Structures for networks of systems

by A. L. Scherr

This paper describes how systems will be interconnected in the future, the roles that they will play, and the trade-offs that affect these roles. Starting with a general model for structuring a network of systems, general trade-offs in cost and performance are discussed relative to where functions are placed in the network. Several general principles for data and function placement in a network of systems are derived from these trade-offs. The optimal roles for each of several layers of a network of systems are discussed. Finally, conclusions are drawn regarding the design of future networks of systems.

Ever since the introduction of minicomputers and particularly since the advent of personal microcomputers, there have been ongoing debates about the ultimate usage of the various classes of computer installations: the large corporate mainframe computer, the departmental minicomputer, the desktop personal computer, etc. Some have argued that one or more of these classes of machine and/or usage will disappear. Others have asserted that each class will survive and flourish. Many of the arguments are based on price considerations that appear to be transitory in nature. As technology changes and improves, the advantage may shift from one class of machine to another. In many cases, organizational factors and management preferences have dictated the particular types of systems to be installed. In this paper, these factors—as real and as important as they are—are ignored. Rather, an attempt is made to discover the underlying technology-based pressures that will motivate the design of complex networks of systems.

History of networks of systems

It is worthwhile to review the history of configuring systems in networks to see the variations that have been used and the trends that can be discovered from seeing developments over time. In the early 1960s, when the first interactive systems were configured, the most prevalent structure was simply a large-scale system connected to end-user terminals through voice-grade telephone lines. This configuration is illustrated in Figure 1A.

The fundamental trade-offs in such systems were based on two factors: (1) the largest systems had the best price/performance ratios, and (2) the cost of computing was relatively high compared to the cost of communications. As an aside, it was not uncommon for intercontinental connections to be made to provide access to large, interactive systems. The very low bandwidth requirement of the typewriter-based terminals contributed to this low cost.

In the early 1970s, this near-universal approach to structuring interactive systems changed. An example that received attention at the time was that of a company that was acquiring computing capacity to

automate its fifteen nationwide warehouses. The application was inventory control, order processing, accounts receivable, etc. The goods in this company's warehouses were replacement parts for household appliances, and inventory was replenished twice a week by a truck from a central point.

Two distinct structures of systems were proposed: (1) a large central machine connected by telecommunications lines to terminals in each of the fifteen warehouses; and (2) fifteen minicomputers, one for each of the fifteen warehouses, with terminals attached locally and no telecommunications connection to any central computer. (See Figure 1B.) The second alternative was chosen. The rationale was that the ability to communicate did not add any value. One warehouse did not require access to another's inventory status, and resupply communications occurred only twice a week and could be accomplished with a voice message. Further, the price/performance ratios of the two classes of machines were sufficiently close to remove this as a factor.

In the 1970s, two additional structures—based upon using minicomputers—emerged as practical alternatives to using a single large system: (1) Figure 1C shows the use of multiple minicomputers connected in a peer fashion by telecommunications lines, and (2) Figure 1D shows a two-tiered system made up of a large central computer connected to remote minicomputers through telecommunications lines. The latter structure, termed a *distributed system*, was used in complex application environments for a variety of purposes. Among these purposes are those of offloading capacity from the large, central system, of providing continuing operation in the event of a failure of the central system or the communications network, and of providing the end users a degree of scheduling autonomy.[1]

One of the general characteristics that led to the use of these structures in different situations was a larger range of available storage sizes and instruction execution rates. In fact, minicomputers in the 1970s had the capacity of the largest systems of the previous decade. Another characteristic was that the price/performance-ratio curves that strongly favored the largest systems of the 1960s were straightening out, so as to reduce or eliminate this strong advantage. Communications costs were relatively higher due both to the continuing strong reductions in the cost of computing capacity and the relatively higher bandwidth requirements of the display terminals introduced in the 1970s.

So far in the 1980s, all of these configurations are being used. The advent of the microcomputer has added another range of possibilities. Thus, the four generic configurations shown in Figures 1A to 1D that use display workstations with no computing capacity can also be used with intelligent workstations based on personal computers. Further, in some cases, applications can be satisfied by an array of personal machines, either interconnected or not.

Figure 1 (A) 1960s: Interactive systems, (B) 1970s: Minicomputers locally attached to terminals; no telecommunications

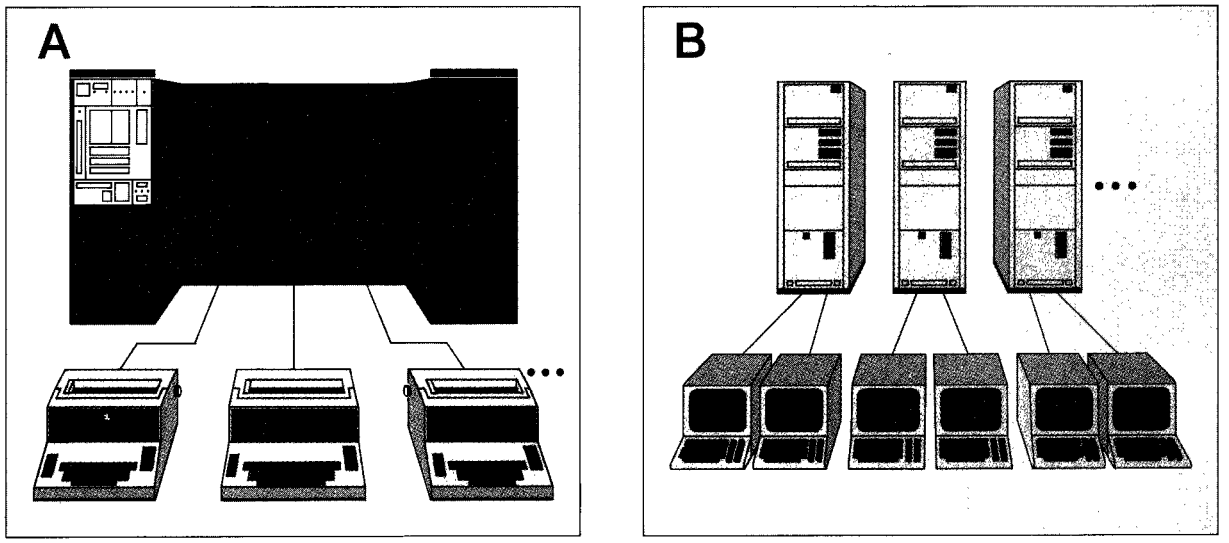
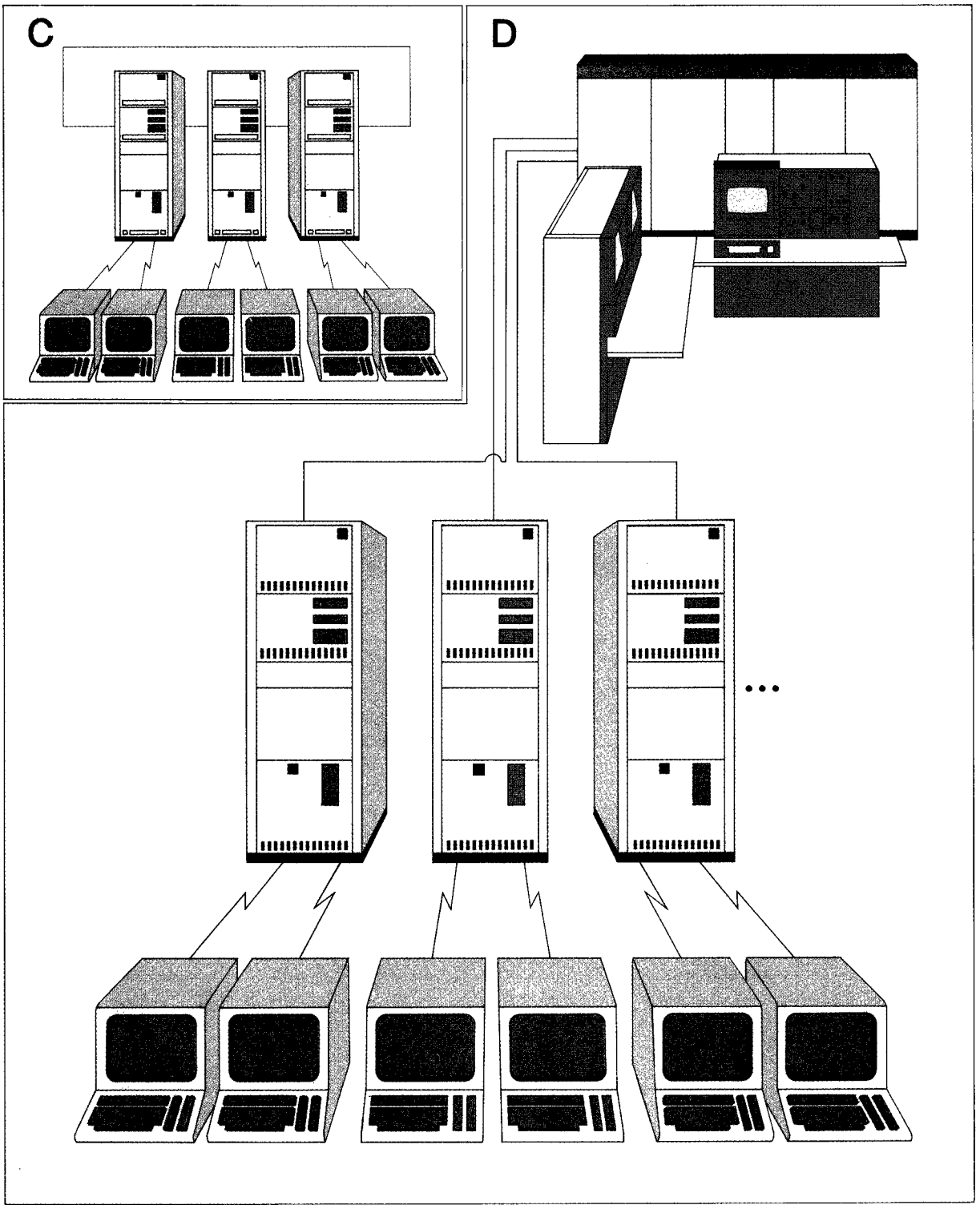


Figure 1 (C) 1970s: Multiple minicomputers connected in peer fashion via telecommunications, (D) 1970s: Two-tiered system



Current trends

Modern systems are characterized by a continuation of the trends already noted:

- An even larger range of capacity available, with the personal computer having a capacity greater than that of the largest computer available in the early 1960s.
- No discernible economy of scale.
- Communications cost even greater relative to computing costs. This is being accelerated by advances in user interfaces that require significantly higher bandwidth to support.

On the basis of these trends and for the purposes of this article, we have assumed that any given amount of hardware capacity, expressed in terms of millions of instructions executed per second (MIPS) or megabytes of memory or disk storage, can be placed anywhere with the same price/performance. Thus, the one MIPS of processing capacity in a single-user microprocessor system has the same price/performance ratio as the processor in a large-scale system. Obviously, it is extremely difficult to make accurate statements about the price/performance ratios of different classes of hardware systems. The productivity of a system depends upon the characteristics of the workload associated with it, the efficiency of the software, and the underlying architecture of the system itself. Large machines have the advantage of richer instruction sets, software designed to manage large workloads, and the inherent efficiency of heavy workloads. (When the work queue is empty, efficiency is zero.) Smaller machines profit from lower software overhead and the inherent price/performance advantage of a simpler structure.

Another assumption about hardware costs is that communication costs will not decrease as fast as the cost of computing capacity. Even the dramatic improvements made possible by such techniques as local area networks, fiber optics, and satellites have not matched the steady and spectacular progress made by computer hardware technology. One has only to look at the relative change in the cost of a long-distance telephone call versus the cost of a one-MIPS computer over the last 25 years to see this point clearly. Communications also require a substantial amount of computing resource to sustain them. Table 1 shows that the total of the instructions executed to retrieve a disk record across a communications network is roughly five times that required to access a local disk. For the remote case, two send-receive pairs are needed plus the disk access. For the local case, only the disk access is required. Therefore, as time passes, using computing capacity to avoid communications will become an increasingly favorable trade-off, both to reduce hardware costs and to avoid time delays in the performance of work.

Function placement

Both the cost of hardware for communications and computing and the time delays associated with performing work play key roles in determining the placement of function and capacity in a network of systems. We now assume that there is a spectrum of computing capacity available that ranges from the desktop workstation associated with each user of the computer complex up to a large system accessible to all users via communication lines. This section discusses some of the principles associated with choosing optimal placement of function and capacity.

The advantages of highly interactive user interfaces have become very clear.

User interface. Examined first is the support of the user interface. In recent years, the advantages of highly interactive user interfaces have become very clear. Windowing, keystroke-by-keystroke responses as seen in the popular spreadsheet programs, light pens, mice, etc. to select icons or menu items, graphics, and so forth all have increased the bandwidth required for communication between the user and the program providing the interface. Moreover, the response time requirements have become substantially more stringent as these interfaces have developed. Future improvements to the user interfaces will likely create even more demand for bandwidth and responsiveness.

The classical techniques for supporting keyboard/displays that rely on remote computing capacity to support the user interface will not be able to provide these capabilities. This assertion is based on the fact

Table 1 Example of local data and application execution

Function	Thousands of Instructions Executed
Receive message	5
Ten disk accesses	50
Send output message	5
Application and additional overhead	100
Total	160

Table 2 Example of local application execution with remote data

A. Local Node

Function	Thousands of Instructions Executed
Receive message	5
Send ten messages for disk records	50
Receive ten messages with disk records	50
Send output message	5
Application and additional overhead	100
Total	210

B. Remote Node

Function	Thousands of Instructions Executed
Receive ten messages for disk records	50
Ten disk accesses	50
Send ten messages with disk records	50
Total	150

that the need to use communication facilities, even high-speed local-area networks, and the attendant queuing delays of shared processing capacity will make many of the interfacing techniques infeasible with respect to either performance or cost. Therefore, we conclude that the optimal place to implement the user interface is in the workstation itself.

Applications. The second principle has to do with the relationship of the placement of application programs and their associated data. When the application program requests a data record that is on a remote system, the request must be placed in a message and sent across the network to the remote system.[1,2] Then the remote system receives the message, accesses the data storage device, places the record in a message, and sends it across the network back to the originating system. Finally, the originating system has to receive the message and provide it to the application program.

Consider the simple example of an application program that receives a message from the user, accesses ten data records, computes, and responds to the user with an output message. Typical figures for instructions executed for each of these functions might be as shown in Table 1. Table 2A shows the numbers of instructions executed in the original node when the data are placed in a remote node and messages are sent to that node to retrieve the data records. In addition, Table 2B shows the number of instructions executed in the remote node with the data records available at the node.

As can be seen from this simple example displayed in the three tables, using remote data is very costly in terms of additional instructions executed. If the additional delays are calculated, taking into account the transmission times across the network and the possible queuing delays from busy systems, the cost is even higher. It may be convenient to process data this way, but the extreme inefficiency precludes designing a mainline set of applications to work in this mode. Therefore, a second principle emerges: To the greatest extent possible, data should be stored in the node where the application program using the data will be executed. The cost of violating this guideline will be greater where wide-area communications is involved than for the local-area case.

Communications network designs. The third principle has to do with communications network design. Consider the case of a large corporation with its people located at many sites. In this case, there are two distinct types of communication: intrasite and intersite communication. The costs and techniques for providing these two types will differ, as will the performance seen by a user. The intrasite, local-area communications facilities will generally provide higher performance (bandwidth and responsiveness) and less cost per transmission than the intersite, wide-area communications facilities. If a given site is large, occupying several separated buildings, there may also be an intermediate type of communications with its own specific implementation and associated costs and performance.

A general design principle in networks of this type is to locate data and program execution sites so as to minimize communication delays and costs. Thus, if data and programs are to be used by only a single individual, the optimal site for storage and execution

will be the individual's own workstation. If a collection of data is accessed and updated by a group of people working in the same building, the optimal placement is in a node located in that building. If all of the users in an enterprise use (and update) a particular set of data and programs, the optimal placement is in one central site. Note that so far nothing has been said that would preclude using, for example, an individual's workstation as a central execution point for shared applications and their data.

This discussion assumes that the data being shared are being actively updated. To the degree that the data are static and unchanging, multiple copies distributed in the network become feasible. In this way, access can occur in multiple nodes and closer to the users, thus avoiding some or all of the communications overhead. Of course, the trade-off here is the complexity of managing and updating these multiple copies, and the cost of storing them, compared with the cost and delays of accessing one central copy. The direction of this trade-off changes as the frequency of data updates changes.

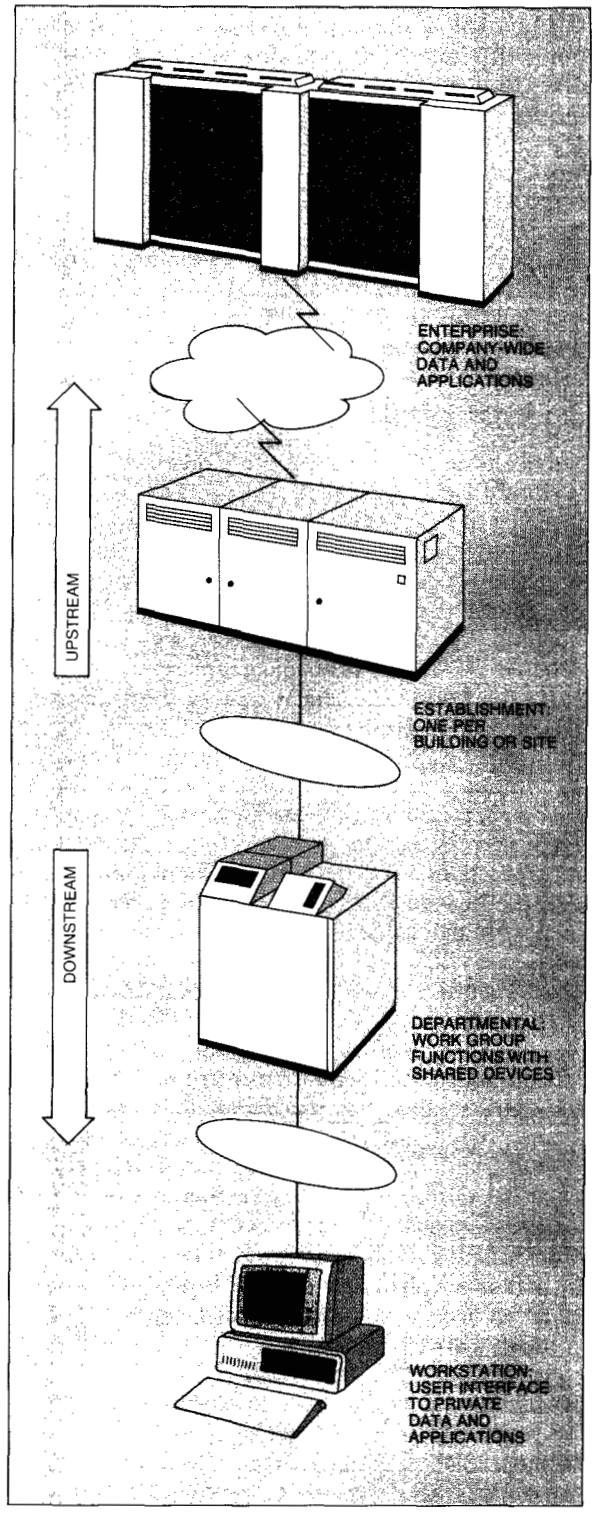
Trade-offs

There are a variety of trade-offs that modify the principles just stated. Again, assume that the choices for placing data are in an individual's workstation, a shared workgroup or departmental system located in the same building as the organization that it serves, and remote systems. Remote systems exist, according to these principles, because there are applications and associated data that are shared across a user population resident in multiple sites.

In the following discussion, the term *downstream* refers to processing closer to the user, away from the classical computing center. The term *upstream* means closer to the central computing facilities, away from the end user. See Figure 2.

From the earlier discussion, a result of placing program execution and associated data storage downstream is a reduction in both response time and communications costs. (In some cases, communications costs may even be eliminated.) The following is based on the premise that the farther upstream the node, the greater the available computing capacity. This premise has been true historically and will continue to be true as long as the preponderance of additional computing capacity is installed for applications and data shared among users. Consider the

Figure 2 Concept of upstream and downstream



case where capacity must be added for data and an application to serve a number of users simultaneously. If each user is given the capacity to execute this function at the local workstation, each workstation will require sufficient disk storage to hold the data, thus multiplying the required storage by the number of users. Furthermore, if the data are to be continuously updated, there must be an immense amount of overhead in the network to transmit and coordinate these updates. Clearly, there are other considerations regarding efficiency, and also there are special cases where placing the required capacity in each workstation is desirable. However, in general, it follows that the greater the fraction of an enterprise's users served by a new application and/or a collection of data, the farther upstream the required computing capacity must be placed.

Service time. The simplest trade-off in this context is simply the fact that because the larger capacity is installed there, the raw time required to execute any given program, that is, the *service time*, will be the minimum at the upstream node. However, because of the number of users sharing the system, the queuing delays before a program is executed will generally be larger at an upstream node. Also, because the workload queue is rarely empty, upstream systems can generally use a much higher fraction of the available cycles than less heavily loaded systems. In addition, higher degrees of function are also to be found in the larger, upstream systems. This is true for historical reasons (large-scale systems have had more years of software function development) and because high function has historically been associated with high capacity.

Data control and integrity. The next trade-off we discuss has to do with data control and integrity. Because there are fewer upstream nodes than downstream nodes and because upstream nodes generally have greater capacity, the administrative and systems management cost for managing data access, data backup, and integrity is more affordable per unit capacity. On the other hand, the nature of upstream nodes is that they are accessible to many users. Consequently, if data sharing is to be limited, it may be safer from the point of view of preventing unauthorized access (all other factors being equal) on a more downstream node.

Availability. A similar trade-off exists for availability. The cost and complexity of a high-availability system is more easily justified for a critical subset of an enterprise's data. Moreover, the fact that upstream systems are shared among many users makes them critical to the enterprise's operation. Therefore, upstream systems are more affordable, when they are used in a high-availability configuration. The trade-off in the other direction is that the farther downstream usage can be confined, the fewer operational dependencies there are. Thus, for example, if execution can be confined to the workstation and local, establishment-level systems, failures in the wide-area communications network and/or any systems upstream of it will have no effect. This principle is frequently used in the design of departmental- or establishment-level systems to achieve a high availability or at least a "fail-soft" capability, without resorting to specialized fail-safe systems.

Horizontal distribution. *Horizontal distribution*, that is, the replication of nodes in order to implement standardized function for similar workgroups or individuals, has several advantages in addition to providing higher availability. If current capacity is fully utilized and, for example, another department is added, it is less expensive to add a minicomputer than another large mainframe system. Therefore, horizontally distributed outboard systems have a more granular unit of capacity growth. Finally, the farther outboard a system is, the more control the users have on the scheduling of the associated processing resources. This is because the outboard systems are smaller, serve fewer users each, and, therefore, do not usually have the more formal and less flexible operational disciplines associated with larger systems.

Thus, in the downstream direction, the trend is toward finer granularity and increased responsiveness and usability. Upstream systems generally have greater capacity, availability, adminstrative and operational capability, and efficiency.

Configurations

Taking the general case of a large, multisite corporation and using the principles established in this paper, the possibility exists for there to be several tiers in the system: individual workstations, departmental systems, establishment systems (i.e., site-wide), corporate systems, etc. Looking at specific application usage and hardware placement, it is possible to draw conclusions about the presence or absence of specific tiers of the network of systems.

So far, the parameters have dealt with communication costs, response times, and so on. For shared

applications and data, these are the parameters that are optimized. For other shared resources, such as high-speed or high-quality printers, facsimile equipment, and plotters, an additional parameter is the physical proximity to the end users of this equipment. Using all that has been discussed, a number of generalized conclusions can be drawn about configurations, starting from the downstream workstation and working upstream.

The downstream workstation will house all private data and execute programs associated with these data, except where the capacity of a larger system is needed. This workstation will provide the primary control for the user interface. Data will be migrated to the workstation for the purposes of using programs whose user interfaces depend on the proximity to the user's display, keyboard, etc. Examples of this type of application are text editing, design graphics, and spreadsheet programs. Input and output with programs in upstream nodes will pass through a generalized interfacing program in the workstation that will handle such procedures as windowing and formatting. The workstation may have local printers, plotters, etc., but ordinarily these facilities will be provided at the next upstream-level node.

The workgroup or departmental system has two characteristics: (1) it is accessible to all of the workstations associated with users who work together on related tasks, and (2) it is in close physical proximity to end users. General shared applications at this level are office systems support, including office correspondence storage and retrieval, calendar management and the scheduling of meetings, follow-up systems for management, etc. Also present are specific applications and data shared by the workgroup. Typical examples are design data for a group project, the source libraries for a programming effort, and prospect lists for a group of salespeople. Also, this node is the connection point for shared equipment such as printers, telefacsimile devices, plotters, managed disk storage (automatically backed up and perhaps duplexed for additional reliability), and a floating-point array processor. Many of these devices need to be close to their users to be useful. The implementation of this node may vary from a large-scale mainframe system all the way to a system that is simply another microprocessor-based workstation. If no applications are run at this level, the functions may simply be those of *servers* that provide shared printers and disk storage. On the other hand, in an engineering design environment, the departmental system may turn out to be a very large system.

The next node up is the largest in the building (or site), sometimes called the *establishment system*. It may very well be the departmental system or one of the departmental systems, or it may operate on its own. Functions are implemented here that occur once per building or site: management and control of the local-area network, communications concentration/gateway/bridge from the local-area network to the wide-area network, interface to the site digital telephone private branch exchange (PBX) for voice-based office applications and program-controlled telephony. There are applications that occur once per building or site and that are executed in this node: building access control, site telephone directory management, employee time and attendance, building power, heating and cooling, and so on.

Each type of system has a significant role to play.

Finally, there are the remote, centralized, corporate-level systems, which, in a small company, may be the same as the departmental system. All of the once-per-enterprise functions are performed here. For instance, the company-wide data network is managed from such a system. In this node are the applications and data that classically have been present in such systems because of the need to share them, because of their capacity or security/integrity requirements, or for reasons of policy. At the top tier in the hierarchy of this class of systems are located such items as the corporate document file, the corporate telephone directory, and the corporate payroll and accounting journals. Another function typically found at the corporate or at least the establishment level is that of providing problem determination support, systems programming, trouble shooting, and operational and administrative support. It is advantageous to share these skills and the associated costs of these groups among as many systems as possible.

The future of networks of systems

A large number of simplifying assumptions have been made in this paper, and many details have been overlooked. Given the same facts, other authors have drawn different conclusions. In this paper, we have looked at a full spectrum of possibilities. It is difficult to escape the conclusion that each type of system has a significant role to play, and it is difficult to imagine technology changes that would eliminate any one of the types. Each type of system represents unique advantages that argue strongly for its usage. Thus, we conclude that multiple-tier systems will be in general use, particularly in large corporations, for many years to come.

There is a related question, however, which has to do with the number of lines of equipment and software that it will take to implement the three or four tiers of systems. The goals of each type of system dictate different design approaches. Size and noise are key design parameters for office-system and personal workstations. Indeed, portability may well become the fundamental parameter for personal machines of the future. For now, desktop versus office floor are the distinguishing features. As for the largest systems, technology dictates significant measures for cooling, and this—combined with the size of such computers and their associated I/O and the cabling necessary for communication with remote systems and workstations—implies the traditional central computer complex environment with raised floors, air conditioning, and so on. Therefore, at least three classes of systems will exist: the personal workstation, the computer adapted to the office environment, and the all-out technology-based high-performance system for which a special environment is required (e.g., raised floors, water cooling).

In the area of supporting software, a similar conclusion can be drawn. The personal workstation as the site of the fundamental user interface with its ultrafast response requirements will have fundamentally different software support than the large-scale, central system, with large batch jobs, high-availability features, network-management facilities, complex workloads, etc. The fact that the range of capacity will differ by two to three orders of magnitude also calls for different software bases. No single operating system existing today covers this range of capacity and operational environments, and it is difficult to imagine one that would. Thus we conclude that there is a need for systems at each level in the hierarchy and that these systems will have fundamental hardware and software differences from one another.

Acknowledgment

The author owes much to his colleagues for their input, criticism, and support. I particularly thank my friend Hal Lorin for his assistance and patience.

Cited references

1. A. L. Scherr, "Distributed data processing," *IBM Systems Journal* **17**, No. 4, 324–343 (1978).
2. A. L. Scherr, "Distributed data processing," *Proceedings of COMPCON '82*, IEEE Computer Society, Los Angeles, CA, 1982, pp. 15–23.

Allan L. Scherr *IBM Corporation, Software Development, 472 Wheelers Farms Road, Milford, Connecticut 06460.*

Dr. Scherr is an IBM Fellow who is currently on special assignment, investigating new business opportunities. Earlier in his career, he had directed several development organizations, including general-purpose advanced systems, communications programming, and distributed systems programming. Prior to that, Dr. Scherr had managed several development projects, including system design, software design, the MVS system, TSO design, and address translation architecture for System/360. He has also held key staff positions, including the Corporate Engineering, Programming, and Technology Staff and the Corporate Technical Committee. Dr. Scherr joined IBM as a staff engineer in Poughkeepsie, New York, after receiving his Ph.D. at the Massachusetts Institute of Technology in 1965. He received his B.S. and M.S. degrees in 1962, also at MIT. While at MIT, Dr. Scherr was a member of Project MAC, doing research on time-sharing system performance and user characteristics. His Ph.D. thesis, showing performance measurements and modeling of time-sharing systems, was the first such study ever made.

Reprint Order No. G321-5284.