

Joe Casad

Fourth Edition

Sams **Teach Yourself**

# TCP/IP

in **24**  
**Hours**



**SAMS**

## **Sams Teach Yourself TCP/IP in 24 Hours**

Copyright © 2009 by Pearson Education, Inc.

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

ISBN-13: 978-0-672-32996-8

ISBN-10: 0-672-32996-4

*Library of Congress Cataloging-in-Publication Data:*

Casad, Joe, 1958-

Sams teach yourself TCP/IP in 24 hours / Joe Casad. — 4th ed.

p. cm.

Includes index.

ISBN-13: 978-0-672-32996-8 (pbk.)

ISBN-10: 0-672-32996-4 (pbk.)

1. TCP/IP (Computer network protocol) I. Title. II. Title: Teach yourself TCP/IP in 24 hours.

III. Title: TCP/IP in 24 hours.

TK5105.585.C37 2009

005.7'1376—dc22

2008031826

Printed in the United States of America

First Printing September 2008

### **Trademarks**

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

### **Warning and Disclaimer**

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an “as is” basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

### **Bulk Sales**

Sams Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

**U.S. Corporate and Government Sales**

**1-800-382-3419**

**corpsales@pearsontechgroup.com**

For sales outside of the U.S., please contact

**International Sales**

**international@pearson.com**

### **Editor-in-Chief**

*Mark Taub*

### **Acquisitions Editor**

*Trina MacDonald*

### **Development Editor**

*Michael Thurston*

### **Managing Editor**

*Kristy Hart*

### **Project Editor**

*Betsy Harris*

### **Indexer**

*Lisa Stumpf*

### **Proofreader**

*San Dee Phillips*

### **Technical Editor**

*Ravi Prakash*

### **Publishing Coordinator**

*Olivia Basegio*

### **Book Designer**

*Gary Adair*

### **Compositor**

*Nonie Ratcliff*

# Introduction

Welcome to *Sams Teach Yourself TCP/IP in 24 Hours, Fourth Edition*. This book provides a clear and concise introduction to TCP/IP for newcomers, and also for users who have worked with TCP/IP but would like a little more of the inside story. The fourth edition includes new material on recent developments in TCP/IP and offers a closer look at topics such as

- ▶ Firewalls
- ▶ Streaming
- ▶ Web services

You'll find new chapters on casting and streaming, web services, and the new Web, as well as several new sections throughout the book on recent developments in TCP/IP.

## Does Each Chapter Take an Hour?

Each chapter is organized so that you can learn the concepts within one hour. The chapters are designed to be short enough to read all at once. In fact, you should be able to read a chapter in less than one hour and still have time to take notes and reread more complex sections in your one-hour study session.

## How to Use This Book

The books in the *Sams Teach Yourself* series are designed to help you learn a topic in a few easy and accessible sessions. *Sams Teach Yourself TCP/IP in 24 Hours, Fourth Edition*, is divided into six parts. Each part brings you a step closer to mastering the goal of proficiency in TCP/IP.

- ▶ Part I, "TCP/IP Basics," introduces you to TCP/IP and the TCP/IP protocol stack.
- ▶ Part II, "The TCP/IP Protocol System," takes a close look at each of TCP/IP's protocol layers: the Network Access, Internet, Transport, and Application layers. You learn about IP addressing and subnetting, as well as physical networks and application services. You'll also learn about the protocols that operate at each of TCP/IP's layers.

## Sams Teach Yourself TCP/IP in 24 Hours

- ▶ Part III, “Networking with TCP/IP,” describes some of the devices, services, and utilities necessary for supporting TCP/IP networks. You learn about routing and network hardware, DHCP, DNS, and IPv6.
- ▶ Part IV, “TCP/IP Utilities,” introduces some of the common utilities used to configure, manage, and troubleshoot TCP/IP networks. You learn about Ping, Netstat, FTP, Telnet, and other network utilities.
- ▶ Part V, “TCP/IP and the Internet,” describes the world’s largest TCP/IP network: the Internet. You learn about the structure of the Internet. You also learn about HTTP, HTML, XML, email, and Internet streaming.
- ▶ Part VI, “Advanced Topics,” describes topics such as web services, messaging, the semantic web, and TCP/IP security. Part VI ends with a case study showing how the components of TCP/IP interact in a real working environment.

The concepts in this book, like TCP/IP itself, are independent of a system and descend from the standards defined in Internet Requests for Comment (RFCs).

## How This Book Is Organized

Each hour in *Sams Teach Yourself TCP/IP in 24 Hours, Fourth Edition*, begins with a quick introduction and a list of goals for the hour. You can also find the following elements.

### Main Section

Each hour contains a main section that provides a clear and accessible discussion of the hour’s topic. You’ll find figures and tables helping to explain the concepts described in the text. Interspersed with the text are special notes labeled *By The Way?* These notes come with definitions, descriptions, or warnings that help you build a better understanding of the material.

### ***By the Way***

These boxes clarify a concept that is discussed in the text. A *By The Way* might add some additional information or provide an example, but they typically aren’t essential for a basic understanding of the subject. If you’re in a hurry, or if you want to know only the bare essentials, you can bypass these sidebars.

## Q&A

Each hour ends with some questions designed to help you explore and test your understanding of the concepts described in the hour. Complete answers to the questions are also provided.

Additionally, some hours include Workshops—exercises designed to help you through the details or give you practice with a particular task. You’ll find them only in hours where a little real-world exploration will help build a better understanding of the material. Even if you don’t have the necessary software and hardware to undertake some of the exercises in the Workshop, you might benefit from reading through the exercises to see how the tools work in a real network implementation.

**By the  
Way**

## Key Terms

Each hour includes a summary of important key terms that are introduced in the hour. The key terms are compiled into an alphabetized list at the end of each hour.

## HOUR 20

# Web Services

---

### ***What you'll learn in this hour:***

- ▶ Web services
- ▶ XML
- ▶ SOAP
- ▶ WSDL
- ▶ Web transactions

The technologies of the Web have led to a new revolution in software development. The web service architecture lets the programmer leverage the tools of the Web for complex tasks never envisioned by the creators of HTML. This hour examines the web services infrastructure. You'll also get a quick look at how e-commerce websites process web transactions.

At the completion of this hour, you will be able to

- ▶ Discuss the web service architecture
- ▶ Understand the role of XML, SOAP, and WSDL in the web service paradigm
- ▶ Describe how e-commerce websites process monetary transactions

## **Understanding Web Services**

Now that almost every computer has a web browser, and web servers are widely understood, visionaries and software developers have been hard at work devising new ways to use the tools of the Web. In the old days, a programmer who wanted to write a network application had to create a custom server program, a custom client program, and a custom syntax or format for the two applications to exchange information. The effort of

writing all this software was a huge expense of time and brain space, but with the rising importance of computer networking, the goals of data integration and centralized management was driving the demand client server applications. Network program interfaces existed of course—otherwise many of the classic applications described in this book would have never evolved—but network programming typically required some significant, high-priced coding at the network interface.

An easier solution that emerged over time is to use the existing tools, technologies, and protocols of the Web as a basis for creating custom network applications. This approach, which is supported by big companies such as IBM and Microsoft, as well as open source advocates and development tool vendors around the world, is known as the **web services architecture**.

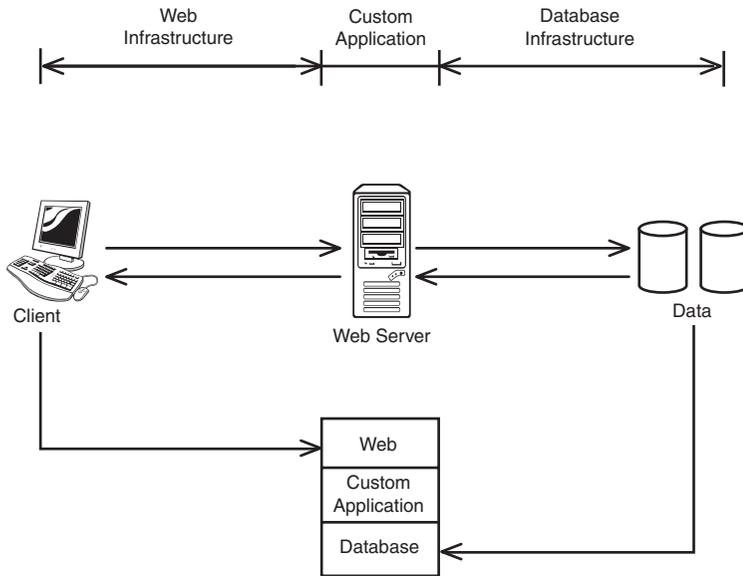
The idea behind the web services architecture is that the web browser, web server, and TCP/IP protocol stack handle the details of networking so the programmer can concentrate of the details of the application. In recent years, this technology has outgrown the original vision of the Web as a manifestation of the global Internet. This web services architecture is regarded now as an approach to building any sort of network application, whether that application is actually connected to the Internet. Large and powerful vendors such as Sun, Microsoft, and IBM have invested enormous resources in building component infrastructures to support this web services vision.

The HTTP delivery system is only part of what we know as web services. Also significant is the arrival of component architectures that provide ready-made classes, functions, and programming interfaces for working within a web-based environment.

Web service applications are often used in situations that require a simple client connection to a server that maintains inventory or processes orders. For instance, a manufacturing company might use a web services program to place orders, track deliveries, and maintain up-to-date information on the contents of the warehouse.

Almost any big company has a need for software that tracks appointments, orders, and inventory. A web service framework is good for gluing together disparate services and transactions into a single, unified environment.

Figure 20.1 shows a complete web services scenario. On the front end (the left side of Figure 20.1), the programmer can take advantage of the preexisting web infrastructure, which handles data transmission and also provides a user interface through the web browser application on the client computer. On the back end, the programmer relies on the preexisting data storage system provided by an SQL database. The programmer is left to concentrate on the center section of Figure 20.1, where the ready-made components of the web services platform further simplify the task of programming.



**FIGURE 20.1**  
The web services programming model.

Data passes through the components of the web services system in XML format. XML is an efficient, universal means for assigning values to attributes. Experts quickly recognized that the system would work even better if they could use the XML format to actually invoke services or generate responses over the network. Simple Object Access Protocol (SOAP) offers a standard method for passing XML-based data between web service processes. SOAP also describes how to use the XML and HTTP to invoke remote procedures. As you learn later in this hour, SOAP messages pass to and from network services defined through the Web Services Description Language (WSDL).

## XML

As soon as users, vendors, and web designers became accustomed to HTML, they started to ask for more. The growth of server-side and client-side programming techniques caused many experts to wonder if there might be a way to extend the rigid tag system of HTML. Their goal was to get beyond the conception of a markup language as a means for formatting text and graphics and to employ the language simply as a means for transmitting *data*. The result of this discussion was a new markup language called Extensible Markup Language, or XML.

As you learned earlier in this hour, the meaning and context for HTML data is limited to what you can express through a set of predefined HTML tags. If the data is

enclosed in <H1> tags, it is interpreted as a heading. If the data is enclosed in <A> tags, it is interpreted as a link. XML, on the other hand, lets users define their own elements. The data can signify whatever you want it to signify, and you can invent the tag you will use to mark the data. For instance, if you follow horse racing, you could create an XML file with information on your favorite horses. That file might contain entries such as:

```
<horses>
  <horse_name="winky" breed="Thoroughbred">
    <sex="male" />
    <age="3" />
  </horse>
  <horse_name="Goddess" breed="Arabian">
    <sex="female" />
    <age="3" />
  </horse>
  <horse_name="Gecko" breed="Uncertain">
    <sex="male" />
    <age="14" />
  </horse>
</horses>
```

XML format looks a little like HTML, but it certainly isn't HTML. (Can you imagine how much your browser would choke if you tried to pass off <horse\_name> as an HTML tag?) You can use whatever tag you want to use in XML, because you aren't preparing the data for some specific, rigidly predefined application like a web browser. The data is just data. The idea is that whoever creates the structure for the file will come along later to create an application or style sheet that will read the file and understand what the data means.

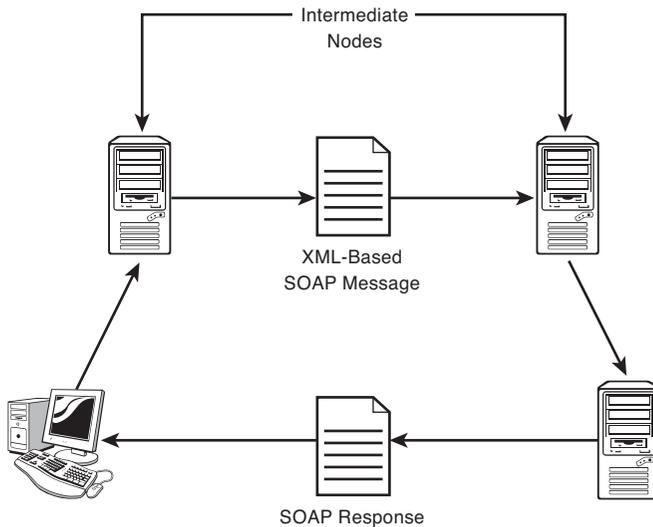
XML is an extremely powerful tool for passing data between applications. It is easy for a script or homegrown application to create XML as output or read XML as input. Even though a browser can't read XML directly, XML is still used extensively on the Web. In some cases, the XML data is generated on the server side and then converted to display-ready HTML before it is transmitted to the browser. Another technique is to provide an accompanying file called a Cascading Style Sheet (CSS) that tells how to interpret and display the XML data. However, XML is not limited to the web. Programmers now use XML for other contexts that require a simple, convenient format for assigning values to attributes.

XML now reaches far beyond the ordinary web as a format for storing and transmitting data. As long as the application that writes the XML data and the application that reads the data agree on the meaning of the elements, the data passes easily and economically between the applications through the miracle of XML.

## SOAP

XML defines a universal format for exchanging application data. The universal XML specification alone, however, is not enough to provide developers with the infrastructure they need to create easy and elegant web services. Although XML provides an efficient format for reading and writing program data, XML alone does not provide a standard format for structuring and interpreting that data. The SOAP specification fills that role. SOAP is a standard protocol for exchanging XML-based messages that pass between the web-service client and server.

SOAP is designed to support communication between so-called SOAP nodes. (A SOAP node is basically a computer or application that supports SOAP.) The SOAP specification defines the structure of a message that passes from the SOAP sender to the SOAP receiver. Along the way, the message might pass through intermediate nodes that process the information in some way (see Figure 20.2). An intermediate node might provide logging, or it might modify the message somehow in transit to its final destination.



**FIGURE 20.2**  
A SOAP message passes from the sender to the receiver and may pass through intermediate nodes.

At the conceptual level, a SOAP message from the client says “Here is some input. Process this and send me the output.” The functionality of the application derives from a series of these XML-based SOAP messages in which the endpoints send information and receive responses. The formal structure of the SOAP message allows the software developer to easily create a SOAP-based client application that interacts with the server. For instance, a rental company that provides car rental reservations

through a web-based server application could easily make the specifications available for a developer to write a custom client application that could connect to the server and reserve a car.

The structure of a SOAP message consists of an optional header and a message body. The header contains callouts, definitions, and meta-information that will be used by any node along the message path. The body includes data intended for the message recipient. For example, in the case of the car reservation service, the message body might contain data from the client describing the car the customer would like to rent and the date the vehicle must be available.

## WSDL

The Web Services Description Language (WSDL) provides an XML format for describing the services associated with the web service application. According to the W3C's WSDL specification, "WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information." WSDL is a format for defining the services that exchange information through SOAP messages.

A WSDL document is primarily a set of definitions. The definitions within the document specify information on the data being transmitted and the operations associated with that data, as well as other data related to the service and the service location.

WSDL is not confined to SOAP but is also used with other web service communication protocols. In some cases, WSDL is used directly with HTTP to simplify the design and restrict the actions to more fundamental GET and POST-style operations at the heart of HTTP.

## Web Service Stacks

Armed with XML, SOAP, WSDL, and the underlying components of TCP/IP and web service frameworks, a developer can easily create light and simple client and server applications that communicate through a web interface. Like TCP/IP itself, a web service environment consists of a stack of components. Major vendors have their own web service stacks that they provide to customers. The complete system forms a package of server software, developer tools, and even computer hardware that is provided to the client, along with consulting services and, sometimes, made-to-order custom applications.

Linux vendors and developers often talk about the LAMP stack, a collection of open source components that is easily tailored for web service environments. The memorable acronym LAMP spells out the principal components of the stack:

- ▶ **Linux**—An operating system that supports server applications running on the server system
- ▶ **Apache**—A web server that serves up XML-based SOAP messages
- ▶ **MySQL**—A database system that provides access to back-end data services
- ▶ **PHP (or Perl or Python)**—A web-ready programming language used to code the details of the custom web service application

Proprietary web service infrastructures provide similar features. The Java programming language is often used with web services—not just by Sun (the creators of Java), but also in IBM's WebSphere and other systems. Microsoft provides equivalents to Java through the tools of the .NET framework.

## E-Commerce

An e-commerce site is not necessarily an implementation of the web service paradigm described earlier in this hour; however, it still might use some web-service techniques, especially on the back end. E-commerce is a high-profile example of the way applications and components can be combined together using the tools of the Web.

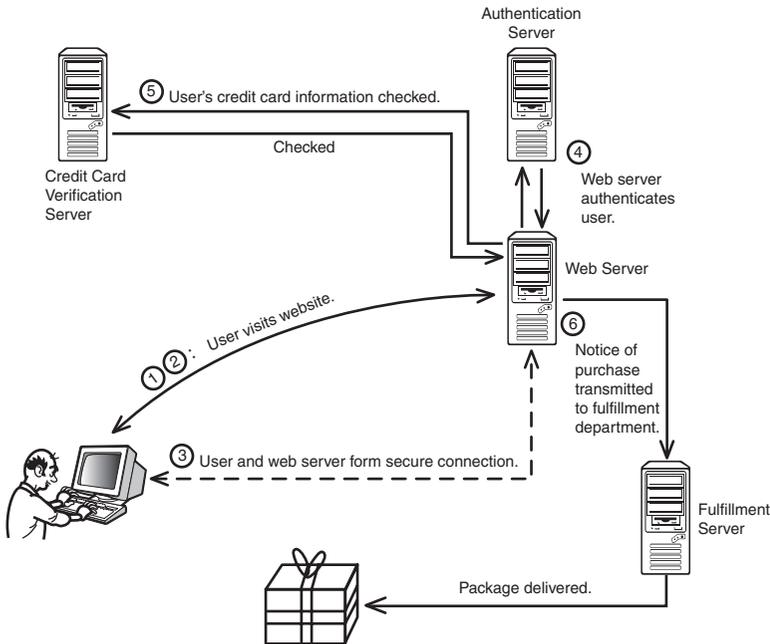
Vendors and advertisers began to notice early on that the Web is a great way to get people to buy things. It is no secret that many websites look like long, intricate advertisements. Despite the hype, which is enough to make anyone doubt the validity of the design, the fact is that the Web is a convenient and cost-effective way to shop. Rather than sending thousands of catalogs by direct mail, a vendor can simply post the catalog on the Web and let the customers find it through searches and links.

The business of buying over the Web did not really get started until vendors solved the security issues related to sending credit card information over the open Internet. In fact, Internet sales would not even be possible without the secure networking techniques. Most browsers are now capable of opening a secure communications channel with the server. This secure channel makes it impossible for a cyber thief to listen for passwords or credit card information.

A typical web transaction scenario is shown in Figure 20.3. The process is as follows:

- 1.** A web server provides an online catalog accessible from the Web. A user browses through the product offerings from a remote location across the Internet.
- 2.** The user decides to buy a product and clicks a Buy This Product link on the web page.
- 3.** The server and browser establish a secure connection. (See Hour 23, “TCP/IP Security,” for more on SSL and other secure communication techniques.) At this point, the browser sometimes displays a message that says something like “You are now entering a secure area....” Different browsers have different methods for indicating a secure connection.
- 4.** After the connection is established, some form of authentication usually follows. On most transaction sites, the buyer establishes some form of user account with the vendor. This is partly for security reasons and partly for convenience (so the user can track the status of purchases). The user account information also lets the vendor track the behavior of the user and correlate the user’s demographic information and purchase history. This logon step requires the web server to contact some form of back end database server—either to establish a new account or to check the credentials for logon to an existing account.
- 5.** After the user is logged in, the server (or some application working on the server back end) must verify the credit card information and register the transaction with some credit card authority. Often this credit card authority is a commercial service affiliated with the credit card company.
- 6.** If the transaction is approved, notice of the purchase and mailing information is transmitted to the vendor’s fulfillment department, and the transaction application attends to the final details of confirming the purchase with the user and updating the user’s account profile.

Operating system vendors such as Sun and Microsoft offer transaction server applications to assist with the important task of processing orders over the web. Because web transactions are highly specialized, and because they require an interface with existing applications on the vendor’s network, application frameworks often provide special tools to assist with the task of constructing a transaction infrastructure.



**FIGURE 20.3**  
A typical web transaction scenario.

Note that Figure 20.3 omits the role of the firewall within the transaction infrastructure. A large-scale commercial network might include a firewall behind the web server, protecting the network, and another firewall in front of the web server that blocks some traffic but leaves the server open to web requests. Also, on high-volume websites, you're more likely to find a collection of web servers sharing the load, rather than a single server.

Connections from the web server to the back-end servers could be across a protected internal network. Alternatively, the connection to the back end could be through a dedicated line that is separate from the main network. The credit card verification server is often an off-site service provided by a different company and accessed through a secure Internet connection.

**By the  
Way**

## Summary

The tools of the Web provide a backdrop for many kinds of application development. In addition to simple web pages and web forms, developers are putting together complex applications that place reservations, track inventory, and process purchase orders. This hour described some of the technologies at the heart of the web service paradigm. You learned about the web service infrastructure and why it

is important. This hour also discussed three important web service components: XML, SOAP, and WSDL. Lastly, this hour took a look at the structure of web-based transactions.

## Q&A

- Q. *What is the advantage of the web service model over conventional client-server programming?***
- A.** The web service model is design to integrate standard components that are already present on most networks, such as web server and web browser applications.
- Q. *Why is the web service model based on XML instead of HTML?***
- A.** HTML is predefined collection of tags intended specifically as a markup language for web pages. XML has nearly unlimited capacity for defining new elements and assigning values to variables.
- Q. *Considering that countless vendors all have their own languages and components for supporting web services, what is the benefit of uniform standards like SOAP and WSDL?***
- A.** Standards like SOAP and WSDL provide a common format so that components written for different vendor environments can easily interact.

## Key Terms

Review the following list of key terms:

- ▶ **LAMP**—An open source web service stack consisting of the Linux operating system, the Apache web server, the MySQL database system, and any of three programming languages that start with “P” (PHP, Perl, or Python).
- ▶ **SOAP**—A message exchange protocol for web applications.
- ▶ **Web service architecture**—A paradigm for building custom network applications around web components.
- ▶ **WSDL (Web Services Description Language)**—An XML-based format for describing network services.
- ▶ **XML (eXtensible Markup Language)**—A markup language used for defining and transmitting program data in a web service application.

# Index

## Symbols

// (double slash), 302  
32-bit binary addresses,  
converting to dotted decimal  
format, 57-59  
802.11 frame formats, 159-160  
802.11 networks, 156-157  
independent and  
infrastructure networks,  
157-160  
security, 161-162  
802.11a, 157  
802.11b, 157  
802.11i, 162

## A

A, 198  
access method (network  
architecture), 38  
access points, 157  
ACK, 95, 104  
Ack (acknowledge), 150  
Acknowledgment Number  
field, 104  
active open TCP connections, 96  
address classes, 55, 138  
Address Resolution Protocol.  
See ARP  
addresses  
destination addresses, 159  
logical addressing, 12

loopback addresses, 188  
receiver addresses, 159  
source addresses, 159  
transmitter addresses, 159  
addressing  
Internet layer, 48-50  
IP addressing, 55-57  
converting 32-bit binary  
addresses to dotted  
decimal format, 57-59  
converting decimal  
numbers to binary  
octets, 59-61  
special IP addresses, 61  
IPv6, 236-237  
adolescent amateurs, 376  
Advanced Research Projects  
Agency (ARPA), 10  
AES (Advanced Encryption  
Standard), 397  
algorithms  
dynamic routing, 130-131  
encryption, 393-395  
AOL Instant Messenger  
(AIM), 371  
APIPA (Automatic Private IP  
Addressing), 225  
APIs (Application Programming  
Interfaces), 114-115  
Application layer, 107-108  
APIs, 114-115  
network services, 109-113  
protocols, 110

## Application layer

- TCP/IP protocol system, 24
- TCP/IP utilities, 115-116
- TCP/IP versus OSI, 108-109
- application support, 16-17**
- Application-level, 108**
- application-level attacks, 377, 383-385**
- Archie, 116**
- architectures**
  - Network Access layer, 39
  - network architecture, 38-40
- ARP (Address Resolution Protocol), 30, 41, 48, 62-63, 115**
  - cache, 62
  - protocol dysfunction/misconfiguration, 245, 251-252
  - request frame, 62
  - table, 62
- arp -a, 251**
- arp -d, 252**
- arp -g, 251**
- arp -s, 251**
- ARPA (Advanced Research Projects Agency), 10**
- ARPAnet, 10**
- ascii command, 265**
- association, 160**
- asymmetric encryption, 397-398**
  - digital certificates, 400-402
  - digital signatures, 398-400
- attackers, 376-378**
  - application-level attacks, 383-385
  - credential attacks, 378-382
  - network-level attacks, 382-383
- attacks**
  - application-level attacks, 377, 383-385
  - credential attacks, 377-382
  - denial-of-service attacks, 377, 387-388
  - network-level attacks, 377, 382-383

- phishing, 386-387
- root access, 385
- authentication, 161, 392**
- Authentication header, 235**
- authenticators, 399**
- Automatic Private IP Addressing (APIPA), 225**
- autonomous systems, 134**
- .AVI (Audio Visual Interleave), 344**

## B

- B-Node name resolution, 205**
- back door, 389**
- batch mode, 201**
- Bellman Ford routing, 131**
- Berkeley Internet Name Domain (BIND), 197**
- Berkeley r\* utilities, 278-279**
  - rcp, 280
  - rexec, 281
  - rlogin, 279-280
  - rsh, 280-281
  - ruptime, 281
  - nwho, 281-282
- Berkeley Systems Design (BSD) Unix, 278**
- Berners-Lee, Tim, 306**
- BGP (Border Gateway Protocol), 64, 136**
- binary command, 265**
- binary octets, converting decimal numbers to, 59-61**
- binary patterns, 78-79**
- BIND (Berkeley Internet Name Domain), 197**
- BinHex, 323**
- blogs, 364-365**
- blowfish, 410**
- Bluetooth, 165-167**
- body, 319**
- BOOTP (Boot PROM), 63, 228**
- Border Gateway Protocol (BGP), 64, 136**
- bridges, 167-168**
- broadcast storms, 254**
- broadcasts, 99, 205**

- browsers, 116, 319, 344**
- buffer overflow, 383-384**
- buffer size field, 98**
- buffers, 342, 383**
- bye command (FTP), 266**

## C

- CA (certificate authority), 410**
- cable broadband, 151-153**
- Cable Modem Termination System (CMTS), 152**
- cabling rules, 39**
- cabling type, 38**
- caches, 196**
- caching only servers, 196**
- Carrier Sense Multiple Access with Collision Detect (CSMA/CD), 41**
- cd command, 264**
- Cerf, Vinton, 10**
- certificate authority (CA), 410**
- certificate servers, 400**
- certification paths, 402**
- CGI (Common Gateway Interface), 317**
- checksum (Transport layer), 104**
- CIDR (Classless Internet Domain Routing), 52, 70, 80-81, 138**
- Class A addresses, 55**
- Class B addresses, 55**
- Class C addresses, 55**
- classless routing, 138-139**
- client commands (SMTP), 327**
- clients, 300**
- close command (FTP), 266**
- closing TCP connections, 92, 99**
- CMTS (Cable Modem Termination System), 152**
- CNAME, 198**
- .com top-level domain, 195**
- Common Gateway Interface (CGI), 317**
- community strings, 285**
- confidentiality, 161**
- configuration information utilities, 245, 248-250**

## Digital Service Line Access Multiplexer (DSLAM)

- configurationless connectivity, 225
  - configuring
    - DHCP, 220-221
    - DHCP servers, 221-222
    - DNS servers, 197
      - reverse lookup zone files, 200
      - zone files, 198-200
  - connection-oriented protocols, 85-86
  - connectionless protocols, 85-86
  - connections. *See also* connectivity problems
    - cable broadband, 151-153
    - dial-up networking, 144
      - modem protocols, 145-147
      - point-to-point connections, 144-145
      - PPP, 147-151
    - DSL (Digital Subscriber Line), 153
    - PPP connections, 149
    - TCP connections, 96-98
    - WANs (Wide Area Networks), 154-155
    - wireless networking, 155-156
      - 802.11 networks, 156-162
      - Bluetooth, 165-167
      - Mobile IP, 164-165
      - WAP (Wireless Application Protocol), 162-164
  - connectivity devices, 167
    - bridges, 167-168
    - hubs, 168-169
    - switches, 169-171
  - connectivity problems, 244-245
    - excessive traffic, 245
    - faulty name resolution, 245, 253-254
    - line problems, 244, 252
    - network performance problems, 254
      - nbtstat, 259-260
      - netstat, 257-258
      - packet sniffers, 260-261
      - route, 256-257
      - traceroute, 254-256
    - protocol dysfunction/misconfiguration, 244-245
      - ARP, 245, 251-252
      - configuration information utilities, 245, 248-250
      - ping, 245-247
    - troubleshooting with connectivity utilities, 261
  - connectivity utilities
    - Application layer, 115
    - troubleshooting connectivity problems, 261
  - content caching, 182
  - control flags, 105
  - conventional encryption, 395-397
  - converting
    - 32-bit binary addresses to dotted decimal format, 57-59
    - decimal numbers to binary octets, 59-61
    - subnet masks to dotted decimal notation, 73-75
  - core routers, 135
  - CRC (Cyclical Redundancy Check), 44
  - credential attacks, 377-382
    - guessing, 380
    - intercepting, 380-381
    - Trojan horses, 379-380
  - CSMA/CD (Carrier Sense Multiple Access with Collision Detect), 41
  - Cyclical Redundancy Check (CRC), 44
- ## D
- daemons, 262
  - Data Encryption Standard (DES), 397
  - data frame format (network architecture), 38
  - data frames, 41
  - data packages, 26-28
  - Datagram Congestion Control Protocol (DCCP), 101, 344
  - datagrams, 27, 405
    - Internet layer, 49
    - PPP, 148
    - UDP, 100
  - DCCP (Datagram Congestion Control Protocol), 101, 344
  - decimal numbers, converting to binary octets, 59-61
  - default gateways, 126
  - default routers, 126
  - delivering data, 48-50
  - demultiplexing, 84, 90-91
  - denial-of-service attacks, 377, 387-388
  - DES (Data Encryption Standard), 397
  - destination addresses, 159
  - Destination Options header, 234
  - destination port, 105
  - Destination Unreachable, 64
  - DHCP (Dynamic Host Configuration), 203, 216-217
    - configuring, 220-221
    - leasing IP addresses, 217-218
    - relay agents, 219
    - time fields, 220
  - DHCP servers, configuring, 221-222
  - DHCPACK, 218
  - DHCPDISCOVER, 217
  - DHCPOFFER, 217
  - DHCPREQUEST, 218
  - diagrams, 24
  - dial-up networking, 144
    - modem protocols, 145-147
    - point-to-point connections, 144-145
    - PPP, 147-151
  - digital certificates, 400-402
  - Digital Over Cable Service Interface Specification (DOCSIS), 153
  - Digital Service Line Access Multiplexer (DSLAM), 153

## digital signatures

digital signatures, 398-400  
 Digital Subscriber Line (DSL), 153  
 dir command, 264  
 direct routing versus indirect routing, 128-130  
 Direct Sequence Spread Spectrum (DSSS), 156  
 distance vector routing, 131-133  
 dividing networks, 70-73  
 DMZ firewalls, 178-180  
 DNS (domain name service), 112  
   dynamic DNS, 203  
   managing, 196  
     configuring DNS servers, 197-200  
     utilities, 200-203  
   name resolution, 187-195  
 DNS Service Discovery, 226  
 DNS SP, 226  
 DOCSIS (Digital Over Cable Service Interface Specification), 153  
 !DOCTYPE  
   HTML, 310  
 domain name resolution (Hypothetical Inc. case study), 419-420  
 domains, 187  
   registering, 195-196  
 dotted decimal format, 55, 240  
   converting 32-bit binary addresses to, 57-59  
   converting subnet masks to, 73-75  
 double slash (/), 302  
 downloadable workspace, 240  
 DSL (Digital Subscriber Line), 153  
 DSLAM (Digital Service Line Access Multiplexer), 153  
 DSSS (Direct Sequence Spread Spectrum), 156  
 dynamic addresses (Hypothetical Inc. case study), 418  
 dynamic DNS, 203  
 Dynamic Host Configuration Protocol. *See* DHCP  
 Dynamic HTML, 316-317

dynamic routing, 125, 129  
   algorithms, 130-131  
   development of TCP/IP, 11

**E**

e-commerce, 359-361  
 EAP (Extensible Authentication Protocol), 162  
 Echo Reply, 64  
 Echo Request, 64  
 EGP (Exterior Gateway Protocol), 135  
 email, 321-322. *See also* SMTP  
   formats, 322-324  
   how it works, 324-326  
   MIME, 323  
   retrieving, 328-330  
   spam, 334-336  
   webmail, 333-334  
 email readers, 116, 331-333  
 email viruses, 333  
 email worm, 389  
 encapsulation, 24  
 Encrypted Security Payload header (ESP), 235  
 encryption, 391-392  
   algorithms and keys, 393-395  
   asymmetric encryption, 397-398  
     digital certificates, 400-402  
     digital signatures, 398-400  
   symmetric encryption, 395-397  
 encryption algorithms, 393  
 encryption keys, 393  
 end node verification, 11  
 error control, 15  
 ESP (Encrypted Security Payload header), 235  
 ethernet, 41-42  
   ethernet frames, 43-44  
   physical addressing, 41  
 ethernet frames, 43-44  
 excessive traffic, 245

expiration time, 199  
 Extensible Authentication Protocol (EAP), 162  
 eXtensible Markup Language (XML), 318  
 eXtensible Messaging and Presence Protocol (XMPP), 371  
 Exterior Gateway Protocol (EGP), 135  
 exterior routers, 135

**F**

faulty name resolution (connectivity problems), 245  
 FCS (Frame Check Sequence), 44  
 feature negotiation, 348  
 FHSS (Frequency Hopping Spread Spectrum), 156  
 fields  
   buffer size field, 98  
   header fields (IP), 52-54  
   Window field, 98  
 file and print services, 111-112  
 file formats  
   browsers, 344  
   video file formats, 344  
 File Transfer Protocol, 110-112, 262-263  
 File Transfer utilities, 115  
 FIN, 95, 99  
 fin-wait state, 99  
 Finger, 110, 116  
 firewall rules, 180-181  
 firewall/router devices, 177  
 firewalls, 177  
   defined, 175-176  
   DMZ, 178-180  
   firewall/router devices, 177  
   home firewall, 387  
   Hypothetical Inc. case study, 420  
   options for, 177-178  
   packet filters, 176  
   personal firewalls, 177  
   proxy servers, 181-182  
   reverse proxy, 182-183

- stateful firewalls, 177
- Transport layer, 101-102
- flow control, 15**
  - TCP connections, 98
  - TCP Transport layer, 92
- flow level, 240**
- FONT tag, 312**
- footprinting, 378**
- formats of email, 322**
  - header fields, 322-324
- FQDN (fully qualified domain name), 187**
- Fragment header, 235**
- Fragmentation Needed, 64**
- Frame Check Sequence (FCS), 44**
- frames, 28**
- Frequency Hopping Spread Spectrum (FHSS), 156**
- ftp, 17, 262-266**
- FTP (File Transfer Protocol), 110-112, 262-263**
- ftp command, 263**
- fully qualified domain name (FQDN), 187**

## G-H

- Gateway-to-Gateway Protocol (GGP), 135**
- gateways, 12**
  - default gateway, 126
  - interior gateways, 136
- get command**
  - FTP, 265
  - HTTP, 314
- GGP (Gateway-to-Gateway Protocol), 135**
- Gopher, 116**
  - .gov, 195
- graylists, 336**
- guessing credential attacks, 380**
- head, 319**
- header fields, 26**
  - email, 322-324
  - HTTP, 315-316
  - IP (Internet Protocol), 52-54

- IPv6, 232-234
  - Authentication header, 235
  - Destination Options header, 234
  - ESP header, 235
  - Fragment header, 235
  - Hop-by-Hop Options header, 234
  - Routing header, 234-235
  - pseudo-headers, 101
- help command, 264**
- High Rate Direct Sequence Multiplexing (HR/DSSS), 156**
- home firewalls, 387**
- Hop-by-Hop Options header, 234**
- hop counts, 131**
- hop limit, 240**
- host files, 186-189**
- host ID, 51**
- Hostname, 115, 254**
- hostnames, 186**
- HR/DSSS (High Rate Direct Sequence Multiplexing), 156**
- HTML (Hypertext Markup Language), 305, 308, 313**
  - !DOCTYPE, 310
  - Dynamic HTML, 316-317
  - tags, 309-312
- HTTP (Hypertext Transfer Protocol), 111, 305, 313-316**
  - GET command, 314
  - header fields, 315-316
  - status codes, 314-315
  - web services, 113
- hubs, 168-169**
- hypertext, 306**
- Hypertext Markup Language. See HTML**
- Hypertext Transfer Protocol. See HTTP**
- Hypothetical, Inc.**
  - history of, 413-414
  - TCP/IP
    - domain name resolution, 419-420
    - dynamic addresses, 418

- firewalls, 420
  - getting started with, 415-417
  - segmenting, 417-418
  - signatures and VPNs, 421-422
  - web services, 421

- IAB (Internet Architecture Board), 17**
- IBSS (Independent Basic Service Set), 157**
- ICANN (Internet Corporation for Assigned Names and Numbers), 13, 17, 195**
- ICMP (Internet Control Message Protocol), 63-64**
- IEEE 802.11, 39**
- IEEE 802.16, 39**
- IEEE 802.3, 39**
- IETF (Internet Engineering Task Force), 17**
- ifconfig command, 248-250**
- IGP (Interior Gateway Protocols), 136**
- IM (Instant Messaging), 369-371**
- IMAP (Internet Message Access Protocol), 111**
- IMAP4 (Internet Message Access Protocol version 4), 330**
- Independent Basic Service Set (IBSS), 157**
- independent networks, 157-160**
- indirect routing, 128-130**
- Infrastructure Basic Service Set (Infrastructure BSS), 157**
- Infrastructure BSS, 158**
- infrastructure networks, 157-160**
- initial sequence number (ISN), 97**
- Instant Messaging (IM), 369-371**
- integrating network file access, 268-269**
  - NFS, 269
  - SMB, 270
- integrity, 161, 392**
- intelligent hubs, 169**

## interactive mode

- interactive mode, 201
  - intercepting credential attacks, 380-381
  - Interior Gateway Protocols (IGP), 136
  - interior gateways, 136
  - interior routers, 136
    - OSPF (Open Shortest Path First), 138
    - RIP (Routing Information Protocol), 137-138
  - Internet, 306
    - development of TCP/IP, 10-11
    - security, 299
    - what happens on Internet, 299-301
  - Internet Architecture Board (IAB), 17
  - Internet Control Message Protocol (ICMP), 63-64
  - Internet Corporation for Assigned Names and Numbers (ICANN), 17
  - Internet Engineering Task Force (IETF), 17
  - Internet eXchange Points (IXPs), 298
  - Internet intruders, 375
  - Internet layer
    - addressing and delivering, 48-50
    - ARP, 62-63
    - ICMP (Internet Control Message Protocol), 63-64
    - IP (Internet Protocol), 50-52
      - IP addressing, 55-61
      - IP header fields, 52-54
    - IPsec protocols, 65
    - RARP, 63
    - TCP/IP protocol system, 23
  - Internet Message Access Protocol (IMAP), 111
  - Internet Protocol. *See* IP
  - Internet Relay Chat (IRC), 369-371
  - Internet Research Task Force (IRTF), 17
  - Internet service provider (ISP), 134
  - Internet topology, 297-299
  - Internet utilities, Application layer, 116
  - internetwork, 23
  - IP (Internet Protocol), 50-52
    - IP addressing, 48, 55-57
      - address class system, 55
      - converting 32-bit binary addresses to dotted decimal format, 57-59
      - converting decimal numbers to binary octets, 59-61
      - leasing addresses from DHCP, 217-218
      - server-supplied IP addresses, 215-216
      - special IP addresses, 61
      - static IP addressing, 216
    - IP forwarding, 127-128
    - IP header fields, 52-54
    - IP next generation. *See* IPv6
    - IP Reservation, 222
    - IPv4, 237-238
      - reasons for updating, 229-230
  - IPConfig, 115
  - IPSec (IP Security), 65, 404
  - IPv4-compatible IPv6 address, 238
  - IPv4-mapped IPv6 address, 238
  - IPv6, 230-232
    - addressing, 236-237
    - goals of, 231
    - header formats, 232-234
      - Authentication header, 235
      - Destination Options header, 234
      - ESP header, 235
      - Fragment header, 235
      - Hop-by-Hop Options header, 234
      - Routing header, 234-235
    - with IPv4, 237-238
    - QoS, 238-239
  - IRC (Internet Relay Chat), 369-371
  - IRTF (Internet Research Task Force), 17
  - ISN (initial sequence number), 97
  - ISPs (Internet service providers), 134
  - iterative queries, 193
  - IXPs (Internet eXchange Points), 298
- ## J-K
- jumbo payload option, 234
  - Kahn, Robert E., 10
  - KDC (Key Distribution Center), 407
  - Kerberos, 406-409
  - Key Distribution Center (KDC), 407
  - keys
    - encryption, 393-395
    - long-term keys, 407
    - private keys, 397
    - public keys, 397
- ## L
- L2TP (Layer 2 Tunneling Protocol), 405
  - LAMP stack, 359
  - LANs (local area networks), 11
  - Layer 2 switches, 171
  - Layer 2 Tunneling Protocol (L2TP), 405
  - Layer 3 switches, 171
  - layers, 24
  - LCP (Link Control Protocol), 149-151
  - LDAP (Lightweight Directory Access Protocol), 111
  - line problems, 244, 252
  - Link Control Protocol (LCP), 149-151
  - Link Local Addressing, 225-226
  - link state routing, 132-134, 138
  - link status lights, 252
  - Link-Local Multicast Name Resolution (LLNR), 226

LLC (Logical Link Control), 37  
 LLNR (Link-Local Multicast Name Resolution), 226  
 LMHosts files, 205-207  
 local area networks (LANs), 11  
 logical addressing, 12-13  
 Logical Link Control (LLC), 37  
 long-term keys, 407  
 loopback addresses, 188  
 lpr, 17  
 ls command, 264

## M

MAC (Media Access Control), 12, 37. *See also* physical addressing  
 mailboxes, 338  
 Management Information Base (MIB), 285  
 managing DNS, 196
 

- configuring DNS servers, 197-200
- utilities, 200-203

 markup language, 306  
 Maximum Receive Unit (MRU), 150  
 maximum transmission unit (MTU), 235  
 mDNS, 226  
 Media Access Control (MAC), 12, 37. *See also* physical addressing  
 messages, 27  
 methods for NetBIOS name resolution, 204-205
 

- broadcasts, 205
- LMHosts files, 205-207
- WINS (Windows Internet Name Service), 207-210

 mget command, 265  
 MIB (Management Information Base), 285-287  
 .mil, 195  
 MIME (Multipurpose Internet Mail Extensions), 322-323  
 minimum time-to-live (TTL), 199  
 mkdir command, 265

Mobile IP, 164-165  
 modems, 144
 

- cable modems, 152
- modem protocols, 145-151

 mounting, 269  
 .MOV (QuickTime), 345  
 .MPEG (Motion Picture Experts Group), 345  
 mput command, 265  
 MRU (Maximum Receive Unit), 150  
 MTU (maximum transmission unit), 235  
 multicasts, 67  
 multihomed computers, 122-123  
 multiple DNS, 226  
 multiplexing, 84, 90-91  
 Multipurpose Internet Mail Extensions (MIME), 322

## N

NaK (not acknowledged), 150  
 name resolution, 15, 185-187
 

- checking
  - with NSLookup, 201-203
  - with Ping, 201
- connectivity problems, 253-254
- DNS (domain name system), 187, 189-195
- host files, 186-189
- hostnames, 186
- NetBIOS, 204
  - methods for, 204-210
  - testing, 210-211
- network services, 112-113
- subdomains, 193

 name servers, 15, 112  
 NAT (Network Address Translation), 61, 223-224, 230  
 NBTstat, 115, 259-260
 

- nbtstat -A, 260
- nbtstat -c, 259
- nbtstat -n, 259
- nbtstat -r, 259
- nbtstat -S, 260

 NDIS (Network Driver Interface Specification), 38  
 .net, 195  
 net view command, 210  
 NetBIOS name resolution, 204
 

- methods for, 204-210
- testing, 210-211

 Netstat, 115, 257-258
 

- netstat -a, 258
- netstat -e, 258
- netstat -n, 258
- netstat -p TCP, 258
- netstat -p UDP, 258
- netstat -r, 258
- netstat -s, 257

 Network Access layer, 35-36
 

- architectures, 39
- ethernet. *See* ethernet, 43
- OSI model and, 37-38
- physical addressing, 40-41
- TCP/IP protocol system, 23-24

 Network Address Translation (NAT), 61, 223-224, 230  
 network architecture, 38-40  
 Network Driver Interface Specification (NDIS), 38  
 network file access, integrating, 268-269
 

- NFS, 269
- SMB, 270

 Network File System (NFS), 111, 269  
 network ID, 51  
 network intruders, 375
 

- denial-of-service attacks, 387-388
- phishing, 386-387
- root access, 385
- what they want, 376-378
  - application-level attacks, 383-385
  - credential attacks, 378-382
  - network-level attacks, 382-383

## network performance problems

## network performance problems, 254

- nbtstat, 259-260
- netstat, 257-258
- packet sniffers, 260-261
- route, 256-257
- tracert, 254-256

## network protocols, 8-9

## network services, Application layer, 109-110

- file and print services, 111-112
- name resolution services, 112-113
- remote access, 113
- web services, 113

## Network Time Protocol (NTP), 111

## network-level attacks, 377, 382-383

## networks

- defined, 8
- dial-up networking, 144
  - modem protocols, 145-147
  - point-to-point connections, 144-145
  - PPP, 147-151
- dividing, 70-73
- organizing, 71
- routing on complex networks, 134-136
- wireless networking, 155-156
  - 802.11 networks, 156-162
  - Bluetooth, 165-167
  - Mobile IP, 164-165
  - WAP (Wireless Application Protocol), 162-164

## Newsreaders, 116

## NFS (Network File System), 111, 269

## NS (Name Server), 198

## NSLookup, 201-203, 253

## NTP (Network Time Protocol), 111

**O**

## octets, 55

## ODI (Open Data-Link Interface), 38

## OFDM (Orthogonal Frequency Division Multiplexing), 156

## open command, 265

## Open Data-Link Interface (ODI), 38

## Open Shortest Path First (OSPF), 136-138

## Open Systems Interconnection model. See OSI model

## OpenSSH, 282

## .org, 195

## organizing networks, 71

## Orthogonal Frequency Division Multiplexing (OFDM), 156

## OSI (Open Systems Interconnection) model, 24

## OSI model

- Application layer, 108-109
- Network Access layer, 37-38
- Physical layer, 37
- TCP/IP and, 24-26

## OSPF (Open Shortest Path First), 136-138

**P**

## P2P (peer-to-peer), 368-369

## packets

- packet filters, 176
- packet sniffers, 260-261
- PPP RFCs, 148

## PAN (Personal Area Network), 166

## passive open, 96

## path MTU, 235

## payload length, 240

## peer-to-peer (P2P), 368-369

## Personal Area Network (PAN), 166

## personal firewalls, 177

## phishing, 386-387

## PHP, 317

## physical addressing, 40-41, 48

## ping, 17, 115

- checking name resolution, 201
- protocol dysfunction/misconfiguration, 245-247

## podcasting, 346-347

## Point of Presence (POP), 298

## point-to-point connections, 144-145

## Point to Point Protocol. See PPP

## Point to Point Tunneling Protocol (PPTP), 405

## POP (Point of Presence), 298

## POP (Post Office Protocol), 111

## POP3 (Post Office Protocol version 3), 330

## ports

- destination ports, 105
- Transport layer, 87-88
  - TCP 88-89
  - UDP 89
- well-known ports, 88

## Post Office Protocol (POP), 111

## PPP (Point to Point Protocol), 39, 147-149

- connections, 149
- datagrams, 148
- LCP, 149-151
- PPPoE (PPP over Ethernet), 153
- RFCs, 148

## PPPoE (PPP over Ethernet), 153

## PPTP (Point to Point Tunneling Protocol), 405

## preambles, 43

## Presentation layer, 109

## primary name servers, 196

## print service, 117

## private keys, 397

## problems

- with connections. See connectivity problems
- network performance problems. See network performance problems
- with streaming, 339-340

process/Application-level, 108  
 professional attackers, 376  
 protocols, 8-9  
   Application layer, 110  
   modem protocols  
     dial-up networking,  
     145-147  
   PPP, 147-151  
   protocol dysfunction, 244-245  
     ARP, 245, 251-252  
     configuration information  
     utilities, 245, 248-250  
     ping, 245-247  
 proxy service  
   content caching, 182  
   firewalls, 181-182  
   reverse proxy, 182-183  
 pseudo-headers, 101, 105  
 PSH, TCP data format, 95  
 PTR, 198  
 public key encryption, 398  
 public keys, 397  
 put command, 265  
 pwd command, 264

## Q-R

QoS (Quality of Service), 238-239  
 queries, 193  
 quit command, 266  
  
 RARP (Reverse Address  
 Resolution Protocol), 41, 63  
 RCP (Remote Procedure Call),  
 111, 115, 267-268  
   Berkeley r\* utilities, 280  
 RDF (Resource Description  
 Framework), 372  
 Realtime Control Protocol  
 (RTCP), 341  
 Realtime Streaming Protocol  
 (RTSP), 342  
 Realtime Transport Protocol  
 (RTP), 101, 341-343  
 reassociates, 160  
 receiver addresses, 159  
 recreational intruders, 376

recursive queries, 193  
 redirectors, 113, 268  
 refresh time, 199  
 registering domains, 195-196  
 registrars, 195  
 relay agents, 219  
 remote access  
   Berkeley r\* utilities, 278-279  
     rcp, 280  
     rexec, 281  
     rlogin, 279-280  
     rsh, 280-281  
     ruptime, 281  
     rwho, 281-282  
   network services, 113  
 RMON (Remote Monitoring),  
 289-291  
 screen sharing, 283-284  
 Secure Shell (SSH), 282-283  
 SNMP (Simple Network  
 Management Protocol),  
 284-285  
   SNMP address space,  
   285-287  
   SNMP commands,  
   287-289  
 telnet, 17, 116, 275-278  
 VPNs (Virtual Private  
 Networks), 404-406  
 remote copy, 267-268  
 Remote Monitoring. *See* RMON  
 Remote Procedure Calls (RPCs),  
 111, 269  
 Remote utilities, Application  
 layer, 116  
 requesters, 113, 268  
 Requests for Comment (RFCs),  
 17-18  
 resequencing TCP, 92  
 Resource Description Framework  
 (RDF), 372  
 resource records, 198  
 retrieving email, 328-330  
 retry time, 199  
 Reverse Address Resolution  
 Protocol (RARP), 41, 63  
 reverse lookup zone files, 200

reverse proxy, 182-183  
 Rexec, 116, 281  
 RFCs (Requests for Comment),  
 17-18  
 RIP (Routing Information  
 Protocol), 64, 136-138  
 rlogin, 279-280  
 rmdir command, 265  
 RMON (Remote Monitoring),  
 289-291  
   RMON 1, 289  
   RMON 2, 290  
 root access, 385  
 rootkit, 385  
 route, 17, 115, 256-257  
   route add, 256  
   route change, 257  
   route delete, 257  
   route print, 256  
 routers, 14  
   core routers, 135  
   default routers, 126  
   defined, 121-123  
   exterior routers, 135  
   higher stack levels, 139  
   interior routers, 136  
     OSPF (Open Shortest Path  
     First), 138  
     RIP (Routing Information  
     Protocol), 137-138  
 routing, 14-15  
   classless routing, 138-139  
   on complex networks,  
   134-136  
   dynamic routing, 125,  
   129-131  
   routing loops, 64  
   routing tables, 126  
   static routing, 125  
   in TCP/IP, 121-122  
     direct versus indirect  
     routing, 128-130  
     distance vector routing,  
     131-133  
     dynamic routing  
     algorithms, 130-131

## routing

- IP forwarding, 127-128
- link state routing, 132-134
- process of, 124-126
- routing tables, 126
- Routing header, 234-235**
- Routing Information Protocol (RIP), 64, 136-138**
- RPCs (Remote Procedure Calls), 111, 269**
- Rsh, 116, 280-281**
- RST, 95**
- RTCP (Realtime Control Protocol), 341**
- RTP (Realtime Transport Protocol), 101, 341-343**
- RTSP (Realtime Streaming Protocol), 342**
- rules, firewall, 180-181
- uptime, 281
- rwho, 281-282

## S

- scanning tools, 382
- schemes, 304
- screen sharing, 283-284
- script kiddies, 376
- scripting, server-side, 316-317
- SCTP (Stream Control Transmission Protocol), 101, 344**
- secondary name servers, 196
- Secure File Transfer Protocol (SFTP), 266**
- Secure Shell (SSH), 101, 282-283**
- security
  - 802.11 networks, 161-162
  - encryption, 391-392
    - algorithms and keys, 393-395
    - asymmetric encryption, 397-402
    - symmetric encryption, 395-397
  - Internet, 299
  - IPSec, 404
  - Kerberos, 406-409

- SSL (Secure Sockets Layer), 402-403
- TCP Transport layer, 92
- TLS (Transport Layer Security), 402-403
- segmenting, 27, 105, 417-418
- semantic web, 371-372
- Sendmail buffer overflow, 384
- Serial Line Internet Protocol (SLIP), 147**
- Server Message Block (SMB), 111, 269, 270**
- server response codes, 327
- server-side scripting, 316-317
- server-supplied IP addresses, 215-216
- servers, 300
  - caching only servers, 196
  - DHCP servers, 221-222
  - DNS servers, 197-200
  - primary name servers, 196
  - secondary name servers, 196
  - WINS servers, 204
- Service Location Protocol (SLP), 226**
- services, 262
- session hijacking, 383
- Session layer, 109
- SFTP (Secure File Transfer Protocol), 266**
- share point, 210
- Shortest Path Tree (SPT), 138**
- signatures, 421-422
- Simple Mail Transfer Protocol (SMTP), 324, 326-328**
- Simple Network Management Protocol. See SNMP**
- Simple Service Discovery Protocol (SSDP), 226**
- sliding window method, 98, 105
- SLIP (Serial Line Internet Protocol), 147**
- SLP (Service Location Protocol), 226**
- SMB (Server Message Block), 111, 269-270**
- SMTP (Simple Mail Transfer Protocol), 324-328**

- sniffers, 260**
- SNMP (Simple Network Management Protocol), 111, 284-285**
  - disadvantages of, 288
  - SNMP address space, 285-287
  - SNMP commands, 287-289
- SOA (Start of Authority), 198**
- SOAP, 357-358**
- social networking sites, 367
- sockets, 87-88, 114, 117
- source addresses, 159
- Source Port, 100**
- Source Quench, 64**
- spam, 334-336
- spambots, 336
- special IP addresses, 61
- SPREAD, 135**
- SPT (Shortest Path Tree), 138**
- SSDP (Simple Service Discovery Protocol), 226**
- SSH (Secure Shell), 101, 282-283**
- SSL (Secure Sockets Layer), 402-403**
  - Alert Protocol, 403
  - Change Cipher Spec Protocol, 403
  - Handshake Protocol, 403
- stack levels, 139
- stateful firewalls, 177
- static IP addressing, 216
- static routing, 125
- status codes (HTTP), 314-315
- status command, 265
- Stream Control Transmission Protocol (SCTP), 101, 344**
- stream-oriented processing, 92
- streaming
  - podcasting, 346-347
  - problems with, 339-340
  - RTP (Realtime Transport Protocol), 341-343
  - transport options, 343-346
  - Voice over IP (VoIP), 347-348
- subdomains, 193**
- subnet IDs, 75-77**

**subnet masks, 72**

- converting to dotted decimal notation, 73-75
- dotted notation to binary patterns, 78-79
- expressing in dotted decimal notation, 77

**subnets, 13, 57, 69-70****subnetting, 57, 69-73****supernet masks, 82****.SWF, 345****switches, 169-171****symmetric encryption, 395-397****SYN, 95****T****tags (HTML), 309-312****TCP (Transmission Control Protocol), 28, 32, 85**

- connections, 96-97
  - closing, 99
  - establishing, 97-98
  - flow control, 98
- data format, 94-96
- ports, 88-89
- transport layer, 91-93
  - TCP connections, 96-97
  - TCP connections, closing, 99
  - TCP connections, establishing, 97-98
  - TCP connections, flow control, 98
  - TCP data format, 94-96

**TCP/IP**

- application layer, 108-109
- development of
  - Internet, 10-11
  - LANs, 11
- dial-up connections, 146
- domain name resolution (Hypothetical Inc.), 419-420
- dynamic addresses (Hypothetical Inc.), 418

**features of, 12**

- application support, 16-17
- error control, 15
- flow control, 15
- logical addressing, 12-13
- name resolution, 15
- routing, 14-15

**firewalls (Hypothetical Inc.), 420****getting started with, 415-417****implementation, 9****model, 22****networking system, 28-30****OSI model and, 24-26****protocol stacks, 26-28****protocol system, 22-23****Application layer, 24****Internet layer, 23****Network Access layer, 23-24****Transport layer, 23****routing, 121-122****direct versus indirect****routing, 128-130****distance vector routing, 131-133****dynamic routing****algorithms, 130-131****IP forwarding, 127-128****link state routing, 132-134****process of, 124-126****routing tables, 126****segmenting (Hypothetical Inc.), 417-418****signatures and VPNs (Hypothetical Inc.), 421-422****standard, 9****utilities, 115-116****web services (Hypothetical Inc.), 421****telnet, 17, 116, 275-278****testing NetBIOS name resolution, 210-211****TFTP (Trivial File Transfer Protocol), 112, 115, 267****three-way handshake, 97****Time Exceeded, 64****time fields (DHCP), 220****TLDs (top level domains), 191, 195****TLS (Transport Layer Security), 402-403****top level domains (TLDs), 191, 195****traceroute, 17, 115, 254-256****tracert, 254, 256****transmitter addresses, 159****Transport Control Protocol. See TCP****Transport layer, 83-85****connection-oriented protocols, 85-86****connectionless protocols, 85-86****firewalls, 101-102****multiplexing/demultiplexing, 90-91****ports, 87-88****TCP 88-89****UDP, 89****sockets, 87-88****TCP, 91-93****connections, 96-97****connections, closing, 99****connections, establishing, 97-98****connections, flow control, 98****data format, 94-96****TCP/IP protocol system, 23****UDP, 91-92, 99-101****Transport Layer Security (TLS), 402-403****Transport mode, 404****transport options, 343-346****trap messages, 288****traps, 288****Trivial File Transfer Protocol (TFTP), 112, 267****trojan horses, 379-380****troubleshooting connectivity problems, 261****trusted access, 279**

## trusted hosts

trusted hosts, 279  
 trusted users, 279  
 TTL (time-to-live), 199  
 Tunnel mode, 404  
 type command, 265

**U-V**

UDP (User Datagram Protocol), 28, 85  
   ports, 89  
   Transport layer, 91-92, 99-101

Uniform Resource Identifiers (URIs), 302-303

Uniform Resource Locator (URL), 301

URG, 95

URIs (Uniform Resource Identifiers), 302-303

URLs (Uniform Resource Locators), 301, 307

user command, 264

User Datagram Protocol.  
 See UDP

utilities  
   configuration information  
     utilities, 248-250  
   connectivity problems, 261  
   DNS utilities, 200  
     NSLookup, 201-203  
     Ping, 201  
   nbtstat, 259-260  
   netstat, 257-258  
   nslookup, 253  
   packet sniffers, 260-261  
   ping. See ping  
   route, 256-257  
   TCP/IP utilities, 17, 115-116  
   traceroute, 254-256

vendor's implementation of TCP/IP, 8

video file formats, 344

viruses, 333

VoIP (Voice over IP), 347-348

VPNs (Virtual Private Networks), 404-406, 421-422

**W**

W3C (World Wide Web Consortium), 372

WANs (Wide Area Networks), 154-155, 405

WAP (Wireless Application Protocol), 162-164

WAP Datagram Transport Protocol (WDP), 163

WAP Session Protocol (WSP), 163

WAP Transaction Layer Security (WTLS), 163

WAP Transaction Protocol (WTP), 163

WDP (WAP Datagram Transport Protocol), 163

Web 2.0, 363-364  
   blogs, 364-365  
   social networking sites, 367  
   wikis, 366-367

web services, 353-355  
   architecture, 354  
   Hypothetical Inc., 421  
   network services, 113  
   stacks, 358-359

Web Services Description Language (WSDL), 358

weblogs, 364-365

webmail, 333-334

WECA (Wireless Ethernet Compatibility Alliance), 161

well-known ports, 88

WEP (Wired Equivalent Privacy), 161-162

WEP2, 162

whitelists, 336

Whois, 116

Wi-Fi (Wireless Fidelity), 161

Wide Area Networks (WANs), 154-155, 405

Wikipedia, 366

wikis, 366-367

Window field, 98

Windows Internet Name Service.  
 See WINS

Windows Vista, configuring as DHCP client, 220-221

WINS (Windows Internet Name Service)  
   name resolution, 207-210  
   servers, 204

Wired Equivalent Privacy (WEP), 161

Wireless Application Protocol (WAP), 162-164

Wireless Ethernet Compatibility Alliance (WECA), 161

wireless networking, 155-156  
   802.11 networks, 156-157  
     independent and infrastructure networks, 157-160  
     security, 161-162  
   Bluetooth, 165-167  
   Mobile IP, 164-165  
   WAP (Wireless Application Protocol), 162-164

World Wide Web, 305-308

World Wide Web Consortium (W3C), 372

WSDL (Web Services Description Language), 358

WSP (WAP Session Protocol), 163

WTLS (WAP Transaction Layer Security), 163

WTP (WAP Transaction Protocol), 163

WYSIWYG (What You See Is What You Get), 364

**X-Y-Z**

X.509 certificate process, 402

XHTML, 367

XML (eXtensible Markup Language), 318, 355-356

XMPP (eXtensible Messaging and Presence Protocol), 371

zero configuration, 224-227

Zeroconf, 225

zone files, configuring DNS servers, 198-200

zone transfers, 196

zones, 197