

Computer Networking

A Top-Down Approach



sixth edition

KUROSE | ROSS

FreeWheel, Ex. 1096
FreeWheel v. AlmondNet
IPR2024-00419

COMPUTER NETWORKING

SIXTH EDITION

A Top-Down Approach



JAMES F. KUROSE

University of Massachusetts, Amherst

KEITH W. ROSS

Polytechnic Institute of NYU

PEARSON

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

Vice President and Editorial Director, ECS:

Marcia Horton

Editor in Chief: Michael Hirsch

Editorial Assistant: Emma Snider

Vice President Marketing: Patrice Jones

Marketing Manager: Yez Alayan

Marketing Coordinator: Kathryn Ferranti

Vice President and Director of Production:

Vince O'Brien

Managing Editor: Jeff Holcomb

Senior Production Project Manager:

Marilyn Lloyd

Manufacturing Manager: Nick Sklitsis

Operations Specialist: Lisa McDowell

Art Director, Cover: Anthony Gemmellaro

Art Coordinator: Janet Theurer/

Theurer Briggs Design

Art Studio: Patrice Rossi Calkin/

Rossi Illustration and Design

Cover Designer: Liz Harasymcuk

Text Designer: Joyce Cosentino Wells

Cover Image: ©Fancy/Alamy

Media Editor: Dan Sandin

Full-Service Vendor: PreMediaGlobal

Senior Project Manager: Andrea Stefanowicz

Printer/Binder: Edwards Brothers Malloy

Cover Printer: Lehigh-Phoenix Color

This book was composed in Quark. Basal font is Times. Display font is Berkeley.

Copyright © 2013, 2010, 2008, 2005, 2003 by Pearson Education, Inc., publishing as Addison-Wesley. All rights reserved. Manufactured in the United States of America. This publication is protected by Copyright, and permission should be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission(s) to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to 201-236-3290.

Many of the designations by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed in initial caps or all caps.

Library of Congress Cataloging-in-Publication Data

Kurose, James F.

Computer networking : a top-down approach / James F. Kurose, Keith W. Ross.—6th ed.
p. cm.

Includes bibliographical references and index.

ISBN-13: 978-0-13-285620-1

ISBN-10: 0-13-285620-4

1. Internet. 2. Computer networks. I. Ross, Keith W., 1956- II. Title.

TK5105.875.I57K88 2012

004.6—dc23

2011048215

10 9 8 7 6 5 4 3 2

PEARSON

ISBN-10: 0-13-285620-4

ISBN-13: 978-0-13-285620-1

In fact, the intermediate routers are completely oblivious to TCP connections; they see datagrams, not connections.

A TCP connection provides a **full-duplex service**; If there is a TCP connection between Process A on one host and Process B on another host, then application-layer data can flow from Process A to Process B at the same time as application-layer data flows from Process B to Process A. A TCP connection is also always **point-to-point**, that is, between a single sender and a single receiver. So-called “multicasting” (see Section 4.7)—the transfer of data from one sender to many receivers in a single send operation—is not possible with TCP. With TCP, two hosts are company and three are a crowd!

Let’s now take a look at how a TCP connection is established. Suppose a process running in one host wants to initiate a connection with another process in another host. Recall that the process that is initiating the connection is called the *client process*, while the other process is called the *server process*. The client application process first informs the client transport layer that it wants to establish a connection to a process in the server. Recall from Section 2.7.2, a Python client program does this by issuing the command

```
clientSocket.connect((serverName,serverPort))
```

where `serverName` is the name of the server and `serverPort` identifies the process on the server. TCP in the client then proceeds to establish a TCP connection with TCP in the server. At the end of this section we discuss in some detail the connection-establishment procedure. For now it suffices to know that the client first sends a special TCP segment; the server responds with a second special TCP segment; and finally the client responds again with a third special segment. The first two segments carry no payload, that is, no application-layer data; the third of these segments may carry a payload. Because three segments are sent between the two hosts, this connection-establishment procedure is often referred to as a **three-way handshake**.

Once a TCP connection is established, the two application processes can send data to each other. Let’s consider the sending of data from the client process to the server process. The client process passes a stream of data through the socket (the door of the process), as described in Section 2.7. Once the data passes through the door, the data is in the hands of TCP running in the client. As shown in Figure 3.28, TCP directs this data to the connection’s **send buffer**, which is one of the buffers that is set aside during the initial three-way handshake. From time to time, TCP will grab chunks of data from the send buffer and pass the data to the network layer. Interestingly, the TCP specification [RFC 793] is very laid back about specifying when TCP should actually send buffered data, stating that TCP should “send that data in segments at its own convenience.” The maximum amount of data that can be grabbed and placed in a segment is limited by the **maximum segment size (MSS)**. The MSS is typically set by first determining the length of the largest link-layer frame that can be sent by the local sending host (the so-called **maximum**