DISTRIBUTED PARALLEL COMPUTING ARCHITECTURE FOR MONITORING AND CONTROL OF LARGE PHYSICAL PROCESSES

Thesis for the Degree of M. S.
MICHIGAN STATE UNIVERSITY
JAMES E. SIEBERT
1977

ABSTRACT

DISTRIBUTED PARALLEL COMPUTING ARCHITECTURE FOR MONITORING AND CONTROL OF LARGE PHYSICAL PROCESSES

By

James E. Siebert

Distributed-intelligence computing systems can successfully automate remote monitoring and control operations within large physical processes in a cost-effective, practical manner. When computing intelligence is located at the process points in addition to a central control location, neither communication channel bandwidth nor sequential task performance limit the real-time operation as they do in a centralized system. Process point control activities and data processing which do not involve the resources of the central host facility are performed locally at the remote sites. This distributed-intelligence approach minimizes data communication; improves real-time performance; isolates performance of system tasks that facilitates system development, expansion, maintenance, and fault tolerance; reduces system complexity and cost; and improves the accuracy of acquired data. Furthermore, the economies of LSI technology can often be exploited while maintaining fast and predictable response times.

The three elements that comprise a distributed computing system--a central host computing facility, satellite computing facilities, and communication links interconnecting host and

satellites--can be arranged in distributed star, common-bus-organized, or multi-level system configurations. Communication links often effect a recurring operating expense that is minimized by the design algorithm presented. The design process specifies a network of dial-up lines by determining the line data rate, host line requirements, optimal message block length, grade of service provided to the satellites, and wait time for outstanding satellite service requests.

The monitoring and control needs of the Water Quality Management Project typify a class of application areas best implemented by a distributed-star computing system. This approach provides the economy, flexibility, and ease of development and expansion needed to successfully manage the land treatment facility.

DISTRIBUTED PARALLEL COMPUTING ARCHITECTURE FOR MONITORING AND CONTROL OF LARGE PHYSICAL PROCESSES

By

James E. Siebert

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Electrical Engineering and Systems Science

1977

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to Dr. P. David Fisher for his thorough critical review of the manuscript. In addition, I wish to thank him and the Institute of Water Research for creating the environment that fostered and facilitated this investigation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Computing systems applied to automatically manage large and diverse physical systems are traditionally very complex, expensive facilities. The classical approach involves one or more large centralized processing systems; a complicated, under-utilized data communication network; and complex operating systems with high overhead. And unfortunately, there exists an inherent conflict between optimization criteria with this solution: good real-time control performance cannot usually be achieved without sacrificing high CPU productivity.

Viable architectural alternatives for such computing systems have recently emerged as the result of diminishing costs of computing intelligence--primarily the advent of Large Scale Integrated (LSI) circuits. By distributing computing intelligence geographically and functionally into remote processing elements, a distributed computing system results that overcomes many of the deficiencies of centralized systems. The distributed architectures exploit the cost-effectiveness of slow, inexpensive computing facilities while achieving ample real-time performance through parallel operation of all system processors.

The primary objectives of this thesis are to delineate the principal design considerations, investigate architectural alternatives, and examine the comprising elements of distributed-intelligence computing systems. Also, it will be shown that distributed systems can automate practicably and cost-effectively monitoring and control operations in large processes.

Relevant large processes are characterized principally by geographical size and may also encompass functional diversity. Throughout the thesis, a physical process is considered to be comprised of process points as depicted in Figure 1. Process points are those locations at which some activity of importance takes place. In general, both real-time monitoring and control operations must be provided at the process points to manage the overall process. And usually, these operations must be synchronized and coordinated among the process points. This in turn necessitates the communication of data and control information to support these operations at the process points.

The performance of a centralized architecture is limited by expensive data communications capacity and sequential task performance. These limiting conditions must be relaxed for improved real-time operating characteristics. This is accomplished in a distributed computing system by locating computing intelligence at the remote process points. Figure 2 shows a distributed computing system composed of a central

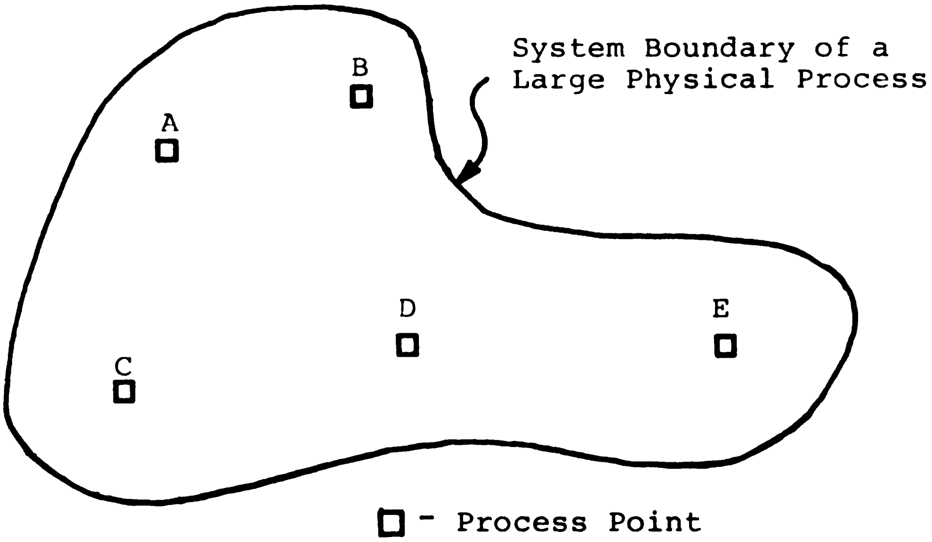


Figure 1. Process Points Within a Large Physical Process

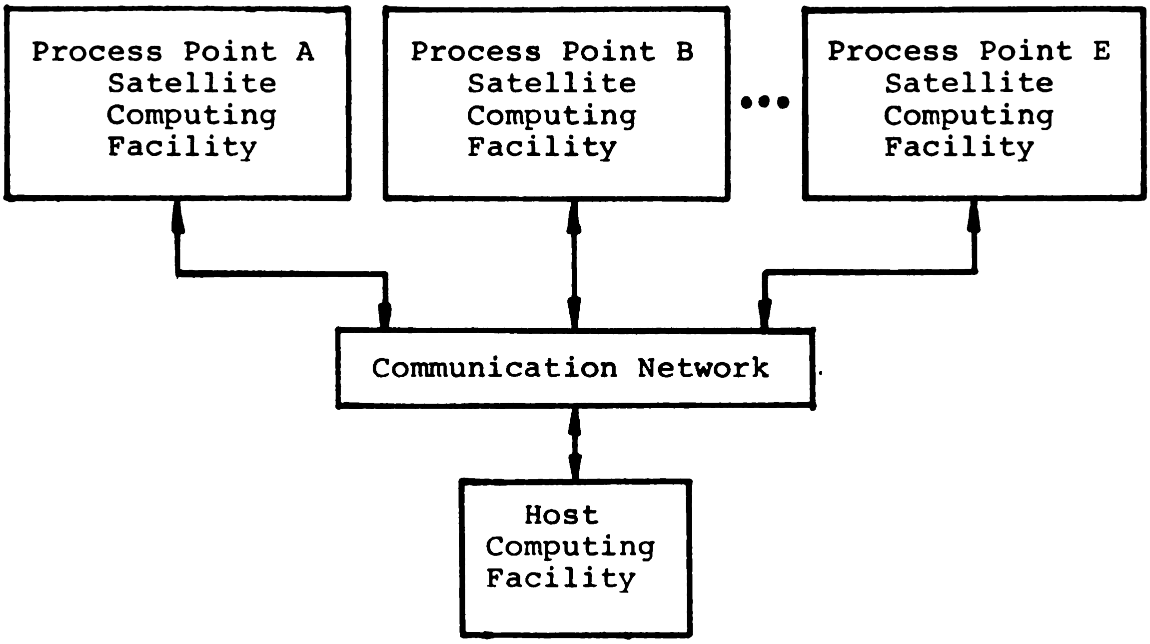


Figure 2. Representation of a Distributed Computing System

computing facility (referred to as the host computing facility), a communication network, and individual computing facilities at each process point (referred to as satellite computing facilities). All system hardware located at a process point comprises the "computing facility" including computers and their attached peripheral equipment, sensors, signal conditioners and converters, controlled process equipment, etc. Simultaneous operation of each computing facility realizes parallel task performance within the system. Additionally, routine indigenous tasks can be scheduled and controlled locally without information exchanges through the communication network. Hence, the two performance limiting factors found in a centralized system do not restrict a distributed system. These topics are developed fully in the succeeding chapters.

Beginning with some example large physical processes, Chapter II presents the general concepts of distributed-intelligence systems and the benefits/deficiencies of centralized and distributed architectures. In Chapter III, architectural alternatives are treated, and Chapter IV then details the individual system elements. Design considerations of the data communication links, especially for dial-up lines, are presented in particular detail. As a representative application, a data acquisition system for the Water Quality Management Project (Michigan State University) is proposed in Chapter V. Results of the investigations are summarized in Chapter VI.

CHAPTER II

BACKGROUND

In the following sections, the rationale for selecting a distributed computing system architecture over a centralized architecture is traced from both economic and performance viewpoints. The first two headings provide the perspective for the latter topics which generalize on the centralized and distributed system architecture alternatives.

2.1 Examples of Large Physical Processes

Managing a large process is intrinsically a remote monitoring, data communication, and control problem. The monitoring process produces the inputs for control algorithms, and for statistical and historical records (data base). The exact structure of the data communication network employed depends to a large extent upon the configuration of the physical process being monitored and controlled. Examples of geographically large processes that often require employment of common-carrier (telephone companies or other companies providing communications services) services for data communications are listed in Table 1 [1,2]. It is in these application areas that the principal impact of distributed computing systems lies since centralized control

Table 1

Application Areas Employing Common-Carrier Services

Environmental Quality Monitoring and Control
Climatological Data Gathering
Urban Automobile Traffic Control
Control of Power Generation, Transmission, Distribution
Oil and Natural Gas Pipeline Transmission and Distribution

Table 2

Application Areas Employing Private Communications Facilities

---

Climate Control, Security, and Fire Protection in Large Structures
Production Process Control
Material Handling Systems
Medical Electronic Monitoring
Retail Credit Authorization System

---

is inconsistent with the distributed nature of the application.

Some application areas also exist that are not geographically large but are themselves distributed in nature or functionally diverse. Table 2 provides a representative listing [3]. Communications in these cases would not occur over common-carrier lines, but rather, privately owned facilities.

2.2 Economics

Economics is the primary forcing function of change in the way computers are applied. Costs of computing hardware have plummeted--in 1975, CPU's cost approximately 0.5% of the 1960 cost. Simultaneously, fast access mass storage cost has dropped about 98% [4]. Today, semiconductor memory is a linear function of size; small memory modules are economically practical.[1] However, present common-carrier communication line costs amount to about 61% of the 1970 price [4]. Of all the elements in computing system cost, data communication costs have decreased the least. And, thus, a computing system architecture should incorporate these improvements in cost-effectiveness. For example, this rearrangement in relative costs produces an economic incentive to distribute intelligence for geographically and

---

[1] Overhead electronics costs in magnetic core memories encourage large centralized memory organization.

functionally distributed processes. But economic considerations are not the sole motivation; improved real-time performance also results.

2.3 The Historical Solution: The Centralized System

The essential elements comprising a centralized monitoring and control system are shown in Figure 3. Also represented are the basic network alternatives for communications with the process points. Local sites acquire and telemeter data and status information to a large computer located at a central managerial site. The required location of data-processing (DP) I/O equipment positions the central site. Here, the central computer executes system operation: processing and analyzing telemetered data, controlling various process loops, managing the data base, supporting interactive operator dialog, along with any background processing tasks. Such a complex approach to automation was dictated by behemoth costs of computing intelligence.

2.3.1 Inherent Deficiencies of the Centralized Plan

Three innate problems impair a centralized system plan: limited response time, deficient expandability, and high communications cost. Moreover, response time[2], throughput, and maximum system size are ultimately constrained by

---

[2] The amount of time which elapses between a stimulus to the system at a process point or at the central facility and the initiation of the required computing system response.

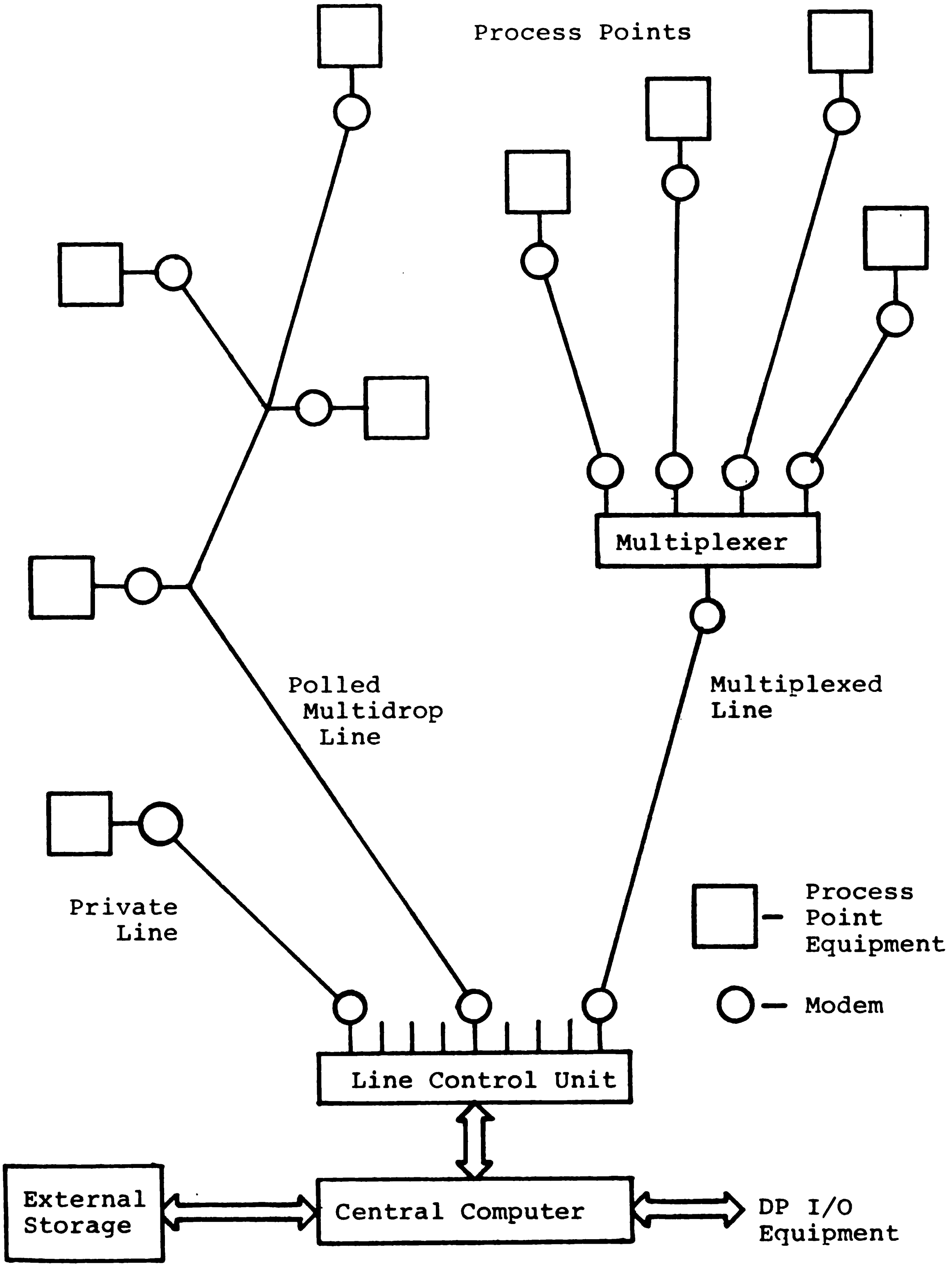


Figure 3. Block Diagram of a Large Centralized Monitoring and Control System

communications capacity and sequential task performance.

Response Time

Both real-time and general data processing functions[3] are interleaved and executed concurrently in a centralized system. Since the central computer is a large expensive resource, maximum investment return necessitates maximum utilization. If the system design is optimized for real-time performance, response times can be minimized. However, high CPU productivity of desired tasks, i.e., good throughput, is compromised heavily by the polling and wait modes that provide fast response.

In the centralized system, response times are unpredictable (within some statistical distribution) since various interleaved tasks possess dynamic computing resource requirements. CPU loading will occur in peaks and valleys causing variance in response delays. This picture deteriorates as the system is expanded. As new applications are added, response times become increasingly unpredictable.

Expandability

Ultimately, as a system is expanded, service delays increase to where real-time monitoring and control is no longer possible. Expansion is limited in terms of response

---

[3] General data processing functions include control processing; data management, statistical analysis, and summaries; and other background tasks.

time and response-time predictability. In addition, high implementation costs arise when new process points or functions are annexed. New functions require complex hardware/software changes to an already complex system and system downtime accompanies the implementation process.

High Communication Cost

Communication channel bandwidths are a key limiting factor on real-time performance. The range of capabilities provideable at a process point is restricted by the large amounts of information that must be exchanged between process points and the central computer, including raw data, control commands, and status dialog. And many controller functions must be executed by the central computer.

In order to achieve acceptable response times, high data rates must be employed. Channels with large bandwidth are required and concomitantly, the incurred costs high. There is a paradox here. A higher data rate is assigned to decrease the service time across the line; consequently, this expensive resource becomes underutilized, i.e., idle most of the time.

2.4 Distributed Intelligence Monitoring and Control Systems

In the distributed system, data acquisition, control, and data processing tasks are distributed spatially and functionally among several processing elements operating concurrently in a network. The difficulty of a task and the level of system resources required to execute a task determine the level (location) in the network at which the task is executed. By distributing system responsibilities away from a central computer, many routine activities can be performed requiring no communications with the central site. Hence, data communications shrinks as the limiting condition, with increased capabilities, real-time performance, and possible system size ensuing.

2.4.1 General Description

Instruments, sensors, and controlled devices are clustered around a computer(s) at a process point. Collectively, this hardware forms a satellite computing facility which serves as the process point manager. The process point computer(s), referred to as a satellite computer, includes the processor, memories, telecommunications interface, and the local I/O interfaces required to connect peripheral equipment (e.g., analog-to-digital converters, digital-to-analog converters, sensors, local display, teletype, etc.). Operations at the process point are controlled directly by the satellite computer which, in turn, is supervised by a central host computer. Located at a primary managerial site, i.e., the convenient location of DP I/O equipment, a host computer coordinates activities among remote sites, manages the data flow, maintains a central data base for analysis and summary reports, and supports operator/analyst interaction. Figure 4 presents some representative system topologies.

Through this approach, many benefits emerge. The host computer system can be optimized for throughput while the satellites are optimized for real-time activities. Execution of system tasks are performed in parallel by dedicated processors; faster response times and increased throughput result. These and other significant improvements are described in the next section.

2.4.2 Attendant Benefits of Distributing

Minimized Data Communications

The amount of communications that must occur between process points and the host computer in system operation is minimized as a result of the following:

1. Tasks indigenous to a process point are performed by a local satellite computer without host involvement
2. Data reduction at the process point
3. Local storage of historical information required by the local control algorithms

Furthermore, communication that must be done in real time is minimized, and the bulk of data communications can occur at the leisure of the network. As a result, the primary limiting factor in managing large processes is now allayed. Greater system size and diversity are possible at heightened performance levels.

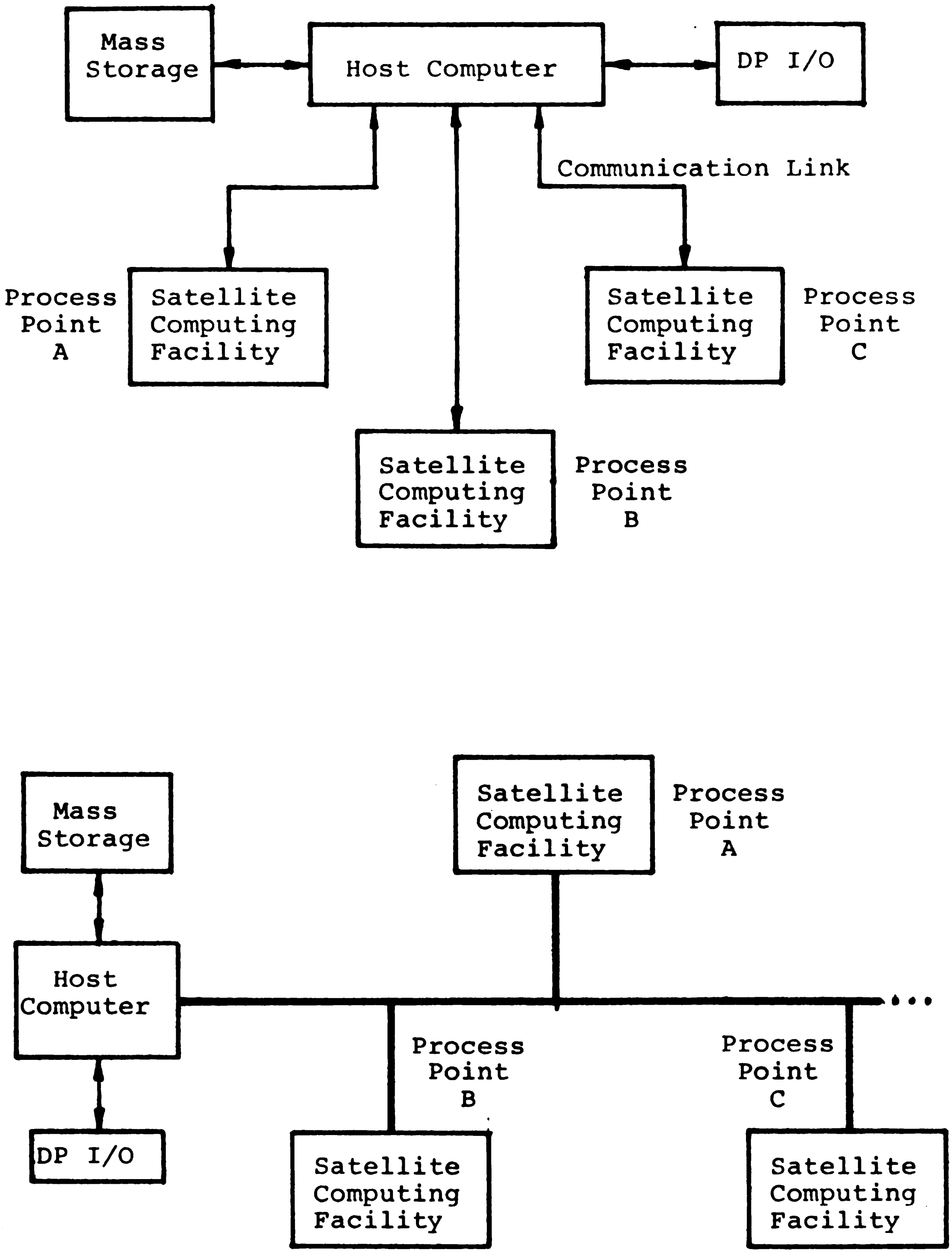


Figure 4. Simplified Distributed System Topologies

Improved Response Times and System Throughput

Satellite computers are machines dedicated to the real-time operations at the process points, and relieve the host computer of many of its traditional tasks. Both satellite hardware and software can be specifically configured for the particular application. For this reason, satellite computers can achieve fast and predictable response times in this simplified operating environment. Since the host is no longer burdened with the routine operations at the process points, its foreground functions consist of the supervision and coordination of the satellite facilities. Along with this reduction in work load, many of the remaining system activiites performed by the host need not occur in real time. Rather, they can be scheduled at the host's convenience. As a result, the availability of the host to execute interleaved background jobs is increased. Summarily, in the global system sense, processing operations are accomplished in parallel in real time with the net effect of enhanced performance characteristics.

Task Isolation

Each computer in the system is dedicated to a specific task group, partitioned by geography or function. Redundancies in both hardware and software characterize this distributed approach. Consequently, maximal independence is achieved at each process point, i.e., distributed intelligence makes possible a level of task isolation not attainable in a centralized system.

Task isolation contributes to ease of development, expansion, and maintenance of the facilities comprising the system. Simpler modular implementation techniques apply directly, allowing off-line development and debugging of a facility (which itself possesses full computer capabilities). In addition, potentially complex control algorithms implemented at a remote site are often independent of the host software greatly facilitating development and system integration. The maintenance advantage arises from the capability to operate a facility off-line and from commonality of hardware and software modules. Spare equipment becomes easily justifiable since the number of different hardware modules is minimized.

In some applications, high-fault tolerance proceeds from task isolation. In systems composed of more or less independent satellites, e.g., a distributed data-acquisition system, a temporary failure of the host or a satellite is not fatal in the global system sense. Unaffected system elements maintain their respective monitoring and logging processes. However, if satellites are dependent on each other for data and control information, then no improvement in fault tolerance over a centralized approach can be claimed.

Reduced System Complexity and Cost

A distributed-system approach substitutes several slower, low-cost, networked computers for the large expensive facility found in the centralized approach. In many instances, the distributed approach produces a lower cost system than a centralized approach, especially when satellite facilities can be microprocessor-based [5].

Modular implementation techniques in both hardware and software facilitate system comprehension, development, integration, and repair. Modularity also results in cheaper application software which is easier to specify and code. The bulk of application software has been removed from the complex host environment, simplifying both the application programs and the host operating system [4]. Additionally, with the reduction of necessary real-time communications, lower data rates often may be employed across the communication channels. Less expensive communication facilities can then be employed and recurring monthly expenses reduced.

Data Accuracy and Reliability

Since the loading and speed constraints of data communications has been eased, very effective error detection and control can be employed. And more complex algorithms become implementable at the process points. If lower data rates can be employed for inter-processor communications, lower error rates result [6,7]. When lowered error rates and improved error detection/correction procedures are coupled, erroneous data or control signals become acceptably rare. (See Section 3.5.2 for detailed treatment.)

At the satellite facility, local diagnostics, error detection, and automatic calibration become possible without loading either the host or the channel. Additionally, independent algorithmic sampling of each input/output port can be provided. Adaptive sampling and digital signal processing are now implementable resources at the remote site.

2.4.3 Some Deficiencies of the Distributed Approach

For the class of applications outlined in Section 2.1, the distributed approach delivers attractive advantages over a centralized approach. However, some difficulties, so far ignored, deserve mentioning.

In many distributed systems, particular satellite facilities would have to be guaranteed fast access to the resources of the host facility. Requests for service from these satellites to the host must then be honored before service requests from satellites having lesser priority. As will become evident in Chapter IV, no way exists to establish a relative priority weighting of satellite service requests to the host without sacrificing communications facility utilization. (This is in specific reference to the use of dial-up lines as the communications channel.) Hence, the assignment of unequal priorities to satellite service requests necessarily increases the recurring monthly communications expense.

Distribution of Application Control

Application programs are distributed on the basis of locality to I/O, response requirements, and dependency of other satellite processes on each other or on the central data base. The point and means for control can become ambiguous and dynamically allocated. In general, there are three possible cases of control roles between a satellite and the host: master-slave, slave-master, and master-master, respectively. Often there will exist a mix of these three environments among a host and its satellites. This is an inherent complexity: software design must allow either host or satellite to assume the control function within some priority scheme [8]. In some cases, this complexity may create a high potential for software failure in loss of synchronization and coherency. Consequently, both foresight and safeguards must be incorporated into the software and communications line control to successfully provide the control mix.

2.4.4 Notes on the Implementation

Some additional comments on the three basic system elements follow. Detailed treatment of each appears in Chapter IV.

Host Facility

Two criteria establish the location of the host computer facility: (1) convenient location of DP I/O, and

(2) minimize cost of communication lines establishing the network.

The host is a commercially available computer whose size is determined by: (1) the demands of the distributed network both the initial design intent and any conceiveable future expansion, (2) the size and number of background data processing tasks.

The resident operating system is integral to the successful implementation of the host and would most likely be a modified commercially available software product.

Satellite Computing Facilities

Historically, cost and size have limited the use of minicomputers as intelligent remote controllers. Minicomputers are too expensive to be economically attractive in applications involving primarily the execution of wait loops. Nevertheless, the satellite facility must possess sufficient intelligence, speed, flexibility, and reliability to satisfy process demands. Generally, for all but the simplest process points, one or more generalized programmable machines will be required to automate a process point. Minicomputers, programmable calculators, microprocessor-based computers (microcomputers), or programmable controllers are possible alternatives for providing intelligence at a process point. Of these, microprocessor-based computers are the least-cost, general-purpose alternative. Although characteristically slow (typical instruction time of one-to-nine microseconds) and simple, a microcomputer can adequately support a small number of peripheral functions and process loops.

Communication Channels

Reduced loading of the communication links is an important accomplishment of the distributed architecture. But communications among the comprising processors--actually between application programs--remains integral to managing the physical process. For geographically large processes (Table 1), few alternative means exist for telecommunicating data and control information. They are: (1) privately-owned lines or other communications facilities, (2) the Switched Telecommunications Network, (3) leased common-carrier lines, and (4) other specialized common-carrier services. Usually, the choice to employ common-carrier services to support inter-processor communications is economically obvious for geographically large systems. For many applications, the Switched Telecommunications Network is a practicable alternative. Three basic strengths make it attractive:

1. Virtual ubiquity
2. Economics gained through sharing facilities
3. Inherent survivability afforded by alternative routing

However, error rates incurred over switched lines are potentially intolerable unless effective error detection and correction is accomplished. These and other design considerations of a communications network utilizing common-carrier services are presented in depth in Chapter IV.

CHAPTER III

DISTRIBUTED SYSTEM ARCHITECTURAL ALTERNATIVES

The following sections develop architectural details and design considerations on the global system level. Two fundamental topologies--Star and Common-Bus-Organized--are presented and their performance characteristics argued. Multi-level configurations, which minimize design decision trade-offs by incorporating the benefits of the two fundamental configurations, are treated in the final sections.

3.1 Required Operating Characteristics

Certain operating characteristics must be realized by the computing system to successfully manage a large physical process. A summary listing of the generalized characteristics is compiled below. A suitable and practicable computing system must in general provide the following:

1. Data acquisition and peripheral control functions at remote process points according to some predetermined algorithm
2. Real-time control at the remote process points
3. Flexibility necessary to support a diversity of functions at the process point

4. Generalized data processing functions for data base management, operations statistics, and summary generation
5. Manual real-time access to system functions from a central location(s) in addition to unattended automatic operation
6. System evolution and future development
7. Accurate reliable data and control responses
8. Cost-effectiveness

Two basic distributed-system architectures will realize the above operating characteristics: star and common-bus-organized system configurations. These are the fundamental topologies with multi-level configurations of the two potentially useful. The following sections present the structure and strengths of each configuration.

3.2 Star Configuration

In this system topology, a host computer is centrally positioned and supports a centralized data base, DP I/O, and supervises system operations. Remote satellite computers each have a communication channel with the host and support sensor and external device I/O. Figure 5 schematically illustrates a star system with four satellite computing facilities.

This configuration yields the simplest distributed system; it is the easiest to develop and manage in terms of hardware, application programs, and host operating system requirements because this arrangement maximizes task isolation.

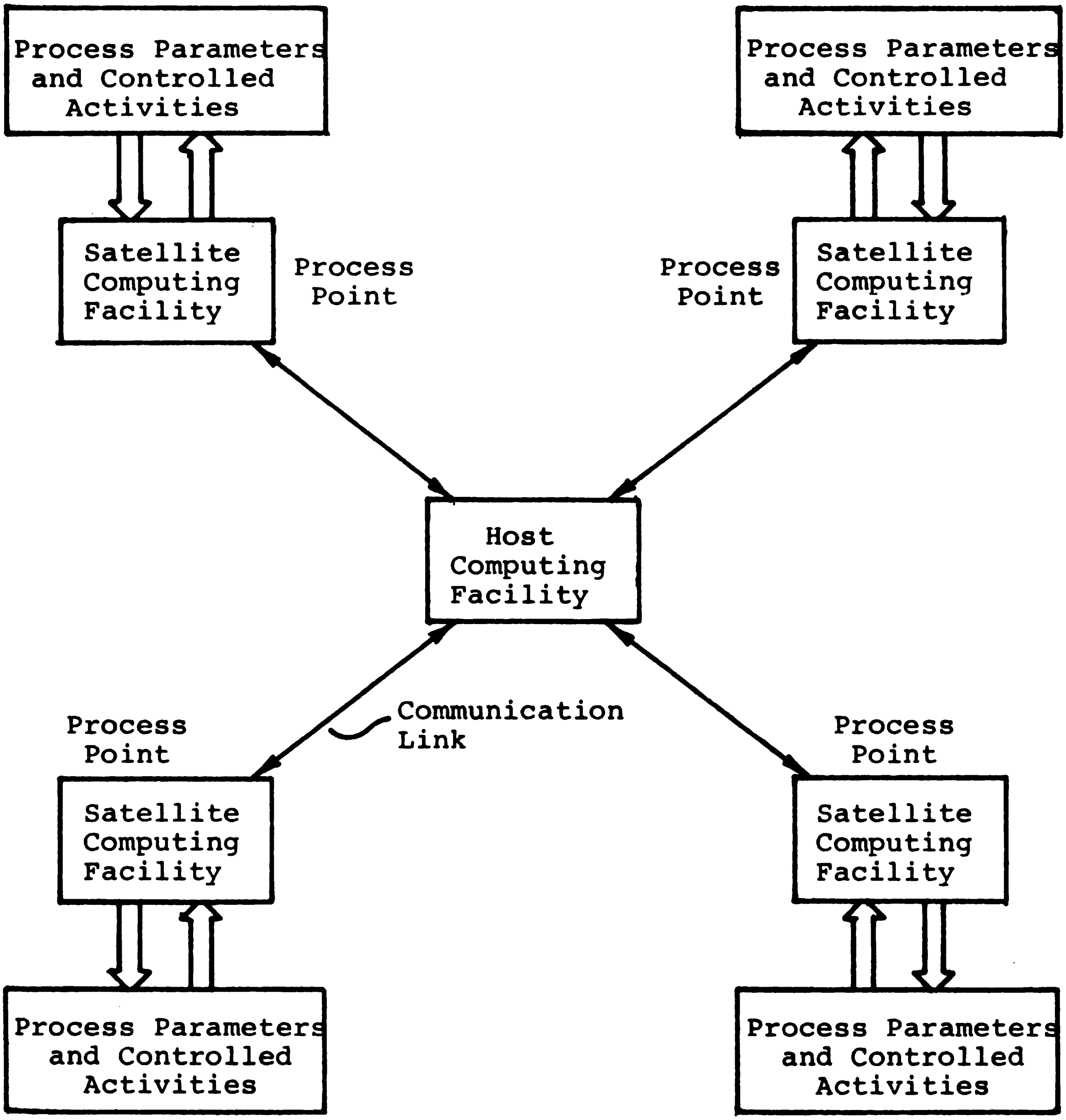


Figure 5. Distributed-Star System Configuration

As a result, the star system is readily expandable in hardware. A satellite facility can be upgraded, developed, tested, and cut-in with minimum system downtime, since implications to other satellites are inherently minimal. This independence of system processing elements can also actualize high fault tolerance. Furthermore, the star system minimizes satellite duties (minimal overhead of internal system control), and, hence, satellite facility cost. The overhead of internal system operations and protocols results only from a singular relationship with the host. Hence, most operating system modules need only reside on the host.

A significant trade-off, however, was paid to gain simplicity. In the star configuration, a satellite computer cannot communicate directly with another satellite facility. Such communications must be done via the host operating system or between respective application programs resident on the host. Because of communication delays (see Section 3.4), inter-satellite facility information exchanges can delay interdependent system responses. Furthermore, these message exchanges could burden the host in extreme cases. Communication delays must be considered especially when dial-up lines are employed. (The average telephone company connect times range from seconds to tens of seconds [9].)

3.3 Common-Bus-Organized Configuration

The common-bus-organized system is characterized by direct inter-satellite communication via a common bus which realizes global addressibility. Figure 6 depicts two perspectives of this configuration.

As in the star topology, remote-site satellite facilities independently perform all possible tasks local to that process point. The task partitioning is assigned by spatial distribution, process time-constants, and the desire to minimize loading on the bus. The host facility, as before, maintains a central data base, system executive software and operating system and DP I/O. However, the host holds no topological supremacy as in the star configuration; the arrangement is egalitarian so to speak.

The fundamental strength of the common-bus-organized configuration exists in the mechanism for fast interaction among application programs resident on separate satellites. Thus, this configuration can find utility in those processes requiring fast execution of interdependent control strategies at coupled process points. But this speed increase is obtained at the compromise of two important design considerations: expandability and implementability (in a pragmatic sense). Since application programs are interdependent, programming the system has grown into a single massive task. A high potential for software failure exists necessitating extensive executive software. Synchronization and fault

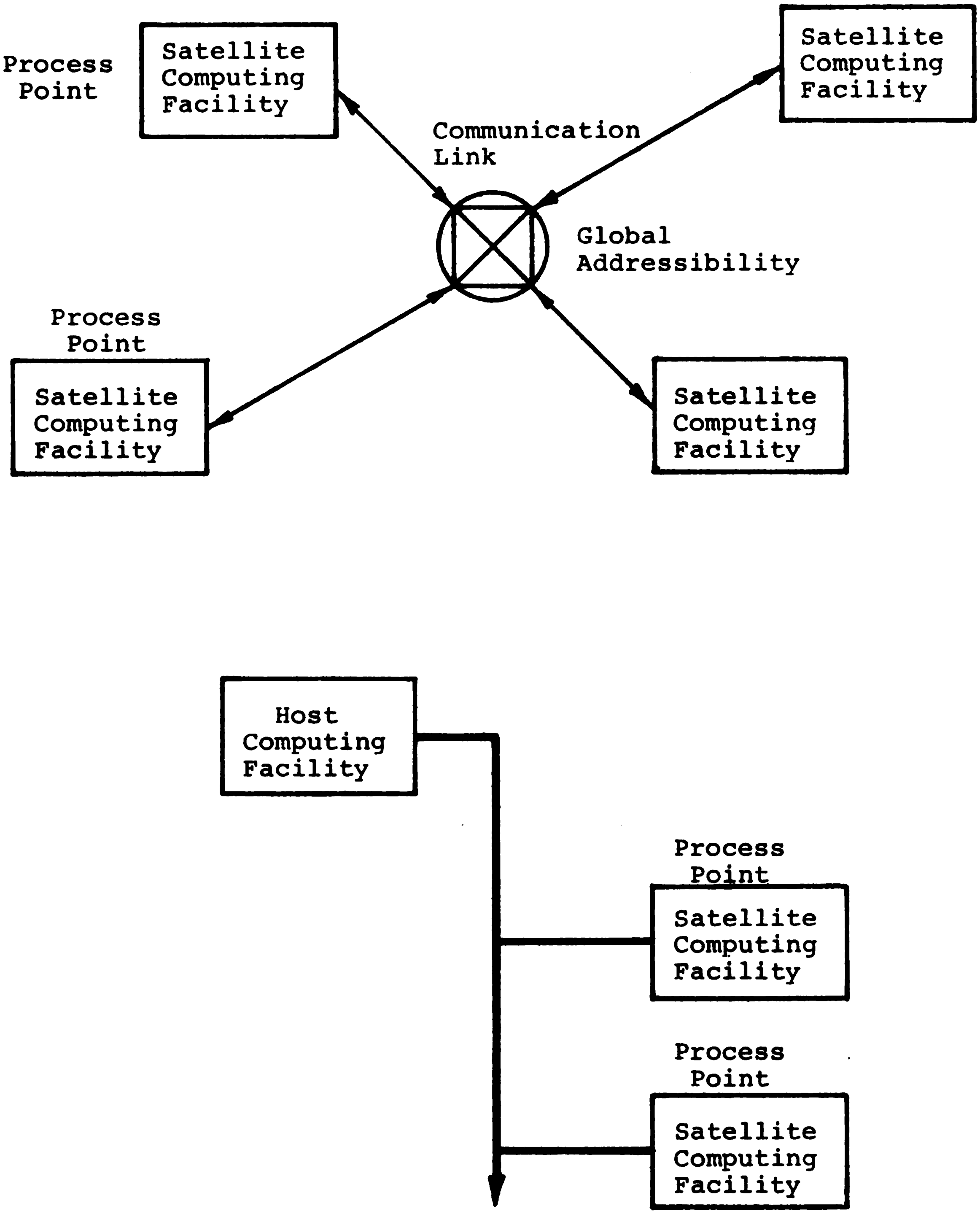


Figure 6. Two Representations of a Common-Bus-Organized Configuration

recovery must be guaranteed. Any expansion to an established system can require potentially extensive software updates to other system computers. Similarly, this is also the case for implementing changes or upgrades in the control system specification.

In summary, the common-bus-organized architecture has its place only in well-defined application areas requiring fast control system response and synchronization among coupled process points. Furthermore, the prospect of future expansion or revision must be known and provided for in the initial design.

3.4 Multi-Level Configurations

The trade-offs sacrificed with the common-bus-organized architecture are severe. Often, a multi-level configuration can provide adequate system response with less drastic compromises. Typically, in systems requiring the benefits of a common-bus approach, only a subset of process points are tightly coupled in required speed of services and interdependence. The simple star architecture cannot economically and successfully service these process points; however, the other process points can be managed by satellite facilities in a star arrangement.

Two alternative approaches exist for the fast, interacting process portions: a multi-level star or a multiple-bus configuration.

3.4.1 Multi-Level Star Configuration

A hierarchy of star arrangements is created to realize fast inter-processor communications among a subset of process points. This is depicted in Figure 7. At the hub of a subset of satellites, the satellite exchange serves as a dedicated "message switching center" and interface with the host. The satellite exchange is a specialized satellite computing facility that places highest priority on satellite communication services while supervising the sub-network operation and interpreting host commands. Note that the satellite exchange effectively isolates the dependent process points from the host and other system remote sites. Operation of the sub-network occurs more-or-less independently of the rest of the system. The group of satellite facilities attached to the satellite exchange can appear as a single satellite facility to the host. Moreover, the other satellite facilities connected directly to the host retain the simplicity of those in a simple star configuration. As a result, the multi-level star system is both expandable and implementable. However, since communications must pass through the satellite exchange, communication delays are greater in the multi-level star than in the common-bus-organized system.

3.4.2 Multiple-Bus Configuration

By employing more than one bus to support inter-processor communications, the compromise on expandability and implementability is somewhat lessened. Figure 8 illustrates groupings

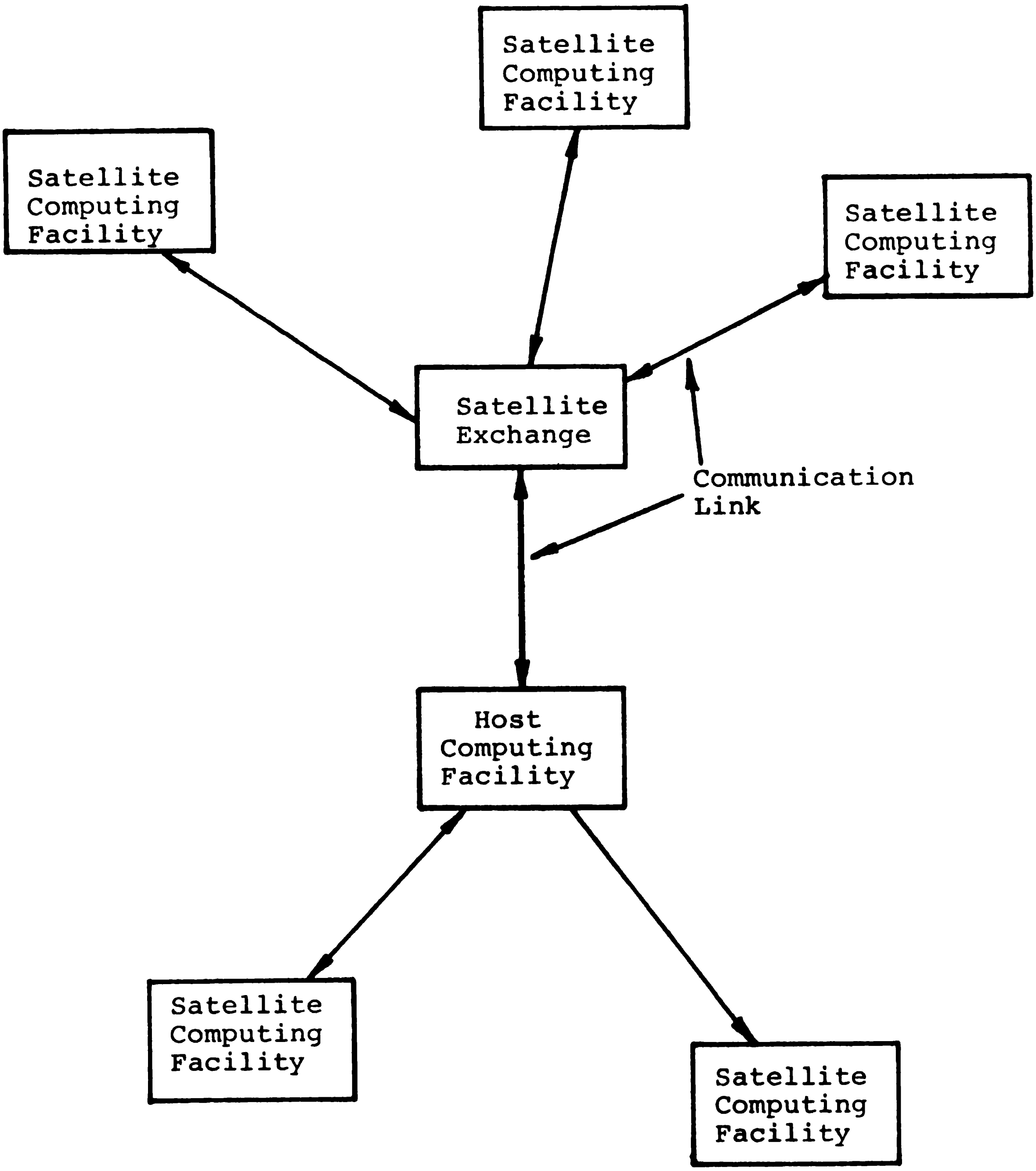


Figure 7. Representation of a Multi-Level Star Configuration

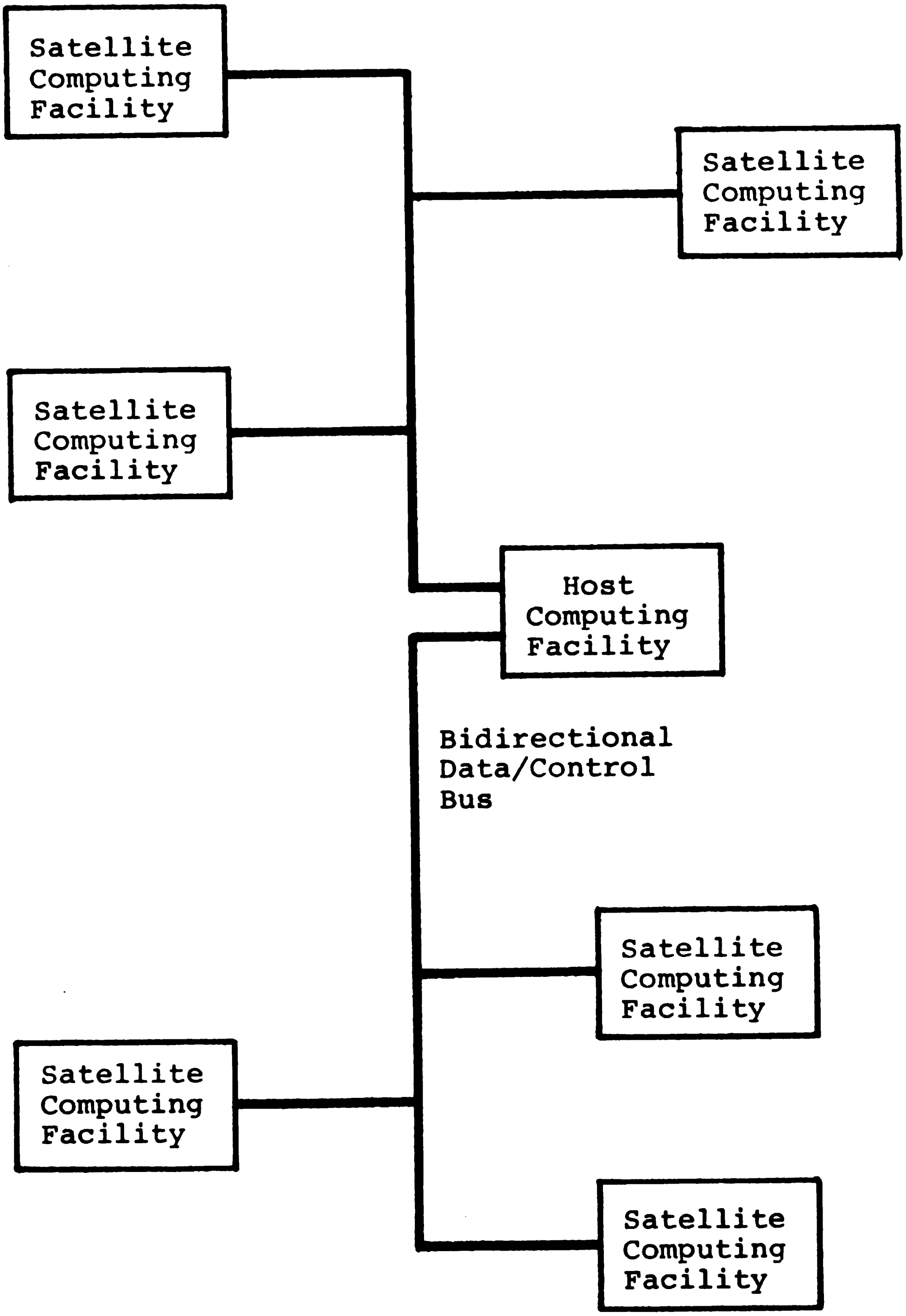


Figure 8. Representation of a Multiple-Bus Configuration

of remote satellites sharing a bus which is common to the group. Satellites are assigned to a bus according to the interdependence and speed requirements of their control algorithms--the objective being the isolation of a portion of the physical process. Each bus then delineates a tightly coupled subsystem with maximal independence from other subsystems connected to the host. Herein lies this configuration's strength; namely, the potential for high-speed processor interactions in an expandable, modular framework. However, the software design task, especially at the host, remains formidable.

3.5 Summary

The common-bus-organized configuration supplies the high-speed inter-processor communications needed in some applications. However, since it is difficult to implement and expand, its attractiveness is limited to a small class of applications. In the majority of cases, a star or multi-level star configuration applies quite satisfactorily, so long as the delays imposed by information transmission can be tolerated.

Details of the comprising computing facilities and communications links have been purposely omitted in these discussions of the system architectures. Comprehensive specification of each would have obscured the topics presented. Chapter IV now details each system element individually. Primarily for the sake of simplicity, a star system configuration is generally assumed in the discussions comprising the remaining sections of the thesis. Most of the material will apply equally well to the alternative configurations; however a star-system point of view is employed.

CHAPTER IV

DESIGN CONSIDERATIONS OF THE HOST, SATELLITE, AND CHANNEL FACILITIES

Specifications and design considerations of the individual system elements are developed in the following sections. First, the host computing facility operating system requirements are examined. Then, satellite computing facility hardware is investigated with particular attention given to microprocessor-based facilities. The final sections deal with practical communication-network design considerations and the network design process. Throughout this chapter, discussions generally focus on the star system architecture, and primary emphasis is placed on the use of the Switched Telecommunications Network.

4.1 The Host Facility and Operating System Functions

Just as in most real-time, multi-task operating environments, the host facility is a disk-based, general-purpose computing system supporting an appropriate amount of main memory and mass storage, DP and interactive I/O, and interface equipment for the telecommunications network. In a star configuration, the generalized elemental duties of the host facility include:

1. Perform system executive functions, i.e., synchronize, coordinate, and supervise remote facility activities; supervise execution of application programs
2. Support global communications over the network
3. Manage data, i.e., maintain a data base and provide the application programs access to the data base
4. Process data
5. Provide access to system operation, i.e., support real-time interactive inquiry and control commands, and document system operation
6. Support human interaction with system operation and control

4.1.1 An Operating System Alternative

In many instances, a commercial software product(s) can be found which provides many of the services listed above. This software package can be modified and supplemented to operate within the environment of a specific application. The majority of modifications will concern the insertion of a Telecommunications Access Package (TAP). Figure 9 depicts the working relationships of the TAP in the overall operating system. TAP provides the faculties for using the telecommunications network, i.e., for host and satellite interactions. It isolates the application programmer from the complexities and pitfalls of the network.

Basic TAP Functions

The listing below outlines the scope of faculties requisite in a TAP. In some cases, specific functions may

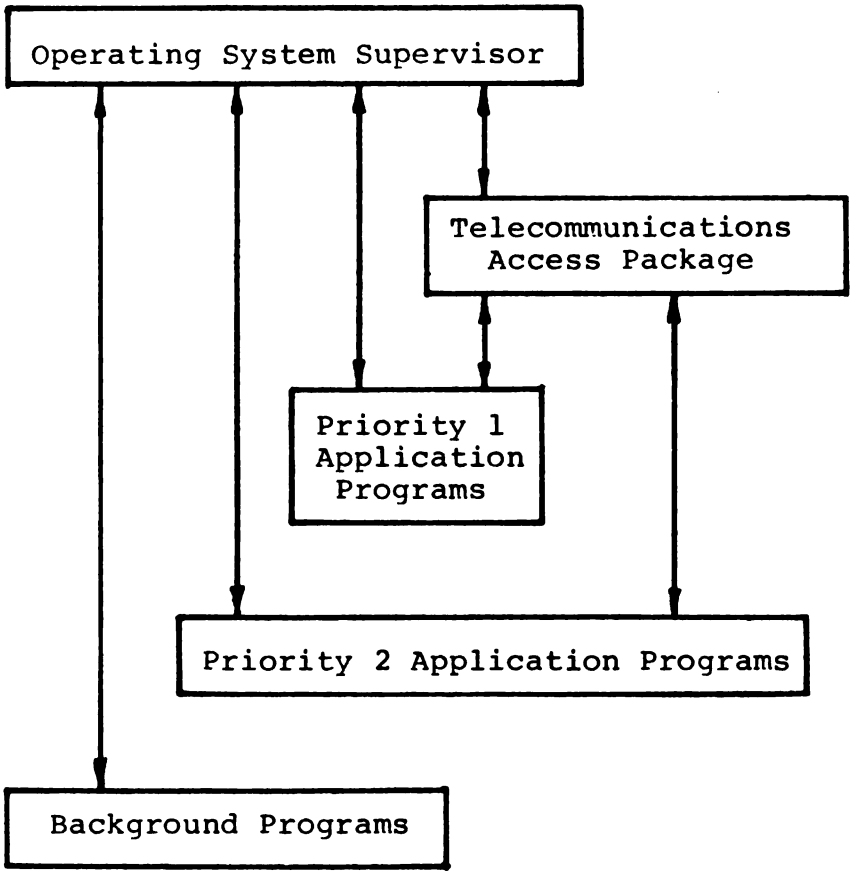


Figure 9. Gross Operating System Structure

be unnecessary, but in general the TAP must include routines that:

1. Address (or automatically dial) satellite facilities over the communication network and establish a circuit for data communications
2. Acknowledge (answering) service requests for host service
3. Initialize operations and execute protocols preceding a data exchange
4. Terminate transactions
5. Transmit and read messages
6. Transfer messages automatically between satellite facilities
7. Handle transmission errors by providing error detection, automatic request for retransmission, automatic retransmission upon request
8. Generate transmission error statistics
9. Test and diagnose telecommunications equipment to isolate equipment failures and validate equipment integrity

Clearly, the majority of the telecommunications package must also reside on the satellite facilities as well. If both the host and satellites employ a common instruction set, system programming chores will be greatly simplified.

4.2 Satellite Facility Specifications and Architecture

Located at a process point, a satellite facility functions as the local data acquisition system, data processor, data logger, and peripheral device controller. Because of task partitioning, the computing speed and capacity required at a remote site can usually be furnished by a microprocessor-based computer(s). Herein lies the economic advantage of the distributed-intelligence approach: the ability to exploit the economies of LSI.

4.2.1 Satellite Computer Hardware

Computer hardware at a satellite facility would typically incorporate the following elements:

1. A general purpose processor that is interruptable
2. Non-volatile storage for the application program
3. A read/write data buffer and work space
4. A RS-232-C interface [10] to connect a modem or other applicable data communication equipment
5. A RS-366 interface [11] to connect an automatic calling unit (ACU)
6. Serial and parallel I/O interfaces that are configurable under program control and employed to connect local equipment, e.g., process monitoring and control equipment
7. A local control console, display, and terminal and the associated interface
8. A priority interrupt controller to support multiple interrupt sources in a priority structure configurable under program control
9. A power fail/restart interrupt generator
10. Interval timers for real-time clock, control delays, and operation monitor (that restarts computer operation automatically when abnormal operation occurs)

Note that items six and eight are specified to be configurable by the application program. With this approach, these interfaces can serve in a range of environments, and, thus, they can be employed in a variety of satellite facilities.

4.2.2 Hardware Organization

The internal structure of a generalized microprocessor-based satellite computer is presented in Figure 10. It is a bus-organized, modular machine with a simple interrupt-priority system. This architecture and the execution of the application program stored in eraseable-programmable read only memory (EPROM) completely determine the operation of the satellite computer. The microprocessor (μP) executes the application program performing all local processing tasks and controlling all bus-connected components. All data and control information transfer between the processor and other computer components via the internal I/O bus. Each bus-connected element shown in Figure 10 is described in the following sections. The function and conceptual design of each function block is explained and illustrated. (Translation of these function blocks into actual computer hardware is not pursued. Recurrent advances in LSI technology and available function blocks dictate that actual circuit hardware be specified at the time of implementation.)

4.2.3 Memory Requirements

The memory requirements of a satellite computer will vary directly with the complexity of the control algorithm, the number of integrated sensors and peripheral devices, and the time interval between data dumps to the host facility.

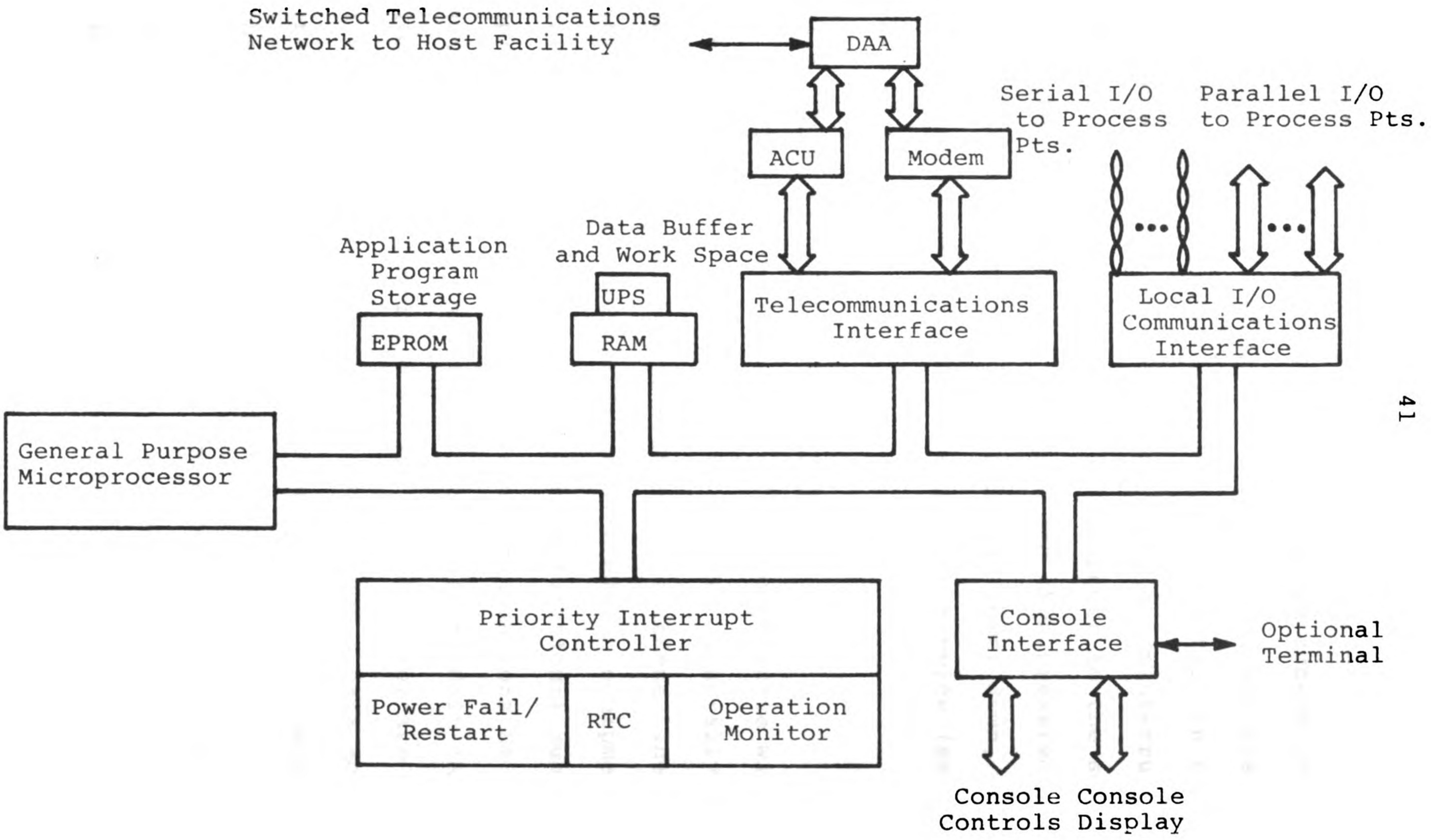


Figure 10. Block Diagram of the Satellite Hardware Organization

A typical application program would occupy about 4K words of EPROM ($K=2^{10}=1024$). Read/write random access memory (RAM) serves primarily as temporary data storage, and its contents are periodically dumped to the host facility. In the event of a power failure at the remote site, an uninterruptable power source (UPS) provides stand-by power to the semiconductor RAM. Data buffer contents are then preserved, and machine status which was stored during power down, can be restored on resumption of power-utility service (see Section 4.2.7).

4.2.4 Telecommunications Interface

Satellite facilities will often communicate with the host facility over common-carrier lines. Generally, to perform digital data communications over these lines, a data modem (data transmitting and receiving equipment that performs modulating and demodulating functions) must be employed. When the Switched Telecommunications Network is used, both a modem and an Automatic Calling Unit (ACU: automatic dialing equipment) are needed. They are connected to the telecommunications network through a Data Access Arrangement (DAA) provided by the telephone company or some other certified protective interface.

The interconnection of the modem and ACU with the data terminal equipment (satellite computer in this case) is standardized and specified by Electronic Industries Association

(EIA) Standards RS-232-C and RS-366 respectively [10,11]. The telecommunications interface handles data, control, and status signals to the modem and ACU according to these standards and performs conversions for bit-serial data transmission. Figure 11 shows the integral function blocks and signal paths needed to produce the RS-232-C and RS-366 interfaces. Both are interrupt serviced since significant time intervals elapse between interface actions (e.g., digit requests from the ACU can occur at up to one second intervals). This approach allows the processor to perform useful activities during the pauses associated with the telecommunication operations.

As illustrated, the port select logic recognizes bus addresses and gates control signals to the addressed function. A Universal Synchronous/Asynchronous Receiver/Transmitter (USART), an LSI function block, derives the RS-232-C signals and performs serial/parallel data conversions. Programmable USARTs are available which operate with virtually any serial data-transmission technique presently in use. The μP can read the status of the USART at any time including data transmission error flags, RS-232-C status lines, and internal status. Data rate is set by a selectable timing chain. The RS-366 signals are handled by a six-bit register and combinational control logic. In the final stages of the interface, signals to/from the modem and ACU are shifted from the logic levels used internally to the EIA standards levels.

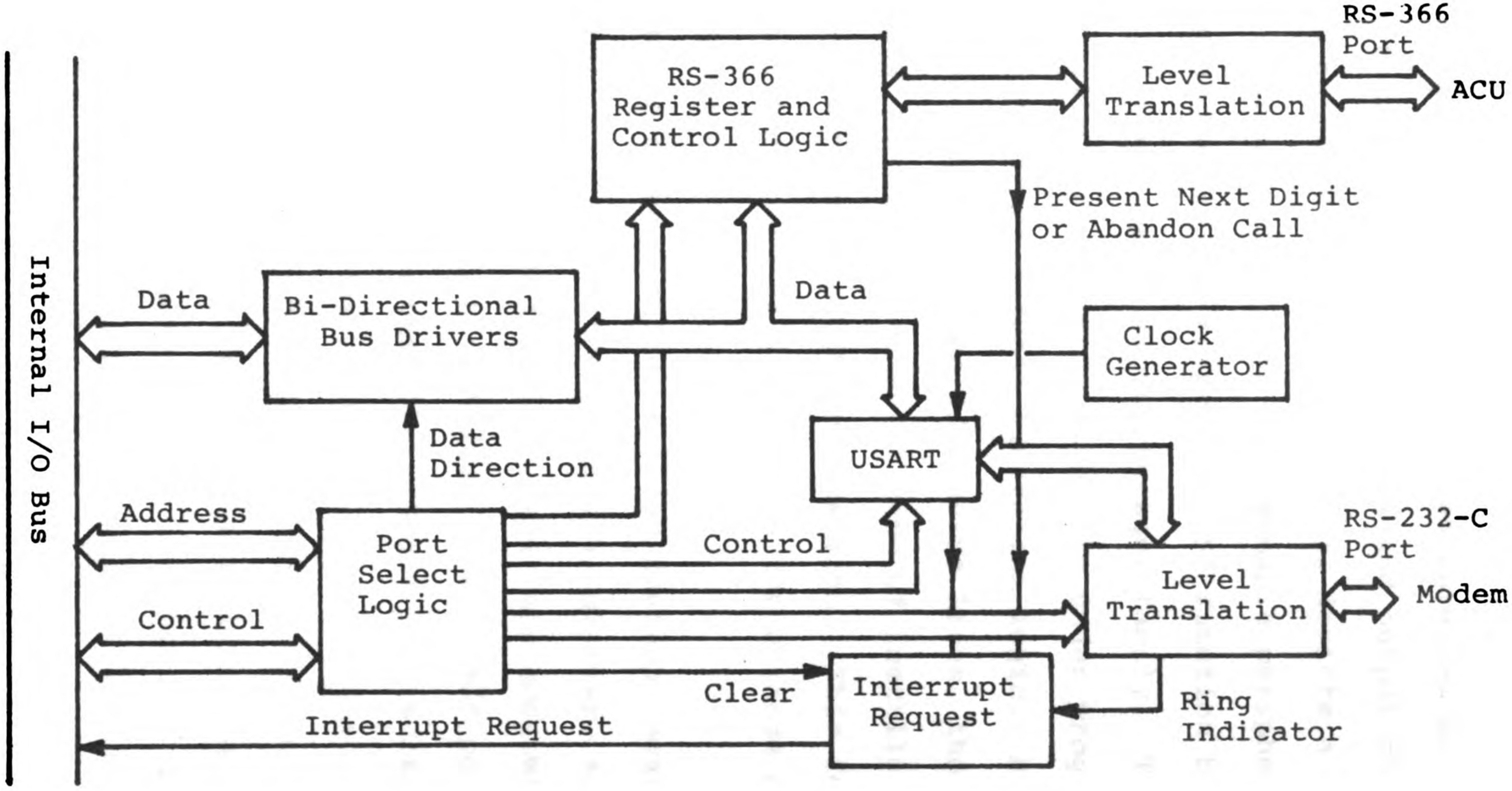


Figure 11. Block Diagram of the Telecommunications Interface

4.2.5 Local I/O Interface

The local I/O interface provides parallel and serial I/O ports to input local sensor data and output control signals to external devices. Figure 12 depicts a practical interface configuration. The programmable peripheral interfaces shown are the respective LSI function blocks available in every major microprocessor family. These chips produce parallel I/O ports configurable under program control and contain interrupt request generation logic. By providing bidirectional drivers on the I/O device lines, the direction of data flow can be determined at time of installation by both program commands and jumper position. This feature enables the common use of this module for a range of I/O port configurations.

For those sensors and controlled devices located more than several meters from the satellite computer, serial communication over a current loop is a very economical and noise-immune transmission technique. The USART performs the serial/parallel conversions; the current-loop multiplexer (MUX) derives, translates, isolates, and selects the current-loop signals. Current-loop data rates are limited by transmission-line effects to 4800 bits-per-second (bps) for lines shorter than 300 meters, decreasing to 600 bps for lines of 3,000 meters (the practical limit of line length using this technique).

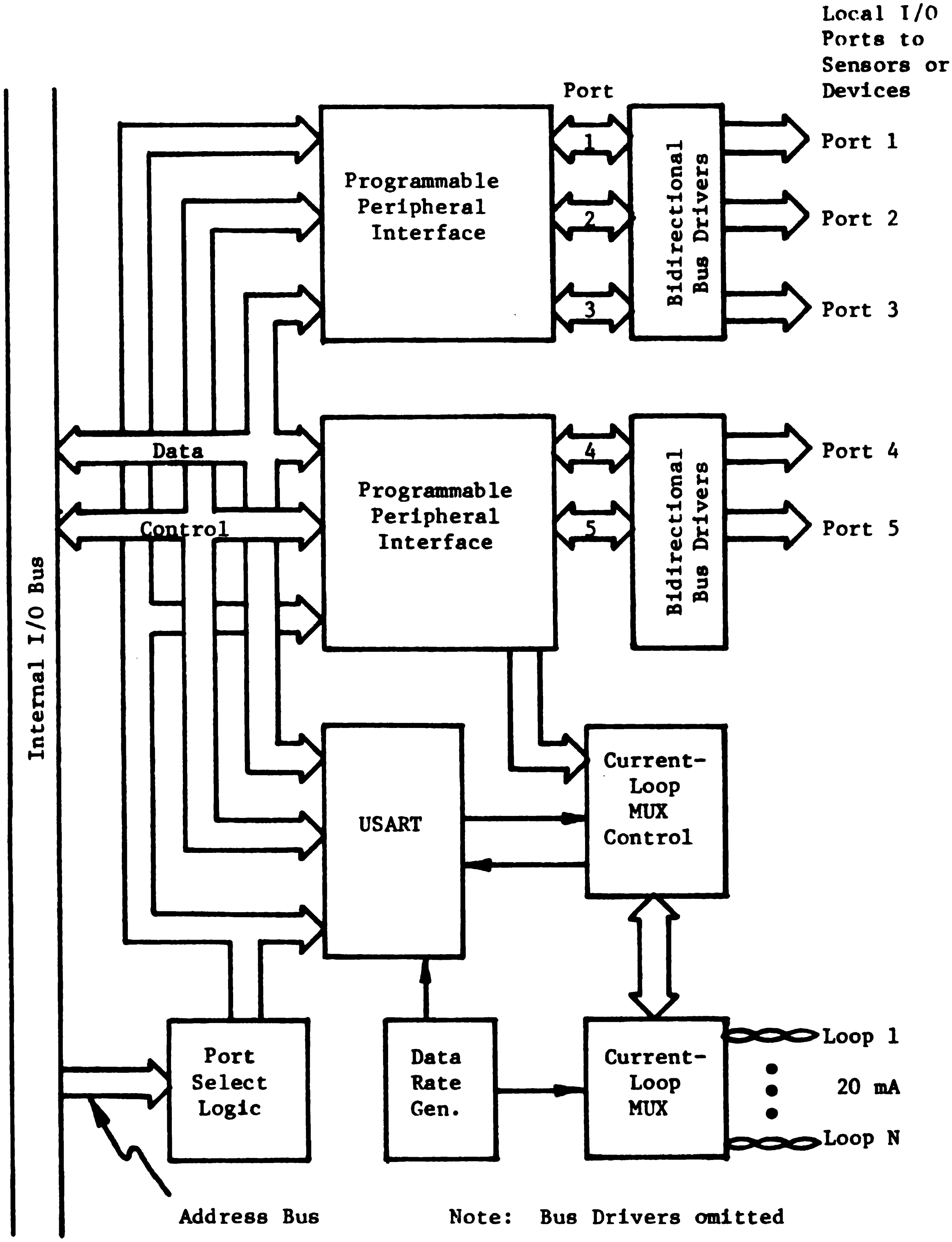


Figure 12. Block Diagram of the Local I/O Interface

4.2.6 Console Interface

Access to the internal operation of the microcomputer must be provided locally at the remote site. The computer console and an optional terminal furnish this access through the console interface. Console operations include: halt/resume application program execution, restart, current status display, and the execution of routine maintenance commands by an operator. Examples of such commands are: resetting an alarm flag after operator correction, checking current equipment status, or simple diagnostic self-checking procedures. Presented in Figure 13, the console interface is composed of function blocks previously described. Note that another USART occurs here, redundant with the local I/O interface. This hardware redundancy finds justification in (1) greater software convenience and (2) data rate flexibility (the local I/O current loops are not constrained to the 110-300 bps bit rate required by most terminals).

Remote station operation under the control of a local terminal is required in the event of an extended failure at the host facility; data buffer contents could be dumped to the terminal. But even more importantly, the terminal's greatest utility is as a development, diagnostic, and maintenance tool. An appropriate design intent could be to provide for the terminal in hardware and software but connecting one only as required.

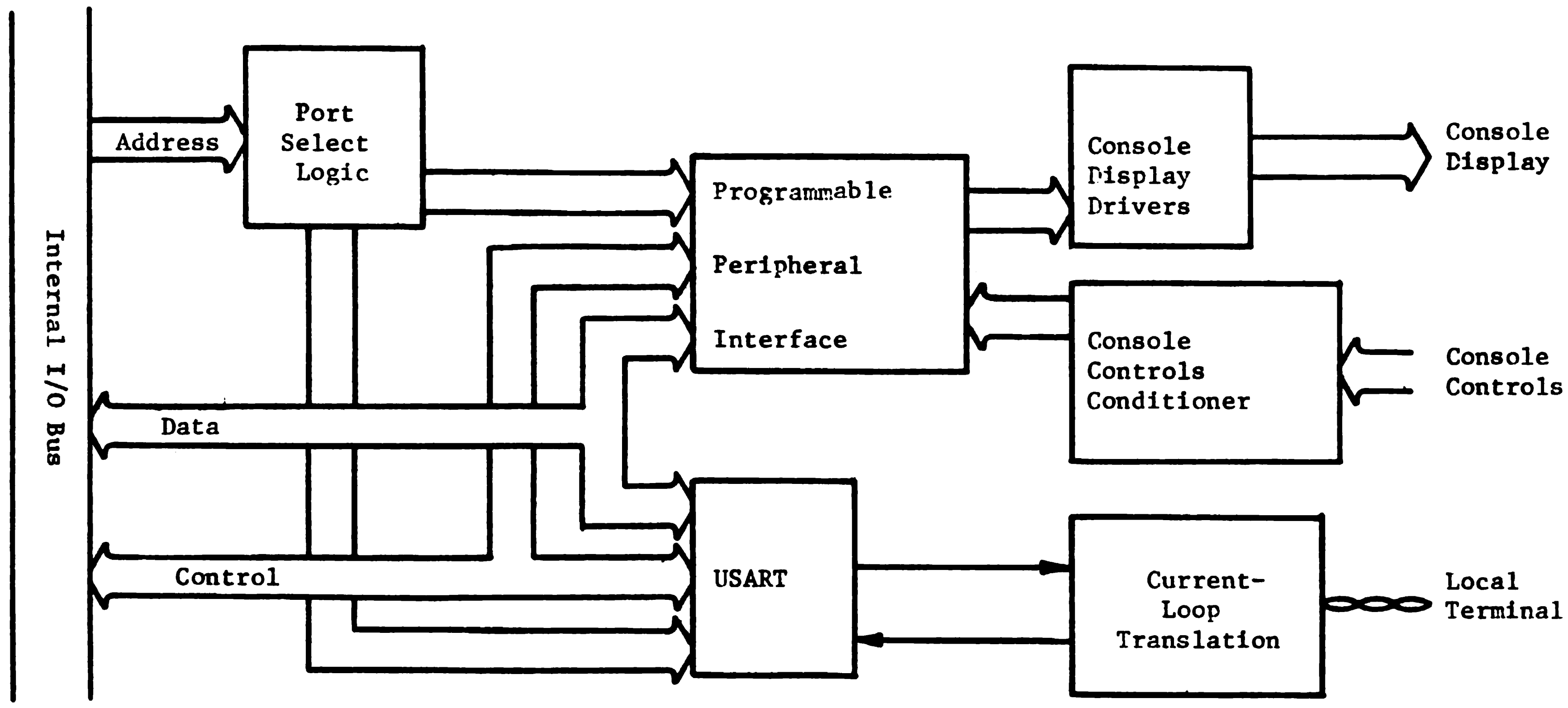


Figure 13. Block Diagram Illustrating the Console Interface

4.2.7 Priority Interrupt Controller

Interrupt capability is a standard feature on general purpose μPs. Placing I/O functions under interrupt control rather than polled service enhances real-time performance and simplifies the application software when independent software modules can handle individual tasks. The following functions, listed roughly in order of decreasing priority, should be interrupt serviced:

1. Power fail/restart
2. Operation monitor
3. Telecommunications
4. Local I/O
5. Real-time clock
6. Programmed control delays
7. Console commands

The block diagram of a priority interrupt controller which provides service to these functions appears in Figure 14. Two LSI function blocks comprise the controller: a programmable interval timer which possesses at least three independent programmable down-counters, and a priority interrupt controller. Both blocks are currently available in most major microprocessor families. The power-fail detect circuit interprets a signal line from the power supply that forewarns a drop in the logic voltage. When AC power fails, an interrupt request immediately ensues that is given highest service priority. Execution of a power-down routine then saves machine status and flags the power outage to the restart routine. On power resumption, execution of a restart routine is forced by an immediate

(1) Real-time Clock
(2) Control Delay Timer
(3) Operations Monitor

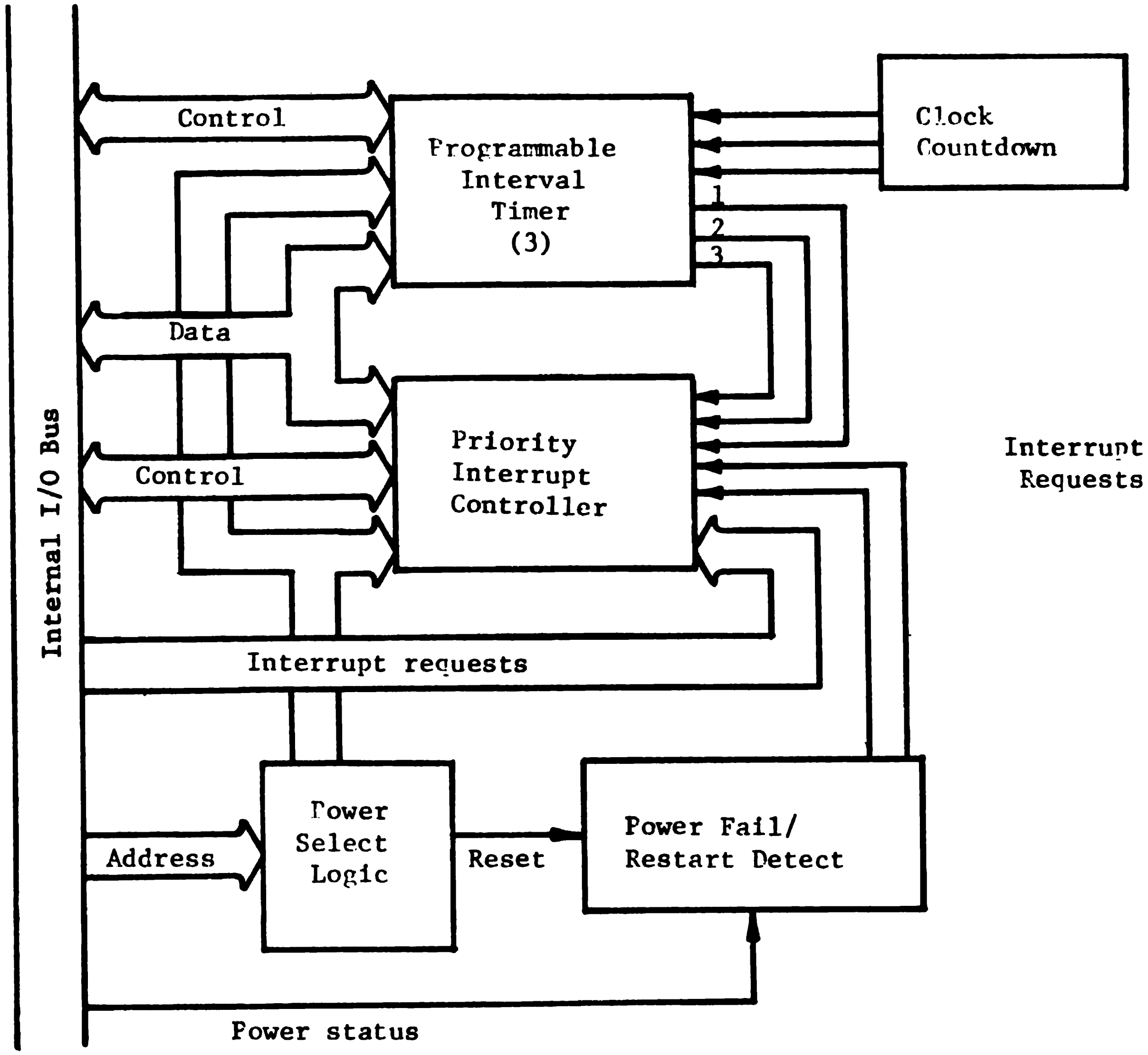


Figure 14. Block Diagram of the Priority Interrupt Controller

interrupt request which restores the former machine status and resumes orderly operation.

One of the programmed timers serves as an operation monitor--a "watchdog" of sorts. The timer is set to the longest acceptable scan interval by a portion of regularly executed code. If a scan cycle ever extends beyond the expected worst case, the operation monitor timer generates an interrupt request which indicates a malfunction. Such an occurrence could result from an external device failure, lightning, or other acts of God. With this technique, facility failures can be easily detected and flagged to the host. A degraded mode of operation can be maintained rather than catastrophic facility failure.

4.3 The Communication Channel and Network

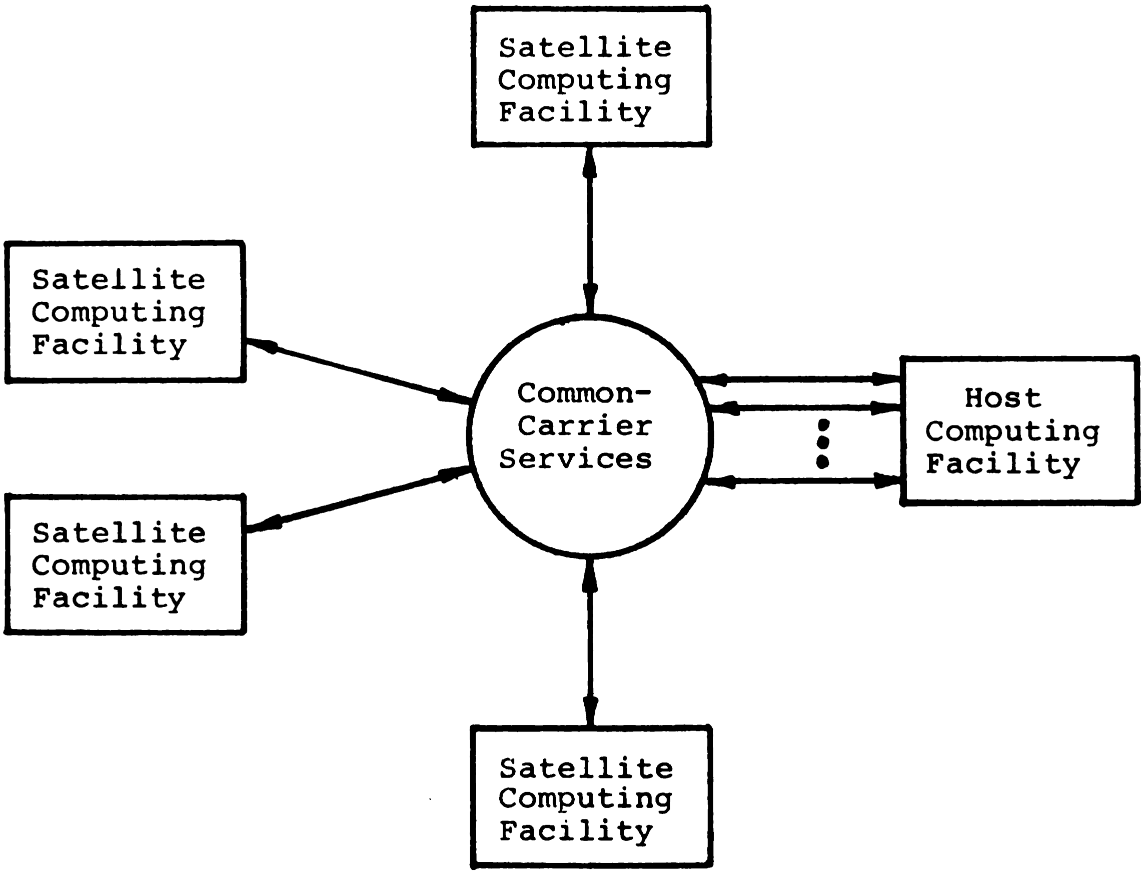
Distributed architectures provide the potential for excellent real-time system behavior. Performance, however, must not be restricted by limitations on interprocessor communications. Careful consideration must be exercised in the communication network design since it is often the system bottleneck, the primary source of errors, and a recurring monthly operating expense. Hence, the network design objectives reduce to the classic optimization problem: achieve adequate performance at a minimized expense. Often, either all or a major portion of the large automated control system can tolerate communication delays as a result of task isolation. In view of geographic scale and this assumption, the Switched Telecommunications Network becomes a workable and attractive basis for a practical network implementation in light of: (1) the universal availability of connections, (2) reliability from alternate routing, and (3) the lack of an economical alternative.

In these sections, an interprocessor communications plan will be proposed along with a formalized quantitative design process. For the remainder of the communications discussion, a star system configuration is generally assumed in which dial-up lines connect the satellite facilities to the host facility. In some cases, leased lines must be specified as discussed later.

In Figure 15, common-carrier services are incorporated into the star system diagram. Several design decisions must be made to specify the network and channels: How many lines are required at the host? When should dial-up or private-leased lines be used? What data rate should be used? How long should messages be? And what about error control? All of these areas are discussed in detail in the remainder of this chapter.

4.3.1 Error Rates and Anomalies of the Switched Telecommunications Network

No modulation technique can be designed to deliver error-free data in spite of the variety of phenomena common to telephone channels. Consequently, transmission errors are inevitable. Telephone channel performance is well



Design Decisions:

- -- Number of Lines to the Host (M)
- -- Data Rate or Line Speed (S)
- -- Optimal Message Block Length (N_M)

Figure 15. Common-Carrier Services Within a Distributed System

documented--the 1969-70 Connection Surveys conducted by Bell Laboratories [6, 7] extensively studied error rates for low- and high-speed bit rates. For low-speed, asynchronous character transmission at 300 bps, the study observed a lost character rate (characters not recovered by the receiving modem) of 13.7×10^{-4} and an incorrect character rate of 1.07×10^{-4}, for an average overall character error rate of 14.77×10^{-4}. This rate represents about 2.2 missing or wrong characters per page of this text! It must be noted, however, that 90% of the calls in that study had no lost characters; a few bad calls dominated in lost characters. In addition, 50% of the calls were error free.

The high-speed voice-band data transmission study concentrated on 1200 bps and 2000 bps. Error rates at 1200 and 2000 bps were observed to be comparable to each other, while error rates at 3600 bps were observed to be somewhat larger. Overall, for the 1200 and 2000 bps calls, about 82% of the calls had error rates of 10^{-5} or better, although 3.5% of the connections had rates poorer than 10^{-4}. This study was also dominated by a few error prone calls: 72% of the observed errors occurred during 5% of the calls. Table 3 provides a rudimentary summary.

Table 3. Bell Connection Survey Summary [6, 7]

Overall Average Bit Error Rates:

2000 bps: 1.9×10^{-5}
1200 bps: 6.6×10^{-5}
300 bps: 1.5×10^{-5}

Percent of Calls with Error Rate Better Than 10^{-4}

2000 bps; 88%
1200 bps: 84%
300 bps: 78%

Another class of telephone channel anomalies were summarized in a Bell study published in 1964 [12]. This study focused on synchronous data transmission at 2000 bps. Of the 548 calls attempted in the study, 10.7% (59 calls) could not be completed. The breakdown is as follows:

1. Long dropouts or fades resulting in loss of synchronism between data sets: 5.5% of attempted calls
2. Inability of data sets to achieve initial synchronism within a reasonable length of time (2-3 minutes): 3.8% of attempted calls
3. Lost connections, i.e., the dial tone returns in middle of a call[1]: 1.5% of calls

Several conclusions can now be stated concerning the requirements and capabilities of a viable error control plan. Error rates vary greatly from call to call, so the error

---

[1] Lost connections were determined to be associated with telephone company maintenance operations.

control employed must accomodate high, raw error-rate conditions. Errors also tend to occur in clusters--the result of burst noise, dropouts, and longer interferences. Long concatenations of errors must be detected. Furthermore, unsuccessful or interrupted connections need to be recognized, terminated, and re-dialed to obtain a different connection.

4.3.2 Error Control and Its Effectiveness

The telephone channel then is a formidable communications environment; nevertheless, accurate reliable operation can be achieved. Reliable detection of errors is the key. Error detection-block retransmission is a simple and very effective system of error handling. Messages to be transmitted are encoded to contain redundant information that can be checked by the receiver to discover transmission errors. Upon detection of an error, the receiver requests retransmission of the block containing the error. (See Section 4.3.5.)

The customary parity check bits are an example of the general encoding technique. However, an AT&T study conducted in 1960 shows that the single parity bit check will fail to detect about 30% of the character errors caused by a telephone channel [13, p. 79; 14]. However, polynomial codes[1] are

---

[1] Martin in reference 13, pp. 81-95, gives a practical condensed treatment of polynomial checking.

very effective in detecting telephone channel transmission errors. This is well documented in another Bell System study [12] using a Bose-Chaudhuri-Hocquenghem (BCH) code. The study employed a polynomial code with a word length of 31 bits of which 21 are information (31, 21) transmitted at 2000 bps over the switched network. An undetected bit error rate of 10^{-9} was observed. That is one expected undetected error in about 10,000 transmissions of the entire text of this thesis. Thus, from these arguments it can be concluded that accurate information transmission is possible over the Switched Telecommunications Network.

4.3.3 The Dial-Up Lines vs. Leased Lines Decision

Selection between switched or leased lines is based on three considerations: (1) required speed of service, (2) monthly cost, and (3) data rates and attendant error rates.

Speed of Service

A significant inescapable delay exists between the time when an originating facility asserts off-hook (a line to the DAA used to request a dial tone) and the start of called party ringing. Figure 16 diagrammatically presents the components comprising this service delay [9, p. 12; 15, pp. 712-15]. Dial tone delay is a function of the type of the local office switching equipment and current traffic load conditions. For light traffic conditions the delay is

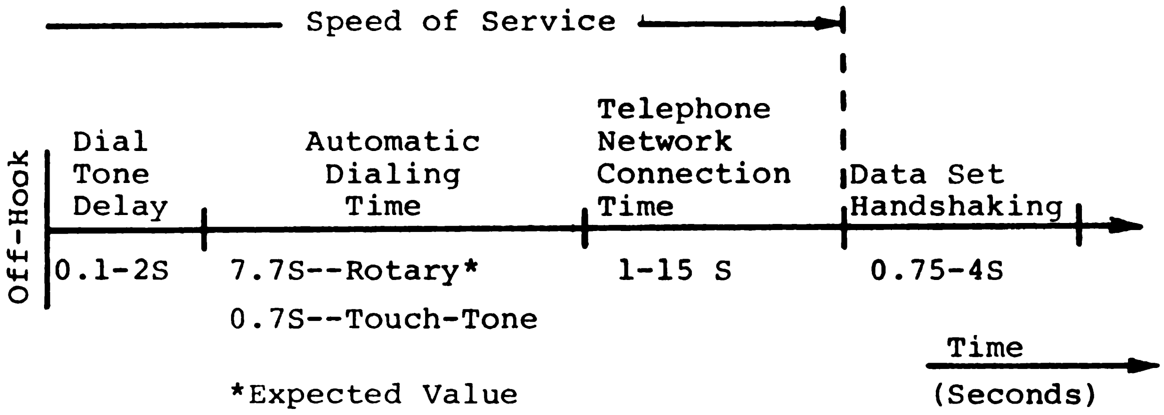


Figure 16. The Components of Switched-Line Service Delays

typically from 100-500 msec. The type of signaling employed sets the dialing time along with the actual number dialed. Rotary dial pulse signaling occurs at 10 ± 0.5 pulses per second, where N pulses required for digit N. Likewise, the Touch-Tone signaling rate can occur up to 10 digits per second. Telephone network connect time can vary from one second to about 20 seconds, and is primarily dependent on the number of network links in tandem for that chance routing. And data set handshaking time is specific to the particular modems in use.

In the case of leased lines, the time delay between the request for transmission and the start of information exchange is determined by the communication hardware arrangements at the computers. Clearly, if the leased lines are interrupt serviced, the communication delay would consist of the data set handshaking time. In any case, service delays for leased lines are far shorter than for switched connections.

Monthly Cost

Within a local dialing area, no broad generalization concerning line costs can be made because each area has a unique definition of communication services and rates (e.g., flat monthly rate or message unit ratings). But usually, dial-up lines will be cheaper except for an application requiring frequent calls in a dialing area that employs message unit ratings for local calling revenue [16]. For leased lines, rentals can range from a flat $7/month to typically $4/month/mile [17].

For channels extending beyond the host local dialing area, the cost of a leased line can be more or less than for the equivalent usage of the Direct Distance Dial Network. It depends on the length of the line and on the usage pattern. If line usage is high--greater than about three hours per day for example--then the leased-line tariff will usually be less expensive. Also, a usage pattern of 25-50 calls per day lasting less than three minutes each will have lower costs over a leased line. (References 15 and 18 give extensive treatment.)

Data Rates and Error Rates

Because switching equipment is not involved with leased lines, the composition of the line is fixed, and these facilities are less prone to the chief source of errors on switched connections--impulse noise originating in switching offices. (No quantitative measurements on the magnitude of this advantage appear to be available [18, Lucky, p. 144].)

The practical data rate limit over dial-up lines is currently 3600-4800 bps. Conditioned leased lines can support data rates up to 10,800 bps. But for the nature of communications involved in the applications listed in Chapter II, the higher data rate capacity of leased lines is not requisite. For example, at 2400 bps, the time required to exchange 1,024 eight-bit characters is 2.8 seconds (an approximated effective line speed used in the calculation, see Section 4.3.5). Thus, in the majority of these cases, dial-up lines will satisfy the data rate needs of large distributed systems.

4.3.4 An Algorithm for Specifying the Dial-Up Communication Network

A practicable iterative design process for designing the interprocessor communication system composed of dial-up lines appears in Figure 17. This algorithm applies directly to the fundamental network configuration that is depicted in Figure 15. Here, each satellite possesses one line for communications with the host facility, and the host maintains several lines, equally loaded, and with equal priority status. This is the simplest case and serves as the starting point for the analysis of network refinements required by specific applications. For example, a mechanism for organizing the queue of outstanding satellite service requests might be incorporated to allocate host service by priority assignments.

It is presumed at this point that an approximation of the protocols and codes to be used is known along with an acceptable range for service delays. The design process starts with a good guess of the line speed that the system requires, and using performance data of various available modems one can proceed through the process. Necessary

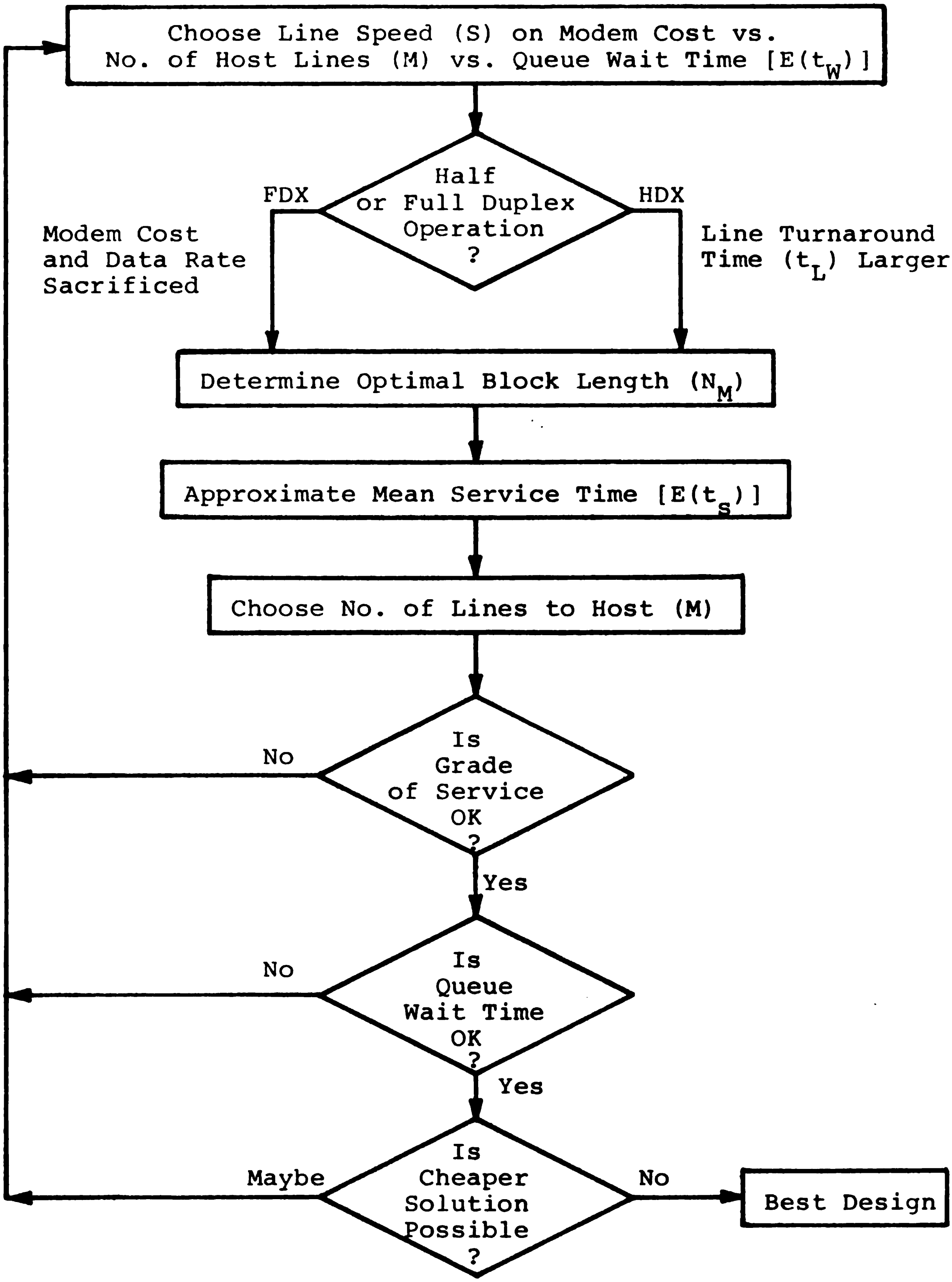


Figure 17. Elementary Communication Network Design Algorithm

design relations are presented in the following sections; the equations are referenced or derived in the Appendix. By employing these relations and iterating, the algorithm converges on the best cost versus performance compromise.

4.3.5 Optimal Block Length

In error detection-retransmission error control, the transmitter sends a block of N_M message characters along with the necessary control characters numbering N_C. The transmitter then stops and waits for acknowledgement from the receiver via some reverse channel. Some line turn-around time, t_L, is associated with the reversal in the direction of transmission. If the block is positively acknowledged, no transmission errors occurred and the next block can be sent. Otherwise, the prior block is retransmitted. Note that two line turn-arounds take place for each block transmission. With this method of error control, the effective line speed, S_E characters per second (cps), is less than the actual line speed, S cps, due to both the required retransmissions and line turn-around delays. In the Appendix, the effective line speed is shown to be:

$$S_E = S \left[\frac{(1-P_E)^{C(N_M+N_C)}}{\frac{N_M+N_C}{N_M} + \frac{2St_L}{N_M}} \right] \qquad (4.1)$$

where P_E = bit error rate over switched telephone network at rate SC bps

C = number of bits per character

ra

av

18

de

wh

In

is

T

1

w

C

T

T

4

c

F

h

c

F

e

r

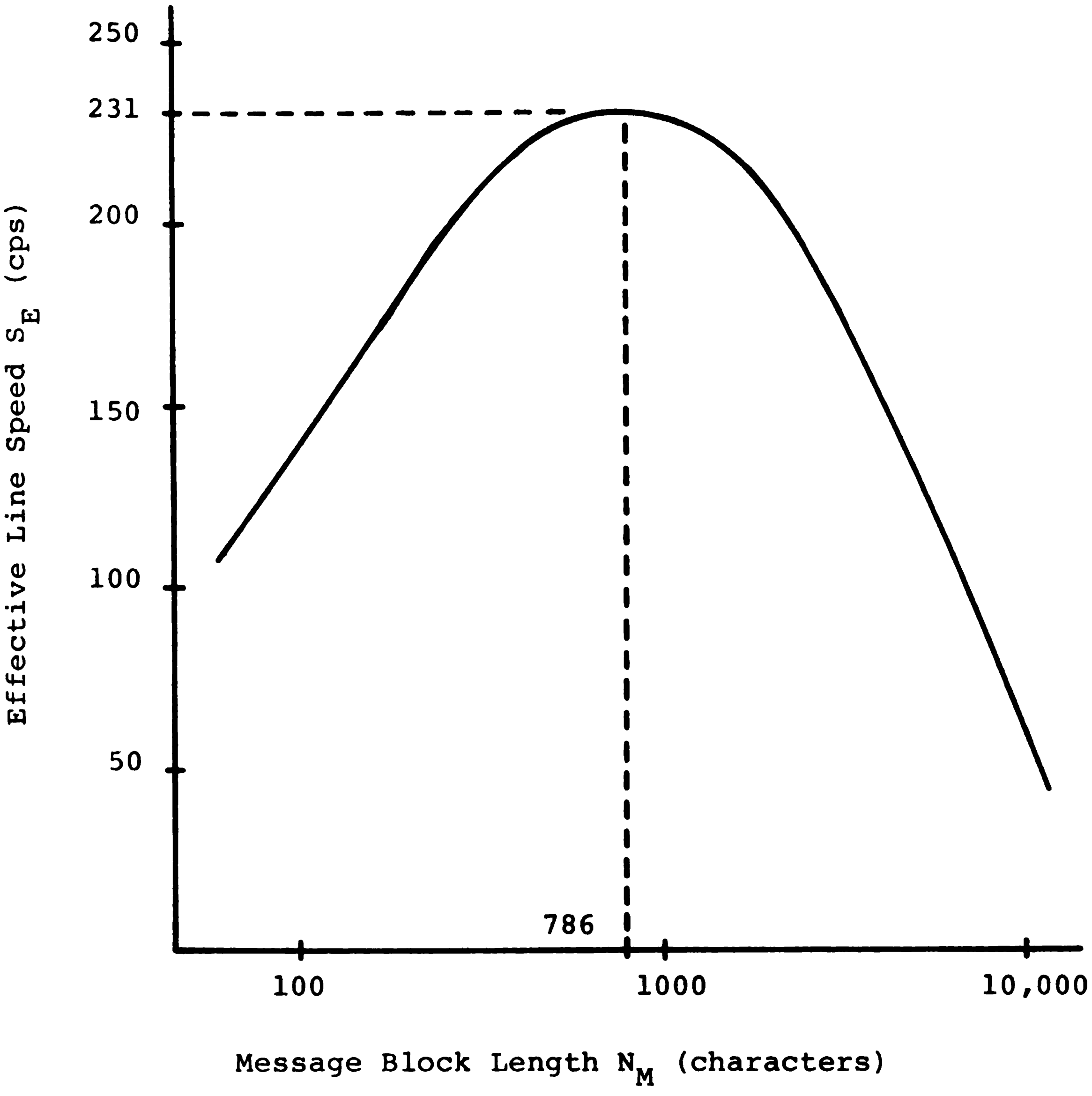
For a given data rate (S) and its associated error rate (P_E), the message block length (N_M) is the only variable available to maximize the effective line speed (S_E). Figure 18 demonstrates the behavior of S_E with varying N_M. As demonstrated in the Figure, there exists a unique N_M for which S_E is maximized for a given set of line conditions. In the Appendix it is shown that the optimal block length is given by:

$$N_M = \tfrac{1}{2}\left[(N_c+2t_L S)^2 - \frac{4(N_c+2t_L S)}{C\ \ln(1-P_E)}\right]^{\frac{1}{2}} - \frac{N_c+2t_L S}{2} \qquad (4.2)$$

The effective line throughput is maximum at this block length. As an example, consider transmission at 2400 bps with t_L = 0.175 sec., N_c = 8 characters, $P_E = 2 \times 10^{-5}$, and C = 8 bits. Optimum block length is N_M = 786 characters. The resultant optimized line speed is S_E = 231 cps = 1848 bps. These values are indicated in Figure 18.

4.3.6 Number of Switched Lines Serving the Host

Since lines connected to the host incur a monthly operating expense, the number of lines must be minimized. But, limited access to the host facility effects a system bottleneck. Contentions for service become inevitable and queues form. A satisfactory compromise must be determined: provide adequate host accessibility at the least monthly expense. To this end, the design goal is to determine the number of lines that provide an adequate grade of service



Given that:

$S = 2400$ bps $= 300$ cps

$P_E = 2 \times 10^{-5}$

$t_L = 0.175$ S

$N_c = 8$ characters

$C = 9$ bits/character

Figure 18. Plot: Effective Line Speed Versus Message Block Length

and an acceptable wait time in the queue of satellites requesting service. Grade of service (P_B) is the probability that all lines to the host are simultaneously busy, and, consequently, a requesting satellite cannot obtain immediate service. P_B determines the mean wait time in the queue, and it is a function of the number of host lines and their utilization. Basic queueing theory will furnish the relations.

The most efficient utilization of the host lines results when all lines are available for use by all satellites. Smaller queues and shorter wait times result with this configuration than with any other. When a satellite and host terminate a call, that line is freed; the host can then dial out or the next satellite to dial the host can obtain a connection. The dispatching discipline at the host, then, is that in which the next satellite to be served is selected at random from the multi-server queue (many lines serving the queue).

To achieve an analytical expression for P_B, the following assumptions are made:

1. The arrival pattern of satellite service requests follow a Poisson distribution
2. Service times (call lengths) are exponentially distributed
3. All lines to the host are equally loaded
4. No service requests leave the queue

From analysis of the application in design, the mean call length or mean service time ($E(t_s)$ seconds) can be approximated using estimated data buffer sizes and the effective line speed (S_E) being considered. Correspondingly, the mean expected number of calls per hour (E(n)/hour) can be estimated. So now,

$$\text{Traffic Volume} = \frac{E(t_s)E(n)}{3600} \text{ erlangs} \tag{4.3}$$

$$= M\rho \text{ erlangs}$$

where M = number of host lines, and ρ = the line utilization. Given these assumptions and definitions, Martin [15, pp. 453-55] gives the grade of service for M lines to the host as:

$$P_B = \frac{1 - \left[\dfrac{\sum\limits_{N=0}^{M-1} \dfrac{(M\rho)^N}{N!}}{\sum\limits_{N=0}^{M} \dfrac{(M\rho)^N}{N!}}\right]}{1-\rho \left[\dfrac{\sum\limits_{N=0}^{M-1} \dfrac{(M\rho)^N}{N!}}{\sum\limits_{N=0}^{M} \dfrac{(M\rho)^N}{N!}}\right]} \tag{4.4}$$

Effective queues of service requests build since satellites will continually dial the host until a connection is obtained. The time a satellite spends dialing can be approximated as the wait time ($E(t_w)$ seconds) in a multi-server queue with random dispatching. Martin [15] gives the mean wait time as:

$$E(t_w) = \frac{P_B}{M} \frac{E(t_s)}{(1-\rho)} \qquad (4.5)$$

Equations 4.4 and 4.5 provide a rudimentary approximation of interprocessor accessibility for a given network composed of equal priority dial-up lines. They serve as the basis for predicting the performance of a communication network in the design process charted in Figure 17.

4.4 Summary

The host facility in a distributed-star system is a general-purpose data processing computer with associated peripherals. Its resident operating system includes a telecommunications access package which provides accessibility among application programs distributed among the host and individual satellites.

Microprocessor-based satellite facilities have sufficient computing capacity to perform the four basic functions at clusters of sensors or controlled devices: local I/O, information concentration and logging, information processing, and telecommunications. The modular satellite implementation provides ease of development and repair, and commonality among satellite facilities comprising a system.

The host-satellite communication links are often dial-up or leased common-carrier lines, the latter employed whenever switched-network service delays cannot be tolerated.

On dial-up lines, the speed of service can be prohibitively long in some cases--including both switched network connect time and queue wait times. Reliable data communications can be achieved with good error control design, e.g., a net bit error rate of 10^{-9} has been demonstrated [13]. In order to maximize the effective data rate, messages should be arranged in an optimum block length which is determined by the error probabilities and turn-around time of the line. An additional consideration in achieving least recurring monthly expense is minimizing the number of lines connecting the host while providing satisfactory accessibility.

CHAPTER V

EXAMPLE APPLICATION: INSTITUTE OF WATER RESEARCH WATER QUALITY MANAGEMENT PROJECT

The remote monitoring and control needs of the Institute of Water Research Water Quality Management Project (WQMP) exemplify a class of processes that are best managed by a distributed computing system. The land treatment ecosystem described below is composed of a multiplicity of individual processes distributed over a reasonably wide geographical region. In fact, this is generally the situation encountered in wastewater reclamation projects, and moreover, for water pollution control in general. Such processes, especially the WQMP, and a distributed-star computing system are perfectly matched.

In this chapter, a distributed computing system is proposed that automates the data acquisition tasks contained in the WQMP. This computing system can naturally evolve to assume more and more project control functions. An overview of the WQMP is presented below followed by an outline of project operation and computing system requirements. Then, a distributed-star computing system is described that employs μP-based satellite facilities to control activities at the process points.

5.1 Description of the WQMP Facility [19]

Michigan State University (MSU) constructed on 500 acres of the main campus a permanent facility for the experimental treatment, recycle and reuse of municipal sewage plant effluents. The facility provides for the diversion of up to 2×10^6 gallons per day of secondary effluent from an activated sludge treatment plant. This waste flow is directed to an intensely managed aquatic and terrestrial nutrient recycling system. The facility consists of a portion of the East Lansing Wastewater Treatment Plant, a transmission line, four experimental lakes, and a spray irrigation site. A primary objective is to strip nutrients from the waste flow as they proceed through the system by incorporating nutrients into harvestable biomass. For discussion purposes here, a simplified version of the land treatment facility is depicted in Figure 19.

Within the watershed, the land is not homogeneous. Soil conditions, ground cover, exposure, and terrain necessitate that the basin be decomposed into a number of distinct spray zones; eight are shown. Except for overland flow, which occurs under certain isolated conditions, each of these eight zones may be managed independently.

A feedlot upstream on Felton Drain may discharge significant pollutants into that stream. Water quality in the stream must be managed, as well as the rate at which wastewater residues move down the stream to the Red Cedar

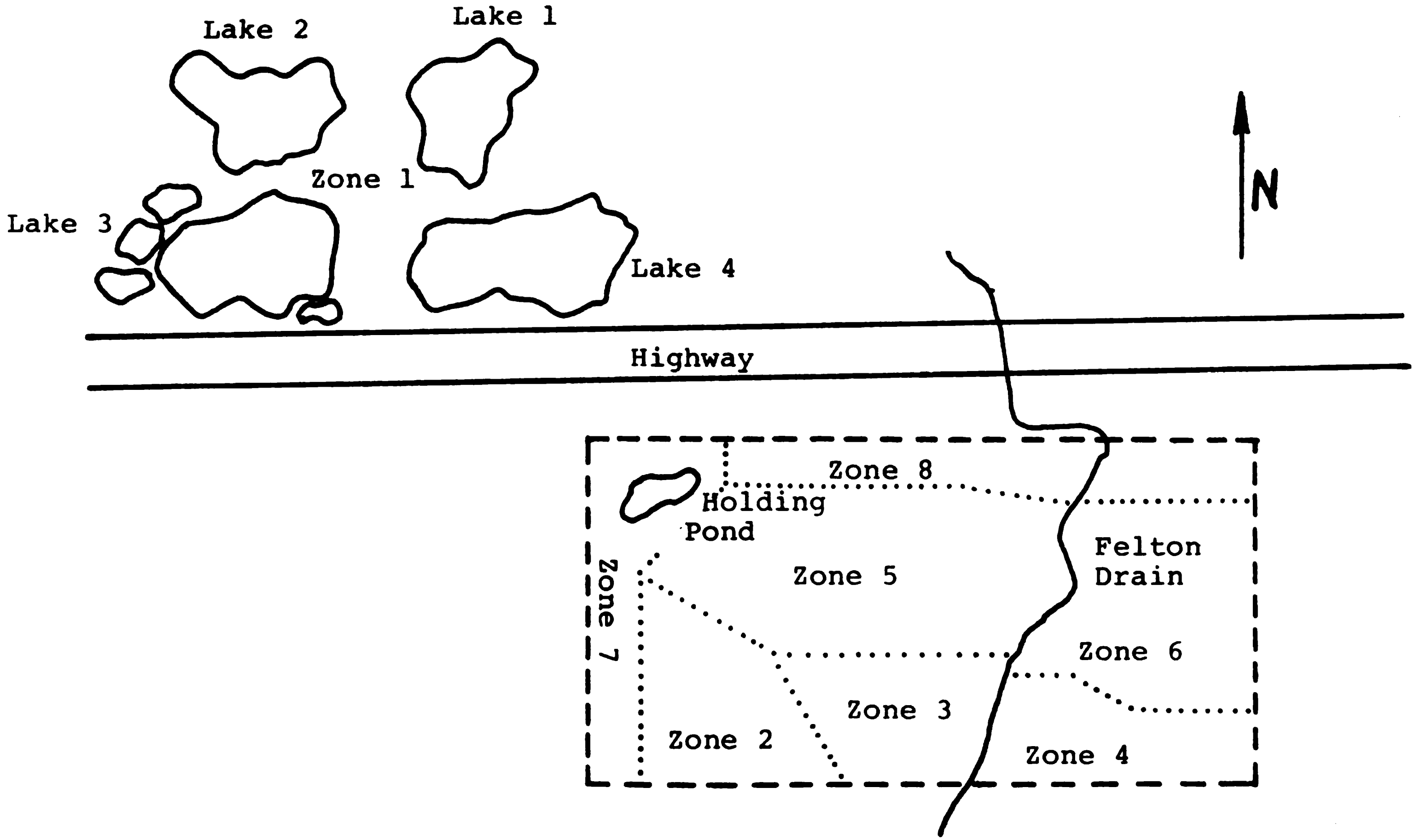


Figure 19. Illustration of the Land Treatment Facility

River. The highway is only shown because, if winds above 25 mph come out of the south for more than ten consecutive minutes, spraying in Zone Eight can result in undesirable aerosols reaching the roadway. The holding pond is necessary because there are times when the rate at which wastewater effluents enter the facility exceed the rate at which they may be applied safely and effectively to the landscape.

Water may enter the ecosystem by Felton Drain, through rainfall, or by the input of wastewater effluent. Water leaves the ecosystem principally by Felton Drain, infiltration-percolation to the aquifer, and through evaporation.

5.2 Project Management

Managing project operations round-the-clock can greatly facilitate optimization efforts and research studies. To this end, an integrated remote monitoring system can supply current measurements of the prevailing ambient conditions and treatment facility status, and can provide the essential data base for developing predictive ecosystem models and optimized project management strategies. As progress in these areas is realized, additional project functions may be automated to achieve safe and optimal project operation.

At the outset, local climatological data acquisition, stream hydrological data acquisition, and *in situ* water quality monitoring stations must be automated to provide current and historical measures of ambient conditions, and facility status and performance. Evaporation rates must be computed for each zone. Local wind conditions must be monitored in Zone 8 so that aerosols can be kept from the highway. Rainfall and overland flow in each of the zones should be known, as well as stream flow at selected points along Felton Drain. As important as these, water quality, including nutrient levels, at the four lakes, holding pond, and Felton Drain facility boundary must be observed and logged.

Ultimately, these abiotic parameters will be continuously monitored and put into the predictive ecosystem models. Application rates for each zone will then be adjusted according to the prevailing conditions, as indicated by the models. The models will also identify key times when stream, shallow-well, and deep-well water quality should be assessed, as well as times for planting or harvesting crops or for conducting biological surveys. Monitoring and managing the holding pond in real time might improve system performance because of the characteristics of the wastewater in the holding pond. For example, at times it might be necessary to bubble oxygen in the pond or to harvest vegetation.

This approach to managing land treatment facilities has greater overhead because of the added things which must be monitored and controlled and because some key parameters must be monitored continuously. Even so, it is not at all obvious that this is more expensive than manual operation.

Maximum application rates are possible for the automated approach; hence, less land area is required to dispose of the wastewater effluents. Secondly, because tighter control is maintained over the ecosystem, accidental discharges of aerosols or pollutants into the streams are minimized. Finally, yield from harvested croplands are maximized with this approach.

Data for archival purposes will be collected in order to document the performance of the land treatment facility and, to track the long-term characteristics and fluctuations in the treatment facility and the associated ecosystem. For example, the following information might be collected in addition to that already stipulated: weekly water quality data in representative shallow and deep wells; daily application rates and duration of the applications for each of the spray zone; weekly planting, crop development, and harvesting records.

5.3 Proposed Computing System

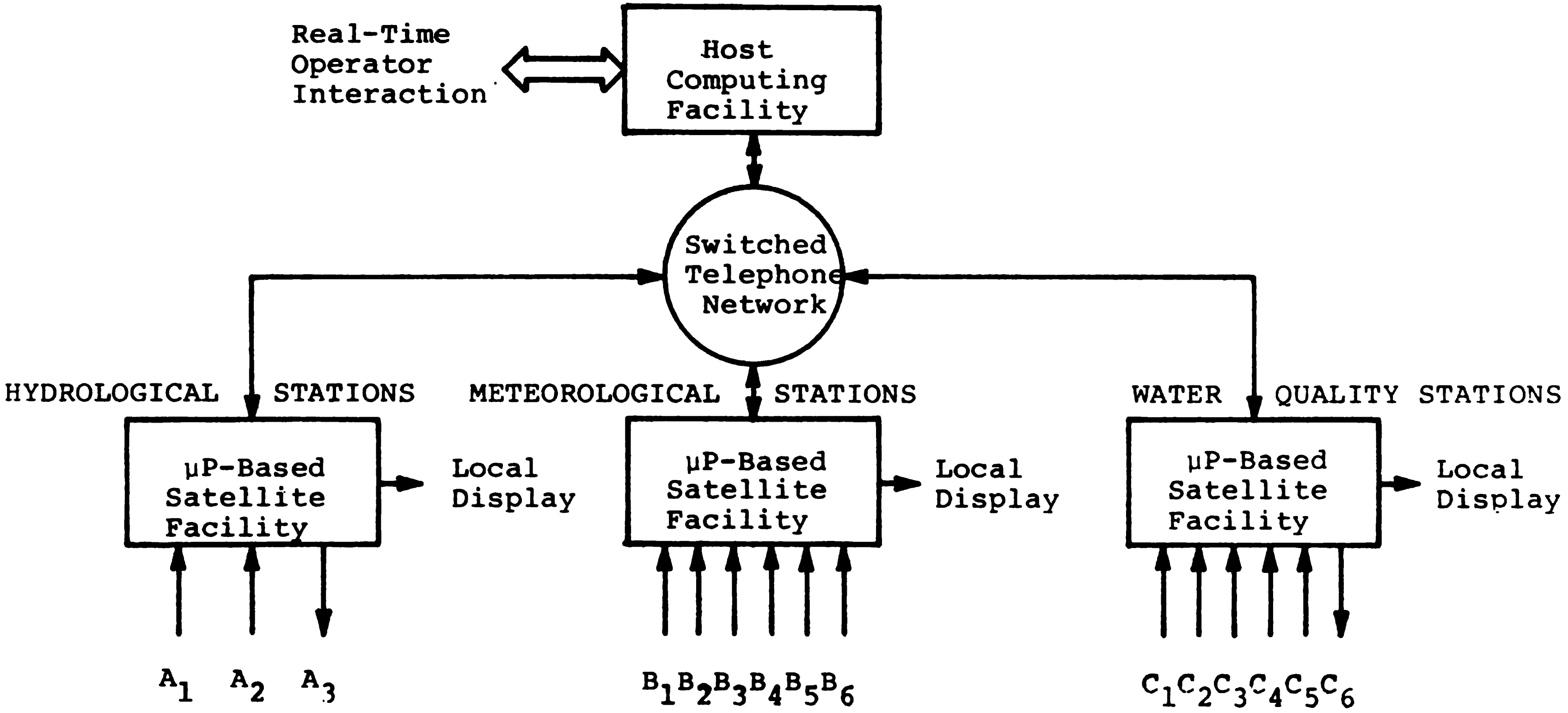
The distributed-star computing system configuration using dial-up lines (detailed in Chapters III and IV) is ideally suited to the WQMP application. Auspiciously, tasks in the land treatment facility occur naturally in groups which can be assigned to dedicated satellites at the remote sites. Also, communication delays imposed by the use of dial-up lines are of no consequence since the process time constants are large in comparison.

Fundamental task groups assigned to individual satellite facilities are: (1) water quality data acquisition, (2) climatological data acquisition, and (3) hydrological data acquisition. In addition to monitoring tasks, some control of external devices must also be performed at the remote sites. Both command and feedback signals must be input/output to regulate water sample collectors, water quality instrumentation, pumps, control valves, and local displays. Figure 20 depicts the fundamental system arrangement and task assignments. The stations shown would occur in multiplicities and be geographically distributed as required.

Functions committed to the host computer facility are: (1) supervision and synchronization of satellite facilities, (2) creation and management of the central system data base, (3) generation of operations statistics, summaries, and other documentation, (4) real-time operator/analyst interaction, (5) support of ecosystem management models. Modelling studies will require the resources of a larger computer facility, i.e., the MSU CDC 6500, to perform demanding simulations. Consequently, the WQMP system host facility must provide a data communication capability compatible with the CDC 6500.

5.3.1 Host Computer Facility

These host functions can be satisfactorily performed by a 16-bit minicomputer with at least 16K-24K main memory, fast-access mass storage (flexible disk or disk pack), an



A_1 Weir Level
A_2 Stream Temperature
A_3 Water-Sample Collector

B_1 Air Temperature
B_2 Dew Point
B_3 Rain Gauge
B_4 Wind Speed
B_5 Wind Azimuth
B_6 Solar Radiation

C_1 Turbidity
C_2 Water Temperature
C_3 Dissolved Oxygen
C_4 pH
C_5 Conductivity
C_6 Instrument and Pump Control

Figure 20. Block Diagram of the Rudimentary WQMP Computing System

appropriate bulk storage peripheral (magnetic tape), interactive terminal(s), and the required data communication equipment.

5.3.2 μP-Based Satellite Facilities

At the remote sites, there are no operations demanding high-speed data manipulation or arithmetic processing. Even if digital filtering (averaging) is employed, the low bandwidths of environmental signals impose minimal processing speed requirements. Consequently, general-purpose microprocessors, with typical instruction cycle times of one to ten microseconds, provide sufficient computing capacity to easily implement the remote-site control algorithms.

The satellite computers in Figure 20 could be comprised of the hardware configurations developed in Chapter IV (see Figures 10-14). Located at the remote sites, the satellite computer functions as the local data acquisition system and data logger, data processor, and external device controller. Peripheral hardware operations required at the satellite facility include signal conditioning and multiplexing of sensor signals, conversion of analog signals to the digital domain, and local external device control. Also, either an answer-only modem or a modem-automatic calling unit combination is attached to the telecommunications interface (see Figures 10 and 11).

Satellite computer memory requirements will vary directly with the complexity of the control algorithms, the number of attached sensors, and the time interval between data dumps to the host computer. Typically, a satellite control program of modest complexity would occupy 4K words of ROM (or EPROM). In addition, a data buffer consisting of 4K words of RAM would be appropriate for most stations performing data acquisition.

5.4 Summary

A distributed-star computing system can successfully and inexpensively automate several portions of the WQMP. Requisite remote-site tasks can be performed comfortably by μP-based satellite computers and associated peripheral equipment. Furthermore, since communication delays between satellites and the host are of no consequence, dial-up lines can be used to interconnect the system computers. By exploiting these economical resources, the resulting distributed-star system becomes a very attractive solution to WQMP automation.

As research progresses and optimized facility management strategies are determined, automation of the project can proceed further. Additional satellite facilities, performing both monitoring and control functions, can be gracefully integrated along with upgrades of the host control algorithm. The distributed-star configuration facilitates this gradual evolution towards project optimization.

CHAPTER VI

SUMMARY

Overall, the primary objective of the investigation reported in this thesis is to study a computing system architecture(s) which successfully automates remote monitoring and control of large physical processes in a cost-effective practical manner. Generally, such a solution to the automation problem was difficult to attain in the past. The high cost of computing intelligence motivated centralized system configurations whose performance was intrinsically limited by: (1) the expensive bandwidth of the communication channels linking remote sites, and (2) sequential performance of tasks. These constraining factors must be diminished to improve the real-time performance of large computing systems. The recent achievement of practical, inexpensive, general-purpose microprocessors greatly improves the cost-effectiveness of a distributed-intelligence approach to performing remote monitoring and control operations. This alternative approach alleviates the previous performance-limiting factors and proves to be advantageous in the real-time management of geographically large and diverse physical systems.

Specifically, computing system architectures were investigated in which computing intelligence is distributed

away from a central control location to the process points. In the distributed architectures, control activities and data processing which do not involve the resources of the host facility are performed locally at the remote site (cluster of process points) by a dedicated satellite computing facility. System operations are synchronized and supervised by the central host computing facility which also supports a central data base. For geographically large systems, common-carrier services (lines) are often employed for interprocessor communications.

Distributed-intelligence system architectures effect several significant improvements over the centralized approach. These improvements are:

1. Minimized communication between the central host and the process points
2. Improved real-time performance from shorter, predictable response delays and a broadened range of implementable control algorithm complexity
3. Isolated performance of tasks allows truly modular implementations that facilitate development, expansion, and maintenance; and provide higher reliability and fault tolerance
4. Reduced system complexity and cost
5. Improved accuracy of acquired data

Furthermore, the conflict of optimization criteria--good real-time performance and high host computer productivity--can be overcome in a distributed system. This is primarily the result of task partitioning, i.e., the performance of most real-time tasks in parallel by dedicated satellite processors.

Some problems remain however. In a distributed system that accesses satellite and host facilities over dial-up lines, significant communication delays must be tolerated. In addition, the use of a priority-structured queue often results in a decrease in the utilization of communication facilities and, hence, increases the recurring monthly operating expense. Also, systems providing fast response among interrelated satellite operations, i.e., common-bus-organized configurations, can be expensive and difficult to implement. Initial system programming, system evolution, and system expansion can be monumental chores. Likewise, in distributed systems that employ a distribution of application control with a mix of control roles among satellites and host, system software becomes complex and expensive.

Three fundamental elements comprise a distributed computing system: a central host computing facility, satellite computing facilities, and communication links interconnecting host and satellites. The host facility is made up of a general purpose computer supporting data processing I/O peripherals and telecommunications hardware, and whose resident operating system contains a telecommunications access package. Satellite facilities are often composed of a μP-based computer(s) equipped with local I/O and telecommunications hardware. The microcomputers are relatively simple machines with moderate memory requirements (ROM and RAM), limited console functions, and a simple interrupt structure. Interprocessor communication lines often effect a recurring monthly expense. An iterative communication network design process was presented that minimizes this operating expense. Determined in the design process were: line data rate, line requirements at the host, optimal message block length, grade of service provided to the satellites, and mean wait time for satellite service requests. In addition to the operating expense, the communication lines are the primary source of errors in the system. However, effective error rates of 10^{-9} have been demonstrated using an error detection-retransmission scheme of error control [12].

The monitoring and control needs of the WQMP typify a class of applications best implemented by a distributed-star system configuration. Since the service delays imposed by dial-up lines can be tolerated, the simple distributed-star system investigated in this thesis can very efficiently manage the operation of the WQMP. This approach provides the economy, flexibility, maintainability, and ease of development and expansion needed to successfully manage the land treatment facility.

With advancements in LSI technology, the electronic elements of computing systems (CPUs, memories, peripheral interfaces, etc.) will continue to decrease in cost. Concomitantly, the cost of data communication will continue to grow in its relative percentage of computing system costs.

So, for now and in the future, the amount and rate of data communication occurring within a computing system must be minimized to reduce operating expense. This is generally accomplished within distributed-intelligence computing systems. For this reason, distributed systems will become increasingly cost-effective, and, thus, increasingly attractive for application in areas not heretofore economically feasible. Distributed system architectures have a growing future role in an expanding family of application areas.

APPENDIX

APPENDIX

DERIVATION OF EFFECTIVE LINE SPEED AND OPTIMUM MESSAGE BLOCK LENGTH

When the error detection-block retransmission procedure for error control is employed, the resulting effective line speed (S_E cps) is less than the modem operating speed (S cps). In this analysis, all data transmission occurs at a constant Baud rate. The sequence of events across the line is:

...[t_L][Msg A][t_L][ACK][t_L][Msg B][t_L][NAK][t_L][Msg B]...

where t_L = total turnaround time of the line and modems

Msg A = duration in seconds of message block A

ACK = duration in seconds of an acknowledgement of accurate message reception

NAK = duration in seconds of a negative acknowledgement (error detected)

Assuming that the line behaves as a binary symetric channel with an error probability of P_E, the mean time elapsed in the transmission of one block is:

$$t_B=2t_L+\frac{N_M+N_C}{S} + [(2t_L+\frac{N_M+N_C}{S})(P_R+P_R{}^2+P_R{}^3\ldots)]$$

$$t_B=2t_L+\frac{N_M+N_C}{S} + [(2t_L+\frac{N_M+N_C}{S})(\frac{P_R}{1-P_R})]$$

$$t_B=[2t_L + \frac{N_M+N_C}{S}][1 + \frac{P_R}{1-P_R}] \qquad (A.1)$$

where t_B = total seconds taken to transmit one message block

N_M = number of message characters contained in the message block

N_C = total number of control characters used in the procedure including ACK, NAK, identification, error check, etc.

$$P_R = 1 - (1-P_E)^{C(N_M+N_C)} \qquad (A.2)$$

= probability that a block must be retransmitted

C = number of bits per character

The effective line speed is:

$$S_E = \frac{N_M}{t_B} \qquad \frac{\text{characters}}{\text{second}} \qquad (A.3)$$

Substituting Equations (A.1) and (A.2) into (A.3) and rearranging terms yields:

$$S_E = S \left[\frac{(1-P_E)^{C(N_M+N_C)}}{(\frac{N_M+N_C}{N_M})+(\frac{2St_L}{N_M})} \right] \qquad (A.4)$$

This result appears as Equation (4.1) in the text.

S_E is a maximum for a unique message block length. The optimum N_M can be found by differentiating Equation (A.4) and equating to zero.

$$\frac{d}{dN_M} S_E = \frac{[\ln(1-P_E)CN_M+1](1-P_E)^{C(N_M+N_C)}}{\frac{N_M+N_C}{S} + 2t_L}$$

$$- \frac{N_M(1-P_E)^{C(N_M+N_C)}}{S(\frac{N_M+N_C}{S} + 2t_L)^2} \qquad \text{(A.5)}$$

Setting Equation (A.5) to zero gives:

$$(N_M)^2 + S(\frac{N_C}{S} + 2t_L)N_M + \frac{S(\frac{N_C}{S} + 2t_L)}{\ln(1-P_E)C} = 0 \qquad \text{(A.6)}$$

Solving Equation (A.6) yields the optimum message block length:

$$N_M = \tfrac{1}{2}[(N_C+2t_LS)^2 - \frac{4(N_C+2t_LS)}{C \ln(1-P_E)}]^{\frac{1}{2}} - \frac{N_C+2t_LS}{2} \qquad \text{(A.7)}$$

This expression is Equation (4.2) in the text.

LIST OF REFERENCES

LIST OF REFERENCES

1. Rodnay Zaks, "A Processor Network for Urban Traffic Control," IEEE 1973 Computer Society International Conference (COMPCON '73) Digest of Papers (February 1973), pp. 215-18.

2. Scott E. Cutler, "Microcomputer Networks in Automobile Traffic Control," Tenth IEEE Computer Society International Conference (COMPCON '75) Digest of Papers (February 1975), pp. 263-66.

3. LeRoy H. Anderson, "The Microcomputer as Distributed Intelligence," Proceedings IEEE 1975 International Symposium on Circuits and Systems (April 1975), pp. 337-40.

4. Frank V. Wagner, "Is Decentralization Inevitable?" Datamation, Vol. 22, No. 11 (November 1976): 86-97.

5. Alan J. Weissberger, "Microprocessors as Intelligent Remote Controllers," 1974 WESCON Technical Papers, Vol. 18 (September 1974), 23/4: 1-4.

6. M.D. Balkovic, H.W. Klancer, S.W. Klare, and W.G. McGruther, "High-Speed Voiceband Data Transmission Performance on the Switched Telecommunications Network," The Bell System Technical Journal, Vol. 50, No. 4 (April 1971): 1349-84.

7. H.C. Fleming, R.M. Hutchinson, Jr., "Low-Speed Data Transmission Performance on the Switched Telecommunications Network," The Bell System Technical Journal, Vol. 50, No. 4 (April 1971): 1385-1406.

8. Bradley J. Beitel, Virginia B. Nagel, "An Approach to Generalized Distributed Systems," Advances in Instrumentation, Vol. 29, Part 1 (1974), 518: 1-5.

9. "Data Communications Using the Switched Telecommunications Network," Bell System Technical Reference, American Telephone and Telegraph Company, PUB 41005, May 1971.

10. EIA Standard RS-232-C, "Interface Between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange," Electronic Industries Association (1969).

11. EIA Standard RS-366, "Interface Between Data Terminal Equipment and Automatic Calling Equipment for Data Communication," Electronic Industries Association.

12. R.L. Townsend, R.N. Watts, "Effectiveness of Error Control in Data Communication over the Switched Telephone Network," The Bell System Technical Journal, Vol. 43, No. 6 (November 1964): 2611-37.

13. James Martin, Teleprocessing Network Organization, Prentice-Hall, Inc., 1970.

14. "Error Distribution and Error Control Evaluation," Extracts from Contribution GT. 43, No. 13, February 1960, CCITT Red Book, Vol. VII, The International Telecommunication Union, Geneva, 1961.

15. James Martin, Systems Analysis for Data Transmission, Prentice-Hall, Inc., 1972.

16. John E. Buckley, "Local System Access," Computer Design, Vol. 15, No. 12 (December 1976): 20-22.

17. Mondy Lariz, "Doing an End Run Round Bell's Plans to Time Local Calls with Message Units," Data Communications, February 1977, pp. 29-32.

18. Robert W. Lucky, "Common-Carrier Data Communication," in Computer-Communication Networks, pp. 142-96, edited by Norman Abramson and Franklin F. Kuo, Prentice-Hall, Inc., 1973.

19. "Utilization of Natural Ecosystems for Wastewater Renovation," Final Report EPA Project No. Y005065, Institute of Water Research, Michigan State University, March 1976.