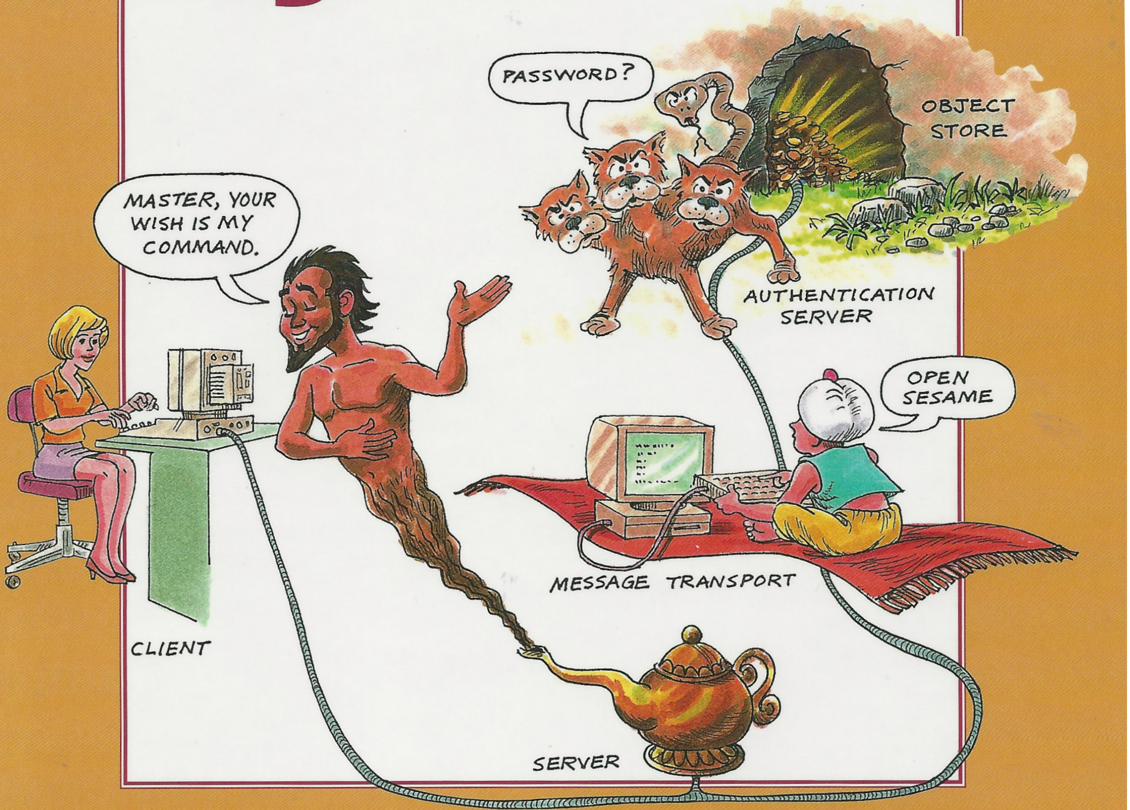
Distributed Operating Systems



Andrew S. Tanenbaum

Distributed Operating Systems

Andrew S. Tanenbaum

Vrije Universiteit
Amsterdam, The Netherlands

PRENTICE HALL
UPPER SADDLE RIVER, NEW JERSEY 07458

Library of Congress Cataloging-in-Publication Data

Tanenbaum, Andrew S.
Distributed operating systems / Andrew S. Tanenbaum.
p. cm.
Includes bibliographical references and index.
ISBN 0-13-219908-4
1. Distributed operating systems (Computers) I. Title.
QA76.76.063T357 1995
005.4'4--dc20 94-27646
CIP

Acquisitions Editor: Bill Zobrist
Production Supervisor: Joe Scordato
Cover Designer: DeLuca Design
Buyer: Linda Behrens
Supplements Editor: Alice Dworkin
Interior Designer: Andrew S. Tanenbaum

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, express or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

Printed in the United States of America

10 9

ISBN 0-13-219908-4

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Pearson Education Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Contents

1

Introduction to Distributed Systems

Computer systems are undergoing a revolution. From 1945, when the modern computer era began, until about 1985, computers were large and expensive. Even minicomputers normally cost tens of thousands of dollars each. As a result, most organizations had only a handful of computers, and for lack of a way to connect them, these operated independently from one another.

Starting in the mid-1980s, however, two advances in technology began to change that situation. The first was the development of powerful microprocessors. Initially, these were 8-bit machines, but soon 16-, 32-, and even 64-bit CPUs became common. Many of these had the computing power of a decent-sized mainframe (i.e., large) computer, but for a fraction of the price.

The amount of improvement that has occurred in computer technology in the past half century is truly staggering and totally unprecedented in other industries. From a machine that cost 10 million dollars and executed 1 instruction per second, we have come to machines that cost 1000 dollars and execute 10 million instructions per second, a price/performance gain of 10^{11}. If cars had improved at this rate in the same time period, a Rolls Royce would now cost 10 dollars and get a billion miles per gallon. (Unfortunately, it would probably also have a 200-page manual telling how to open the door.)

The second development was the invention of high-speed computer networks. The **local area networks** or **LANs** allow dozens, or even hundreds, of machines within a building to be connected in such a way that small amounts of information can be transferred between machines in a millisecond or so. Larger amounts of data can be moved between machines at rates of 10 to 100 million bits/sec and sometimes more. The **wide area networks** or **WANs** allow millions of machines all over the earth to be connected at speeds varying from 64 Kbps (kilobits per second) to gigabits per second for some advanced experimental networks.

The result of these technologies is that it is now not only feasible, but easy, to put together computing systems composed of large numbers of CPUs connected by a high-speed network. They are usually called **distributed systems,** in contrast to the previous **centralized systems** (or **single-processor systems**) consisting of a single CPU, its memory, peripherals, and some terminals.

There is only one fly in the ointment: software. Distributed systems need radically different software than centralized systems do. In particular, the necessary operating systems are only beginning to emerge. The first few steps have been taken, but there is still a long way to go. Nevertheless, enough is already known about these distributed operating systems that we can present the basic ideas. The rest of this book is devoted to studying concepts, implementation, and examples of distributed operating systems.

1.1. WHAT IS A DISTRIBUTED SYSTEM?

Various definitions of distributed systems have been given in the literature, none of them satisfactory and none of them in agreement with any of the others. For our purposes it is sufficient to give a loose characterization:

> *A distributed system is a collection of independent computers that appear to the users of the system as a single computer.*

This definition has two aspects. The first one deals with hardware: the machines are autonomous. The second one deals with software: the users think of the system as a single computer. Both are essential. We will come back to these points later in this chapter, after going over some background material on both the hardware and the software.

Rather than going further with definitions, it is probably more helpful to give several examples of distributed systems. As a first example, consider a network of workstations in a university or company department. In addition to each user's personal workstation, there might be a pool of processors in the machine room that are not assigned to specific users but are allocated dynamically as needed. Such a system might have a single file system, with all files accessible from all machines in the same way and using the same path name. Furthermore, when a user typed a command, the system could look for the best place to execute that command, possibly on the user's own workstation, possibly on an idle workstation belonging to someone else, and possibly on one of the unassigned processors in the machine room. If the system as a whole looked and acted like a classical single-processor timesharing system, it would qualify as a distributed system.

As a second example, consider a factory full of robots, each containing a powerful computer for handling vision, planning, communication, and other tasks. When a robot on the assembly line notices that a part it is supposed to install is defective, it asks another robot in the parts department to bring it a replacement. If all the robots act like peripheral devices attached to the same central computer and the system can be programmed that way, it too counts as a distributed system.

As a final example, think about a large bank with hundreds of branch offices all over the world. Each office has a master computer to store local accounts and handle local transactions. In addition, each computer has the ability to talk to all other branch computers and with a central computer at headquarters. If transactions can be done without regard to where a customer or account is, and the users do not notice any difference between this system and the old centralized mainframe that it replaced, it too would be considered a distributed system.

1.2. GOALS

Just because it is possible to build distributed systems does not necessarily mean that it is a good idea. After all, with current technology it is possible to put four floppy disk drives on a personal computer. It is just that doing so would be pointless. In this section we will discuss the motivation and goals of typical distributed systems and look at their advantages and disadvantages compared to traditional centralized systems.

1.2.1. Advantages of Distributed Systems over Centralized Systems

The real driving force behind the trend toward decentralization is economics. A quarter of a century ago, computer pundit and gadfly Herb Grosch stated what later came to be known as Grosch's law: The computing power of a CPU is proportional to the square of its price. By paying twice as much, you could get four times the performance. This observation fit the mainframe technology of its time quite well, and led most organizations to buy the largest single machine they could afford.

With microprocessor technology, Grosch's law no longer holds. For a few hundred dollars you can get a CPU chip that can execute more instructions per second than one of the largest 1980s mainframes. If you are willing to pay twice as much, you get the same CPU, but running at a somewhat higher clock speed.

Distributed Operating Systems

Andrew S. Tanenbaum

As distributed computer systems become more pervasive, so does the need for understanding how their operating systems are designed and implemented. Andrew S. Tanenbaum's ***Distributed Operating Systems*** fulfills this need. Representing a revised and greatly expanded Part II of the best-selling *Modern Operating Systems,* it covers the material from the original book, including communication, synchronization, processes, and file systems, and adds new material on distributed shared memory, real-time distributed systems, fault-tolerant distributed systems, and ATM networks. It also contains four detailed case studies: Amoeba, Mach, Chorus, and OSF/DCE.

Tanenbaum's trademark writing provides readers with a thorough, concise treatment of distributed systems.

Also by the author and available from Prentice Hall—

—*Computer Networks, Second Edition*
©1989, ISBN: 0-13-162959-X

—*Modern Operating Systems*
©1992, ISBN: 0-13-588187-0

—*Operating Systems: Design and Implementation*
©1987, ISBN: 0-13-637406-9

—*Structured Computer Organization, Third Edition*
©1990, ISBN: 0-13-854662-6

PRENTICE HALL
Upper Saddle River, N J 07548

ISBN 0-13-219908-4
90000
9 780132 199087