EXHIBIT E

Command Execution in a Heterogeneous Environment*

John T. Korb
Craig E. Wills

Department of Computer Science
Purdue University
West Lafayette, IN 47907

Abstract

As a user's computing environment grows from a single time-shared host to a network of specialized and general-purpose machines, the capability for the user to access all of these resources in a consistent and transparent manner becomes desirable. Instead of viewing commands as binary files, we expect the user to view commands as services provided by servers in the network. The user interacts with a personal workstation that locates and executes services on his behalf.

Executing a single service provided by any server in the network is useful, but the user would also like to combine services from different machines to perform complex computations. To provide this facility we expand on the UNIX notion of pipes to a generalized pipeline mechanism containing services from a variety of servers. In this paper we explain the merits of a multi-machine pipeline for solving problems of accessing services in a heterogeneous environment. We also give a design and performance evaluation of a general mechanism for multi-machine pipes using the DARPA UDP and TCP protocols.

1 Introduction

The Tilde project is concerned with computing problems in a distributed environment composed of heterogeneous machines [4]. The computing environment is heterogeneous with respect to processor type and operating system. As a user's computing environment grows from a single time-shared machine to a network of general purpose machines and dedicated processors, the need for the user to access all of these resources in a consistent and transparent manner becomes desirable. The user would like to have easy access to new resources as they become available without learning any new syntax or protocols to do so. Any changes to the "computing engine" should be reflected to the user in only two ways: (1) as an increase in the number of available services; or (2) faster response for old services that are now available on more machines. Our DASH (*D*istributed *A*ccess to *S*hared *H*osts) project is looking at how the user should perceive his Tilde computing environment and the tools he has to access it [6].

In a time-shared computing environment, the user logs into a machine and interacts with a command interpreter, we use the term *executive*, to perform commands. In the UNIX operating system [9], each command is in fact a binary file that is loaded into memory and executed. In our model of a distributed environment the user interacts with an executive on a personal workstation connected to a local network, and views commands not as binary files, but as *services* provided by *servers* in the network. The user no longer is logged in at a particular host in the network, but accesses the computing engine using the workstation to locate and execute services on his behalf.

Breaking the command–binary file link has three principal advantages:

1. A user does not need to be concerned if a particular binary file runs on a particular machine. If a service is advertised by a particular server then that server knows how to provide the service.

2. Each server can decide what services it offers. This concept fits nicely with the idea of dedicated machines offering specialized services.

3. Service names are independent of file names and a different naming convention, taking into consideration the naming requirements for each resource, can be used.

In our *client-server* model, the user only needs to be concerned with *what* services he wants and not *how* they are provided by a particular server. If a service is offered by more than one server then the workstation can intelligently decide the "best" server to use. A simple heuristic,

*This work was supported in part by grants from the National Science Foundation (MCS-8219178), SUN Microsystems Incorporated, and Digital Equipment Corporation.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1986 ACM 0-89791-201-2/86/0800-0068 75¢

for example, is to choose the server with the lowest "load to computing power" ratio. Each server may provide its services as it wishes, perhaps by executing a binary file, or using a special attached processor to implement the service. The workstation may also provide services, and act as both a client and a server. In all cases the implementation details of a service are hidden from the user and he views his world as a set of services that his executive will locate and invoke on his behalf.

The user can access resources from a wide variety of machines in a uniform manner by viewing all commands as services. Executing a single service provided by any server in the network is useful, but the user would also like to combine services from different machines to perform complex computations. An example might be a numerical analyst who generates some set of data on one machine, sends it to a specialized processor for solution, sends the results to a plot service on another machine, and finally gets the results as a plot on his own workstation. The user does not want to perform each of these operations separately, but instead would like to combine them into a single *pipeline* of actions.

Many systems allow remote execution of commands on other machines in the network [12,11]. Others automatically schedule cpu intensive tasks, such as compiling or text processing, on lightly loaded hosts or idle workstations [2,5,10]. But what these systems often lack is an efficient and simple mechanism for feeding the results of one command to the input of the next. In the worst case the user is forced to store intermediate results in a file and invoke each command in the sequence separately. This method is clearly not satisfactory. In the case of a pipeline mechanism on the user's local machine, the user can combine remote commands using local pipes. This solution is acceptable if the local machine has comparable computing power to the other machines executing the commands, but may become a bottleneck if the machine is a workstation in a pipeline of heavy data flow between powerful server machines.

We want a mechanism to provide pipes, but want them implemented in a general manner so pipelined services communicate data directly and these services can be used in a heterogeneous environment. This requirement leads to a multi-machine pipeline facility, similar to the pipe facility introduced by UNIX, but built on standard transport level protocols. Each service in the pipeline is a *filter* that reads from an (unnamed) input stream and writes to an (unnamed) output stream. Extending this simple concept allows the user to compose computations of services from many machines as easily as he composes commands in the UNIX world. The use of multi-machine pipes also allows services from many machines to communicate data without the need for a common file system, which is an important concern in a heterogeneous environment.

In this paper, we look at a design for implementing an efficient mechanism for multi-machine execution of services, and compare an implementation of our mechanism with pipe facilities available in the Stanford V-System [3,1] and in the UNIX operating system.

2 Motivation

In [13], Zwaenepoel identifies two types of implementations for pipes in a client-server based system:

- Introduce a *pipe server* process between two clients that want to communicate over a pipe.
- Abandon the client-server paradigm for this mode of communication and have the operating system kernel implement pipes.

The advantage of the first mechanism is that it does not require any special kernel support thus making it easier to implement and modify. This solution also retains the client-server model, and if transparent interprocess communication facilities are available, then no additional protocol is needed to support pipes. The disadvantage of this approach is the performance penalty that must be paid by not having a kernel implementation. The kernel approach has better performance, but abandons the client-server model, and requires a protocol for communication between kernels in the case of clients on different machines. In the paper, Zwaenepoel shows measured performance for the pipe server implementation to be 8-25% worse than the calculated value for kernel pipes in the V-System. He concludes that the implementation of a pipe server using message passing, the principal means of interprocess communication in V, is quite practical compared to the additional kernel and protocol complexity needed for a kernel implementation.

We are interested in these results because our initial prototype implementation for service location and execution ran on a SUN workstation using the V operating system. From our prototype we gained experience in using the V pipe server to provide pipes for combining services from different machines. Since the pipe server ran on the workstation, all data traffic between pipelined services was routed through the workstation, as illustrated in Figure 1. Each bi-directional connection is implemented with V message passing, with each outgoing line from the workstation carrying input data for the remote server, and each incoming line carrying both output and error data for the workstation.

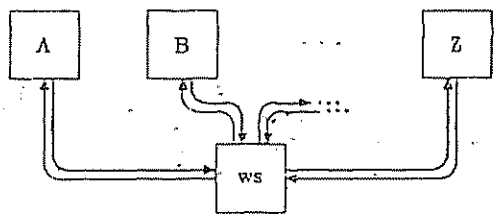


Figure 1: V Pipe Server Pipeline Execution

When the amount of data is large, the pipe server becomes a bottleneck. Our design of a new protocol for handling multi-machine pipes was motivated by the desire to remove the workstation from the data flow path.

Our design is based on standard DARPA protocols and standardizes all interprocess communication at the transport level. Our mechanism is different from both described above in three respects:

- Like the V pipe server, our design follows the client-server model, but unlike the pipe server all data passed between services does not flow through the workstation. Instead, the workstation, acting as the client, communicates with each host in the pipeline to set up TCP [7] protocol connections between services. When the services are executed, the data travels directly between service invocations using TCP connections, with the workstation supplying data to the first service, and receiving data from the last.
- The workstation sets up separate TCP connections with each server so that all error output from executing services is sent directly to the workstation. In the V system, all error messages and output are passed through the same pipeline.
- We do not use V interprocess communication for setting up the pipes, but instead use UDP [8] packets.

The remaining portion of this paper looks at specifics of our mechanism including design, implementation, performance measurements, discussion, and conclusions.

3 Design

To remove the workstation from the data flow path, we have designed a general mechanism for creating a pipe directly between two executing services using DARPA protocols. We assume that each filter-style service in the pipeline follows the UNIX paradigm of reading from a standard input stream and writing to standard output and error streams. This mechanism requires the workstation command interpreter, the executive, to determine what server executes each service, and make connections to the first and last component of the pipeline to handle input and output, as well as a connection with each component to handle any errors that the service may produce. The pipeline configuration at execution time is shown in Figure 2. The thick lines indicate the flow of data through the pipeline, and the thinner lines indicate the path of errors from each component to the workstation.

The mechanism uses UDP datagrams for setting up the pipeline and TCP streams for the pipes between services. Each server machine in the network is required to execute a pipe server that listens at a well-known port for UDP requests. Once the executive determines what server will execute each service in the pipeline, it sets up the pipeline as follows:

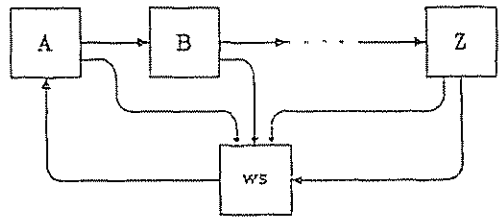


Figure 2: TCP Pipeline Execution

1. Determine a local TCP port for the workstation to listen to for the output of the last service (server Z). Also determine a port to listen for errors from the last service.

2. For each service in the pipeline, beginning with the *last* one, and moving in reverse order, perform the following actions:

   (a) Send a UDP datagram to a server process listening on a well-known port on the host providing the service. This datagram contains the service name and command line arguments. The datagram also contains two TCP (host, port)[1] address pairs to designate where to send the service's output and error data.

   (b) Wait for the server to return a datagram. Meanwhile, the server, listening on this well-known port, receives the incoming datagram, determines a port to listen to for input and sends the port number back to the workstation. The server then must create a process to wait for a TCP connection on the input port. When this connection is made, the process must connect to the output and error ports given in the UDP setup datagram, and execute the service.

   (c) Upon receiving a return datagram, use the port number given in this message as the output port for the preceding service in the pipeline and determine a new local port for receiving errors.

   If a reply is not received from the server, then the executive times out and retransmits the UDP datagram. If a valid reply is not received from the server after a defined number of retransmissions, then the executive connects to the last successful (host, port) pair and closes the connections. This action causes succeeding connections to close as well and "cleans up" the pipeline. An error is reported to the user.

3. After successfully setting up the first service in the pipeline, the executive starts a process that connects to the first server's (server A) listening port and directs all input from the workstation to this port.

[1] A host's internet address and port number determine a unique endpoint for communication called a *socket*.

Then it starts a process to listen on the output port for data from the last command in the pipeline. The executive also starts a process to listen on each error port and collect any errors from the servers.

We based our design on the UDP and TCP protocols because we wanted a general mechanism that could be used by any clients and servers that understand these protocols rather than limiting ourselves to just those that understand the V protocol. Our design requires each server to listen on a well-known port and be able to perform the services it advertises by connecting the service's input, output, and error to the given TCP connections. Otherwise, each server implements its services as it wishes.

This mechanism is very general and can be used by any clients and servers implementing the DARPA TCP and UDP protocols. We find this generality very appealing for work in a heterogeneous environment. We also feel that the design is an efficient mechanism for direct data flow between two services rather than using the user's workstation as an intermediary.

4 Implementation

The server portion of our design has been implemented on VAX, SUN UNIX, and RIDGE machines. The client portion has been implemented as a user program on our VAXes and incorporated into the command interpreter of the V operating system[2] running on SUN workstations. All machines are interconnected by a 10Mbps Ethernet.

V is a message-based operating system using a distributed kernel. A special purpose Inter-Kernel Protocol (IKP) is used for sending messages between processes on the same or different machines. Following the client-server model, the V environment has a server to manage the workstation display, an internet server (providing the IP and TCP protocols), a pipe server, and an executive server among others. Each of these servers usually resides on the user's workstation. In addition, another server (the V UNIX server) executes on back end hosts to provide file access and remote command services.[3] Client programs can access any service by passing messages to the appropriate server.

Within this environment we have implemented a prototype that performs the following service location and execution functions:

- The service location mechanism caches a list of available services from all servers in the network. For VAXes, running the V UNIX server, the workstation communicates with the server using the V protocol, otherwise it uses a TCP-based prototol for communicating with servers on other machines.
- The executive periodically checks the load average on each of the server machines and stores this information locally.
- For each command, the executive chooses the "best" server for execution by checking its local cache for service availability and server load. To execute the service on the user's behalf, the executive communicates with the appropriate server using V or UDP protocols.
- For a pipeline of commands, the executive performs the same service location algorithm for each command, and uses either V or TCP pipes for passing the data between services. The type of pipe used is currently controlled by the user to facilitate taking measurements of pipe performance.
- The services provided by the V UNIX servers are executed in an environment that is maintained at the workstation and passed to these execution servers. The current implementation does not pass the environment to the TCP protocol servers. A default environment is used for services provided by these servers.

[2] We are currently using version 5.0 of V.

[3] The V UNIX server only executes on our VAX machines.

5 Performance Measurements

In this section we look at the performance of executing pipelined services using V pipes versus an implementation of the UDP/TCP mechanism we have described. The measurements were made by repeating each pipeline of commands and computing the average elapsed time. The time includes two parts: (1) the setup time to locate where the commands are to execute and create the necessary connections; and (2) the actual time spent waiting for the pipeline of commands to complete. For this experiment, the server to use for each service was specified to minimize differences in the comparison.

The setup for a V pipeline involves downloading a local (to the workstation) helper program for each remote service to handle input and output for that service. Also, for each pipe, the V pipe server is contacted to create a pipe and the appropriate I/O redirections are made. The executive waits until the last helper program in the pipeline exits.

Setting up a UDP/TCP pipeline requires determination of a local TCP port for the output of the last service in the pipeline, and a local TCP port for the error output of each service. For each component in the pipeline, beginning with the last, the executive contacts the appropriate server and communicates as described. After creation of all connections, the executive downloads a local V program to connect to the input port of the first service, and to listen on all the local TCP ports. The executive waits until this program exits.

The results of setting up and executing services in each of the pipelines are given in Tables 1 and 2. The services chosen require little processing, so that time differences can be accounted for by the particular pipe implementation. The cat service reads the file given in the command line and writes the contents to standard output. If a file

is not given then it simply copies its input to its output. The wc service counts the number of characters, words, and lines in its input and writes the result.

Each service pipeline was executed on files ranging in size from 10 bytes to 1 megabyte. All measurements were taken twenty times (ten or fewer times for the 1 megabyte pipelines) on lightly loaded VAX 11/780 machines. The bracket notation indicates the machine each service was performed on. The setup time is independent of file size, but Table 1 includes all file sizes for completeness.

	setup time in secs (stddev)			
	file size in bytes			
Pipeline Commands	10	1K	100K	1M
[A]*cat file* | [A]*wc*				
V pipes	1.71	1.30	2.01	2.01
	(1.04)	(0.64)	(1.10)	(0.50)
UDP/TCP pipes	1.40	1.76	1.81	1.77
	(0.61)	(0.99)	(0.66)	(0.72)
[A]*cat file* | [B]*wc*				
V pipes	1.52	1.38	1.90	1.28
	(1.07)	(0.67)	(1.01)	(0.32)
UDP/TCP pipes	1.21	1.28	1.22	1.49
	(0.27)	(0.66)	(0.32)	(0.66)
[A]*cat file* | [B]*cat* | [A]*wc*				
V pipes	2.50	2.71	–[a]	–
	(0.90)	(1.11)		
UDP/TCP pipes	1.61	1.81	1.48	1.46
	(0.75)	(0.81)	(0.26)	(0.05)

[a]The V pipe mechanism did not complete.

Table 1: Pipeline Setup Times

	execution time in secs (stddev)			
	file size in bytes			
Pipeline Commands	10	1K	100K	1M
[A]*cat file* | [A]*wc*				
V pipes	3.06	3.24	23.71	196.69
	(0.74)	(0.56)	(1.08)	(2.36)
UDP/TCP pipes	2.72	3.34	7.86	45.58
	(0.60)	(0.24)	(0.38)	(1.04)
[A]*cat file* | [B]*wc*				
V pipes	2.99	3.74	18.28	168.75
	(0.87)	(1.05)	(0.91)	(16.80)
UDP/TCP pipes	2.32	2.42	6.26	31.39
	(0.41)	(0.50)	(1.82)	(2.05)
[A]*cat file* | [B]*cat* | [A]*wc*				
V pipes	4.46	4.03	[a]	–
	(1.08)	(1.24)		
UDP/TCP pipes	3.16	3.35	8.75	60.21
	(0.42)	(0.47)	(0.79)	(4.45).

[a]The V pipe mechanism did not complete

Table 2: Pipeline Execution Times

As a further comparison, we measured the performance of pipes on a UNIX system. The commands were executed on host A (a VAX 11/780) with remote execution of commands on host B performed using the rsh command. Commands were also executed on a SUN workstation running UNIX that used rsh to perform commands on hosts A and B. The rsh command logs the user into the remote machine and sets up the user's execution environment by reading from a standard startup file, then executes the remote command. The remote command reads input, and writes output and errors to the originating host using a TCP protocol connection. To minimize the overhead of the rsh command, we removed the startup file for this experiment. The results are given in Table 3 as the average amount of time taken per pipeline.

One other performance consideration in comparing our mechanism with UNIX pipelines is the implementation of the UDP and TCP protocols in the V system. Just as the V system implements pipes with a pipe server executing outside of the kernel, the UDP and TCP protocols are implemented by servers executing outside of the kernel. Consequently, there is some performance penalty in not implementing other network protocols besides IKP in the kernel.

	total execution time in secs			
	file size in bytes			
Pipeline Commands	10	1K	100K	1M
[A]*cat file* | [A]*wc*				
VAX A	1.21	1.27	5.35	42.46
UDP/TCP pipes	4.12	5.10	9.67	50.35
[A]*cat file* | [B]*wc*				
VAX A	4.64	4.32	8.80	38.86
SUN UNIX	7.17	6.90	11.34	36.79
UDP/TCP pipes	3.53	3.70	7.46	32.88
[A]*cat file* | [B]*cat* | [A]*wc*				
VAX A	4.86	4.78	11.10	70.50
UDP/TCP pipes	4.77	5.16	10.23	61.67

Table 3: Pipeline Mechanism vs UNIX

6 Discussion

Table 1 shows the amount of time to set up a two command pipeline is roughly the same for the two mechanisms, but for three or more commands the V pipeline takes longer to set up. This difference results from the executive downloading a V helper program to handle the input and output of each service in the pipeline. In contrast, the implementation of our design only needs to download one helper program regardless of the length of the pipeline, even though more UDP packets must be sent. Obviously, the cost of exchanging UDP packets with the servers is less expensive than downloading programs to the workstation.

The most obvious observation from the results is the performance penalty incurred in using the V pipe server on the workstation to pass data between services as shown in Table 2. For small amounts of data, the performance difference between our mechanism and V pipes is insignificant, but as the amount of data is increased the difference becomes large. For more data (>100,000 bytes) using just a single pipe, the execution time difference is between 3 and 4 times worse for V pipes than for TCP pipes.

The difference can be explained by looking at the data flow in each implementation. For V pipes, each byte of data must travel from a server to the workstation and on to the next server in the pipeline. For our mechanism, each byte of data travels directly between the two servers. Not only does the V pipe implementation cause more data transmission, but it also forces the V pipe server to handle each byte of data. When the server machines are more powerful hosts, the workstation can become a bottleneck. These factors make our mechanism much better suited for data intensive pipelines.

Not surprisingly, Table 3 shows the performance of pipes on a single VAX machine is much better than for our mechanism because there is no overhead in setting up the pipe and all data is local to the machine. As the amount of data increases the relative difference between the two methods decreases. An interesting anomaly is the performance of our mechanism on a single pipeline of 1 megabyte of data between two VAXes. This pipeline outperforms even a single pipeline on a VAX. The difference might be explained by having two very lightly loaded VAXes instead of just one to handle this large amount of data. In fact, for our mechanism, a pipeline of two commands on different machines almost always outperforms a pipeline on a single machine.

The experiment to execute a pipeline between two VAXes from a SUN workstation running UNIX was made to test our conjecture that the pipeline would be slower because the workstation would be a bottleneck. For smaller amounts of data the pipeline through the workstation was slower, but for 1 megabyte of data, the workstation performs about the same as VAXes when we thought it would perform much worse. We concluded that the lightly loaded workstation is not a bottleneck because the UNIX kernel implementation of pipes is able to handle the incoming data. As the load on the workstation increases, or the pipeline becomes longer, we would expect the performance to deteriorate.

Using the rsh command to perform remote execution of UNIX commands causes the performance of low data pipelines to be slightly worse than our mechanism. This difference is a result of the overhead to perform a login on the remote machine, even though the startup file had been removed for this experiment. In contrast our mechanism executed commands in a default environment. In previous experiments we used the rsh command without removing the user's startup file and measured 50% higher times for low data pipelines.

Future work on our mechanism will include passing information to each server to use in executing a service. For example, this information might include the client's user id or information about file access privileges. The server may use this information or use some default environment. This method is preferable to the rsh approach because a complete login does not need to be performed to execute a simple service, and some servers may provide services without requiring each user to have an account on the machine.

7 Conclusion

Coordinating the activities of many tasks in a heterogeneous environment is a difficult task for the user. The problem can be divided into two parts: (1) locating and invoking a command; and (2) combining many commands into a larger computation. As a solution to the first question we have modeled the system as a set of servers, each providing a set of services. The user then accesses the distributed environment through an intelligent agent, such as a workstation, that can interact with these servers to perform services on his behalf. The user does not need to be concerned with *where* services are located, or *how* they are accessed, but only needs to know *what* services he would like to use.

As a means for combining services in an easy and understandable way we have extended the UNIX pipe facility to coordinate the data movement between services on different machines. This approach provides the user with a simple, yet powerful, mechanism for composing services in a distributed world.

To implement multi-machine pipes in an efficient manner in a heterogeneous environment we have designed a mechanism based on standard DARPA transport protocols. Our design specifically addresses the issues of efficiency and use in a heterogeneous environment. It is efficient by removing the client machine from the data flow path. Existing facilities, such as the V environment, allow remote execution but require the data to be handled by the workstation before proceeding to the next command in the pipeline. We demonstrated the weaknesses of this approach by showing performance penalties in using the V pipe server to handle large amounts of data.

Finally, our design is applicable to a heterogeneous system because it is not dependent upon a particular operating system or machine type. Even though much of the initial implementation has been done on UNIX, the design can be implemented on any machines that support standard DARPA protocols.

References

[1] E. J. Berglund, K. P. Brooks, D. R. Cheriton, D. R. Kaelbling, K. A. Lantz, T. P. Mann, R. J. Nagler, W. I. Nowicki, M. M. Theimer, and W. Zwaenepoel. *V-System Reference Manual*. Distributed Systems Group, Stanford University, Computer Systems Laboratory, 1983.

[2] Brian Bershad. Load balancing with Maitre d'. *;login*, 32–43, January/February 1986.

[3] D. R. Cheriton and W. Zwaenepoel. The distributed V kernel and its performance for diskless workstations. In *Proceedings of the 9th ACM Symposium on Operating Systems Principles*, pages 129–140, October 1983.

[4] Douglas Comer, John T. Korb, Walter Tichy, and Thomas Murtagh. *The TILDE Project*. Technical Report CSD TR 500, Department of Computer Science, Purdue University, November 1984. Also appeared in the Proceedings of the Workshop on Operating Systems in Computer Networks, ACM and IBM, Zurich, Switzerland, January 1985.

[5] Robert Hagmann. Process Server: sharing processing power in a workstation environment. In *Proceedings of the Sixth International Conference on Distributed Computing Systems*, May 1986.

[6] John T. Korb. *An Overview of the DASH Intelligent Terminal Project*. Technical Report CSD TR 492, Department of Computer Science, Purdue University, September 1984.

[7] J. Postel. Transmission Control Protocol- DARPA Internet program protocol specification. September 1981. RFC 793.

[8] J. Postel. User Datagram Protocol. August 1980. RFC 768.

[9] D. M. Ritchie and K. Thompson. The UNIX time-sharing system. *Communications of the ACM*, 17(7):365–375, July 1974.

[10] Marvin M. Theimer, Keith A. Lantz, and David R. Cheriton. Preemptable remote execution facilities for the V-system. In *Proceedings of the 10th ACM Symposium on Operating Systems Principles*, pages 2–12, December 1985.

[11] *UNIX Programmer's Manual, 4.2 Berkeley Software Distribution, Virtual VAX-11 Version*. Computer Science Division, University of California, Berkeley, August 1983.

[12] B. Walker, G. Popek, R. English, C. Kline, and G. Thiel. The LOCUS distributed operating system. In *Proceedings of the 9th ACM Symposium on Operating Systems Principles*, pages 49–70, October 1983.

[13] Willy Zwaenepoel. Implementation and performance of pipes in the V-system. *IEEE Transactions on Computers*, C-34(12):1174–1178, December 1985.

EXHIBIT F

TOWARD A HISTORY OF (PERSONAL) WORKSTATIONS (DRAFT)

C Gordon Bell
Page Farm Rd.
Lincoln, Massachusetts 01773

I originally accepted this keynote honor for five reasons: to respond to Alan Perlis' request (he told me I could present anything from a new taxonomy to personal reminiscences); second, to identify the important artifacts that should be preserved in The Computer Museum; third, to posit a framework of the history of workstations that can be written in the next century (we're all too close to create it); fourth, to summarize my own involvement on interactive computing including timesharing and large personal computers (both styles of use led to personal workstations); finally, to record my work on Digital's VAX Homogeneous Computing Environment that was totally announced last year. I led the architecture for this environment over the decade from 1973 to 1982 and was strongly influenced by the personal workstation computing movement including Ethernet. This work hasn't been described outside of product brochures.

INTRODUCTION TO THE CONFERENCE

In June 1976, an international conference was held resulting in a collection of views published in "The History of Computing in the Twentieth Century" (Metropolis, Howlett and Gian-Carlo Rota, 1980). The participants presented work up to the early 50's and thus had the benefit of 30 years of hindsight. The late mathematic's historian, Kenneth May, outlined the pitfalls of participant-written histories, commenting: "Historical description requires a time-lagged approach and means getting into understanding things as people understood them then, not as we undertstand them now." He also urged everyone to be open. Most autobiographies rewrite the past from the present perspective. Good diarists and notetakers have reference points. The best we can do is provide grist for the mill of the historian.

The History of Programming Languages Conference held in May 1977 (Wexelblat, 1981) provided a timeline to 1970 as context with detailed coverage of: Algol 60, APL, APT, BASIC, COBOL, FORTRAN, GPSS, JOSS, JOVIAL, LISP, PL/1, SIMULA and SNOBOL. This covered a period of initial developments from 1954 (FORTRAN) to 1967 (APL) with evolutions up to the present. Ten to twenty years of hindsight gave them the ability to select topics.

If this conference were limited to a 15-year rule, applied by the Annals of Computing History, no one would be describing personal workstations. Only a few experiments existed, but people were building systems with multiple graphics terminals and a workstation industry on a profession-by-profession basis started to form. Twenty years ago, most of us were trying to make our newly designed interactive, timesharing system work reliably and economically. In my own case, having also worked on small machines that ultimately became the minicomputer, I saw two independent threads for economical, computing -- including interactive computing: large shared systems and dedicated small computers.

Personal workstations, like other man-made objects, appear strictly evolutionary going through the following stages:

50's	idea (documented article, proposal, movie, etc.) stimulated through early standalone use of small computers (e.g. LGP-30, G-15)
60's	breadboards to demonstrate the idea and selected use of large computers with graphic displays (e.g. DEC PDP-1, IBM 7090, LINC, TX-2)
70's	(early) limited use of interactive shared workstations using graphic display terminals connected to minicomputers; establishment of an industry to supply terminals and professional applications software
70's	(late) working prototypes of personal workstations with concept testing through use in a complete environment (Xerox PARC)
81	introduction of personal workstations by Apollo, SUN, Xerox, and first use by early adopters
83	full-scale use for selected professionals, many companies formed, JAWS—Just Another Workstation (term) coined
85	healthy industry with evolutionary product cycle and beginning shake out of suppliers
?	steady state supply to captive users
?	decline through replacement or superposition of functions in some other form of information processing appliance (e.g. conventional Personal Computers)
?	extinction

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1986 ACM-0-89791-176-8-1/86-0001 $00.75

hence, just as we should concentrate on tracing various workstations through these stages, we may also treat this conference about the first phases of development. More hindsight will be needed to write history

Waiting longer, which gives a future view, has the risk that extinction will come through agglomeration with conventional computing. Alan Perlis urged me to come to this conference because he feared this extinction. Conventional Personal Computers, that is the evolving IBM PC will become the de facto workstation for virtually all applications by 2005. The latest chip introduction by Intel strongly permits and supports this to occur, if history is any indicator.

The real workstation phenomena and industry is not well described or recorded at all because most of the products have come from industry where individuals are not rewarded for writing papers but producing and selling machines. For example Doug Drane and the AIEX system that produced law and typesetting offices doing essentially what a STAR does, only five earlier and less expensively; GE's genigraphic system for the graphic artist; Bill Poduska and the Apollo workstations; or Applicon, Calma, Computervision, DEC, Evans and Sutherland, IBM, REDAC, Wang, Datapoint's ARCnet, etc. No university (other than Utah and Brown) really participated until 1980. Noticeably absent are Dave Evans, whose direct contributions or indirect contributions through products at Evans and Sutherland, and graduate students who went onto create much of the science behind workstations. Ivan Sutherland's work was recorded. Xerox PARC was a noteable exception, which published its work in a delayed fashion and was marginally successful at taking the research into products in a commercially viable fashion.

We now need to focus and agree on some definitions, and dimensions making sure the various threads of personal workstation development are explored so the historical facts can be recorded properly, and eventually interpreted by real historians. We need to find the first use of the name, for example, the Second Edition of the Computer Encyclopedia, 1983, uses the word three times but in a somewhat different context. By the summer of 1983, the phrase JAWS (Just Another Workstation company) could be heard in the venture capital community when they realized they had funded too many companies to build personal workstations.

WHAT IS A (PERSONAL) WORKSTATION?

Terminals connected to large computers (supercomputers, mainframes and minicomputers) and utilizing interactive professional-applications programs are the clearest antecedents. These had, as direct descendants, terminals connected to shared, but dedicated, minicomputer systems in the mid 70's to deliver cost-effective computing. An overview of the classic machines and the classes they formed is given in Figure 1.

Figure 2 shows several paths leading to the personal workstation. Large machines (with large memories) provide the major impetus. Large shared systems allow the price per terminal to fall in

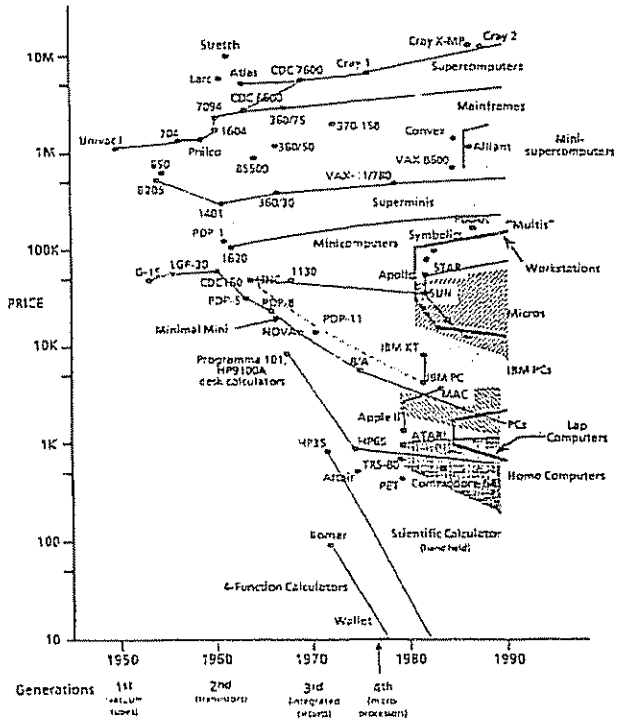


Figure 1. Computer system prices at introduction and classes versus time.

line with the salaries of the professional using the station. Four lines, corrected for inflation, depict the LINC price, the cost of a supported professional (e.g. engineer or scientist), the starting professional's salary, and a clerk's salary. The price of facilities such as a computer are "pegged" to these constants. The justification of equipment is usually based either on productivity or achieving some new level of

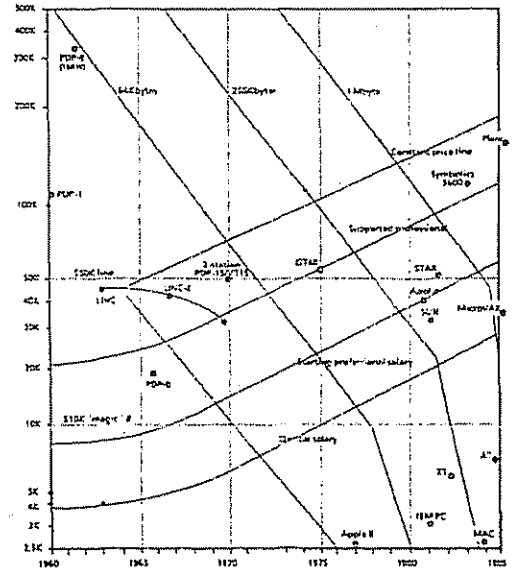


Figure 2. Workstation price versus time for various workstation introductions.

performance or capability. While salaries of clerks are greater than $10K, the workstation industry believes $10K to be magic, highly elastic, price barrier which will increase sales. In the later part of the 80's, large personal computers will easily have enough memory and capability to take over all functions provided by nearly any of today's personal computers or personal workstations. Constant price, very powerful, large workstations will continue to be built, but the mass market will be served by the rapid price decline inherent in the IBM PC evolution.

A personal workstation is a relatively large (>50 pounds) and expensive ($10,000 to over $100,000 in 1985) personal computer, with the appropriate transducers, used by a professional to carry out generic (e.g. calculation, mail and communication) and profession-related activities such as music composition, financial modeling or computer-aided design of integrated circuits.

Personal workstations are necessarily distributed with the person and interconnected to one another forming a single, shared (work and files) but distributed computing environment... the workstation environment. A workstation's location is either with an individual on a dedicated basis or in an area shared by several members of a group. This choice is dictated by the cost and size of the workstation, relative to the cost and value of the work.

Personal workstations appeared by name about 1981 with the concurrent appearance of:

- microprocessors with at least one-half megabyte of physical memory virtual memory addressing,
- memory chips of at least 64K bits providing primary memories of 1/4 to 1 megagbyte, permitting the use of large programs and construction of high resolution bit-mapped terminals
- disks of >10 megabytes
- local area networks for interconnecting the stations

Future descendants are likely to evolve from extended, lower-priced personal computers, interconnected via LANs. They would include appropriate transducers and corresponding professional applications programs.

A profession is any vocation, occupation or business, associated with work, including: secretarial, commercial, science, engineering, mathematics, and the arts.

A personal computer or pc is a self-contained computer with secondary file memory, and appropriate transducers to interface with people. A personal computer is used interactively by one person at a time, at a location convenient to the user and may "belong" either to the person or a group. A personal computer, for a given use, is self-contained, i.e. requiring no external program or data preparation units, permitting a user to go through various stages without external intervention.

The microprocessor, memory, and mass-storage technology appearing in 1975, lead directly to the personal computer industry. Early computers utilized the simple, single process, stand-alone operating systems developed for both interactive, timeshared computers and stand-alone minicomputers. Nevertheless the first personal computer, the LINC (Clark and Molnar, 1964) was built in 1962 long before its predicted technological time.

Personal computers can be used either in a

- personal or private environment encouraging separation of files, resources and work, with personal security whereby the only communication with other computers is via secondary memory (floppy) sharing, or transmission of messages via standard communications lines, or
- shared, workstation environment encouraging communication among the pc's, sharing of resources (eg. files, printers) and working on a single large work assignment or goal.

An IBM Personal Computer or PC is a particular personal computer utilizing the Intel 8086...80386 architectures and the evolving MS DOS operating systems. A PC may be extended and used in a workstation environment when interconnected via LANs and permitting access and sharing of facilities and work. Hence, constant-cost PC's are likely to evolve into the de facto personal workstations.

BELL MODEL OF MEMORY PRICE DECLINE, FORMING NEW COMPUTER CLASSES

Hardware technology improvements specifically in silicon and magnetic storage have been the sole enabling determinants of progress in computing... they are the "technological devils" that drive the formation of our industries. This is the economic basis that forms all computing classes, including all forms of workstations. Since personal workstations evolve from other computer classes, the entire hierarchy of computers must be understood. In effect, we get no product before its (technological) time!

In 1975, I observed that memory price was falling at a rate of 20% per year. Since it was a constant fraction of system prices, they also declined at the same rate. This 20% decline was based on the learning curves for manufacturing core memory manufacture and distribution. Thus, a computer would drop a factor of 10 in price, per decade.

In 1972, with the introduction of semiconductor memories by Intel, price began to decline a factor of four every three years, the cycle for new memory chip introductions. (The Four Phase company produced the first 1K MOS memory chips before 1970.) Table 1 shows the introduction of semiconductor memories and other devices, and the development of specific personal computers and personal workstation. The reduction in the price per bit of semiconductor memories amounts to a 36% price decline each year or a factor of 10 every 5 years, twice as fast as the original core memory (and discrete logic) based model.

In 1975, I also observed that memory size is the determinant of computer use and hence computer classes. That is computer power is proportional to memory size or memory size squared when a proportionally faster processor is used. Looking at the declining prices for various sized systems, one

Table 1 - Memory Chip and Microprocessor Introductions With Resulting Personal Computer and Workstations

Year	Memory	Micro	Width	Memory Memory	Examples
1972	1K	4004	4		
1973		8008	8		
1974		8080/6800	8		
1975	4K	6502	8	4K-8K	Altair, IBM 5100
1976		Z-80	8		
1977					Apple II, TRS-80, PET
1978	16K	8086	16	16K-32K	
1979		68,000	16	16K-64K	Atari
1980		8088	8		
1981	64K			64K-256K	PC, Apollo, SUN 1
1982		186/286	16		Commodore 64
1983					LISA
1984	256K	68020	32	512K-1M	MAC, AT
1985		386	32		SUN 3
1986					?
1987	1M		32	4M	?

can see each computer class taking on the attributes and power of its higher order neighbor. Table 2 classifies what I then thought were the capabilities of various computers versus their memory sizes. I felt then, and still feel, that a one-megabyte address space for a process is the minimum size, and OK at least in terms of what we actually use. Anything smaller (e.g. 64Kbytes) is shear hell to use!

My goal in 1975 was to be able to use the past to project the evolution of simple computer classes to 1980 as the basis for the VAX evolution. Figure 3 shows the price of various systems versus their memory size. Note that a new computer class emerges when the price drops an order of magnitude. This evolutionary model was also described for DEC's hardware development (Bell, Mudge, and McNamara, 1978) shown in Figure 4. This evolutionary model turned out to be quite accurate, predicting both the Apple II (just 3 years away), low-cost shared micros, and the new class of mini-supercomputers that are beginning to emerge this year.

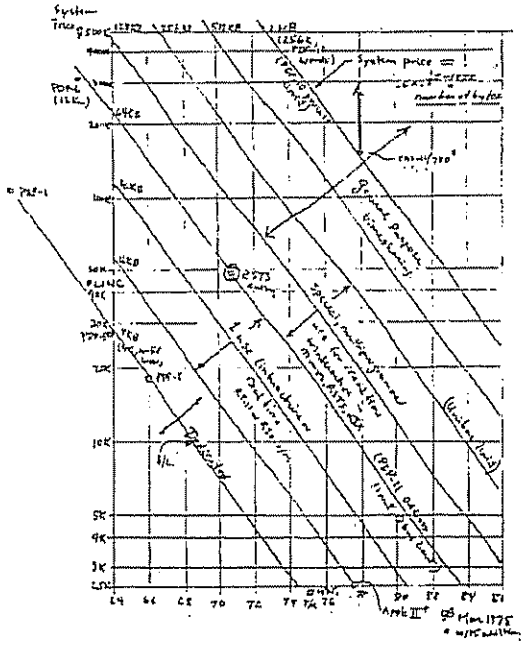


Figure 3 System price versus time for various memory sizes and system types (Bell 1975)

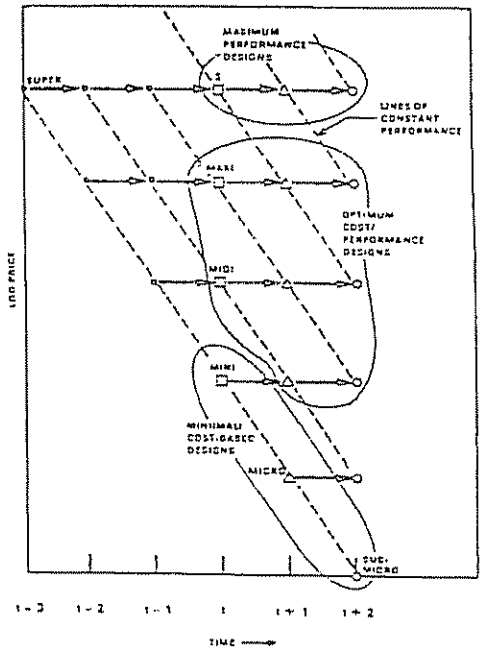


Figure 4. Price versus time for each machine class.

Since computer price determines the performance and therefore the economics and style of use, I began to observe at about the same time, that there were roughly three levels of computer use: central (mainframe), departmental (minicomputer) and personal. Dave Nelson, formerly at DEC, Prime and now at Apollo, extended this view to several other computer classes and introduced a model for price and weight (Table 3). These emerging smaller classes put significant pressure on interconnecting all the computers and forced the need for Local Area Networks (LANs). Figure 5 shows the style of use and environment for various computer classes. Note that personal workstations provide roughly the same capability of larger shared computers, and are large personal computers interconnected via LANs. IBM appears to have recently adopted this hierarchal view of computing and assigning meaningful work to all the levels, so it must be right (at least for a few years).

Table 2 - System Structure, Memory Size, and Resultant Use (C.Bell 1975)

Structure	Memory Range	Function (use)	
Dedicated (fixed-1 use)	4KB - 8KB	Interactive—e.g. POS cash register Real time—e.g. scope, traffic control, automobile	Special purpose, Fixed
Programmable (1 user)	16KB - 65KB	Interactive—RT11 (CP/M) Real time—RSX11S, M	Small scale, generality
Dedicated (multi-programmed) n-users)	65KB - 256KB	Interactive—HUMPS, RSTS, Trans-Proc. Real time—RSX-11M, D	Special purpose
Programmable (multi-programmed n-users)	128KB-1024KB	Interactive—IAS, TOPS 10, RSTS Real time—RSX-11D	Generality

Table 3 - Nelson-Bell Computer Classes*

Tier	Location	Names	Price($)	Weight(# m)
0	wallet	calculator	10	.05
1	pocket	calculator, special function unit	100	.5
2	briefcase	knee-top, small personal	1K	5
3*	office	personal, large personal, "workstation"	10K	50
4*	department	shared micro, minicomputer, super-minicomputer, mini-supercomputer	100K	500
5*	center	mainframe	1M	5,000
6*	region	supercomputer	10M	50,000**

* Founder and Vice President, Research, Apollo Computer Corporation
* Based on Bell's Distributed Processing Computing Model (circa 1975)
** Perhaps, if heat exchanger is included

NETWORKS AND THE SHARED AND PERSONAL WORKSTATIONS EVOLUTION

Although technology is a major variable that creates computer classes, the development of networks and changing patterns of use also influenced the evolution of personal workstations. The impact of the network and local area network on workstations is based on my perspective as chief architect of the ensemble of DEC products until 1983.

DECNET AND THE INFLUENCE OF ARPANET

The various DECNET architectures were initially created in 1974 by Stu Wecker while a member of DEC's research group. They were based on the notion of hierarchial computing according to use, concurrent with Digital's work on distributed processing. The imperative was to interconnect the hierarchy of DEC's minis and mainframes as tightly as possible forming a single computing environment.

The need for DECNET occured because Digital was proliferating timeshared and real-time minicomputers that had to have a way to communicate with one another and with its own and other central mainframes. DECNET was introduced in the following phases:

I	75	task-task, file transfer, file access; point-to-point; 8-bit net address; operating systems: RSX-11 D, M, IAS
II	77	improved system generation; used for first DEC engineering network (now several thousand machines at tens of sites; operating systems: rt-11, rsts
iii	80	routing, virtual terminal, full network management; operating systems: decstystem 20, vms
iv	83	large routing address, full remote rms files; c-terms; ethernet support
	85	portable DECNET (by Wecker's company)

The basic model for DECNET was ARPANET (Roberts, 1970), with the important difference that a network need not have specific IMPs or TIPs to do the packet switching for the worker computers. That is, the worker computers could do the packet switching for the network, and when the packet switching load became too high, one inserted packet-switching computers.

XEROX PARC ETHERNET/ALTO ENVIRONMENT, CARNEGIE-MELLON SPICE PROPOSAL

In 1979, Carnegie-Mellon published a proposal solliciting vendors for a personal workstation environment. Their statement, which had the effect of stimulating the design of various workstations was:

> "The era of time-sharing is ended. Time-sharing evolved as a way to provide users with the power of a large interactive computer system at a time when such systems were too expensive to dedicate to a single individual... Recent advances in hardware open up new possibilities... high resolution color graphics, 1 mip, 16 Kword microprogrammed memory, 1 megabyte primary memory, 100 megabyte secondary memory, special transducers,... We would expect that by the mid-1980's such systems could be priced around $10,000."

Although the concept was correct, the details of the machine were wrong because evolving technology was not considered. This led to selecting the wrong vendor, the late Three Rivers or PERQ Systems. Three Rivers' founder, Brian Rosen, a CMU grad who had worked at Xerox Palo Alto Research Center (PARC), designed PERQ as his version of the ALTO and D-series machines. Carnegie-Mellon may have taken on a leadership role in personal workstation environments by building its SPICE (Scientific Personal Integrated Computing Environment) on the PERQ and then contracting to build a similar environment for IBM, provided the SPICE derivative product is marketed.

The prototype for SPICE, and virtually all other distributed personal workstations environments (eg. MIT's Chaosnet and the personal LISP workstations that were the basis of LISP Machines Incorporated and Symbolics) was PARC's first 3-megabit-per-second Ethernet interconnecing Altos, operational as an environment in 1975 (Perry and Wallich, 1985; Pake, 1985). Datapoint's ARCnet with personal computers using the Intel 8080 was clearly the first commercial workstation environment. The Alto

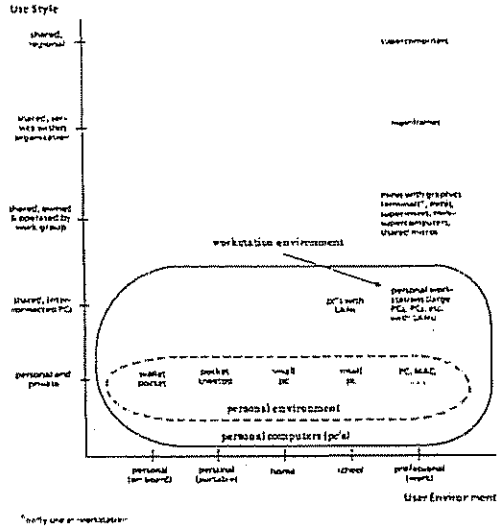


Figure 5 Computer use style versus user environment for various computer classes.

processor was a near derivative to the Data General Nova with an integrated bit-mapped display. It is important to note the PARC environment also included a timeshared computer based on the PDP-10 architecture, MAXC, which could be used for shared files and significant computation. In April 1981, XEROX introduced a product version of Alto, the STAR workstation and various file and print servers utilizing a 10 megabit per second Ethernet but without a real central computer. The PARC Ethernet was the forerunner of today's 10-megabit-per-second IEEE 802.3 standard (as originally drafted by Digital, Intel and Xerox).

THE DIGITAL VAX, HOMOGENEOUS COMPUTING ENVIRONMENT

The idea of a complete, Distributed Computing Environment, based primarily on the VAX architecture (but including the PDP-11 and DECsystem 10/20 computers) was presented to DEC's Board of Directors in December 1978 and approved as the strategy for guiding future product development.

The goals and constraints of this strategy is outlined in Appendix 1. The impetus was based on the early success of VAX, including plans to make a wide range of products to fill each of the 3 classes from MOS micros for personal computers to clusters of high-performance ECL VAXen. Furthermore, it was desirable to limit future evolution of the 10/20 and 11 lines, since both were providing essentially the same capability. The need for interconnection and the availability of LANs were critical as described above. Finally, computing was evolving from a centralized computing style to a more distributed computing environment as demonstrated by the Alto environment (but including high performance and departmental level computers).

The environment, shown in Figure 6 provided a wide range and hierarchy of compatible computers and ways of interconnecting them to provide users with generic (eg. word processing, mail) program

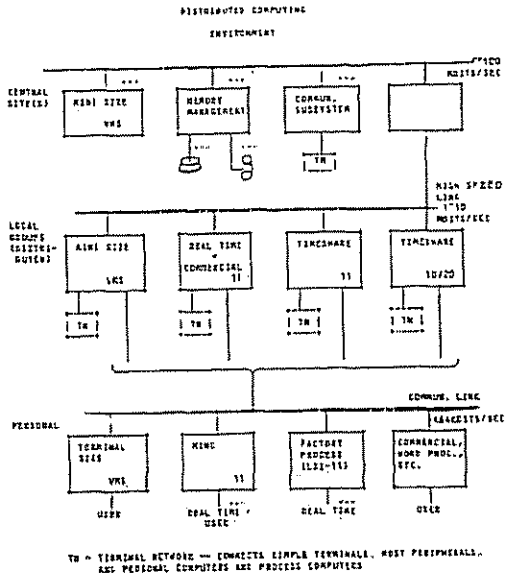


Figure 6. Digital distributed computing environment (Bell, 12-78).

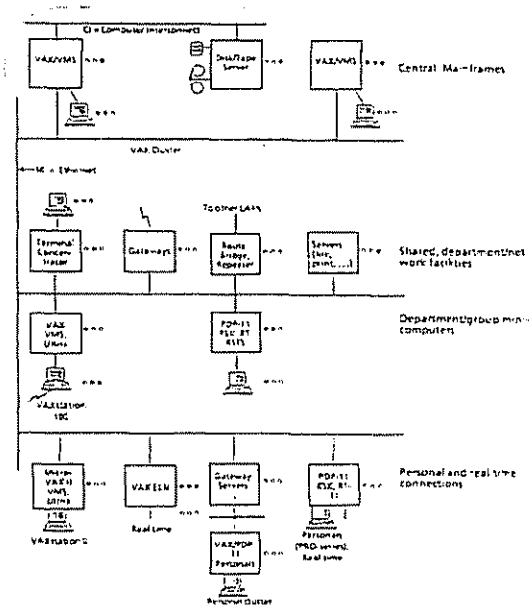


Figure 7 Digital homogeneous computing environment 1984.

development, and profession-specific computing using timeshared minicomputers or personal workstations. The goal was to provide the widest range of choices by having complete compatibility for where and how computing was to be performed without having to make a priori commitments either statically (purchase or installation) or even run-time to a particular computer system class (i.e. mainframe, minicomputer, team or personal workstation). By 1981, the original structure evolved to require the Local Area Network (Ethernet) for connecting all computers, initially called NI for Network Interconnect as part of the hierarchy of interconnects (CI=Computer Interconnect, NI=Network Interconnect, BI=Backplane Interconnect, and II=Interchip Interconnect; II did not materialize) as shown in Figure 7.

The design of the environment is substantially more than the design of a single range of compatible computers because diffent styles of user are required depending on the machine class and all the computers must be interconnected and work together in a multi-level hierarchy.

ETHERNET

While DEC had prototype LANs and proposals for LANs internally, I felt it was essential to have a standard. We hired Bob Metcalfe as a consultant and his missionary role was to make Ethernet the standard using Intel's chip design capacity and Xerox's patent and LAN experience. The IEEE got involved once they heard the idea of the LAN. They went on to facilitate the design of many other "standards", which of course, had the effect of diminishing the notion of a standard. Fortunately, the development of Ethernet crept into existence slightly more rapidly than the IBM Token Ring LAN which was introduced in October 1985, otherwise I doubt if there would be any significant use of LANs today. I understand about 30,000 Ethernets exist today.

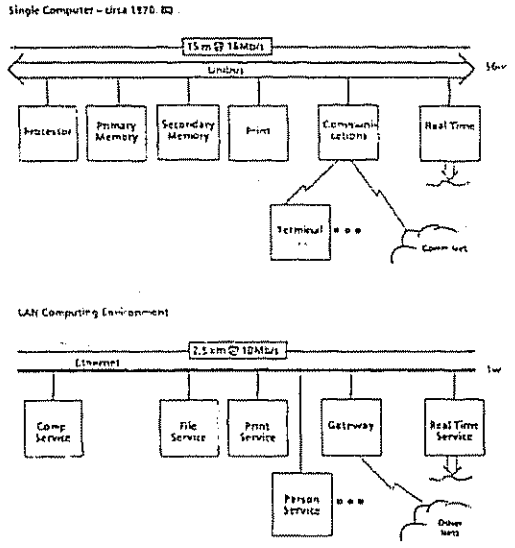


Figure 8. Unibus (c. 1970) and Ethernet (c. 1982) computer structures comparison

In January, 1981 when Ethernet was being introduced by various vendors and being completed as an international standard, several of us from the developer companies (myself from Digital, Noyce from Intel, and Liddle from Xerox) made a broad appeal to the U.S. and European press and manufacturers on the importance of the standard. My theme was: "Ethernet is the Unibus of the Fifth Generation." We all argued that the standard was essential:

- to interconnect department level minicomputers and mainframe computers to one another, permitting file to file transfers, remote-terminal emulation, and distributed processing among the shared computers,
- as a switch for interconnecting terminals to the higher level computers within the environment,
- to centralize the gateway function in one place, versus requiring every computer to have protocol conversions to communicate outside the Ethernet environment,
- for building fully distributed computing environments, The PARC model of a computing system was to decompose computing into a series of functional servers such as printing, filing and with individual Workstations. The analogy with Unibus (circa 1970) as a computer versus Ethernet (circa 1981) as a computing environment is clear when comparing the two structures (see Figure 8).
- as a bus for building computer systems.

THE ENCORE CONTINUUM

With the formation of Encore Computer Corporation in 1983, our goal was to provide a complete environment, The Continuum (Fig. 9) like the VAX environment, but based on a single microprocessor (Bell, 1985a). The "Continuum" provided both the distributed personal workstation style (described in the evolution, Appendix 1) and shared central computing using a multiple, microprocessor architecture, which I have named the Multi (Bell, 1985b). Ethernet provides the switch for interconnecting terminals to the Multimax using concentrators which pre-process terminal requests for communicating with other environments and for distributed-workstation environments. There are no terminal connections to the shared computers (something I have been trying to eliminate for years).

APOLLO DOMAIN

Dave Nelson, founder of Apollo and Domain architect (May 1980), was in the research group at DEC during the formation of DECnet and VAX. In the late 70's he went to Prime to work on the Prime token local area network and then left to develop the Apollo Domain architecture. Domain, introduced in March 1981, most typifies today's personal workstation because it provides a single addressing scheme for accessing all the memory (files) of a system. Dave credits me with stimulating this unifying architecture which is like the Unibus in terms of addressing.

WORKSTATION EVENTS, DIMENSIONS AND TIMELINE EVOLUTION

This section serves as a model for future historical research because it outlines, albeit incompletely, some critical events in workstation evolution.

The evolution of the physical workstation hardware is almost indistinguishable from the computer, and hence the evolution of personal workstations parallels and is directly coupled to the development of mainframes, minicomputers and microprocesssors. The principle difference is that workstations include a significant set of transducers to interface with humans and communication links, and additional user interface (software) advances.

PROCESSOR AND MEMORY COMPUTING ENVIRONMENT

The reinvention of the wheel and features within each of the computer classes has been nicely depicted by Burger, Cavin, Holton and Sunney (1984) as a timeline of the critical processor features (see Figure 10). The single most critical dimension of processing power is virtual memory size. The reinvention of the wheel of virtual memory size is shown in the Figure 11 from Siewiorek, Bell and Newell (1982) as each architecture reinvents and evolves the concepts of virtual memory.

VIDEO ENVIRONMENT

The evolution of displays has occurred along a set of dimensions which are relatively closely correlated with evolutionary time as follows:

- performance as measured in ability to display objects, including the ability to transform the objects
- hardware generation of the picture process. Myers and Sutherland's Wheel of Reincarnation shows the evolution of how pictures are controlled. (Figure 12 reproduced from Bell, Mudge and McNamara, 1978.) Note the evolution: direct control of the display object by the central processor; display list issued to a DMA controller or external controller; separate processor with loops, subroutines, etc.; arithmetic instructions added to aid dynamic displaying; use of virtual memory (beginning to occur to manage large pictures; and removing the display processor to form a separate computer.

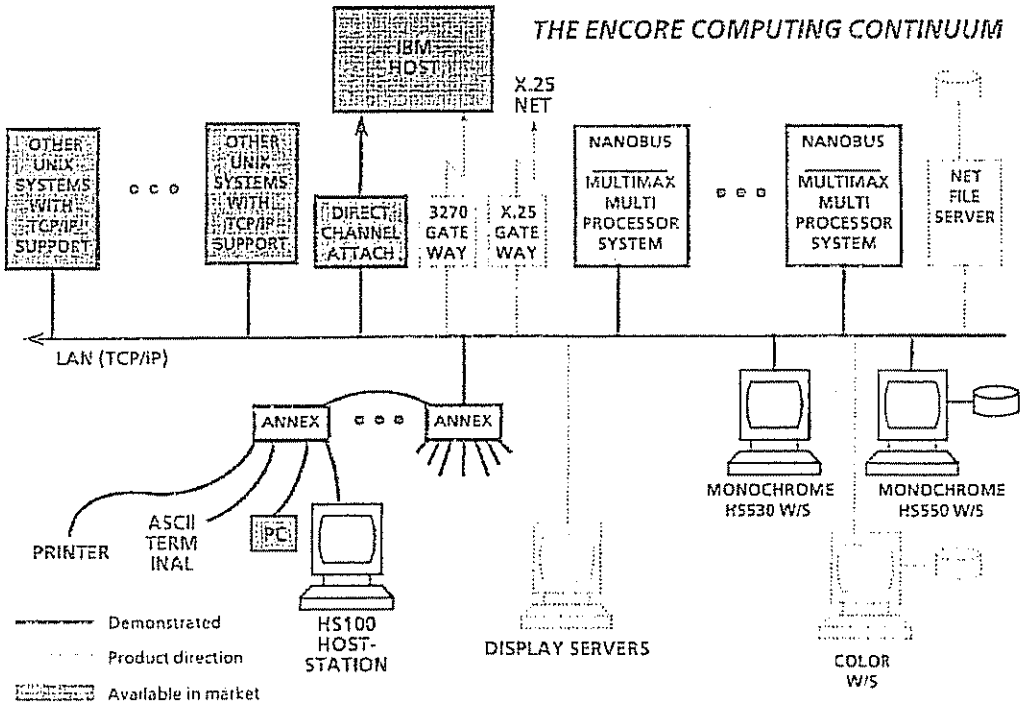


Figure 9. The Encore Computing Continuum.

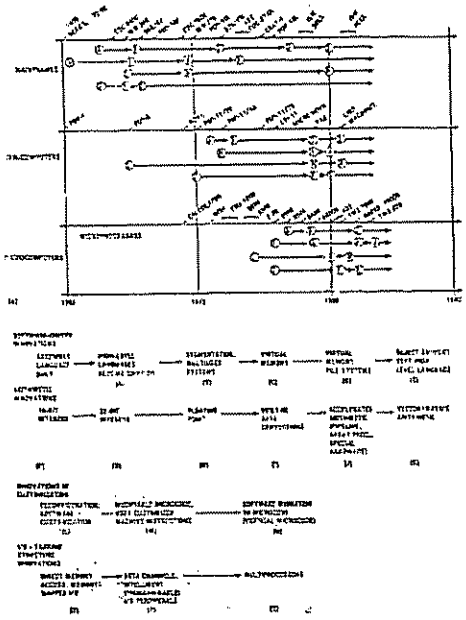


Figure 10. Computer architectural innovations for mainframes, minicomputers, and microcomputers from 1960 to 1990 (a) and the descriptive legend (b). (Taken with permission from Burger, Cavin, Holton, and Sumney, 1984.)

- resolution (in pixels or analog vector resolution)
- number of bits per pixel including use for monochrome/color
- display data-types
  - points, characters, lines, curves,
  - 3D line drawing
  - wireframe meshes, 2D surfaces, 3D surfaces,
  - shading, sculptured surfaces, constructive geometry
- translation of data-types
  - fixed
  - translation, scaling, rotation,
  - 2D clipping, 3D perspective, 3D clipping
- standard interface evolution: direct control, Siggraph CORE Library) VDI, CGI, NAPLPS, etc.
- windowing of output images
- character, line, picture and 3D image input

TACTILE AND MANUAL POSITION OR LINE INPUT
This category includes: keyboard, 2D and 3D light pens, direct touch screen, tablet, digitizing tablet, knob, joystick, track ball, mouse, etc.

PAPER OUTPUT
This section must include the evolution of: conventional printing and plotting along with early film recorders to capture images and extending to modern dot matrix printers of all kinds for monochrome and color.

VOICE I/O
While today's personal workstations do not permit substantial voice communication, voice input such as the Kurzweil voicewriter promises to add a significant dimension for human communication. The critical voice "firsts" for use with personal workstations are relatively sparse at this time.

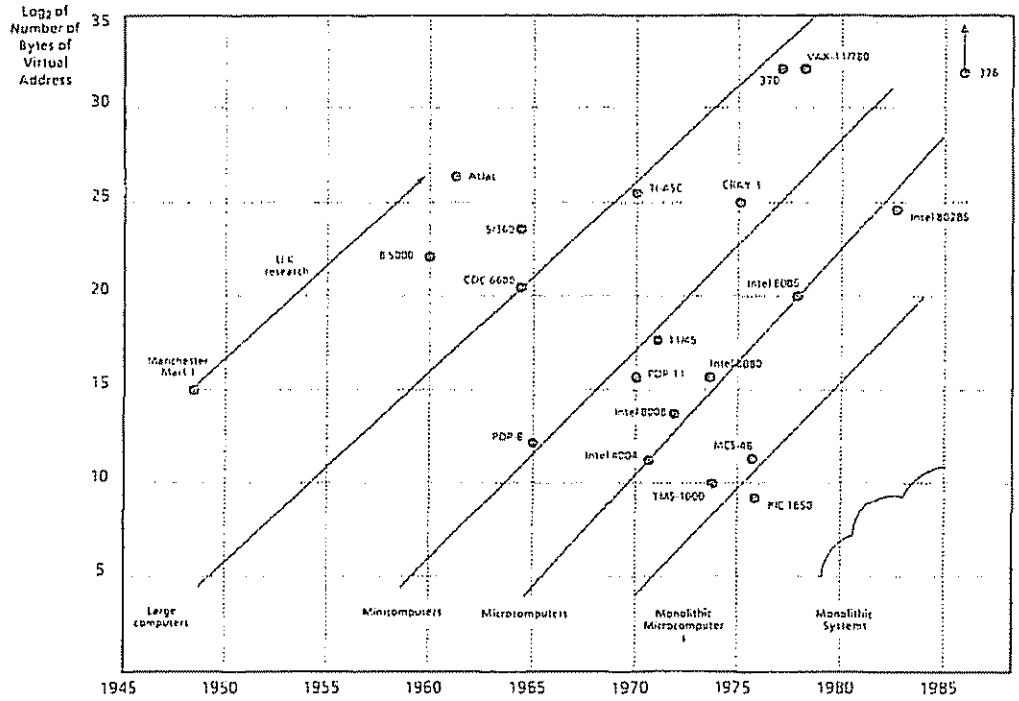


Figure 11 Virtual address space evolution for various computers and computer classes

LINKS, SWITCHES AND NETWORKS
Links, switches and communications networks have influenced the formation of computer networks and computing environments. The previous section described their influence on the formation of the DEC computing environment. Others should be traced.

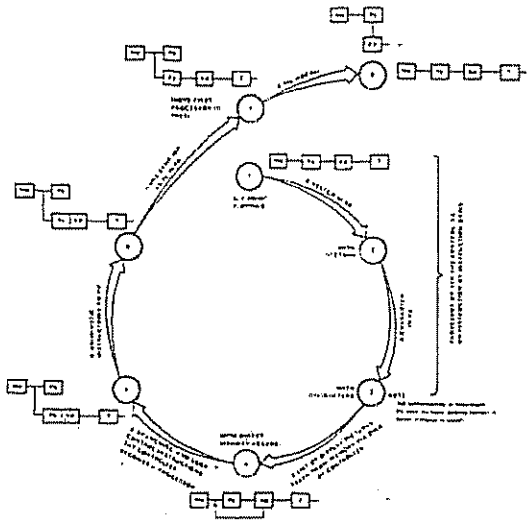


Figure 12 Myer and Sutherland wheel of reincarnation for display processor evolution

SPECIAL PURPOSE I/O AND ROBOTS
The ability to handle arbitrary signals from other environments is crucial to the formation of various profession-specific environments (e.g. music, scientific experiments).

OPERATING SYSTEM, DATA-BASE, LANGUAGE, AND GENERIC ENVIRONMENTS
Traditional computer science history has laid claim on operating systems, databases, and languages including their application to personal workstations. For example, the History of Programming Languages conference constructed an excellent timeline of the various languages including those for specialized environments such as interactive computing and specific application domains such as statistics, science and engineering.

Generic environments which users of traditional, timesharing computers have required include:

- TIMESHARING ORIENTED WITH TEXT AND GRAPHICS TERMINALS (SLOW LINKS)
  - text editing, mail and network mail
  - drawing and graph plotting
  - shared word processing
  - database access
  - transaction processing
- SHARED WORKSTATION AND PERSONAL WORKSTATION ORIENTED (FAST LINKS)
  - multiprocess control including control, windows
  - spread sheets
  - drawing and painting

PROFESSION SPECIFIC WORKSTATIONS

In addition to generic applications programs (eg. word processing and mail), each profession requires an environment to support and "understand" the work of the professional. This may take the form of special hardware (eg. high-resolution mega-color scopes for the graphic artist, high precision analog output for the musician, or fast simulation processors for the VLSI designer) in addition to specific programs that understand and have knowledge or expertise of the work being carried out. The following taxonomy provides an outline of the environments that have developed quite independently and that historians should research:

- SCIENTIFIC, MEDICAL AND MATHEMATICS WORKSTATIONS
  - life science with transducers appropriate to the measurement task
  - medical (eg. radiologist, cardiologist, bone surgeon)
  - geologist, petroleum engineer
  - physics experiment control and analysis
  - statistician and mathematician
  - signal analysis, encryption/decryption
- ENGINEERING WORKSTATIONS
  - electrical or electronic schematic (2D line drawings)
  - VLSI physical design 2-1/2 D
  - mechanical drafting
  - design of sheet metals or piping (2-1/2 D)
  - architectural and structural engineering
  - general mechanical design of arbitrary surfaces for analysis and fabrication
- SOFTWARE AND KNOWLEDGE ENGINEERING WORKSTATIONS
- ARTS (TYPESETTING, GRAPHICS ARTS, MUSIC, ETC.) WORKSTATIONS
  - publisher of: newspaper, magazine, book
  - scholars including: historians, linguists, literary scholars
  - graphic artist, illustrator, cartoonist, choreographer
  - composer
  - artistic film production
- COMMERCIAL WORKSTATIONS
  - general clerical and transaction processing, office word processing, executive
  - financial personal workstations for: banking, insurance broker
- SPECIAL CONTROL-ORIENTED WORKSTATIONS
  - air traffic control
  - process and plant control
- TRAINING WORKSTATIONS - (and are these workstations?)
  - simulation of industrial plants, aircraft, etc
  - Computer Aided Instruction such as Plato

SOME DISPLAYS, PERSONAL COMPUTERS AND WORKSTATIONS I HAVE KNOWN

This section describes various displays connected to computers, which were used as either personal computers or workstations. The MIT and Lincoln Laboratory displays embedded in Whirlwind, TX-0, TX-2, and LINC were among the earliest personal computers. Since these computers (except LINC) were one of a kind, their impact was as prototypes for subsequent developments. One direct impact was in the formation of the DEC computers, many of which were used as personal computers.

The first DEC display, Type 30 for the PDP-1 introduced in 1961, had instructions that permitted direct plotting of points and optionally, characters and vectors. As DEC evolved and built timesharing computers with operating systems which protected the user from accessing input/output equipment directly, the ability to have highly interactive terminals and workstations decreased.

Nearly all of the DEC computers were used as components for building interactive, "workstation-style" systems. DEC and Original Equipment Suppliers (OEMs) built and sold "workstation-style" systems for use in the sciences, printed circuit layout, mechanical and architectural drawing, typesetting and office automation. These pioneering applications (circa 1970) operating on dedicated timesharing systems served as the basis for much of today's evolving workstations.

MIT WHIRLWIND, LINCOLN LABORATORY TX-0, TX-2, AND IBM AN/FSQ7 (SAGE)

During Whirlwind's first year of operation, 1950, Bob Everett published a paper that showed a picture of its display (with an attached camera) and listed some actual problems carried out on the machine:

1. An industrial production problem for the Harvard Economic's School
2. Magnetic flux study for Whirlwind's magnetic storage work
3. Oil reservoir depletion studies
4. Ultra-high frequency television channel allocation for Dumont
5. Optical constants of thin metal films
6. Computation of autocorrelation coefficients
7. Tape generation for a digitally-controlled milling machine

Whirlwind was initially established as a large specialized simulator to the Airplane Stability and Control Analyzer (ASCA). The actual use turned out to be a prototype for the SAGE (Semi Automatic Ground Environment) system for air defense and SAGE, in turn, was the basis for air traffice control. The first successful use of Whirlwind for air defense occurred on April 20, 1951.

MIT's Lincoln Laboratory, established in 1951, moved into its quarters in Bedford, Massachusetts, in the summer of 1953. Lincoln built the TX-0 computer to test the use of transistor circuitry and a large core memory and then the TX-2 for large-scale computing experiments. Both had a point-plotting display with 10" x 10" area and 1K point resolution, light pen, camera, switch input, and abilities to interconnect arbitary i/o devices. Hardware innovations of the TX-2 include: addressable magnetic tape for a filing system; The Lincoln Writer, a typewriter for engineering/scientific use; and the TX-2 multiple sequence operation for rapid context switching. While the TX-2 was initially a personal computer, it operated under control of a timesharing operating system by the mid 60's. The TX computer circuitry was virtually identical to the logic and laboratory modules that DEC sold in its first four

years, prior to the introduction of the PDP-1 in 1961.

These machines used as personal computers pioneered numerous applications, the most famous being Sutherland's Sketchpad.

I spent a brief but wonderful period (1/59 - 6/60) using TX-0 for speech research and writing the program to do "Analysis by Synthesis", (Bell, et al 1961) which took as input speech spectra directly from a filter bank connected to TX-0. I also spent six months designing an IBM-compatible tape TX-0 and exploring hybrid computation (an analog computer used with TX-0). Peter Deutsch, then age 12, acted as a relatively patient user consultant and helped me debug my macros. Gwen Bell entered data on land use in the Boston area, and I wrote a program to display and explore this data. The program was demonstrated to the city-planning faculties at MIT and Harverd. Ten years later, Harvard was able to turn the clock back over 30 years when they established their computer graphics laboratory to do a simpler version of computer mapping, using punched cards and line printers.

DEC PDP-1 DISPLAYS: TYPE 30, 31, AND THE FIRST COLOR DISPLAY

The PDP-1 continued in the tradition of the "MIT personal computers". The PDP-1 had three displays. The Type 30 point plotting display, which took 50 microseconds to display a point on the 10" x 10" tube with 1K resolution. This was one of the first displays connected to a commercial computer, converting it into a personal computer for various scientific data analysis applications. Type 30 was extended to include a character generator which was controlled by loading a 36 bit, 5 x 7 bit raster specifying the character. The circuitry, designed by Ben Gurley, formed the basis of the 3XX displays. The DEC Special Systems Group designed an elaborate display, PEPR, for bubble chamber photograph analysis for MIT using the basic display circuits.

Type 31 designed by Gurley was a 5" high precision display with 4K point resolution. Only two? were built, the first being delivered to Lawrence Livermore Laboratory (LLL, now LLNL—the "N" for National) for use on their PDP-1. The Livermore PDP-1 had every option that DEC had proposed to build including Remington Rand tapes, and was used as a front end and exloratory PC for the large computers including Stretch and LARC. Ben Gurley left DEC and with Ed Fredkin established Information International Inc.(III). The first III product included a high-resolution display for various image input/ouput applications. The basic high resolution display is still in existence.

Two? PDP-1 Color Displays were built, one of which was delivered to the Air Force Cambridge Research Laboratory (the military counterpart to Lincoln Laboratory). The color display was built by modifying a standard RCA color television for point plotting.

At the Fall Joint Computer Conference in 1961, Ed Jacks, from General Motors, spent almost the entire conference watching people use the PDP-1/Type 30 draw and doodle using a program I had written. The display influenced Ed to tackle Computer Aided Design, later using the IBM 7090 with a connected display. The classic program SPACEWAR! was written on the PDP-1 in 1962, by Steve Russell. Today SPACEWAR! continues to be the center to settle litigations about the design of computer games.

LABORATORY PC'S: LINCOLN LABORATORY LINC, DEC PDP-12, AND MINC

Wes Clark and Charles Molnar continued the MIT tradition of building personal computers, and LINC was designed for wider scale use for the life sciences, by being lower cost (about $40K in 1962). The LINC was designed using mostly DEC modules. The original LINC had two 5" displays, a keyboard, 1 or 2Kw, 12-bit memory and two LINCtapes. Each unit had approximately 256 Kbytes of addressable tape memory. I believe LINC was the first production personal computer for scientific use (about 50 were produced, 21 by DEC). Its significance was its completeness at the low cost, yet being open-ended.

DEC went on to build 140 LINC-8's beginning in 1967 (designed by Wes Clark and Dick Clayton) and around 1,000 PDP-12's beginning in 1970 (Bell, Mudge, and McNamara, 1978).

DEC 338, 339, AND 340 DISPLAYS

The 338/PDP-8 display/personal computer, introduced in 1967, was used as a front-end computing terminal. The 338 and 339 (for the 18-bit computers) was, I believe, the first display processor (Bell, Mudge, and McNamara, 1978). It took a display program in the PDP-8's primary memory, which specified points, character strings, vectors, subroutines and other program-control instructions, and permitted completely autonomous operation of the display, independent of the central processor. The 339 was used with the 18-bit computers, especially PDP-9. Both systems were used for standalone data analysis and for display terminal front ends to large computers.

A special version of the 340 was designed on a Saturday morning with Ivan Sutherland for his use within the government just prior to joining ARPA. This PDP-7/340 had every conceivable keyboard and input device we could think of, including switches, trackballs, joysticks, lighted buttons, etc.

DEC PDP-15/VT15, AND GT4X "WORKSTATIONS"

The PDP-15/VT15, introduced in 1970, costing about $90K was used as a standalone for one or two users (to reduce the cost to $50K per terminal using a $15K graphics terminal). About 30% of the systems had two terminals. It took about 100 microseconds to draw a vector using the DDA technique. The PDP-15 had from 8 Kw to 32 Kw of memory. Computer Aided Design system for Printed Circuit layout written and marketed by REDAC (U.K.) was a key application, although it was used extensively for scientific-research applications.

In January 1973, the GT40 was introduced at a price of $25K with 16 Kw of memory based on the lower cost PDP-11/05. The well-known program, Lunar Lander, was written by Jack Burness for the product introduction. The relatively slow speed of the display, coupled with a limited memory and configuration of the 11/05 limited the application to a pre-processor for time-shared systems doing CAD, etc.

The GT44, was introduced in June 1973 at a price of $38K, using the same display processor as the GT40, but with an 11/34 computer and RK05, 5 Megabyte disk. The GT44 was used as a complete, standalone computer.

The GT48, introduced in October 1975, selling for $55K, was a high performance personal computer, using the 11/34, with memory up to 128 Kw. Vectors were generated using analog circuitry at a rate of 100,000 vectors per second.

THE PDP-8 WORD PROCESSING PERSONAL COMPUTERS
In addition to the early use of the PDP-8, with the 338 and various storage tube displays, the PDP-8 began to be used extensively 10 years after its birth for the DECmate series of word processing terminals. The PDP-8 was based on the PDP-5 (1963) 12-bit architecture, and influenced by the CDC 160 and LINC, addressed up to 32 Kw (later expanded to 128 Kw. The following word processing personal computers used the PDP-8:

Year	Model	Price	Comments
1975	DS310	$16K	8/A, desk, 8" floppy and VT52 terminal,
1976	WS200	$40K	8/A, cabinet, 1-8 users, 5 Mbyte disks,
1977	VT78	$15K	Intersil 6100 PDP-8 chip, embedded in VT52, 8" floppies
1981	DECmate I	$12K	6120 chip, embedded in VT100, 8" floppies
1983	DECmate II	$3.5	integrated unit, 5" floppies, bit mapped,
1984	DECmate III	$2.4	Z80/8086 and 5" winchester disk options

By 1984, the 21st birthday of the basic architecture, over 100,000 PDP-8 connected terminals or personal computers were used for word processing. More PDP-8s were built in 1984 than any other year. In a very large fraction of the use, the personal computers connected with larger departmental-level VAX computers for electronic mail, filing and auxiliary word processing for a true, shared workstation environment.

LSI-11 BASED PERSONAL COMPUTERS: TERAK, PDT 11/150, AND VT103
The LSI-11 (11/03), introduced in 1976, with a smaller printed circuit board form factor (approximately 10" square) and using the Qbus proved to be an important building block for Digital's subsequent personal computers.

Terak was founded in 1975 to exploit the notion of a bit-mapped graphics computer and used the LSI-11 (11/03) and subsequently the 11/23. The 8510, first delivered in 1976 to Ken Bowles at the University of California, San Diego for the development of UCSD Pascal, was developed indpendent of the Xerox Alto. The system operated with up to 28 Kw of memory, used 256 Kbyte floppies, and the bit-map display's resolution was 320 x 240. About 3000 were built and sold at a price of $5000 to the education market.

DEC introduced the PDT 150 as a small computer with two 8" floppies and 32 Kw at a price of less than $10K. While all the DEC marketing groups debated the price and their role in distribution, the personal computer industry formed. DEC was able to discourage Dan Bricklin from using PDT to build Visicalc. (He went on to use the newly introduced Apple II.) The VT103, introduced in 1981, was based on the VT100, which had been introduced to accept the Qbus, LSI-11/23 modules.

DEC TYPESETTING TERMINALS: VT20 AND VT71
The PDP-11, introduced in 1974 and 1977, was used as a base for the VT20 and VT71 typesetting terminals. These had a programmable character set of 256 characters, used 15" portrait oriented tubes, and communicated with the host via 9600 baud esynchronous communication. The VT20 was based on the 11/05 and could handle two CRTs, while the VT71 used the 11/03 and sold for $8 K. These terminals connected to host PDP-11's which carried out the main typesetting editing functions.

DEC VAXSTATIONS: 100, I, II AND 500 SERIES
The introduction of high resolution, bit mapped terminals, connected to VAX occurred quite late in VAX's life. A number of attempts were made, including SUVAX (for single user VAX) providing both monochrome and color, operating in 1980.

The VAXstation 100, introduced in May 1983, was a high resolution 19" terminal with bit-mapped graphics, which connected to any of the VAX computers by a high speed fiber optic link connected to the VAX's I/O bus (the Unibus).

VAXSTATION I and II, based on the MicroVAX I and II, were introduced in October 84 and June 1985, respectively. Both units sold for approximately $20K, with 19" bit-mapped CRT, using the Qbus for interconnecting I/O equipment.

The VAXstation 500 family was introduced in October 1985 with 1280 x 1024 pixel resolution color permiting 2D and 3D wire frame and 3D shaded solid images.

CONCLUSIONS
Personal workstations have evolved from many directions and along many dimensions. This conference barely scratches the surfaces of a few dimensions. We should endeavor to be as complete and accurate as possible in what we provide and in going from these papers to a more widely available publiction.

The conference has served to convince me that the critical dimensions of personal workstation evolution should be explored and documented by historians.

ACKNOWLEDGEMENTS
Dave Fanger, Encore Computer, did the figures on a Xerox Star. Designers of DEC products supplied product information: John Clarke and John Kirk (PDP-8), Herve Lavoie (VT15 and GT4x), Jim Milton (typesetting), and Stu Weckar (DECnet). Dave Nelson, Apollo, contributed a table and much to my understanding of workstations. Gwen Bell helped clarify my thoughts well enough to get a first draft, which she then edited. Mary Jane Forbes prepared the final manuscript.

APPENDIX I. THE DIGITAL VAX, HOMOGENEOUS COMPUTING ENVIRONMENT

TECHNOLOGICAL AND MARKET BACKGROUND

BATCH MAINFRAMES FOR CENTRAL SERVICES

In the first two computer generations, 1950-1970, computers were used in batch processing under the name of mainframe computing. During the 70's the mainframe began to be used almost interactively from remote job entry terminals as "glass key punches". The general direction is to have larger mainframes and larger terminal networks that interconnect to a single computer by an array of front-end computers. When more power is required, more switching computers are connected to several mainframes each of which perform a particular function. Attached dual processors are used to provide increased power for what is fundamentally a single system. Over time the evolution will be to small-scale multiprocessors for incremental performance and higher availability.

MINICOMPUTERS AND TIMESHARING FOR A GROUP

In the mid-60's both minicomputers and timesharing were developed at Digital around the PDP-8 and PDP-10 computers, respectively to lower the cost of computing. Minicomputers were initially used as components of real-time systems and for personal computing. The LINC minicomputer, developed at M.I.T.'s Lincoln Laboratory, was the first personal computer, providing a personal filing system and the ability to write and run programs completely on line.

Timesharing started out as a centralized mainframe facility for a large group with the early demonstration at MIT with CTSS, followed by the introduction of products by DEC (PDP-6), GE (using Dartmouth's BASIC) and Scientific Data Systems 940 (based on the Berkeley Timesharing System). Access was via individual Teletypes which were eventually replaced by cathode ray tube terminals, or "glass Teletypes". By the mid-70's low cost PDP-11 timeshared computers began to be used by separate groups and departments to provide "personal computing". By the mid 70's a number of minis used high performance graphics terminals for CAD/CAM, typesetting and other applications requiring graphical output or fast response. These minis were the forerunner of today's distributed, personal workstation environment.

In the early 80's, low cost disks and large memories permitted two evolved computer structures: the 32-bit supermini, and the microprocessor based "team computer". The supermini had all the power of its mainframe ancestors, especially the critical 32 bits to access memory. The "team computer", based on modern, powerful microprocessors, is simply much lower priced (eg. $15,000) providing "personal computing" at a price below personal computers.

WORKSTATIONS

The fourth generation appeared in 1972 with the microprocessor. With the second 8-bit generation microprocessor, floppy disks and 16 Kilobit semiconductor memories (circa 1976), Personal Computers were practical and began to be manufactured by Apple, Commodore, Radio Shack, etc. With 16-bit microprocesor (measured by data-path) and 64 Kbit rams, the second generation of PCs appeared in the early 80's which were suitable for building personal workstation environments.

Table 4 summarizes the characteristics of these styles of computing.

In 1979, Carnegie-Mellon University wrote a proposal for personal computer research to implement a workstation environment, and declared: "The era of timesharing was ending." Today's powerful personal workstations, such as the Apollo or SUN workstation providing a large virtual address, connected with shared facilities on a local area network characterize this type of machine and computing environment.

PERSONAL COMPUTERS CLUSTERS (I.E. THE WORKSTATION ENVIRONMENT) AS AN ALTERNATIVE TO SHARED COMPUTERS

In the mid 70's, Xerox Research Park researchers developed and provided itself with a personal computing environment consisting of powerful personal computers all linked together via the first Ethernet cable (3 Mbits), and created the notion of the Local Area Network. Their network had various specialized function servers, including a shared central computer that was compatible with the DECsystem 10, for archival memory and large scale computation.

Figure 13a shows the hardware and software of a multiprogrammed computer used for timesharing, and the corresponding structure of a Personal Computer Cluster consisting of functional services and interconnected by a common interconnect which provides basically the same capability. The timeshared system has a central memory containing various jobs connected to terminals and operating system which attends to the users and handles the particular functions (eg. real time, files, printing, communication). Personal Computers are connected to timesharing systems as terminals. By comparing the shared system with the systems formed

Table 4 - Computing Style Characteristics (Circa 1978, G.Bell)

STYLE*	CENTRAL	DEPARTMENTAL GROUP/TEAM	PERSONAL
machine	mainframe	mini	micro
price range	500K-5M	10K-500K	1K-10K 10K-50K**
communications	coupled to terminal network	to mainframes, peers and PC's	terminal emulation
database	organization's archives as service	organization, project function	personal files
terminal access	"glass keypunch"	"glass Teletype"	direct CRT
typical uses	corporate accounting, electronic mail	project CAD, order entry, small business "mainframe"	word processing, financial analysis, CAD**
caretaker	a staff providing service	distributed with user group	the user

Comments (1985):
* Not including regional-styles supercomputers
** Workstations used on a one at a time basis.

from functionally independent modules, one would expect two design approaches:

1. decomposing systems to provide shared LAN services; and
2. aggregating Personal Computer to Form PC Networks and Clusters.

Decomposing Systems to Provide Shared LAN Services

As shared computers become more complex and more centralized, it's desireable to decompose the functions for execution on smaller computers that can be distributed to be nearer the use. Thus, the decomposition of a shared system into various boxes, each of which perform a unique function

permits the evolution of the parts independent of the whole, the physical distribution of a function and the ability of several computers to share a function (see Figure 13b). While we have described the evolution of Local Area Networks (LANs) as a decomposition of a single system, LANs are generally an aggregate of heterogeneous systems which access a shared service of some kind as described below.

LANs differ from Wide Area Networks (WANs) in that they assume a low latency, high bandwidth interconnect. This permits file access as well as file transfer applications. With file access, it is possible to remotely locate part or all of a system's mass storage to a file- serving computer. File access requires bandwidth and latency which are roughly equal to that of a disk (i.e. 10 Mhz rates); file transfer can be done at substantially slower rates (56Khz to 1 Mhz).

Using the reasoning which allowed the formation of the file server, we continue the decomposition of a large central system into servers or stations and then combine these servers into a LAN. The major servers include:

1. Person Server (the personal computer used as a workstation) - local computation and human interface, possibly private storage of files
2. File Server - mass storage
3. Compute Server - batch computation or existence of particular programs
4. Print Server - printing and plotting of graphics images
5. Communication Server - terminal, telephone and PABX, Wide Area Network access including interface to international standards, other companies (eg. SNA)
6. Name/Authentication/Directory Server - naming the network's resources and controlling access to them.

A LAN formed as a complete decomposition of a single system and containing no other incompatible servers would be defined as a homogeneous cluster of Personal Computers or Workstations.

Aggregating Personal Computers to Form PC Networks and Clusters

As personal computers require more facilities (e.g. printing, communication and files), and the number and type of PCs grow, the need to directly communicate for sending messages and sharing files. Furthermore, as a collection of computers in one place forms, economy is gained by sharing common facilities such as printers, phone lines, and disks. Applenet and Corvus Omninet are relatively short and low data-rate Local Area Networks used to permit the construction of what might best be called a network of Personal Computers because of the heterogenity of type. The 3Com system for interconnecting IBM PCs is more characteristic of the homogeneous network, or cluster.

For a PC Cluster, one would expect to have a single

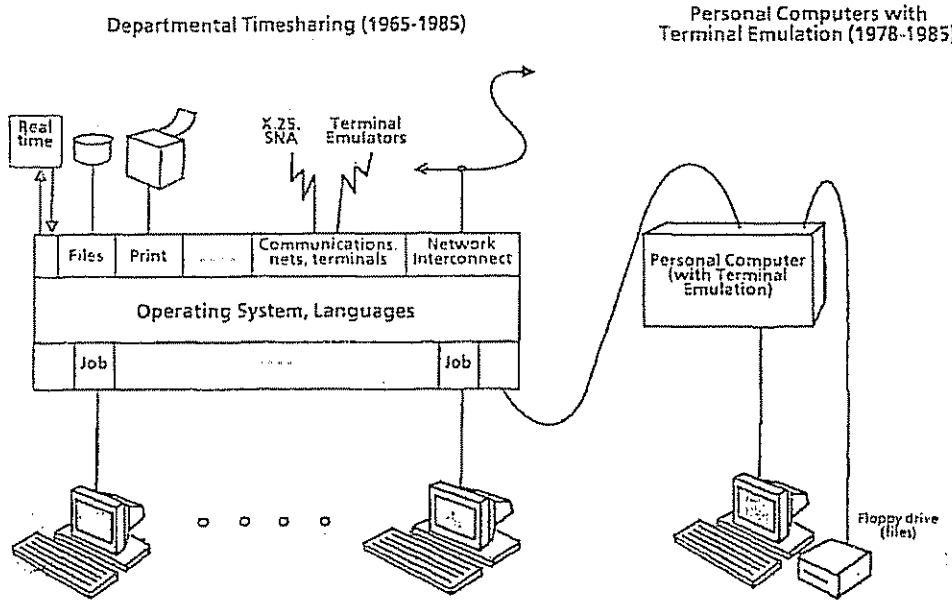


Figure 13a. Departmental timesharing (1965-1985) and personal computers with terminal emulation (1978-1985).

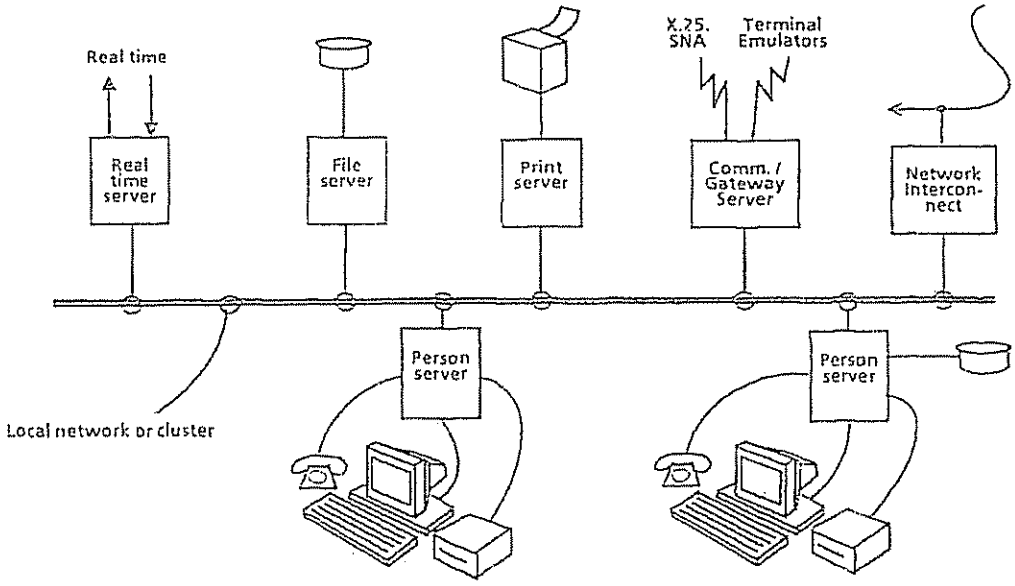


Figure 13b Personal workstation computing environment using LAN clusters (1980's)

File Server which can supply records at random to any of its constituency. Table 5 summarizes what timesharing, PC's and PC Clusters provide.

DISTRIBUTED PROCESSING USING CAMPUS AND WIDE AREA NETWORKS

The proliferation of timeshared computers required the development of networking in order for various systems to communicate with one another and to mainframes. Thus, dispersed computing became distributed computing. Store and forward wide area networks evolved from the ARPA-net, which was used to interconnect timeshared mainframe computers (mostly PDP-10's).

Table 5 - Timeshared, PC, PC Cluster and PC Network Characteristics (Circa 1981 G.Bell)

WHAT	TIMESHARED SYSTEM	PERSONAL COMPUTER	PC CLUSTER*/ NETWORKS
processing	highest peak	lowest, guaranteed	= PC
programs size	very high peak	small to medium	= PC
filing	large	small, guaranteed (+ off line)	= PC and TSS
communication	network	term. emulation	= PC and TSS
CRT	slow response "glass Teletype"	fast response, screen oriented	= PC = PC
cost	fixed, can go to lowest($/terminal	lowest entry	f(no. of PCs)
secure	shared public access	totally private	contained/TSS
prog	explicit costs shared programs, large jobs	low entry cost, "owned" by indiv., security, SW publishing = low cost	ability to expand, shared facilities, better match to org. structure
comm	shared, poor response for terminals, higher entry cost, security	limited capability, but increasing most rapidly	limited prog/prog., shared facilities
heterogenity	one homogeneous system	micro computers	one system/ can be heterogenous

*Personal workstation environment

Campus Area Networks

When a collection of Local Area Networks are connected together in a single area which extends beyond a typical LAN, we call this a Campus. Universities clearly typify the campus as does a collection of buildings. Gateways are used to interconnect LANs of different types (eg. Omninet, Ethernet (802.3), IBM Rings (802.4), Applenet, Arcnet, Broadband token bus (802.2) PCnet), whereas bridges or repeaters are used to interconnect networks of the same type to form one larger network. Bridge technology is necessary in connecting multiple LANs together on a wide-area basis using high-speed links (e.g. satellites).

Wide Area Networks

WANs are characterized by low bandwidth (up to 56 Kbits), high latency, errorful transmission, and autonomous operation of the nodes. The applications typically include: mail, file transfer, database query, and low interaction remote terminal access. Wide Area Networks can be constructed in several ways: direct dial up using conventional circuit switching with voice-grade circuits; an intermediate store and forward network such as Telenet, or a hybrid approach where various worker computers do store and forward switching.

THE VAX, HOMOGENEOUS ENVIRONMENT OR "E"

Although the specific design of the VAX Environment began in December 1978 with the approval of the Corporation, it's origins include:

- the original VAX-11 goals for a 1000:1 range of computers,
- evolution of distributed processing minicomputer networks, in Wide Areas, "Campuses", and Local Areas,
- the appearance of powerful Personal Computers and Local Area Networks, permitting the aggregation of tightly coupled "PC networs and clusters" that provide some of the benefits of timeshared minicompters and mainframes,

. the ability to aggregate minicomputers and mainframes into multiprocessors (the 78.2) and multicomputer clusters (VAX clusters) that appear to be a "single" system in order to provide higher reliability, higher performance and incremental performance.

The December '78 statement of the Distributed Computing Environment (Figure 6) and subsequent evolution [shown in brackets] was:

"Provide a set of homogeneous, distributed-computing-system products based on VAX-11 so a user can interface, store information and compute without re-programming or extra work from the following computer's system sizes and styles:
- via [a cluster of] large, central (mainframe) computers or network;
- at a local, shared departmental/group/team (mini) computer, [and evolving to a minicomputer with shared network servers];
- as a single user personal (micro) computer within a terminal [and evolving to PC Clusters];
- with interfacing to other manufacturer and industry standard information processing systems; and
- all interconnected via the local area Network Interconnect, NI [i.e. Ethernet] in a single area, and the ability of interconnecting the Local Area Networks (LANs) to form Campus Area and Wide Area Networks."

Figure 7 shows the origin of the "E" shape that characterizes the VAX Homogeneous Computing Environment. The three horizontal segments of the E provide the different computing classes which roughly correspond to different-priced computers; the functions are described in Table 4. In order to implement the environment, many requirements were initially posited, and several developments evolved from necessity:
- a range of VAX-11 and 11-compatible computers to meet the requirements of the various computing styles based on different classes of computers;
- interconnection schemes and the corresponding protocols for building multiprocessors, tightly coupled centralized VAX Clusters, LAN-based PC Clusters, LANs, Campus Area Networks and Wide Area Networks;

GOALS OF THE ENVIRONMENT

PRODUCT RANGE

The important goals and constraints of the Environment are contained in the original statement about what the Environment, which is simply "to provide a very wide range of interconnectable VAX-11 computers". The original goal of VAX was to be able to implement the range [for what appears to be a single system] of a factor of 1000 price range... with no time limit given. Since a given implementation tends to provide at a maximum, a range of 2-4 in price and 10 in performance if performance is measured as the product of processor speed times memory size, then many models and ways of interconnection were required.

At the time the 780 was introduced, the total range of products for both the VAX-11 and 11 family was almost 500, beginning with $1,000, LSI-11 boards and going to a $500,000 VAX-11/780. The VAX 8600 increased the performance range by another factor of four. If the LSI-11 is included as a Personal Computer, the price range is reduced to only a factor of 50. While the two ends of the system were "compatible" and could be interconnected via DECnet, they lacked the coherency necessary for a fully homogenous computing environment.

By introducing "VAX Clusters" (Figure 7), the range can be extended by a factor of up to the number in the cluster, or about a factor of 10 more. For VAX, Digital now provides a price range of from about $20,000 for a MicroVAX II to about $7.5 million for a cluster of twelve 8600's and a corresponding performance range of several hundred. The VAX Cluster, shown at the highest level of the hierarchy appears to the user as a single, high availability system.

The VAX Cluster work was initiated concurrent with the VAX Strategy to address the fault-tolerant, and incremental expansion needs for shared computing including a common disk and file system for the cluster. The interconnection of the clusters, using the Computer Interconnect, CI, was the beginning of a standard's activity for interconnecting computers on a standard basis.

STATIC AND DYNAMIC ASSIGNMENT OF PROGRAMS TO NODES

Ideally, a user can decide on how to compute on a completely variable basis at the following times:
- at sytem purchase or rent time ranging from outside facilities reached via gateways, to a central facility, to a shared department or team computer, to a users own personal computer
- at system-use time, ranging from access via a terminal, or personal computer interconnected to the system LAN or a particular shared computer. Here, work is bound statically to a particular set of system resources. Most likely, particular nodes would execute special programs on data located at the node.
- at task time on the basis of reliability. VAX clusters provide for dynamic allocation of work among the computers in the cluster to effect load balancing since files are centralized.
- at task-use time on a completely dynamic basis, ranging from computing on his own local system to being able to collect any resources and move work dynamically while programs are in execution. With this ability, as a program goes though its various stages of development, it might be moved from small system to large system to take advantage of increased computational power at higher level nodes.
- at task time on a dynmic basis with the ability to acquire arbitrary resources to engage in parallel computation.

CONCLUSIONS

The VAX Homogeneous Computing Environment is an important computer structure for providing a computing environment. This environment encompasses all styles, sizes and computing classes from the traditional mainframe and the high availability cluster (utilizing a fault tolerant architecture) to distributed minicomputers, and now the distributed personal workstations. It was stimulated by a number of factors including the distributed-workstation-environment idea. In terms

of architectural scope, it is much broader than simply a range of computers as typified by the evolving IBM 360 -> 370 -> 43xx -> 30xx series because it provides full compatibility through homogeniety for a complete range of computing styles and not requiring separate architectures for shared-minicomputer systems (System 36, System 38, and Series 1), personal computing (IBM PC, word processor) or terminal switching and gateways (e.g. IBM 8100).

Note: this Appendix is solely the view of the author and is not necessarily the view of Digital Equipment Corporation or its engineers.

REFERENCES

Bell, C.G., "The Multi: A New Class of Multiprocessor Computers", Science Vol 228, No. 4698, 26 April 1985, 462-467.

Bell, C.G., R.Burkhardt, S.Emmerich, A.Anzelmo, R.Moore, D.Schanin, I.Nassi, and C.Rupp, "The Encore Continuum: A Complete Distributed workstation-Multprocessor Computing Enviroment", AFIPS Conference Proceeding, 1985, Vol 54, 147-155.

Bell, C.G., G.Fujisaki, J.M.Heinz, A.S.Stevens, and A.S.House, "Reduction of Speech Spectra by Analysis-by-Synthesis Techniques", The Journal of the Acoustical Society of America, Vol 33, No. 12, 1961, pp1725-1736.

Bell, C.G., J.C.Mudge, and J.E.McNamara, Computer Engineering, Digital Press, Bedford, Massachusetts, 1978.

Bell, C.G. and A.Newell, Computer Structures: Readings and Examples, McGraw-Hill, New York, 1971

Burger, R.M., R.K.Caving, W.C.Holton, and L.W.Sumney, "The Impact of ICs on Computer Technology", Computer, Vol. 10, No. 10, October 1984, 88-96.

Clark, W.A. and C.E.Molnar, "The LINC: A Description of the Laboratory Instrument Computer", Annals of the New York Academy of Sciences, Vol 115, July 1964, 653-668.

Metropolis, N., J.Howlett, and R.Gian-Carlo, editors, A History of Computing in the Twentieth Century, Academic Press, New York, 1980.

Pake, G.E., "Research at Xerox PARC: A Founder's Assessment", IEEE Spectrum, Vol. 22, No. 10, October 1985, 54-61.

Perry, T.S., and P.Wallich, "Inside the PARC: The Information Architects", IEEE Spectrum, Vol. 22, No. 10, October 1985, 62-76.

Ralston, A. (Editor) and E.D.Reilly (Assoc. Editor), Encyclopedia of Computer Science and Engineering, Second Edition, Van Nostrand Reinhold Company, 1983, New York.

Redmond, K.C. and T.M.Smith, Project Whirlwind: A History of a Pioneer Computer, Digital Press, Bedford, Mass., 1980.

Roberts, L.G., "Computer Network Development to Achieve Resource Sharing", AFIPS Conference Proceeedings, SJCC, Vol 36, 1970, 543-549.

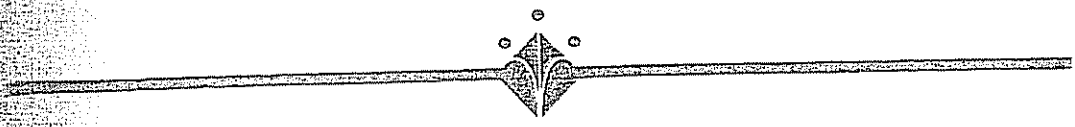
Siewiorek, D.S., C.G.Bell, and A.Newell, Computer Structures, McGraw-Hill, New York, 1982.

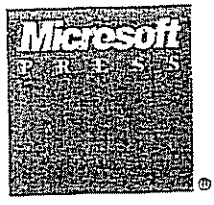
Wexelblat, R.L., editor, History of Programming Languages, Academic Press, New York, 1981.

EXHIBIT G

MICROSOFT PRESS®

COMPUTER DICTIONARY





PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Copyright © 1991 by Microsoft Press, a division of Microsoft Corporation

All rights reserved. No part of the contents of this book may be reproduced or transmitted in any form or by any means without the written permission of the publisher.

Library of Congress Cataloging-in-Publication Data
Microsoft Press computer dictionary : the comprehensive standard for business, school, library, and home
p cm.
ISBN 1-55615-231-0
1 Computers--Dictionaries. 2 Microcomputers--Dictionaries.
I Microsoft Press.
QA76.15.M54 1991
004.16'03--dc20 91-9904
CIP

Printed and bound in the United States of America

4 5 6 7 8 9 MLML 6 5 4 3 2

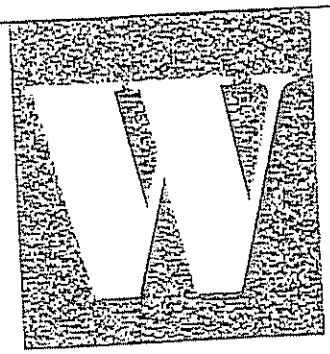
Distributed to the book trade in Canada by Macmillan of Canada, a division of Canada Publishing Corporation

Distributed to the book trade outside the United States and Canada by Penguin Books Ltd

Penguin Books Ltd., Harmondsworth, Middlesex, England
Penguin Books Australia Ltd., Ringwood, Victoria, Australia
Penguin Books N.Z. Ltd., 182–190 Wairau Road, Auckland 10, New Zealand

British Cataloging-in-Publication Data available

Acquisitions Editor: Marjorie Schlaikjer
Project Editor: Mary Ann Jones
Technical Editors: David Rygmyr, Jeff Hinsch, Mary DeJong, Dail Magee, Jr.
Manuscript Editor: Pamela Beason
Copy Editor: Alice Copp Smith



wafer A thin, flat piece of semiconductor crystal that is used in the fabrication of integrated circuits. Various etching, doping, and layering techniques are used to create the circuit components on the surface of the wafer. Usually, multiple identical integrated circuits are formed on the surface of a single wafer, which is then cut into individual sections. Each integrated circuit then has leads attached and is packaged in a plastic, metal, or ceramic holder.

wafer-scale integration A reference to the fabrication of integrated circuits (ICs) with such a large number of components that only a single IC can be fabricated from one wafer. Normally, multiple ICs are formed on a single wafer of semiconductor material, which is then cut apart. *See also* wafer.

wait state A pause of one or more clock cycles during which a microprocessor waits for data from an input/output device or from memory. Wait states are most often used to control the speed at which the microprocessor receives data from random access memory (RAM). A wait state is not noticeable to a human because it is based on the computer's internal clock, which runs at millions of cycles per second. Given the speed at which a computer operates, however, wait states are, like the speed of the clock itself, a factor that can slow system performance. In terms of system memory, "zero wait states" means that the microprocessor does not have to idle for one or more clock cycles while waiting for data from random access memory.

WAN *See* wide area network.

wand Any pen-shaped object, including a graphics tablet's stylus, but generally the pen-shaped scanning mechanism used with many bar code readers. The user passes the tip of the bar code wand, which contains optical scanning equipment, over a bar code to read it. *Compare* stylus; *see also* optical scanner, scan head.

warm boot A system restart that does not involve turning on the power and waiting for the computer to check itself and its devices. A warm boot typically means loading or reloading the computer's operating system. On IBM and compatible personal computers, a warm boot is accomplished by using the Ctrl-Alt-Del key sequence. On Apple Macintosh computers, a warm boot can be requested with the Restart command on the Special menu.

warm start *See* warm boot.

watt The unit of electrical power equal to the expenditure of 1 joule of energy in 1 second. The power of a circuit is a function of the potential across the circuit and the current flowing through the circuit. If E = potential, I = current, R = resistance, and W = watts, power in watts can be calculated as $W = (I) \times (E)$, $W = (I^2) \times (R)$, or $W = (E^2) \div (R)$. A small flashlight uses 1–2 watts, a car radio has an output of roughly 5 watts, and a toaster uses approximately 1200 watts. For low-power circuits, power is often measured in microwatts (0.000001 watt) or milliwatts (0.001 watt). In high-power circuits, units of kilowatts (1000 watts) or megawatts (1,000,000 watts) are often used.

wave Any disturbance or change that has an oscillatory, periodic nature—for example, light or sound waves. In electronics, *wave* (or *waveform*) is used to refer to the time-amplitude profile of an electrical signal.

waveform A general term used to refer to the manner in which a wave's amplitude changes with

time. *See also* period, phase, wavelength.

wavelength The distance between successive peaks or troughs in a periodic signal that is propagated through space. Wavelength is symbolized by the Greek letter lambda (λ). Wavelength is directly related to the frequency of the signal and the speed of propagation, and it can be calculated as speed divided by frequency. For electromagnetic radiation, wavelength in meters equals 300,000,000 meters per second divided by frequency in hertz. For sound traveling through air, wavelength in meters equals 335 meters per second divided by frequency in hertz.

weak typing A characteristic of a programming language, such as C, that allows the program to change the data type of a variable during program execution. *Compare* strong typing; *see also* data type, variable.

weighted code Data representation code in which each bit position has a specified inherent value, which might or might not be included in the interpretation of the data, depending on whether the bit is on or off.

well-behaved Adjective describing a program that performs properly, even when given extreme or erroneous input values. A program that obeys the rules of a particular programming environment can also be described as well-behaved. Operating-system vendors often promise that well-behaved programs will be upwardly compatible with future enhancements of the operating system.

"what-if" evaluation A kind of spreadsheet evaluation in which certain values in a spreadsheet are changed in order to reveal the effects of those changes—for example, trying different mortgage rates and terms to see the effect on monthly payments and on total interest paid over the life of the loan. Spreadsheet programs allow values in an existing model to be changed and recalculated with little effort, so these programs are considered ideal for the otherwise tedious task of preparing and comparing financial alternatives.

wheel printer *See* daisy-wheel printer.

Whetstone A benchmark test that attempts to measure the speed and efficiency with which a computer carries out floating-point operations. The result of the test is given in units called whetstones. The Whetstone benchmark has fallen out of favor because it produces inconsistent results compared to other benchmarks such as the Dhrystone and the sieve of Eratosthenes. *See also* benchmark, Dhrystone, sieve of Eratosthenes.

white noise Noise that contains components at all frequencies, at least within the frequency band of interest. It is called "white" by analogy to white light, which contains light at all the visible frequencies. In the audible spectrum, white noise is a hiss or a roar, as when a television set is tuned to a channel over which no station is broadcasting.

whole number A number without a fractional component—for example, 1 or 123; an integer.

wide area network A communications network that connects geographically separated areas.

wideband *See* broadband network.

widow A single word, a portion of a word, or a few short words left on a line by themselves at the end of a paragraph or column of type on a page. A widow is considered visually undesirable on the printed page. Because it is short, however, a widow can generally be eliminated by editing or rebreaking preceding text. *Compare* orphan.

wildcard character A keyboard character that can be used to represent one or many characters; usually encountered with operating systems as a means of specifying more than one file by name. In MS-DOS, for example, the question mark (?) wildcard character can be used to represent any single character, and the asterisk (*) can be used to represent any number of characters. Thus, ?OOK.DOC would refer to BOOK.DOC, COOK.DOC, LOOK.DOC, and so on; *.DOC would refer to any filename ending in the extension *DOC*; and *.* would refer to any filename and any extension—in other words, to all files on the specified disk or in a specified directory.

Winchester disk An early IBM name for a hard disk. The term is derived from IBM's internal code name for the first hard disk that stored 30 megabytes (MB) and had a 30-millisecond access time, reminding its inventors of a Winchester 30-caliber rifle known as a "30-30."

window In applications and graphical interfaces, a

EXHIBIT H

IEEE Std 100-1992

The New IEEE Standard Dictionary of Electrical and Electronics Terms

[Including Abstracts of All Current IEEE Standards]

Fifth Edition

Gediminas P. Kurpis, Chair

Christopher J. Booth, Editor

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

Copyright © 1993 by the
Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 1993
Printed in the United States of America

ISBN 1-55937-240-0

*No part of this publication may be reproduced in any form,
in an electronic retrieval system or otherwise,
without the prior written permission of the publisher.*

January 15, 1993

SH15594

Acronyms and Abbreviations

The Acronyms and Abbreviations Section was added to this dictionary to help the user identify some of the acronyms and abbreviations that have become part of the engineering vocabulary. It is not intended to aid in the proliferation or creation of new acronyms and abbreviations or to encourage their indiscriminate usage.

0V RTN: zero volt return (signal ground reference)

1GL: first generation language

2GL: second generation language

3GL: third generation language

4GL: fourth generation language

5GL: fifth generation language

A

A/m: ampere per meter (unit of magnetic field strength)

A/N: alphanumeric

A: ampere

A: asynchronous (type of line)

AAAS: American Association for the Advancement of Science

AAES: American Association of Engineering Societies

AB: Aeronautical Board

ABC: automatic bandwidth control

ABC: automatic brightness control

ABM: asynchronous balanced mode

AC: Advisory Committee

AC: analog computer

AC: automatic control

ac: alternating current

ACA: adjacent channel attenuation

ACB: air circuit breaker

ACI: automatic card identification

ACO: alarm cut-off

ACS: auxiliary cooling system

ACS: auxiliary core storage

ACSR: aluminum cable steel reinforced

ACT: automatic code translation

ACU: automatic calling unit

ACUI: automatic calling unit interface

ACV: alarm check valve

A-D: analog to digital

ADC: analog-to-digital converter

ADC: automatic data collection

ADL: automatic data link

ADMSC: automatic digital message switching centers

ADP: automatic data processing; automated data processing; administrative data processing

ADPC: automatic data processing center

ADS: automatic depressurization system

ADU: automatic dialing unit

AEC: American Engineering Council

AEC: Atomic Energy Commission

AES/CoS: Control Systems Panel (IEEE. AES)

AES/CrS: Crew Stations Panel (IEEE. AES)

AES/ETS: Electrical Test Systems Panel (IEEE. AES)

AES/GAP: Gyro Accelerometer Panel (IEEE. AES)

AES/GTF: Ground Test Facilities Panel (IEEE. AES)

AES/IA: Integrated Avionics Panel (IEEE. AES)

AES/ISS: Inertial Sensors and Systems Panel (IEEE AES)

AES/MEP: Mobile Electric Power Panel (IEEE. AES)

AES/RN: Radar Navigation Panel (IEEE. AES)

AES/RS: Radio Systems Panel (IEEE. AES)

AES/SS: Space Systems Panel (IEEE. AES)

AF: audio frequency

AFB: antifriction bearing

AFC: automatic frequency control

AFS: audio frequency shift

AGA: American Gas Association

AGC: automatic gain control

AGDS: American Gage Design Standard

AH: ampere hour

AI: artificial intelligence

AICE: American Institute of Consulting Engineers

AIChE: American Institute of Chemical Engineers

AIME: American Institute of Mechanical Engineers

AIS: alarm indication signal

AKWIC: author and keyword in context

ALC: automatic load control

ALD: analog line driver

ALP: automated language processing

IROD: instantaneous readout detector

IRS: isotope removal service

IS: incomplete sequence (relay)

IS: information science

IS: information separator

IS: interference suppressor

ISA: international standard atmosphere

ISAM: indexed sequential access method

ISB: independent sideband

ISD: International Subscriber Dialing

ISDN: integrated services digital network

ISFET: ion-sensitive field-effect transistor

ISI: in-service inspection

ISI: intersymbol interference

ISL: integrated Schottky logic

ISO: International Organization for Standardization

ISP: information structure perspective

ISR: interrupt service routine

ISS: information storage system

ISSN: international standard serial number

IST: international standard thread (metric)

IT: Information Theory (IEEE Society)

IT: input translator

IT: instantaneous trip (device)

IT: insulating transformer

ITV: industrial television

IU: interference unit

IUNDH: in-service unit derated hours

IV&V: independent verification and validation

IV: intermediate voltage

IZ: isolation zone

J

J/g: joules/gram

J/I: power-to-current transducer

J: joule

JCL: job control language

JI: power indicator

JR: power recorder

JSAC: Journal for Selected Areas in Communication (IEEE)

K

K: cathode (vacuum tube)

K: kelvin

k: Boltzmann's constant

k: kilo- (prefix for 10^3)

KAS: keep alive signal

KDR: keyboard data recorder

KE: kinetic energy

keV: kilo-electron-volt

kgm: kilogram-meter

kg/m^3: kilogram per cubic meter

kHz: kilohertz

KLIC: key letter in context

kMc: kilomegacycle (obsolete)—use gigahertz (GHz)

KOPS: kilo-operations per second—i.e., thousands of operations per second

KPIC: key phrase in context

kver: kilover

KWAC: keyword and context

kWh: kilowatt-hour

KWIC: keyword in context

KWOC: keyword out of context

L

L/M: lines per minute

LAN: local area network

LANNET: large artificial neuron network

LAPD: limited axial power distribution

Laser: light amplification by stimulated emission of radiation

lb: pound (1 lb = 16 ounces = 453.59 grams)

LBO: line buildout

LC-MARC (or LCMARC): Library of Congress MAchine-Readable Cataloging

LC: inductance-capacitance

LC: level control

LC: line connector

LC: line of communication

LC: liquid crystals

LCC: life cycle cost

LCD: liquid crystal display

LCS: loop control system

LD: level drive
LD: long distance
LDB: light distribution box
LDC: line drop compensator
LDDS: low density data system
LDR: low data rate
LED: light-emitting diode
LEF: light-emitting film
LEO: Lasers and Electro-Optics (IEEE Society)
LF: low frequency (30 to 300 kHz)
LFC: laminar flow control
LFO: low-frequency oscillator
LH: left-hand
LH: liquid hydrogen
LHC: level hand/automatic control station
LI: Longitudinal Interference Working Group (IEEE. COM)
LIFO: last-in. first-out
LKY: Lefschetz-Kalman-Yakubovitch (lemma)
LL: low level
llc: logical link control
LMS: least-mean-square
ln: natural logarithm (base e = 2.7183 . .)
LO: local oscillator
LO: lock-on
LO: lunar orbiter
LOF: loss of flow
LOF: loss of frame
LOF: lowest operating frequency
LOFA: loss of frame alignment
log: logarithm (base 10)
LORAN: long range navigation
LOS: line of sight
LOS: loss of signal
LOX: liquid oxygen
LP: linear programming
LP: low pass
LP: low pressure
LPD: linear power density
LPM: lines per minute
LR: level recorder
LR: load ratio
LS: level switch
LSAP: link layer service access point
LSB: least significant bit
LSB: low speed breaker (relay)
LSB: lower sideband
LSD: low-speed data
LSI: large scale integration
LSM: linear synchronous motor
LSR: load shifting resistor
LSTTL: low-level Schottky transistor-transistor logic
LT: level transmitter
LT: line telecommunications
LT: low tension
LTC: long time constant
LTS: long-term stability
LTTL: low-power transistor-transistor logic
LV: low voltage
LVDT: linear variable differential transformer
LVP: low voltage protection
lx: lux (1 lux = 1 lumen/m^2)

M

M: mega - (prefix for 10^6)
m: milli - (prefix for 10^{-3})
MAC: media access control
MAC: Membership Affairs Council (IEEE COM)
MAD: multiple access device
MAG/ET: Electronic Transformers Committee (IEEE. IM)
MAG/Meg Rec: Magnetic Recording Committee (IEEE. MAG)
MAG/Relay Meg: Relay Magnetics Committee (IEEE. IM)
MAG: Magnetics (IEEE Society)
MAGAMP: magnetic amplifier
MAN: metropolitan area network
MAR: memory address register
MARC: MAchine Readable Cataloging
MASER: microwave amplification by stimulated emission of radiation
MAT: machine-aided translation
MAU: medium attachment unit
max: maximum
Mb/s: megabits per second
MBD: magnetic-bubble device
MBM: magnetic bubble memory
MBT: metal-base transistor
MC: magnetic core

USITA: United States Independent Telephone Association

USNC: United States National Committee (IEC)

USWB: U.S. Weather Bureau

UT: universal time

UV: ultraviolet

UV: under voltage

V

V: volt, voltmeter, vacuum tube (in U.K., valve)

VA: Veterans Administration

VA: voltampere

VAM: voltammeter

V&V: verification and validation

VARACTOR: variable-reactance diode

VARISTOR: variable resistor

VBD: voice band data

VBD: voice band data

VC: voice coil

VC: voice conference

Vc: voltage common

VCCS: voltage-controlled current source

VCD: variable capacitance diode

VCM: common mode voltage

VCO: voltage-controlled oscillator

VCR: video cartridge (cassette) recorder

V/D: voice/data

Vdc: volts dc

VDD: version description document

VDM: differential mode voltage

VF: variable frequency

VF: voice frequency

VFO: variable frequency oscillator

VFR: visual flight rules

VHF: very high frequency

VLF: very low frequency (3 to 30kHz)

VLSI: very large scale integration

VM: velocity modulation

VM: voltmeter

VMOS: virtual memory operating system

VN: noise voltage

VOCODER: voice-operated coder

VP: vertical polarization

VP: voltage plus

VR: voltage regulator

VS: signal voltage

VS: variable speed

VSAM: virtual sequential access method

VSB: vestigial sideband

VT: vacuum tube

VT: Vehicular Technology (IEEE Society)

VT: vertical tabulation character

VT: virtual tributary

VT: voltage transformer

VT/Lnd: Land Transportation Committee (IEEE VT)

VU: voice unit

W

W: watt

W: write

WADEX: word and author index

WAN: wide area network

WATS: wide area telephone service

WB: wide band

WBD: wide-band data

WBDL: wide-band data link

WDM: wavelength division multiplexing

WG: waveguide

WG: wire gage

WM: wattmeter

WP: word processing; word processor

WPM: words per minute

WPOM: word processing output microfilm

WPS: words per second

WW: wire-wound

X

X ray: energetic high-frequency electromagnetic radiation

XCONN: cross connection

XO: crystal oscillator

XOR: exclusive OR

XTAL: crystal

CERTIFICATE OF SERVICE

I hereby certify that on February 28, 2007 I electronically filed the foregoing document with the Clerk of Court using CM/ECF which will send notification of such filing(s) and Hand Delivered to the following:

Richard K. Herrmann, Esquire
Mary B. Matterer, Esquire
Morris James Hitchens & Williams
222 Delaware Avenue, 10th Floor
Wilmington, DE 19899-0951

I hereby certify that on February 28, 2007, I have sent by Federal Express, the foregoing document to the following non-registered participants:

Mark A. Pals, P.C., Esquire
Kirkland & Ellis, LLP
200 East Randolph Drive
Chicago, IL 60601

Steven J. Fineman (#4025)