Larry L. Peterson & Bruce S. Davie

COMPUTER NETWORKS

A Systems Approach

Morgan Kaufmann Publishers, Inc.

Sponsoring Editor Jennifer Mann
Production Manager Yonie Overton
Production Editor Julie Pabst
Editorial Assistant Jane Elliott
Cover and Text Design Ross Carron Design
Illustration ST Associates, Inc.
Copyeditor Jeff Van Bueren
Proofreader Ken DellaPenta
Composition Ed Sznyter, Babel Press
Indexer Steve Rath
Printer Courier Corporation

Morgan Kaufmann Publishers, Inc.
Editorial and Sales Office
340 Pine Street, Sixth Floor
San Francisco, CA 94104-3205 USA
Telephone 415/392-2665
Facsimile 415/982-2665
Internet mkp@mkp.com
Order toll free 800/745-7323

Printed in the United States of America

00 99 98 97 96 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data
Peterson, Larry L.
Computer networks : a systems approach / Larry L. Peterson and Bruce S. Davie.
p. cm.
Includes bibliographical references and index.
ISBN 1-55860-368-9
1. Computer networks. I. Davie, Bruce S. II. Title.
TK5105.5.P479 1996
004.6'5—dc20

variable, however, then the delay experienced by the sequence of packets must have also been variable, and the network is said to have introduced jitter into the packet stream. Such variation is generally not introduced in a single physical link, but it can happen when packets experience different queuing delays in a multihop packet-switched network. This queuing delay corresponds to the Queue component of latency defined earlier in this section, which varies with time.

To understand the relevance of jitter, suppose that the packets being transmitted over the network contain video frames, and in order to display these frames on the screen the receiver needs to receive a new one every 33 ms. If a frame arrives early, then it can simply be saved by the receiver until it is time to display it. Unfortunately, if a frame arrives late, then the receiver will not have the frame it needs in time to update the screen, and the video quality will suffer; it will not be smooth. Note that it is not necessary to eliminate jitter, only to know how bad it is. The reason for this is that if the receiver knows the upper and lower bounds on the latency that a packet can experience, it can delay the time at which it starts playing back the video (i.e., displays the first frame) long enough to ensure that in the future it will always have a frame to display when it needs it. We return to the topic of jitter in Chapter 9.

1.3 Network Architecture

In case you hadn't noticed, the previous section established a pretty substantial set of requirements for network design—a computer network must provide general, cost-effective, fair, robust, and high-performance connectivity among a large number of computers. As if this weren't enough, networks do not remain fixed at any single point in time, but must evolve to accommodate changes in both the underlying technologies upon which they are based as well as changes in the demands placed on them by application programs. Designing a network to meet these requirements is no small task.

To help deal with this complexity, network designers have developed general blueprints—usually called a *network architecture*—that guide the design and implementation of networks. This section defines more carefully what we mean by a network architecture by introducing the central ideas that are common to all network architectures. It also introduces two of the most widely referenced architectures—the OSI architecture and the Internet architecture.

1.3.1 Layering and Protocols

When the system gets complex, the system designer introduces another level of abstraction. The idea of an abstraction is to define a unifying model that can capture some important aspect of the system, encapsulate this model in an object that provides an interface that can be manipulated by other components of the system, and hide the details of how the object is implemented from the users of the object. The challenge is to identify abstractions that si-

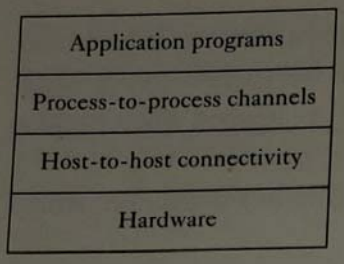

Figure 1.13 Example of a layered network system.

we introduced the idea of a channel in the previous section: we were providing an abstraction for applications that hides the complexity of the network from application writers.

Abstractions naturally lead to layering, especially in network systems. The general idea is that you start with the services offered by the underlying hardware, and then add a sequence of layers, each providing a higher (more abstract) level of service. The services provided at the high layers are implemented in terms of the services provided by the low layers. Drawing on the discussion of requirements given in the previous section, for example, one might imagine a network as having two layers of abstraction sandwiched between the application program and the underlying hardware, as illustrated in Figure 1.13. The layer immediately above the hardware in this case might provide host-to-host connectivity, abstracting away the fact that there may be an arbitrarily complex network topology between any two hosts. The next layer up builds on the available host-to-host communication service and provides support for process-to-process channels, abstracting away the fact that the network occasionally loses messages, for example.

Layering provides two nice features. First, it decomposes the problem of building a network into more manageable components. Rather than implementing a monolithic piece of software that does everything you will ever want, you can implement several layers, each of which solves one part of the problem. Second, it provides a more modular design. If you decide that you want to add some new service, you may only need to modify the functionality at one layer, reusing the functions provided at all the other layers.

Thinking of a system as a linear sequence of layers is an oversimplification, however. Many times there are multiple abstractions provided at any given level of the system, each providing a different service to the higher layers but building on the same low-level abstractions. To see this, consider the two types of channels discussed in Section 1.2.3: one provides a request/reply service and one supports a message stream service. These two channels might be alternative offerings at some level of a multilevel networking system, as illustrated in Figure 1.14.

Using this discussion of layering as a foundation, we are now ready to discuss the architecture of a network more precisely. For start-

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.