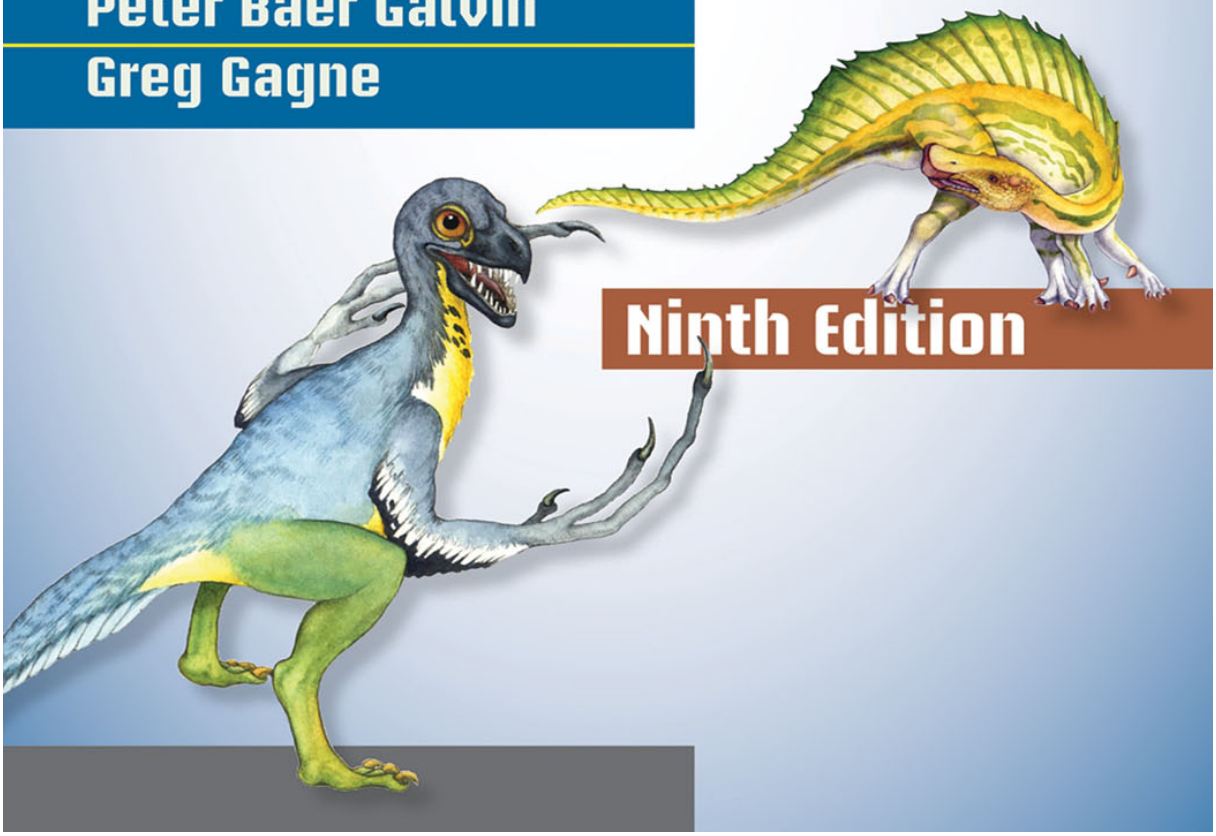
OPERATING SYSTEM CONCEPTS

Abraham Silberschatz
Peter Baer Galvin
Greg Gagne



Ninth Edition

Exhibit 1060
Microsoft Corporation v. Edge Networking Systems, LLC

OPERATING SYSTEM CONCEPTS

ABRAHAM SILBERSCHATZ

Yale University

PETER BAER GALVIN

Pluribus Networks

GREG GAGNE

Westminster College

NINTH EDITION

WILEY

Exhibit 1060

Microsoft Corporation v. Edge Networking Systems, LLC

Vice President and Executive Publisher	Don Fowley
Executive Editor	Beth Lang Golub
Editorial Assistant	Katherine Willis
Executive Marketing Manager	Christopher Ruel
Senior Production Editor	Ken Santor
Cover and title page illustrations	Susan Cyr
Cover Designer	Madelyn Lesure
Text Designer	Judy Allan

This book was set in Palatino by the author using LaTeX and printed and bound by Courier-Kendallville. The cover was printed by Courier.

Copyright © 2013, 2012, 2008 John Wiley & Sons, Inc. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc. 222 Rosewood Drive, Danvers, MA 01923, (978)750-8400, fax (978)750-4470. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030 (201)748-6011, fax (201)748-6008, E-Mail: PERMREQ@WILEY.COM.

Evaluation copies are provided to qualified academics and professionals for review purposes only, for use in their courses during the next academic year. These copies are licensed and may not be sold or transferred to a third party. Upon completion of the review period, please return the evaluation copy to Wiley. Return instructions and a free-of-charge return shipping label are available at www.wiley.com/go/evalreturn. Outside of the United States, please contact your local representative.

Founded in 1807, John Wiley & Sons, Inc. has been a valued source of knowledge and understanding for more than 200 years, helping people around the world meet their needs and fulfill their aspirations. Our company is built on a foundation of principles that include responsibility to the communities we serve and where we live and work. In 2008, we launched a Corporate Citizenship Initiative, a global effort to address the environmental, social, economic, and ethical challenges we face in our business. Among the issues we are addressing are carbon impact, paper specifications and procurement, ethical conduct within our business and among our vendors, and community and charitable support. For more information, please visit our website: www.wiley.com/go/citizenship.

ISBN: 978-1-118-06333-0
ISBN BRV: 978-1-118-12938-8

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Exhibit 1060
Microsoft Corporation v. Edge Networking Systems, LLC

Contents

PART ONE ■ OVERVIEW

Chapter 1 Introduction

1.1 What Operating Systems Do 4
1.2 Computer-System Organization 7
1.3 Computer-System Architecture 12
1.4 Operating-System Structure 19
1.5 Operating-System Operations 21
1.6 Process Management 24
1.7 Memory Management 25
1.8 Storage Management 26
1.9 Protection and Security 30
1.10 Kernel Data Structures 31
1.11 Computing Environments 35
1.12 Open-Source Operating Systems 43
1.13 Summary 47
Exercises 49
Bibliographical Notes 52

Chapter 2 Operating-System Structures

2.1 Operating-System Services 55
2.2 User and Operating-System Interface 58
2.3 System Calls 62
2.4 Types of System Calls 66
2.5 System Programs 74
2.6 Operating-System Design and Implementation 75
2.7 Operating-System Structure 78
2.8 Operating-System Debugging 86
2.9 Operating-System Generation 91
2.10 System Boot 92
2.11 Summary 93
Exercises 94
Bibliographical Notes 101

PART TWO ■ PROCESS MANAGEMENT

Chapter 3 Processes

3.1 Process Concept 105
3.2 Process Scheduling 110
3.3 Operations on Processes 115
3.4 Interprocess Communication 122
3.5 Examples of IPC Systems 130
3.6 Communication in Client–Server Systems 136
3.7 Summary 147
Exercises 149
Bibliographical Notes 161

Exhibit 1060
Microsoft Corporation v. Edge Networking Systems, LLC

Chapter 4 Threads

4.1 Overview 163
4.2 Multicore Programming 166
4.3 Multithreading Models 169
4.4 Thread Libraries 171
4.5 Implicit Threading 177
4.6 Threading Issues 183
4.7 Operating-System Examples 188
4.8 Summary 191
Exercises 191
Bibliographical Notes 199

Chapter 5 Process Synchronization

5.1 Background 203
5.2 The Critical-Section Problem 206
5.3 Peterson's Solution 207
5.4 Synchronization Hardware 209
5.5 Mutex Locks 212
5.6 Semaphores 213
5.7 Classic Problems of Synchronization 219
5.8 Monitors 223
5.9 Synchronization Examples 232
5.10 Alternative Approaches 238
5.11 Summary 242
Exercises 242
Bibliographical Notes 258

Chapter 6 CPU Scheduling

6.1 Basic Concepts 261
6.2 Scheduling Criteria 265
6.3 Scheduling Algorithms 266
6.4 Thread Scheduling 277
6.5 Multiple-Processor Scheduling 278
6.6 Real-Time CPU Scheduling 283
6.7 Operating-System Examples 290
6.8 Algorithm Evaluation 300
6.9 Summary 304
Exercises 305
Bibliographical Notes 311

Chapter 7 Deadlocks

7.1 System Model 315
7.2 Deadlock Characterization 317
7.3 Methods for Handling Deadlocks 322
7.4 Deadlock Prevention 323
7.5 Deadlock Avoidance 327
7.6 Deadlock Detection 333
7.7 Recovery from Deadlock 337
7.8 Summary 339
Exercises 339
Bibliographical Notes 346

PART THREE ■ MEMORY MANAGEMENT

Chapter 8 Main Memory

8.1 Background 351
8.2 Swapping 358
8.3 Contiguous Memory Allocation 360
8.4 Segmentation 364
8.5 Paging 366
8.6 Structure of the Page Table 378
8.7 Example: Intel 32 and 64-bit Architectures 383
8.8 Example: ARM Architecture 388
8.9 Summary 389
Exercises 390
Bibliographical Notes 394

Exhibit 1060
Microsoft Corporation v. Edge Networking Systems, LLC

Chapter 9 Virtual Memory

9.1 Background 397
9.2 Demand Paging 401
9.3 Copy-on-Write 408
9.4 Page Replacement 409
9.5 Allocation of Frames 421
9.6 Thrashing 425
9.7 Memory-Mapped Files 430
9.8 Allocating Kernel Memory 436
9.9 Other Considerations 439
9.10 Operating-System Examples 445
9.11 Summary 448
Exercises 449
Bibliographical Notes 461

PART FOUR ■ STORAGE MANAGEMENT

Chapter 10 Mass-Storage Structure

10.1 Overview of Mass-Storage Structure 467
10.2 Disk Structure 470
10.3 Disk Attachment 471
10.4 Disk Scheduling 472
10.5 Disk Management 478
10.6 Swap-Space Management 482
10.7 RAID Structure 484
10.8 Stable-Storage Implementation 494
10.9 Summary 496
Exercises 497
Bibliographical Notes 501

Chapter 11 File-System Interface

11.1 File Concept 503
11.2 Access Methods 513
11.3 Directory and Disk Structure 515
11.4 File-System Mounting 526
11.5 File Sharing 528
11.6 Protection 533
11.7 Summary 538
Exercises 539
Bibliographical Notes 541

Chapter 12 File-System Implementation

12.1 File-System Structure 543
12.2 File-System Implementation 546
12.3 Directory Implementation 552
12.4 Allocation Methods 553
12.5 Free-Space Management 561
12.6 Efficiency and Performance 564
12.7 Recovery 568
12.8 NFS 571
12.9 Example: The WAFL File System 577
12.10 Summary 580
Exercises 581
Bibliographical Notes 585

Chapter 13 I/O Systems

13.1 Overview 587
13.2 I/O Hardware 588
13.3 Application I/O Interface 597
13.4 Kernel I/O Subsystem 604
13.5 Transforming I/O Requests to Hardware Operations 611
13.6 STREAMS 613
13.7 Performance 615
13.8 Summary 618
Exercises 619
Bibliographical Notes 621

Exhibit 1060
Microsoft Corporation v. Edge Networking Systems, LLC

PART FIVE ■ PROTECTION AND SECURITY

Chapter 14 Protection

14.1 Goals of Protection 625
14.2 Principles of Protection 626
14.3 Domain of Protection 627
14.4 Access Matrix 632
14.5 Implementation of the Access Matrix 636
14.6 Access Control 639
14.7 Revocation of Access Rights 640
14.8 Capability-Based Systems 641
14.9 Language-Based Protection 644
14.10 Summary 649
Exercises 650
Bibliographical Notes 652

Chapter 15 Security

15.1 The Security Problem 657
15.2 Program Threats 661
15.3 System and Network Threats 669
15.4 Cryptography as a Security Tool 674
15.5 User Authentication 685
15.6 Implementing Security Defenses 689
15.7 Firewalling to Protect Systems and Networks 696
15.8 Computer-Security Classifications 698
15.9 An Example: Windows 7 699
15.10 Summary 701
Exercises 702
Bibliographical Notes 704

PART SIX ■ ADVANCED TOPICS

Chapter 16 Virtual Machines

16.1 Overview 711
16.2 History 713
16.3 Benefits and Features 714
16.4 Building Blocks 717
16.5 Types of Virtual Machines and Their Implementations 721
16.6 Virtualization and Operating-System Components 728
16.7 Examples 735
16.8 Summary 737
Exercises 738
Bibliographical Notes 739

Chapter 17 Distributed Systems

17.1 Advantages of Distributed Systems 741
17.2 Types of Network-based Operating Systems 743
17.3 Network Structure 747
17.4 Communication Structure 751
17.5 Communication Protocols 756
17.6 An Example: TCP/IP 760
17.7 Robustness 762
17.8 Design Issues 764
17.9 Distributed File Systems 765
17.10 Summary 773
Exercises 774
Bibliographical Notes 777

Exhibit 1060
Microsoft Corporation v. Edge Networking Systems, LLC

PART SEVEN ■ CASE STUDIES

Chapter 18 The Linux System

18.1 Linux History 781
18.2 Design Principles 786
18.3 Kernel Modules 789
18.4 Process Management 792
18.5 Scheduling 795
18.6 Memory Management 800
18.7 File Systems 809
18.8 Input and Output 815
18.9 Interprocess Communication 818
18.10 Network Structure 819
18.11 Security 821
18.12 Summary 824
Exercises 824
Bibliographical Notes 826

Chapter 19 Windows 7

19.1 History 829
19.2 Design Principles 831
19.3 System Components 838
19.4 Terminal Services and Fast User Switching 862
19.5 File System 863
19.6 Networking 869
19.7 Programmer Interface 874
19.8 Summary 883
Exercises 883
Bibliographical Notes 885

Chapter 20 Influential Operating Systems

20.1 Feature Migration 887
20.2 Early Systems 888
20.3 Atlas 895
20.4 XDS-940 896
20.5 THE 897
20.6 RC 4000 897
20.7 CTSS 898
20.8 MULTICS 899
20.9 IBM OS/360 899
20.10 TOPS-20 901
20.11 CP/M and MS/DOS 901
20.12 Macintosh Operating System and Windows 902
20.13 Mach 902
20.14 Other Systems 904
Exercises 904
Bibliographical Notes 904

PART EIGHT ■ APPENDICES

Appendix A BSD UNIX

A.1 UNIX History A1
A.2 Design Principles A6
A.3 Programmer Interface A8
A.4 User Interface A15
A.5 Process Management A18
A.6 Memory Management A22
A.7 File System A24
A.8 I/O System A32
A.9 Interprocess Communication A36
A.10 Summary A40
Exercises A41
Bibliographical Notes A42

Exhibit 1060
Microsoft Corporation v. Edge Networking Systems, LLC

Appendix B The Mach System

B.1 History of the Mach System B1
B.2 Design Principles B3
B.3 System Components B4
B.4 Process Management B7
B.5 Interprocess Communication B13
B.6 Memory Management B18
B.7 Programmer Interface B23
B.8 Summary B24
Exercises B25
Bibliographical Notes B26

Exhibit 1060
Microsoft Corporation v. Edge Networking Systems, LLC

timer and scheduling algorithms to cycle processes rapidly through the CPU, giving each user a share of the resources.

Today, traditional time-sharing systems are uncommon. The same scheduling technique is still in use on desktop computers, laptops, servers, and even mobile computers, but frequently all the processes are owned by the same user (or a single user and the operating system). User processes, and system processes that provide services to the user, are managed so that each frequently gets a slice of computer time. Consider the windows created while a user is working on a PC, for example, and the fact that they may be performing different tasks at the same time. Even a web browser can be composed of multiple processes, one for each website currently being visited, with time sharing applied to each web browser process.

1.11.2 Mobile Computing

Mobile computing refers to computing on handheld smartphones and tablet computers. These devices share the distinguishing physical features of being portable and lightweight. Historically, compared with desktop and laptop computers, mobile systems gave up screen size, memory capacity, and overall functionality in return for handheld mobile access to services such as e-mail and web browsing. Over the past few years, however, features on mobile devices have become so rich that the distinction in functionality between, say, a consumer laptop and a tablet computer may be difficult to discern. In fact, we might argue that the features of a contemporary mobile device allow it to provide functionality that is either unavailable or impractical on a desktop or laptop computer.

Today, mobile systems are used not only for e-mail and web browsing but also for playing music and video, reading digital books, taking photos, and recording high-definition video. Accordingly, tremendous growth continues in the wide range of applications that run on such devices. Many developers are now designing applications that take advantage of the unique features of mobile devices, such as global positioning system (GPS) chips, accelerometers, and gyroscopes. An embedded GPS chip allows a mobile device to use satellites to determine its precise location on earth. That functionality is especially useful in designing applications that provide navigation—for example, telling users which way to walk or drive or perhaps directing them to nearby services, such as restaurants. An accelerometer allows a mobile device to detect its orientation with respect to the ground and to detect certain other forces, such as tilting and shaking. In several computer games that employ accelerometers, players interface with the system not by using a mouse or a keyboard but rather by tilting, rotating, and shaking the mobile device! Perhaps more a practical use of these features is found in ***augmented-reality*** applications, which overlay information on a display of the current environment. It is difficult to imagine how equivalent applications could be developed on traditional laptop or desktop computer systems.

To provide access to on-line services, mobile devices typically use either IEEE standard 802.11 wireless or cellular data networks. The memory capacity and processing speed of mobile devices, however, are more limited than those of PCs. Whereas a smartphone or tablet may have 64 GB in storage, it is not uncommon to find 1 TB in storage on a desktop computer. Similarly, because

Exhibit 1060
Microsoft Corporation v. Edge Networking Systems, LLC

power consumption is such a concern, mobile devices often use processors that are smaller, are slower, and offer fewer processing cores than processors found on traditional desktop and laptop computers.

Two operating systems currently dominate mobile computing: **Apple iOS** and **Google Android**. iOS was designed to run on Apple iPhone and iPad mobile devices. Android powers smartphones and tablet computers available from many manufacturers. We examine these two mobile operating systems in further detail in Chapter 2.

1.11.3 Distributed Systems

A distributed system is a collection of physically separate, possibly heterogeneous, computer systems that are networked to provide users with access to the various resources that the system maintains. Access to a shared resource increases computation speed, functionality, data availability, and reliability. Some operating systems generalize network access as a form of file access, with the details of networking contained in the network interface's device driver. Others make users specifically invoke network functions. Generally, systems contain a mix of the two modes—for example FTP and NFS. The protocols that create a distributed system can greatly affect that system's utility and popularity.

A **network**, in the simplest terms, is a communication path between two or more systems. Distributed systems depend on networking for their functionality. Networks vary by the protocols used, the distances between nodes, and the transport media. **TCP/IP** is the most common network protocol, and it provides the fundamental architecture of the Internet. Most operating systems support TCP/IP, including all general-purpose ones. Some systems support proprietary protocols to suit their needs. To an operating system, a network protocol simply needs an interface device—a network adapter, for example—with a device driver to manage it, as well as software to handle data. These concepts are discussed throughout this book.

Networks are characterized based on the distances between their nodes. A **local-area network (LAN)** connects computers within a room, a building, or a campus. A **wide-area network (WAN)** usually links buildings, cities, or countries. A global company may have a WAN to connect its offices worldwide, for example. These networks may run one protocol or several protocols. The continuing advent of new technologies brings about new forms of networks. For example, a **metropolitan-area network (MAN)** could link buildings within a city. BlueTooth and 802.11 devices use wireless technology to communicate over a distance of several feet, in essence creating a **personal-area network (PAN)** between a phone and a headset or a smartphone and a desktop computer.

The media to carry networks are equally varied. They include copper wires, fiber strands, and wireless transmissions between satellites, microwave dishes, and radios. When computing devices are connected to cellular phones, they create a network. Even very short-range infrared communication can be used for networking. At a rudimentary level, whenever computers communicate, they use or create a network. These networks also vary in their performance and reliability.

Some operating systems have taken the concept of networks and distributed systems further than the notion of providing network connectivity.

Exhibit 1060
Microsoft Corporation v. Edge Networking Systems, LLC

A **network operating system** is an operating system that provides features such as file sharing across the network, along with a communication scheme that allows different processes on different computers to exchange messages. A computer running a network operating system acts autonomously from all other computers on the network, although it is aware of the network and is able to communicate with other networked computers. A distributed operating system provides a less autonomous environment. The different computers communicate closely enough to provide the illusion that only a single operating system controls the network. We cover computer networks and distributed systems in Chapter 17.

1.11.4 Client–Server Computing

As PCs have become faster, more powerful, and cheaper, designers have shifted away from centralized system architecture. Terminals connected to centralized systems are now being supplanted by PCs and mobile devices. Correspondingly, user-interface functionality once handled directly by centralized systems is increasingly being handled by PCs, quite often through a web interface. As a result, many of today's systems act as **server systems** to satisfy requests generated by **client systems**. This form of specialized distributed system, called a **client–server** system, has the general structure depicted in Figure 1.18.

Server systems can be broadly categorized as compute servers and file servers:

- The **compute-server system** provides an interface to which a client can send a request to perform an action (for example, read data). In response, the server executes the action and sends the results to the client. A server running a database that responds to client requests for data is an example of such a system.
- The **file-server system** provides a file-system interface where clients can create, update, read, and delete files. An example of such a system is a web server that delivers files to clients running web browsers.

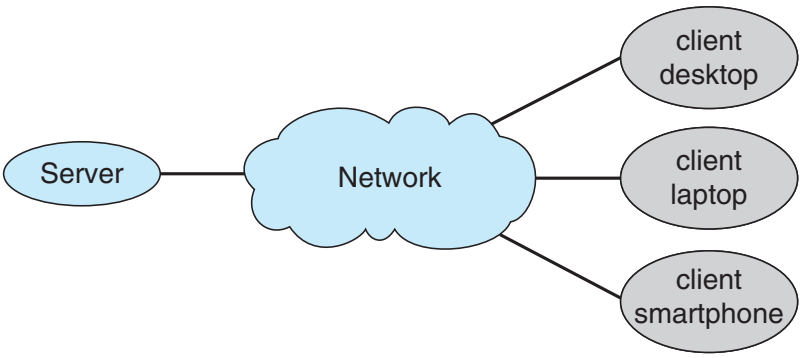


Figure 1.18 General structure of a client–server system.

Exhibit 1060
Microsoft Corporation v. Edge Networking Systems, LLC

1.11.5 Peer-to-Peer Computing

Another structure for a distributed system is the peer-to-peer (P2P) system model. In this model, clients and servers are not distinguished from one another. Instead, all nodes within the system are considered peers, and each may act as either a client or a server, depending on whether it is requesting or providing a service. Peer-to-peer systems offer an advantage over traditional client-server systems. In a client-server system, the server is a bottleneck; but in a peer-to-peer system, services can be provided by several nodes distributed throughout the network.

To participate in a peer-to-peer system, a node must first join the network of peers. Once a node has joined the network, it can begin providing services to—and requesting services from—other nodes in the network. Determining what services are available is accomplished in one of two general ways:

- When a node joins a network, it registers its service with a centralized lookup service on the network. Any node desiring a specific service first contacts this centralized lookup service to determine which node provides the service. The remainder of the communication takes place between the client and the service provider.
- An alternative scheme uses no centralized lookup service. Instead, a peer acting as a client must discover what node provides a desired service by broadcasting a request for the service to all other nodes in the network. The node (or nodes) providing that service responds to the peer making the request. To support this approach, a ***discovery protocol*** must be provided that allows peers to discover services provided by other peers in the network. Figure 1.19 illustrates such a scenario.

Peer-to-peer networks gained widespread popularity in the late 1990s with several file-sharing services, such as Napster and Gnutella, that enabled peers to exchange files with one another. The Napster system used an approach similar to the first type described above: a centralized server maintained an index of all files stored on peer nodes in the Napster network, and the actual

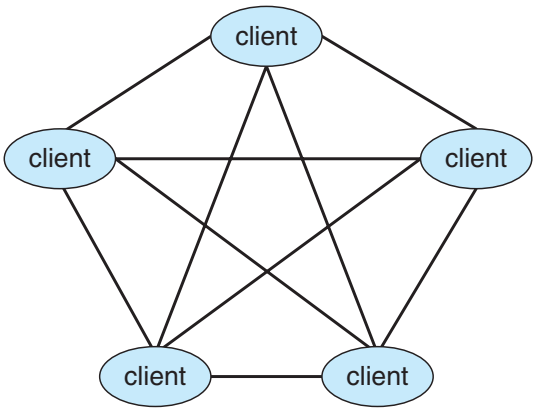


Figure 1.19 Peer-to-peer system with no centralized service.

Exhibit 1060
Microsoft Corporation v. Edge Networking Systems, LLC

exchange of files took place between the peer nodes. The Gnutella system used a technique similar to the second type: a client broadcasted file requests to other nodes in the system, and nodes that could service the request responded directly to the client. The future of exchanging files remains uncertain because peer-to-peer networks can be used to exchange copyrighted materials (music, for example) anonymously, and there are laws governing the distribution of copyrighted material. Notably, Napster ran into legal trouble for copyright infringement and its services were shut down in 2001.

Skype is another example of peer-to-peer computing. It allows clients to make voice calls and video calls and to send text messages over the Internet using a technology known as **voice over IP** (**VoIP**). Skype uses a hybrid peer-to-peer approach. It includes a centralized login server, but it also incorporates decentralized peers and allows two peers to communicate.

1.11.6 Virtualization

Virtualization is a technology that allows operating systems to run as applications within other operating systems. At first blush, there seems to be little reason for such functionality. But the virtualization industry is vast and growing, which is a testament to its utility and importance.

Broadly speaking, virtualization is one member of a class of software that also includes emulation. **Emulation** is used when the source CPU type is different from the target CPU type. For example, when Apple switched from the IBM Power CPU to the Intel x86 CPU for its desktop and laptop computers, it included an emulation facility called "Rosetta," which allowed applications compiled for the IBM CPU to run on the Intel CPU. That same concept can be extended to allow an entire operating system written for one platform to run on another. Emulation comes at a heavy price, however. Every machine-level instruction that runs natively on the source system must be translated to the equivalent function on the target system, frequently resulting in several target instructions. If the source and target CPUs have similar performance levels, the emulated code can run much slower than the native code.

A common example of emulation occurs when a computer language is not compiled to native code but instead is either executed in its high-level form or translated to an intermediate form. This is known as **interpretation**. Some languages, such as BASIC, can be either compiled or interpreted. Java, in contrast, is always interpreted. Interpretation is a form of emulation in that the high-level language code is translated to native CPU instructions, emulating not another CPU but a theoretical virtual machine on which that language could run natively. Thus, we can run Java programs on "Java virtual machines," but technically those virtual machines are Java emulators.

With **virtualization**, in contrast, an operating system that is natively compiled for a particular CPU architecture runs within another operating system also native to that CPU. Virtualization first came about on IBM mainframes as a method for multiple users to run tasks concurrently. Running multiple virtual machines allowed (and still allows) many users to run tasks on a system designed for a single user. Later, in response to problems with running multiple Microsoft Windows XP applications on the Intel x86 CPU, VMware created a new virtualization technology in the form of an application that ran on XP. That application ran one or more **guest** copies of Windows or other native

Exhibit 1060
Microsoft Corporation v. Edge Networking Systems, LLC

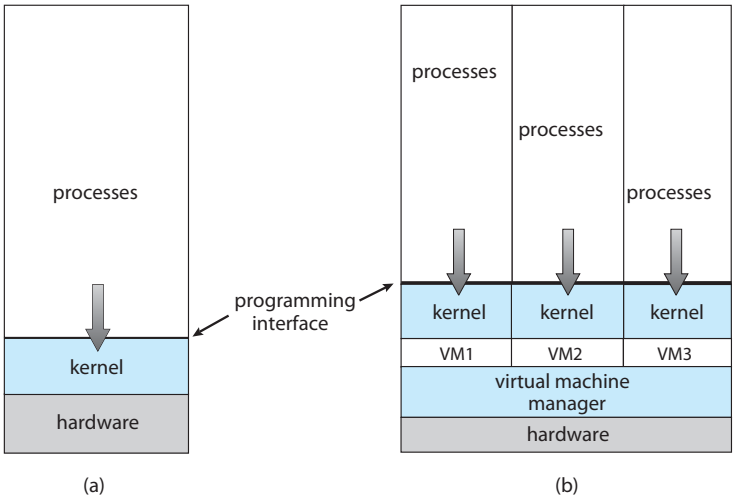


Figure 1.20 VMware.

x86 operating systems, each running its own applications. (See Figure 1.20.) Windows was the **host** operating system, and the VMware application was the virtual machine manager VMM. The VMM runs the guest operating systems, manages their resource use, and protects each guest from the others.

Even though modern operating systems are fully capable of running multiple applications reliably, the use of virtualization continues to grow. On laptops and desktops, a VMM allows the user to install multiple operating systems for exploration or to run applications written for operating systems other than the native host. For example, an Apple laptop running Mac OS X on the x86 CPU can run a Windows guest to allow execution of Windows applications. Companies writing software for multiple operating systems can use virtualization to run all of those operating systems on a single physical server for development, testing, and debugging. Within data centers, virtualization has become a common method of executing and managing computing environments. VMMs like VMware, ESX, and Citrix XenServer no longer run on host operating systems but rather ***are*** the hosts. Full details of the features and implementation of virtualization are found in Chapter 16.

1.11.7 Cloud Computing

Cloud computing is a type of computing that delivers computing, storage, and even applications as a service across a network. In some ways, it's a logical extension of virtualization, because it uses virtualization as a base for its functionality. For example, the Amazon Elastic Compute Cloud (**EC2**) facility has thousands of servers, millions of virtual machines, and petabytes of storage available for use by anyone on the Internet. Users pay per month based on how much of those resources they use.

There are actually many types of cloud computing, including the following:

- **Public cloud**—a cloud available via the Internet to anyone willing to pay for the services

Exhibit 1060
Microsoft Corporation v. Edge Networking Systems, LLC

- **Private cloud**—a cloud run by a company for that company's own use
- **Hybrid cloud**—a cloud that includes both public and private cloud components
- Software as a service **(SaaS)**—one or more applications (such as word processors or spreadsheets) available via the Internet
- Platform as a service **(PaaS)**—a software stack ready for application use via the Internet (for example, a database server)
- Infrastructure as a service **(IaaS)**—servers or storage available over the Internet (for example, storage available for making backup copies of production data)

These cloud-computing types are not discrete, as a cloud computing environment may provide a combination of several types. For example, an organization may provide both SaaS and IaaS as a publicly available service.

Certainly, there are traditional operating systems within many of the types of cloud infrastructure. Beyond those are the VMMs that manage the virtual machines in which the user processes run. At a higher level, the VMMs themselves are managed by cloud management tools, such as Vware vCloud Director and the open-source Eucalyptus toolset. These tools manage the resources within a given cloud and provide interfaces to the cloud components, making a good argument for considering them a new type of operating system.

Figure 1.21 illustrates a public cloud providing IaaS. Notice that both the cloud services and the cloud user interface are protected by a firewall.

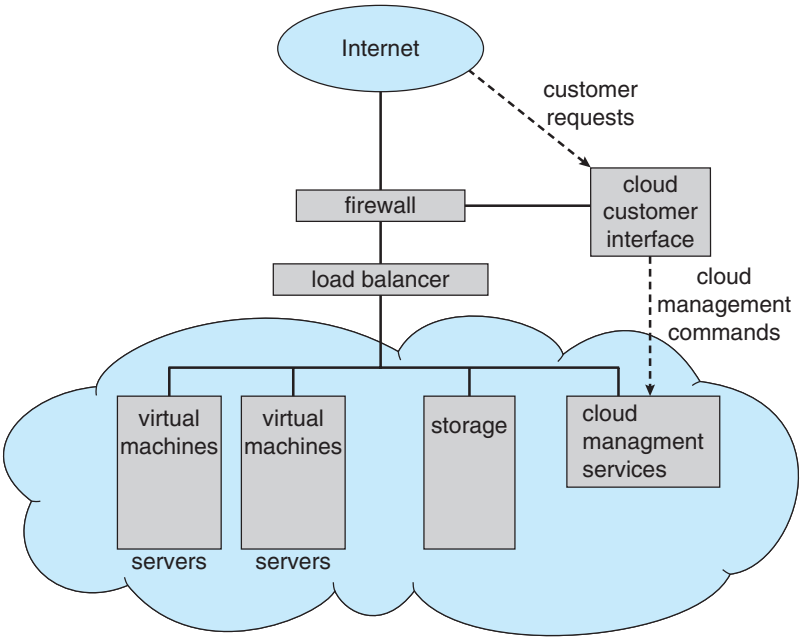


Figure 1.21 Cloud computing.

Exhibit 1060
Microsoft Corporation v. Edge Networking Systems, LLC

1.11.8 Real-Time Embedded Systems

Embedded computers are the most prevalent form of computers in existence. These devices are found everywhere, from car engines and manufacturing robots to DVDs and microwave ovens. They tend to have very specific tasks. The systems they run on are usually primitive, and so the operating systems provide limited features. Usually, they have little or no user interface, preferring to spend their time monitoring and managing hardware devices, such as automobile engines and robotic arms.

These embedded systems vary considerably. Some are general-purpose computers, running standard operating systems—such as Linux—with special-purpose applications to implement the functionality. Others are hardware devices with a special-purpose embedded operating system providing just the functionality desired. Yet others are hardware devices with application-specific integrated circuits (**ASICs**) that perform their tasks without an operating system.

The use of embedded systems continues to expand. The power of these devices, both as standalone units and as elements of networks and the web, is sure to increase as well. Even now, entire houses can be computerized, so that a central computer—either a general-purpose computer or an embedded system—can control heating and lighting, alarm systems, and even coffee makers. Web access can enable a home owner to tell the house to heat up before she arrives home. Someday, the refrigerator can notify the grocery store when it notices the milk is gone.

Embedded systems almost always run **real-time operating systems**. A real-time system is used when rigid time requirements have been placed on the operation of a processor or the flow of data; thus, it is often used as a control device in a dedicated application. Sensors bring data to the computer. The computer must analyze the data and possibly adjust controls to modify the sensor inputs. Systems that control scientific experiments, medical imaging systems, industrial control systems, and certain display systems are real-time systems. Some automobile-engine fuel-injection systems, home-appliance controllers, and weapon systems are also real-time systems.

A real-time system has well-defined, fixed time constraints. Processing ***must*** be done within the defined constraints, or the system will fail. For instance, it would not do for a robot arm to be instructed to halt ***after*** it had smashed into the car it was building. A real-time system functions correctly only if it returns the correct result within its time constraints. Contrast this system with a time-sharing system, where it is desirable (but not mandatory) to respond quickly, or a batch system, which may have no time constraints at all.

In Chapter 6, we consider the scheduling facility needed to implement real-time functionality in an operating system. In Chapter 9, we describe the design of memory management for real-time computing. Finally, in Chapters 18 and 19, we describe the real-time components of the Linux and Windows 7 operating systems.

1.12 Open-Source Operating Systems

We noted at the beginning of this chapter that the study of operating systems has been made easier by the availability of a vast number of open-source

Exhibit 1060
Microsoft Corporation v. Edge Networking Systems, LLC