THE DISTRIBUTED SYSTEM ENVIRONMENT

SOME PRACTICAL APPROACHES

Grayce M. Booth
Honeywell Information Systems Inc.

McGraw-Hill Book Company

New York St. Louis San Francisco Auckland
Bogotá Singapore Johannesburg London
Madrid Mexico Montreal New Delhi
Panama Sao Paulo Hamburg
Sydney Tokyo Paris
Toronto

Library of Congress Cataloging in Publication Data

Booth, Grayce M., date.
The distributed system environment.

Bibliography: p.
Includes index.
1. Electronic data processing—Distributed processing. I. Title.
QA76.9.D5B66 001.64 80-13494
ISBN 0-07-006507-1

234567890 KPKP 8987654321

The editors for this book were Barry Richman and Susan Thomas, the designer was Mark E. Safran, and the production supervisor was Teresa F. Leaden. It was set in Baskerville by the Fuller Organization.

Printed and bound by The Kingsport Press.

THE DISTRIBUTED SYSTEM ENVIRONMENT

PREFACE

An important trend toward the distribution of computing resources is under way. One facet of this trend involves the interconnection of hitherto independent computer centers, while another facet involves the creation of complex information systems consisting of large-scale, general-purpose computers, minicomputers, and large numbers of terminal devices—often intelligent terminals.

Many industry buzzwords are being used to describe this trend toward distributed functionality: *distributed systems, distributed processing, distributed databases, distributive processing,* and so on. Unfortunately, these buzzwords have no generally understood definitions, and so they mean many different things to different people. One of the goals of this book is to provide a common set of terminology to describe distributed information systems so that the reader can grasp the fundamental principles which lie beneath the buzzwords.

This book will provide the information system analyst, designer, or programmer with a good understanding of this new distributed system environment. Through the presentation of numerous examples, the reader will learn how distributed systems can be configured and used. The reader will also learn how to evaluate the advantages and disadvantages of distributed systems.

Major sections are devoted to distributed processing and distributed database structures, as well as to the network or data communications structures used to interconnect the component elements of distributed systems.

Using the above basic descriptive material, this book analyzes the network or system architectures of several vendors, describing how each vendor attacks the distributed system problem and which system structures each supports.

The reader will also be given a thorough grasp of how to design distributed systems, including how to make strategic decisions about whether to distribute or centralize computing facilities.

The emphasis throughout is on providing a practical understanding of workable methods for the implementation of distributed systems. Advanced concepts such as dynamic load leveling are described, but in the perspective of what can generally be implemented today at a reasonable cost. Pitfalls and problems are also described, so that the reader can avoid these by benefitting from the experience of others.

Perhaps the most important function of this book is to bring together all of the pieces of the emerging technology for the distributed system environment and to present them in a logical, coherent manner. Most readers will have encountered many of the concepts presented—in trade journals, computer conferences, and so on. Viewed independently and in fragmented form, however, many of these concepts are quite confusing. This book brings them all together and puts distributed systems into clear focus—perhaps for the first time.

With this thorough grounding in the subject, the designer, analyst, or programmer will be able to make an intelligent decision about whether a distributed system would best serve the needs of his or her own organization.

GRAYCE M. BOOTH □

ACKNOWLEDGMENTS

The material collected in this volume reflects not only my experience with the analysis of distributed systems but also the results of many, many discussions with people involved in various aspects of system design and use. Since I cannot individually acknowledge all of these contributions, I will simply thank the three people who gave me the most help.

First, I must thank Michel Godard, of Cii Honeywell Bull in Paris. I first became interested in distributed systems in 1974, when Michel invited me to visit a large Paris-based bank engaged in the design of a complex system. That introduction to distributed processing in the real world set me on the path which led to this book.

I would also particularly like to acknowledge the contributions of Hal B. Becker and of Walter O. Bailey, Jr., both of Advanced Computer Techniques Corporation. A number of the ideas presented in this volume are Hal's or Walt's. I have simply collected and arranged them, adding my own contributions. Without our many discussions over the years, this book would not have been the same.

THE DISTRIBUTED SYSTEM ENVIRONMENT

CONTENTS

SECTION • 1

INTRODUCTION

1

BACKGROUND

Distributed information systems are built upon the principle of functional distribution. This means that not all processing functions are performed by a single element; rather, they are distributed to multiple elements, and some of these elements may be moved close to where the users of the functions are located. This contrasts strongly with the concept of centralized functionality, in which all computing power is located in one processing element. Centralized functionality requires users to come to the computer or, at best, to have remote access to centralized computing resources.

1-1. TRENDS

Looking back over the last twenty years, we can see how distributed systems evolved. During the late 1950s and early 1960s, when computers were coming into general use, many organizations acquired first one computer, then another, and another, and another. . . . Each was an independent entity, and often these *decentralized* systems led to duplication of staffing, to duplication of hardware and facilities costs, to incompatible procedures and methods, and far too often to lack of management control.

Starting in the mid-60s and continuing to the early 1970s, computer *centralization* for economy of scale became popular. Larger and larger computers were being built, and "Grosch's law"* stated that for twice as much money you could buy a processor with four times as much computing power as a smaller machine. In addition, duplicate staffs and facilities were no longer needed and—most important—management control could be exercised more easily.

Although attractive in terms of cost and control, centralization sometimes led to lack of responsiveness. The centralized data processing (DP) department served many user groups (e.g., engineering, accounting, marketing . . .) with conflicting needs. When trade-offs between those needs were made, none of the users might be completely happy with the

*Stated by Herb Grosch, noted industry personality.

results. User groups began to say, in effect, "If only we could have our own computer. . . ."

It was difficult, however, for departments other than data processing to acquire computers, because all purchase or lease requests above a certain dollar value typically had to be approved by the central DP staff. Naturally, it was hard to get this approval except in special cases such as process control.

This situation was changed by the advent of inexpensive minicomputers. With minis such as Digital Equipment's PDP-8, it became possible to buy a fairly significant amount of computing power for a relatively small amount of money. Small enough, in fact, to be within local management approval authority—without central data processing management review.

So, the "minicomputer revolt" began, in an attempt to return responsive computing to individual using departments. However, many things had changed since the early 1960s. Business, government, and other organizations had become far more complex, leading to greater need for integrated information-processing systems. In most cases, it was no longer feasible to install truly independent decentralized computers, because each computer needed data available from some other computer.

Distributed information systems, which evolved as a result of this situation, consist of multiple computers and/or intelligent terminal or device controllers interconnected to form a coordinated system. For example, a user who has a large central system such as an IBM 370 might expand by installing minis in remote locations, connected to the 370 for the exchange of data to coordinate the operations of the maxi and the minis. Or a user with two independent data centers, both using large-scale general-purpose computers, might interconnect the two centers—again for data exchange.

These are the two basic trends in distributed systems:

1. To expand an existing central system by connecting intelligent remote devices to it.

2. To interconnect previously independent decentralized computers for greater coordination and cooperation.

Looking at a single computer system, we can also see that the trend to distributed functionality has been under way for some time. Early computers such as the IBM 650, 305 RAMAC, and 1401 were fully integrated: the processor performed I/O (input/output) operations as well as computations, and so these functions could only proceed alternately. However, this situation soon changed. Peripheral controllers (also called

peripheral processors) were given their own logic for performing I/O in parallel with computation. This was the first step toward the distribution of functionality.

Front-end network processors (or *front-ends*) evolved next to perform remote I/O functions, interfacing with terminal devices, in parallel with local I/O and processing. Further distribution in the form of a database processor, or *back-end processor,* has been implemented in some systems. This trend toward functional distribution within a single computer system is shown in Figure 1-1.

This type of functional distribution can be called horizontal, in the sense that there is generally no master-slave relationship among the elements. All cooperate at an equal level, logically, to perform a complex set of tasks. Such functional distribution within a single computer system has been commonplace for some time—so much so, in fact, that it tends to be taken for granted. In general, these techniques are not included when the terms *distributed processing* or *distributed systems* are used.

A more recent trend, emerging from the evolution described earlier, is vertical distribution. In this case, functionality is distributed across relatively independent but interconnected elements that form a hierarchy. These elements share tasks in a structured way. Each component either is controlled to some degree by the higher-level member(s) of the hierarchy—or is at least limited to a given set of functionality by its position in the hierarchy.

The trend toward vertical distribution usually involves multiple separate elements—information processors of various sizes, concentrators, terminals, and perhaps real-time devices. Because these elements are logically related in the form of a hierarchy, vertical distribution is called *hierarchical distribution.* This form of distributed functionality is shown in Figure 1-2.

Hierarchical distribution often results when minis and/or intelligent terminals are attached to an existing central information processor in accord with one of the trends mentioned earlier.

Horizontal distribution, which began within a single computer system (as in Figure 1-1), is now expanding to encompass multiple computer systems, as shown in Figure 1-3.

Horizontal distribution usually results from the interconnection of computers which were previously free-standing or decentralized. The

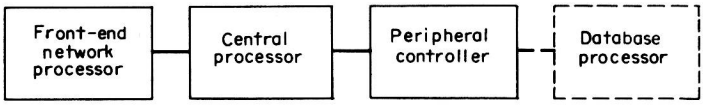

FIG. 1-1 Intrasystem functional distribution.

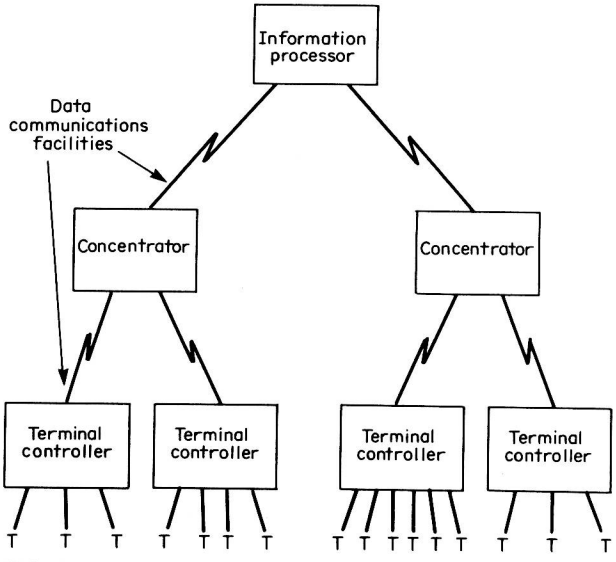

FIG. 1-2 Hierarchical distribution. T = terminal device.

difference between this form of distribution and the hierarchical form shown in Figure 1-2 is that these horizontally distributed computers cooperate as equals, or peers, sharing the total workload. In contrast to the hierarchical system, no concept of "level" exists to specialize the functionality of the information processors.

The widespread use of distributed functionality is relatively recent. Programmable logic formerly existed only in central computer systems. Devices such as terminals were hard-wired to perform a specific set of functions, because any additional logic for added flexibility would increase the cost of these devices unacceptably. Today, however, low-cost technologies such as large-scale integration (LSI) and the coming very large-scale integration (VLSI) make it feasible to include intelligence in

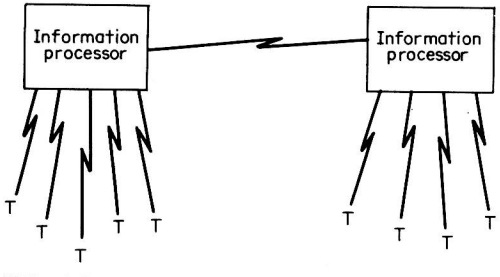

FIG. 1-3 Horizontal distribution. T = terminal device.

devices which were formerly hard-wired. The explosive growth in the use of minicomputers, one result of today's low-cost technologies, provides a major impetus for the growth of distributed systems.

Another reason for the growth of distributed functionality is the increasing use of communications-based systems. These systems are by nature physically distributed, although initially it is usually the *human* functions which are distributed, rather than the *computer-based* functions. The trend, however, tends to favor evolution from distributed access (distributed users) to distributed computing functionality.

Remotely oriented systems are often larger and more complex than the average batch system, because they are designed to handle large and complex real-world problems.

1-2. BENEFITS

All of these trends lead toward increased use of distributed systems because, in the right circumstances, these systems offer significant benefits as contrasted with centralized or decentralized information systems.

Coordination can be better in a distributed system than in a decentralized system. This means that the information system will serve its users more efficiently, often at a lower cost.

Response—how quickly a terminal user can get an answer—is often better in a distributed system than a centralized system. This is particularly true as the centralized system becomes larger and larger, serving more and more on-line users. At some point these users interfere with each other so much that response degrades unacceptably. Moving functions out of the central system (often called the "host" in a distributed system) lessens interference and improves responsiveness.

Availability or more accurately *survivability,* is an inherent characteristic of most distributed systems. Because system components are distributed, and relatively independent, it is extremely rare for any problem to cause failure of the entire system. In a branch banking system, for example, with a mini in each branch, failure of one mini affects only one branch. If the branches were all connected to a single central computer system in contrast, failure of that computer would affect all branches. Survivability is very important where on-line computer systems are tied directly into the day-to-day operations of the organization, and survivability may be achieved more economically in a distributed system than in a centralized system.

Lower cost of data communications facilities can result from moving processing—and perhaps databases—close to system users, instead of transmitting large amounts of data to and from a central computer site.