Computer Interconnection Structures: Taxonomy, Characteristics, and Examples

GEORGE A. ANDERSON
and
E. DOUGLAS JENSEN

Honeywell, Inc., Systems & Research Center, 2600 Ridgway Parkway NE, Minneapolis, Minnesota 55413

This paper presents a taxonomy, or naming scheme, for systems of interconnected computers. It is an attempt to provide an implementation-independent method by which to identify designs, and a common context in which to discuss them. The taxonomy is based on interprocessor message handling and hardware interconnection topology, and distinguishes ten basic multiple-computer architectures. Various relevant attributes are identified and discussed, and examples of actual designs are given for each architecture.

Keywords and Phrases: distributed processing, distributed computers, multiprocessors, multicomputers bus structures, computer networks

CR Categories: 3.81 4.32 6.20

INTRODUCTION

Currently, one of the most active areas in computer architecture is the interconnection of computers to form systems which are called "distributed processors," "distributed-function computers," "computer networks," and similar names. These systems range in organization from two processors sharing a memory to large numbers of relatively independent computers connected over geographically long distances. A discouraging aspect of this activity, however, is the almost total lack of published information describing the rationale for various designs, or comparing the results achieved by various approaches. In part, the authors believe this condition exists because there has been no common context in which such discussion could take place, no set of design issues, no list of system characteristics to be traded off, and, in fact, not even a common nomenclature for system identification. Our paper is an attempt to begin filling this need. In it we present a naming scheme, or taxonomy, for identifying various systems of interconnected computers, and we discuss design decisions and system characteristics which we believe are germane to these architectures.

The authors know of only one other general taxonomy for interconnected computers and that is a brief one (having different dimensions) with few system characteristics and no nomenclature [SIEW74]. Some interconnection topology issues are also considered in [CHEN74] and [THUR72], although these are primarily concerned with the next lower level of the interconnection design—control and communication. One level beneath these are a number of papers dealing with the design of "explicit" switches, such as crossbars [PIPP75] and permutation/sorting networks [THUR 74]. In addition, there is a wide variety of digital

CONTENTS

data communications literature pertaining to queueing, routing, multiplexing, etc. [MART72]; much of this is relevant for certain computer interconnection architectures and implementations.

For the first step toward developing a common nomenclature for system identification we have restricted ourselves in several significant ways. First, we are concerned solely with interconnected hardware units in which "processes" can execute. We use the word process in the conventional sense, and designate the hardware units as Processing Elements, or PEs. By this definition we specifically exclude single-instruction stream, multiple-data stream machines such as ILLIAC IV and PEPE. We further limit ourselves to systems in which any PE can communicate with any other through the system interconnection mechanism.

Our method for identifying the interconnection structure of a system is to isolate the major hardware units involved in the transfer of information between processes in different PEs. We call this transfer a "message transmission," and do not distinguish between instances of *message* such as data blocks, service requests, semaphores, etc. In the interconnection structure itself we distinguish two functional entities—paths and switching elements.

A *path* is the medium by which a message is transferred between the other system elements. Some examples of paths are wires or busses, radio links, common-carrier data-transmission facilities, and memories. The transmission of a message over a path results in no alteration of the message.

A *switching element* is an entity which may be thought of as an "intervening intelligence" between the sender and receiver of a message. A switching element affects the destination of a message in some way—by altering the message (e.g., changing its destination address), by routing it to one of a number of alternative paths, or by both actions. These notions of message, path, and switch are basic to the approach we have taken.

Our taxonomy thus describes configurations of three hardware archetypes: PEs, paths, and switching elements. This small number of types leads to several simplifications which serve on one hand to make system organizational issues clear, but on the other, admittedly obscure the noninterconnection aspects of system design. For example, we do not distinguish between a computer and its interface to the rest of the system—both are part of the PE. Neither do we make a distinction between circuit switching and message-switching—both are accomplished by the switching entity. Perhaps the most significant issue that is not treated is interprocess communication strategies and problems, such as message addressing, deadlock, etc. By these omissions we do not imply any relative importance, but rather, we stress that we have taken a limited step in but one of several important areas. It is our hope that this step will stimulate similar work in such complementary areas as interprocess communication, as well as encourage improvements in our taxonomy.

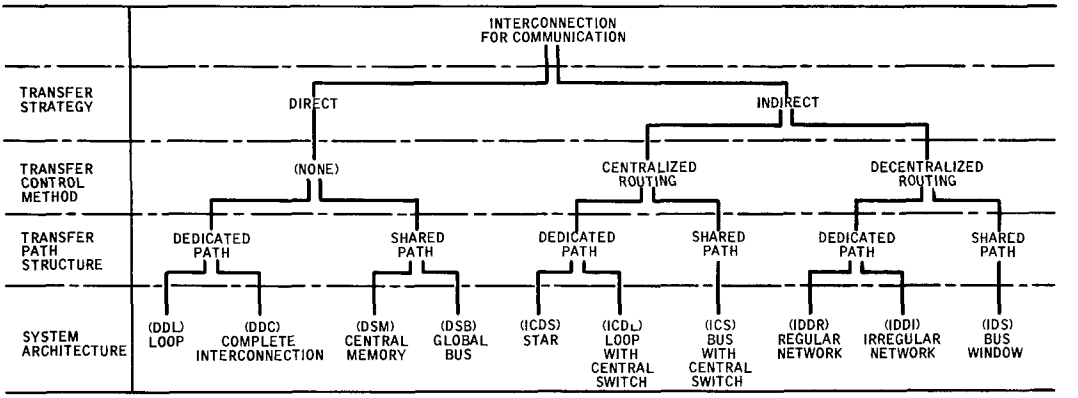

FIGURE 1. The taxonomy.

DESIGN DECISIONS—THE TAXONOMY

An interconnected computer system is the result of a series of design decisions, and the decision space can be considered to be a tree. Our model for this interconnection design process, shown in Figure 1, is a tree of four levels with alternative system architectures represented as leaves. The root of the tree is the decision to interconnect a number of computers for complete intercommunication. Below this are decision levels representing choice of message transfer strategy, the method of controling transfers, and choice of the type of path over which the transfer is to be made. The first two levels are concerned with strategic (policy) issues, and the third and fourth levels with tactical (implementation) issues.

The first strategic choice is between *direct* transmission of messages from source to destination, and *indirect* transmission in which an intervening operation is required. For purposes of the taxonomy, our criterion for distinguishing between these is the existence of one or more switching entities which make decisions for every message. Thus, intervening repeater circuits or storage elements are simply instances of paths, and do not affect the directness of the communication; but an intervenor that alters the message (e.g., address transformation), or an intervenor that routes the message onto one of a number of alternative output paths, is effecting an indirect communication. Another way to make this distinction is to determine whether control information is contained in or sent to the intervenor (e.g., address transformation tables). Decisions made by the sender (for example, which port to transmit on) and decisions made by the receiver (such as whether to accept a given message) do not affect the directness of communication.

If indirect communication is chosen, a further decision concerning the switching method must be made. This is shown at the second level of the tree. The alternatives are *centralization*, in which a single entity switches all messages, and *decentralization*, in which a number of intervenors are used.

The third level involves the choice of *dedicated* or *shared* message transfer paths. We define a shared path as one which is accessible from more than two points. In reality, there are at least three alternatives that may be distinguished: paths that are unidirectional point-to-point; paths that are bidirectional point-to-point; and paths that are bidirectional and visit more than two points. In the first case no contention can occur; but in the second, a rudimentary sharing exists, and hence contention can occur. In the third case, however, contention becomes a major consideration, so we define it as the "shared" path case and classify the other two as "dedicated" connections. We reiterate that the notion of "path" does not imply an implementation, and that both busses and memories can be appropriately used as message transfer paths. It should also be noted that paths which are redundant

for fault tolerance or bandwidth reasons are here considered logically singular.

The final level of the taxonomical tree comprises the leaf nodes representing specific system designs.

Before discussing the characteristics of the system types, we digress here to explain the various attributes which we feel are most significant, and to define our nomenclature. Our emphasis will be on the implementation-independent issues, and on the qualitative characteristics of systems. We avoid quantitative measures such as bandwidths and throughputs because they can only be representative of rapidly changing technologies and therefore must be evaluated within the constraints of a specific application.

For brevity, we will use sequences of capital letters to describe paths down the tree, with lower-case "x"s denoting "unmade" decisions. A direct, dedicated-path system is thus DDx, and so on.

SYSTEM CHARACTERISTICS

Modularity, the ability to make incremental changes in system capability, is a major characteristic to be considered in the design of a computer system. In instances where a specific design is to be configured for a variety of applications, it is often desirable to vary the number of processors according to the computational requirements of the particular problem. This requirement occurs both in homogeneous systems (having only a single processor type) and in nonhomogeneous systems. One measure of system modularity is the incremental cost of adding an element, such as a processor. If this cost is simply that of the element, then the system is indeed modular; but if the addition of the nth processor requires the addition of $n-1$ interconnection paths, then the system is not so modular. At the third level of the tree, some decisions involving this *cost-modularity* measure have already been made. For instance, selection between Direct (Dxx) and Indirect (Ixx) paths involves tradeoffs between the poorer cost-modularity of dedicated paths and the vulnerability of shared paths to bottlenecking.

Another measure of modularity is the degree to which the location and function of the incremental element is restricted. For instance, in a given design there may be particular places where a resource (processor, switch, or path) could be easily added to produce a specific performance increase, and other types of performance increase which are difficult or impossible to obtain in a modular fashion. Again in this case, decisions made in the progression to the third level of the tree have affected modularity. For instance, this *place-modularity* characteristic of Indirect Centralized (ICx) systems is poor with respect to the central switch. Replication of the central switch to achieve an increase in throughput changes the basic architecture to Indirect Decentralized (IDx). A place restriction can also occur in any nonhomogeneous Indirect (Ixx) architecture, since a special-purpose processor which must be added to the system usually cannot occupy a place that must perform a switching function.

Connection flexibility, a characteristic akin to modularity, must be considered for Ixx architectures. In Dxx architectures, the decision to add a processor requires no deliberation on the method of connection; it is fixed by the system type. For architectures allowing indirect communication, there can be alternatives with different costs. For example, in a geographically dispersed system, the cost of adding another processor at the location of an already existing one is significantly affected by whether the incremental processor must have its own paths to the rest of the system, or whether it can share the paths already installed.

Another important design characteristic is the cost of fault tolerance and the method by which a system is reconfigured to mask faults in processors and intercommunication paths. The first measure of goodness here is the effect of a fault. In designs where specific elements are shared (DSx, ICx, IDSx), a single failure of the shared element can completely halt system operation. In other designs the structure is such that failures have less catastrophic results. In addition to this *failure-effect* aspect, it is also necessary to determine the costs of alternative methods

of masking faults to allow operation in a degraded mode. (We maintain that a system operating in the presence of a fault is functioning in a degraded mode, regardless of whether or not the effect is observable using a performance measure.) Design decisions allow this *failure-reconfiguration* measure to range from excellent in systems requiring no overt reconfiguration and having minimal spare hardware, to very poor for those requiring that the entire intercommunication system be made redundant. A reconfiguration may even change the system's basic architecture. For instance, an ICx architecture that experiences a failure of the centralized switching resource may reconfigure to allow decentralized communication (becoming IDx) and thus avoid the cost of replicating all or part of the switch. This is an obvious area in which a hardware/software tradeoff exists, since a dynamic reconfiguration from one taxonomic architecture to another has significant software ramifications.

Inherent performance limitations, and the cost incurred in overcoming them, must also be considered in the choice of an architecture. The problem here is one of *bottlenecks* in resources, due either to a nonuniform flow of communication within a system, or to saturation of a shared resource. At the third level of the tree, DSx, IDSx and ICx architectures can be seen to be limited (i.e., to have poor cost-modularity for increased communications rates), since increasing their performance in areas served by the single shared path or switch requires significant hardware changes.

The nature and number of decisions that must be made to effect communications within a system are an important consideration. We call this attribute *logical complexity* and use the term to refer to the totality of decisions made during communications, whether made by source and destination processes, or by switching entities. Logical complexity is a characteristic that is significantly affected by the architecture, but its major effect is on software cost. From the standpoint of the architect, this makes logical complexity an almost unquantifiable element in tradeoffs, and the best that can be done is to make relative rankings for systems under consideration. In Ixx systems, the method by which the switching information is communicated is a major logical complexity issue. A "chicken and egg" puzzle pertains; the switching information that must be communicated (from somewhere in the system) to the switching resource comprises a message, but messages cannot be sent unless information for switching them exists in the switching resource. The magnitude of problems arising in the communication of switching information depends both on the system's type and on its operating environment. ICx systems are better in this regard than IDx systems; the more dynamic the processing environment, the more complicated the problem.

In addition to the characteristics just listed, which are largely determined by strategic decisions, there are a number of characteristics that are the result of implementation decisions. Among these are the physical dispersibility of the system, compatibility with commercial communication paths, message transfer delay between sender and receiver, and the cost of the interconnection paths. These, together with the strategy-dependent characteristics, are detailed in the following sections describing the architectural alternatives.

SYSTEM DESIGN TYPES

In the following paragraphs we discuss the significant features of each of the system species in our taxonomy. Our first attempts at these descriptions were made from a completely implementation-independent viewpoint, a perspective which we found untenable unless important design issues were to be omitted. Because of this, the discussions represent a compromise and certain observations are made both from a strict taxonomical viewpoint as well as from research and experience with actual designs. We also identify particular implementations of each interconnection type.

DDL—Loop

Loop architectures (Figure 2) have evolved from the data communications environment, and consist of a number of individual

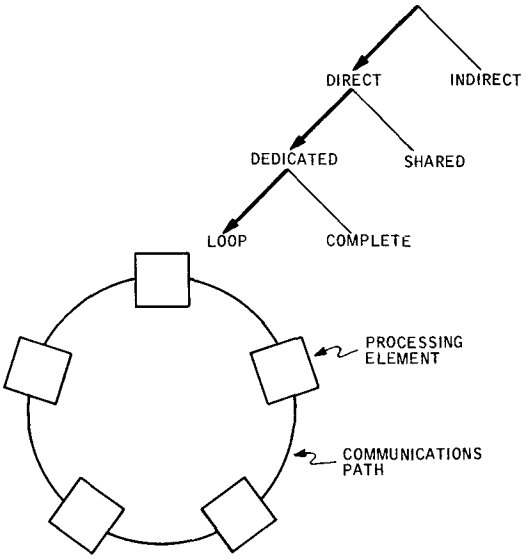

FIGURE 2. DDL (Loop).

processing elements (PEs), each of which is connected to two neighboring processing elements. The traffic in a loop could, in principle, flow both directions. In practice, the complexity of bidirectional traffic has constrained all the loops (of which the authors are aware) to only unidirectional traffic. In a unidirectional loop, one neighbor of a PE can be regarded as the source neighbor and the other as the destination neighbor. A given PE receives messages only from its source neighbor and sends messages only to its destination neighbor. Messages circulate around the loop from source to destination with intermediate PEs acting as relay or buffer units. DDL systems may allow one [FARM69] or more [REAM75] messages to circulate simultaneously, and messages are of either fixed or variable length [WEST72]. Some systems which have been referred to in the literature as loops contain a centralized switching function, and thus appear as ICDL in our taxonomy. Other systems are coupled loops with decentralized control, which we classify as IDDI.

Both the cost-modularity and the place-modularity of DDL systems are very good. An additional PE can be inserted anywhere in the loop with the addition of a single communication path, and the flow of messages is not significantly affected by its presence. The failure-effect and failure-reconfiguration characteristics of DDL systems are poor, however. A single failure in a path or a PE interface causes intercommunication to stop (at least between PEs separated by the failed resource). If reconfiguration to mask the fault is necessary, there must be a fully redundant path structure and some type of bypass switching in the PE interfaces. Reconfiguration from DDL to another structure is not an obvious option either, since the paths are unidirectional and the interfaces are relatively simple. The logical complexity of communications in a DDL system is low; a PE must only relay messages, originate messages and transmit them to a single destination, recognize messages destined for itself, and strip off messages according to the discipline. The bandwith of the single loop is, of course, a potential bottleneck as communication rates increase. In addition, some loop disciplines have the weakness that a single user, possibly with malicious intentions, can saturate the entire available bandwidth.

DDL architectures that have been proposed or implemented have almost all used bit-serial data links as the communication paths between PEs. This, together with the delay involved in relaying the messages, has resulted in significant increases in message transit times around the loop. In general, these systems have been designed for applications where reliability and performance constraints were not stringent. The primary goal of most designs has been the interconnection of geographically dispersed minicomputer systems to allow file and resource sharing. Thus, reconfiguration after failure has not been performed, nor has the message delay-time been a problem.

The best-known example of a DDL computer system is the Distributed Computer System at the University of California, Irvine [FARB72]. This system originally developed out of an interest in data communications rather than from a concern with distributed computing, although that emphasis was reversed early in the history of the project. The Distributed Computer System currently consists of five minicomputers and a number of peripheral devices looped around the Irvine campus. The loop (or

"ring") is bit-serial and operates at a data rate of 2.3 Mbs. A multiplicity of variable-length messages can circulate simultaneously. Fault tolerance is provided by a redundant loop and bypass switches.

The already good place-modularity inherent in loops has been enhanced in the Distributed Computer System by the incorporation of "soft" or "associative" addressing of messages. Rather than sending a message to a physical processor, it is sent to a logical process; the "Ring Interface" corresponding to the processor in which the destination process currently resides recognizes the address and accepts the messages. This allows communication to be independent of the number of processors in the system, and of process/processor assignments. This idea has also been incorporated into other distributed architectures.

A loop version of the CAMAC data multiplexing system has been defined for both bit-serial and byte-serial transfers [AEC73]. Originally designed for nuclear laboratory instrumentation, the CAMAC loop is appearing in distributed computing schemes, although its protocol is not well suited for such use.

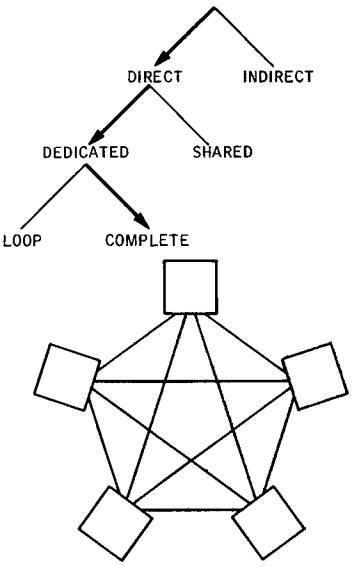

FIGURE 3. DDC (Complete interconnection).

DDC—Complete Interconnection

The DDC architecture is perhaps the conceptually simplest design type in the taxonomy. In it (Figure 3), each processor is connected by a dedicated path to every other processor in the system, and messages between processors are transferred only on the path connecting them. The source processor must choose the path to the destination processor from the alternative paths available, and all processors must be equipped to handle incoming messages on a multiplicity of paths.

The most significant characteristic of DDC systems is their poor cost-modularity. The addition of the nth processor to a DDC system requires not only the addition of n-1 paths between it and the other processors, but also, all processors in the system must have facilities for accepting the incremental PE as a data source. Thus, their interfaces must have at least M-1 ports, where M is the maximum size of the system. Alternatively, it must be possible for all processors to accept extra connections when the number of PEs must be increased beyond the number of available ports. Place-modularity of DDC systems is good, as are failure-effect and failure-reconfiguration characteristics. The DDC architecture is one which can be easily degraded in the event of a failure without changing its interconnection class—a failed processor, or one of the two processors terminating a failed path, can simply be disconnected from the system. In addition, reconfiguration to an Ixx system could be used in event of a failed path if the software cost and increased message transit time incurred were acceptable. DDC systems have no obvious bottlenecks, and their logical complexity is relatively low. It should be noted, though, that the architecture forces a location-addressing policy on interprocess communication, since switching within the processors and message relaying activities would put a design into the ICDx or IDDx categories.

DDC systems may be geographically either localized or dispersed, although there are few examples of either case. The best-known instance of a localized DDC architecture is a fully connected version of the IBM Attached Support Processor System [IBM],

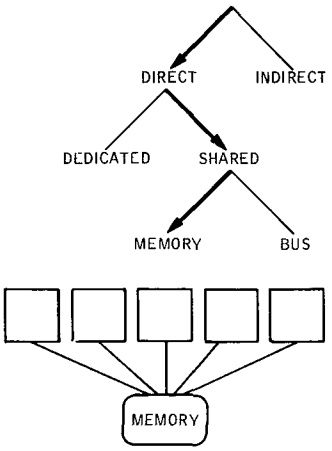

FIGURE 4. DSM (Multiprocessor).

in which up to four System/360 or /370 computers may be linked through I/O channel couplers.

Virtually all extant examples of geographically dispersed DDC systems are small ($\leq$3 PEs), and appear to be ad hoc interconnections of formerly existing computer installations, as exemplified by the fully connected configuration of the MERIT system [BECH72]. MERIT consists of two IBM 360/67s and a CDC 6500 located on three separate Michigan University campuses, and connected by common carrier lines. The MERIT system design does have the potential for less fully connected configurations (such as IDDI) to reduce communications costs.

DSM—Multiprocessor

Certainly the most common way to interconnect computer systems is the DSM or multiprocessor architecture (Figure 4), in which two or more processors communicate by leaving messages for one another in a commonly-accessible memory. The key characteristic of DSM architectures is that the memory is, or can be, used as a path rather than solely as storage.

The place-modularity of DSM systems is very good; it is possible to add processors arbitrarily (since the processors are not topologically distinguished), and it is also possible to increase the in-transit message capacity of the path simply by increasing the size of the memory. The cost-modularity of DSM systems depends almost completely on the path structure by which the processors access the memory system. If each processor is provided with a direct path, then cost-modularity can be poor, since an incremental processor can possibly bring the total to greater than the number of available memory ports. Alternatively, if the memory is accessed through a single bus with a suitable allocation mechanism, cost-modularity can be very good. A DSM system is quite vulnerable to a bottleneck in which the memory's bandwidth becomes a restriction on communication rates. Cost-modularity is poorer in this case, as it is expensive to increase bandwidth of the memory or the access path. Logical complexity of DSM systems is quite low. The failure-effect and failure-reconfiguration characteristics of DSM systems are good in the case of processor failures, but poor in the event of failure of the central memory unit (or of a shared access bus). There is also a software failure-effect problem because processors normally have unrestricted access to the central memory, thus faulty or malicious software can prevent or damage message transactions to which it is not a party.

Almost every implementation of a DSM system has occurred because the designer(s) wished the memory to be shared as a storage place for programs and data—use of the memory as a communication path has almost been a side effect. In implementing this multipurpose sharing, it has been found that the systems' performance has increased more slowly as the number of processors increased and, in general, systems consisting of more than about four processors have not been cost-effective. The reason for this has been the extreme contention for memory bandwidth when the (functionally) single memory must serve for all purposes. The bandwidth required for communications alone is, however, unlikely to cause bottlenecking in a memory solely dedicated to this function.

An example of a contemporary multiprocessor is the Carnegie-Mellon C.mmp [WULF72], which allows up to 16 processors to share up to 16 memory modules through a crossbar switch. Currently, five PDP-11/20

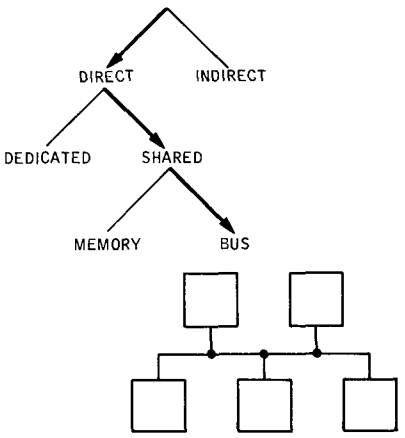

FIGURE 5. DSB (Global bus).

processors are operating with negligible interference.

The surveys of Miller, et al. [MILL70], and Enslow [ENSL74] include a wide variety of both commercial and aerospace DSM machines.

DSB—Global Bus

The DSB architecture, shown in Figure 5, comprises a number of processing elements interconnected by a common, or global, bus. Access to this bus is shared among the processors by some allocation scheme, and messages are sent directly from the source PE onto the bus, to be recognized and accepted by the proper destination(s).

Both the cost- and the place-modularity of DSB systems are good with respect to the PEs. Depending on the choice of bus allocation scheme, it can be possible to add a processor to the system in any position with little or no effect on the other PEs. Cost- and place-modularity of the communications path are poor, however. It is not possible to increase the bandwidth easily as needed, nor is it often possible to increase performance only where needed. Rather, to increase performance it is usually necessary to change the implementation of the entire bus or to replicate it, alternatives which have significant design impact on the bus interfaces of all PEs in the system. Similarly, the failure-effect and failure-reconfiguration characteristics of DSB systems are very good with respect to the processors, and poor with respect to the bus. For PE failures, the DSB architecture requires no overt hardware reconfiguration activity to continue operation as a DSB system. Given reasonable care in the design of the bus interfaces, processor failures will have little effect on system operation. Failures of the bus, however, are inevitably catastrophic, and replication is required if the DSB architecture is to be retained after configuration. The global bus is, of course, a potential bandwidth bottleneck.

Much of the current interest in DSB systems has occurred in the aerospace environment, where both serial and parallel paths are being used. In these applications, the place- and cost-modularity characteristics of DSB systems are particularly advantageous, and allow flexibility in configuring systems for specific applications, including both homogeneous and nonhomogeneous instances of the same basic system. The failure-reconfiguration characteristics of the DSB architecture are useful for these applications, too, because most aerospace applications are of a real-time nature and, therefore, excessive reconfiguration delays must be avoided. Serial bussing for the communication path seems to be the more popular approach, primarily because the system applications tend to have high costs associated with physical wiring, a need for physical dispersibility over electrically long distances, and relatively low data rates. Replication of the communications path is the predominant technique for mitigating both the bandwidth restriction and the fault vulnerability of the shared bus.

Most of the aerospace DSB-like architectures have been hybrids with respect to our taxonomy [SYMS72], [ANDE73]. However, the military services are beginning to establish data multiplexing standards [USAF73] that will require the use of DSB approaches in future aerospace systems (although these initial standards leave something to be desired, from a distributed computing standpoint).

A pure DSB architecture is typified by one utilizing word-wide busses [JENS75], and a philosophical derivative of it (using a bit-

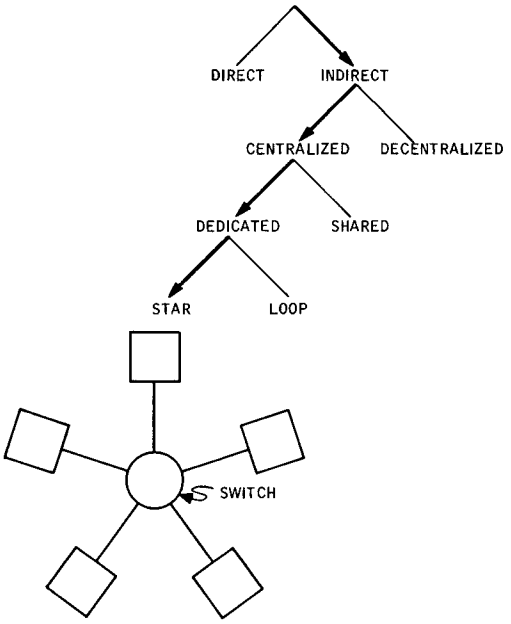

FIGURE 6. ICDS (Star).

serial bus), which has been constructed by the authors for the US Navy (for which there are as yet no public references).

DSB is also a popular line-sharing discipline in industrial and laboratory automation applications [ARON71], as exemplified by the CAMAC [COST72] and IEEE standard 488-1975 (originally Hewlett-Packard [KNOB 75]) data-multiplexing systems.

ICDS—Star

ICDS systems (Figure 6) consist of a central switching resource to which a number of processors are connected, each by a functionally single, bidirectional path. Messages are exchanged among the PEs using the central switch as an intermediary; it is the apparent destination and source for all messages. The function of the switching resource is usually seen as "insulating" the processes running on a given PE from physical knowledge of the system, and protecting them from each other.

The ICDS architecture has most features in common with the DSx architectures because both have shared message transfer facilities. Its cost- and place-modularity are good with respect to the PEs, and poor with respect to the central resource. Similarly, failure-effect and failure-reconfiguration characteristics are good for the PEs, and poor with respect to the switch. Bottlenecking in the switch is a potential problem. The connection flexibility of ICDS systems is poor because incremental PEs must always be provided with individual paths to the central switch. The logical complexity of ICDS systems is moderate. Sufficient information (e.g., routing tables) must be provided within the switching resource to allow communications to take place, but the fact that there is only a single copy of this (usually dynamic) information simplifies its handling during reconfiguration. The poor failure-effect and failure-reconfiguration characteristics of the central resource are extended to one of the PEs in the system if switching information is maintained outside the resource (i.e., where the switch uses address translation to accomplish message routing, and the mapping registers are under control of one of the PEs). ICDS architectures are quite common. One example is IBM's Network/440 [MCKA70], in which remote System/360 user nodes are connected over leased lines to a 360/91 central controler.

ICDL—Loop with Central Switch

In a manner analogous to Direct, Dedicated path (DDx) systems, the direct connections required for ICDx architectures can be implemented either in the ICDS star pattern or in a loop, which we call ICDL (Figure 7). In such a system, messages are placed on the loop by senders, removed for an address-mapping operation by a central switching element, then replaced on the loop properly addressed to their intended destination.

ICDL systems share characteristics with both the ICDS and the DDL organizations. Their failure characteristics are those of DDL with respect to the data paths, and those of ICDS with respect to the central resource. Connection flexibility is improved over the star in that the incremental PE need be connected only to its physically nearest neighbors, not to a possibly remote control element. As in DDL systems, bottlenecks and vulnerability to malicious users

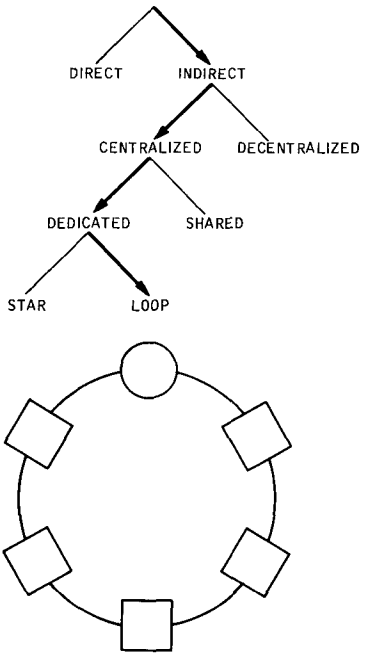

FIGURE 7. ICDL (Loop with central switch).

are potential problems, but since these are risks inherent also in the centralized switching of the ICx approach, they do not necessarily weaken the system over ICDS. The logical complexity of ICDL systems is moderate, though slightly increased over ICDS due to the additional demands the loop discipline places on the PE interfaces.

An example of the ICDL architecture is SPIDER, an experimental data communications system interconnecting eleven computers at Bell Laboratories in Murray Hill, N. J. [FRAS75]. In this system, multiple fixed-size data messages circulate over 1.544 Mbs common carrier lines; currently there are three loops, but only one is in use. The central switch for all loops is a minicomputer. Any computer in the loop (except the central switch) can be switched to receive-only in case of failure.

ICS—Bus with Central Switch

The ICS architecture shown in Figure 8 is functionally equivalent to ICDS, with the major exception that the processors are not individually connected to the switching resource but, instead, share a path by which to access it. Thus, when a PE wishes to transmit a message, it must first acquire the bus, then transmit the message to the switch. From the switch, the message is retransmitted over the (functionally) same bus to its proper destination. (This retransmission is the characteristic by which ICS systems can be distinguished from DSB organizations and from DSM systems using a single bus to memory.)

As might be expected, the characteristics of the ICS architecture are similar to those of ICDS systems. By the failure-effect measure it is poorer, since the access path is no longer replicated once for each processor. The existence of the shared path to the switch need not contribute significantly to bottlenecking, however, since it is quite feasible to balance its performance with that of the switch. Then, as long as the two saturate at the same time, or the switch saturates first, the bottlenecking risk is not increased over the ICDS organization. The cost-modularity of the ICS system is influenced positively by the fact that the incremental processor need only be connected to the shared bus, not directly to the (possibly remote) switch. As mentioned in the discussion of direct, shared bus (DSB) systems, certain bus designs make this connection cost very low.

The low popularity of ICS systems may be inferred from the fact that there is no com-

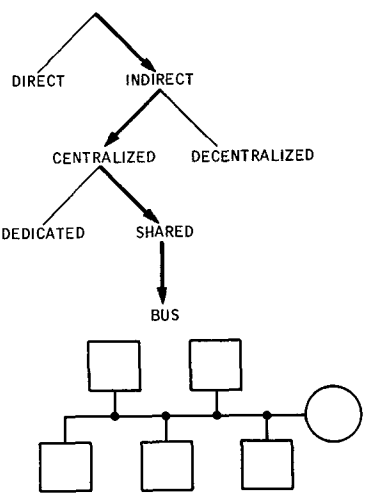

FIGURE 8. ICS (Bus with central switch).

mon name for them. The designs that do exist, however, use processors as the switching resource, with processors dispersed over geographically short, but electrically long distances. Although nothing intrinsic to the architecture prevents the use of hardware (such as a crossbar) for the switch, the speed implications on the bus, if balance is to be achieved, probably preclude its use as a practical matter. As mentioned previously, the shared bus is not normally available from commercial carriers, so ICS systems are restricted geographically.

An ICS system has been designed by Hughes for the US Navy [ROWA74]. The centralized switching resource is a special-purpose processor called the Network Manager, which interfaces with a number of functional nodes over one or more shared busses. In addition to handling interprocessor communication, the Network Manager also provides a variety of other executive services for the nodes. Because the system is physically localized, the bus(ses) to the Network Manager are bit-parallel.

The ALOHA system [ABRA73] is a unique ICS computer network which uses a radio "bus." Hardwired and incoherent light links are also accommodated. A 24K-baud full duplex radio channel connects the central IBM 360/65 to remote terminals, which include a number of minicomputers.

IDDR—Regular Network

The IDDR organization (Figure 9) comprises a number of PEs interconnected with dedicated paths and having identical neighbor relationships (except, perhaps, at the boundaries). In the figure, each PE has "left," "right," "above," and "below" neighbors, although other geometries have been proposed. Messages are routed through the network from source to destination with each intervening PE determining which of its alternative neighbors should be the next recipient of the message. (It may be noted that DDL is a special IDDR case in which each PE has two neighbors and there is no switching decision to be made.)

The modularity and failure characteristics of IDDR systems are significantly and negatively affected by the requirement for absolute regularity. It is not possible to add only a single PE or path; rather, the size of the increment depends on the number of PEs in the system, the number of neighbors per PE, and the interconnection pattern. Thus, both cost-modularity and place-modularity are extremely poor. There is no connection flexibility. Logical complexity of the design is moderate; although many switches exist, the regularity of connection simplifies routing. The failure-effect measure of IDDR systems is moderate-to-good, depending on the method by which message routing is performed. A single PE or path failure does not stop communication entirely in any case, nor need it stop any messages to which the failed element is not a party. Failure-reconfiguration of IDDR systems is exceedingly poor, with 100% sparing of PEs and paths being required if the basic structure is to be unaltered after reconfiguration. If reconfiguration to an irregular structure (IDDI) is acceptable, however, this shortcoming is eliminated.

The elegance of IDDR structures has caused significant academic interest but, to the authors' knowledge, their practical

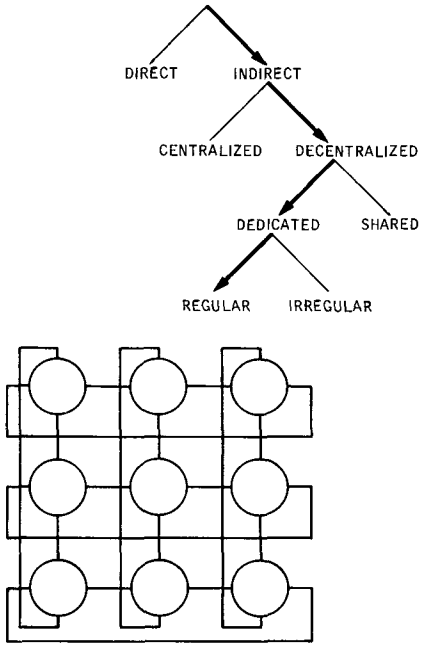

FIGURE 9. IDDR (Regular network).

difficulties have prevented actual implementations. Paper designs have usually been for four-neighbor (rectangular) systems, with either busses or memories proposed as paths. One nonrectangular IDDR system is TREE, designed at the US Navy Postgraduate School [GOOD73]. This machine comprises a number of PEs connected as a tree. A PE may communicate with its superior or any of its subordinates in the hierarchy; because there is only one path between any two nodes in a tree, the logical complexity of the message routing is not severe. TREE was envisioned primarily as being physically centralized, but could instead be implemented with common carrier lines for greater dispersal.

IDDI—Irregular Networks

The distinction between IDDR and IDDI (Figure 10) systems is simply that consistent neighbor relationships are not required for IDDI. Thus, a given PE may have from one to an arbitrary number of neighbors with which it communicates. Many of the system characteristics vary with the degree of interconnection regularity. Place-modularity of IDDI systems tends to be extremely good, with both processors and paths added as, and where needed. Cost-modularity is similarly good, with incremental PEs being architecturally required to have only one (or two in loop-oriented approaches) connections to the rest of the system. Connection flexibility is another advantage, with connections allowed to any PE in the system. The more irregular the interconnection pattern, the more the system's failure-effect and failure-reconfiguration characteristics can be enhanced by providing a multiplicity of potential paths between PEs. Likewise, irregularity improves the extent to which reconfiguration can be performed without departing from the IDDI category. The logical complexity of IDDI systems is usually very high; at each switch, routing decisions must be based on knowledge of the overall system topology. Due to the good place-modularity, bottlenecks are not a likely problem.

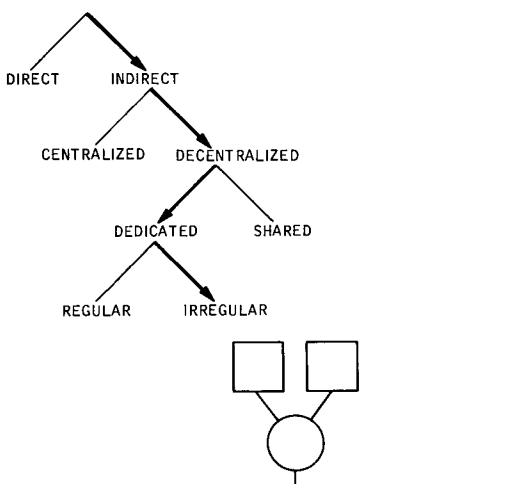

FIGURE 10. IDDI (Irregular network).

The dominant current application of IDDI interconnection is to geographically dispersed computer networks. In such systems the paths are supplied by a common carrier and the switching is done by processors dedicated to that function. The (significant) cost due to logical complexity, and represented by the switch-processors, is incurred to minimize the number of high-cost interconnection paths required. Routing algorithms are usually inelegant, especially for the less regular systems, and ad hoc solutions are apparent. It is the authors' feeling that hardware switching facilities are ruled out by the complexity of the decisions to be made, and that the current practice of using processors for this function must necessarily continue.

Most systems commonly called "computer networks" are IDDI structures. These are discussed in the surveys by Schneider [SCHN73], Abramson [ABRA73a], and Rustin [RUST72], and in the bibliography by Blanc, et al. [BLAN73].

A more regular sort of IDDI system is represented by the coupled loops of Pierce [PIER72], [COKE72], [KROP72], intended for telecommunications. A number of independently controled DDL "local" loops are coupled together, directly and/or by "trunk" loops (which may themselves be

DDL). A variety of alternate, redundant, and bypass-switched loops is proposed to provide failure protection.

IDS—Bus Window

In IDS architectures, an example of which is shown in Figure 11, access to the switching resources is via a path shared by multiple PEs. Switching is performed by more than one resource, and messages may be retransmitted onto the path from which they were received, or onto another.

The modularity characteristics of the IDS architecture are similar to those of the IDDI structure. The failure-effect and failure-reconfiguration characteristics are poorer, however, because multiple PEs and switches can be affected by the failure of a single path. Also, systems of this type are not easily dispersible due to the shared busses.

Digital Equipment Corporation manufactures a device called the DA11-F Unibus® Window to facilitate the implementation of IDS architectures [FITZ73], [DEC75]. Similar mechanisms have been designed at Carnegie-Mellon University for their Computer Module System [FULL73], and at BBN for their PLURIBUS IMP [HEAR73] (which we classify as a hybrid, as discussed in the Conclusion of this paper). All of these units provide a bidirectional path between the busses of two minicomputers. Blocks or segments in the source-address space are translated to the destination-address space. A number of these interfaces may be used to construct hierarchies of processing elements. Simulation and programming experience by users of all three mechanisms indicate that the logical complexity of this approach grows rapidly as the number of translation levels increases, and as the translation binding becomes more dynamic. This type of interconnection is also subject to deadlock [BELL73], [CHEN74], unless designed and used with great care.

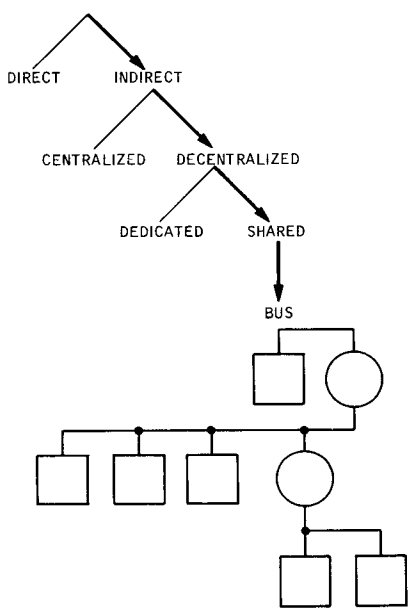

FIGURE 11 IDS (Bus window)

FUTURE DIRECTIONS

Current activity in distributed architectures is undeniably more on paper than in hardware, but there do seem to be trends emerging in designs for several application areas.

DDL organizations seem to predominate in designs where their excellent modularity and compatibility with common-carrier data paths can be used to advantage, and where their poor failure characteristics and long message transit-times are not a problem. Typically, the DDL designs are used to connect multiple minicomputer systems, dispersed over electrically long distances in university, research laboratory, or industrial automation environments. The individual PEs are usable both as stand-alone systems and with other PEs for resource (file, peripheral) and load sharing.

The DSB organization with serial bussing is becoming the dominant architecture in real-time control environments. These applications typically have a number of iterative, sampled-data control loop functions loosely connected by a requirement for information exchange and centralized monitoring. The environment occurs in both the conventional process-control type application and in aerospace applications such as integrated avionics processing. In such systems, the good place-modularity for processors in DSB organizations is particularly

valuable. DSB is chosen over DDL because simple redundancy of the global bus allows fast and almost automatic reconfiguration after failures, and because of the reduced message transit-time.

In environments where sophisticated interprocess communication is a requirement, the ICx organizations are a common choice. Compared to IDx, the lower logical complexity of centralized switching is believed to outweigh its failure-effect and failure-reconfiguration disadvantages. There is a great variety of system designs, particularly in the ICDS category, with varying levels of performance in the central switch area. It is the feeling of the authors that as understanding of interprocess communication matures, movement toward IDx structures may occur.

The fourth "dominant" system is IDDI, which is uniquely suited to applications requiring interconnection of relatively large computers over geographically long distances. In such systems, the high interconnection cost necessitates that only large computers be involved. Furthermore, significant processing power should be dedicated to the switching function in order to minimize those interconnection costs. At this time, such applications have used almost exclusively ad hoc solutions to controling the complex IDDI structures, with the majority of the efforts going toward more fundamental problems of intercommunications protocols and experiments in load sharing. It is hoped that future work in such areas as routing algorithms and recovery techniques for IDDI structures will complement interprocess communication research currently being conducted with ICDx structures, to the ultimate benefit of both.

In our opinion, the remaining five system types each have one or more significant weaknesses not sufficiently compensated for by strengths that are useful in real-world applications. DDC systems have poor modularity and high interconnection costs, to the extent that large systems are impractical. DSM architectures, from an intercommunications viewpoint, have little to offer over DDL or DSB structures except an ability to communicate large messages quickly by use of pointers. (They will undoubtedly continue to exist for their other advantages, which are not related to communications.) IDDR and IDS organizations both incur significant logical complexity with few compensating advantages. Further, IDDR has the poorest modularity and failure characteristics of any of the organizations.

CONCLUSIONS

The notion of a taxonomy requires both that all species be clearly identified, and that the methods used to make distinctions be clear and unambiguous. Our discussion here has failed to achieve either of these goals completely, but has, we hope, been a worthwhile beginning. All species (systems) are not clearly identified; there are many hybrids using combinations of our leaf-node architectures. The PRIME system at the University of California, Berkeley [QUAT72] is physically a combination of DSM (multiprocessor) and ICDS (star), although logically it is purely ICDS since all interprocessor communication goes over the External Access Network, rather than through the shared memory. Honeywell's DP/M system for the US Air Force [ANDE73], with its global and local busses, uses two levels of DSB. The BBN PLURIBUS IMP for the ARPANET [HEAR73] is structurally an amalgam of DSM (although by convention there is no interprocessor communication using the local memories), IDS (for accessing shared memory), and ICS (the Pseudo-Interrupt Device).

Neither are all distinctions crystal clear; we have classified systems on the basis of "dominant" features, and to some degree even on our understanding of the intent, as well as on prima facie evidence. Our goals in attempting to construct a taxonomy will be satisfied, however, if two effects occur. First, we hope to stimulate a refinement of this approach or the development of a better one. Secondly, we hope to have contributed to a common ground for discussion in the meantime.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the valuable comments and criticisms of the Editors and the referees of COMPUTING SURVEYS. We are also indebted to many colleagues who informally refereed this paper. Special thanks go to Kenneth J. Thurber for his suggestions regarding the figures, which substantially improved the readability of the paper.

REFERENCES

[ABRA73] ABRAMSON, NORMAN, "The ALOHA System," *Computer-communication networks*, Prentice-Hall, Englewood Cliffs, N J , 1973. (ICS)

[ABRA73a] ABRAMSON, NORMAN; AND KUO, FRANKLIN F (Eds), *Computer-communication networks*, Prentice-Hall, Englewood Cliffs, N. J , 1973 (Survey)

[AEC73] AEC/NIM AND ESONE, CAMAC *Serial System Organization*, National Tech. Information Service, TID-26488, December 1973, *Addendum and Errata*, May 1975. (DDL)

[ANDE73] ANDERSON, GEORGE A., "Interconnecting a distributed processor system for avionics," *Proc. Symposium on Computer Architecture*, December 1973, IEEE, N Y., 1973 (Hybrid)

[ARON71] ARONSON, RICHARD L , "Line-sharing systems for plant monitoring and control," *Control Engineering*, January 1971. (Survey)

[BECH72] BECHER, WILLIAM D.; AND AUPPERLE, ERIC M. "The communications computer hardware of the MERIT Computer Network," *IEEE Trans. Communications*, June 1972. (DDC)

[BLAN73] BLANC, ROBERT P.; COTTON, IRA W ; PYKE, THOMAS N JR.; AND WATKINS, SHIRLEY W *Annotated bibliography of the literature on resource sharing computer networks*, National Technical Information Service, COM-73-50750, September 1973. (Bibliography)

[CHEN74] CHEN, ROBERT C , *Bus communication systems*, University Microfilms Dissertation 74-20493, 1974. (Control and Communication)

[COKE72] COKER, C. H , "An experimental interconnection of computers through a loop transmission system," *Bell System Tech J.*, July/August 1972. (IDDI)

[COST72] COSTRELL, LOUIS, CAMAC—*a modular instrumentation system for data handling; revised description and specification*, National Technical Information Service, TID-25875, July 1972. (DSB)

[DEC75] DIGITAL EQUIPMENT CORPORATION, *PDP-11 peripherals handbook*, 1975 (IDS)

[ENSL74] ENSLOW, PHILLIP H (ED.), *Multiprocessors and parallel processors*, John Wiley and Sons, New York, 1974. (Survey)

[FARB72] FARBER, DAVID J.; AND LARSON, KENNETH C. "The system architecture of the distributed computer system—the communications system," *Proc. Symposium on Computer-Communications Networks and Teletraffic*, April 1972, Polytechnic Press, Brooklyn, New York, 1972 (DDL)

[FAEM69] FARMER, W O ; AND NEWHALL, E. E "An experimental distributed switching system to handle bursty computer traffic," *Proc. ACM Symposium on Problems in the Optimization of Data Communications*, October 1969, ACM, New York, 1969 (DDL)

[FITZ73] FITZGERALD, BRIAN, "Standard interfaces promote new minicomputer networks," *Electronics*, September 13, 1973. (IDS)

[FRAS75] FRASER, A. G., "SPIDER—An experimental data communications system," *Proc. Internal Conf on Communications*, June 1974, IEEE, N. Y., 1974 (ICDL)

[FULL73] FULLER, SAMUEL H.; SIEWIOREK, DANIEL P.; AND SWAN, RICHARD J. "Computer modules· an architecture for large digital modules," *Proc. Symposium on Computer Architecture*, December 1973, IEEE, N. Y., 1973. (IDS)

[GOOD73] GOODWIN, RICHARD J., *A design for distributed-control multiple-processor computer system*, National Technical Information Service, AD-772 883, December 1973. (IDDR)

[HEAR73] HEART, F. E ; ORNSTEIN, S. M.; CROWTHER, W R ; AND BARKER, W B. "A new minicomputer/multiprocessor for the ARPA Network," *Proc AFIPS 1973 National Computer Conf.*, AFIPS Press, Montvale, N. J , 1973. (Hybrid)

[IBM] IBM, *IBM System/360 and System/370 Attached Support Processor Version 3 Asymmetrical Multiprocessor System: general information manual*, GH20-1173. (DDC)

[JENS75] JENSEN, E. DOUGLAS, "A distributed function computer for real-time control," *Proc Symposium on Computer Architecture*, January 1975, IEEE, N. Y., 1975. (DSB)

[KNOB75] KNOBLOCK, DARYL E ; LOUGHRY, DONALD C.; AND VISSERS, CHRIS A "Insight into interfacing," *IEEE Spectrum*, May 1975. (DSB)

[KROP72] KROPFL, W. J., "An experimental data block switching system," *Bell System Tech J.*, July/August 1972. (IDDI)

[MART72] MARTIN, JAMES, *Systems analysis for data transmission*, Prentice-Hall, Englewood Cliffs, N. J., 1972. (Digital Data Communication)

[MCKA70] MCKAY, DOUGLAS B.; AND KARP, DONALD P. "IBM Computer Network/440," *Computer Networks—Proc.*

Courant Institute Symposium, November 1970, Prentice-Hall, Englewood Cliffs, N J , 1972. (ICD)

[MILL70] MILLER, JAMES S.; LICKLY, DANIEL J.; KOSMALA, ALEX L.; AND SAPONARO, JOSEPH A. *Multiprocessor computer system study*, National Technical Information Service, N70-41238, March 1970. (Survey)

[PIER72] PIERCE, JOHN R., "Network for block switching of data," *Bell System Tech J* , July/August 1972. (IDDI)

[PIPP75] PIPPENGER, NICHOLAS, "On crossbar switching networks," *IEEE Trans Communications*, June 1975. (Explicit Switch)

[QUAT72] QUATSE, JESSE T.; GAULENE, PIERRE; AND DODGE, DONALD, "The external access network of a modular computer system," *Proc AFIPS 1972 Spring Jt. Computer Conf.*, AFIPS Press, Montvale, N. J., 1972. (Hybrid)

[REAM75] REAMES, CECIL C., AND LIU, MING T. "A loop network for simultaneous transmission of variable-length messages," *Proc. Symposium on Computer Architecture*, January 1975, IEEE, N. Y., 1975. (DDL)

[ROWA74] ROWAN, J H ; SMITH, D. A ; AND SWENSEN, M. D. "Toward the design of a network manager for a distributed computer network," *Proc. Sagamore Conference on Parallel Processors*, August 1974, American Elsevier Publ. Co., New York, 1975. (ICS)

[RUST72] RUSTIN, RANDALL (ED.), *Computer Networks—Proc Courant Institute Symposium*, November 1970, Prentice-Hall, Englewood Cliffs, N. J , 1972. (Survey)

[SCHN73] SCHNEIDER, MICHAEL, *A survey report on computer networks*, National Technical Information Service, PB-231876, March 1973. (Survey)

[SIEW74] SIEWIOREK, DANIEL P., "Modularity and multi-microprocessor structures," *Proc. Seventh Annual Workshop on Microprogramming*, October 1974, ACM, N. Y., 1974. (Taxonomy)

[SYMS72] SYMS, G. H., *All applications digital computer: course notes*, National Technical Information Service, AD-767 327 March 1972. (Hybrid)

[THUR72] THURBER, KENNETH J.; JENSEN, E. DOUGLAS; JACK, LARRY A.; KINNEY, LARRY L.; PATTON, PETER C.; AND ANDERSON, LYNN C. "A systematic approach to the design of digital bussing structures," *Proc. AFIPS 1972 Fall Jt. Computer Conf.*, AFIPS Press, Montvale, N. J., 1972 (Control and Communication)

[THUR74] THURBER, KENNETH J , "Interconnection networks—a survey and assessment," *Proc. AFIPS 1974 National Computer Conf.*, AFIPS Press, Montvale, N. J., 1974. (Explicit Switches)

[USAF73] US AIR FORCE, *MIL-STD-1553: military standard aircraft internal time division multiplex data bus*, August 1973. (DSB)

[WEST72] WEST, LYNN P., "Loop-transmission control structures," *IEEE Trans. Communications*, June 1972. (DDL)

[WULF72] WULF, WILLIAM A.; AND BELL, C. GORDON, "C.mmp—a multi-mini-processor," *Proc. AFIPS 1972 Fall Jt Computer Conf.*, AFIPS Press, Montvale, N. J., 1972. (DSM)