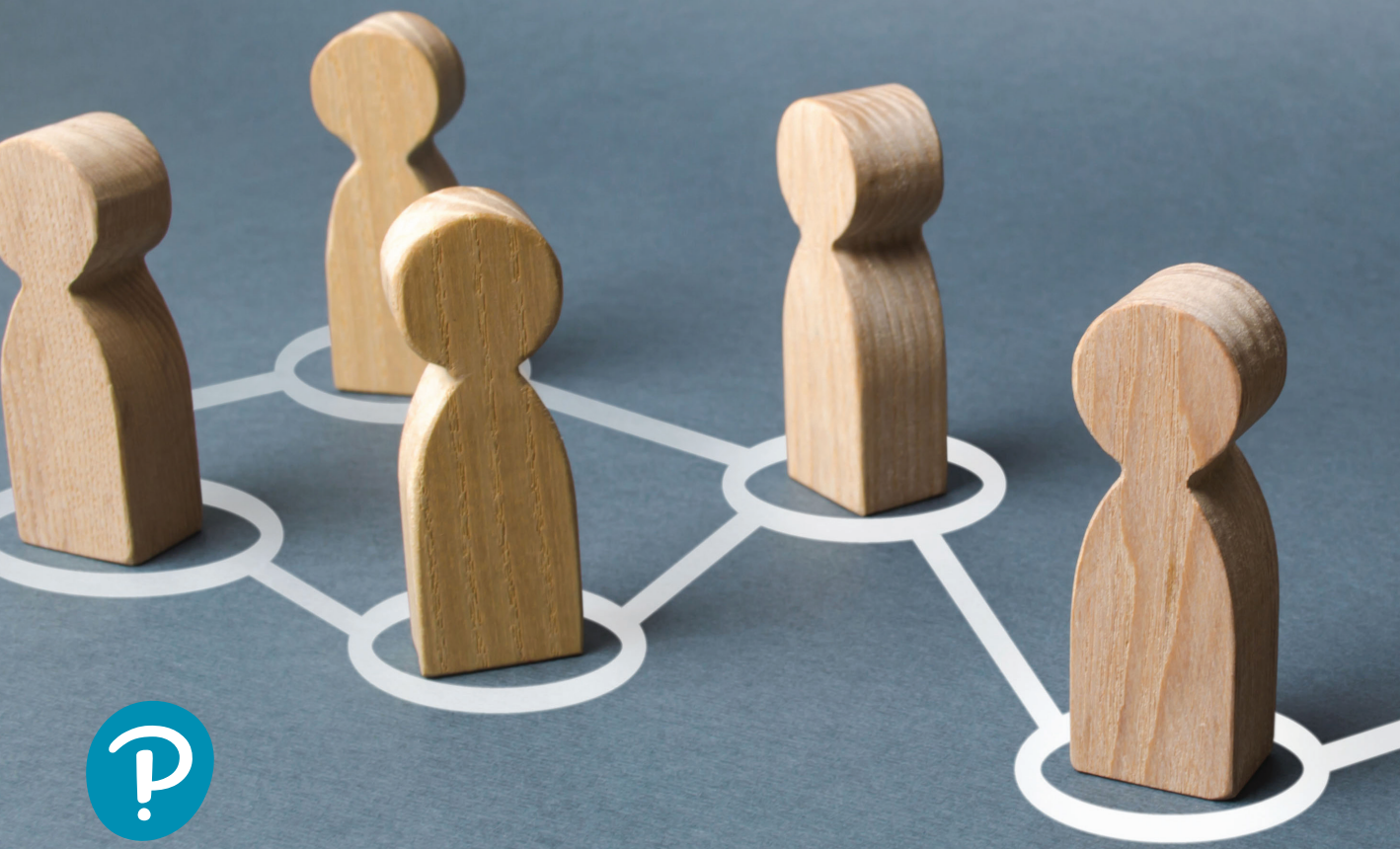
GLOBAL EDITION

# Computer Networking

## *A Top-Down Approach*

EIGHTH EDITION

James F. Kurose • Keith W. Ross



# DIGITAL RESOURCES FOR STUDENTS

Your new textbook provides 12-month access to digital resources that may include VideoNotes (illustrating key concepts from the text), interactive exercises, interactive animations, quizzes, and more. Refer to the preface in the textbook for a detailed list of resources.

Follow the instructions below to register for the Companion Website for James F. Kurose and Keith W. Ross's ***Computer Networking: A Top-Down Approach, Eighth Edition, Global Edition.***

1. Go to **www.pearsonglobaleditions.com**.
2. Enter the title of your textbook or browse by author name.
3. Click Companion Website.
4. Click Register and follow the on-screen instructions to create a login name and password.

ISSJKK-FROMM-DAIRY-CUPPA-PLUSH-POSES

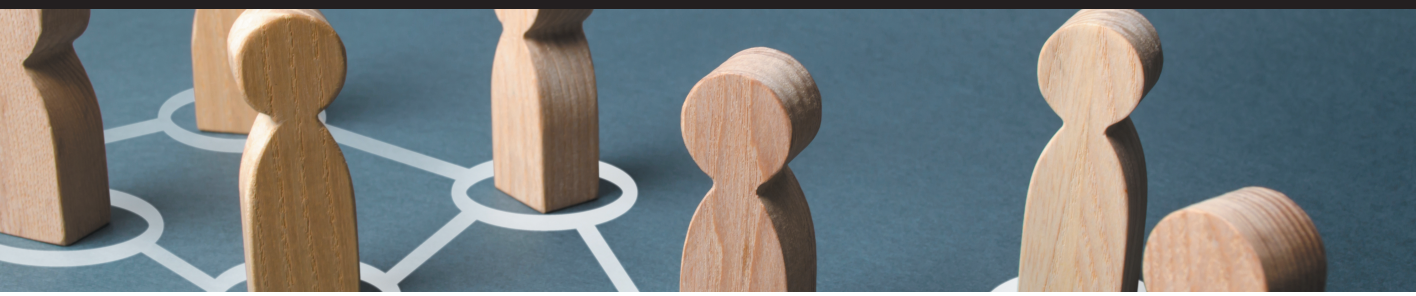
**Use the login name and password you created during registration to start using the digital resources that accompany your textbook.**

For technical support go to **https://support.pearson.com/getsupport**

# COMPUTER NETWORKING

EIGHTH EDITION
GLOBAL EDITION

## A Top-Down Approach




**JAMES F. KUROSE**
*University of Massachusetts, Amherst*

**KEITH W. ROSS**
*NYU and NYU Shanghai*



Pearson

*Pearson Education Limited*
KAO Two
KAO Park
Hockham Way
Harlow
CM17 9SR
United Kingdom

and Associated Companies throughout the world

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

Please contact https://support.pearson.com/getsupport/s/contactsupport with any queries on this content



*Authorized adaptation from the United States edition, entitled* Computer Networking: A Top-Down Approach, *8th Edition, ISBN 978-0-13-668155-7 by James F. Kurose and Keith W. Ross, published by Pearson Education © 2021.*



**ISBN 10:** 1-292-40546-5
**ISBN 13:** 978-1-292-40546-9
**eBook ISBN 13:** 978-1-292-40551-3
**British Library Cataloguing-in-Publication Data**
A catalogue record for this book is available from the British Library

Typeset by SPi Global
eBook formatted by B2R Technologies Pvt. Ltd.

# About the Authors

## Jim Kurose

Jim Kurose is a Distinguished University Professor in the College of Information and Computer Sciences at the University of Massachusetts Amherst, where he has been on the faculty since receiving his PhD in computer science from Columbia University. He received a BA in physics from Wesleyan University. He has held a number of visiting scientist positions in the United States and abroad, including IBM Research, INRIA, and the Sorbonne University in France. He recently completed a five-year term as Assistant Director at the US National Science Foundation, where he led the Directorate of Computer and Information Science and Engineering in its mission to uphold the nation's leadership in scientific discovery and engineering innovation.

Jim is proud to have mentored and taught an amazing group of students, and to have received a number of awards for his research, teaching, and service, including the IEEE Infocom Award, the ACM SIGCOMM Lifetime Achievement Award, the ACM Sigcomm Test of Time Award, and the IEEE Computer Society Taylor Booth Education Medal. Dr. Kurose is a former Editor-in-Chief of IEEE Transactions on Communications and of IEEE/ACM Transactions on Networking. He has served as Technical Program co-Chair for IEEE Infocom, ACM SIGCOMM, ACM Internet Measurement Conference, and ACM SIGMETRICS. He is a Fellow of the IEEE, the ACM and a member of the National Academy of Engineering. His research interests include network protocols and architecture, network measurement, multimedia communication, and modeling and performance evaluation.

## Keith Ross

Keith Ross is the Dean of Engineering and Computer Science at NYU Shanghai and the Leonard J. Shustek Chair Professor in the Computer Science and Engineering Department at NYU. Previously he was at University of Pennsylvania (13 years), Eurecom Institute (5 years) and NYU-Poly (10 years). He received a B.S.E.E from Tufts University, a M.S.E.E. from Columbia University, and a Ph.D. in Computer and Control Engineering from The University of Michigan. Keith Ross is also the co-founder and original CEO of Wimba, which develops online multimedia applications for e-learning and was acquired by Blackboard in 2010.

Professor Ross's research interests have been in modeling and meaurement of computer networks, peer-to-peer systems, content distribution networks, social networks, and privacy. He is currently working in deep reinforcement

learning. He is an ACM Fellow, an IEEE Fellow, recipient of the Infocom 2009 Best Paper Award, and recipient of 2011 and 2008 Best Paper Awards for Multimedia Communications (awarded by IEEE Communications Society). He has served on numerous journal editorial boards and conference program committees, including *IEEE/ACM Transactions on Networking, ACM SIGCOMM, ACM CoNext, and ACM Internet Measurement Conference*. He also has served as an advisor to the Federal Trade Commission on P2P file sharing.

To Julie and our three precious
ones—Chris, Charlie, and Nina
JFK

A big THANKS to my professors, colleagues,
and students all over the world.
KWR

*This page is intentionally left blank*

# Preface

Welcome to the eighth edition of *Computer Networking: A Top-Down Approach.* Since the publication of the first edition 20 years ago, our book has been adopted for use at many hundreds of colleges and universities, translated into 14 languages, and used by many hundreds of thousands students and practitioners worldwide. We've heard from many of these readers and have been overwhelmed by the positive response.

## What's New in the Eighth Edition?

We think one important reason for this success has been that our book continues to offer a fresh and timely approach to computer networking instruction. We've made changes in this eighth edition, but we've also kept unchanged what we believe (and the instructors and students who have used our book have confirmed) to be the most important aspects of this book: its top-down approach, its focus on the Internet and a modern treatment of computer networking, its attention to both principles and practice, and its accessible style and approach toward learning about computer networking. Nevertheless, the eighth edition has been revised and updated substantially.

Readers of earlier editions of our book may recall that in moving from the sixth to the seventh edition, we deepened our coverage of the network layer, expanding material which had been previously covered in a single chapter into a new chapter focused on the so-called "data plane" component of the network layer (Chapter 4) and a new chapter focused on the network layer's "control plane" (Chapter 5). That change turned out to be prescient, as software-defined networking (SDN), arguably the most important and exciting advance in networking in decades, has been rapidly adopted in practice—so much so that it's already hard to imagine an introduction to modern computer networking that doesn't cover SDN. SDN has also enabled new advances in the practice of network management, which we also cover in modernized and deeper detail in this edition. And as we'll see in Chapter 7 of this eighth edition, the separation of the data and control planes is now also deeply embedded in 4G/5G mobile cellular network architectures, as is an "all-IP" approach to their core networks. The rapid adoption of 4G/5G networks and the mobile applications they enable are undoubtedly the most significant changes we've seen in networking since the publication of our seventh edition. We've thus significantly updated and deepened our treatment of this exciting area. Indeed, the ongoing wireless network revolution is so important that we think it has become a critical part of an introductory networking course.

In addition to these changes, we've also updated many sections throughout the book and added new material to reflect changes across the breadth of networking. In some cases, we have also retired material from the previous edition. As always,

material that has been retired from the printed text can always be found on our book's Companion Website. The most important changes in this eighth edition are the following:

- **Chapter 1** has been updated to reflect the ever-growing reach and use of the Internet, and of 4G/5G networks.
- **Chapter 2**, which covers the application layer, has been significantly updated, including material on the new HTTP/2 and HTTP/3 protocols for the Web.
- **Chapter 3**, has been updated to reflect advances in, and evolution in use of, transport-layer congestion control and error-control protocols over the past five years. While this material had remained relatively stable for quite some time, there have been a number of important advances since the seventh edition. Several new congestion-control algorithms have been developed and deployed beyond the "classic" TCP algorithms. We provide a deeper coverage of TCP CUBIC, the default TCP protocol in many deployed systems, and examine delay-based approaches to congestion control, including the new BBR protocol, which is deployed in Google's backbone network. We also study the QUIC protocol, which is being incorporated into the HTTP/3 standard. Although QUIC is technically not a transport-layer protocol—it provides application-layer reliability, congestion control, and connection multiplexing services at the application layer—it uses many of the error- and congestion-control principles that we develop in the early sections of Chapter 3.
- **Chapter 4**, which covers the network-layer data plane, has general updates throughout. We've added a new section on so-called middleboxes, which perform network-layer functions other than routing and forwarding, such as firewalling and load balancing. Middleboxes build naturally on the generalized "match plus action" forwarding operation of network-layer devices that we cover earlier in Chapter 4. We've also added timely new material on topics such as the amount of buffering that is "just right" in network routers, on net neutrality, and on the architectural principles of the Internet.
- **Chapter 5**, which cover the network-layer's control plane, contains updated material on SDN, and a significantly new treatment of network management. The use of SDN has evolved beyond management of packet-forwarding tables to include configuration management of network devices as well. We introduce two new protocols, NETCONF and YANG, whose adoption and use have fueled this new approach toward network management.
- **Chapter 6**, which covers the link layer, has been updated to reflect the continuing evolution of link-layer technologies such as Ethernet. We have also updated and expanded our treatment of datacenter networks, which are at the heart of the technology driving much of today's Internet commerce.
- As noted earlier, **Chapter 7** has been significantly updated and revised to reflect the many changes in wireless networking since the seventh edition, from short-range Bluetooth piconets, to medium-range wireless 802.11 local area networks (WLANs), to wide-area 4G/5G wireless cellular networks. We have retired our

coverage of earlier 2G and 3G networks in favor of a broader and deeper treatment of today's 4G LTE networks and tomorrow's 5G networks. We have also updated our coverage of mobility issues, from the local issue of handover of mobile devices between base stations to the global issue of identity management and mobile device roaming among different global cellular networks.

- **Chapter 8**, which covers network security, has been updated to reflect changes in wireless network security in particular, with new material on WPA3 security in WLANs, and mutual device/network mutual authentication and confidentiality in 4G/5G networks.

We have also retired Chapter 9, on multimedia networking, from this edition. Over time, as multimedia applications became more prevalent, we had already migrated Chapter 9 material on topics such as video streaming, packet scheduling, and content distribution networks into earlier chapters. As noted earlier, all retired material from this and earlier editions can be found on our book's Companion Website.

### Audience

This textbook is for a first course on computer networking. It can be used in both computer science and electrical engineering departments. In terms of programming languages, the book assumes only that the student has experience with C, C++, Java, or Python (and even then only in a few places). Although this book is more precise and analytical than many other introductory computer networking texts, it rarely uses any mathematical concepts that are not taught in high school. We have made a deliberate effort to avoid using any advanced calculus, probability, or stochastic process concepts (although we've included some homework problems for students with this advanced background). The book is therefore appropriate for undergraduate courses and for first-year graduate courses. It should also be useful to practitioners in the networking industry.

## What Is Unique About This Textbook?

The subject of computer networking is enormously complex, involving many concepts, protocols, and technologies that are woven together in an intricate manner. To cope with this scope and complexity, many computer networking texts are often organized around the "layers" of a network architecture. With a layered organization, students can see through the complexity of computer networking—they learn about the distinct concepts and protocols in one part of the architecture while seeing the big picture of how all parts fit together. From a pedagogical perspective, our personal experience has been that such a layered approach indeed works well. Nevertheless, we have found that the traditional approach of teaching—bottom up; that is, from the physical layer toward the application layer—is not the best approach for a modern course on computer networking.

## A Top-Down Approach

Our book broke new ground 20 years ago by treating networking in a top-down manner—that is, by beginning at the application layer and working its way down toward the physical layer. The feedback we received from teachers and students alike have confirmed that this top-down approach has many advantages and does indeed work well pedagogically. First, it places emphasis on the application layer (a "high growth area" in networking). Indeed, many of the recent revolutions in computer networking—including the Web, and media streaming—have taken place at the application layer. An early emphasis on application-layer issues differs from the approaches taken in most other texts, which have only a small amount of material on network applications, their requirements, application-layer paradigms (e.g., client-server and peer-to-peer), and application programming interfaces. Second, our experience as instructors (and that of many instructors who have used this text) has been that teaching networking applications near the beginning of the course is a powerful motivational tool. Students are thrilled to learn about how networking applications work—applications such as e-mail, streaming video, and the Web, which most students use on a daily basis. Once a student understands the applications, the student can then understand the network services needed to support these applications. The student can then, in turn, examine the various ways in which such services might be provided and implemented in the lower layers. Covering applications early thus provides motivation for the remainder of the text.

Third, a top-down approach enables instructors to introduce network application development at an early stage. Students not only see how popular applications and protocols work, but also learn how easy it is to create their own network applications and application-layer protocols. With the top-down approach, students get early exposure to the notions of socket programming, service models, and protocols—important concepts that resurface in all subsequent layers. By providing socket programming examples in Python, we highlight the central ideas without confusing students with complex code. Undergraduates in electrical engineering and computer science will have no difficulty following the Python code.

## An Internet Focus

Although we dropped the phrase "Featuring the Internet" from the title of this book with the fourth edition, this doesn't mean that we dropped our focus on the Internet. Indeed, nothing could be further from the case! Instead, since the Internet has become so pervasive, we felt that any networking textbook must have a significant focus on the Internet, and thus this phrase was somewhat unnecessary. We continue to use the Internet's architecture and protocols as primary vehicles for studying fundamental computer networking concepts. Of course, we also include concepts and protocols from other network architectures. But the spotlight is clearly on the Internet, a fact reflected in our organizing the book around the Internet's five-layer architecture: the application, transport, network, link, and physical layers.

Another benefit of spotlighting the Internet is that most computer science and electrical engineering students are eager to learn about the Internet and its protocols. They know that the Internet has been a revolutionary and disruptive technology and can see that it is profoundly changing our world. Given the enormous relevance of the Internet, students are naturally curious about what is "under the hood." Thus, it is easy for an instructor to get students excited about basic principles when using the Internet as the guiding focus.

## Teaching Networking Principles

Two of the unique features of the book—its top-down approach and its focus on the Internet—have appeared in the titles of our book. If we could have squeezed a *third* phrase into the subtitle, it would have contained the word *principles.* The field of networking is now mature enough that a number of fundamentally important issues can be identified. For example, in the transport layer, the fundamental issues include reliable communication over an unreliable network layer, connection establishment/teardown and handshaking, congestion and flow control, and multiplexing. Three fundamentally important network-layer issues are determining "good" paths between two routers, interconnecting a large number of heterogeneous networks, and managing the complexity of a modern network. In the link layer, a fundamental problem is sharing a multiple access channel. In network security, techniques for providing confidentiality, authentication, and message integrity are all based on cryptographic fundamentals. This text identifies fundamental networking issues and studies approaches toward addressing these issues. The student learning these principles will gain knowledge with a long "shelf life"—long after many of today's network standards and protocols have become obsolete, the principles they embody will remain important and relevant. We believe that the combination of using the Internet to get the student's foot in the door and then emphasizing fundamental issues and solution approaches will allow the student to quickly understand just about any networking technology.

## Student Resources

Student resources are available on the Companion Website (CW) at www.pearsonglobaleditions.com. Resources include:

- *Interactive learning material.* The book's Website contains VideoNotes—video presentations of important topics throughout the book done by the authors, as well as walkthroughs of solutions to problems similar to those at the end of the chapter. We've seeded the Website with VideoNotes and online problems for Chapters 1 through 5. As in earlier editions, the Website contains the interactive animations that illustrate many key networking concepts. Professors can integrate these interactive features into their lectures or use them as mini labs.

- *Additional technical material.* As we have added new material in each edition of our book, we've had to remove coverage of some existing topics to keep the book at manageable length. Material that appeared in earlier editions of the text is still of interest, and thus can be found on the book's Website.
- *Programming assignments.* The Website also provides a number of detailed programming assignments, which include building a multithreaded Web server, building an e-mail client with a GUI interface, programming the sender and receiver sides of a reliable data transport protocol, programming a distributed routing algorithm, and more.
- *Wireshark labs.* One's understanding of network protocols can be greatly deepened by seeing them in action. The Website provides numerous Wireshark assignments that enable students to actually observe the sequence of messages exchanged between two protocol entities. The Website includes separate Wireshark labs on HTTP, DNS, TCP, UDP, IP, ICMP, Ethernet, ARP, WiFi, TLS and on tracing all protocols involved in satisfying a request to fetch a Web page. We'll continue to add new labs over time.

## Pedagogical Features

We have each been teaching computer networking for more than 30 years. Together, we bring more than 60 years of teaching experience to this text, during which time we have taught many thousands of students. We have also been active researchers in computer networking during this time. (In fact, Jim and Keith first met each other as master's students in a computer networking course taught by Mischa Schwartz in 1979 at Columbia University.) We think all this gives us a good perspective on where networking has been and where it is likely to go in the future. Nevertheless, we have resisted temptations to bias the material in this book toward our own pet research projects. We figure you can visit our personal Websites if you are interested in our research. Thus, this book is about modern computer networking—it is about contemporary protocols and technologies as well as the underlying principles behind these protocols and technologies. We also believe that learning (and teaching!) about networking can be fun. A sense of humor, use of analogies, and real-world examples in this book will hopefully make this material more fun.

## Supplements for Instructors

We provide a complete supplements package to aid instructors in teaching this course. This material can be accessed from Pearson's Instructor Resource Center (http://www.pearsonglobaleditions.com). Visit the Instructor Resource Center for information about accessing these instructor's supplements.

- *PowerPoint® slides.* We provide PowerPoint slides for all eight chapters. The slides have been completely updated with this eighth edition. The slides cover each chapter in detail. They use graphics and animations (rather than relying only on monotonous text bullets) to make the slides interesting and visually appealing. We provide the original PowerPoint slides so you can customize them to best suit your own teaching needs. Some of these slides have been contributed by other instructors who have taught from our book.
- *Homework solutions.* We provide a solutions manual for the homework problems in the text, programming assignments, and Wireshark labs. As noted earlier, we've introduced many new homework problems at each chapter's end. For additional interactive problems and solutions, an instructor (and students) can consult this books Companion Website at Pearson.

## Chapter Dependencies

The first chapter of this text presents a self-contained overview of computer networking. Introducing many key concepts and terminology, this chapter sets the stage for the rest of the book. All of the other chapters directly depend on this first chapter. After completing Chapter 1, we recommend instructors cover Chapters 2 through 6 in sequence, following our top-down philosophy. Each of these five chapters leverages material from the preceding chapters. After completing the first six chapters, the instructor has quite a bit of flexibility. There are no interdependencies among the last two chapters, so they can be taught in any order. However, the last two chapters depends on the material in the first six chapters. Many instructors first teach the first six chapters and then teach one of the last two chapters for "dessert."

## One Final Note: We'd Love to Hear from You

We encourage students and instructors to e-mail us with any comments they might have about our book. It's been wonderful for us to hear from so many instructors and students from around the world about our first seven editions. We've incorporated many of these suggestions into later editions of the book. We also encourage instructors

to send us new homework problems (and solutions) that would complement the current homework problems. We'll post these on the instructor-only portion of the Website. We also encourage instructors and students to create new interactive animations that illustrate the concepts and protocols in this book. If you have an animation that you think would be appropriate for this text, please submit it to us. If the animation (including notation and terminology) is appropriate, we'll be happy to include it on the text's Website, with an appropriate reference to the animation's authors.

So, as the saying goes, "Keep those cards and letters coming!" Seriously, please *do* continue to send us interesting URLs, point out typos, disagree with any of our claims, and tell us what works and what doesn't work. Tell us what you think should or shouldn't be included in the next edition. Send your e-mail to kurose@cs.umass.edu and keithwross@nyu.edu.

## Acknowledgments

Since we began writing this book in 1996, many people have given us invaluable help and have been influential in shaping our thoughts on how to best organize and teach a networking course. We want to say A BIG THANKS to everyone who has helped us from the earliest first drafts of this book, up to this eighth edition. We are also *very* thankful to the thousands of readers from around the world—students, faculty, practitioners—who have sent us thoughts and comments on earlier editions of the book and suggestions for future editions of the book. Special thanks go out to:

Al Aho (Columbia University)
Hisham Al-Mubaid (University of Houston-Clear Lake)
Pratima Akkunoor (Arizona State University)
Paul Amer (University of Delaware)
Shamiul Azom (Arizona State University)
Lichun Bao (University of California at Irvine)
Paul Barford (University of Wisconsin)
Bobby Bhattacharjee (University of Maryland)
Steven Bellovin (Columbia University)
Pravin Bhagwat (Wibhu)
Supratik Bhattacharyya (Amazon)
Ernst Biersack (Eurécom Institute)
Shahid Bokhari (University of Engineering & Technology, Lahore)
Jean Bolot (Technicolor Research)
Daniel Brushteyn (former University of Pennsylvania student)
Ken Calvert (University of Kentucky)
Evandro Cantu (Federal University of Santa Catarina)
Jeff Case (SNMP Research International)
Jeff Chaltas (Sprint)
Vinton Cerf (Google)

Byung Kyu Choi (Michigan Technological University)
Bram Cohen (BitTorrent, Inc.)
Constantine Coutras (Pace University)
John Daigle (University of Mississippi)
Edmundo A. de Souza e Silva (Federal University of Rio de Janeiro)
Philippe Decuetos (former Eurecom Institute student)
Christophe Diot (Google)
Prithula Dhunghel (Akamai)
Deborah Estrin (Cornell University)
Michalis Faloutsos (University of California at Riverside)
Wu-chi Feng (Oregon Graduate Institute)
Sally Floyd (ICIR, University of California at Berkeley)
Paul Francis (Max Planck Institute)
David Fullager (Netflix)
Lixin Gao (University of Massachusetts)
JJ Garcia-Luna-Aceves (University of California at Santa Cruz)
Mario Gerla (University of California at Los Angeles)
David Goodman (NYU-Poly)
Yang Guo (Alcatel/Lucent Bell Labs)
Tim Griffin (Cambridge University)
Max Hailperin (Gustavus Adolphus College)
Bruce Harvey (Florida A&M University, Florida State University)
Carl Hauser (Washington State University)
Rachelle Heller (George Washington University)
Phillipp Hoschka (INRIA/W3C)
Wen Hsin (Park University)
Albert Huang (former University of Pennsylvania student)
Cheng Huang (Microsoft Research)
Esther A. Hughes (Virginia Commonwealth University)
Van Jacobson (Google)
Pinak Jain (former NYU-Poly student)
Jobin James (University of California at Riverside)
Sugih Jamin (University of Michigan)
Shivkumar Kalyanaraman (IBM Research, India)
Jussi Kangasharju (University of Helsinki)
Sneha Kasera (University of Utah)
Parviz Kermani (U. Massachusetts)
Hyojin Kim (former University of Pennsylvania student)
Leonard Kleinrock (University of California at Los Angeles)
David Kotz (Dartmouth College)
Beshan Kulapala (Arizona State University)
Rakesh Kumar (Bloomberg)
Miguel A. Labrador (University of South Florida)
Simon Lam (University of Texas)

Steve Lai (Ohio State University)
Tom LaPorta (Penn State University)
Tim-Berners Lee (World Wide Web Consortium)
Arnaud Legout (INRIA)
Lee Leitner (Drexel University)
Brian Levine (University of Massachusetts)
Chunchun Li (former NYU-Poly student)
Yong Liu (NYU-Poly)
William Liang (former University of Pennsylvania student)
Willis Marti (Texas A&M University)
Nick McKeown (Stanford University)
Josh McKinzie (Park University)
Deep Medhi (University of Missouri, Kansas City)
Bob Metcalfe (International Data Group)
Vishal Misra (Columbia University)
Sue Moon (KAIST)
Jenni Moyer (Comcast)
Erich Nahum (IBM Research)
Christos Papadopoulos (Colorado Sate University)
Guru Parulkar (Open Networking Foundation)
Craig Partridge (Colorado State University)
Radia Perlman (Dell EMC)
Jitendra Padhye (Microsoft Research)
Vern Paxson (University of California at Berkeley)
Kevin Phillips (Sprint)
George Polyzos (Athens University of Economics and Business)
Sriram Rajagopalan (Arizona State University)
Ramachandran Ramjee (Microsoft Research)
Ken Reek (Rochester Institute of Technology)
Martin Reisslein (Arizona State University)
Jennifer Rexford (Princeton University)
Leon Reznik (Rochester Institute of Technology)
Pablo Rodrigez (Telefonica)
Sumit Roy (University of Washington)
Catherine Rosenberg (University of Waterloo)
Dan Rubenstein (Columbia University)
Avi Rubin (Johns Hopkins University)
Douglas Salane (John Jay College)
Despina Saparilla (Cisco Systems)
John Schanz (Comcast)
Henning Schulzrinne (Columbia University)
Mischa Schwartz (Columbia University)
Ardash Sethi (University of Delaware)
Harish Sethu (Drexel University)

K. Sam Shanmugan (University of Kansas)
Prashant Shenoy (University of Massachusetts)
Clay Shields (Georgetown University)
Subin Shrestra (University of Pennsylvania)
Bojie Shu (former NYU-Poly student)
Mihail L. Sichitiu (NC State University)
Peter Steenkiste (Carnegie Mellon University)
Tatsuya Suda (University of California at Irvine)
Kin Sun Tam (State University of New York at Albany)
Don Towsley (University of Massachusetts)
David Turner (California State University, San Bernardino)
Nitin Vaidya (Georgetown University)
Michele Weigle (Clemson University)
David Wetherall (Google)
Ira Winston (University of Pennsylvania)
Di Wu (Sun Yat-sen University)
Shirley Wynn (former NYU-Poly student)
Raj Yavatkar (Google)
Yechiam Yemini (Columbia University)
Dian Yu (former NYU-Shanghai student)
Ming Yu (State University of New York at Binghamton)
Ellen Zegura (Georgia Institute of Technology)
Honggang Zhang (Suffolk University)
Hui Zhang (Carnegie Mellon University)
Lixia Zhang (University of California at Los Angeles)
Meng Zhang (former NYU-Poly student)
Shuchun Zhang (former University of Pennsylvania student)
Xiaodong Zhang (Ohio State University)
ZhiLi Zhang (University of Minnesota)
Phil Zimmermann (independent consultant)
Mike Zink (University of Massachusetts)
Cliff C. Zou (University of Central Florida)

We also want to thank the entire Pearson team—in particular, Carole Snyder and Tracy Johnson—who have done an absolutely outstanding job on this eighth edition (and who have put up with two very finicky authors who seem congenitally unable to meet deadlines!). Thanks also to artists, Janet Theurer and Patrice Rossi Calkin, for their work on the beautiful figures in earlier editions of our book, and to Manas Roy and his team at SPi Global for their wonderful production work on this edition. Finally, a most special thanks go to our previous editors at Addison-Wesley and Pearson—Matt Goldstein, Michael Hirsch, and Susan Hartman. This book would not be what it is (and may well not have been at all) without their graceful management, constant encouragement, nearly infinite patience, good humor, and perseverance.

# Acknowledgments for the Global Edition

Pearson would like to thank and acknowledge the following people for their contributions to the Global Edition.

**Contributors**
Vangelis Angelakis (Linköping University)

**Reviewers**
Wim Lamotte (Universiteit Hasselt)
Wei Tsang Ooi (National University of Singapore)
Peter Quax (Universiteit Hasselt)

**Contributor and Reviewer**
Patrik Österberg (Mid Sweden University)

# Brief Contents

*This page is intentionally left blank*

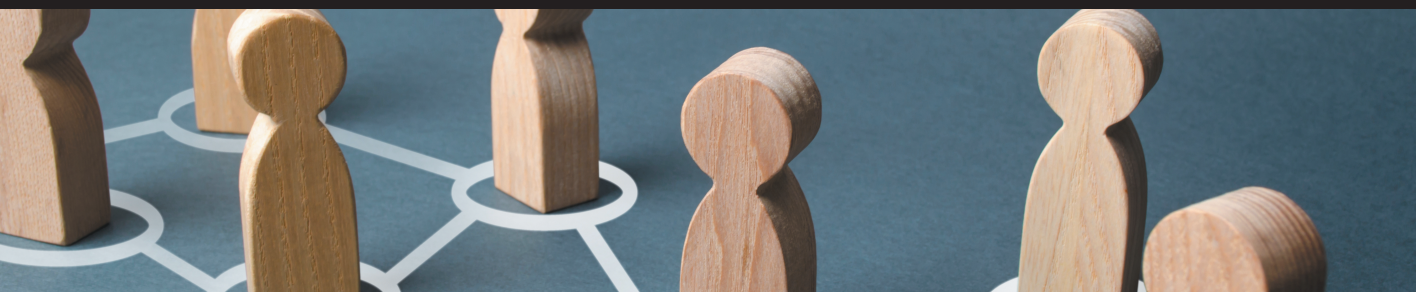
# Table of Contents

*This page is intentionally left blank*

# COMPUTER NETWORKING

EIGHTH EDITION
GLOBAL EDITION

## A Top-Down Approach



*This page is intentionally left blank*

CHAPTER 1

# Computer Networks and the Internet

Today's Internet is arguably the largest engineered system ever created by mankind, with hundreds of millions of connected computers, communication links, and switches; with billions of users who connect via laptops, tablets, and smartphones; and with an array of new Internet-connected "things" including game consoles, surveillance systems, watches, eye glasses, thermostats, and cars. Given that the Internet is so large and has so many diverse components and uses, is there any hope of understanding how it works? Are there guiding principles and structure that can provide a foundation for understanding such an amazingly large and complex system? And if so, is it possible that it actually could be both interesting *and* fun to learn about computer networks? Fortunately, the answer to all of these questions is a resounding YES! Indeed, it's our aim in this book to provide you with a modern introduction to the dynamic field of computer networking, giving you the principles and practical insights you'll need to understand not only today's networks, but tomorrow's as well.

This first chapter presents a broad overview of computer networking and the Internet. Our goal here is to paint a broad picture and set the context for the rest of this book, to see the forest through the trees. We'll cover a lot of ground in this introductory chapter and discuss a lot of the pieces of a computer network, without losing sight of the big picture.

We'll structure our overview of computer networks in this chapter as follows. After introducing some basic terminology and concepts, we'll first examine the basic hardware and software components that make up a network. We'll begin at the network's edge and look at the end systems and network applications running in the network. We'll then explore the core of a computer network, examining the links

and the switches that transport data, as well as the access networks and physical media that connect end systems to the network core. We'll learn that the Internet is a network of networks, and we'll learn how these networks connect with each other.

After having completed this overview of the edge and core of a computer network, we'll take the broader and more abstract view in the second half of this chapter. We'll examine delay, loss, and throughput of data in a computer network and provide simple quantitative models for end-to-end throughput and delay: models that take into account transmission, propagation, and queuing delays. We'll then introduce some of the key architectural principles in computer networking, namely, protocol layering and service models. We'll also learn that computer networks are vulnerable to many different types of attacks; we'll survey some of these attacks and consider how computer networks can be made more secure. Finally, we'll close this chapter with a brief history of computer networking.

## 1.1 What Is the Internet?

In this book, we'll use the public Internet, a specific computer network, as our principal vehicle for discussing computer networks and their protocols. But what *is* the Internet? There are a couple of ways to answer this question. First, we can describe the nuts and bolts of the Internet, that is, the basic hardware and software components that make up the Internet. Second, we can describe the Internet in terms of a networking infrastructure that provides services to distributed applications. Let's begin with the nuts-and-bolts description, using Figure 1.1 to illustrate our discussion.

### 1.1.1 A Nuts-and-Bolts Description

The Internet is a computer network that interconnects billions of computing devices throughout the world. Not too long ago, these computing devices were primarily traditional desktop computers, Linux workstations, and so-called servers that store and transmit information such as Web pages and e-mail messages. Increasingly, however, users connect to the Internet with smartphones and tablets—today, close to half of the world's population are active mobile Internet users with the percentage expected to increase to 75% by 2025 [Statista 2019]. Furthermore, nontraditional Internet "things" such as TVs, gaming consoles, thermostats, home security systems, home appliances, watches, eye glasses, cars, traffic control systems, and more are being connected to the Internet. Indeed, the term *computer network* is beginning to sound a bit dated, given the many nontraditional devices that are being hooked up to the Internet. In Internet jargon, all of these devices are called **hosts** or **end systems**. By some estimates, there were about 18 billion devices connected to the Internet in 2017, and the number will reach 28.5 billion by 2022 [Cisco VNI 2020].

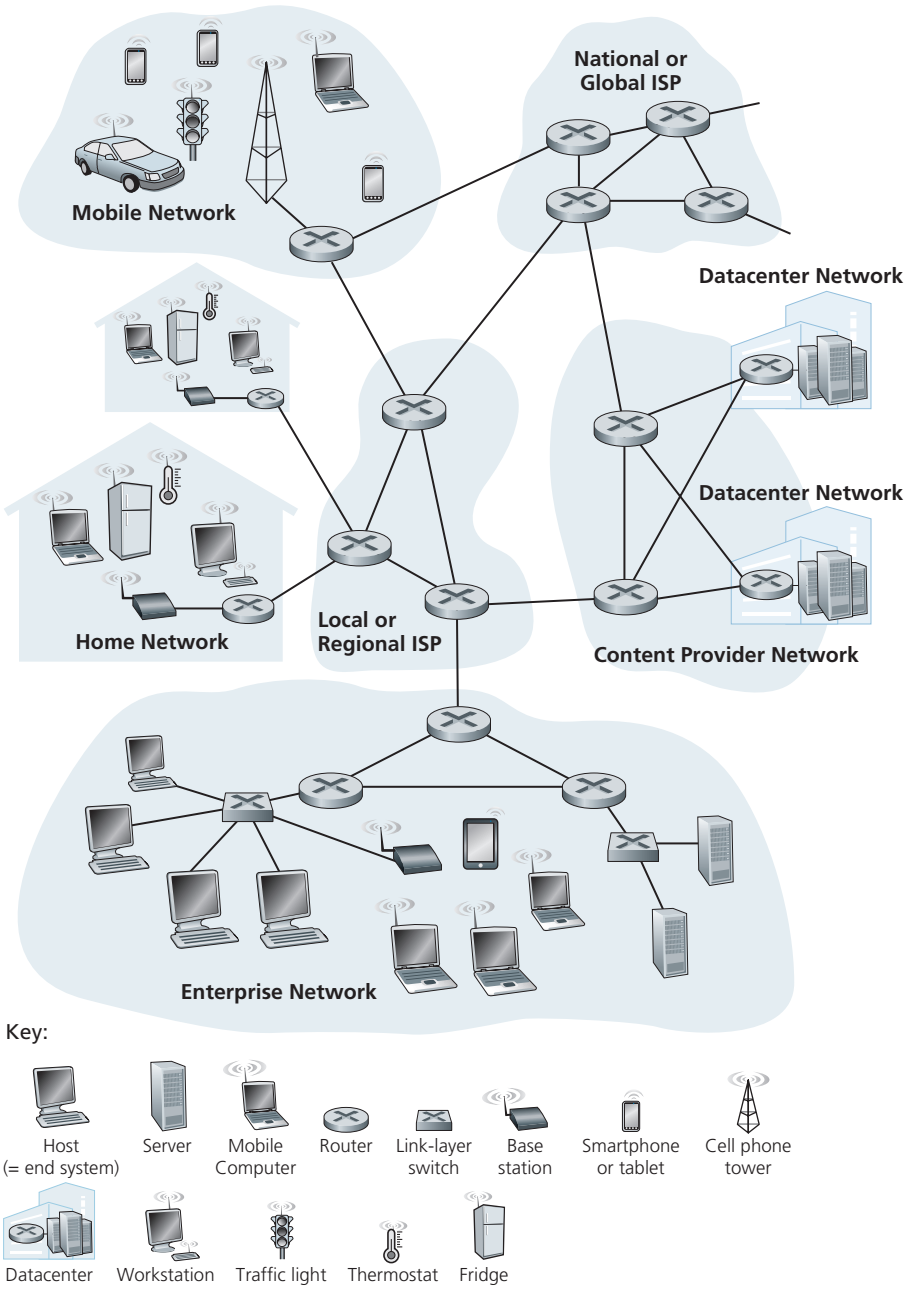



**Figure 1.1** ♦ Some pieces of the Internet

End systems are connected together by a network of **communication links** and **packet switches**. We'll see in Section 1.2 that there are many types of communication links, which are made up of different types of physical media, including coaxial cable, copper wire, optical fiber, and radio spectrum. Different links can transmit data at different rates, with the **transmission rate** of a link measured in bits/second. When one end system has data to send to another end system, the sending end system segments the data and adds header bytes to each segment. The resulting packages of information, known as **packets** in the jargon of computer networks, are then sent through the network to the destination end system, where they are reassembled into the original data.

A packet switch takes a packet arriving on one of its incoming communication links and forwards that packet on one of its outgoing communication links. Packet switches come in many shapes and flavors, but the two most prominent types in today's Internet are **routers** and **link-layer switches**. Both types of switches forward packets toward their ultimate destinations. Link-layer switches are typically used in access networks, while routers are typically used in the network core. The sequence of communication links and packet switches traversed by a packet from the sending end system to the receiving end system is known as a **route** or **path** through the network. Cisco predicts annual global IP traffic will reach nearly five zettabytes ($10^{21}$ bytes) by 2022 [Cisco VNI 2020].

Packet-switched networks (which transport packets) are in many ways similar to transportation networks of highways, roads, and intersections (which transport vehicles). Consider, for example, a factory that needs to move a large amount of cargo to some destination warehouse located thousands of kilometers away. At the factory, the cargo is segmented and loaded into a fleet of trucks. Each of the trucks then independently travels through the network of highways, roads, and intersections to the destination warehouse. At the destination warehouse, the cargo is unloaded and grouped with the rest of the cargo arriving from the same shipment. Thus, in many ways, packets are analogous to trucks, communication links are analogous to highways and roads, packet switches are analogous to intersections, and end systems are analogous to buildings. Just as a truck takes a path through the transportation network, a packet takes a path through a computer network.

End systems access the Internet through **Internet Service Providers (ISPs)**, including residential ISPs such as local cable or telephone companies; corporate ISPs; university ISPs; ISPs that provide WiFi access in airports, hotels, coffee shops, and other public places; and cellular data ISPs, providing mobile access to our smartphones and other devices. Each ISP is in itself a network of packet switches and communication links. ISPs provide a variety of types of network access to the end systems, including residential broadband access such as cable modem or DSL, high-speed local area network access, and mobile wireless access. ISPs also provide Internet access to content providers, connecting servers directly to the Internet. The Internet is all about connecting end systems to each other, so the

ISPs that provide access to end systems must also be interconnected. These lower-tier ISPs are thus interconnected through national and international upper-tier ISPs and these upper-tier ISPs are connected directly to each other. An upper-tier ISP consists of high-speed routers interconnected with high-speed fiber-optic links. Each ISP network, whether upper-tier or lower-tier, is managed independently, runs the IP protocol (see below), and conforms to certain naming and address conventions. We'll examine ISPs and their interconnection more closely in Section 1.3.

End systems, packet switches, and other pieces of the Internet run **protocols** that control the sending and receiving of information within the Internet. The **Transmission Control Protocol (TCP)** and the **Internet Protocol (IP)** are two of the most important protocols in the Internet. The IP protocol specifies the format of the packets that are sent and received among routers and end systems. The Internet's principal protocols are collectively known as **TCP/IP**. We'll begin looking into protocols in this introductory chapter. But that's just a start—much of this book is concerned with networking protocols!

Given the importance of protocols to the Internet, it's important that everyone agree on what each and every protocol does, so that people can create systems and products that interoperate. This is where standards come into play. **Internet standards** are developed by the Internet Engineering Task Force (IETF) [IETF 2020]. The IETF standards documents are called **requests for comments (RFCs)**. RFCs started out as general requests for comments (hence the name) to resolve network and protocol design problems that faced the precursor to the Internet [Allman 2011]. RFCs tend to be quite technical and detailed. They define protocols such as TCP, IP, HTTP (for the Web), and SMTP (for e-mail). There are currently nearly 9000 RFCs. Other bodies also specify standards for network components, most notably for network links. The IEEE 802 LAN Standards Committee [IEEE 802 2020], for example, specifies the Ethernet and wireless WiFi standards.

### 1.1.2 A Services Description

Our discussion above has identified many of the pieces that make up the Internet. But we can also describe the Internet from an entirely different angle—namely, as *an infrastructure that provides services to applications*. In addition to traditional applications such as e-mail and Web surfing, Internet applications include mobile smartphone and tablet applications, including Internet messaging, mapping with real-time road-traffic information, music streaming movie and television streaming, online social media, video conferencing, multi-person games, and location-based recommendation systems. The applications are said to be **distributed applications**, since they involve multiple end systems that exchange data with each other. Importantly, Internet applications run on end systems—they do not run in the packet switches in the network core. Although packet switches facilitate the exchange of data among end systems, they are not concerned with the application that is the source or sink of data.

Let's explore a little more what we mean by an infrastructure that provides services to applications. To this end, suppose you have an exciting new idea for a distributed Internet application, one that may greatly benefit humanity or one that may simply make you rich and famous. How might you go about transforming this idea into an actual Internet application? Because applications run on end systems, you are going to need to write programs that run on the end systems. You might, for example, write your programs in Java, C, or Python. Now, because you are developing a distributed Internet application, the programs running on the different end systems will need to send data to each other. And here we get to a central issue—one that leads to the alternative way of describing the Internet as a platform for applications. How does one program running on one end system instruct the Internet to deliver data to another program running on another end system?

End systems attached to the Internet provide a **socket interface** that specifies how a program running on one end system asks the Internet infrastructure to deliver data to a specific destination program running on another end system. This Internet socket interface is a set of rules that the sending program must follow so that the Internet can deliver the data to the destination program. We'll discuss the Internet socket interface in detail in Chapter 2. For now, let's draw upon a simple analogy, one that we will frequently use in this book. Suppose Alice wants to send a letter to Bob using the postal service. Alice, of course, can't just write the letter (the data) and drop the letter out her window. Instead, the postal service requires that Alice put the letter in an envelope; write Bob's full name, address, and zip code in the center of the envelope; seal the envelope; put a stamp in the upper-right-hand corner of the envelope; and finally, drop the envelope into an official postal service mailbox. Thus, the postal service has its own "postal service interface," or set of rules, that Alice must follow to have the postal service deliver her letter to Bob. In a similar manner, the Internet has a socket interface that the program sending data must follow to have the Internet deliver the data to the program that will receive the data.

The postal service, of course, provides more than one service to its customers. It provides express delivery, reception confirmation, ordinary use, and many more services. In a similar manner, the Internet provides multiple services to its applications. When you develop an Internet application, you too must choose one of the Internet's services for your application. We'll describe the Internet's services in Chapter 2.

We have just given two descriptions of the Internet; one in terms of its hardware and software components, the other in terms of an infrastructure for providing services to distributed applications. But perhaps you are still confused as to what the Internet is. What are packet switching and TCP/IP? What are routers? What kinds of communication links are present in the Internet? What is a distributed application? How can a thermostat or body scale be attached to the Internet? If you feel a bit overwhelmed by all of this now, don't worry—the purpose of this book is to introduce you to both the nuts and bolts of the Internet and the principles that govern how and

why it works. We'll explain these important terms and questions in the following sections and chapters.

## 1.1.3 What Is a Protocol?

Now that we've got a bit of a feel for what the Internet is, let's consider another important buzzword in computer networking: *protocol*. What is a protocol? What does a protocol do?

### A Human Analogy

It is probably easiest to understand the notion of a computer network protocol by first considering some human analogies, since we humans execute protocols all of the time. Consider what you do when you want to ask someone for the time of day. A typical exchange is shown in Figure 1.2. Human protocol (or good manners, at

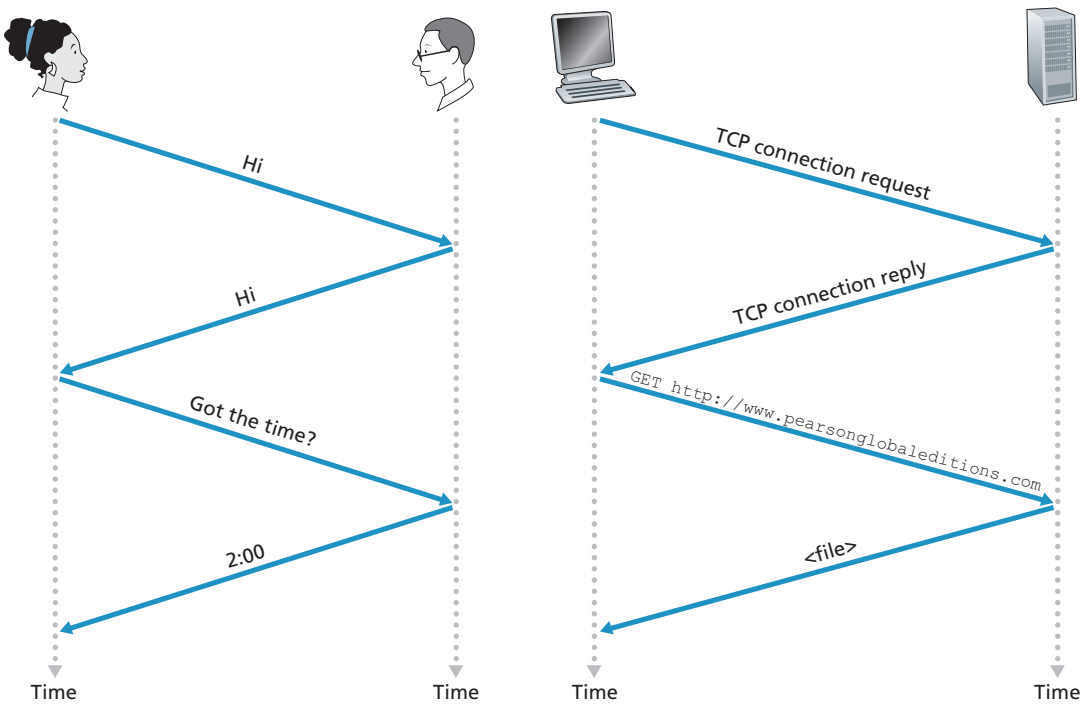



**Figure 1.2** ♦ A human protocol and a computer network protocol

least) dictates that one first offer a greeting (the first "Hi" in Figure 1.2) to initiate communication with someone else. The typical response to a "Hi" is a returned "Hi" message. Implicitly, one then takes a cordial "Hi" response as an indication that one can proceed and ask for the time of day. A different response to the initial "Hi" (such as "Don't bother me!" or "I don't speak English," or some unprintable reply) might indicate an unwillingness or inability to communicate. In this case, the human protocol would be not to ask for the time of day. Sometimes one gets no response at all to a question, in which case one typically gives up asking that person for the time. Note that in our human protocol, *there are specific messages we send, and specific actions we take in response to the received reply messages or other events* (such as no reply within some given amount of time). Clearly, transmitted and received messages, and actions taken when these messages are sent or received or other events occur, play a central role in a human protocol. If people run different protocols (for example, if one person has manners but the other does not, or if one understands the concept of time and the other does not) the protocols do not interoperate and no useful work can be accomplished. The same is true in networking—it takes two (or more) communicating entities running the same protocol in order to accomplish a task.

Let's consider a second human analogy. Suppose you're in a college class (a computer networking class, for example!). The teacher is droning on about protocols and you're confused. The teacher stops to ask, "Are there any questions?" (a message that is transmitted to, and received by, all students who are not sleeping). You raise your hand (transmitting an implicit message to the teacher). Your teacher acknowledges you with a smile, saying "Yes . . ." (a transmitted message encouraging you to ask your question—teachers *love* to be asked questions), and you then ask your question (that is, transmit your message to your teacher). Your teacher hears your question (receives your question message) and answers (transmits a reply to you). Once again, we see that the transmission and receipt of messages, and a set of conventional actions taken when these messages are sent and received, are at the heart of this question-and-answer protocol.

## Network Protocols

A network protocol is similar to a human protocol, except that the entities exchanging messages and taking actions are hardware or software components of some device (for example, computer, smartphone, tablet, router, or other network-capable device). All activity in the Internet that involves two or more communicating remote entities is governed by a protocol. For example, hardware-implemented protocols in two physically connected computers control the flow of bits on the "wire" between the two network interface cards; congestion-control protocols in end systems control the rate at which packets are transmitted between sender and receiver; protocols in routers determine a packet's path from source to destination. Protocols are running

everywhere in the Internet, and consequently much of this book is about computer network protocols.

As an example of a computer network protocol with which you are probably familiar, consider what happens when you make a request to a Web server, that is, when you type the URL of a Web page into your Web browser. The scenario is illustrated in the right half of Figure 1.2. First, your computer will send a connection request message to the Web server and wait for a reply. The Web server will eventually receive your connection request message and return a connection reply message. Knowing that it is now OK to request the Web document, your computer then sends the name of the Web page it wants to fetch from that Web server in a GET message. Finally, the Web server returns the Web page (file) to your computer.

Given the human and networking examples above, the exchange of messages and the actions taken when these messages are sent and received are the key defining elements of a protocol:

> *A **protocol** defines the format and the order of messages exchanged between two or more communicating entities, as well as the actions taken on the transmission and/or receipt of a message or other event.*

The Internet, and computer networks in general, make extensive use of protocols. Different protocols are used to accomplish different communication tasks. As you read through this book, you will learn that some protocols are simple and straightforward, while others are complex and intellectually deep. Mastering the field of computer networking is equivalent to understanding the what, why, and how of networking protocols.

## 1.2 The Network Edge

In the previous section, we presented a high-level overview of the Internet and networking protocols. We are now going to delve a bit more deeply into the components of the Internet. We begin in this section at the edge of the network and look at the components with which we are most familiar—namely, the computers, smartphones and other devices that we use on a daily basis. In the next section, we'll move from the network edge to the network core and examine switching and routing in computer networks.

Recall from the previous section that in computer networking jargon, the computers and other devices connected to the Internet are often referred to as end systems. They are referred to as end systems because they sit at the edge of the Internet, as shown in Figure 1.3. The Internet's end systems include desktop computers

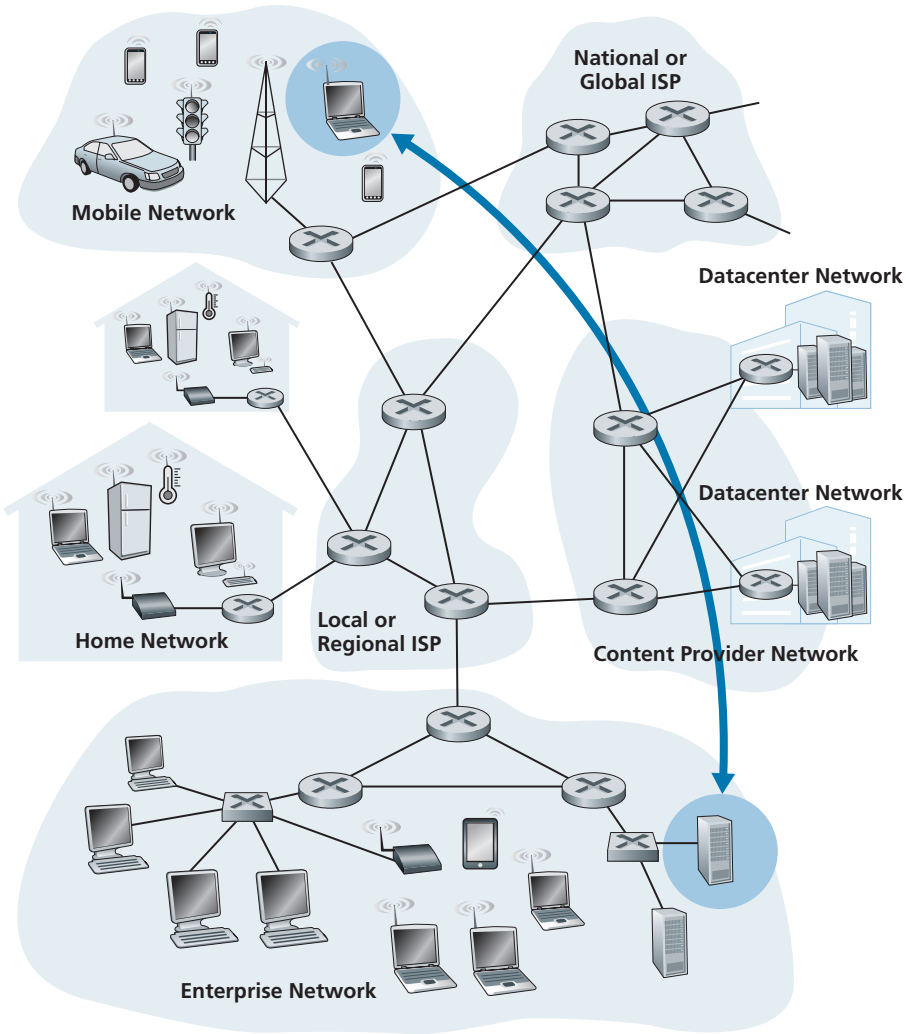



**Figure 1.3** ♦ End-system interaction

(e.g., desktop PCs, Macs, and Linux boxes), servers (e.g., Web and e-mail servers), and mobile devices (e.g., laptops, smartphones, and tablets). Furthermore, an increasing number of non-traditional "things" are being attached to the Internet as end systems (see the Case History feature).

End systems are also referred to as *hosts* because they host (that is, run) application programs such as a Web browser program, a Web server program, an e-mail

## CASE HISTORY

### DATA CENTERS AND CLOUD COMPUTING

Internet companies such as Google, Microsoft, Amazon, and Alibaba have built massive data centers, each housing tens to hundreds of thousands of hosts. These data centers are not only connected to the Internet, as shown in Figure 1.1, but also internally include complex computer networks that interconnect the datacenter's hosts. The data centers are the engines behind the Internet applications that we use on a daily basis.

Broadly speaking, data centers serve three purposes, which we describe here in the context of Amazon for concreteness. First, they serve Amazon e-commerce pages to users, for example, pages describing products and purchase information. Second, they serve as massively parallel computing infrastructures for Amazon-specific data processing tasks. Third, they provide **cloud computing** to other companies. Indeed, today a major trend in computing is for companies to use a cloud provider such as Amazon to handle essentially *all* of their IT needs. For example, Airbnb and many other Internet-based companies do not own and manage their own data centers but instead run their entire Web-based services in the Amazon cloud, called Amazon Web Services (AWS).

The worker bees in a data center are the hosts. They serve content (e.g., Web pages and videos), store e-mails and documents, and collectively perform massively distributed computations. The hosts in data centers, called blades and resembling pizza boxes, are generally commodity hosts that include CPU, memory, and disk storage. The hosts are stacked in racks, with each rack typically having 20 to 40 blades. The racks are then interconnected using sophisticated and evolving data center network designs. Data center networks are discussed in greater detail in Chapter 6.

client program, or an e-mail server program. Throughout this book we will use the terms hosts and end systems interchangeably; that is, *host = end system*. Hosts are sometimes further divided into two categories: **clients** and **servers**. Informally, clients tend to be desktops, laptops, smartphones, and so on, whereas servers tend to be more powerful machines that store and distribute Web pages, stream video, relay e-mail, and so on. Today, most of the servers from which we receive search results, e-mail, Web pages, videos and mobile app content reside in large **data centers**. For example, as of 2020, Google has 19 data centers on four continents, collectively containing several million servers. Figure 1.3 includes two such data centers, and the Case History sidebar describes data centers in more detail.

### 1.2.1 Access Networks

Having considered the applications and end systems at the "edge of the network," let's next consider the access network—the network that physically connects an end system to the first router (also known as the "edge router") on a path from the end system to any other distant end system. Figure 1.4 shows several types of access

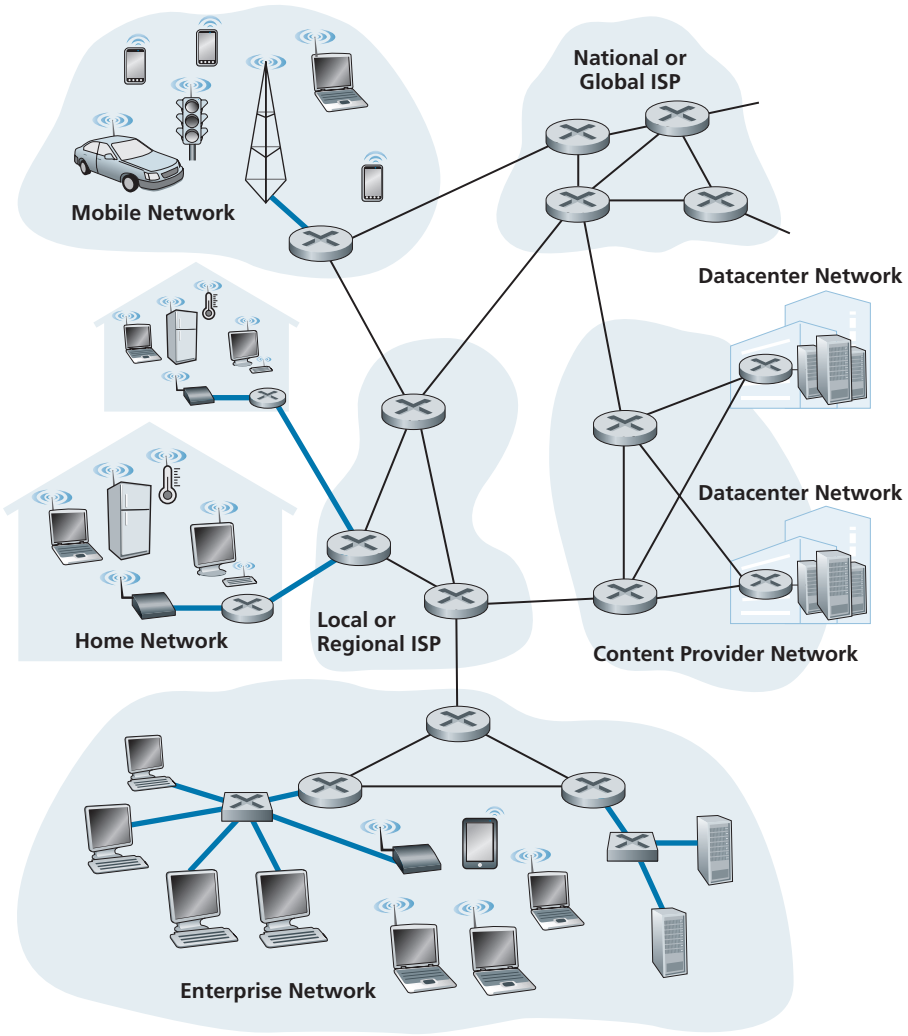



**Figure 1.4** ♦ Access networks

networks with thick, shaded lines and the settings (home, enterprise, and wide-area mobile wireless) in which they are used.

## Home Access: DSL, Cable, FTTH, and 5G Fixed Wireless

As of 2020, more than 80% of the households in Europe and the USA have Internet access [Statista 2019]. Given this widespread use of home access networks let's begin our overview of access networks by considering how homes connect to the Internet.

Today, the two most prevalent types of broadband residential access are **digital subscriber line (DSL)** and cable. A residence typically obtains DSL Internet access from the same local telephone company (telco) that provides its wired local phone access. Thus, when DSL is used, a customer's telco is also its ISP. As shown in Figure 1.5, each customer's DSL modem uses the existing telephone line exchange data with a digital subscriber line access multiplexer (DSLAM) located in the telco's local central office (CO). The home's DSL modem takes digital data and translates it to high-frequency tones for transmission over telephone wires to the CO; the analog signals from many such houses are translated back into digital format at the DSLAM.

The residential telephone line carries both data and traditional telephone signals simultaneously, which are encoded at different frequencies:

- A high-speed downstream channel, in the 50 kHz to 1 MHz band
- A medium-speed upstream channel, in the 4 kHz to 50 kHz band
- An ordinary two-way telephone channel, in the 0 to 4 kHz band

This approach makes the single DSL link appear as if there were three separate links, so that a telephone call and an Internet connection can share the DSL link at

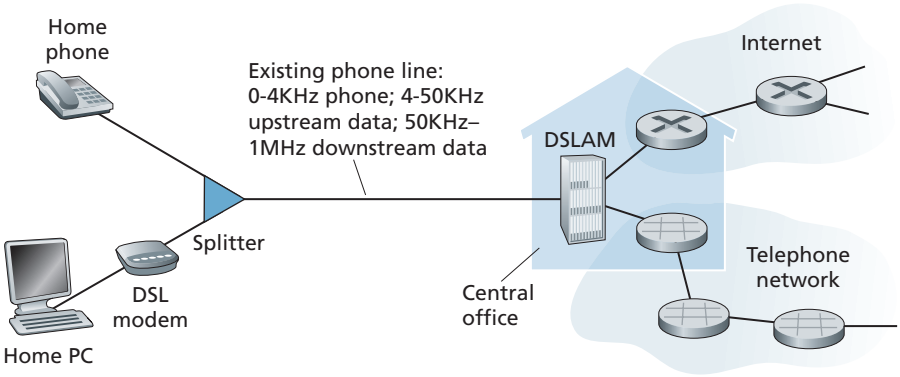



**Figure 1.5** ♦ DSL Internet access

the same time. (We'll describe this technique of frequency-division multiplexing in Section 1.3.1.) On the customer side, a splitter separates the data and telephone signals arriving to the home and forwards the data signal to the DSL modem. On the telco side, in the CO, the DSLAM separates the data and phone signals and sends the data into the Internet. Hundreds or even thousands of households connect to a single DSLAM.

The DSL standards define multiple transmission rates, including downstream transmission rates of 24 Mbs and 52 Mbs, and upstream rates of 3.5 Mbps and 16 Mbps; the newest standard provides for aggregate upstream plus downstream rates of 1 Gbps [ITU 2014]. Because the downstream and upstream rates are different, the access is said to be asymmetric. The actual downstream and upstream transmission rates achieved may be less than the rates noted above, as the DSL provider may purposefully limit a residential rate when tiered service (different rates, available at different prices) are offered. The maximum rate is also limited by the distance between the home and the CO, the gauge of the twisted-pair line and the degree of electrical interference. Engineers have expressly designed DSL for short distances between the home and the CO; generally, if the residence is not located within 5 to 10 miles of the CO, the residence must resort to an alternative form of Internet access.

While DSL makes use of the telco's existing local telephone infrastructure, **cable Internet access** makes use of the cable television company's existing cable television infrastructure. A residence obtains cable Internet access from the same company that provides its cable television. As illustrated in Figure 1.6, fiber optics

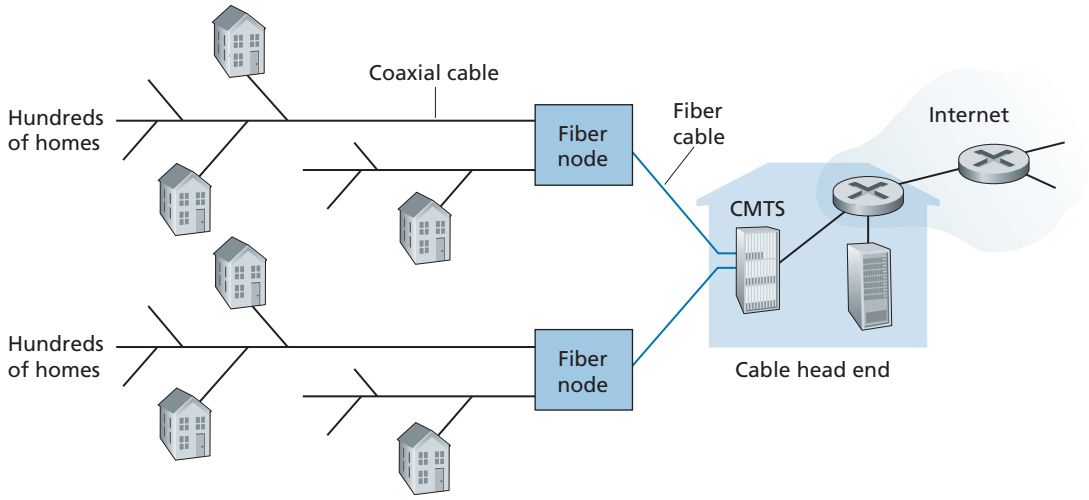



**Figure 1.6** ♦ A hybrid fiber-coaxial access network

connect the cable head end to neighborhood-level junctions, from which traditional coaxial cable is then used to reach individual houses and apartments. Each neighborhood junction typically supports 500 to 5,000 homes. Because both fiber and coaxial cable are employed in this system, it is often referred to as hybrid fiber coax (HFC).

Cable internet access requires special modems, called cable modems. As with a DSL modem, the cable modem is typically an external device and connects to the home PC through an Ethernet port. (We will discuss Ethernet in great detail in Chapter 6.) At the cable head end, the cable modem termination system (CMTS) serves a similar function as the DSL network's DSLAM—turning the analog signal sent from the cable modems in many downstream homes back into digital format. Cable modems divide the HFC network into two channels, a downstream and an upstream channel. As with DSL, access is typically asymmetric, with the downstream channel typically allocated a higher transmission rate than the upstream channel. The DOCSIS 2.0 and 3.0 standards define downstream bitrates of 40 Mbps and 1.2 Gbps, and upstream rates of 30 Mbps and 100 Mbps, respectively. As in the case of DSL networks, the maximum achievable rate may not be realized due to lower contracted data rates or media impairments.

One important characteristic of cable Internet access is that it is a shared broadcast medium. In particular, every packet sent by the head end travels downstream on every link to every home and every packet sent by a home travels on the upstream channel to the head end. For this reason, if several users are simultaneously downloading a video file on the downstream channel, the actual rate at which each user receives its video file will be significantly lower than the aggregate cable downstream rate. On the other hand, if there are only a few active users and they are all Web surfing, then each of the users may actually receive Web pages at the full cable downstream rate, because the users will rarely request a Web page at exactly the same time. Because the upstream channel is also shared, a distributed multiple access protocol is needed to coordinate transmissions and avoid collisions. (We'll discuss this collision issue in some detail in Chapter 6.)

Although DSL and cable networks currently represent the majority of residential broadband access in the United States, an up-and-coming technology that provides even higher speeds is **fiber to the home (FTTH)** [Fiber Broadband 2020]. As the name suggests, the FTTH concept is simple—provide an optical fiber path from the CO directly to the home. FTTH can potentially provide Internet access rates in the gigabits per second range.

There are several competing technologies for optical distribution from the CO to the homes. The simplest optical distribution network is called direct fiber, with one fiber leaving the CO for each home. More commonly, each fiber leaving the central office is actually shared by many homes; it is not until the fiber gets relatively close to the homes that it is split into individual customer-specific fibers. There are two competing optical-distribution network architectures that perform

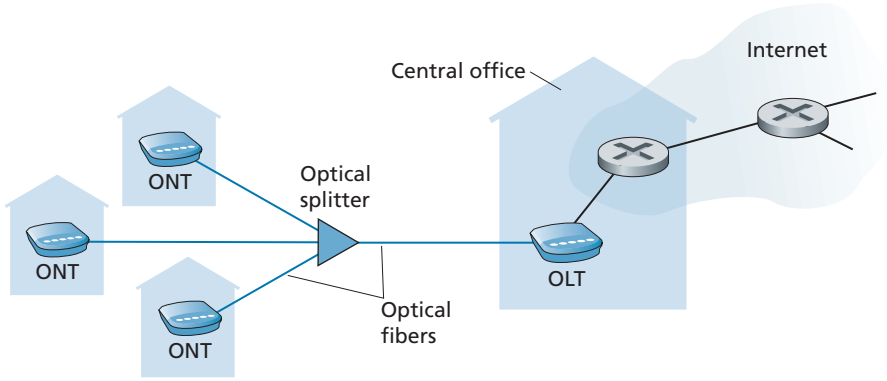



**Figure 1.7** ♦ FTTH Internet access

this splitting: active optical networks (AONs) and passive optical networks (PONs). AON is essentially switched Ethernet, which is discussed in Chapter 6.

Here, we briefly discuss PON, which is used in Verizon's FiOS service. Figure 1.7 shows FTTH using the PON distribution architecture. Each home has an optical network terminator (ONT), which is connected by dedicated optical fiber to a neighborhood splitter. The splitter combines a number of homes (typically less than 100) onto a single, shared optical fiber, which connects to an optical line terminator (OLT) in the telco's CO. The OLT, providing conversion between optical and electrical signals, connects to the Internet via a telco router. At home, users connect a home router (typically a wireless router) to the ONT and access the Internet via this home router. In the PON architecture, all packets sent from OLT to the splitter are replicated at the splitter (similar to a cable head end).

In addition to DSL, Cable, and FTTH, **5G fixed wireless** is beginning to be deployed. 5G fixed wireless not only promises high-speed residential access, but will do so without installing costly and failure-prone cabling from the telco's CO to the home. With 5G fixed wireless, using beam-forming technology, data is sent wirelessly from a provider's base station to the a modem in the home. A WiFi wireless router is connected to the modem (possibly bundled together), similar to how a WiFi wireless router is connected to a cable or DSL modem. 5G cellular networks are covered in Chapter 7.

### Access in the Enterprise (and the Home): Ethernet and WiFi

On corporate and university campuses, and increasingly in home settings, a local area network (LAN) is used to connect an end system to the edge router. Although there are many types of LAN technologies, Ethernet is by far the most prevalent access technology in corporate, university, and home networks. As shown in

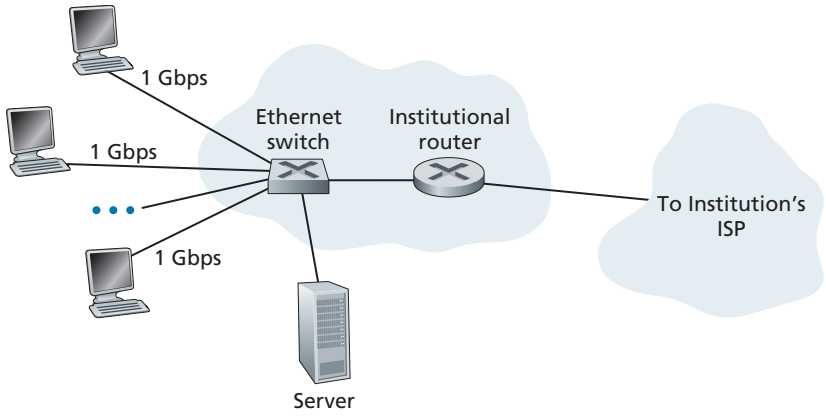



**Figure 1.8** ♦ Ethernet Internet access

Figure 1.8, Ethernet users use twisted-pair copper wire to connect to an Ethernet switch, a technology discussed in detail in Chapter 6. The Ethernet switch, or a network of such interconnected switches, is then in turn connected into the larger Internet. With Ethernet access, users typically have 100 Mbps to tens of Gbps access to the Ethernet switch, whereas servers may have 1 Gbps 10 Gbps access.

Increasingly, however, people are accessing the Internet wirelessly from laptops, smartphones, tablets, and other "things". In a wireless LAN setting, wireless users transmit/receive packets to/from an access point that is connected into the enterprise's network (most likely using wired Ethernet), which in turn is connected to the wired Internet. A wireless LAN user must typically be within a few tens of meters of the access point. Wireless LAN access based on IEEE 802.11 technology, more colloquially known as WiFi, is now just about everywhere—universities, business offices, cafes, airports, homes, and even in airplanes. As discussed in detail in Chapter 7, 802.11 today provides a shared transmission rate of up to more than 100 Mbps.

Even though Ethernet and WiFi access networks were initially deployed in enterprise (corporate, university) settings, they are also common components of home networks. Many homes combine broadband residential access (that is, cable modems or DSL) with these inexpensive wireless LAN technologies to create powerful home networks Figure 1.9 shows a typical home network. This home network consists of a roaming laptop, multiple Internet-connected home appliances, as well as a wired PC; a base station (the wireless access point), which communicates with the wireless PC and other wireless devices in the home; and a home router that connects the wireless access point, and any other wired home devices, to the Internet. This network allows household members to have broadband access to the Internet with one member roaming from the kitchen to the backyard to the bedrooms.

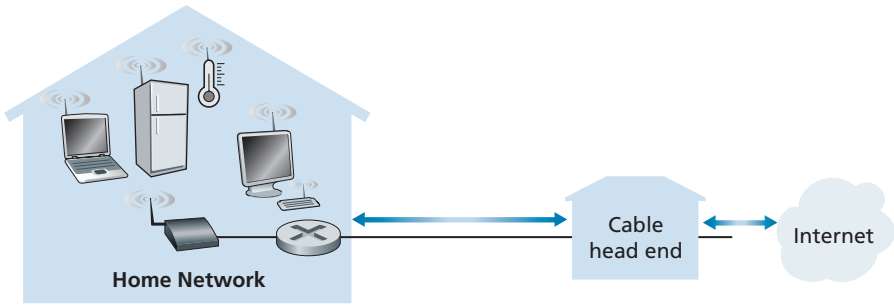



**Figure 1.9** ♦ A typical home network

### Wide-Area Wireless Access: 3G and LTE 4G and 5G

Mobile devices such as iPhones and Android devices are being used to message, share photos in social networks, make mobile payments, watch movies, stream music, and much more while on the run. These devices employ the same wireless infrastructure used for cellular telephony to send/receive packets through a base station that is operated by the cellular network provider. Unlike WiFi, a user need only be within a few tens of kilometers (as opposed to a few tens of meters) of the base station.

Telecommunications companies have made enormous investments in so-called fourth-generation (4G) wireless, which provides real-world download speeds of up to 60 Mbps. But even higher-speed wide-area access technologies—a fifth-generation (5G) of wide-area wireless networks—are already being deployed. We'll cover the basic principles of wireless networks and mobility, as well as WiFi, 4G and 5G technologies (and more!) in Chapter 7.

## 1.2.2 Physical Media

In the previous subsection, we gave an overview of some of the most important network access technologies in the Internet. As we described these technologies, we also indicated the physical media used. For example, we said that HFC uses a combination of fiber cable and coaxial cable. We said that DSL and Ethernet use copper wire. And we said that mobile access networks use the radio spectrum. In this subsection, we provide a brief overview of these and other transmission media that are commonly used in the Internet.

In order to define what is meant by a physical medium, let us reflect on the brief life of a bit. Consider a bit traveling from one end system, through a series of links and routers, to another end system. This poor bit gets kicked around and transmitted many, many times! The source end system first transmits the

bit, and shortly thereafter the first router in the series receives the bit; the first router then transmits the bit, and shortly thereafter the second router receives the bit; and so on. Thus our bit, when traveling from source to destination, passes through a series of transmitter-receiver pairs. For each transmitter-receiver pair, the bit is sent by propagating electromagnetic waves or optical pulses across a **physical medium**. The physical medium can take many shapes and forms and does not have to be of the same type for each transmitter-receiver pair along the path. Examples of physical media include twisted-pair copper wire, coaxial cable, multimode fiber-optic cable, terrestrial radio spectrum, and satellite radio spectrum. Physical media fall into two categories: **guided media** and **unguided media**. With guided media, the waves are guided along a solid medium, such as a fiber-optic cable, a twisted-pair copper wire, or a coaxial cable. With unguided media, the waves propagate in the atmosphere and in outer space, such as in a wireless LAN or a digital satellite channel.

But before we get into the characteristics of the various media types, let us say a few words about their costs. The actual cost of the physical link (copper wire, fiber-optic cable, and so on) is often relatively minor compared with other networking costs. In particular, the labor cost associated with the installation of the physical link can be orders of magnitude higher than the cost of the material. For this reason, many builders install twisted pair, optical fiber, and coaxial cable in every room in a building. Even if only one medium is initially used, there is a good chance that another medium could be used in the near future, and so money is saved by not having to lay additional wires in the future.

### Twisted-Pair Copper Wire

The least expensive and most commonly used guided transmission medium is twisted-pair copper wire. For over a hundred years it has been used by telephone networks. In fact, more than 99 percent of the wired connections from the telephone handset to the local telephone switch use twisted-pair copper wire. Most of us have seen twisted pair in our homes (or those of our parents or grandparents!) and work environments. Twisted pair consists of two insulated copper wires, each about 1 mm thick, arranged in a regular spiral pattern. The wires are twisted together to reduce the electrical interference from similar pairs close by. Typically, a number of pairs are bundled together in a cable by wrapping the pairs in a protective shield. A wire pair constitutes a single communication link. **Unshielded twisted pair (UTP)** is commonly used for computer networks within a building, that is, for LANs. Data rates for LANs using twisted pair today range from 10 Mbps to 10 Gbps. The data rates that can be achieved depend on the thickness of the wire and the distance between transmitter and receiver.

When fiber-optic technology emerged in the 1980s, many people disparaged twisted pair because of its relatively low bit rates. Some people even felt

that fiber-optic technology would completely replace twisted pair. But twisted pair did not give up so easily. Modern twisted-pair technology, such as category 6a cable, can achieve data rates of 10 Gbps for distances up to a hundred meters. In the end, twisted pair has emerged as the dominant solution for high-speed LAN networking.

As discussed earlier, twisted pair is also commonly used for residential Internet access. We saw that dial-up modem technology enables access at rates of up to 56 kbps over twisted pair. We also saw that DSL (digital subscriber line) technology has enabled residential users to access the Internet at tens of Mbps over twisted pair (when users live close to the ISP's central office).

### Coaxial Cable

Like twisted pair, coaxial cable consists of two copper conductors, but the two conductors are concentric rather than parallel. With this construction and special insulation and shielding, coaxial cable can achieve high data transmission rates. Coaxial cable is quite common in cable television systems. As we saw earlier, cable television systems have recently been coupled with cable modems to provide residential users with Internet access at rates of hundreds of Mbps. In cable television and cable Internet access, the transmitter shifts the digital signal to a specific frequency band, and the resulting analog signal is sent from the transmitter to one or more receivers. Coaxial cable can be used as a guided **shared medium**. Specifically, a number of end systems can be connected directly to the cable, with each of the end systems receiving whatever is sent by the other end systems.

### Fiber Optics

An optical fiber is a thin, flexible medium that conducts pulses of light, with each pulse representing a bit. A single optical fiber can support tremendous bit rates, up to tens or even hundreds of gigabits per second. They are immune to electromagnetic interference, have very low signal attenuation up to 100 kilometers, and are very hard to tap. These characteristics have made fiber optics the preferred long-haul guided transmission media, particularly for overseas links. Many of the long-distance telephone networks in the United States and elsewhere now use fiber optics exclusively. Fiber optics is also prevalent in the backbone of the Internet. However, the high cost of optical devices—such as transmitters, receivers, and switches—has hindered their deployment for short-haul transport, such as in a LAN or into the home in a residential access network. The Optical Carrier (OC) standard link speeds range from 51.8 Mbps to 39.8 Gbps; these specifications are often referred to as OC-$n$, where the link speed equals $n \times 51.8$ Mbps. Standards in use today include OC-1, OC-3, OC-12, OC-24, OC-48, OC-96, OC-192, OC-768.

### Terrestrial Radio Channels

Radio channels carry signals in the electromagnetic spectrum. They are an attractive medium because they require no physical wire to be installed, can penetrate walls, provide connectivity to a mobile user, and can potentially carry a signal for long distances. The characteristics of a radio channel depend significantly on the propagation environment and the distance over which a signal is to be carried. Environmental considerations determine path loss and shadow fading (which decrease the signal strength as the signal travels over a distance and around/through obstructing objects), multipath fading (due to signal reflection off of interfering objects), and interference (due to other transmissions and electromagnetic signals).

Terrestrial radio channels can be broadly classified into three groups: those that operate over very short distance (e.g., with one or two meters); those that operate in local areas, typically spanning from ten to a few hundred meters; and those that operate in the wide area, spanning tens of kilometers. Personal devices such as wireless headsets, keyboards, and medical devices operate over short distances; the wireless LAN technologies described in Section 1.2.1 use local-area radio channels; the cellular access technologies use wide-area radio channels. We'll discuss radio channels in detail in Chapter 7.

### Satellite Radio Channels

A communication satellite links two or more Earth-based microwave transmitter/receivers, known as ground stations. The satellite receives transmissions on one frequency band, regenerates the signal using a repeater (discussed below), and transmits the signal on another frequency. Two types of satellites are used in communications: **geostationary satellites** and **low-earth orbiting (LEO) satellites**.

Geostationary satellites permanently remain above the same spot on Earth. This stationary presence is achieved by placing the satellite in orbit at 36,000 kilometers above Earth's surface. This huge distance from ground station through satellite back to ground station introduces a substantial signal propagation delay of 280 milliseconds. Nevertheless, satellite links, which can operate at speeds of hundreds of Mbps, are often used in areas without access to DSL or cable-based Internet access.

LEO satellites are placed much closer to Earth and do not remain permanently above one spot on Earth. They rotate around Earth (just as the Moon does) and may communicate with each other, as well as with ground stations. To provide continuous coverage to an area, many satellites need to be placed in orbit. There are currently many low-altitude communication systems in development. LEO satellite technology may be used for Internet access sometime in the future.

## 1.3 The Network Core

Having examined the Internet's edge, let us now delve more deeply inside the network core—the mesh of packet switches and links that interconnects the Internet's end systems. Figure 1.10 highlights the network core with thick, shaded lines.

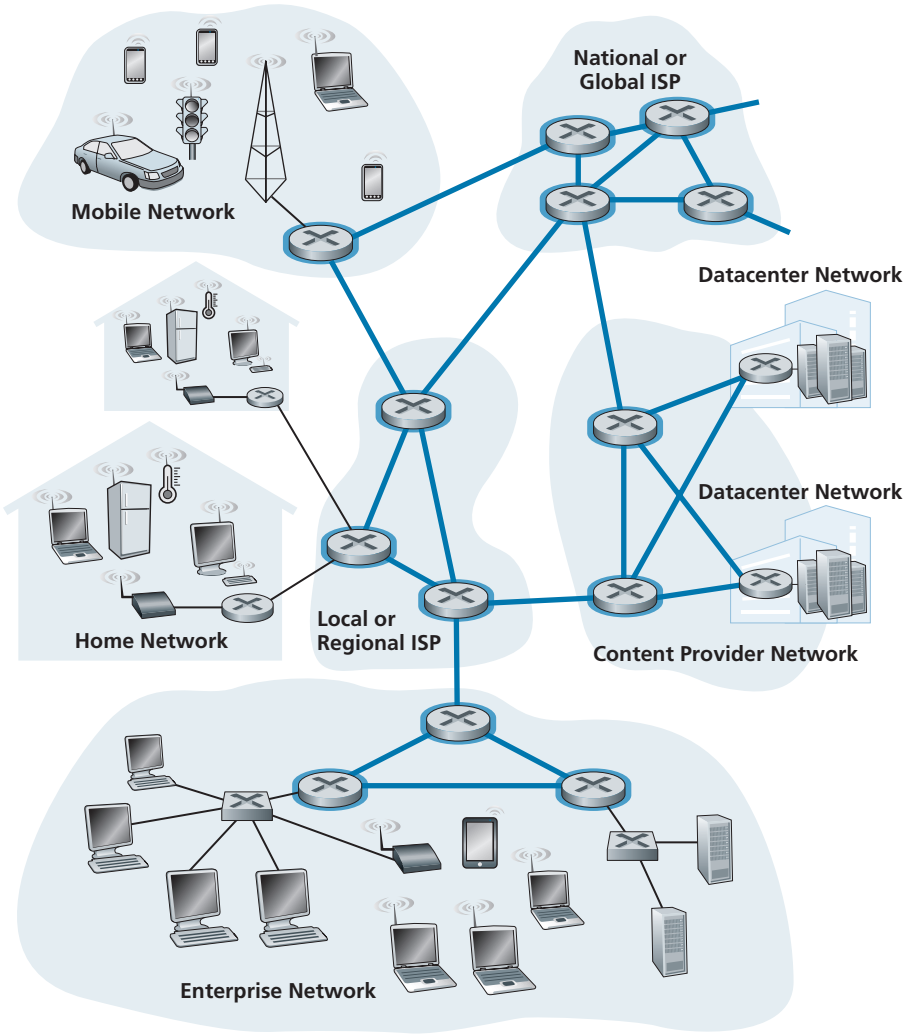



**Figure 1.10** ♦ The network core

## 1.3.1 Packet Switching

In a network application, end systems exchange **messages** with each other. Messages can contain anything the application designer wants. Messages may perform a control function (for example, the "Hi" messages in our handshaking example in Figure 1.2) or can contain data, such as an e-mail message, a JPEG image, or an MP3 audio file. To send a message from a source end system to a destination end system, the source breaks long messages into smaller chunks of data known as **packets**. Between source and destination, each packet travels through communication links and **packet switches** (for which there are two predominant types, **routers** and **link-layer switches**). Packets are transmitted over each communication link at a rate equal to the *full* transmission rate of the link. So, if a source end system or a packet switch is sending a packet of $L$ bits over a link with transmission rate $R$ bits/sec, then the time to transmit the packet is $L/R$ seconds.

### Store-and-Forward Transmission

Most packet switches use **store-and-forward transmission** at the inputs to the links. Store-and-forward transmission means that the packet switch must receive the entire packet before it can begin to transmit the first bit of the packet onto the outbound link. To explore store-and-forward transmission in more detail, consider a simple network consisting of two end systems connected by a single router, as shown in Figure 1.11. A router will typically have many incident links, since its job is to switch an incoming packet onto an outgoing link; in this simple example, the router has the rather simple task of transferring a packet from one (input) link to the only other attached link. In this example, the source has three packets, each consisting of $L$ bits, to send to the destination. At the snapshot of time shown in Figure 1.11, the source has transmitted some of packet 1, and the front of packet 1 has already arrived at the router. Because the router employs store-and-forwarding, at this instant of time, the router cannot transmit the bits it has received; instead it must first buffer (i.e., "store") the packet's bits. Only after the router has received *all* of the packet's bits can it begin to transmit (i.e., "forward") the packet onto the outbound link. To gain some insight into store-and-forward transmission, let's now calculate the amount of time that elapses from when the source begins to send the packet until the destination has received the entire packet. (Here we will ignore propagation delay—the time it takes for the bits to travel across the wire at near the speed of light—which will be discussed in Section 1.4.) The source begins to transmit at time 0; at time $L/R$ seconds, the source has transmitted the entire packet, and the entire packet has been received and stored at the router (since there is no propagation delay). At time $L/R$ seconds, since the router has just received the entire packet, it can begin to transmit the packet onto the outbound link towards the destination; at time $2L/R$, the router has transmitted the entire packet, and the entire packet has been received by the destination. Thus, the total delay is $2L/R$. If the

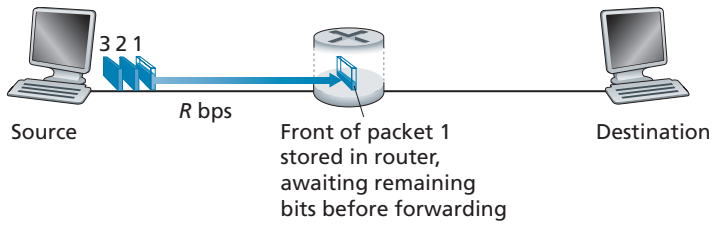



**Figure 1.11** ♦ Store-and-forward packet switching

switch instead forwarded bits as soon as they arrive (without first receiving the entire packet), then the total delay would be $L/R$ since bits are not held up at the router. But, as we will discuss in Section 1.4, routers need to receive, store, and *process* the entire packet before forwarding.

Now let's calculate the amount of time that elapses from when the source begins to send the first packet until the destination has received all three packets. As before, at time $L/R$, the router begins to forward the first packet. But also at time $L/R$ the source will begin to send the second packet, since it has just finished sending the entire first packet. Thus, at time $2L/R$, the destination has received the first packet and the router has received the second packet. Similarly, at time $3L/R$, the destination has received the first two packets and the router has received the third packet. Finally, at time $4L/R$ the destination has received all three packets!

Let's now consider the general case of sending one packet from source to destination over a path consisting of $N$ links each of rate $R$ (thus, there are $N$-1 routers between source and destination). Applying the same logic as above, we see that the end-to-end delay is:

$$d_{\text{end-to-end}} = N\frac{L}{R} \tag{1.1}$$

You may now want to try to determine what the delay would be for $P$ packets sent over a series of $N$ links.

## Queuing Delays and Packet Loss

Each packet switch has multiple links attached to it. For each attached link, the packet switch has an **output buffer** (also called an **output queue**), which stores packets that the router is about to send into that link. The output buffers play a key role in packet switching. If an arriving packet needs to be transmitted onto a link but finds the link busy with the transmission of another packet, the arriving packet must wait in the output buffer. Thus, in addition to the store-and-forward delays, packets suffer output buffer **queuing delays**. These delays are variable and depend on the

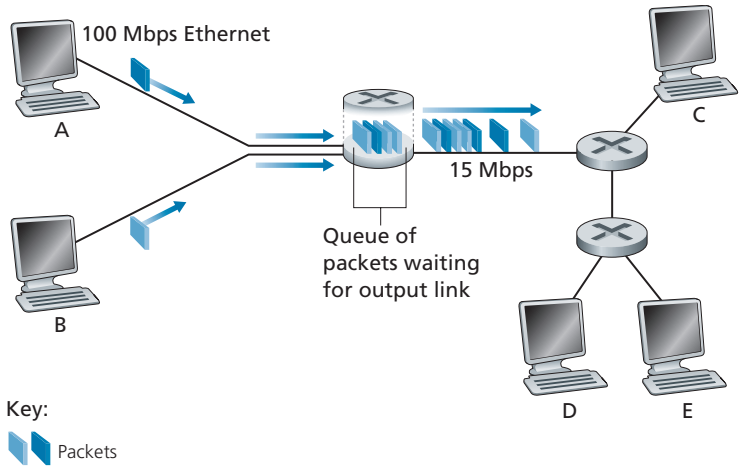



**Figure 1.12** ♦ Packet switching

level of congestion in the network. Since the amount of buffer space is finite, an arriving packet may find that the buffer is completely full with other packets waiting for transmission. In this case, **packet loss** will occur—either the arriving packet or one of the already-queued packets will be dropped.

Figure 1.12 illustrates a simple packet-switched network. As in Figure 1.11, packets are represented by three-dimensional slabs. The width of a slab represents the number of bits in the packet. In this figure, all packets have the same width and hence the same length. Suppose Hosts A and B are sending packets to Host E. Hosts A and B first send their packets along 100 Mbps Ethernet links to the first router. The router then directs these packets to the 15 Mbps link. If, during a short interval of time, the arrival rate of packets to the router (when converted to bits per second) exceeds 15 Mbps, congestion will occur at the router as packets queue in the link's output buffer before being transmitted onto the link. For example, if Host A and B each send a burst of five packets back-to-back at the same time, then most of these packets will spend some time waiting in the queue. The situation is, in fact, entirely analogous to many common-day situations—for example, when we wait in line for a bank teller or wait in front of a tollbooth. We'll examine this queuing delay in more detail in Section 1.4.

### Forwarding Tables and Routing Protocols

Earlier, we said that a router takes a packet arriving on one of its attached communication links and forwards that packet onto another one of its attached communication links. But how does the router determine which link it should

forward the packet onto? Packet forwarding is actually done in different ways in different types of computer networks. Here, we briefly describe how it is done in the Internet.

In the Internet, every end system has an address called an IP address. When a source end system wants to send a packet to a destination end system, the source includes the destination's IP address in the packet's header. As with postal addresses, this address has a hierarchical structure. When a packet arrives at a router in the network, the router examines a portion of the packet's destination address and forwards the packet to an adjacent router. More specifically, each router has a **forwarding table** that maps destination addresses (or portions of the destination addresses) to that router's outbound links. When a packet arrives at a router, the router examines the address and searches its forwarding table, using this destination address, to find the appropriate outbound link. The router then directs the packet to this outbound link.

The end-to-end routing process is analogous to a car driver who does not use maps but instead prefers to ask for directions. For example, suppose Joe is driving from Philadelphia to 156 Lakeside Drive in Orlando, Florida. Joe first drives to his neighborhood gas station and asks how to get to 156 Lakeside Drive in Orlando, Florida. The gas station attendant extracts the Florida portion of the address and tells Joe that he needs to get onto the interstate highway I-95 South, which has an entrance just next to the gas station. He also tells Joe that once he enters Florida, he should ask someone else there. Joe then takes I-95 South until he gets to Jacksonville, Florida, at which point he asks another gas station attendant for directions. The attendant extracts the Orlando portion of the address and tells Joe that he should continue on I-95 to Daytona Beach and then ask someone else. In Daytona Beach, another gas station attendant also extracts the Orlando portion of the address and tells Joe that he should take I-4 directly to Orlando. Joe takes I-4 and gets off at the Orlando exit. Joe goes to another gas station attendant, and this time the attendant extracts the Lakeside Drive portion of the address and tells Joe the road he must follow to get to Lakeside Drive. Once Joe reaches Lakeside Drive, he asks a kid on a bicycle how to get to his destination. The kid extracts the 156 portion of the address and points to the house. Joe finally reaches his ultimate destination. In the above analogy, the gas station attendants and kids on bicycles are analogous to routers.

We just learned that a router uses a packet's destination address to index a forwarding table and determine the appropriate outbound link. But this statement begs yet another question: How do forwarding tables get set? Are they configured by hand in each and every router, or does the Internet use a more automated procedure? This issue will be studied in depth in Chapter 5. But to whet your appetite here, we'll note now that the Internet has a number of special **routing protocols** that are used to automatically set the forwarding tables. A routing protocol may, for example, determine the shortest path from each router to each destination and use the shortest path results to configure the forwarding tables in the routers.

### 1.3.2 Circuit Switching

There are two fundamental approaches to moving data through a network of links and switches: **circuit switching** and **packet switching**. Having covered packet-switched networks in the previous subsection, we now turn our attention to circuit-switched networks.

In circuit-switched networks, the resources needed along a path (buffers, link transmission rate) to provide for communication between the end systems are *reserved* for the duration of the communication session between the end systems. In packet-switched networks, these resources are *not* reserved; a session's messages use the resources on demand and, as a consequence, may have to wait (that is, queue) for access to a communication link. As a simple analogy, consider two restaurants, one that requires reservations and another that neither requires reservations nor accepts them. For the restaurant that requires reservations, we have to go through the hassle of calling before we leave home. But when we arrive at the restaurant we can, in principle, immediately be seated and order our meal. For the restaurant that does not require reservations, we don't need to bother to reserve a table. But when we arrive at the restaurant, we may have to wait for a table before we can be seated.

Traditional telephone networks are examples of circuit-switched networks. Consider what happens when one person wants to send information (voice or facsimile) to another over a telephone network. Before the sender can send the information, the network must establish a connection between the sender and the receiver. This is a *bona fide* connection for which the switches on the path between the sender and receiver maintain connection state for that connection. In the jargon of telephony, this connection is called a **circuit**. When the network establishes the circuit, it also reserves a constant transmission rate in the network's links (representing a fraction of each link's transmission capacity) for the duration of the connection. Since a given transmission rate has been reserved for this sender-to-receiver connection, the sender can transfer the data to the receiver at the *guaranteed* constant rate.

Figure 1.13 illustrates a circuit-switched network. In this network, the four circuit switches are interconnected by four links. Each of these links has four circuits, so that each link can support four simultaneous connections. The hosts (for example, PCs and workstations) are each directly connected to one of the switches. When two hosts want to communicate, the network establishes a dedicated **end-to-end connection** between the two hosts. Thus, in order for Host A to communicate with Host B, the network must first reserve one circuit on each of two links. In this example, the dedicated end-to-end connection uses the second circuit in the first link and the fourth circuit in the second link. Because each link has four circuits, for each link used by the end-to-end connection, the connection gets one fourth of the link's total transmission capacity for the duration of the connection. Thus, for example, if each link between adjacent switches has a transmission rate of 1 Mbps, then each end-to-end circuit-switch connection gets 250 kbps of dedicated transmission rate.

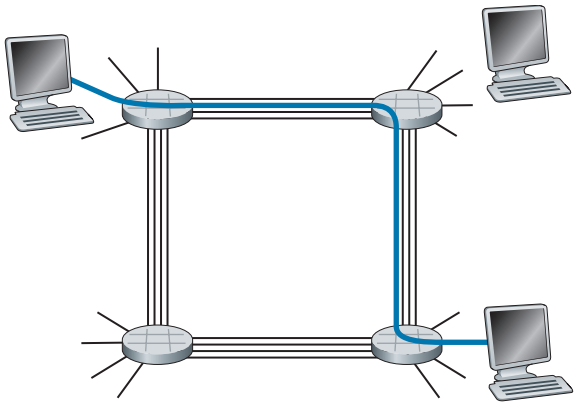


**Figure 1.13** ♦ A simple circuit-switched network consisting of four switches and four links

In contrast, consider what happens when one host wants to send a packet to another host over a packet-switched network, such as the Internet. As with circuit switching, the packet is transmitted over a series of communication links. But different from circuit switching, the packet is sent into the network without reserving any link resources whatsoever. If one of the links is congested because other packets need to be transmitted over the link at the same time, then the packet will have to wait in a buffer at the sending side of the transmission link and suffer a delay. The Internet makes its best effort to deliver packets in a timely manner, but it does not make any guarantees.

### Multiplexing in Circuit-Switched Networks

A circuit in a link is implemented with either **frequency-division multiplexing (FDM)** or **time-division multiplexing (TDM)**. With FDM, the frequency spectrum of a link is divided up among the connections established across the link. Specifically, the link dedicates a frequency band to each connection for the duration of the connection. In telephone networks, this frequency band typically has a width of 4 kHz (that is, 4,000 hertz or 4,000 cycles per second). The width of the band is called, not surprisingly, the **bandwidth**. FM radio stations also use FDM to share the frequency spectrum between 88 MHz and 108 MHz, with each station being allocated a specific frequency band.

For a TDM link, time is divided into frames of fixed duration, and each frame is divided into a fixed number of time slots. When the network establishes a connection across a link, the network dedicates one time slot in every frame to this connection. These slots are dedicated for the sole use of that connection, with one time slot available for use (in every frame) to transmit the connection's data.

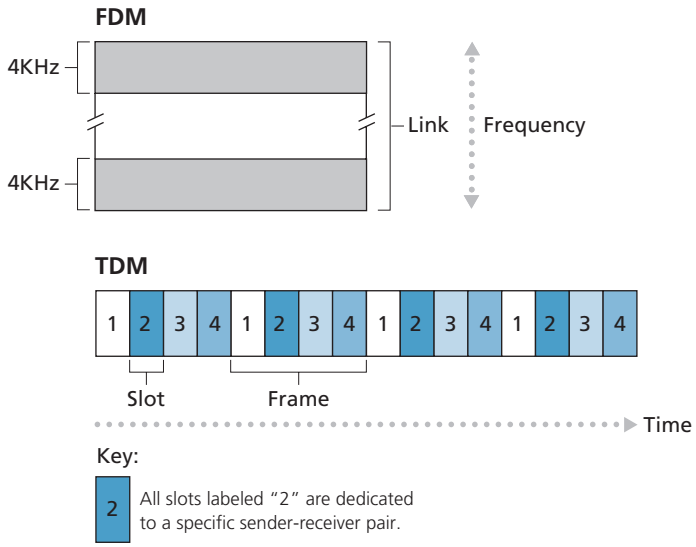



**Figure 1.14** ♦ With FDM, each circuit continuously gets a fraction of the bandwidth. With TDM, each circuit gets all of the bandwidth periodically during brief intervals of time (that is, during slots)

Figure 1.14 illustrates FDM and TDM for a specific network link supporting up to four circuits. For FDM, the frequency domain is segmented into four bands, each of bandwidth 4 kHz. For TDM, the time domain is segmented into frames, with four time slots in each frame; each circuit is assigned the same dedicated slot in the revolving TDM frames. For TDM, the transmission rate of a circuit is equal to the frame rate multiplied by the number of bits in a slot. For example, if the link transmits 8,000 frames per second and each slot consists of 8 bits, then the transmission rate of each circuit is 64 kbps.

Proponents of packet switching have always argued that circuit switching is wasteful because the dedicated circuits are idle during **silent periods**. For example, when one person in a telephone call stops talking, the idle network resources (frequency bands or time slots in the links along the connection's route) cannot be used by other ongoing connections. As another example of how these resources can be underutilized, consider a radiologist who uses a circuit-switched network to remotely access a series of x-rays. The radiologist sets up a connection, requests an image, contemplates the image, and then requests a new image. Network resources are allocated to the connection but are not used (i.e., are wasted) during the radiologist's contemplation periods. Proponents of packet switching also enjoy pointing out that establishing end-to-end circuits and reserving end-to-end transmission capacity is complicated and requires complex signaling software to coordinate the operation of the switches along the end-to-end path.

Before we finish our discussion of circuit switching, let's work through a numerical example that should shed further insight on the topic. Let us consider how long it takes to send a file of 640,000 bits from Host A to Host B over a circuit-switched network. Suppose that all links in the network use TDM with 24 slots and have a bit rate of 1.536 Mbps. Also suppose that it takes 500 msec to establish an end-to-end circuit before Host A can begin to transmit the file. How long does it take to send the file? Each circuit has a transmission rate of (1.536 Mbps)/24 = 64 kbps, so it takes (640,000 bits)/(64 kbps) = 10 seconds to transmit the file. To this 10 seconds we add the circuit establishment time, giving 10.5 seconds to send the file. Note that the transmission time is independent of the number of links: The transmission time would be 10 seconds if the end-to-end circuit passed through one link or a hundred links. (The actual end-to-end delay also includes a propagation delay; see Section 1.4.)

### Packet Switching Versus Circuit Switching

Having described circuit switching and packet switching, let us compare the two. Critics of packet switching have often argued that packet switching is not suitable for real-time services (for example, telephone calls and video conference calls) because of its variable and unpredictable end-to-end delays (due primarily to variable and unpredictable queuing delays). Proponents of packet switching argue that (1) it offers better sharing of transmission capacity than circuit switching and (2) it is simpler, more efficient, and less costly to implement than circuit switching. An interesting discussion of packet switching versus circuit switching is [Molinero-Fernandez 2002]. Generally speaking, people who do not like to hassle with restaurant reservations prefer packet switching to circuit switching.

Why is packet switching more efficient? Let's look at a simple example. Suppose users share a 1 Mbps link. Also suppose that each user alternates between periods of activity, when a user generates data at a constant rate of 100 kbps, and periods of inactivity, when a user generates no data. Suppose further that a user is active only 10 percent of the time (and is idly drinking coffee during the remaining 90 percent of the time). With circuit switching, 100 kbps must be *reserved* for *each* user at all times. For example, with circuit-switched TDM, if a one-second frame is divided into 10 time slots of 100 ms each, then each user would be allocated one time slot per frame.

Thus, the circuit-switched link can support only 10 (= 1 Mbps/100 kbps) simultaneous users. With packet switching, the probability that a specific user is active is 0.1 (that is, 10 percent). If there are 35 users, the probability that there are 11 or more simultaneously active users is approximately 0.0004. (Homework Problem P8 outlines how this probability is obtained.) When there are 10 or fewer simultaneously active users (which happens with probability 0.9996), the aggregate arrival rate of data is less than or equal to 1 Mbps, the output rate of the link. Thus, when there are 10 or fewer active users, users' packets flow through the link essentially

without delay, as is the case with circuit switching. When there are more than 10 simultaneously active users, then the aggregate arrival rate of packets exceeds the output capacity of the link, and the output queue will begin to grow. (It continues to grow until the aggregate input rate falls back below 1 Mbps, at which point the queue will begin to diminish in length.) Because the probability of having more than 10 simultaneously active users is minuscule in this example, packet switching provides essentially the same performance as circuit switching, *but does so while allowing for more than three times the number of users.*

Let's now consider a second simple example. Suppose there are 10 users and that one user suddenly generates one thousand 1,000-bit packets, while other users remain quiescent and do not generate packets. Under TDM circuit switching with 10 slots per frame and each slot consisting of 1,000 bits, the active user can only use its one time slot per frame to transmit data, while the remaining nine time slots in each frame remain idle. It will be 10 seconds before all of the active user's one million bits of data has been transmitted. In the case of packet switching, the active user can continuously send its packets at the full link rate of 1 Mbps, since there are no other users generating packets that need to be multiplexed with the active user's packets. In this case, all of the active user's data will be transmitted within 1 second.

The above examples illustrate two ways in which the performance of packet switching can be superior to that of circuit switching. They also highlight the crucial difference between the two forms of sharing a link's transmission rate among multiple data streams. Circuit switching pre-allocates use of the transmission link regardless of demand, with allocated but unneeded link time going unused. Packet switching on the other hand allocates link use *on demand.* Link transmission capacity will be shared on a packet-by-packet basis only among those users who have packets that need to be transmitted over the link.

Although packet switching and circuit switching are both prevalent in today's telecommunication networks, the trend has certainly been in the direction of packet switching. Even many of today's circuit-switched telephone networks are slowly migrating toward packet switching. In particular, telephone networks often use packet switching for the expensive overseas portion of a telephone call.

### 1.3.3 A Network of Networks

We saw earlier that end systems (PCs, smartphones, Web servers, mail servers, and so on) connect into the Internet via an access ISP. The access ISP can provide either wired or wireless connectivity, using an array of access technologies including DSL, cable, FTTH, Wi-Fi, and cellular. Note that the access ISP does not have to be a telco or a cable company; instead it can be, for example, a university (providing Internet access to students, staff, and faculty), or a company (providing access for its employees). But connecting end users and content providers into an access ISP is only a small piece of solving the puzzle of connecting the billions of end systems that make up the Internet. To complete this puzzle, the access ISPs themselves must be

interconnected. This is done by creating a *network of networks*—understanding this phrase is the key to understanding the Internet.

Over the years, the network of networks that forms the Internet has evolved into a very complex structure. Much of this evolution is driven by economics and national policy, rather than by performance considerations. In order to understand today's Internet network structure, let's incrementally build a series of network structures, with each new structure being a better approximation of the complex Internet that we have today. Recall that the overarching goal is to interconnect the access ISPs so that all end systems can send packets to each other. One naive approach would be to have each access ISP *directly* connect with every other access ISP. Such a mesh design is, of course, much too costly for the access ISPs, as it would require each access ISP to have a separate communication link to each of the hundreds of thousands of other access ISPs all over the world.

Our first network structure, *Network Structure 1*, interconnects all of the access ISPs with a *single global transit ISP*. Our (imaginary) global transit ISP is a network of routers and communication links that not only spans the globe, but also has at least one router near each of the hundreds of thousands of access ISPs. Of course, it would be very costly for the global ISP to build such an extensive network. To be profitable, it would naturally charge each of the access ISPs for connectivity, with the pricing reflecting (but not necessarily directly proportional to) the amount of traffic an access ISP exchanges with the global ISP. Since the access ISP pays the global transit ISP, the access ISP is said to be a **customer** and the global transit ISP is said to be a **provider**.

Now if some company builds and operates a global transit ISP that is profitable, then it is natural for other companies to build their own global transit ISPs and compete with the original global transit ISP. This leads to *Network Structure 2*, which consists of the hundreds of thousands of access ISPs and *multiple* global transit ISPs. The access ISPs certainly prefer Network Structure 2 over Network Structure 1 since they can now choose among the competing global transit providers as a function of their pricing and services. Note, however, that the global transit ISPs themselves must interconnect: Otherwise access ISPs connected to one of the global transit providers would not be able to communicate with access ISPs connected to the other global transit providers.

Network Structure 2, just described, is a two-tier hierarchy with global transit providers residing at the top tier and access ISPs at the bottom tier. This assumes that global transit ISPs are not only capable of getting close to each and every access ISP, but also find it economically desirable to do so. In reality, although some ISPs do have impressive global coverage and do directly connect with many access ISPs, no ISP has presence in each and every city in the world. Instead, in any given region, there may be a **regional ISP** to which the access ISPs in the region connect. Each regional ISP then connects to **tier-1 ISPs**. Tier-1 ISPs are similar to our (imaginary) global transit ISP; but tier-1 ISPs, which actually do exist, do not have a presence in every city in the world. There are approximately a dozen tier-1 ISPs, including Level 3 Communications, AT&T, Sprint, and NTT. Interestingly, no group officially

sanctions tier-1 status; as the saying goes—if you have to ask if you're a member of a group, you're probably not.

Returning to this network of networks, not only are there multiple competing tier-1 ISPs, there may be multiple competing regional ISPs in a region. In such a hierarchy, each access ISP pays the regional ISP to which it connects, and each regional ISP pays the tier-1 ISP to which it connects. (An access ISP can also connect directly to a tier-1 ISP, in which case it pays the tier-1 ISP). Thus, there is customer-provider relationship at each level of the hierarchy. Note that the tier-1 ISPs do not pay anyone as they are at the top of the hierarchy. To further complicate matters, in some regions, there may be a larger regional ISP (possibly spanning an entire country) to which the smaller regional ISPs in that region connect; the larger regional ISP then connects to a tier-1 ISP. For example, in China, there are access ISPs in each city, which connect to provincial ISPs, which in turn connect to national ISPs, which finally connect to tier-1 ISPs [Tian 2012]. We refer to this multi-tier hierarchy, which is still only a crude approximation of today's Internet, as *Network Structure 3*.

To build a network that more closely resembles today's Internet, we must add points of presence (PoPs), multi-homing, peering, and Internet exchange points (IXPs) to the hierarchical Network Structure 3. PoPs exist in all levels of the hierarchy, except for the bottom (access ISP) level. A **PoP** is simply a group of one or more routers (at the same location) in the provider's network where customer ISPs can connect into the provider ISP. For a customer network to connect to a provider's PoP, it can lease a high-speed link from a third-party telecommunications provider to directly connect one of its routers to a router at the PoP. Any ISP (except for tier-1 ISPs) may choose to **multi-home**, that is, to connect to two or more provider ISPs. So, for example, an access ISP may multi-home with two regional ISPs, or it may multi-home with two regional ISPs and also with a tier-1 ISP. Similarly, a regional ISP may multi-home with multiple tier-1 ISPs. When an ISP multi-homes, it can continue to send and receive packets into the Internet even if one of its providers has a failure.

As we just learned, customer ISPs pay their provider ISPs to obtain global Internet interconnectivity. The amount that a customer ISP pays a provider ISP reflects the amount of traffic it exchanges with the provider. To reduce these costs, a pair of nearby ISPs at the same level of the hierarchy can **peer**, that is, they can directly connect their networks together so that all the traffic between them passes over the direct connection rather than through upstream intermediaries. When two ISPs peer, it is typically settlement-free, that is, neither ISP pays the other. As noted earlier, tier-1 ISPs also peer with one another, settlement-free. For a readable discussion of peering and customer-provider relationships, see [Van der Berg 2008]. Along these same lines, a third-party company can create an **Internet Exchange Point (IXP)**, which is a meeting point where multiple ISPs can peer together. An IXP is typically in a stand-alone building with its own switches [Ager 2012]. There are over 600 IXPs in the Internet today [PeeringDB 2020]. We refer to this ecosystem—consisting of access ISPs, regional ISPs, tier-1 ISPs, PoPs, multi-homing, peering, and IXPs—as *Network Structure 4*.

We now finally arrive at *Network Structure 5,* which describes today's Internet. Network Structure 5, illustrated in Figure 1.15, builds on top of Network Structure 4 by adding **content-provider networks**. Google is currently one of the leading examples of such a content-provider network. As of this writing, it Google has 19 major data centers distributed across North America, Europe, Asia, South America, and Australia with each data center having tens or hundreds of thousands of servers. Additionally, Google has smaller data centers, each with a few hundred servers; these smaller data centers are often located within IXPs. The Google data centers are all interconnected via Google's private TCP/IP network, which spans the entire globe but is nevertheless separate from the public Internet. Importantly, the Google private network only carries traffic to/from Google servers. As shown in Figure 1.15, the Google private network attempts to "bypass" the upper tiers of the Internet by peering (settlement free) with lower-tier ISPs, either by directly connecting with them or by connecting with them at IXPs [Labovitz 2010]. However, because many access ISPs can still only be reached by transiting through tier-1 networks, the Google network also connects to tier-1 ISPs, and pays those ISPs for the traffic it exchanges with them. By creating its own network, a content provider not only reduces its payments to upper-tier ISPs, but also has greater control of how its services are ultimately delivered to end users. Google's network infrastructure is described in greater detail in Section 2.6.

In summary, today's Internet—a network of networks—is complex, consisting of a dozen or so tier-1 ISPs and hundreds of thousands of lower-tier ISPs. The ISPs are diverse in their coverage, with some spanning multiple continents and oceans, and others limited to narrow geographic regions. The lower-tier ISPs connect to the higher-tier ISPs, and the higher-tier ISPs interconnect with one another. Users and content providers are customers of lower-tier ISPs, and lower-tier ISPs are customers of higher-tier ISPs. In recent years, major content providers have also created their own networks and connect directly into lower-tier ISPs where possible.

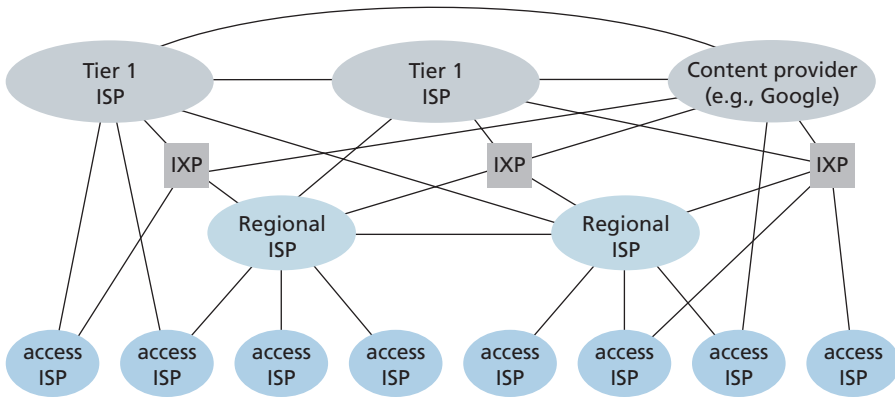



**Figure 1.15** ♦ Interconnection of ISPs

# 1.4 Delay, Loss, and Throughput in Packet-Switched Networks

Back in Section 1.1 we said that the Internet can be viewed as an infrastructure that provides services to distributed applications running on end systems. Ideally, we would like Internet services to be able to move as much data as we want between any two end systems, instantaneously, without any loss of data. Alas, this is a lofty goal, one that is unachievable in reality. Instead, computer networks necessarily constrain throughput (the amount of data per second that can be transferred) between end systems, introduce delays between end systems, and can actually lose packets. On one hand, it is unfortunate that the physical laws of reality introduce delay and loss as well as constrain throughput. On the other hand, because computer networks have these problems, there are many fascinating issues surrounding how to deal with the problems—more than enough issues to fill a course on computer networking and to motivate thousands of PhD theses! In this section, we'll begin to examine and quantify delay, loss, and throughput in computer networks.

## 1.4.1 Overview of Delay in Packet-Switched Networks

Recall that a packet starts in a host (the source), passes through a series of routers, and ends its journey in another host (the destination). As a packet travels from one node (host or router) to the subsequent node (host or router) along this path, the packet suffers from several types of delays at *each* node along the path. The most important of these delays are the **nodal processing delay**, **queuing delay**, **transmission delay**, and **propagation delay**; together, these delays accumulate to give a **total nodal delay**. The performance of many Internet applications—such as search, Web browsing, e-mail, maps, instant messaging, and voice-over-IP—are greatly affected by network delays. In order to acquire a deep understanding of packet switching and computer networks, we must understand the nature and importance of these delays.

### Types of Delay

Let's explore these delays in the context of Figure 1.16. As part of its end-to-end route between source and destination, a packet is sent from the upstream node through router A to router B. Our goal is to characterize the nodal delay at router A. Note that router A has an outbound link leading to router B. This link is preceded by a queue (also known as a buffer). When the packet arrives at router A from the upstream node, router A examines the packet's header to determine the appropriate outbound link for the packet and then directs the packet to this link. In this example, the outbound link for the packet is the one that leads to router B. A packet can be transmitted on a link only if there is no other packet currently being transmitted on the link and if there are no other packets preceding it in the queue; if the link is

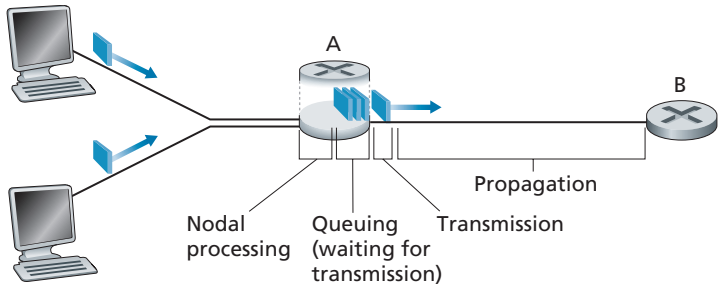



**Figure 1.16** ♦ The nodal delay at router A

currently busy or if there are other packets already queued for the link, the newly arriving packet will then join the queue.

### *Processing Delay*

The time required to examine the packet's header and determine where to direct the packet is part of the **processing delay**. The processing delay can also include other factors, such as the time needed to check for bit-level errors in the packet that occurred in transmitting the packet's bits from the upstream node to router A. Processing delays in high-speed routers are typically on the order of microseconds or less. After this nodal processing, the router directs the packet to the queue that precedes the link to router B. (In Chapter 4 we'll study the details of how a router operates.)

### *Queuing Delay*

At the queue, the packet experiences a **queuing delay** as it waits to be transmitted onto the link. The length of the queuing delay of a specific packet will depend on the number of earlier-arriving packets that are queued and waiting for transmission onto the link. If the queue is empty and no other packet is currently being transmitted, then our packet's queuing delay will be zero. On the other hand, if the traffic is heavy and many other packets are also waiting to be transmitted, the queuing delay will be long. We will see shortly that the number of packets that an arriving packet might expect to find is a function of the intensity and nature of the traffic arriving at the queue. Queuing delays can be on the order of microseconds to milliseconds in practice.

### *Transmission Delay*

Assuming that packets are transmitted in a first-come-first-served manner, as is common in packet-switched networks, our packet can be transmitted only after all the packets that have arrived before it have been transmitted. Denote the length of the

packet by $L$ bits, and denote the transmission rate of the link from router A to router B by $R$ bits/sec. For example, for a 10 Mbps Ethernet link, the rate is $R = 10$ Mbps; for a 100 Mbps Ethernet link, the rate is $R = 100$ Mbps. The **transmission delay** is $L/R$. This is the amount of time required to push (that is, transmit) all of the packet's bits into the link. Transmission delays are typically on the order of microseconds to milliseconds in practice.

### *Propagation Delay*

Once a bit is pushed into the link, it needs to propagate to router B. The time required to propagate from the beginning of the link to router B is the **propagation delay**. The bit propagates at the propagation speed of the link. The propagation speed depends on the physical medium of the link (that is, fiber optics, twisted-pair copper wire, and so on) and is in the range of

$$2 \cdot 10^8 \text{ meters/sec to } 3 \cdot 10^8 \text{ meters/sec}$$

which is equal to, or a little less than, the speed of light. The propagation delay is the distance between two routers divided by the propagation speed. That is, the propagation delay is $d/s$, where $d$ is the distance between router A and router B and $s$ is the propagation speed of the link. Once the last bit of the packet propagates to node B, it and all the preceding bits of the packet are stored in router B. The whole process then continues with router B now performing the forwarding. In wide-area networks, propagation delays are on the order of milliseconds.

### Comparing Transmission and Propagation Delay

VideoNote
Exploring propagation delay and transmission delay

Newcomers to the field of computer networking sometimes have difficulty understanding the difference between transmission delay and propagation delay. The difference is subtle but important. The transmission delay is the amount of time required for the router to push out the packet; it is a function of the packet's length and the transmission rate of the link, but has nothing to do with the distance between the two routers. The propagation delay, on the other hand, is the time it takes a bit to propagate from one router to the next; it is a function of the distance between the two routers, but has nothing to do with the packet's length or the transmission rate of the link.

An analogy might clarify the notions of transmission and propagation delay. Consider a highway that has a tollbooth every 100 kilometers, as shown in Figure 1.17. You can think of the highway segments between tollbooths as links and the tollbooths as routers. Suppose that cars travel (that is, propagate) on the highway at a rate of 100 km/hour (that is, when a car leaves a tollbooth, it instantaneously accelerates to 100 km/hour and maintains that speed between tollbooths). Suppose next that 10 cars, traveling together as a caravan, follow each other in a fixed order. You can think of each car as a bit and the caravan as a packet. Also suppose that each

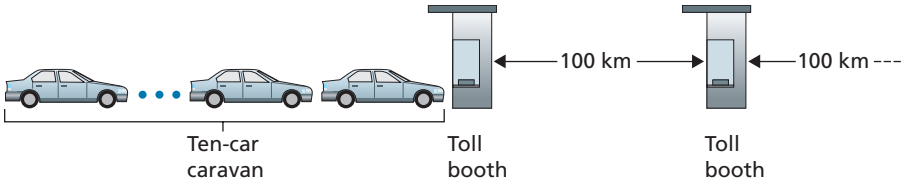



**Figure 1.17** ♦ Caravan analogy

tollbooth services (that is, transmits) a car at a rate of one car per 12 seconds, and that it is late at night so that the caravan's cars are the only cars on the highway. Finally, suppose that whenever the first car of the caravan arrives at a tollbooth, it waits at the entrance until the other nine cars have arrived and lined up behind it. (Thus, the entire caravan must be stored at the tollbooth before it can begin to be forwarded.) The time required for the tollbooth to push the entire caravan onto the highway is (10 cars)/(5 cars/minute) = 2 minutes. This time is analogous to the transmission delay in a router. The time required for a car to travel from the exit of one tollbooth to the next tollbooth is 100 km/(100 km/hour) = 1 hour. This time is analogous to propagation delay. Therefore, the time from when the caravan is stored in front of a tollbooth until the caravan is stored in front of the next tollbooth is the sum of transmission delay and propagation delay—in this example, 62 minutes.

Let's explore this analogy a bit more. What would happen if the tollbooth service time for a caravan were greater than the time for a car to travel between tollbooths? For example, suppose now that the cars travel at the rate of 1,000 km/hour and the tollbooth services cars at the rate of one car per minute. Then the traveling delay between two tollbooths is 6 minutes and the time to serve a caravan is 10 minutes. In this case, the first few cars in the caravan will arrive at the second tollbooth before the last cars in the caravan leave the first tollbooth. This situation also arises in packet-switched networks—the first bits in a packet can arrive at a router while many of the remaining bits in the packet are still waiting to be transmitted by the preceding router.

If a picture speaks a thousand words, then an animation must speak a million words. The Web site for this textbook provides an interactive animation that nicely illustrates and contrasts transmission delay and propagation delay. The reader is highly encouraged to visit that animation. [Smith 2009] also provides a very readable discussion of propagation, queuing, and transmission delays.

If we let $d_{\text{proc}}$, $d_{\text{queue}}$, $d_{\text{trans}}$, and $d_{\text{prop}}$ denote the processing, queuing, transmission, and propagation delays, then the total nodal delay is given by

$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

The contribution of these delay components can vary significantly. For example, $d_{\text{prop}}$ can be negligible (for example, a couple of microseconds) for a link connecting two routers on the same university campus; however, $d_{\text{prop}}$ is hundreds of milliseconds for two routers interconnected by a geostationary satellite link, and can be the

dominant term in $d_{nodal}$. Similarly, $d_{trans}$ can range from negligible to significant. Its contribution is typically negligible for transmission rates of 10 Mbps and higher (for example, for LANs); however, it can be hundreds of milliseconds for large Internet packets sent over low-speed dial-up modem links. The processing delay, $d_{proc}$, is often negligible; however, it strongly influences a router's maximum throughput, which is the maximum rate at which a router can forward packets.

## 1.4.2 Queuing Delay and Packet Loss

The most complicated and interesting component of nodal delay is the queuing delay, $d_{queue}$. In fact, queuing delay is so important and interesting in computer networking that thousands of papers and numerous books have been written about it [Bertsekas 1991; Kleinrock 1975, Kleinrock 1976]. We give only a high-level, intuitive discussion of queuing delay here; the more curious reader may want to browse through some of the books (or even eventually write a PhD thesis on the subject!). Unlike the other three delays (namely, $d_{proc}$, $d_{trans}$, and $d_{prop}$), the queuing delay can vary from packet to packet. For example, if 10 packets arrive at an empty queue at the same time, the first packet transmitted will suffer no queuing delay, while the last packet transmitted will suffer a relatively large queuing delay (while it waits for the other nine packets to be transmitted). Therefore, when characterizing queuing delay, one typically uses statistical measures, such as average queuing delay, variance of queuing delay, and the probability that the queuing delay exceeds some specified value.

When is the queuing delay large and when is it insignificant? The answer to this question depends on the rate at which traffic arrives at the queue, the transmission rate of the link, and the nature of the arriving traffic, that is, whether the traffic arrives periodically or arrives in bursts. To gain some insight here, let $a$ denote the average rate at which packets arrive at the queue ($a$ is in units of packets/sec). Recall that $R$ is the transmission rate; that is, it is the rate (in bits/sec) at which bits are pushed out of the queue. Also suppose, for simplicity, that all packets consist of $L$ bits. Then the average rate at which bits arrive at the queue is $La$ bits/sec. Finally, assume that the queue is very big, so that it can hold essentially an infinite number of bits. The ratio $La/R$, called the **traffic intensity**, often plays an important role in estimating the extent of the queuing delay. If $La/R > 1$, then the average rate at which bits arrive at the queue exceeds the rate at which the bits can be transmitted from the queue. In this unfortunate situation, the queue will tend to increase without bound and the queuing delay will approach infinity! Therefore, one of the golden rules in traffic engineering is: *Design your system so that the traffic intensity is no greater than 1.*

Now consider the case $La/R \leq 1$. Here, the nature of the arriving traffic impacts the queuing delay. For example, if packets arrive periodically—that is, one packet arrives every $L/R$ seconds—then every packet will arrive at an empty queue and there will be no queuing delay. On the other hand, if packets arrive in bursts but periodically, there can be a significant average queuing delay. For example, suppose $N$ packets arrive simultaneously every $(L/R)N$ seconds. Then the first packet transmitted has no queuing delay; the second packet transmitted has a queuing delay

of $L/R$ seconds; and more generally, the $n$th packet transmitted has a queuing delay of $(n - 1)L/R$ seconds. We leave it as an exercise for you to calculate the average queuing delay in this example.

The two examples of periodic arrivals described above are a bit academic. Typically, the arrival process to a queue is *random;* that is, the arrivals do not follow any pattern and the packets are spaced apart by random amounts of time. In this more realistic case, the quantity $La/R$ is not usually sufficient to fully characterize the queuing delay statistics. Nonetheless, it is useful in gaining an intuitive understanding of the extent of the queuing delay. In particular, if the traffic intensity is close to zero, then packet arrivals are few and far between and it is unlikely that an arriving packet will find another packet in the queue. Hence, the average queuing delay will be close to zero. On the other hand, when the traffic intensity is close to 1, there will be intervals of time when the arrival rate exceeds the transmission capacity (due to variations in packet arrival rate), and a queue will form during these periods of time; when the arrival rate is less than the transmission capacity, the length of the queue will shrink. Nonetheless, as the traffic intensity approaches 1, the average queue length gets larger and larger. The qualitative dependence of average queuing delay on the traffic intensity is shown in Figure 1.18.

One important aspect of Figure 1.18 is the fact that as the traffic intensity approaches 1, the average queuing delay increases rapidly. A small percentage increase in the intensity will result in a much larger percentage-wise increase in delay. Perhaps you have experienced this phenomenon on the highway. If you regularly drive on a road that is typically congested, the fact that the road is typically congested means that its traffic intensity is close to 1. If some event causes an even slightly larger-than-usual amount of traffic, the delays you experience can be huge.

To really get a good feel for what queuing delays are about, you are encouraged once again to visit the textbook Web site, which provides an interactive animation for a queue. If you set the packet arrival rate high enough so that the traffic intensity exceeds 1, you will see the queue slowly build up over time.

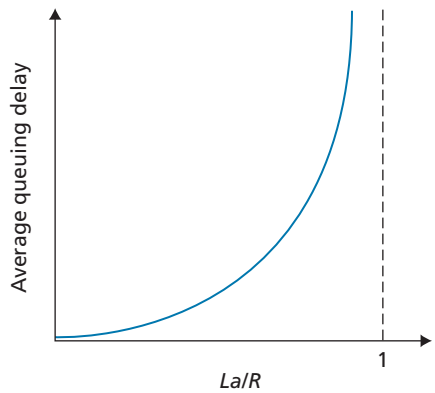



**Figure 1.18** ♦ Dependence of average queuing delay on traffic intensity

### Packet Loss

In our discussions above, we have assumed that the queue is capable of holding an infinite number of packets. In reality a queue preceding a link has finite capacity, although the queuing capacity greatly depends on the router design and cost. Because the queue capacity is finite, packet delays do not really approach infinity as the traffic intensity approaches 1. Instead, a packet can arrive to find a full queue. With no place to store such a packet, a router will **drop** that packet; that is, the packet will be **lost**. This overflow at a queue can again be seen in the interactive animation when the traffic intensity is greater than 1.

From an end-system viewpoint, a packet loss will look like a packet having been transmitted into the network core but never emerging from the network at the destination. The fraction of lost packets increases as the traffic intensity increases. Therefore, performance at a node is often measured not only in terms of delay, but also in terms of the probability of packet loss. As we'll discuss in the subsequent chapters, a lost packet may be retransmitted on an end-to-end basis in order to ensure that all data are eventually transferred from source to destination.

## 1.4.3 End-to-End Delay

Our discussion up to this point has focused on the nodal delay, that is, the delay at a single router. Let's now consider the total delay from source to destination. To get a handle on this concept, suppose there are $N - 1$ routers between the source host and the destination host. Let's also suppose for the moment that the network is uncongested (so that queuing delays are negligible), the processing delay at each router and at the source host is $d_{\text{proc}}$, the transmission rate out of each router and out of the source host is $R$ bits/sec, and the propagation on each link is $d_{\text{prop}}$. The nodal delays accumulate and give an end-to-end delay,

$$d_{\text{end-end}} = N\,(d_{\text{proc}} + d_{\text{trans}} + d_{\text{prop}}) \tag{1.2}$$

where, once again, $d_{\text{trans}} = L/R$, where $L$ is the packet size. Note that Equation 1.2 is a generalization of Equation 1.1, which did not take into account processing and propagation delays. We leave it to you to generalize Equation 1.2 to the case of heterogeneous delays at the nodes and to the presence of an average queuing delay at each node.

### Traceroute

**VideoNote**
**Using Traceroute to discover network paths and measure network delay**

To get a hands-on feel for end-to-end delay in a computer network, we can make use of the Traceroute program. Traceroute is a simple program that can run in any Internet host. When the user specifies a destination hostname, the program in the source host sends multiple, special packets toward that destination. As these packets work their way toward the destination, they pass through a series of routers. When a router receives one of these special packets, it sends back to the source a short message that contains the name and address of the router.

More specifically, suppose there are $N - 1$ routers between the source and the destination. Then the source will send $N$ special packets into the network, with each packet addressed to the ultimate destination. These $N$ special packets are marked *1* through *N,* with the first packet marked *1* and the last packet marked *N*. When the *n*th router receives the *n*th packet marked *n,* the router does not forward the packet toward its destination, but instead sends a message back to the source. When the destination host receives the *N*th packet, it too returns a message back to the source. The source records the time that elapses between when it sends a packet and when it receives the corresponding return message; it also records the name and address of the router (or the destination host) that returns the message. In this manner, the source can reconstruct the route taken by packets flowing from source to destination, and the source can determine the round-trip delays to all the intervening routers. Traceroute actually repeats the experiment just described three times, so the source actually sends $3 \cdot N$ packets to the destination. RFC 1393 describes Traceroute in detail.

Here is an example of the output of the Traceroute program, where the route was being traced from the source host gaia.cs.umass.edu (at the University of Massachusetts) to a host in the computer science department at the University of Sorbonne in Paris (formerly the university was known as UPMC). The output has six columns: the first column is the *n* value described above, that is, the number of the router along the route; the second column is the name of the router; the third column is the address of the router (of the form xxx.xxx.xxx.xxx); the last three columns are the round-trip delays for three experiments. If the source receives fewer than three messages from any given router (due to packet loss in the network), Traceroute places an asterisk just after the router number and reports fewer than three round-trip times for that router.

```
1  gw-vlan-2451.cs.umass.edu (128.119.245.1)  1.899 ms 3.266 ms  3.280 ms
2  j-cs-gw-int-10-240.cs.umass.edu (10.119.240.254) 1.296 ms 1.276 ms
   1.245 ms
3  n5-rt-1-1-xe-2-1-0.gw.umass.edu (128.119.3.33) 2.237 ms  2.217 ms
   2.187 ms
4  core1-rt-et-5-2-0.gw.umass.edu (128.119.0.9) 0.351 ms 0.392 ms 0.380 ms
5  border1-rt-et-5-0-0.gw.umass.edu (192.80.83.102) 0.345 ms 0.345 ms
   0.344 ms
6  nox300gw1-umass-re.nox.org (192.5.89.101) 3.260 ms  0.416 ms 3.127 ms
7  nox300gw1-umass-re.nox.org (192.5.89.101) 3.165 ms 7.326 ms  7.311 ms
8  198.71.45.237 (198.71.45.237) 77.826 ms 77.246 ms 77.744 ms
9  renater-lb1-gw.mx1.par.fr.geant.net (62.40.124.70) 79.357 ms 77.729
   79.152 ms
10 193.51.180.109 (193.51.180.109) 78.379 ms  79.936 80.042 ms
11 * 193.51.180.109 (193.51.180.109) 80.640 ms *
12 * 195.221.127.182 (195.221.127.182) 78.408 ms *
13 195.221.127.182 (195.221.127.182) 80.686 ms 80.796 ms 78.434 ms
14 r-upmc1.reseau.jussieu.fr (134.157.254.10) 78.399 ms * 81.353 ms
```

In the trace above, there are 14 routers between the source and the destination. Most of these routers have a name, and all of them have addresses. For example, the name of Router 4 is `core1-rt-et-5-2-0.gw.umass.edu` and its address is `128.119.0.9`. Looking at the data provided for this same router, we see that in the first of the three trials the round-trip delay between the source and the router was 0.351 msec. The round-trip delays for the subsequent two trials were 0.392 and 0.380 msec. These round-trip delays include all of the delays just discussed, including transmission delays, propagation delays, router processing delays, and queuing delay.

Because the queuing delay is varying with time, the round-trip delay of packet $n$ sent to a router $n$ can sometimes be longer than the round-trip delay of packet $n$+1 sent to router $n$+1. Indeed, we observe this phenomenon in the above example: the delay to Router 12 is smaller than the delay to Router 11! Also note the big increase in the round-trip delay when going from router 7 to router 8. This is due to a transatlantic fiber-optic link between routers 7 and 8, giving rise to a relatively large propagation delay. There are a number of free software programs that provide a graphical interface to Traceroute; one of our favorites is PingPlotter [PingPlotter 2020].

### End System, Application, and Other Delays

In addition to processing, transmission, and propagation delays, there can be additional significant delays in the end systems. For example, an end system wanting to transmit a packet into a shared medium (e.g., as in a WiFi or cable modem scenario) may *purposefully* delay its transmission as part of its protocol for sharing the medium with other end systems; we'll consider such protocols in detail in Chapter 6. Another important delay is media packetization delay, which is present in Voice-over-IP (VoIP) applications. In VoIP, the sending side must first fill a packet with encoded digitized speech before passing the packet to the Internet. This time to fill a packet—called the packetization delay—can be significant and can impact the user-perceived quality of a VoIP call. This issue will be further explored in a homework problem at the end of this chapter.

## 1.4.4 Throughput in Computer Networks

In addition to delay and packet loss, another critical performance measure in computer networks is end-to-end throughput. To define throughput, consider transferring a large file from Host A to Host B across a computer network. This transfer might be, for example, a large video clip from one computer to another. The **instantaneous throughput** at any instant of time is the rate (in bits/sec) at which Host B is receiving the file. (Many applications display the instantaneous throughput during downloads in the user interface—perhaps you have observed this before! You might like to try

measuring the end-to-end delay and download throughput between your and servers around the Internet using the speedtest application [Speedtest 2020].) If the file consists of $F$ bits and the transfer takes $T$ seconds for Host B to receive all $F$ bits, then the **average throughput** of the file transfer is $F/T$ bits/sec. For some applications, such as Internet telephony, it is desirable to have a low delay and an instantaneous throughput consistently above some threshold (for example, over 24 kbps for some Internet telephony applications and over 256 kbps for some real-time video applications). For other applications, including those involving file transfers, delay is not critical, but it is desirable to have the highest possible throughput.

To gain further insight into the important concept of throughput, let's consider a few examples. Figure 1.19(a) shows two end systems, a server and a client, connected by two communication links and a router. Consider the throughput for a file transfer from the server to the client. Let $R_s$ denote the rate of the link between the server and the router; and $R_c$ denote the rate of the link between the router and the client. Suppose that the only bits being sent in the entire network are those from the server to the client. We now ask, in this ideal scenario, what is the server-to-client throughput? To answer this question, we may think of bits as *fluid* and communication links as *pipes*. Clearly, the server cannot pump bits through its link at a rate faster than $R_s$ bps; and the router cannot forward bits at a rate faster than $R_c$ bps. If $R_s < R_c$, then the bits pumped by the server will "flow" right through the router and arrive at the client at a rate of $R_s$ bps, giving a throughput of $R_s$ bps. If, on the other hand, $R_c < R_s$, then the router will not be able to forward bits as quickly as it receives them. In this case, bits will only leave the router at rate $R_c$, giving an end-to-end throughput of $R_c$. (Note also that if bits continue to arrive at the router at rate $R_s$, and continue to leave the router at $R_c$, the backlog of bits at the router waiting for transmission to the client will grow and grow—a most undesirable situation!)

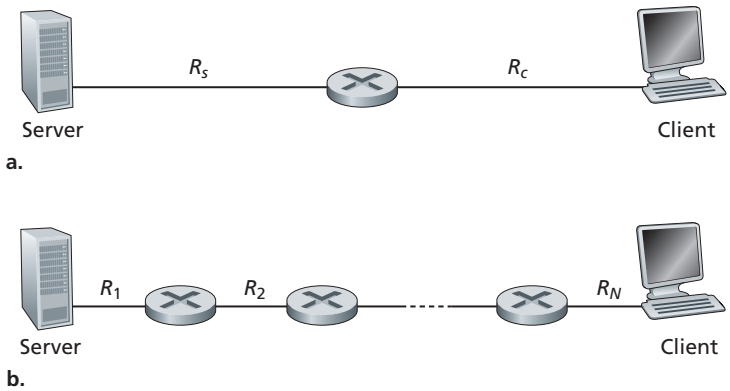



**Figure 1.19** ♦ Throughput for a file transfer from server to client

Thus, for this simple two-link network, the throughput is $\min\{R_c, R_s\}$, that is, it is the transmission rate of the **bottleneck link**. Having determined the throughput, we can now approximate the time it takes to transfer a large file of $F$ bits from server to client as $F/\min\{R_s, R_c\}$. For a specific example, suppose that you are downloading an MP3 file of $F = 32$ million bits, the server has a transmission rate of $R_s = 2$ Mbps, and you have an access link of $R_c = 1$ Mbps. The time needed to transfer the file is then 32 seconds. Of course, these expressions for throughput and transfer time are only approximations, as they do not account for store-and-forward and processing delays as well as protocol issues.

Figure 1.19(b) now shows a network with $N$ links between the server and the client, with the transmission rates of the $N$ links being $R_1, R_2, \ldots, R_N$. Applying the same analysis as for the two-link network, we find that the throughput for a file transfer from server to client is $\min\{R_1, R_2, \ldots, R_N\}$, which is once again the transmission rate of the bottleneck link along the path between server and client.

Now consider another example motivated by today's Internet. Figure 1.20(a) shows two end systems, a server and a client, connected to a computer network. Consider the throughput for a file transfer from the server to the client. The server is connected to the network with an access link of rate $R_s$ and the client is connected to the network with an access link of rate $R_c$. Now suppose that all the links in the core of the communication network have very high transmission rates, much higher than $R_s$ and $R_c$. Indeed, today, the core of the Internet is over-provisioned with high speed links that experience little congestion. Also suppose that the only bits being sent in the entire network are those from the server to the client. Because the core of the computer network is like a wide pipe in this example, the rate at which bits can flow from source to destination is again the minimum of $R_s$ and $R_c$, that is, throughput $= \min\{R_s, R_c\}$. Therefore, the constraining factor for throughput in today's Internet is typically the access network.

For a final example, consider Figure 1.20(b) in which there are 10 servers and 10 clients connected to the core of the computer network. In this example, there are 10 simultaneous downloads taking place, involving 10 client-server pairs. Suppose that these 10 downloads are the only traffic in the network at the current time. As shown in the figure, there is a link in the core that is traversed by all 10 downloads. Denote $R$ for the transmission rate of this link $R$. Let's suppose that all server access links have the same rate $R_s$, all client access links have the same rate $R_c$, and the transmission rates of all the links in the core—except the one common link of rate $R$—are much larger than $R_s$, $R_c$, and $R$. Now we ask, what are the throughputs of the downloads? Clearly, if the rate of the common link, $R$, is large—say a hundred times larger than both $R_s$ and $R_c$—then the throughput for each download will once again be $\min\{R_s, R_c\}$. But what if the rate of the common link is of the same order as $R_s$ and $R_c$? What will the throughput be in this case? Let's take a look at a specific example. Suppose $R_s = 2$ Mbps, $R_c = 1$ Mbps, $R = 5$ Mbps, and the common link divides its transmission rate equally among the 10 downloads. Then the

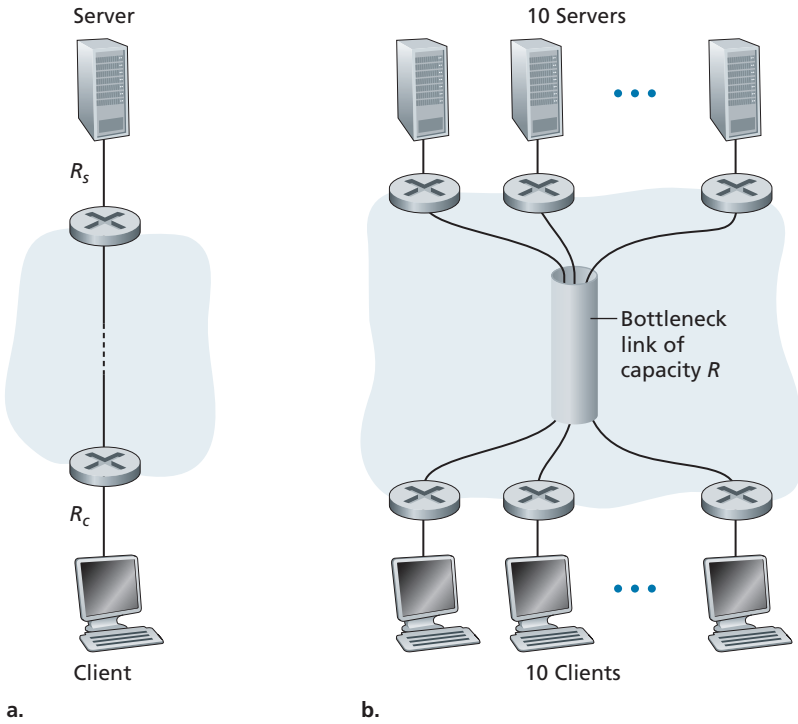



**Figure 1.20** ♦ End-to-end throughput: (a) Client downloads a file from server; (b) 10 clients downloading with 10 servers

bottleneck for each download is no longer in the access network, but is now instead the shared link in the core, which only provides each download with 500 kbps of throughput. Thus, the end-to-end throughput for each download is now reduced to 500 kbps.

The examples in Figure 1.19 and Figure 1.20(a) show that throughput depends on the transmission rates of the links over which the data flows. We saw that when there is no other intervening traffic, the throughput can simply be approximated as the minimum transmission rate along the path between source and destination. The example in Figure 1.20(b) shows that more generally the throughput depends not only on the transmission rates of the links along the path, but also on the intervening traffic. In particular, a link with a high transmission rate may nonetheless be the bottleneck link for a file transfer if many other data flows are also passing through that link. We will examine throughput in computer networks more closely in the homework problems and in the subsequent chapters.

# 1.5 Protocol Layers and Their Service Models

From our discussion thus far, it is apparent that the Internet is an *extremely* complicated system. We have seen that there are many pieces to the Internet: numerous applications and protocols, various types of end systems, packet switches, and various types of link-level media. Given this enormous complexity, is there any hope of organizing a network architecture, or at least our discussion of network architecture? Fortunately, the answer to both questions is yes.

## 1.5.1 Layered Architecture

Before attempting to organize our thoughts on Internet architecture, let's look for a human analogy. Actually, we deal with complex systems all the time in our everyday life. Imagine if someone asked you to describe, for example, the airline system. How would you find the structure to describe this complex system that has ticketing agents, baggage checkers, gate personnel, pilots, airplanes, air traffic control, and a worldwide system for routing airplanes? One way to describe this system might be to describe the series of actions you take (or others take for you) when you fly on an airline. You purchase your ticket, check your bags, go to the gate, and eventually get loaded onto the plane. The plane takes off and is routed to its destination. After your plane lands, you deplane at the gate and claim your bags. If the trip was bad, you complain about the flight to the ticket agent (getting nothing for your effort). This scenario is shown in Figure 1.21.

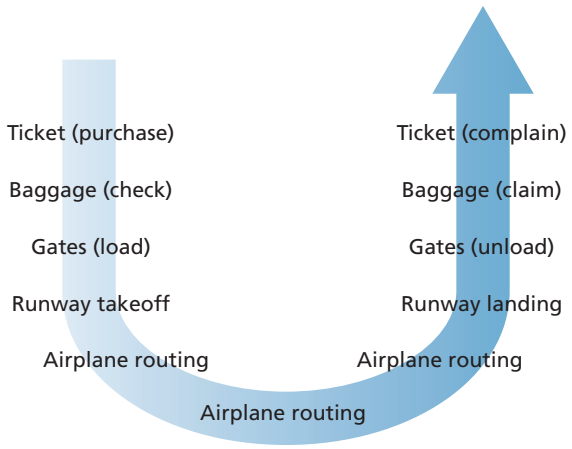



**Figure 1.21** ♦ Taking an airplane trip: actions

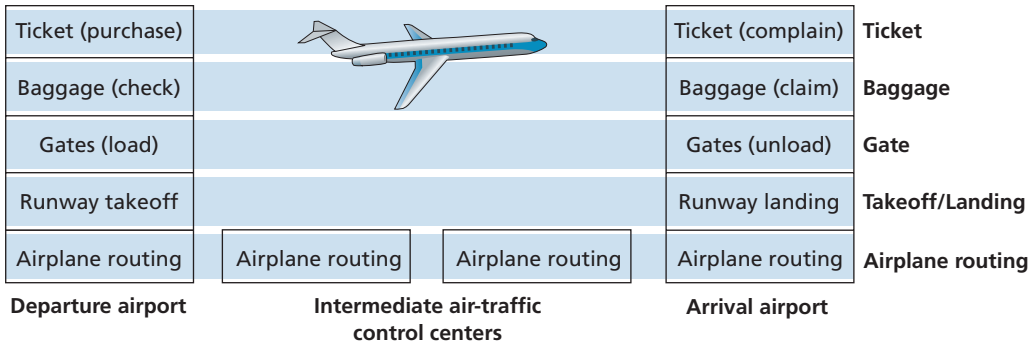



**Figure 1.22** ♦ Horizontal layering of airline functionality

Already, we can see some analogies here with computer networking: You are being shipped from source to destination by the airline; a packet is shipped from source host to destination host in the Internet. But this is not quite the analogy we are after. We are looking for some *structure* in Figure 1.21. Looking at Figure 1.21, we note that there is a ticketing function at each end; there is also a baggage function for already-ticketed passengers, and a gate function for already-ticketed and already-baggage-checked passengers. For passengers who have made it through the gate (that is, passengers who are already ticketed, baggage-checked, and through the gate), there is a takeoff and landing function, and while in flight, there is an airplane-routing function. This suggests that we can look at the functionality in Figure 1.21 in a *horizontal* manner, as shown in Figure 1.22.

Figure 1.22 has divided the airline functionality into layers, providing a framework in which we can discuss airline travel. Note that each layer, combined with the layers below it, implements some functionality, some *service.* At the ticketing layer and below, airline-counter-to-airline-counter transfer of a person is accomplished. At the baggage layer and below, baggage-check-to-baggage-claim transfer of a person and bags is accomplished. Note that the baggage layer provides this service only to an already-ticketed person. At the gate layer, departure-gate-to-arrival-gate transfer of a person and bags is accomplished. At the takeoff/landing layer, runway-to-runway transfer of people and their bags is accomplished. Each layer provides its service by (1) performing certain actions within that layer (for example, at the gate layer, loading and unloading people from an airplane) and by (2) using the services of the layer directly below it (for example, in the gate layer, using the runway-to-runway passenger transfer service of the takeoff/landing layer).

A layered architecture allows us to discuss a well-defined, specific part of a large and complex system. This simplification itself is of considerable value by providing modularity, making it much easier to change the implementation of the service provided by the layer. As long as the layer provides the same service to the layer above it, and uses the same services from the layer below it, the remainder of the system remains unchanged when a layer's implementation is changed. (Note