letter, but the telecommunications companies were always trying to wriggle out of commitments.

Planning for Personal Computers

Sources, reliable and unreliable

Between 1980 and 1995, networks of personal computers replaced timesharing as the core of academic computing. For much of this period, academic computing and the commercial mainstream continued to follow separate paths, but they eventually converged in the 1990s.

Sources of information about this transformation are scattered and unreliable. Much of what was written at the time described what was planned, which was often not what was achieved. Many of the newspaper articles were university publicity laden with euphoria. *The New York Times* had a correspondent who was well known for presenting his personal biases in the guise of factual reporting. Perhaps the most reliable source would be the notes of Judith Turner from the *Chronicle of Higher Education*, but she has wisely kept her notes private.

Much of what has been written subsequently is pure revisionist history. Individuals and organizations describe what they wish had happened. Wikipedia is frequently a good source of technical details, but the contextual information is often tendentious. In practice we went through a decade of hyperbole, resistance to change, blind alleys, failed plans, and spectacular achievements.

Three books published by EDUCOM are a good contemporary source. EDUCOM was the university computing society. Its publications, annual conference, and corporate associates' program were an important exchange of information. I was a member of the EDUCOM board for most of the 1980s and its chair for six years. Under two outstanding presidents, Jack McCredie and Ken King, EDUCOM was a major conduit for contacts with corporations and the government.

One day in fall 1980 Jack McCredie and I were walking across the Stanford campus. He mentioned that he was planning a monograph on university computing. Ten universities would each write a chapter about their strategic planning. I casually agreed to be one of the authors without admitting that at Dartmouth we were so busy building our computer system that we did not do any formal planning. This was the first of the three books. My wife edited the other two, which were about campus networking and libraries.

Two of Ken King's creations were the Coalition for Networked Information, which brought librarians and computing center directors together, and a networking task force that lobbied for higher education in the development of national networks. Al Gore is rightly ridiculed for his claim to have invented the Internet, but he was the first high-level politician to recognize its potential and a great supporter of these efforts.

Because I served on numerous committees, I often saw planning papers from other universities. About 1980, Stanford wrote a particularly insightful set of papers on topics such as networking, office automation, libraries, and so on, but, for me, the most illuminating was the *Preliminary Report of the Task Force for the Future of Computing at Carnegie Mellon*.

Managing the first personal computers

Personal computers posed a dilemma for universities and their computing centers. After years of struggle, we had learned how to manage timeshared computers. A few years earlier, some computing directors had resisted the purchase of minicomputers by departments, but eventually they accepted that minicomputers were frequently cost-effective. Now they had to face a wave of incompatible personal computers that were often purchased by people with little knowledge of computing. The total expenditure on computing was going up steadily and it was not clear what the university was gaining. For a generation who believed that every computer cycle was precious, it was painful to see computers sitting on desks to be used for only a few hours every day. Some computing directors thought that their mission was to protect the university from this wasteful proliferation.

Gradually people began to realize that personal computers were more than a fad. Admittedly they could not do many of the tasks that the larger machines did well, but every year they became more powerful and the application programs grew better. For example, in summer 1982, a student and I developed a planning model for hardware purchases using Visicalc on an Apple II. The time and effort were a fraction of what we would have taken on a timeshared computer. Above all, people began to appreciate the benefits of having their own machines. I used to compare personal computers to private cars. It might be cheaper and safer for everybody to travel by bus, but individuals pay for the convenience of having their own cars.

The showpiece projects

At the beginning of the 1980s, the academic world was excited by the launching of ambitious computing projects at several universities. The most prominent was the Andrew project at Carnegie Mellon University, which included a massive grant from IBM. Never to be outdone, MIT soon afterwards announced the Athena project, jointly with IBM and Digital. In the same year, 1982, Drexel University announced that every freshman would be expected to have a personal computer, and some time later they put out a press release stating that they had chosen an unnamed computer from Apple (which they described as being supplied with a "house", a typo for "mouse"). This computer was later known as the Macintosh.

Several other universities, notably Brown University and Franklin and Marshall College, had important initiatives, some universities passively accepted the arrival of personal computers on campus, and some universities actively discouraged the proliferation. Where there was no campus-wide project, departments or schools, such as engineering or business, often built their own local networks. Harvard is an example of a university that deliberately held back, waited to see what consensus emerged, and then purchased a very fine network, but Harvard is a rich university and could afford to wait. The pioneers endured a great deal of hassle, but received major grants of equipment and money. Brian Hawkins, a force behind the Drexel program, once observed that the universities that took the lead were usually the universities with strong academic leadership.

Whether students should be required to own personal computers formed a lively topic throughout the decade. At Dartmouth we standardized on the Macintosh in 1984 and at our urging, more than 75 percent of freshmen bought them, but not until years later were students required to have computers. Carnegie Mellon was more typical. Although the university had a huge project with IBM, no explicit requirements were made to buy computers. The campus store sold personal computers from both IBM and Apple, as well as workstations from several other manufacturers, with deep discounts for hardware and software.

Organization for distributed computing

The story of these large projects is incomplete without a discussion of the struggles that the universities went through to fit them into their organizations. Universities always have great difficulty in coming to a consensus, particularly when resources have to be reallocated, and these projects were no exception.

In a talk in 1985, Steven Lerman, one of the joint heads of the Athena project at MIT, described a problem which he called "the management of expectations." Projects such as Andrew or Athena need great initial enthusiasm to marshal the resources that they require. Then follows a long gestation period before any signs of progress are seen, and the first results are a pale reflection of the dream with which the projects began. Carnegie Mellon is an extraordinarily entrepreneurial university in its willingness to tackle new ideas, but the university is not rich. The growth in computing demanded resources of money and space, which were tough to find, at exactly the same time that the Andrew project was struggling to live up to its expectations. Fortunately, President Cyert was personally committed to the project and the resources were found.

As the importance of computing grew, job titles were inflated. At the Open University in the 1970s, the head of academic computing was simply the manager of student computing. At Dartmouth my title changed from director of computing services to vice provost and at Carnegie Mellon it was vice president. These distinctions seemed important at the time.

Whatever their titles, computing directors were in a difficult situation during the transition to decentralized computing. Some saw their duty as protecting the central computing budget from the inefficiencies of departmental and personal computing. Others allowed the central service to atrophy while they invested in networks and decentralized computing. No university had the resources to do both really well. During the transition almost every university changed its computing director. Sometimes the university forced them out. Others left for better opportunities or because of frustration with the university's administration. I had been drawn to Dartmouth and Carnegie Mellon because of two outstanding presidents, John Kemeny and Richard Cyert. Working for their successors was no longer enjoyable and I moved on.

Until about 1980, most computers were purchased with central funds and a handful of people made all the decisions. Only in rare instances were computers seen as personal equipment. A decade later the position had been reversed. In 1990, the annual expenditures on computing equipment at Carnegie Mellon were about $12 million, with only $2 million coming from central funds. Purchases made with personal funds through the computer store were over $3 million, and the rest came from departmental funds or research grants.

People who spend their own money naturally expect to choose their own equipment. Yet, everybody benefits from a certain level of campus-wide coordination. The organizational challenge was to reconcile independence in decision making with sufficient standardization to create a coherent campus environment. At Dartmouth, the campus was dominated by Macintoshes, but there were still large numbers of other computers. At Carnegie Mellon, although no formal commitment was made, the expectation had been that the Andrew project would lead to a largely homogeneous environment dominated by IBM. This did not happen. A tour of the campus in 1990 would have found Apple Macintosh and IBM personal computers everywhere, with hundreds of workstations made by IBM, Sun, Digital, Hewlett-Packard, and NeXT. These computers had been selected by many different people at different times with differing needs and resources. Each selected the equipment which best fitted current needs. Yet, at every university, a certain level of standardization emerged because of the difficulty of supporting too wide a range of equipment. The computing environment was so complex that only a few types of computer could be well integrated into it. Most people bought those types of computers.

Gradually universities came to realize that they needed somebody who would oversee the computing strategy for the entire university. This person was not necessarily the person who managed the central computers and did not have to be in charge of both academic and administrative computing. At Carnegie Mellon I had responsibility for academic computing and infrastructure, such as networking, but not for administrative computing. This division between academic and administrative worked well. I am surprised that it is not more common.

At Dartmouth, the Kiewit Network and the Macintosh project were creations of the computer center and there was never any discussion of creating a new organization. Andrew at Carnegie Mellon and Athena at MIT both began as separate organizations outside the existing computer centers, but thereafter followed different paths. Athena remains a separate unit, but at Carnegie Mellon I transferred the responsibility for deploying Andrew to the old computer center. This required a painful reorganization, but the organizational framework has survived.

The institutional structures for academic computing that have emerged at most universities usually parallel the organization of the rest of the university. The organization that I see today at Cornell is typical of large research universities, partly centralized and partly decentralized. It combines aspects of both extremes: entrepreneurial faculty with powerful and independent deans, and yet a fairly strong computing center that provides shared services such as networking and email to the whole university. This central organization is also responsible for the university's administrative data processing

Chapter 4

Networks

Installing a network

This photograph from the University of Michigan shows the unglamorous side of networking, running cables within and between buildings. In the 1980s, the labor costs of wiring old buildings was more than $200 per outlet.

Photograph by Geotech, Inc.

Campus Networks

The starting point

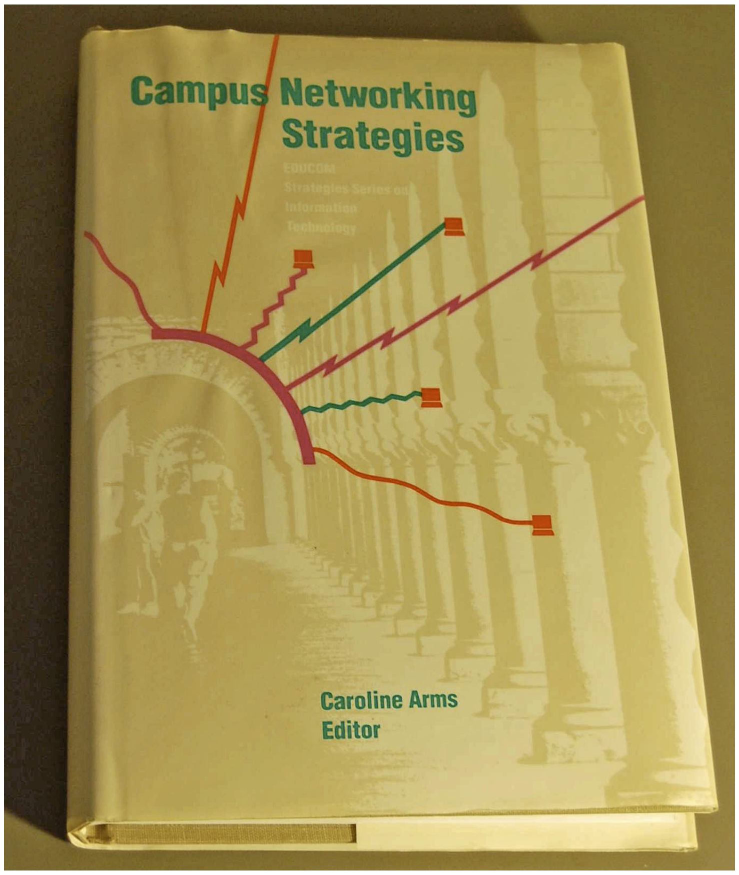
In October 1977, while on my way to be interviewed at Dartmouth, I spent some time at Cornell and contacted the computer center to ask if somebody could discuss Cornell's computing with me. By a spectacular piece of fortune my guide was Douglas Van Houweling, who later became one of the leaders of networked computing. As we walked around the campus on a beautiful fall day he described the need for computer networks in universities.

The concept of networking was not new. Universities already connected large numbers of terminals to central computers. A common architecture was to run groups of asynchronous lines to a terminal concentrator and a higher-speed, synchronous line from there to the central computer. The lines might be hard-wired or dial-up telephone lines. IBM had an architecture known as SNA, which provided hierarchical ways to connect terminals and minicomputers to a central computer, and Digital had built its DECnet protocols into the VAX and DEC-20 operating systems. The ARPANET was well established among a privileged group of researchers, and Xerox had begun the experiments that eventually led to the dominance of Ethernet.

Doug's insight was to realize that these activities were not isolated. At Cornell, departmental computing centers were springing up independently from the main computing center. Very soon they would have to be connected together. Universities would need to have networks that covered the entire campus and connected every type of computing device. They had to use protocols that were not tied to any specific vendor.

Campus strategies

In the EDUCOM book on campus networking, ten universities, large and small, reported on their mixtures of terminal concentrators, SNA, DECnet, Wangnet, X.25, AppleTalk and many more. In the larger universities, the mixture of


Campus Networking Strategies
EDUCOM Strategies Series on Information Technology
Caroline Arms
Editor

Campus networking strategies

The photograph shows the second of the three EDUCOM books about campus strategies. The book about networking was particularly well-timed.

Photograph by William Arms

networks was proving a nightmare. Well-funded areas of the university might have several disjointed networks, while others had none. Providing gateways between networks sounded fine in theory but never seemed to work in practice. Network management was rudimentary and trouble-shooting was expensive. Yet when the chapters were assembled it emerged that, with one exception, the universities had independently decided on the same solution. The strategy was to create a homogenous campus network and to use the Internet protocols for the backbone.

I was responsible for two campus networks. The similarities and differences between them reflect the different priorities and resources of the two universities. At Dartmouth we built a campus network that emphasized moderate performance at a low cost. Because we started early, the problem of incompatible protocols was never as severe as at other universities, and the campus-wide adoption of Macintosh computers allowed us to use AppleTalk as a default protocol. Later, when I went to Carnegie Mellon we built a much more powerful network. The university recognized the strategic value of the campus network and provided a very substantial budget. With generous support from IBM, and with collaboration among many groups at the university, we built a TCP/IP network and connected a wide variety of computers to it. Both networks proved successful and their successors are still in operation today.

Topology and wiring

Networks are expensive. In many universities, although the computing directors and leading faculty members recognized the need for a network, it took years to persuade the financial administrators that universities needed to build them. Even at Carnegie Mellon, part of the justification for wiring the campus was the promise of a much improved telephone system.

A crucial decision was the topology of the campus wiring, within buildings and between them. With several options to choose from, a university could easily build an expensive network that went out-of-date fast. Initially the leading candidates were variants of a bus. Early Ethernet ran on a 50-ohm coaxial cable. This would be snaked through a building and individual devices clamped onto it. IBM had a token ring architecture. AppleTalk used a variant of a bus in which devices were daisy chained together using special cables. An Ethernet bus or a token ring were possibilities for the backbone between buildings, but another candidate was to use 75-ohm video cable with broadband modems.

A bus appears appealing, but has some serious difficulties in practice, and the topology that emerged was a star shape. Within buildings, communication lines were run back to a building hub, and from there lines were led to a central hub. Ethernet and other protocols were modified so that they could run over these lines. The star-shaped configuration has many practical advantages. If part of the network becomes overloaded, extra capacity can be installed by upgrading the equipment in selected building hubs. Almost all trouble-shooting can be done at the hubs. On a bus, when there is a problem, it may be necessary to visit every device.

Eventually star-shaped Ethernet over copper wires became the standard at almost every university and the Internet protocols squeezed out the vendor networks, but for many years we had to support a variety of interfaces and protocols. A later section describes the detailed choices made at Dartmouth and Carnegie Mellon.

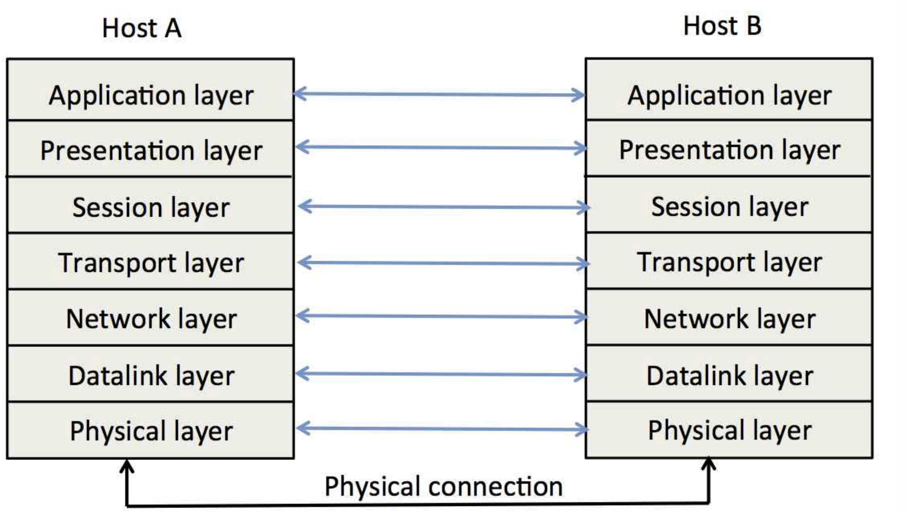
Running wires within buildings was a major expense and the choices were not easy. At Dartmouth, we made a typically pragmatic decision. We decided to use cheap copper wire within buildings, which was easy to install, expecting that it would have a limited life. Fortunately, when the university became interested in our efforts, the trustee who was asked to look at the network was an engineer and he agreed with our approach. However, ten years later, when the time came to renew the wiring, I have heard that this rationale had been forgotten by the new administration. A few years later, Carnegie Mellon made a different decision. The network that we installed there used IBM's cabling system. This was designed to provide a choice of high-quality networking options for many years into the future, for both data and voice communication. It was expensive to install but provided excellent service.

National Networks

The Open Systems Interconnection (OSI) model

By the early 1980s the benefits of networking had become generally recognized, but building the networks proved to be challenging. The developments fell into two categories: the campus networks and the networks between universities. Together they developed into the Internet that we know today. If you read the conventional history of the Internet, you could easily believe that the Internet protocols were so superior that it was inevitable that they would be used for both a national network and for campus networking. At the time, however, it was far from inevitable.

Few people remember that the telecommunications companies and the computing industry fought for years to kill the Internet and its TCP/IP family of protocols. They recognized the potential market and the need for a shared set of protocols, but they wanted to control everything. They embarked on an intense lobbying effort to suppress any competition, using every political resource to stop alternatives, and were particularly outspoken about wasting government money on TCP/IP developments. The telecommunications companies, who have made such enormous profits from the Internet, were the most opposed to its development.



The OSI seven-layer protocol model

Diagram by William Arms

The diagram above shows a popular model that provided a framework for discussing networks. The model divides communication protocols into seven logical layers. Each layer serves the layer above it and is served by the layer below. In this model, Ethernet combines the physical and datalink layers, IP and X.25 are network protocols, TCP is a transport protocol, and FTP and Z39.50 are application protocols.

Rather than adopt the Internet protocols, the industry set out on an ambitious program known as the Open Systems Interconnection (OSI). OSI was supported by all the major companies. Technically it had two main parts: standardization and creating commercial products. IBM alone spent hundreds of millions of dollars. It seemed inevitable that the national and international networks would be built to these standards. Parts of OSI, such as X.25, were successful, but as a whole it suffered from over-complexity and eventually collapsed under its own weight.

Meanwhile universities had difficult decisions to make about what protocols to use on campus and for academic networks. For years, they wrote that their networking plan was to use TCP/IP "until OSI becomes available." Eventually they dropped the final clause. How did this happen?

The first national networks

When universities began to recognize the potential of a national network there were several possibilities. The dominant network between universities was Bitnet. This was a store-and-forward network that ran between IBM mainframes. It used software that IBM had developed for its own internal network. Bitnet provided the first widespread email service between universities and was used for most bulletin boards (called LISTSERVs) until the web became established in the 1990s. As late as 1986, there was no way to send an email message between Dartmouth and Carnegie Mellon except over Bitnet.

The first commercial network was Telenet, which offered X.25 services for a monthly fee. CSnet was an X.25 network developed by those computer science departments who were excluded from the ARPANET. At Dartmouth, we used Telenet for remote terminal access and CSnet for email.

The ARPANET was a higher-speed network developed by the Department of Defense's Advanced Research Project Agency (ARPA). The TCP/IP protocols were developed for the ARPANET. A select group of universities were members of ARPANET, but its use was restricted to people who were doing ARPA-related work. At Carnegie Mellon we were always careful to observe this restriction until I learned that MIT ignored it. They considered that everybody on their campus was doing ARPA-related work.

The success of TCP/IP

One reason that OSI failed was that it attempted to standardize everything. With little operational experience to guide them, the standardization groups added more and more features. The early Internet was much more pragmatic. Its motto was "rough consensus and running code." TCP/IP specified the network and transport protocols, but made no attempt to define the underlying network technology, while protocols for applications such as email were not standardized until there was practical experience with running code.

A key to the success of TCP/IP was the decision to include the protocols in the Berkeley Software Distribution of Unix (BSD). This was also funded by ARPA. BSD 4.2, released in 1983, had an "open source" version of the full family of protocols. This code, which was soon ported to many other operating systems, was the basis for the first generation of the Internet.

In hindsight, the benefits to university computing are obvious. We could use the same protocols on campus and off-campus without the need for complex and inefficient gateways. Because all the versions were based on the same implementation, there were few compatibility problems. TCP/IP came from the computer science research community, which has a tradition of shared problem solving, and it was backed by a well-funded government agency with strong interests in its success.

The NSFnet

The crucial event was the 1985 decision by the National Science Foundation to establish five supercomputer centers for academic research, to connect them with a high-speed network, and to base it on the Internet protocols. Again this was not inevitable. The NSF's newly formed networking group had a hard fight to stop the traditional disciplines from taking their money, and to resist lobbying by commercial interests who were opposed to something that they did not control. It helped that the NSF's effort was led by an Irishman who could charm anybody. The NSFnet backbone was built by a consortium put together by Doug Van Houweling at the University of Michigan with generous support from IBM and MCI. IBM provided the routers and MCI supplied the T-1 connections, which at 1.5 Mbits per second were far beyond the speed of any previous data networks.

The NSF also funded an enlightened program to extend the NSFnet by creating regional networks. Our experience at Carnegie Mellon was typical. The computing directors of the major Pennsylvania universities formed a team led by Gary Auguston of Penn State. His design criteria were, “TCP/IP, T-1, free.” With support from Bell Atlantic, we met with a senior member of the governor’s staff. His first question was, “How will this get the governor re-elected?” His second was, “How can I help?” We then collectively applied to the NSF for a substantial grant. This grant supported the Pennsylvania regional network for long enough to become self-sustaining.

Finally, the NSF successfully transferred management of the network to the commercial sector in 1995. This was a shock to the universities, as henceforward we had to pay for our networking.

Case Studies: Dartmouth and Carnegie Mellon

The Kiewit Network at Dartmouth

Dartmouth can claim to have had the first complete campus network, the Kiewit Network. The architect of the network was Stan Dunten, who had earlier been the developer of the communications for the Dartmouth Time Sharing System (DTSS) and before that a major contributor to MIT’s Multics. His concept was to create a network of minicomputers dedicated to communications, interconnected by high-speed links. He called them nodes. The nodes would be used as terminal concentrators, as routers to connect computers together, or as protocol converters. Earlier, when the university had been laying coaxial cable for television, he had arranged for a second cable to be pulled for networking. This cable was never actually used for its purpose but its existence was an important stimulus.

The first two nodes were the Honeywell 716 minicomputers that were the front ends for the central DTSS computer. They acted as terminal concentrators and provided front-end services such as buffering lines of input before sending them to the central computer.

During the 1970s, Dartmouth sold time on the DTSS computer to colleges that were too small to have their own computers. The first extension of the network was to support these remote users. The front-end software was ported to Prime 200 minicomputers, which had the same instruction set as the 716s. They were deployed as terminal concentrators at several colleges, with the first being installed at the US Coast Guard Academy in 1977. These nodes were linked to Dartmouth over leased telephone lines using HDLC, a synchronous data link protocol.

In fall 1978 the decision was made to build a new set of nodes, using 16-bit minicomputers from New England Digital. Eventually about 100 of the New England Digital minicomputers were deployed on and around campus. The original development team was Stan Dunten and David Pearson, who were later joined by Rich Brown. Each node could support 56 asynchronous terminal ports at 9,600 bits/second, or 12 HDLC synchronous interface at speeds up to 56Kbits/sec. An important feature was a mechanism to reload a node automatically from a master node. After a failure, a dump of the node’s memory was stored on DTSS and the master node sent out a new version of the network software. The master node also performed routine monitoring and individual nodes could be accessed remotely for trouble-shooting.

The first new service was to allow any terminal or personal computer emulating a terminal to connect to any computer on the network. In 1981 the university library connected its first experimental online catalog to the network, which was immediately accessible from every terminal at Dartmouth. The production version of the catalog ran on a Digital VAX computer. Other computers that were soon attached to the network included a variety of minicomputers, mainly from Prime and Digital. The first protocol gateway provided support for X.25, which was used for off-campus connections to the commercial Telenet service and later to the computer science network, CSnet.

Wiring the buildings

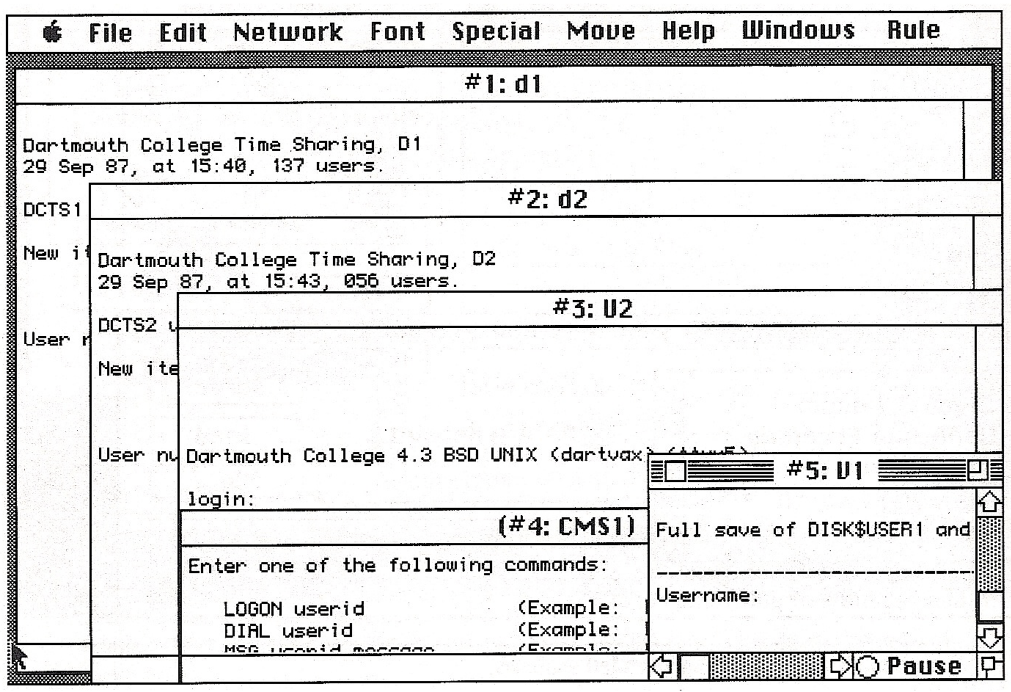
Before the Kiewit Network was built, terminals were connected to DTSS over standard telephone lines using acoustic couplers. Dedicated hard-wires were run where the terminals were close to a front-end computer, such as in the computer center or at the Coast Guard Academy. As the New England Digital nodes were deployed, the number of hardwired terminals increased and as funds became available the campus buildings were wired. The funding model was simple. Hundreds of people around the university were accustomed to paying $16 per month for a 300 bits/second acoustic coupler and a special telephone line. As the hardwired network was gradually installed, users paid $15 per month for a hardwired connection at 2,400 bits/second (later 9,600 bits/second) and the funds were used for the next stage of expansion.

The Dartmouth campus is crisscrossed by roads and in those days the telephone company had a monopoly on transmission lines that crossed the roads. For many years the high prices charged by the telephone company forced the network to use low-speed links between the nodes. At a time when Carnegie Mellon, with no roads across its central campus, was laying its own fiber optics cables, many of Dartmouth's links were only 19.2 Kbits/second. Yet even with these constraints the performance was excellent for the primary purpose of linking terminals and personal computers to central computers.

The Macintosh network

In 1983, Dartmouth decided to urge all freshmen to buy Macintosh computers. This required a rapid extension of the network to support Macintoshes and to wire the dormitories. Before this expansion, the university's central administration had essentially ignored the development of the Kiewit Network. It began as an internal project of the computer center, appeared on no public plan, and never had a formal budget. Now, the university recognized its importance and asked the Pew Foundation for the funds to extend the network to the dormitories. With a generous grant from the foundation, the network was fully deployed by fall 1984, with hardwired ports in every room including the dormitories. The nodes were configured so that each hardwired connection could be used for either AppleTalk or as an asynchronous terminal port. By default the dormitory ports were set to AppleTalk, one connection per student, but could be changed on request.

Originally communication between the nodes used the unique DTSS protocol, but it became clear that something more modern was needed. TCP/IP was rejected for a variety of reasons. The overhead was too high for the slower network links (some off-campus nodes were 2,400 bits/second) and there was no TCP/IP support for the major computers on campus. Before the Internet Domain Name Service was introduced in 1983, whenever a machine was added to a TCP/IP network, a system administrator had to update all the name tables. Dartmouth was considering a stripped down version of TCP/IP when, in spring 1984, Apple showed us the draft protocols for what became AppleTalk. These protocols were very similar to our own design, being packet switched but simpler than TCP/IP. The networking team decided to adopt Apple's protocols and persuaded them to make some changes, notably in the size of the address field. These protocols were adopted for the internal links of the network and the Kiewit Network became a large AppleTalk network in 1984, with the nodes acting as gateways for other protocols.



```
Dartmouth College Time Sharing, D1
29 Sep 87, at 15:40, 137 users.

DCTS1

New i

User r
```

```
Dartmouth College Time Sharing, D2
29 Sep 87, at 15:43, 056 users.

DCTS2 u

New ite

User nu
```

```
Dartmouth College 4.3 BSD UNIX (dartvax)

login:
```

```
Enter one of the following commands:

    LOGON userid            (Example:
    DIAL userid             (Example:
```

```
Full save of DISK$USER1 and
----------------------------
Username:
```

Using a Macintosh as a terminal

This photograph shows five connections from a Macintosh to timeshared computers. Terminal emulation was an important service during the transition from timesharing to personal computing.

Screen image by Rich Brown

There was perilously little time to complete the changeover before the freshmen arrived in fall 1984. It would not have been possible without special support from Apple. The company helped in two vital ways. The first began as a courtesy call by Martin Haeberli who had just completed MacTerminal, the terminal emulator for the Macintosh. For a vital few days he became a member of the Kiewit team. The second was a personal intervention from Steve Jobs. The isolation transformers that protect AppleTalk devices from power surges were in very short supply. Jobs made sure that we were supplied, even before the Apple developers, rather than risk deploying a thousand machines without protection from lighting strikes. AppleTalk was officially released in 1985, six months after it was in production at Dartmouth.

The adoption of AppleTalk for the campus network had a benefit that nobody fully anticipated. Apple rapidly developed a fine array of network services, such as file servers and shared printers. As these were released they immediately became available to the entire campus. Thus the Macintosh computers came to have three functions: free-standing personal computers, terminals to timeshared computers including the library, and members of a rich distributed environment.

Later developments

By building its own nodes, Dartmouth was able to offer the campus a convenient, low-cost network before commercial components became available. It served its purpose well and was able to expand incrementally as the demand grew for extensions.

Even as the AppleTalk network was being deployed, other systems were growing in importance. Scientists and engineers acquired Unix workstations. The business school was using IBM Personal Computers. Ethernet and TCP/IP became established as the universal networking standards and eventually both Dartmouth and Apple moved away from AppleTalk. During the 1990s Dartmouth steadily replaced and upgraded all the original components. 1n 1991 the connections between the building were upgraded to 10 Mbits/second and today they are fiber optic links at 10 Gbits/second. The building wiring was steadily replaced and by 1998 all the network outlets had been converted to Ethernet. In 2001, Dartmouth was the first campus to install a comprehensive wireless network. The last New England Digital nodes were retired many years ago, but the basic architecture that Stan Dunten designed in 1978 has stood the test of time.

The Andrew Network at Carnegie Mellon

As part of the Andrew project at Carnegie Mellon, a state-of-the-art campus network was deployed between 1985 and 1988. By designing this network somewhat later than Dartmouth, Carnegie Mellon was able to build a higher-performance network, but had to accommodate the independent networks that had been installed on campus. The most important of these were a large terminal network using commercial switches, several networks running DECnet over Ethernet, and the Computer Science department's large TCP/IP network. Early in the decade, a working group convened by Howard Wactlar persuaded the university to accept the TCP/IP protocols for interdepartmental communications, but the departmental networks were much too important to be abandoned.

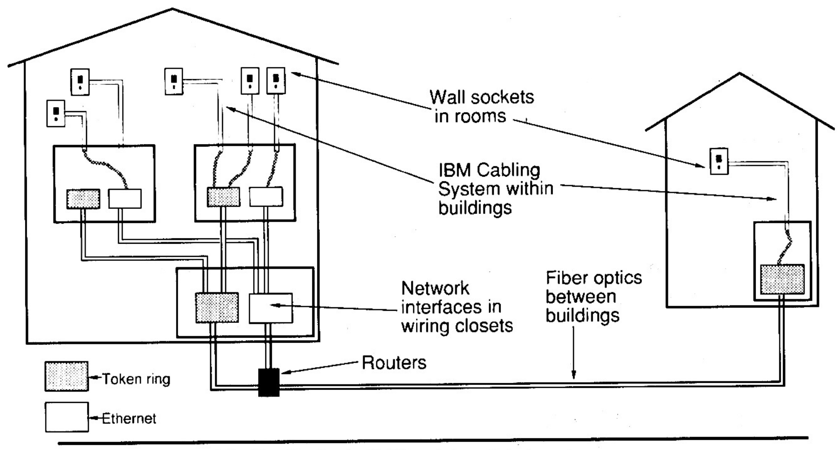
Early Ethernet posed problems for a campus network. The network management tools were primitive and large Ethernets had a reputation for failing unpredictably when components from different manufacturers were used together. To manage the load, the typical configuration was to divide a large Ethernet into subnets with bridges between them which transmitted only those packets that were addressed to another subnet. The bridges operated at the datalink protocol layer so that higher-level protocols such as DECnet and TCP/IP could use the same physical network. This worked well if most traffic was within a subnet, but the Andrew project chose a system architecture that relied on very high performance connections between workstations anywhere on campus and a large central file system, the Andrew File System.

IBM created Token Ring as an alternative to Ethernet. Technically it had several advantages, including better network management and being designed from the start to run over a shielded twisted pair, rather than the coaxial cable in the original Ethernet specification. With the benefit of hindsight we now know that Ethernet was able to overcome its limitations and soon became the dominant standard for campus networks, but at the time IBM Token Ring looked very promising. Since IBM money was paying for much of the Andrew project, the network needed to support Token Ring.

The Andrew Network therefore had to support both Ethernet and Token Ring at the datalink level. Although TCP/IP was the standard, it was also necessary to support the higher-level DECnet protocols over Ethernet. For example, scientific users needed DECnet to connect their VAX computers to the Pittsburgh Supercomputing Center.

Building the network

The network was built by a team led by John Leong. Major contributions came from Computer Science, Electrical and Computer Engineering, and the Pittsburgh Supercomputing Center. Don Smith led the IBM group.


Wall sockets in rooms
IBM Cabling System within buildings
Network interfaces in wiring closets
Fiber optics between buildings
Routers
Token ring
Ethernet

Wiring design used for the Carnegie Mellon network.

The Carnegie Mellon campus network

This greatly oversimplified diagram shows how the campus was wired. Copper wiring was used within the buildings and fiber optics between them.

Diagram by William Arms

Wiring the campus was seen as a long-term investment with the flexibility to adapt to different networking technologies. As the diagram shows, the wiring has a star-shaped topology. The IBM Cabling System was used within buildings, providing high-quality shielded pairs of copper wire. They could be used for either Token Ring or Ethernet protocols. The cables also included telephone circuits. IBM donated the equipment and contributed half a million dollars towards the costs of installation. The wires in each building came together in a wiring closet where there were network interfaces for Token Ring and Ethernet. This provided a Token Ring subnet and an Ethernet subnet in each building. AppleTalk connections were added later. By 1990, there were 1,900 Ethernet, 1,500 Token Ring, and 1,600 AppleTalk ports active.

Leong often described the network as having "an inverted backbone". One spur of each subnet was to the computer center, where they were all connected to a short backbone Ethernet. Routers were used to direct the TCP/IP traffic between the subnets. They were originally developed by the Computer Science and Electrical and Computer Engineering Departments and then the software was ported to IBM PC/ATs. Compared with the nodes on the Dartmouth network, these routers had to handle much higher data rates, but they did not act as terminal concentrators. Separate servers were used as gateways to higher-level protocols, such as AppleTalk.

In practice, there was one major departure from this architecture. Ethernets that carried DECnet traffic were connected to the backbone via bridges, not routers. At the cost of extra load on the backbone, the bridges connected these Ethernets with no constraints on the higher-level protocols.

Connections

To make full use of the network it was desirable that every computer at Carnegie Mellon use the Internet protocols. Support for Unix already existed. The version of Unix that was widely used on campus was the Berkeley Software Distribution (BSD), which included an open source implementation of TCP/IP. BSD was developed on Digital VAX computers. It was standard on Sun workstations and a variant was used by IBM on their Unix workstations.

For the VAX computers, a member of the Computer Science department ported the Internet protocols to run on the VMS operating system. It was distributed for a nominal fee and widely used around the world. This was an early example of the benefits of an open source distribution. Numerous users reported bug fixes that were incorporated into subsequent releases.

The first port of TCP/IP to the IBM PC came from MIT. When we came to deploy it at Carnegie Mellon we encountered a problem. With the Internet protocols, each IP address was allocated to a specific computer. When many copies of the communications software were distributed on floppy disk, we needed a dynamic way to assign addresses. The solution was to extend BootP, a protocol that enabled computers to obtain an IP address from a server. The modern version of BootP is known as DHCP.

In this way, TCP/IP was available for all the most common types of computers at Carnegie Mellon except the Macintosh. The development of TCP/IP for Macintoshes over twisted-pair wiring was a splendid example of cooperation among universities. Carnegie Mellon was one of about eight contributors and at least two start-up companies sold products that came out of this work. TCP/IP service for Macintoshes over the Andrew network was released at the beginning of 1987.

The initial connection of the Andrew network to the external Internet was through the Pittsburgh Supercomputing Center. Since the center was one of the hubs on the NSFnet, the campus network was immediately connected to the national network. Later, a direct connection was made that did not go through the supercomputing center.

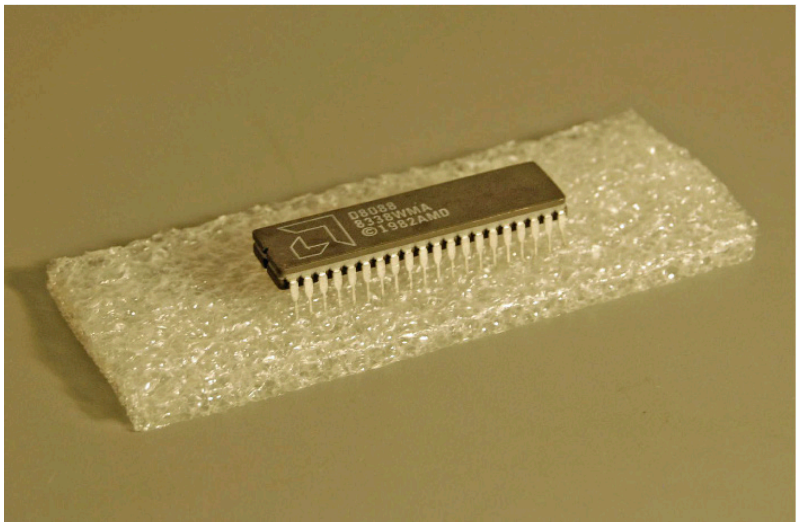
Later developments

In the past thirty years, the network has been continually upgraded. The major developments have been higher speed and greatly increased capacity, beginning with the replacement of the routers and the inverted backbone by high-performance Cisco routers. As the Internet and the TCP/IP family of protocols became widely accepted the need to support alternative protocols diminished. Digital and DECnet no longer exist. Apple uses Ethernet and TCP/IP. Token Ring never succeeded in the market place and was replaced by higher-speed Ethernets. Wired networks are steadily being replaced by wireless networks.

Most of John Leong's team followed the gold rush to Silicon Valley. With their practical experience in engineering a state-of-the-art network, several of them became successful entrepreneurs.

Chapter 5

Networks of Personal Computers



An early microprocessor

Personal computers were made possible by moderately priced microprocessors and memory. This is an Intel 8088 processor, which was used for the IBM Personal Computer in 1981. It has a 16-bit processor but its performance was slowed by an 8-bit external bus.

Photograph by William Arms

Personal Computers and Workstations

Early personal computers

Minicomputers were followed by microcomputers, which soon became known as personal computers. They were made by a new group of companies, such as Tandy, Commodore, Apple, and Acorn, which build the very popular BBC Micro.

The first personal computer that I owned was a Terak. It was a graphics workstation with an elegant Pascal interpreter. The Terak had a 16-bit processor based on Digital's LSI/11 chip set, but most of the early computers used lower cost 8-bit microprocessors. A typical computer of this generation had 4K to 32K of memory, a keyboard, a simple monitor, and two 5-inch floppy disk drives for backup storage. The early monitors were black and white only, but low resolution color monitors followed rapidly. Dot matrix printers from companies such as Paper Tiger and Epson completed the configuration. Tandy's TSR-80 was an early success with hobbyists, but the Apple II was the machine that captured the imagination of schools and universities.

The standard programming language was a simple version of Basic. They were easy machines to program but their success came from the rapid development of a market for software applications. This included general purpose applications, such as word processing, databases, etc. An enormous number of simple programs became available for every level of education. Computer games such as Pacman were a great success, but the application that made the commercial marketplace accept personal computers was the first ever spreadsheet, Visicalc for the Apple II.

Most of the established computer companies missed out on the personal computer and many of them went out of business. IBM was late but when they entered the market they did so with a bang. The original IBM Personal Computer (PC) was only an incremental advance technically, but IBM put its full marketing power behind it. Organizations that had been reluctant to buy personal computers from upstart companies were comfortable buying from IBM. Software developers made the safe choice of developing their products for IBM.

An Apple II+ computer

Most Apple IIs had two floppy disk drives but no hard disk. Floppy disks could be used to distribute software and a software market developed. The computer is running a popular flight simulator.

Photograph by John Miranda

The IBM salesmen had been trained to throw scorn on personal computers and had a hard struggle adapting. At the first Dartmouth presentation, one of our staff took pity on them and fielded questions from the floor while the salesmen took notes. In contrast, the Apple representative was always welcome because she was continually showing us neat things that she did on her own computer.

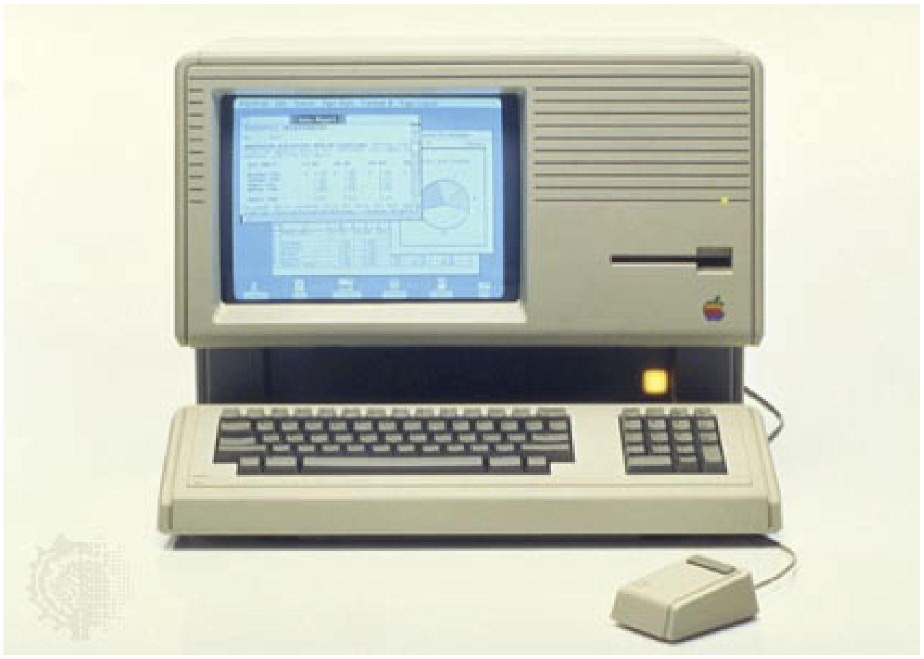
An IBM personal computer

This is a typical configuration of an early IBM Personal Computer. It has two floppy drives and a dot-matrix printer. The addition of a hard disk with the later model XT transformed the usability and performance. Notice the very basic user interface.

German Federal Archives

Most of the early makers of personal computers could not compete with this onslaught, but a new group of manufacturers emerged selling IBM-compatible computers, popularly known as "clones." In their hurry to bring out the PC, IBM used components that they bought from other companies, such as the Intel 8088 processor and a primitive operating system from Microsoft, a small company that was previously known for its Basic interpreter. I think that Compaq was the first company to build an exact IBM clone. Any software that would run on an IBM PC would run on a Compaq. For example, the dominant spreadsheet for many years was Lotus 1-2-3. The very first release stated on the box that it would run on IBM PCs and Compaqs. Although IBM dominated the personal computer market for many years, they never controlled it. When, in 1987, they introduced an incompatible replacement, the PS/2, it was too late. Microsoft, Intel, and the clones controlled the market.

Lotus 1-2-3 is often credited as being the killer application that made the success of the IBM PC. I think that is an overstatement, but it was very important in forcing the clones to be completely compatible. Previously, every manufacturer had its own operating environment, and software vendors were forced to create separate versions for each of them. I recall visiting Informix, who had a relational database system that ran on all types of computers. In the warehouse they had dozens if not hundreds of different versions packed up ready to ship. It must have been a nightmare. When Bill Gates announced that Microsoft would release a product only if they expected to sell at least 100,000 identical binary copies, the standardization was complete.



An Apple Lisa

This is a Lisa II. It has a built-in hard disk and a single floppy. It was a commercial failure, and deserved to be, but compare its modern windowing interface with the text-based interface on the IBM PC.

Photograph courtesy of Apple Computer

The user interface for these early personal computers were much worse than the good timesharing systems. Some applications, such as Lotus 1-2-3, were well designed but they were controlled by a crude command language. The first personal computer with a modern interface was the Apple Lisa, introduced in 1983. The Lisa was over-priced, unreliable, and woefully slow, but it had a few superb applications, such as the LisaDraw graphics program. Four models were built and I had them all.

Workstations

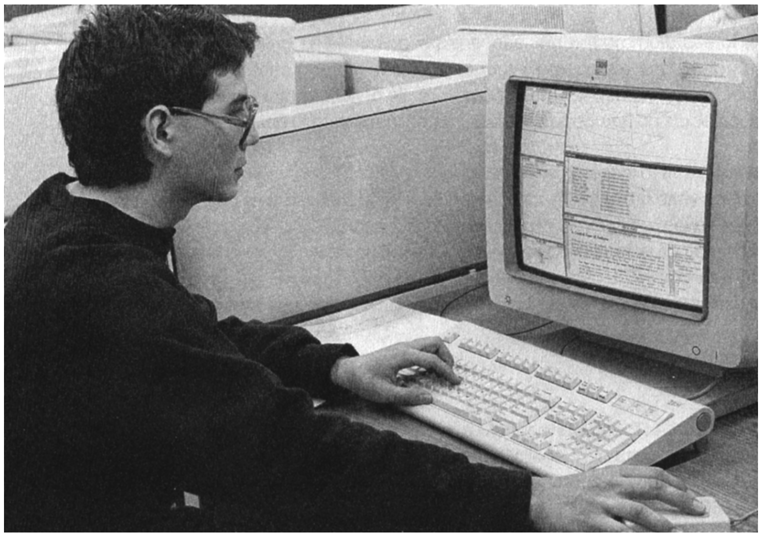
In the mid-1980s there was a rush to build large personal workstations. They were jokingly called "3M" machines, because they had a megapixel display, a megabyte of memory, and a processor that executed one million instructions per second. The early models also had a 10 or 20 Mbyte hard disk and cost about $10,000, which is a million pennies.

The first sales were to engineers and scientists, but prices fell rapidly and sales grew fast. New companies emerged such as Apollo, Silicon Graphics, and Three Rivers Corporation, but they often floundered because of overly ambitious software plans. For instance, the Three Rivers Perq had a beautiful bit-mapped interface but no applications.

Digital had a fast microprocessor, the microVAX, but the company hesitated. Their profits came from minicomputers and they were reluctant to undermine sales of the VAX minicomputers with much cheaper workstations. Unix was developed on Digital computers, but VAX/VMS was Digital's pride and joy, and their support for Unix was always lukewarm. Their later workstations were well engineered, fast, and reasonably priced, but they misjudged the market. Digital, which had been the second largest computer company in the world, never recovered and went out of business a few years later.

The company that came to dominate the workstation market was Sun Microsystems. One of the founders, Bill Joy, was also a principal developer of BSD Unix. Since BSD included the definitive implementation of the Internet protocols, the Sun workstations ran well on the emerging networks. In the early days Sun's business model was to be first to market with advanced hardware and innovative software, often at the expense of reliability. When I joined Carnegie Mellon in 1985 my office computer was a Sun 2. It crashed frequently. This was a mystery until a colleague pointed out that it failed when the sun fell on it in the early afternoon. Later Sun workstations were much more reliable. They were powered by Sun's own SPARC processors.

The Andrew project at Carnegie Mellon, funded by IBM, was built on BSD. Every program that ran on Andrew had to run on standard BSD. When the project began, IBM was not yet ready to ship its own Unix workstation. To everybody's surprise, IBM allowed Carnegie Mellon to buy more than 100 Sun computers for the project.



A Carnegie Mellon student at an IBM RS 6000 workstation in 1986

This computer is running the Andrew interface. The display in the photograph was an experimental display and not available commercially.

Photograph by Kenneth Andreyo

IBM made a marketing mistake in delaying the announcement of their Unix workstation until their engineers were comfortable with the reliability of the hardware and had reworked parts of Unix. As a result they were late to market and the operating system was not quite compatible with BSD. A further mistake was not to offer a high-resolution display with the first release. While IBM was resolving these problems, Sun was consolidating its position as the market leader.

NeXT

Steve Jobs's NeXT computer was a glorious failure. When he was forced out of Apple in 1985 he set out to build a computer with the power of a workstation but the usability of a Macintosh. His target market was education and the business model did not depend on the economies of scale of mass manufacturing.

A NeXT Computer

This photograph is from http://www.johnmirandaphoto.com/next.htm, which has an excellent description of the NeXT computer.

Photograph by John Miranda

Many of the technical concepts were influenced by the several visits that he paid to Carnegie Mellon in 1985 and 1986. When my family and I moved to Pittsburgh we lived for a few weeks in a large house that belonged to Carnegie Mellon. For some of that time, Jobs was also living there. Later I was the first member of the NeXT advisory board, but his main contacts were with members of the Computer Science Department. By them he was persuaded to use Carnegie Mellon's new Unix kernel (known as Mach), and to include a digital signal processor, which made the NeXT computer particularly suitable for applications such as speech recognition.

Technically the NeXT computer was a powerful Unix workstation with a large bit-mapped screen, but it was full of features that were characteristic of Jobs's creativity. It was packaged as an elegant black cube, with few cables visible. When I first visited NeXT in Palo Alto, much of the time was spent in showing off the audio quality and color animations, which were far beyond other machines of the time. Unfortunately, the NeXT machine suffered from another of Jobs's characteristics: it was very late. He would tell you, "It's worth waiting for", but the market did not agree.

Many features of the NeXT computer are now standard but its importance lies in the software. Jobs eventually returned to Apple and the software came with him. Today's Mac OS X is a direct successor to the NeXT operating system. The Interface Builder used to program iPhones and iPads is a descendant of tools built at NeXT to overcome the difficulties that developers had faced in writing programs for the early Macintosh.

The Macintosh at Dartmouth

The dilemma

In the early 1980s, Dartmouth faced a dilemma in deciding how to use personal computers. The success of the timesharing system could have been a barrier to change. It was much more user-friendly than the primitive environments provided by the Apple II or IBM PC; it was much faster for scientific computation; and it was cheaper per user than personal computers. The Kiewit Network already provided a campus email service and access to the library catalog. Yet when we looked at the trends, we saw such rapid improvements in personal computers that they were clearly the path of the future. The Apple II already had a large pool of educational applications and the IBM PC had applications such as word processing and spreadsheets that were far superior to anything that we could produce locally.

Dartmouth had a number of advantages. Dartmouth is large enough to have substantial resources but small enough to be more nimble than larger universities. We had a strong technical staff with close relationships to the faculty. The emphasis on timesharing had made us experts in campus networking. Most importantly, Dartmouth was accustomed to being a leader in academic computing and wanted to remain so.

The solution to this dilemma was to envisage a dual role for a personal computer. A small computer could be used by itself for the tasks that it did well, such as spreadsheets and word processing, but it would also be a terminal to the timesharing system, library catalog, and other network services. Recollections are notoriously unreliable, and I have no records of the process by which this concept emerged, but I remember two key events.

We were all concerned that people at Dartmouth would reject personal computers because of their crude user interfaces, but in late 1982, I heard a leak about a secret Apple project called the Macintosh, and learned extensive technical information about it. When our first Apple sales representative visited us she was horrified to hear of the leak, but after I suggested that we might buy a thousand she arranged for me to visit the Macintosh group in California. I came home convinced that the Macintosh had the potential to succeed at Dartmouth and that Apple was a good company to do business with.

The next event was in spring 1983. The provost, Agnar Pytte, attended a meeting of provosts from other universities where computing strategies were discussed. I provided him with a set of planning slides. One morning soon afterwards he called me to say that he was meeting with the president that afternoon and planned to propose universal ownership of personal computers for Dartmouth freshmen. I hastily put together a short briefing paper. I do not have a copy of that paper, but the main points were to use a personal computer as a terminal to the timeshared computers and the library catalog, to extend the network into every dormitory room, and to use our existing organization to support the initiative. I included an outline budget, which he wisely did not show to the president, but otherwise this paper became the basis for our personal computing plan. Pytte's idea was to begin with the class that entered in fall 1985, but because of a miscommunication when we talked over the telephone I gave him a plan for 1984. He noticed the mistake just before his meeting, gave me a quick call, and kept the 1984 date when he presented the plan to the president.

This paper did not specify which type of computer we would select. I was personally enthusiastic about the Macintosh and lukewarm about the alternatives, but the president came from a commercial background and his ideas about computing were fixated on IBM. IBM wanted us to adopt a stripped computer known as the PC Junior, and Digital made a strong effort to persuade us to adopt their Rainbow computer. The Macintosh project was top secret and I was the only person at Dartmouth who had seen one, but eventually Apple was persuaded to demonstrate the prototype to a group of a dozen people. At the end of the demonstration, Pytte said, "I think that the humanities faculty would like that," and the decision was made. It was typical of Dartmouth that the user interface was the decisive factor in the decision.

An early Macintosh

This picture shows the distinctive shape of the first Macintoshes. It had 128 Kbytes of memory and a single 3½" diskette with a capacity of 400 Kbytes. The small size helped when using the computers in dormitory rooms.

Photograph by Stuart Bratesman, © 1984 Stuart Bratesman - All rights reserved

It is hard to recall just how primitive the Macintosh was when it was released in winter 1984. It was slow, with very limited memory, and had only a single diskette, so that copying files was nearly impossible. There were only two applications: MacWrite, a word processor that crashed when it ran out of memory, and a pixel editor called MacPaint. A decent spreadsheet, Microsoft Multiplan, followed soon afterwards, but the terminal emulator, MacTerminal, was not released until late summer.

Yet the Macintosh had some features that appealed enormously to a segment of the academic community. First, it appealed to the eye. The machine itself was cute, the screen display was attractive, and accessories such as the traveling case were well designed and well made. For the first time we had a computer that was intuitive to use. When he first showed me the machine, Dan'l Lewin of Apple dumped the prototype on the table in its traveling case and went to fetch coffee. By the time that he returned I had set up the machine and was moving windows about on the desktop. I still have the coffee mug.

From the day that it was launched, the Macintosh was a joy to use. It was a complete transformation from the command line interfaces used by other computers. Microsoft took ten years to produce a similar interface with Windows 95. Text is central to academic life and the Macintosh was the first moderately priced computer to support full character sets, including diacritics, and to use bitmapped fonts for both the screen and the printer.

Without acceptance by a few universities the Macintosh might well have died as so many other personal computers did during that period. When commercial companies saw the early Macintosh they saw a toy with very little software that was a brute to program. Their price was more than twice the deeply discounted price that we paid. Companies looked at the Macintosh and did what they had always done: they bought from IBM. As the saying went, "Nobody ever lost their job by buying from IBM."

Dan'l Lewin, the head of university marketing, was the person who saved the Macintosh. When I first met him he showed me the draft of an agreement for what became the Apple University Consortium. The basic idea was to offer Macintoshes to selected universities for a very low price, $1,000. At that time, we were wrestling with contracts from IBM's lawyers who wanted to use mainframe concepts for the sale of personal computers. It was a joy to read Lewin's straightforward draft and be invited to suggest changes that would make it more acceptable to universities. Unlike IBM, Apple made no attempt to state what our community would do with the machines or to guarantee how many we would sell, both of which are impossible commitments for universities.

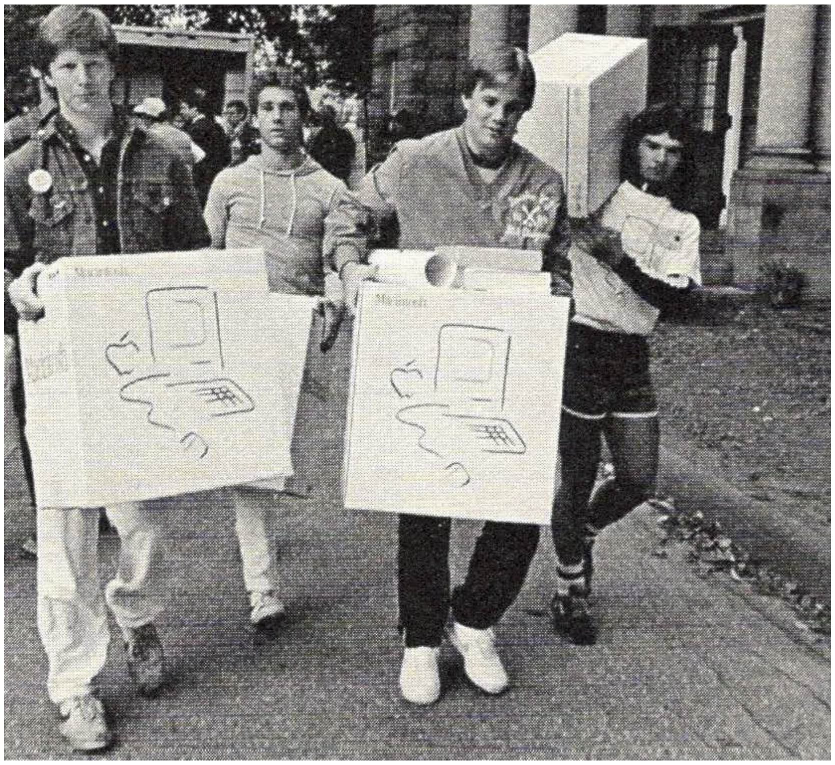
About a dozen leading universities joined the Apple University Consortium before the formal release of the Macintosh, and Apple supported us well over the years. Dartmouth was unusual in that most of the other universities offered Macintoshes as one of several brands of computer that they supported, but we were determined to select one brand and to put all our effort into it. Drexel was the first university to select the Macintosh for all its students, but Apple particularly valued Dartmouth's reputation in educational computing. While IBM and Digital were prepared to make major gifts to gain our business, Apple was more restrained except for the deep discounting. However, they gave us a grant of several machines, which we used as seed machines for faculty to use in education.

The Macintosh at Dartmouth

When Dartmouth decided to adopt the Macintosh, the provost presented the plan to every single committee of which he was a member, but that does not mean that everybody welcomed the initiative. He and I had to endure a fairly rough session at a meeting of the university faculty, and the president never really accepted that we had rejected his friends at IBM.

The first year, 1984, was hectic. The resources of the computing center were focused on the arrival of the freshmen in September, but all the usual services had to be kept running. Converting the network to AppleTalk was a major engineering feat. At the same time, the telecommunications team had to wire the dormitories, install new nodes, and make the cables to connect Macintoshes to the wall outlets. The communications package of MacTerminal and the special cable were slightly late, but were delivered to the students a few weeks after the beginning of the term. Logistics were a challenge. The computing center had a small store selling manuals and supplies. This was transformed into a computer store supplying Macintoshes and IBM PCs, with their software and supplies. Technicians had to be trained in repair work. The handout of the computers to the freshmen was turned into an event, with a convoy of trucks delivering them from the warehouse.

For those of us who were advocates of the Macintosh, the challenge was to balance the long-term potential against the deficiencies, which we hoped were short term. The enthusiasm of a few core faculty helped solve the shortage of good applications. Although the early Macintosh was an awkward machine to program, the qual ity of the QuickDraw tools enabled determined individuals to create good programs quite quickly. We were able to support them with the machines that we had received from Apple and grants from several private foundations

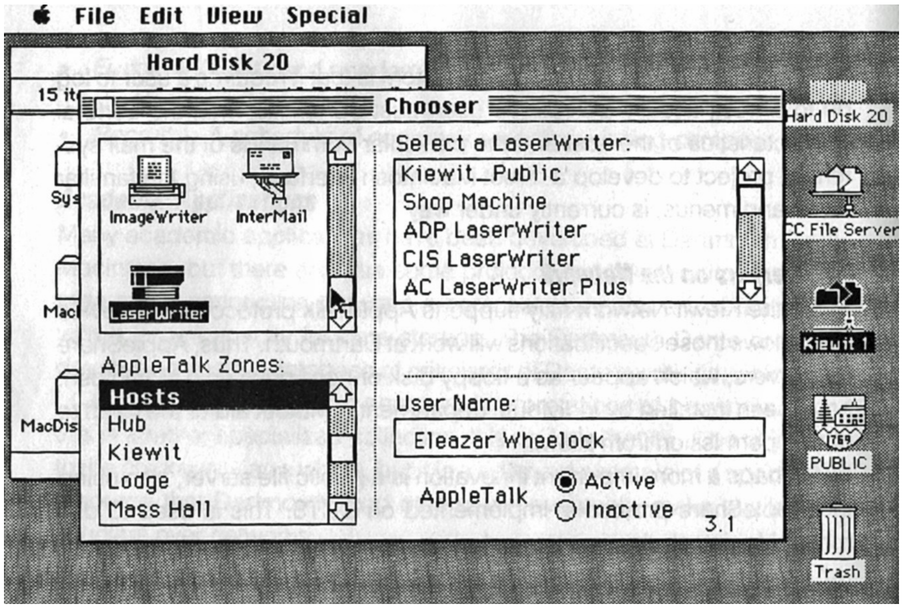


Dartmouth freshmen collect their Macintoshes

In 1984, the standard package was a Macintosh with MacWrite, and MacPaint; a communications package was delivered later. Many students also chose a carrying case and a printer. Students were urged to buy computers, and between 75 percent and 80 percent of the freshmen did so. Owning a computer was made a requirement in 1991, but it did not have to be a Macintosh.

Photograph by Stuart Bratesman, © 1984 Stuart Bratesman - All rights reserved

Apple worked effectively to overcome many of the limitations without compromising the overall design. At the first meeting of the Apple University Consortium, Jobs spoke of several developments, notably the plans for a small laser printer, which was the first for any personal computer. Extending the memory to four times its original size and adding a hard disk solved two crucial problems.



Chooser

Select a LaserWriter:
Kiewit_Public
Shop Machine
ADP LaserWriter
CIS LaserWriter
AC LaserWriter Plus

AppleTalk Zones:
Hosts
Hub
Kiewit
Lodge
Mass Hall

User Name: Eleazar Wheelock

AppleTalk [✓ Active] Inactive

3.1

DarTerminal

This is a DarTerminal screen dump. It shows how Apple's Chooser could be used to connect to computers and services on the Kiewit Network.

Screen image by Rich Brown

Over the next few years Dartmouth used its technical resources to provide networked services that augmented those provided by Apple. The original MacTerminal was replaced by a more flexible program, DarTerminal, and the Avatar distributed editor was ported to the Macintosh. Dartmouth's Fetch program, an FTP client, is still going strong as a file transfer program for Macintoshes. A new mail system, BlitzMail, was so well received that there was dismay when it was withdrawn in 2011. A group of universities collaborated in providing AppleTalk service over dial-up telephone lines. Many of these developments took place after I moved to Carnegie Mellon in summer 1985.

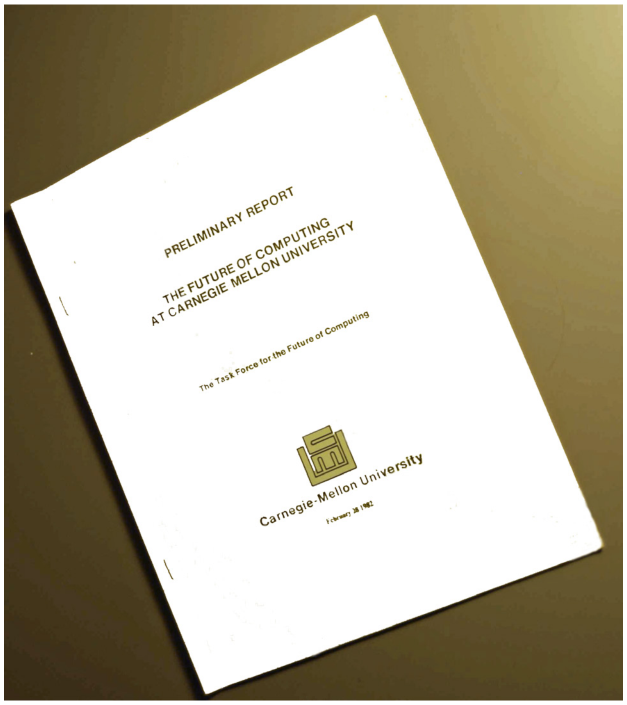
The decision to select one type of computer and support it well proved a great success, but Dartmouth was never exclusively a Macintosh campus. IBM PCs were always preferred by the business school. Other individuals chose them or later the Windows clones. From the earliest days, the computer store sold IBM PCs and the computing center steadily extended the distributed environment to support them. Networked ports in the dormitories could be configured for asynchronous connections and later for Ethernet. BlitzMail and the Avatar were ported to the IBM PC. A report in 1991 estimated that there were about 10,000 computers on campus of which 90 percent were Macintoshes.

Carnegie Mellon and the Andrew Project

Planning

At Carnegie Mellon, the initiative came from the top. The president, Richard Cyert, summarized the university's strategy as "a campus saturated with computing." He saw that state-of-the-art computing could benefit every aspect of the university, including all academic departments, the library, and the administration. He saw research opportunities in every field and the potential to transform education. As a canny businessman he rightly anticipated that an aggressive computing strategy would receive financial support from corporations, government agencies, and private foundations. In his keynote address to the 1986 EDUCOM Conference, Herbert Simon aptly described the strategy as "computing by immersion." Provide faculty and students with a great array of computing and they will use it creatively.

The timing of the plan was based on a deep understanding of trends in computing. In 1979, Allen Newell published a report known as the Spice Report. This report extrapolated hardware trends to predict the spectacular advances in large personal computers that took place during the 1980s. It would soon be possible to build a computer on a single chip that would be as powerful as the super-minicomputers. These large personal computers were often called "workstations".


PRELIMINARY REPORT

THE FUTURE OF COMPUTING
AT CARNEGIE MELLON UNIVERSITY

The Task Force for the Future of Computing

Carnegie-Mellon University

February 28 1982

The Preliminary Report on the Future of Computing at Carnegie Mellon University

Photograph by William Arms

Newell later chaired the Task Force for the Future of Computing, which articulated how Carnegie Mellon could use these advances. The Xerox Palo Alto Research Center had already demonstrated how workstations could be linked by a high-speed network to form a new model of computing, but this was used by only a small number of well-funded computer scientists. The vision was to recognize that this style of computing was appropriate for an entire university and that declining hardware costs would make it affordable. Carnegie Mellon set out to build such a system.

To achieve the ambition, Carnegie Mellon needed an industrial sponsor and they found it in Lewis Branscomb, the chief scientist of IBM. With his support IBM gave the university a five-year grant, later renewed, to build an advanced campus computing environment known as Andrew. The grant was huge. It included an R&D center, the Information Technology Center (ITC) with thirty computer scientists and engineers, very large donations of equipment, support for educational computing initiatives, and social science studies of the impact. A separate grant helped build the campus network.

People

In writing about the Andrew project it is important to stress that this was a university-wide initiative. The various groups that contributed are too numerous to list, but two were so important that their leaders had university titles. Howard Wactlar, Vice Provost for Research Computing, was responsible for the extensive computing facilities in the Computer Science Department. He had exceptional insight into the boundary between research and practice, and how to create partnerships with industry. Michael Levine, a physicist, was one of the founding directors of the Pittsburgh Supercomputing Center. He had the title Associate Provost for Scientific Computing and was a vigorous advocate for those researchers who needed computing facilities beyond those that a Unix-based project could provide.

My predecessor, Doug Van Houweling, was a key person in creating the strong relationship with IBM and recruited Jim Morris to be the first director of the ITC. They were ably supported by John Howard, the senior IBMer on campus. While the ITC created the prototype system and was a catalyst for the rest of the university, components were contributed by many departments, by other universities, and by several manufacturers.

In conjunction with the Andrew project, Carnegie Mellon created the Center for the Design of Educational Computing, under the leadership first of Jill Larkin and later Preston Covey. Year after year, the center received at least one of the annual EDUCOM awards for excellence in educational software.

As the project moved past the prototype stage, responsibility moved to the university computing center. Two people transferred from the ITC to lead this effort: Bob Cosgrove who had responsibility for the servers and the centrally run systems, and John Leong who built the campus network. Without their efforts we would never have overcome the myriad of technical challenges that lie between a research project and a stable production environment. Later, when Cosgrove left and the IBM funding expired, Leong became the overall technical director.

The Andrew project

In 1982 Carnegie Mellon was already a leader in computer science and campus computing. At the time of the Newell report, 7,800 faculty, students, and staff had 1,090 terminals connected to a variety of timeshared computers. The computing center ran six large DEC-20 computers and Computer Science had several more. Numerous departmental centers ran VAXes. Several departments and the computing center operated a large Ethernet using DECnet protocols. The business school became the domain of the IBM PC. The nucleus of a modern workstation environment existed in Computer Science, where 18 Xerox Altos, 44 Perqs, and a Dover laser printer were connected over an experimental 3 Mbit/sec Ethernet. The challenge was to transform this nucleus, so that it would have a major impact on research, on the administration of the university, and especially on students.

When the ITC began work, none of the components that are now considered fundamental to distributed computing existed, except in research laboratories, yet the ITC's aim was to support thousands of personal computers. Ethernet was still experimental; workstations such as the Sun 1 were expensive, slow, and unreliable; Unix was considered impossible to support without an army of system programmers; and the Apple Lisa and Macintosh computers had not yet shown the advantages of a graphical user interface. Nobody had experience with large-scale distributed computing.

The members of the ITC had to work in all these fields and more. They developed a campus-wide file system, a window manager and applications toolkit, and did much of the work in creating the campus network. Morris and his colleagues also recognized that building the technical framework was only part of their task. They had to help people to use it. For this purpose, they supported a number of faculty initiatives, and wrote an excellent mail and messaging system.

The ITC was funded by IBM, but its work was not restricted to IBM equipment and software. The Andrew software was based on BSD Unix and ran on IBM, Sun, and Digital workstations. The port to the Sun 3 had an interesting bug. The Sun 3 was generally very reliable, but it crashed when running the Andrew window manager. A member of the ITC traced the problem to a memory boundary problem with a specific sequence of instructions that Sun's software never used. He reverse engineered the paging firmware, made a minor hardware modification, and fixed the problem. Unfortunately, while doing this he was being paid from the IBM grant; IBM was not amused.

Network services

By themselves, personal computers could not replace timesharing. Timeshared computers shared much more than the central processor. Their central file systems allowed users to store programs and data, to share printers and other peripherals, and to communicate with each other. One of the tasks of the Andrew project was to provide this sharing through network services. The Andrew File System consisted of a number of server computers, each with several large disks. A Unix workstation on the network saw one large file system in which there was no distinction amongst the servers, nor between files stored locally on the workstation and files stored

The Andrew File System

The photograph shows the Andrew File System in 1987. The servers are Sun 2s. Each server has three Fujitsu Eagle disk drives, each with a capacity of more than 400 Mbytes.

Photograph by William Arms

on the file system. In fall 1989, the main campus file system had 15 server computers and more than 30 gigabytes of storage. It was used by 5,000 users every month. In addition to the central file system, which was available to the entire campus, there were several operated by departments.

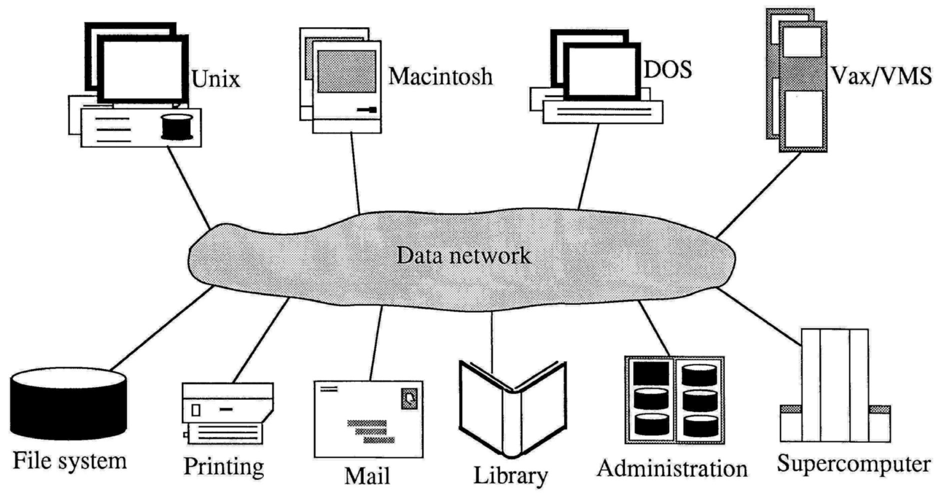
The Andrew message system supported electronic mail and bulletin boards. It was created by the ITC and extended by the computing center. The mail delivery used small post office computers and the user interface ran on personal computers. Mail between users of the Andrew File System was deposited directly into personal mailboxes. Mail to other mail services, including the Internet and Bitnet, was handled by the post office computers. The software had good algorithms for resolving ambiguous or duplicate names, and for handling difficulties. There were separate user interfaces for different types of personal computer. The interface for Unix workstations supported structured text, bit-mapped images, and other formats. Other interfaces were tailored for IBM PCs, Macintoshes, and simple ASCII terminals. Carnegie Mellon was an early advocate of the IMAP mail protocol.

The applications gap

These were tremendous achievements, but there was a gap. It was the same gap that eventually doomed time-sharing at Dartmouth. Unix was undoubtedly the right choice for the system's infrastructure that the Andrew project built, but Unix has always been weak on applications.

Many academic units were opposed to the emphasis on Unix workstations and many who supported the overall vision held back from using the new software until it had proved itself. People refused to commit to Andrew-Unix because the best applications in their areas ran on other computers. Macintoshes and IBM PCs were everywhere; the School of Industrial Administration, and the School of Urban and Public Affairs selected PCs; the new School of Computer Science and the department of Electrical and Computer Engineering used Unix but not the Andrew window manger and tool kit; humanities and social sciences chose Macintoshes; engineering and science departments had a combination of Unix and VMS workstations.

My major contribution to the Andrew project was to recognize this gap and refocus the resources of the computing center to tackle it. To their credit, the IBMers on campus also recognized the situation and were supportive. If today you search the web for Andrew, you will find a narrow description of the Unix software that the ITC developed, but a 1986 brochure from IBM's Academic Information Systems is much broader. Naturally it emphasizes what IBM contributed, but its focus is on networking, and the importance of interoperating with many types of computers and with other universities. In the brochure, I am quoted as saying, "...the plans assumed that there would be nothing else on campus except Andrew. [The concept] was comparable to building an expressway across Wyoming, working with virgin territory, when in fact we were talking about running one through something like Chicago. The unavoidable interrelationship with other computer systems on campus was not foreseen."



Andrew Plus

This is a diagram that I used to describe Carnegie Mellon computing in the late 1980s. The top row shows the supported types of personal computers and workstations. The bottom row shows some of the services that these personal computers could connect to over the network. It does not show the departmental computing centers. Unix computers could connect directly to the networked services. IBM PCs and Macintoshes connected via proxy servers.

Diagram by William Arms

When personal computers were first developed, most people expected that they would supplement timesharing, not replace it. A common question was "what balance would emerge?", but when the central DEC-20 computers were withdrawn in 1988, they were not replaced.

Although personal computers and networked servers dominated the environment, several large computers remained. The largest was the Cray computer at the Pittsburgh Supercomputing Center, operated jointly with the University of Pittsburgh. The library, under the leadership of Thomas Michalak, was a leader in building online information systems. Taking advantage of a lightly used IBM mainframe, the library provided a single user interface to the library's catalog, secondary information services, reference materials, and university information such as the staff directory. Most administrative data processing continued to be done on VAX computers.

Deployment

To build this environment we had to create a new type of organization. Originally, the Andrew initiative had by-passed the computing center and the staff were naturally dispirited. One of my first actions was to reverse the decision to build a separate team to deploy the new environment and to give the responsibility to the computing center.

The transition from timesharing to personal computing was an enormous wrench. The mature and well-loved DEC-20 systems received little attention while the new environment was developed. In 1988, when they were withdrawn, the distributed environment was far from complete. Many components were rough and ready, some were incomplete or temporary, performance was erratic, and many skilled people were needed to carry out tasks that later were routine. In a 1989 survey, students ranked computing well ahead of all other student services at the university, yet several faculty reports were disappointed at the impact on education. Carnegie Mellon was probably the most computer-intensive university in the world, yet shortage of resources was a constant complaint.

The planning papers rightly forecast the enormous demand for personal computers and gave high priority to ensuring that everybody had access to one. In 1981 there was one terminal on campus for every eight people. In 1990 there were as many computers as faculty, students, and staff combined. Amongst faculty, computer usage was almost universal.

As the Andrew project gathered momentum, many of us wanted to require all students to own a computer. The idea was discussed on several occasions, but the university was never prepared to add the cost of a personal computer to the high cost of tuition. Many students, however, had their own computers and the percentage grew year by year. A survey of freshmen in fall 1989 found that 55 percent either owned a computer or were planning to buy one. In addition, the university operated public computing laboratories with almost 700 personal computers for the students. Half were maintained centrally and half by departments. The 1989 survey found 210 Unix workstations, 269 Macintoshes (including 46 large Mac IIs), and 210 IBM personal computers in public laboroties. For personal ownership, the most popular computer was the Macintosh, and every year the university sold about 1,200 Macintosh computers through the campus store. People turned to the Macintosh and the IBM PC as a straightforward way to get their work done, and were delighted by the stream of innovative software developed by the commercial market.

Reflections

The announcement of the Andrew project generated a wave of publicity in both the press and television. In the hyperbole, Carnegie Mellon forgot to manage expectations. Impossible predictions were made about the transformation of education that would come out of this project and how the technology would sweep the market.

Years later a colleague at Cornell asked me if Andrew was a success. I was too close to the project to give an impartial answer but here is an attempt.

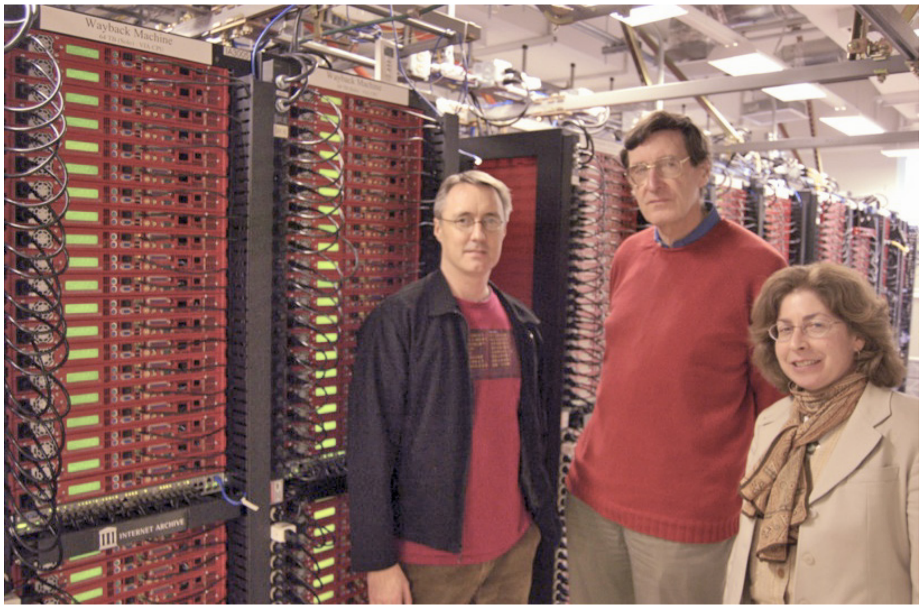
The 1980s were a good decade for Carnegie Mellon by almost every measure: quality of students, research volume, financial strength, and so on. They all improved dramatically and the emphasis on computing was a major contributor. This strength is seen today everywhere on campus.

If the early planning papers are contrasted with what was achieved, the picture which emerges is that the overall strategy advocated in 1982 was remarkably accurate. The forecasts of the tactical steps which would be taken were less accurate. The biggest area not anticipated was the impact of commercial software running on small personal computers. The Andrew Network, the Andrew File System, the message system, and other servers were technical triumphs. For the first time, a major organization ran its computing services without a large central computer. There were also disappointments. The Andrew tool kit and user interface were never widely used, though they had impacts on other projects, and IBM never developed the commercial products that we hoped would emerge from our joint efforts.

At the time we were often swamped by the challenges, both technical and organizational. Now, with the distance of time, it is clear that the strategy of saturating the campus with computing worked brilliantly. Everybody involved can be proud of the outcome.

Chapter 6

Modern Times



A server farm

Personal computers, networked services, the web, cloud computing, and server farms have replaced central computers as the heart of academic computing. This photograph shows a server farm at the Internet Archive in 2009. The Internet Archive is a not-for-profit organization with close ties to academia. Its founder is Brewster Kahle.

Photograph by Felix Weigel

Academic Computing Today

The changing landscape

My role in academic computing changed when I left Carnegie Mellon in 1995. After seventeen years as a computing director I became a user again. With the major exception of digital libraries, I was no longer an insider. This section describes some of the trends that I see as an outsider. Undoubtedly my different viewpoint has obscured important developments, but there seems to have been a fundamental change in academic computing.

The underlying theme of the early years of academic computing was that universities were different. Because the computing industry was not providing the systems that they needed, they built their own. More recently, however, end-user computing has become universal and academic computing has merged back into the commercial mainstream. The projects of the 1980s brought an end to the period in which universities created their own computing environments. The excellent computing that Cornell now provides for its faculty and students is built almost entirely upon standard components. The computer on my lap, the networks it connects to, the programs that I run, the email and web servers are commercial products that are available to everybody.

In 1990, in a review of the Andrew project, I wrote:

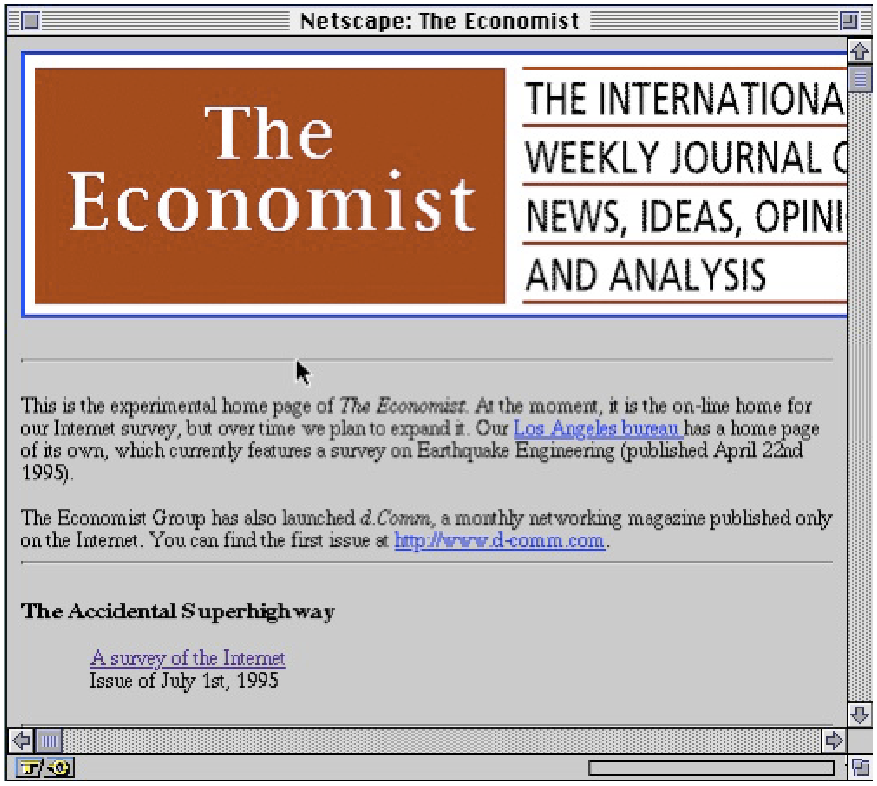
> "As we begin the next decade, nothing on the horizon suggests any change as dramatic as the move from timesharing. The Andrew environment, with personal computers and shared information, appears to be the right model for the years ahead. Many skilled people are needed to refine and improve this environment, but assuredly the next few years will see ever improving computing, which will be based around cheaper and more powerful hardware, better system software, more elegant applications, more dependable service, and a higher level of sophistication amongst users. The university will continue to wrestle with the organizational and financial issues, and will eventually find a balance between decentralized control and university-wide coordination.

"However, the next decade will see much more than an elegant completion to the Andrew project. If we simply extend hardware trends for ten years, we can see that personal computers will execute more than 100 million instructions per second and have screens so good that reading from them will be as pleasant as a printed book, that much of the campus network will be operating at speeds measured in gigabits per second, that it will be cheaper to store almost everything on computer disks than on paper, and that the distinction between video technology and computing will be blurred. These are more than quantitative changes; they are opportunities to open up new areas. The university has vigorous programs in many areas which are poised to take advantage of more powerful computing. These include electronic libraries, speech recognition, natural language processing, chess playing, image processing, and extensive areas of scientific computation. The next decade will see many of these move into the every day life of the campus. Surely, also, the next decade will see developments in new areas that nobody yet foresees."

The final sentence of the quotation has proved abundantly correct. Nobody predicted the uses that would be made of this new world of computing. Less than five years after this sentence was written, the web began its breakneck expansion. Soon afterwards mobile computing began its equally dizzy growth. Both were predicted in general terms, but nobody could imagine their impact. Fifty years ago computing was for highly skilled specialists; twenty-five years ago it was available to the members of well-funded organizations such as universities; today it is universal.

The web

Universities did not create the web, though they were significant contributors. In the early 1990s there were several competing systems for distributing information on the Internet. They included Gopher from the University of Minnesota, the World Wide Web from CERN in Geneva, Z39.50 from the library community, and WAIS by Brewster Kahle, which used a modified version of Z39.50. In retrospect, the greatest strength of the web was its simplicity. Z39.50 failed because it tried to do too much. It assumed that nobody would place valuable information online unless it was protected by an authorization system. The early web was particularly easy to use. I personally created Carnegie Mellon's first home page in about an hour.



This is the experimental home page of *The Economist*. At the moment, it is the on-line home for our Internet survey, but over time we plan to expand it. Our Los Angeles bureau has a home page of its own, which currently features a survey on Earthquake Engineering (published April 22nd 1995).

The Economist Group has also launched *d.Comm*, a monthly networking magazine published only on the Internet. You can find the first issue at http://www.d-comm.com.

The Accidental Superhighway

A survey of the Internet
Issue of July 1st, 1995

An early browser

This is a screen dump from the Netscape browser in 1995. This was the first time that I saw a well-known publication place some of its content on a web site.

Screen image by William Arms

The browser that established the popularity of the web was Mosaic from the University of Illinois, released in 1993. It brought proper fonts, color, and images to the Internet and people loved it. Mosaic was initially developed for Unix workstations but it was rapidly ported to all standard personal computers. The same group at the University of Illinois also developed the web server that is now called Apache.

Most of the web search services, except Altavista, began as university projects or used university search engines, including Infoseek, Lycos, Yahoo (which began as a catalog), and Google, but they all rapidly became start-up companies. More recently, Facebook began at a university but it quickly became a start-up company. MIT was the founder of the World Wide Web Consortium, which has played a major role in standardizing and enhancing the web technology, but overall, universities have been users of the web rather than the creators of the technology.

Open source software

It is hard to exaggerate the importance of open source software in academic computing. Many articles have stressed the benefits of collaboration and the excellence of the best open source software, such as the Linux operating system, Eclipse development environment, Apache web server, Python programming language, Lucene search engine, and the Hadoop-distributed file system and map/reduce engine. The zero cost is of course important, but the availability is even more so.

In my software course at Cornell, I never worry that the students do not have access to the software that they need. Students can download the open source packages onto their own computers and be up and running immediately. Many of the best students never buy any software. They rely on what comes with their computer and what they can download for free. We are educating a generation of students who are experts in the open source packages, but have little knowledge of the commercial alternatives.

Rather surprisingly, universities are not very active in creating open source software. All of the packages listed above are maintained by not-for-profit organizations. Individuals from universities may be contributors, but they get no academic reward for such activities, whereas many corporations contribute staff time as a substitute for building their own software. For instance, Hadoop provides companies that lack Google's expertise with an alternative platform on which to build very large Internet services. Facebook could never have grown so fast without it.

University research and spin-off companies

This is a remarkable time for technology transfer from computing research to the outside world. Ideas developed in universities are making their way into the marketplace very rapidly. Computer science and electrical engineering are in a particularly productive period, where many long-standing research areas have matured into practical products, but people in all disciplines are finding high-powered computing to be a never-ending source of innovation and entrepreneurship.

A dominant theme is the spin-off, a start-up company that individuals create based on their university research. Spin-offs are not new. As early as the 1950s, Stanford University was encouraging the growth of Silicon Valley, while Route 128 around Boston was synonymous with spin-offs from MIT and Harvard. In recent years, the pace has changed as venture capitalists became ever more willing to fund start-ups early in their life cycle. Sun Microsystems, which combined expertise from Stanford and Berkeley, was a good example. Mosaic was developed at the University of Illinois, but a year later, moved to a start-up, Netscape. Search engines such as Lycos (Carnegie Mellon) and Google (Stanford) began as research projects but rapidly formed spin-offs.

Over-exuberance about the potential of the Internet led to a speculative bubble, the dot.com boom, and its stock-market crash in 2000; but the success of companies such as Microsoft, Amazon, Google, and Facebook are an inspiration for individuals to turn their ideas into products and bring them to market. Each of these four companies continued to be led by its technical founders. In the past there was an attitude that companies should be led by people with business and financial backgrounds, but today's students have realized that founders who come from a technical background can be successful entrepreneurs.

Computing in Education

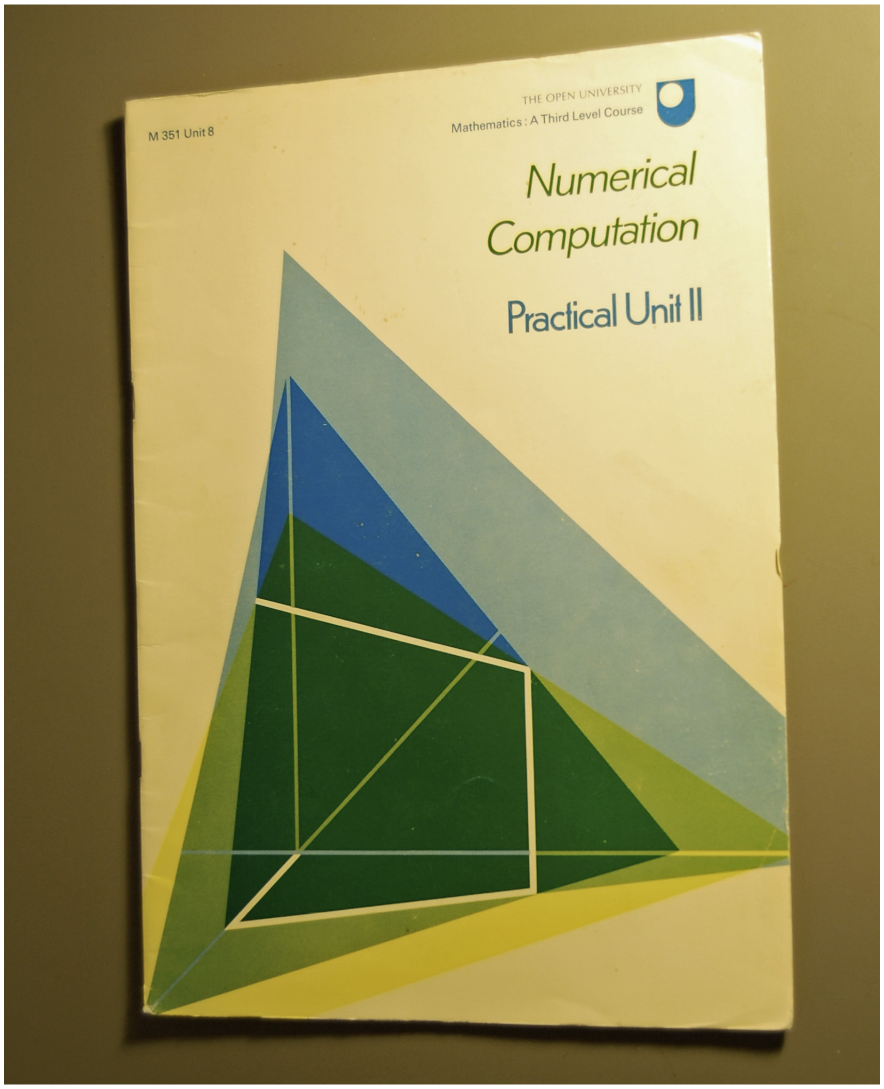
The advocates of every project described in this narrative hoped to have a major impact on education. Dartmouth Basic was designed for students; Athena at MIT and Andrew at Carnegie Mellon were inspired by the goals of better education; when companies such as IBM and Apple gave personal computers to universities they expected to create educational breakthroughs. Meanwhile, every university has its own programs to support innovative courses, and organizations such as the National Science Foundation and the Howard Hughes Medical Institute have well-funded initiatives. During the dot.com boom several universities created web start-up companies. I was a director of eCornell at Cornell.

The talent and energy behind these efforts has been remarkable, but overall the impact has been disappointing. A Carnegie Mellon survey in 1989 found that 44 percent of the faculty used computing in their courses but only 9 percent used programs that had been written for instruction. The real success of computing in higher education comes from faculty offering students the same computing tools that they use for their own research. In addition, we must not ignore the increased productivity provided by simple tools such as word processing, graphics, email, course web sites, and bulletin boards, and the value of online access to the library and other sources of information.

With all the disappointments it is rash to be too optimistic, but there are signs that times are changing. The first sign is financial. Our universities are becoming unaffordable. Partly this is because of inefficiencies, but the underlying cause is more fundamental. Economists call it the cost disease of service industries. An organization that depends on large numbers of well-paid professionals cannot increase its productivity unless it changes its mode of work. As a result it becomes steadily more expensive relative to the overall cost of living. Higher education in the United States is under relentless financial pressure and people are becoming increasingly willing to look for alternatives.

A second sign is that web-based distance education courses are being offered by a wide variety of organizations. Some are of dubious academic merit, but some are excellent. Cornell gives full academic credit for courses offered through the summer school program, and there is increasing willingness to consider other courses that are partially or fully online. We are slowly building an understanding of what works in our culture.

The past few years have seen the introduction of massive open online courses (MOOCs). As usual the hyperbole and enthusiasm have run ahead of the actual achievements, but these courses have some impressive features. Most importantly their advocates are leading faculty from some of our top universities. They include experts in those branches of artificial intelligence that are used to construct courses that need minimal human intervention.



An Open University course unit from about 1975

Photograph by William Arms

My own thinking is strongly influenced by the years that I spent at the British Open University in the 1970s. The Open University does not follow the traditional format of residential education. It was founded to serve adult students, living at home, usually with regular jobs. In the United States this huge group of students is served by a mixed bag of community colleges and for-profit schools, but only too often the result is little education and large debts.

Although the Open University always has had the latest technology available, it is cautious in its use of technology. When I was there, forty years ago, we had a dedicated BBC television studio, but the most effective technology that we had was printed course materials, backed up by human tutors. Nowadays, distance education has few technical barriers. We can assume that our students and faculty have good computers and network connections, and that they are skilled users of them. The technical tools that faculty need to create educational materials are at our fingertips.

Sometime in the future these threads are going to come together and we will see high-quality degree programs based on educational technology. Nobody can tell whether they will be developed by existing universities or by new organizations. The most obvious opportunity is to provide high-quality education for part-time, non-residential students. The developments of academic computing have been led by the elite residential universities, but the greatest benefits may be to people who have never been able to attend those universities.

Sources

In writing this memoir, I began by putting down what I can remember. I then searched for contemporary documents to check my memory of facts and dates. The information that I have been able to find is often incomplete and unbalanced. For example, there is a great deal of information about Multics at MIT, but very little about the Dartmouth Time Sharing System.

Two of the three EDUCOM books cover topics that are covered in this narrative. They are:

Campus computing strategies, edited by John W. McCredie. Bedford MA: Digital Press. 1983

Campus networking strategies, edited by Caroline R. Arms. Bedford MA: Digital Press. 1988

Many of the details about computing at Dartmouth come from the reports and brochures that the Kiewit Computation Center produced intermittently. I have the publications from 1969-71, 1973-76, 1985, and 1986.

IBM's Academic Information Systems division produced a well-balanced brochure on the Andrew project:

Carnegie Mellon University Reaching for World Leadership in Educational Computing and Communications, IBM. 1986

Many of the facts and figures about Carnegie Mellon are drawn from a 1986 brochure and a paper that I wrote when Richard Cyert resigned after nineteen years as president:

Arms, William Y., Reflections on Andrew, EDUCOM Review 25(3):33-43. 1990

Wikipedia has technical articles on many of the computer systems. Some are excellent, e.g., the article on PDP-11 minicomputers, but there are some notable gaps. Wikipedia says very little about how the computers were used and next to nothing on their impact.

Errors and mistakes

Much of this material is my memory of events that happened many years ago. I am sure that there are mistakes. Please send me corrections whether of facts, misunderstandings, or failures to give credit to the right people.

William Y. Arms
Cornell University
wya@cs.cornell.edu