DISTRIBUTED SYSTEMS

Concepts and Design

Fifth Edition



International Edition

George Coulouris
Jean Dollimore
Tim Kindberg
Gordon Blair





PEARSON

DISTRIBUTED SYSTEMS

Concepts and Design

Fifth Edition

This page intentionally left blank

DISTRIBUTED SYSTEMS

Concepts and Design

Fifth Edition

George Coulouris
Cambridge University

Jean Dollimore
formerly of Queen Mary, University of London

Tim Kindberg
matter 2 media

Gordon Blair
Lancaster University

International Edition contributions by:
Arup Kumar Bhattacharjee
RCC Institute of Information Technology, Kolkata

Soumen Mukherjee
RCC Institute of Information Technology, Kolkata

Addison-Wesley

Boston Columbus Indianapolis New York San Francisco Upper Saddle River
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montréal Toronto
Delhi Mexico City São Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

Editorial Director: Marcia Horton
Editor-in-Chief: Michael Hirsch
Executive Editor: Matt Goldstein
Editorial Assistant: Chelsea Bell
Vice President, Marketing: Patrice Jones
Marketing Manager: Yezan Alayan
Marketing Coordinator: Kathryn Ferranti
Vice President, Production: Vince O'Brien
Managing Editor: Jeff Holcomb
Senior Production Project Manager: Marilyn Lloyd
Publisher, International Edition: Angshuman Chakraborty
Acquisitions Editor, International Edition: Arunabha Deb
Publishing Assistant, International Edition: Shokhi Shah
Senior Operations Supervisor: Alan Fischer
Manufacturing Buyer: Lisa McDowell
Art Director: Jayne Conte
Cover Designer: Suzanne Duda
Cover Image: Sky: © amygdala_imagery; Kite: © Alamy; Mobile phone: © yasinguneysu/iStock
Media Editor: Daniel Sandin
Media Project Manager: Wanda Rockwell
Cover Printer: Lehigh-Phoenix Color

Pearson Education Limited
Edinburgh Gate
Harlow
Essex CM20 2JE
England

and Associated Companies throughout the world

Visit us on the World Wide Web at:
www.pearsoninternationaleditions.com

© Pearson Education Limited 2012

The rights of George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair to be identified as authors of this work have been asserted by them in accordance with the Copyright, Designs and Patents Act 1988.

Authorized adaptation from the United States edition, entitled Distributed Systems, Concepts and Design, 5th edition*, ISBN 978-0-13-214301-1 by George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair published by Pearson Education © 2012.*

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 or otherwise, without either the prior written permission of the publisher or a licence permitting restricted copying in the United Kingdom issued by the Copyright Licensing Agency Ltd, Saffron House, 6–10 Kirby Street, London EC1N 8TS.

All trademarks used herein are the property of their respective owners. The use of any trademark in this text does not vest in the author or publisher any trademark ownership rights in such trademarks, nor does the use of such trademarks imply any affiliation with or endorsement of this book by such owners.

Microsoft® and Windows® are registered trademarks of the Microsoft Corporation in the U.S.A. and other countries. Screen shots and icons reprinted with permission from the Microsoft Corporation. This book is not sponsored or endorsed by or affiliated with the Microsoft Corporation.

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

10 9 8 7 6 5 4 3 2 1
14 13 12 11 10

Typeset in Times New Roman by authors using FrameMaker
Printed and bound by Edwards Brothers in The United States of America

The publisher's policy is to use paper manufactured from sustainable forests.

ISBN 10: 0-273-76059-9
ISBN 13: 978-0-273-76059-7

Addison-Wesley
is an imprint of

PEARSON

www.pearsonhighered.com

CONTENTS

PREFACE 11

1 CHARACTERIZATION OF DISTRIBUTED SYSTEMS 17
1.1 Introduction 18
1.2 Examples of distributed systems 19
1.3 Trends in distributed systems 24
1.4 Focus on resource sharing 30
1.5 Challenges 32
1.6 Case study: The World Wide Web 42
1.7 Summary 49

2 SYSTEM MODELS 53
2.1 Introduction 54
2.2 Physical models 55
2.3 Architectural models 56
2.4 Fundamental models 77
2.5 Summary 92

3 NETWORKING AND INTERNETWORKING 97
3.1 Introduction 98
3.2 Types of network 102
3.3 Network principles 105
3.4 Internet protocols 122
3.5 Case studies: Ethernet, WiFi and Bluetooth 144
3.6 Summary 157

4 INTERPROCESS COMMUNICATION 161
4.1 Introduction 162
4.2 The API for the Internet protocols 163
4.3 External data representation and marshalling 174
4.4 Multicast communication 185
4.5 Network virtualization: Overlay networks 190
4.6 Case study: MPI 194
4.7 Summary 197

5 REMOTE INVOCATION 201
5.1 Introduction 202
5.2 Request-reply protocols 203
5.3 Remote procedure call 211
5.4 Remote method invocation 220
5.5 Case study: Java RMI 233
5.6 Summary 241

6 INDIRECT COMMUNICATION 245
6.1 Introduction 246
6.2 Group communication 248
6.3 Publish-subscribe systems 258
6.4 Message queues 270
6.5 Shared memory approaches 278
6.6 Summary 290

7 OPERATING SYSTEM SUPPORT 295
7.1 Introduction 296
7.2 The operating system layer 297
7.3 Protection 300
7.4 Processes and threads 302
7.5 Communication and invocation 319
7.6 Operating system architecture 330
7.7 Virtualization at the operating system level 334
7.8 Summary 347

8 DISTRIBUTED OBJECTS AND COMPONENTS 351
8.1 Introduction 352
8.2 Distributed objects 353
8.3 Case study: CORBA 356
8.4 From objects to components 374
8.5 Case studies: Enterprise JavaBeans and Fractal 380
8.6 Summary 394

9 WEB SERVICES 397
9.1 Introduction 398
9.2 Web services 400
9.3 Service descriptions and IDL for web services 416
9.4 A directory service for use with web services 420
9.5 XML security 422
9.6 Coordination of web services 427
9.7 Applications of web services 429
9.8 Summary 435

10 PEER-TO-PEER SYSTEMS 439
10.1 Introduction 440
10.2 Napster and its legacy 444
10.3 Peer-to-peer middleware 446
10.4 Routing overlays 449
10.5 Overlay case studies: Pastry, Tapestry 452
10.6 Application case studies: Squirrel, OceanStore, Ivy 465
10.7 Summary 474

11 SECURITY 479
11.1 Introduction 480
11.2 Overview of security techniques 488
11.3 Cryptographic algorithms 500
11.4 Digital signatures 509
11.5 Cryptography pragmatics 516
11.6 Case studies: Needham–Schroeder, Kerberos, TLS, 802.11 WiFi 519
11.7 Summary 534

12 DISTRIBUTED FILE SYSTEMS 537
12.1 Introduction 538
12.2 File service architecture 546
12.3 Case study: Sun Network File System 552
12.4 Case study: The Andrew File System 564
12.5 Enhancements and further developments 573
12.6 Summary 579

13 NAME SERVICES 581
13.1 Introduction 582
13.2 Name services and the Domain Name System 585
13.3 Directory services 600
13.4 Case study: The Global Name Service 601
13.5 Case study: The X.500 Directory Service 604
13.6 Summary 608

14 TIME AND GLOBAL STATES 611
14.1 Introduction 612
14.2 Clocks, events and process states 613
14.3 Synchronizing physical clocks 615
14.4 Logical time and logical clocks 623
14.5 Global states 626
14.6 Distributed debugging 635
14.7 Summary 642

15 COORDINATION AND AGREEMENT 645
15.1 Introduction 646
15.2 Distributed mutual exclusion 649
15.3 Elections 657
15.4 Coordination and agreement in group communication 662
15.5 Consensus and related problems 675
15.6 Summary 687

16 TRANSACTIONS AND CONCURRENCY CONTROL 691
16.1 Introduction 692
16.2 Transactions 695
16.3 Nested transactions 706
16.4 Locks 708
16.5 Optimistic concurrency control 723
16.6 Timestamp ordering 727
16.7 Comparison of methods for concurrency control 734
16.8 Summary 736

17 DISTRIBUTED TRANSACTIONS 743
17.1 Introduction 744
17.2 Flat and nested distributed transactions 744
17.3 Atomic commit protocols 747
17.4 Concurrency control in distributed transactions 756
17.5 Distributed deadlocks 759
17.6 Transaction recovery 767
17.7 Summary 777

18 REPLICATION 781
18.1 Introduction 782
18.2 System model and the role of group communication 784
18.3 Fault-tolerant services 791
18.4 Case studies of highly available services:
The gossip architecture, Bayou and Coda 798
18.5 Transactions with replicated data 818
18.6 Summary 830

19 MOBILE AND UBIQUITOUS COMPUTING 833
19.1 Introduction 834
19.2 Association 843
19.3 Interoperation 851
19.4 Sensing and context awareness 860
19.5 Security and privacy 873
19.6 Adaptation 882
19.7 Case study: Cooltown 887
19.8 Summary 894

20 DISTRIBUTED MULTIMEDIA SYSTEMS 897
20.1 Introduction 898
20.2 Characteristics of multimedia data 902
20.3 Quality of service management 903
20.4 Resource management 913
20.5 Stream adaptation 915
20.6 Case studies: Tiger, BitTorrent and End System Multicast 917
20.7 Summary 929

21 DESIGNING DISTRIBUTED SYSTEMS: GOOGLE CASE STUDY 931
21.1 Introduction 932
21.2 Introducing the case study: Google 933
21.3 Overall architecture and design philosophy 938
21.4 Underlying communication paradigms 944
21.5 Data storage and coordination services 951
21.6 Distributed computation services 972
21.7 Summary 980

REFERENCES 983

INDEX 1041

PREFACE

This fifth edition of our textbook appears at a time when the Internet and the Web continue to grow and have an impact on every aspect of our society. For example, the introductory chapter of the book notes their impact on application areas as diverse as finance and commerce, arts and entertainment and the emergence of the information society more generally. It also highlights the very demanding requirements of application domains such as web search and multiplayer online games. From a distributed systems perspective, these developments are placing substantial new demands on the underlying system infrastructure in terms of the range of applications and the workloads and system sizes supported by many modern systems. Important trends include the increasing diversity and ubiquity of networking technologies (including the increasing importance of wireless networks), the inherent integration of mobile and ubiquitous computing elements into the distributed systems infrastructure

New to the fifth edition

New chapters:

Indirect Communication: Covering group communication, publish-subscribe and case studies on JavaSpaces, JMS, WebSphere and Message Queues.

Distributed Objects and Components: Covering component-based middleware and case studies on Enterprise JavaBeans, Fractal and CORBA.

Designing Distributed Systems: Devoted to a major new case study on the Google infrastructure.

Topics added to other chapters: Cloud computing, network virtualization, operating system virtualization, message passing interface, unstructured peer-to-peer, tuple spaces, loose coupling in relation to web services.

Other new case studies: Skype, Gnutella, TOTA, L^{2}imbo, BitTorrent, End System Multicast.

See the table on page 15 for further details of the changes.

(leading to radically different physical architectures), the need to support multimedia services and the emergence of the cloud computing paradigm, which challenges our perspective of distributed systems services.

The book aims to provide an understanding of the principles on which the Internet and other distributed systems are based; their architecture, algorithms and design; and how they meet the demands of contemporary distributed applications. We begin with a set of seven chapters that together cover the building blocks for a study of distributed systems. The first two chapters provide a conceptual overview of the subject, outlining the characteristics of distributed systems and the challenges that must be addressed in their design: scalability, heterogeneity, security and failure handling being the most significant. These chapters also develop abstract models for understanding process interaction, failure and security. They are followed by other foundational chapters devoted to the study of networking, interprocess communication, remote invocation, indirect communication and operating system support.

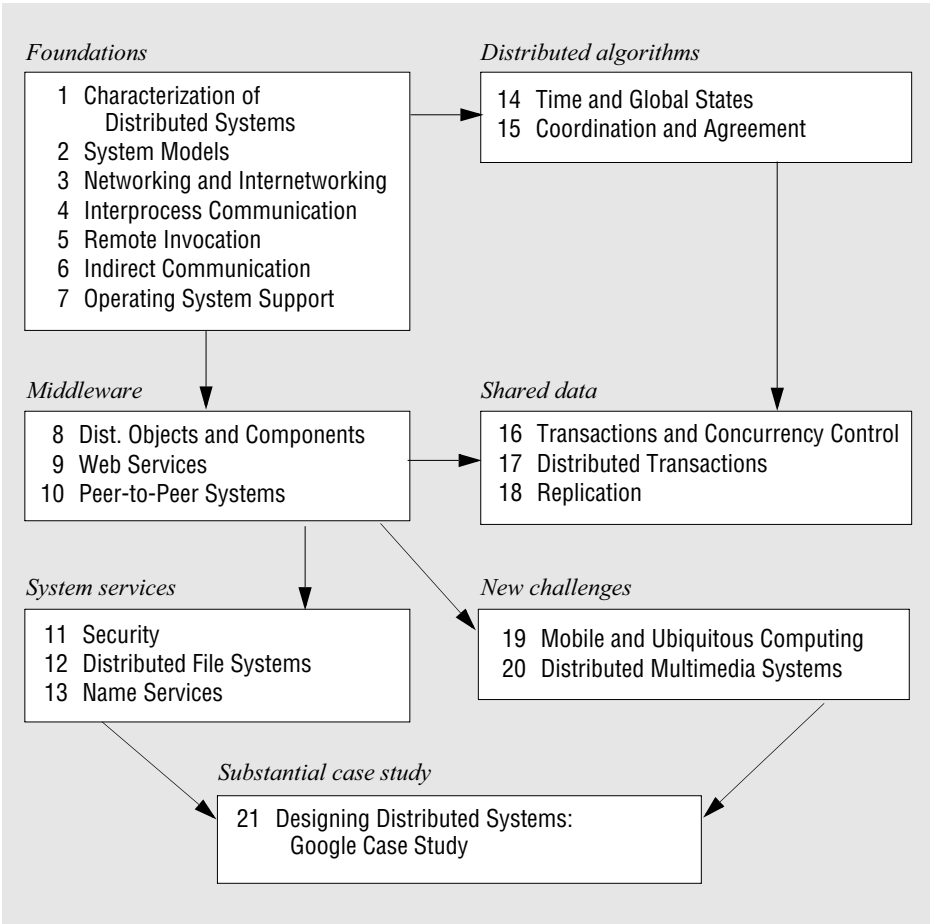
The next set of chapters covers the important topic of middleware, examining different approaches to supporting distributed applications including distributed objects and components, web services and alternative peer-to-peer solutions. We then cover the well-established topics of security, distributed file systems and distributed naming before moving on to important data-related aspects including distributed transactions and data replication. Algorithms associated with all these topics are covered as they arise and also in separate chapters devoted to timing, coordination and agreement.

The book culminates in chapters that address the emerging areas of mobile and ubiquitous computing and distributed multimedia systems before presenting a substantial case study focusing on the design and implementation of the distributed systems infrastructure that supports Google both in terms of core search functionality and the increasing range of additional services offered by Google (for example, Gmail and Google Earth). This last chapter has an important role in illustrating how all the architectural concepts, algorithms and technologies introduced in the book can come together in a coherent overall design for a given application domain.

Purposes and readership

The book is intended for use in undergraduate and introductory postgraduate courses. It can equally be used for self-study. We take a top-down approach, addressing the issues to be resolved in the design of distributed systems and describing successful approaches in the form of abstract models, algorithms and detailed case studies of widely used systems. We cover the field in sufficient depth and breadth to enable readers to go on to study most research papers in the literature on distributed systems.

We aim to make the subject accessible to students who have a basic knowledge of object-oriented programming, operating systems and elementary computer architecture. The book includes coverage of those aspects of computer networks relevant to distributed systems, including the underlying technologies for the Internet and for wide area, local area and wireless networks. Algorithms and interfaces are presented throughout the book in Java or, in a few cases, ANSI C. For brevity and clarity of presentation, a form of pseudo-code derived from Java/C is also used.



Organization of the book

The diagram shows the book's chapters under seven main topic areas. It is intended to provide a guide to the book's structure and to indicate recommended navigation routes for instructors wishing to provide, or readers wishing to achieve, understanding of the various subfields of distributed system design.

References

The existence of the World Wide Web has changed the way in which a book such as this can be linked to source material, including research papers, technical specifications and standards. Many of the source documents are now available on the Web; some are available only there. For reasons of brevity and readability, we employ a special form of reference to web material that loosely resembles a URL: references such as [www.omg.org] and [www.rsasecurity.com I] refer to documentation that is available

only on the Web. They can be looked up in the reference list at the end of the book, but the full URLs are given only in an online version of the reference list at the book's web site, www.cdk5.net/refs where they take the form of clickable links. Both versions of the reference list include a more detailed explanation of this scheme.

Changes relative to the fourth edition

Before embarking on the writing of this new edition, we carried out a survey of teachers who used the fourth edition. From the results, we identified the new material required and a number of changes to be made. In addition, we recognized the increasing diversity of distributed systems, particularly in terms of the range of architectural approaches available to distributed systems developers today. This required significant changes to the book, especially in the earlier (foundational) chapters.

Overall, this led to our writing three entirely new chapters, making substantial changes to a number of other chapters and making numerous insertions throughout the book to fold in new material. Many of the chapters have been changed to reflect new information that has become available about the systems described. These changes are summarized in the table below. To help teachers who have used the fourth edition, wherever possible we have preserved the structure adopted from the previous edition. Where material has been removed, we have placed this on our companion web site together with material removed from previous editions. This includes the case studies on ATM, interprocess communication in UNIX, CORBA (a shortened version of which remains in Chapter 8), the Jini distributed events specification and Grid middleware (featuring OGSA and the Globus toolkit), as well as the chapter on distributed shared memory (a brief summary of which is now included in Chapter 6).

Some of the chapters in the book, such as the new chapter on indirect communication (Chapter 6), cover a lot of material. Teachers may elect to cover the broad spectrum before choosing two or three techniques to examine in more detail (for example, group communication, given its foundational role, and publish-subscribe or message queues, given their prevalence in commercial distributed systems).

The chapter ordering has been changed to accommodate the new material and to reflect changes in the relative importance of certain topics. For a full understanding of some topics readers may find it necessary to follow a forward reference. For example, there is material in Chapter 9 on XML security techniques that will make better sense once the sections that it references in Chapter 11 Security have been absorbed.

Acknowledgements

We are very grateful to the following teachers who participated in our survey: Guohong Cao, Jose Fortes, Bahram Khalili, George Blank, Jinsong Ouyang, JoAnne Holliday, George K. Thiruvathukal, Joel Wein, Tao Xie and Xiaobo Zhou.

We would like to thank the following people who reviewed the new chapters or provided other substantial help: Rob Allen, Roberto Baldoni, John Bates, Tom Berson, Lynne Blair, Geoff Coulson, Paul Grace, Andrew Herbert, David Hutchison, Laurent Mathy, Rajiv Ramdhany, Richard Sharp, Jean-Bernard Stefani, Rip Sohan, Francois

New chapters:	
6 Indirect Communication	Includes events and notification from 4th edition.
8 Distributed Objects and Components	Includes a precised version of the CORBA case study from the 4th edition.
21 Designing Distributed Systems	Includes a major new case study on Google
Chapters which have undergone substantial changes:	
1 Characterization of DS	*Significant restructuring of material* New Section 1.2: Examples of distributed systems Section 1.3.4: Cloud computing introduced
2 System Models	*Significant restructuring of material* New Section 2.2: Physical models Section 2.3: Major rewrite to reflect new book content and associated architectural perspectives
4 Interprocess Communication	*Several updates* Client-server communication moved to Chapter 5 New Section 4.5: Network virtualization (includes case study on Skype) New Section 4.6: Case study on MPI Case study on IPC in UNIX removed
5 Remote Invocation	*Significant restructuring of material* Client-server communication moved to here Progression introduced from client-server communication through RPC to RMI Events and notification moved to Chapter 6
Chapters to which new material has been added/removed, but without structural changes:	
3 Networking and Internetworking	*Several updates* Section 3.5: material on ATM removed
7 Operating System Support	New Section 7.7: OS virtualization
9 Web Services	Section 9.2: Discussion added on loose coupling
10 Peer-to-Peer Systems	New Section 10.5.3: Unstructured peer-to-peer (including a new case study on Gnutella)
15 Coordination and Agreement	Material on group communication moved to Ch. 6
18 Replication	Material on group communication moved to Ch. 6
19 Mobile and Ubiquitous Computing	Section 19.3.1: New material on tuple spaces (TOTA and L^2imbo)
20 Distributed Multimedia Systems	Section 20.6: New case studies added on BitTorrent and End System Multicast
The remaining chapters have received only minor modifications.	

Taiani, Peter Triantafillou, Gareth Tyson and the late Sir Maurice Wilkes. We would also like to thank the staff at Google who provided insights into the design rationale for Google Infrastructure, namely: Mike Burrows, Tushar Chandra, Walfredo Cirne, Jeff Dean, Sanjay Ghemawat, Andrea Kirmse and John Reumann.

Our copy editor, Rachel Head also provided outstanding support.

Web site

As before, we continue to maintain a web site with a wide range of material designed to assist teachers and readers. This web site can be accessed via the URL:

www.cdk5.net

The web site includes:

Instructor's Guide: We provide supporting material for teachers comprising:

- complete artwork of the book available as PowerPoint files;
- chapter-by-chapter teaching hints;
- solutions to the exercises, protected by a password available only to teachers.

Instructor resources for the International Edition are available at www.pearsoninternationaleditions.com/coulouris

Reference list: The list of references that can be found at the end of the book is replicated at the web site. The web version of the reference list includes active links for material that is available online.

Errata list: A list of known errors in the book is maintained, with corrections. The errors will be corrected when new impressions are printed and a separate errata list will be provided for each impression. (Readers are encouraged to report any apparent errors they encounter to the email address below.)

Supplementary material: We maintain a set of supplementary material for each chapter. This consists of source code for the programs in the book and relevant reading material that was present in previous editions of the book but was removed for reasons of space. References to this supplementary material appear in the book with links such as www.cdk5.net/ipc (the URL for supplementary material relating to the Interprocess Communication chapter). Two entire chapters from the 4th edition are not present in this one; they can be accessed at the URLs:

CORBA Case Study	www.cdk5.net/corba
Distributed Shared Memory	www.cdk5.net/dsm

George Coulouris
Jean Dollimore
Tim Kindberg
Gordon Blair
London, Bristol and Lancaster, 2011
authors@cdk5.net

1

CHARACTERIZATION OF DISTRIBUTED SYSTEMS

1.1 Introduction
1.2 Examples of distributed systems
1.3 Trends in distributed systems
1.4 Focus on resource sharing
1.5 Challenges
1.6 Case study: The World Wide Web
1.7 Summary

A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages. This definition leads to the following especially significant characteristics of distributed systems: concurrency of components, lack of a global clock and independent failures of components.

We look at several examples of modern distributed applications, including web search, multiplayer online games and financial trading systems, and also examine the key underlying trends driving distributed systems today: the pervasive nature of modern networking, the emergence of mobile and ubiquitous computing, the increasing importance of distributed multimedia systems, and the trend towards viewing distributed systems as a utility. The chapter then highlights resource sharing as a main motivation for constructing distributed systems. Resources may be managed by servers and accessed by clients or they may be encapsulated as objects and accessed by other client objects.

The challenges arising from the construction of distributed systems are the heterogeneity of their components, openness (which allows components to be added or replaced), security, scalability – the ability to work well when the load or the number of users increases – failure handling, concurrency of components, transparency and providing quality of service. Finally, the Web is discussed as an example of a large-scale distributed system and its main features are introduced.

1.1 Introduction

Networks of computers are everywhere. The Internet is one, as are the many networks of which it is composed. Mobile phone networks, corporate networks, factory networks, campus networks, home networks, in-car networks – all of these, both separately and in combination, share the essential characteristics that make them relevant subjects for study under the heading *distributed systems*. In this book we aim to explain the characteristics of networked computers that impact system designers and implementors and to present the main concepts and techniques that have been developed to help in the tasks of designing and implementing systems that are based on them.

We define a distributed system as one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages. This simple definition covers the entire range of systems in which networked computers can usefully be deployed.

Computers that are connected by a network may be spatially separated by any distance. They may be on separate continents, in the same building or in the same room. Our definition of distributed systems has the following significant consequences:

Concurrency: In a network of computers, concurrent program execution is the norm. I can do my work on my computer while you do your work on yours, sharing resources such as web pages or files when necessary. The capacity of the system to handle shared resources can be increased by adding more resources (for example. computers) to the network. We will describe ways in which this extra capacity can be usefully deployed at many points in this book. The coordination of concurrently executing programs that share resources is also an important and recurring topic.

No global clock: When programs need to cooperate they coordinate their actions by exchanging messages. Close coordination often depends on a shared idea of the time at which the programs' actions occur. But it turns out that there are limits to the accuracy with which the computers in a network can synchronize their clocks – there is no single global notion of the correct time. This is a direct consequence of the fact that the *only* communication is by sending messages through a network. Examples of these timing problems and solutions to them will be described in Chapter 14.

Independent failures: All computer systems can fail, and it is the responsibility of system designers to plan for the consequences of possible failures. Distributed systems can fail in new ways. Faults in the network result in the isolation of the computers that are connected to it, but that doesn't mean that they stop running. In fact, the programs on them may not be able to detect whether the network has failed or has become unusually slow. Similarly, the failure of a computer, or the unexpected termination of a program somewhere in the system (a *crash*), is not immediately made known to the other components with which it communicates. Each component of the system can fail independently, leaving the others still running. The consequences of this characteristic of distributed systems will be a recurring theme throughout the book.

The prime motivation for constructing and using distributed systems stems from a desire to share resources. The term 'resource' is a rather abstract one, but it best characterizes the range of things that can usefully be shared in a networked computer system. It

extends from hardware components such as disks and printers to software-defined entities such as files, databases and data objects of all kinds. It includes the stream of video frames that emerges from a digital video camera and the audio connection that a mobile phone call represents.

The purpose of this chapter is to convey a clear view of the nature of distributed systems and the challenges that must be addressed in order to ensure that they are successful. Section 1.2 gives some illustrative examples of distributed systems, with Section 1.3 covering the key underlying trends driving recent developments. Section 1.4 focuses on the design of resource-sharing systems, while Section 1.5 describes the key challenges faced by the designers of distributed systems: heterogeneity, openness, security, scalability, failure handling, concurrency, transparency and quality of service. Section 1.6 presents a detailed case study of one very well known distributed system, the World Wide Web, illustrating how its design supports resource sharing.

1.2 Examples of distributed systems

The goal of this section is to provide motivational examples of contemporary distributed systems illustrating both the pervasive role of distributed systems and the great diversity of the associated applications.

As mentioned in the introduction, networks are everywhere and underpin many everyday services that we now take for granted: the Internet and the associated World Wide Web, web search, online gaming, email, social networks, eCommerce, etc. To illustrate this point further, consider Figure 1.1, which describes a selected range of key commercial or social application sectors highlighting some of the associated established or emerging uses of distributed systems technology.

As can be seen, distributed systems encompass many of the most significant technological developments of recent years and hence an understanding of the underlying technology is absolutely central to a knowledge of modern computing. The figure also provides an initial insight into the wide range of applications in use today, from relatively localized systems (as found, for example, in a car or aircraft) to global-scale systems involving millions of nodes, from data-centric services to processor-intensive tasks, from systems built from very small and relatively primitive sensors to those incorporating powerful computational elements, from embedded systems to ones that support a sophisticated interactive user experience, and so on.

We now look at more specific examples of distributed systems to further illustrate the diversity and indeed complexity of distributed systems provision today.

1.2.1 Web search

Web search has emerged as a major growth industry in the last decade, with recent figures indicating that the global number of searches has risen to over 10 billion per calendar month. The task of a web search engine is to index the entire contents of the World Wide Web, encompassing a wide range of information styles including web pages, multimedia sources and (scanned) books. This is a very complex task, as current estimates state that the Web consists of over 63 billion pages and one trillion unique web

Figure 1.1 Selected application domains and associated networked applications

Finance and commerce	The growth of eCommerce as exemplified by companies such as Amazon and eBay, and underlying payments technologies such as PayPal; the associated emergence of online banking and trading and also complex information dissemination systems for financial markets.
The information society	The growth of the World Wide Web as a repository of information and knowledge; the development of web search engines such as Google and Yahoo to search this vast repository; the emergence of digital libraries and the large-scale digitization of legacy information sources such as books (for example, Google Books); the increasing significance of user-generated content through sites such as YouTube, Wikipedia and Flickr; the emergence of social networking through services such as Facebook and MySpace.
Creative industries and entertainment	The emergence of online gaming as a novel and highly interactive form of entertainment; the availability of music and film in the home through networked media centres and more widely in the Internet via downloadable or streaming content; the role of user-generated content (as mentioned above) as a new form of creativity, for example via services such as YouTube; the creation of new forms of art and entertainment enabled by emergent (including networked) technologies.
Healthcare	The growth of health informatics as a discipline with its emphasis on online electronic patient records and related issues of privacy; the increasing role of telemedicine in supporting remote diagnosis or more advanced services such as remote surgery (including collaborative working between healthcare teams); the increasing application of networking and embedded systems technology in assisted living, for example for monitoring the elderly in their own homes.
Education	The emergence of e-learning through for example web-based tools such as virtual learning environments; associated support for distance learning; support for collaborative or community-based learning.
Transport and logistics	The use of location technologies such as GPS in route finding systems and more general traffic management systems; the modern car itself as an example of a complex distributed system (also applies to other forms of transport such as aircraft); the development of web-based map services such as MapQuest, Google Maps and Google Earth.
Science	The emergence of the Grid as a fundamental technology for eScience, including the use of complex networks of computers to support the storage, analysis and processing of (often very large quantities of) scientific data; the associated use of the Grid as an enabling technology for worldwide collaboration between groups of scientists.
Environmental management	The use of (networked) sensor technology to both monitor and manage the natural environment, for example to provide early warning of natural disasters such as earthquakes, floods or tsunamis and to co-ordinate emergency response; the collation and analysis of global environmental parameters to better understand complex natural phenomena such as climate change.

addresses. Given that most search engines analyze the entire web content and then carry out sophisticated processing on this enormous database, this task itself represents a major challenge for distributed systems design.

Google, the market leader in web search technology, has put significant effort into the design of a sophisticated distributed system infrastructure to support search (and indeed other Google applications and services such as Google Earth). This represents one of the largest and most complex distributed systems installations in the history of computing and hence demands close examination. Highlights of this infrastructure include:

- an underlying physical infrastructure consisting of very large numbers of networked computers located at data centres all around the world;
- a distributed file system designed to support very large files and heavily optimized for the style of usage required by search and other Google applications (especially reading from files at high and sustained rates);
- an associated structured distributed storage system that offers fast access to very large datasets;
- a lock service that offers distributed system functions such as distributed locking and agreement;
- a programming model that supports the management of very large parallel and distributed computations across the underlying physical infrastructure.

Further details on Google's distributed systems services and underlying communications support can be found in Chapter 21, a compelling case study of a modern distributed system in action.

1.2.2 Massively multiplayer online games (MMOGs)

Massively multiplayer online games offer an immersive experience whereby very large numbers of users interact through the Internet with a persistent virtual world. Leading examples of such games include Sony's EverQuest II and EVE Online from the Finnish company CCP Games. Such worlds have increased significantly in sophistication and now include, complex playing arenas (for example EVE, Online consists of a universe with over 5,000 star systems) and multifarious social and economic systems. The number of players is also rising, with systems able to support over 50,000 simultaneous online players (and the total number of players perhaps ten times this figure).

The engineering of MMOGs represents a major challenge for distributed systems technologies, particularly because of the need for fast response times to preserve the user experience of the game. Other challenges include the real-time propagation of events to the many players and maintaining a consistent view of the shared world. This therefore provides an excellent example of the challenges facing modern distributed systems designers.

A number of solutions have been proposed for the design of massively multiplayer online games:

- Perhaps surprisingly, the largest online game, EVE Online, utilises a *client-server* architecture where a single copy of the state of the world is maintained on a

centralized server and accessed by client programs running on players' consoles or other devices. To support large numbers of clients, the server is a complex entity in its own right consisting of a cluster architecture featuring hundreds of computer nodes (this client-server approach is discussed in more detail in Section 1.4 and cluster approaches are discussed in Section 1.3.4). The centralized architecture helps significantly in terms of the management of the virtual world and the single copy also eases consistency concerns. The goal is then to ensure fast response through optimizing network protocols and ensuring a rapid response to incoming events. To support this, the load is partitioned by allocating individual 'star systems' to particular computers within the cluster, with highly loaded star systems having their own dedicated computer and others sharing a computer. Incoming events are directed to the right computers within the cluster by keeping track of movement of players between star systems.

- Other MMOGs adopt more distributed architectures where the universe is partitioned across a (potentially very large) number of servers that may also be geographically distributed. Users are then dynamically allocated a particular server based on current usage patterns and also the network delays to the server (based on geographical proximity for example). This style of architecture, which is adopted by EverQuest, is naturally extensible by adding new servers.
- Most commercial systems adopt one of the two models presented above, but researchers are also now looking at more radical architectures that are not based on client-server principles but rather adopt completely decentralized approaches based on peer-to-peer technology where every participant contributes resources (storage and processing) to accommodate the game. Further consideration of peer-to-peer solutions is deferred until Chapters 2 and 10).

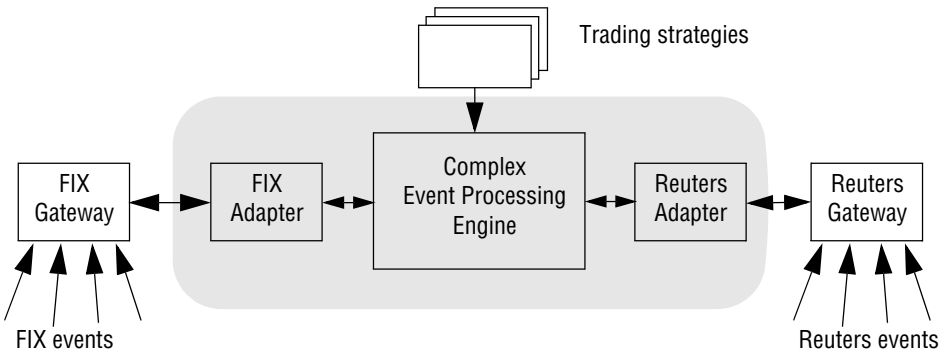
1.2.3 Financial trading

As a final example, we look at distributed systems support for financial trading markets. The financial industry has long been at the cutting edge of distributed systems technology with its need, in particular, for real-time access to a wide range of information sources (for example, current share prices and trends, economic and political developments). The industry employs automated monitoring and trading applications (see below).

Note that the emphasis in such systems is on the communication and processing of items of interest, known as *events* in distributed systems, with the need also to deliver events reliably and in a timely manner to potentially very large numbers of clients who have a stated interest in such information items. Examples of such events include a drop in a share price, the release of the latest unemployment figures, and so on. This requires a very different style of underlying architecture from the styles mentioned above (for example client-server), and such systems typically employ what are known as *distributed event-based systems*. We present an illustration of a typical use of such systems below and return to this important topic in more depth in Chapter 6.

Figure 1.2 illustrates a typical financial trading system. This shows a series of event feeds coming into a given financial institution. Such event feeds share the

Figure 1.2 An example financial trading system



following characteristics. Firstly, the sources are typically in a variety of formats, such as Reuters market data events and FIX events (events following the specific format of the Financial Information eXchange protocol), and indeed from different event technologies, thus illustrating the problem of heterogeneity as encountered in most distributed systems (see also Section 1.5.1). The figure shows the use of adapters which translate heterogeneous formats into a common internal format. Secondly, the trading system must deal with a variety of event streams, all arriving at rapid rates, and often requiring real-time processing to detect patterns that indicate trading opportunities. This used to be a manual process but competitive pressures have led to increasing automation in terms of what is known as Complex Event Processing (CEP), which offers a way of composing event occurrences together into logical, temporal or spatial patterns.

This approach is primarily used to develop customized algorithmic trading strategies covering both buying and selling of stocks and shares, in particular looking for patterns that indicate a trading opportunity and then automatically responding by placing and managing orders. As an example, consider the following script:

```
WHEN
    MSFT price moves outside 2% of MSFT Moving Average
FOLLOWED-BY (
    MyBasket moves up by 0.5%
    AND
        HPQ's price moves up by 5%
        OR
        MSFT's price moves down by 2%
    )
)
ALL WITHIN
    any 2 minute time period
THEN
    BUY MSFT
    SELL HPQ
```

This script is based on the functionality provided by Apama [www.progress.com], a commercial product in the financial world originally developed out of research carried out at the University of Cambridge. The script detects a complex temporal sequence based on the share prices of Microsoft, HP and a basket of other share prices, resulting in decisions to buy or sell particular shares.

This style of technology is increasingly being used in other areas of financial systems including the monitoring of trading activity to manage risk (in particular, tracking exposure), to ensure compliance with regulations and to monitor for patterns of activity that might indicate fraudulent transactions. In such systems, events are typically intercepted and passed through what is equivalent to a compliance and risk firewall before being processed (see also the discussion of firewalls in Section 1.3.1 below).

1.3 Trends in distributed systems

Distributed systems are undergoing a period of significant change and this can be traced back to a number of influential trends:

- the emergence of pervasive networking technology;
- the emergence of ubiquitous computing coupled with the desire to support user mobility in distributed systems;
- the increasing demand for multimedia services;
- the view of distributed systems as a utility.

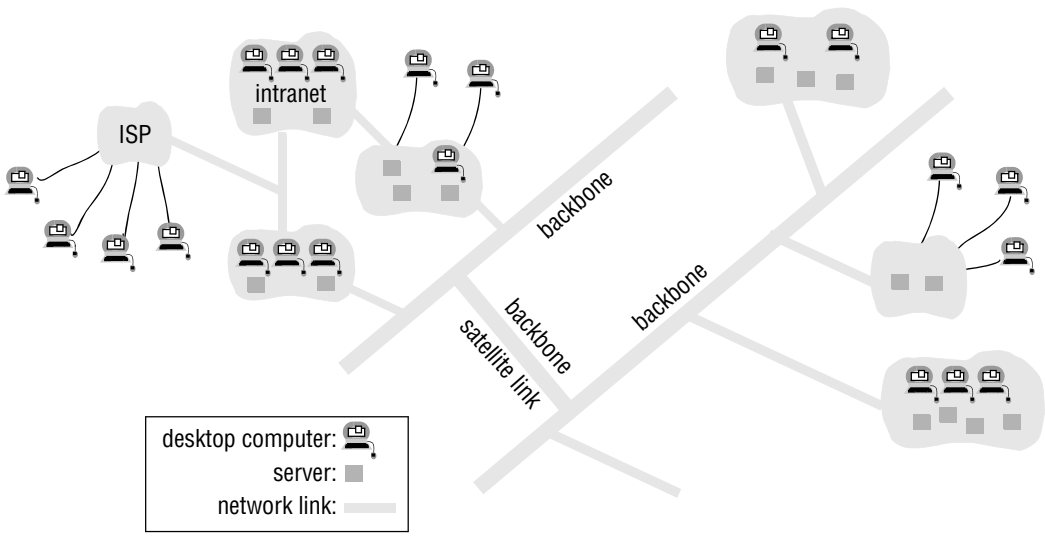
1.3.1 Pervasive networking and the modern Internet

The modern Internet is a vast interconnected collection of computer networks of many different types, with the range of types increasing all the time and now including, for example, a wide range of wireless communication technologies such as WiFi, WiMAX, Bluetooth (see Chapter 3) and third-generation mobile phone networks. The net result is that networking has become a pervasive resource and devices can be connected (if desired) at any time and in any place.

Figure 1.3 illustrates a typical portion of the Internet. Programs running on the computers connected to it interact by passing messages, employing a common means of communication. The design and construction of the Internet communication mechanisms (the Internet protocols) is a major technical achievement, enabling a program running anywhere to address messages to programs anywhere else and abstracting over the myriad of technologies mentioned above.

The Internet is also a very large distributed system. It enables users, wherever they are, to make use of services such as the World Wide Web, email and file transfer. (Indeed, the Web is sometimes incorrectly equated with the Internet.) The set of services is open-ended – it can be extended by the addition of server computers and new types of service. The figure shows a collection of intranets – subnetworks operated by companies and other organizations and typically protected by firewalls. The role of a *firewall* is to protect an intranet by preventing unauthorized messages from leaving or entering. A

Figure 1.3 A typical portion of the Internet



firewall is implemented by filtering incoming and outgoing messages. Filtering might be done by source or destination, or a firewall might allow only those messages related to email and web access to pass into or out of the intranet that it protects. Internet Service Providers (ISPs) are companies that provide broadband links and other types of connection to individual users and small organizations, enabling them to access services anywhere in the Internet as well as providing local services such as email and web hosting. The intranets are linked together by backbones. A *backbone* is a network link with a high transmission capacity, employing satellite connections, fibre optic cables and other high-bandwidth circuits.

Note that some organizations may not wish to connect their internal networks to the Internet at all. For example, police and other security and law enforcement agencies are likely to have at least some internal intranets that are isolated from the outside world (the most effective firewall possible – the absence of any physical connections to the Internet). Firewalls can also be problematic in distributed systems by impeding legitimate access to services when resource sharing between internal and external users is required. Hence, firewalls must often be complemented by more fine-grained mechanisms and policies, as discussed in Chapter 11.

The implementation of the Internet and the services that it supports has entailed the development of practical solutions to many distributed system issues (including most of those defined in Section 1.5). We shall highlight those solutions throughout the book, pointing out their scope and their limitations where appropriate.

1.3.2 Mobile and ubiquitous computing

Technological advances in device miniaturization and wireless networking have led increasingly to the integration of small and portable computing devices into distributed systems. These devices include:

- Laptop computers.
- Handheld devices, including mobile phones, smart phones, GPS-enabled devices, pagers, personal digital assistants (PDAs), video cameras and digital cameras.
- Wearable devices, such as smart watches with functionality similar to a PDA.
- Devices embedded in appliances such as washing machines, hi-fi systems, cars and refrigerators.

The portability of many of these devices, together with their ability to connect conveniently to networks in different places, makes *mobile computing* possible. Mobile computing is the performance of computing tasks while the user is on the move, or visiting places other than their usual environment. In mobile computing, users who are away from their 'home' intranet (the intranet at work, or their residence) are still provided with access to resources via the devices they carry with them. They can continue to access the Internet; they can continue to access resources in their home intranet; and there is increasing provision for users to utilize resources such as printers or even sales points that are conveniently nearby as they move around. The latter is also known as *location-aware* or *context-aware computing*. Mobility introduces a number of challenges for distributed systems, including the need to deal with variable connectivity and indeed disconnection, and the need to maintain operation in the face of device mobility (see the discussion on mobility transparency in Section 1.5.7).

Ubiquitous computing is the harnessing of many small, cheap computational devices that are present in users' physical environments, including the home, office and even natural settings. The term 'ubiquitous' is intended to suggest that small computing devices will eventually become so pervasive in everyday objects that they are scarcely noticed. That is, their computational behaviour will be transparently and intimately tied up with their physical function.

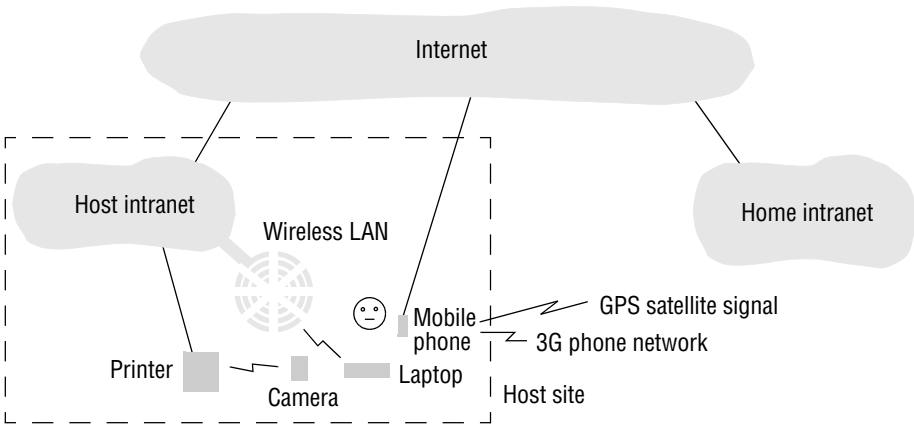
The presence of computers everywhere only becomes useful when they can communicate with one another. For example, it may be convenient for users to control their washing machine or their entertainment system from their phone or a 'universal remote control' device in the home. Equally, the washing machine could notify the user via a smart badge or phone when the washing is done.

Ubiquitous and mobile computing overlap, since the mobile user can in principle benefit from computers that are everywhere. But they are distinct, in general. Ubiquitous computing could benefit users while they remain in a single environment such as the home or a hospital. Similarly, mobile computing has advantages even if it involves only conventional, discrete computers and devices such as laptops and printers.

Figure 1.4 shows a user who is visiting a host organization. The figure shows the user's home intranet and the host intranet at the site that the user is visiting. Both intranets are connected to the rest of the Internet.

The user has access to three forms of wireless connection. Their laptop has a means of connecting to the host's wireless LAN. This network provides coverage of a

Figure 1.4 Portable and handheld devices in a distributed system



few hundred metres (a floor of a building, say). It connects to the rest of the host intranet via a gateway or access point. The user also has a mobile (cellular) telephone, which is connected to the Internet. The phone gives access to the Web and other Internet services, constrained only by what can be presented on its small display, and may also provide location information via built-in GPS functionality. Finally, the user carries a digital camera, which can communicate over a personal area wireless network (with range up to about 10m) with a device such as a printer.

With a suitable system infrastructure, the user can perform some simple tasks in the host site using the devices they carry. While journeying to the host site, the user can fetch the latest stock prices from a web server using the mobile phone and can also use the built-in GPS and route finding software to get directions to the site location. During the meeting with their hosts, the user can show them a recent photograph by sending it from the digital camera directly to a suitably enabled (local) printer or projector in the meeting room (discovered using a location service). This requires only the wireless link between the camera and printer or projector. And they can in principle send a document from their laptop to the same printer, utilizing the wireless LAN and wired Ethernet links to the printer.

This scenario demonstrates the need to support *spontaneous interoperation,* whereby associations between devices are routinely created and destroyed – for example by locating and using the host's devices, such as printers. The main challenge applying to such situations is to make interoperation fast and convenient (that is, spontaneous) even though the user is in an environment they may never have visited before. That means enabling the visitor's device to communicate on the host network, and associating the device with suitable local services – a process called *service discovery*.

Mobile and ubiquitous computing represent lively areas of research, and the various dimensions mentioned above are discussed in depth in Chapter 19.

1.3.3 Distributed multimedia systems

Another important trend is the requirement to support multimedia services in distributed systems. Multimedia support can usefully be defined as the ability to support a range of media types in an integrated manner. One can expect a distributed system to support the storage, transmission and presentation of what are often referred to as discrete media types, such as pictures or text messages. A distributed multimedia system should be able to perform the same functions for continuous media types such as audio and video; that is, it should be able to store and locate audio or video files, to transmit them across the network (possibly in real time as the streams emerge from a video camera), to support the presentation of the media types to the user and optionally also to share the media types across a group of users.

The crucial characteristic of continuous media types is that they include a temporal dimension, and indeed, the integrity of the media type is fundamentally dependent on preserving real-time relationships between elements of a media type. For example, in a video presentation it is necessary to preserve a given throughput in terms of frames per second and, for real-time streams, a given maximum delay or latency for the delivery of frames (this is one example of quality of service, discussed in more detail in Section 1.5.8).

The benefits of distributed multimedia computing are considerable in that a wide range of new (multimedia) services and applications can be provided on the desktop, including access to live or pre-recorded television broadcasts, access to film libraries offering video-on-demand services, access to music libraries, the provision of audio and video conferencing facilities and integrated telephony features including IP telephony or related technologies such as Skype, a peer-to-peer alternative to IP telephony (the distributed system infrastructure underpinning Skype is discussed in Section 4.5.2). Note that this technology is revolutionary in challenging manufacturers to rethink many consumer devices. For example, what is the core home entertainment device of the future – the computer, the television, or the games console?

Webcasting is an application of distributed multimedia technology. Webcasting is the ability to broadcast continuous media, typically audio or video, over the Internet. It is now commonplace for major sporting or music events to be broadcast in this way, often attracting large numbers of viewers (for example, the Live8 concert in 2005 attracted around 170,000 simultaneous users at its peak).

Distributed multimedia applications such as webcasting place considerable demands on the underlying distributed infrastructure in terms of:

- providing support for an (extensible) range of encoding and encryption formats, such as the MPEG series of standards (including for example the popular MP3 standard otherwise known as MPEG-1, Audio Layer 3) and HDTV;
- providing a range of mechanisms to ensure that the desired quality of service can be met;
- providing associated resource management strategies, including appropriate scheduling policies to support the desired quality of service;
- providing adaptation strategies to deal with the inevitable situation in open systems where quality of service cannot be met or sustained.

Further discussion of such mechanisms can be found in Chapter 20.

1.3.4 Distributed computing as a utility

With the increasing maturity of distributed systems infrastructure, a number of companies are promoting the view of distributed resources as a commodity or utility, drawing the analogy between distributed resources and other utilities such as water or electricity. With this model, resources are provided by appropriate service suppliers and effectively rented rather than owned by the end user. This model applies to both physical resources and more logical services:

- Physical resources such as storage and processing can be made available to networked computers, removing the need to own such resources on their own. At one end of the spectrum, a user may opt for a remote storage facility for file storage requirements (for example, for multimedia data such as photographs, music or video) and/or for backups. Similarly, this approach would enable a user to rent one or more computational nodes, either to meet their basic computing needs or indeed to perform distributed computation. At the other end of the spectrum, users can access sophisticated *data centres* (networked facilities offering access to repositories of often large volumes of data to users or organizations) or indeed computational infrastructure using the sort of services now provided by companies such as Amazon and Google. Operating system virtualization is a key enabling technology for this approach, implying that users may actually be provided with services by a virtual rather than a physical node. This offers greater flexibility to the service supplier in terms of resource management (operating system virtualization is discussed in more detail in Chapter 7).
- Software services (as defined in Section 1.4) can also be made available across the global Internet using this approach. Indeed, many companies now offer a comprehensive range of services for effective rental, including services such as email and distributed calendars. Google, for example, bundles a range of business services under the banner Google Apps [www.google.com I]. This development is enabled by agreed standards for software services, for example as provided by web services (see Chapter 9).

The term *cloud computing* is used to capture this vision of computing as a utility. A cloud is defined as a set of Internet-based application, storage and computing services sufficient to support most users' needs, thus enabling them to largely or totally dispense with local data storage and application software (see Figure 1.5). The term also promotes a view of everything as a service, from physical or virtual infrastructure through to software, often paid for on a per-usage basis rather than purchased. Note that cloud computing reduces requirements on users' devices, allowing very simple desktop or portable devices to access a potentially wide range of resources and services.

Clouds are generally implemented on cluster computers to provide the necessary scale and performance required by such services. A *cluster computer* is a set of interconnected computers that cooperate closely to provide a single, integrated high-performance computing capability. Building on projects such as the NOW (Network of Workstations) Project at Berkeley [Anderson *et al*. 1995, now.cs.berkeley.edu] and Beowulf at NASA [www.beowulf.org], the trend is towards utilizing commodity hardware both for the computers and for the interconnecting networks. Most clusters