EDITION 4

COMPUTER NETWORKS

A SYSTEMS APPROACH

LARRY L. PETERSON & BRUCE S. DAVIE

ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

MK

Acquisitions Editor	Rick Adams
Publishing Services Manager	George Morrison
Production Editor	Dawnmarie Simpson
Associate Acquisitions Editor	Rachel Roumeliotis
Design Direction	Louis Forgione
Composition	VTEX
Copyeditor	Multiscience Press, Inc.
Proofreader	Jodie Allen
Indexer	Multiscience Press, Inc.
Interior printer	Courier Westford
Cover printer	Phoenix Color, Inc.

Morgan Kaufmann Publishers is an imprint of Elsevier.
500 Sansome Street, Suite 400, San Francisco, CA 94111

This book is printed on acid-free paper.

ISBN 13: 978-0-12-370548-8 (Case bound)
ISBN 10: 0-12-370548-7 (Case bound)

ISBN 13: 978-0-12-374013-7 (Paperback)
ISBN 10: 0-12-374013-4 (Paperback)

Library of Congress Cataloging-in-Publication Data
Peterson, Larry L.
Computer networks : a systems approach / Larry L. Peterson & Bruce S. Davi – 4th ed.
p. cm.
Includes bibliographical references and index.
ISBN-13: 978-0-12-370548-8 (hardcover : alk. paper)
ISBN-10: 0-12-370548-7 (hardcover : alk. paper)
ISBN-13: 978-0-12-374013-7 (pbk. : alk. paper)
ISBN-10: 0-12-374013-4 (pbk. : alk. paper) 1. Computer networks. I. Davie, Bruce S. II. Title.
TK5105.5.P479 2007
004.6'5–dc22

2006102454

For information on all Morgan Kaufmann publications, visit our Web site at *www.mkp.com* or *www.books.elsevier.com*

Printed in the United States of America.
06 07 08 09 10 5 4 3 2 1

1.1 Applications

Most people know the Internet through its applications: the World Wide Web, email, streaming audio and video, chat rooms, and music (file) sharing. The Web, for example, presents an intuitively simple interface. Users view pages full of textual and graphical objects, click on objects that they want to learn more about, and a corresponding new page appears. Most people are also aware that just under the covers, each selectable object on a page is bound to an identifier for the next page to be viewed. This identifier, called a Uniform Resource Locator (URL), is used to provide a way of identifying all the possible pages that can be viewed from your web browser. For example,

http://www.cs.princeton.edu/~llp/index.html

is the URL for a page providing information about one of this book's authors: the string http indicates that the HyperText Transfer Protocol (HTTP) should be used to download the page, www.cs.princeton.edu is the name of the machine that serves the page, and

/~llp/index.html

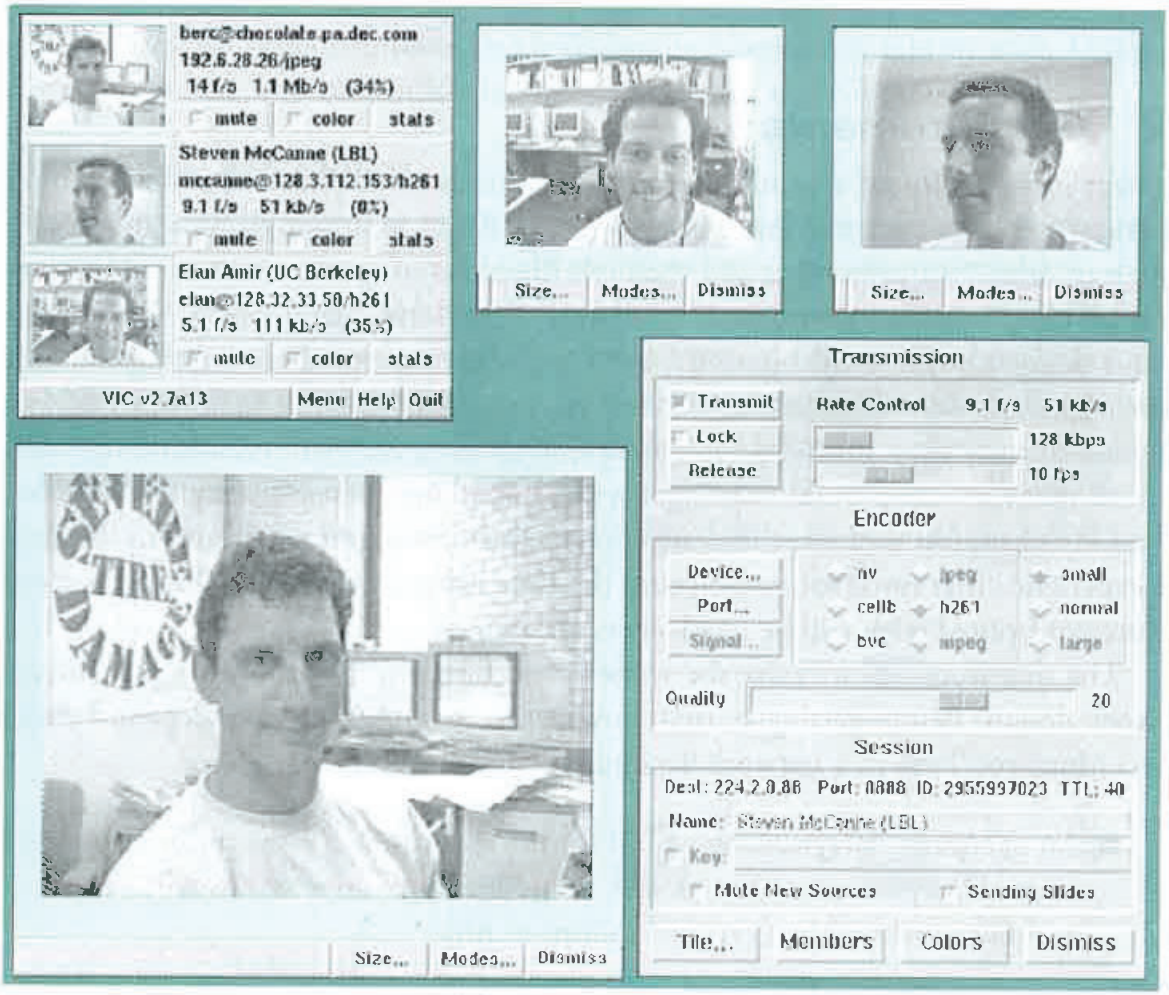
uniquely identifies Larry's home page at this site.

What most Web users are not aware of, however, is that by clicking on just one such URL, as many as 17 messages may be exchanged over the Internet, and this assumes the page itself is small enough to fit in a single message. This number includes up to six messages to translate the server name (www.cs.princeton.edu) into its Internet address (128.112.136.35), three messages to set up a Transmission Control Protocol (TCP) connection between your browser and this server, four messages for your browser to send the HTTP "get" request and the server to respond with the requested page (and for each side to acknowledge receipt of that message), and four messages to tear down the TCP connection. Of course, this does not include the millions of messages exchanged by Internet nodes throughout the day, just to let each other know that they exist and are ready to serve web pages, translate names to addresses, and forward messages toward their ultimate destination.

Another widespread application of the Internet is the delivery of "streaming" audio and video. While an entire video file could first be fetched from a remote machine and then played on the local machine, similar to the process of downloading and displaying a web page, this would entail waiting for the last second of the video file to be delivered before starting to look at it. Streaming video implies that the sender and the receiver are, respectively, the source and the sink for the video stream. That is, the source generates a video stream (perhaps using a video capture card), sends it across the Internet in messages, and the sink displays the stream as it arrives.

There are a variety of different classes of video applications. One class of video application is video-on-demand, which reads a preexisting movie from disk and transmits it over the network. Another kind of application is videoconferencing, which is in some ways the more challenging (and, for networking people, interesting) case because it has very tight timing constraints. Just as when using the telephone, the interactions among the participants must be timely. When a person at one end gestures, then that action must be displayed at the other end as quickly as possible. Too much delay makes the system unusable. Contrast this with video-on-demand where, if it takes several seconds from the time the user starts the video until the first image is displayed, the service is still deemed satisfactory. Also, interactive video usually implies that video is flowing in both directions, while a video-on-demand application is most likely sending video in only one direction.

One pioneering example of a videoconferencing tool, developed in the early and mid-1990s, is vic. Figure 1.1 shows the control panel for a vic session. vic is actually



one of a suite of conferencing tools designed at Lawrence Berkeley Laboratory and UC Berkeley. The others include a whiteboard application (wb) that allows users to send sketches and slides to each other, a visual audio tool called vat, and a session directory (sdr) that is used to create and advertise videoconferences. All these tools run on Unix—hence their lowercase names—and are freely available on the Internet. Many similar tools are available for other operating systems. It is interesting to note that while video over the Internet is still considered to be in its relative infancy at the time of this writing (2006), that the tools to support video over IP have existed for well over a decade.

Although they are just two examples, downloading pages from the Web and participating in a videoconference demonstrate the diversity of applications that can be built on top of the Internet, and hint at the complexity of the Internet's design. Starting from the beginning, and addressing one problem at time, the rest of this book explains how to build a network that supports such a wide range of applications. Chapter 9 concludes the book by revisiting these two specific applications, as well as several others that have become popular on today's Internet.

1.2 Requirements

We have just established an ambitious goal for ourselves: to understand how to build a computer network from the ground up. Our approach to accomplishing this goal will be to start from first principles, and then ask the kinds of questions we would naturally ask if building an actual network. At each step, we will use today's protocols to illustrate various design choices available to us, but we will not accept these existing artifacts as gospel. Instead, we will be asking (and answering) the question of *why* networks are designed the way they are. While it is tempting to settle for just understanding the way it's done today, it is important to recognize the underlying concepts because networks are constantly changing as the technology evolves and new applications are invented. It is our experience that once you understand the fundamental ideas, any new protocol that you are confronted with will be relatively easy to digest.

The first step is to identify the set of constraints and requirements that influence network design. Before getting started, however, it is important to understand that the expectations you have of a network depend on your perspective:

- An *application programmer* would list the services that his application needs, for example, a guarantee that each message the application sends will be delivered without error within a certain amount of time.
- A *network designer* would list the properties of a cost-effective design, for example, that network resources are efficiently utilized and fairly allocated to different users.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.