Second Edition

Data Networks

DIMITRI BERTSEKAS

Massachusetts Institute of Technology

ROBERT GALLAGER

Massachusetts Institute of Technology



PRENTICE HALL, Englewood Cliffs, New Jersey 07632

Samsung Exhibit 1015, Page 1 of 573

Library of Congress Cataloging-in-Publication Data

Bertsekas, Dimitri P.
Data networks / Dimitri Bertsekas, Robert Gallager. -- 2nd ed.
p. cm.
Includes bibliographical references and index.
ISBN 0-13-200916-1
1. Data transmission systems. I. Gallager, Robert G. II. Title.
TK5105.B478 1992
004.6--dc20 91-35561
CIP

Acquisitions editor: Pete Janzow
Production editor: Bayani Mendoza de Leon
Copy editor: Zeiders & Associates
Cover designer: Butler/Udell
Prepress buyer: Linda Behrens
Manufacturing buyer: Dave Dickey
Editorial assistant: Phyllis Morgan



©1992, 1987 by Prentice-Hall, Inc.
A Simon & Schuster Company
Englewood Cliffs, New Jersey 07632

All rights reserved. No part of this book may be
reproduced, in any form or by any means,
without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-200916-1

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S. A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Contents

PREFACE xv

1 INTRODUCTION AND LAYERED NETWORK ARCHITECTURE 1

1.1 Historical Overview **1**

1.1.1 Technological and Economic Background, 5
1.1.2 Communication Technology, 6
1.1.3 Applications of Data Networks, 7

1.2 Messages and Switching **9**

1.2.1 Messages and Packets, 9
1.2.2 Sessions, 11
1.2.3 Circuit Switching and Store-and-Forward Switching, 14

1.3 Layering **17**

1.3.1 The Physical Layer, 20
1.3.2 The Data Link Control Layer, 23

The MAC sublayer, 24

1.3.3 The Network Layer, 25

The Internet sublayer, 28

1.3.4 The Transport Layer, 29
1.3.5 The Session Layer, 30

Samsung Exhibit 1015, Page 3 of 573

1.3.6 The Presentation Layer, 31
1.3.7 The Application Layer, 31

1.4 A Simple Distributed Algorithm Problem **32**

Notes and Suggested Reading **35**

Problems **35**

2 POINT-TO-POINT PROTOCOLS AND LINKS 37

2.1 Introduction **37**

2.2 The Physical Layer: Channels and Modems **40**

2.2.1 Filtering, 41
2.2.2 Frequency Response, 43
2.2.3 The Sampling Theorem, 46
2.2.4 Bandpass Channels, 47
2.2.5 Modulation, 48
2.2.6 Frequency- and Time-Division Multiplexing, 52
2.2.7 Other Channel Impairments, 53
2.2.8 Digital Channels, 53

ISDN, 54

2.2.9 Propagation Media for Physical Channels, 56

2.3 Error Detection **57**

2.3.1 Single Parity Checks, 58
2.3.2 Horizontal and Vertical Parity Checks, 58
2.3.3 Parity Check Codes, 59
2.3.4 Cyclic Redundancy Checks, 61

2.4 ARQ: Retransmission Strategies **64**

2.4.1 Stop-and-Wait ARQ, 66

Correctness of stop and wait, 69

2.4.2 Go Back n ARQ, 72

Rules followed by transmitter and receiver in go back n, *74*
Correctness of go back n, *76*
Go back n *with modulus* m $>$ n, *78*
Efficiency of go back n *implementations, 80*

2.4.3 Selective Repeat ARQ, 81
2.4.4 ARPANET ARQ, 84

2.5 Framing **86**

2.5.1 Character-Based Framing, 86
2.5.2 Bit-Oriented Framing: Flags, 88
2.5.3 Length Fields, 90
2.5.4 Framing with Errors, 92
2.5.5 Maximum Frame Size, 93

Variable frame length, 93
Fixed frame length, 97

2.6 Standard DLCs **97**

2.7 Initialization and Disconnect for ARQ Protocols **103**

2.7.1 Initialization in the Presence of Link Failures, 103
2.7.2 Master–Slave Protocol for Link Initialization, 104
2.7.3 A Balanced Protocol for Link Initialization, 107
2.7.4 Link Initialization in the Presence of Node Failures, 109

2.8 Point-to-Point Protocols at the Network Layer **110**

2.8.1 Session Identification and Addressing, 111

Session identification in TYMNET, 112
Session identification in the Codex networks, 113

2.8.2 Packet Numbering, Window Flow Control, and Error Recovery, 114

Error recovery, 115
Flow control, 116
Error recovery at the transport layer versus the network layer, 117

2.8.3 The X.25 Network Layer Standard, 118
2.8.4 The Internet Protocol, 120

2.9 The Transport Layer **123**

2.9.1 Transport Layer Standards, 123
2.9.2 Addressing and Multiplexing in TCP, 124
2.9.3 Error Recovery in TCP, 125
2.9.4 Flow Control in TCP/IP, 127
2.9.5 TP Class 4, 128

2.10 Broadband ISDN and the Asynchronous Transfer Mode **128**

2.10.1 Asynchronous Transfer Mode (ATM), 132
2.10.2 The Adaptation Layer, 135

Class 3 (connection-oriented) traffic, 136
Class 4 (connectionless) traffic, 137
Class 1 and 2 traffic, 137

2.10.3 Congestion, 138

Summary **139**

Notes, Sources, and Suggested Reading **140**

Problems **141**

3 DELAY MODELS IN DATA NETWORKS **149**

3.1 Introduction **149**

3.1.1 Multiplexing of Traffic on a Communication Link, 150

3.2 Queueing Models: Little's Theorem **152**

3.2.1 Little's Theorem, 152
3.2.2 Probabilistic Form of Little's Theorem, 154
3.2.3 Applications of Little's Theorem, 157

3.3 The $M/M/1$ Queueing System **162**

3.3.1 Main Results, 164

Arrival statistics—the Poisson process, 164
Service statistics, 165
Markov chain formulation, 166
Derivation of the stationary distribution, 167

3.3.2 Occupancy Distribution upon Arrival, 171
3.3.3 Occupancy Distribution upon Departure, 173

3.4 The $M/M/m$, $M/M/\infty$, $M/M/m/m$, and Other Markov Systems **173**

3.4.1 $M/M/m$: The m-Server Case, 174
3.4.2 $M/M/\infty$: The Infinite-Server Case, 177
3.4.3 $M/M/m/m$: The m-Server Loss System, 178
3.4.4 Multidimensional Markov Chains: Applications in Circuit Switching, 180

Truncation of independent single-class systems, 182
Blocking probabilities for circuit switching systems, 185

3.5 The $M/G/1$ System **186**

3.5.1 $M/G/1$ Queues with Vacations, 192
3.5.2 Reservations and Polling, 195

Single-user system, 196
Multi-user system, 198
Limited service systems, 201

3.5.3 Priority Queueing, 203

Nonpreemptive priority, 203

Preemptive resume priority, 205

3.5.4 An Upper Bound for the $G/G/1$ System, 206

3.6 Networks of Transmission Lines **209**

3.6.1 The Kleinrock Independence Approximation, 211

3.7 Time Reversibility—Burke's Theorem **214**

3.8 Networks of Queues—Jackson's Theorem **221**

Heuristic explanation of Jackson's Theorem, 227

3.8.1 Extensions of Jackson's Theorem, 229

State-dependent service rates, 229
Multiple classes of customers, 230

3.8.2 Closed Queueing Networks, 233
3.8.3 Computational Aspects—Mean Value Analysis, 238

Summary **240**

Notes, Sources, and Suggested Reading **241**

Problems **242**

Appendix A: Review of Markov Chain Theory **259**

3A.1 Discrete-Time Markov Chains, 259
3A.2 Detailed Balance Equations, 261
3A.3 Partial Balance Equations, 262
3A.4 Continuous-Time Markov Chains, 262
3A.5 Drift and Stability, 264

Appendix B: Summary of Results **265**

4 MULTIACCESS COMMUNICATION 271

4.1 Introduction **271**

4.1.1 Satellite Channels, 273
4.1.2 Multidrop Telephone Lines, 274
4.1.3 Multitapped Bus, 274
4.1.4 Packet Radio Networks, 275

4.2 Slotted Multiaccess and the Aloha System **275**

4.2.1 Idealized Slotted Multiaccess Model, 275

Discussion of assumptions, 276

Samsung Exhibit 1015, Page 7 of 573

4.2.2 Slotted Aloha, 277
4.2.3 Stabilized Slotted Aloha, 282

Stability and maximum throughput, 282
Pseudo-Bayesian algorithm, 283
Approximate delay analysis, 284
Binary exponential backoff, 286

4.2.4 Unslotted Aloha, 287

4.3 Splitting Algorithms **289**

4.3.1 Tree Algorithms, 290

Improvements to the tree algorithm, 292
Variants of the tree algorithm, 293

4.3.2 First-Come First-Serve Splitting Algorithms, 293

Analysis of FCFS splitting algorithm, 297
Improvements in the FCFS splitting algorithm, 301
Practical details, 302
Last-come first-serve (LCFS) splitting algorithm, 302
Delayed feedback, 303
Round-robin splitting, 304

4.4 Carrier Sensing **304**

4.4.1 CSMA Slotted Aloha, 305
4.4.2 Pseudo-Bayesian Stabilization for CSMA Aloha, 307
4.4.3 CSMA Unslotted Aloha, 309
4.4.4 FCFS Splitting Algorithm for CSMA, 310

4.5 Multiaccess Reservations **312**

4.5.1 Satellite Reservation Systems, 313
4.5.2 Local Area Networks: CSMA/CD and Ethernet, 317

Slotted CSMA/CD, 317
Unslotted CSMA/CD, 318
The IEEE 802 standards, 320

4.5.3 Local Area Networks: Token Rings, 320

IEEE 802.5 token ring standard, 323
Expected delay for token rings, 324
FDDI, 326
Slotted rings and register insertion rings, 330

4.5.4 Local Area Networks: Token Buses and Polling, 331

IEEE 802.4 token bus standard, 332
Implicit tokens: CSMA/CA, 333

4.5.5 High-Speed Local Area Networks, 333

Distributed queue dual bus (IEEE 802.6), 335

Expressnet, 339
Homenets, 341

4.5.6 Generalized Polling and Splitting Algorithms, 342

4.6 Packet Radio Networks **344**

4.6.1 TDM for Packet Radio Nets, 346
4.6.2 Collision Resolution for Packet Radio Nets, 347
4.6.3 Transmission Radii for Packet Radio, 349
4.6.4 Carrier Sensing and Busy Tones, 350

Summary **351**

Notes, Sources, and Suggested Reading **352**

Problems **353**

5 ROUTING IN DATA NETWORKS 363

5.1 Introduction **363**

5.1.1 Main Issues in Routing, 365
5.1.2 Wide-Area Network Routing: An Overview, 367

Flooding and broadcasting, 368
Shortest path routing, 370
Optimal routing, 372
Hot potato (deflection) routing schemes, 372
Cut-through routing, 373
ARPANET: An example of datagram routing, 374
TYMNET: An example of virtual circuit routing, 376
Routing in SNA, 378
Routing in circuit switching networks, 379

5.1.3 Interconnected Network Routing: An Overview, 379

Bridged local area networks, 382
Spanning tree routing in bridged local area networks, 383
Source routing in bridged local area networks, 385

5.2 Network Algorithms and Shortest Path Routing **387**

5.2.1 Undirected Graphs, 387
5.2.2 Minimum Weight Spanning Trees, 390
5.2.3 Shortest Path Algorithms, 393

The Bellman-Ford algorithm, 396
Bellman's equation and shortest path construction, 399
Dijkstra's algorithm, 401
The Floyd-Warshall algorithm, 403

5.2.4 Distributed Asynchronous Bellman–Ford Algorithm, 404

5.2.5 Stability of Adaptive Shortest Path Routing Algorithms, 410

Stability issues in datagram networks, 410
Stability issues in virtual circuit networks, 414

5.3 Broadcasting Routing Information: Coping with Link Failures **418**

5.3.1 Flooding: The ARPANET Algorithm, 420
5.3.2 Flooding without Periodic Updates, 422
5.3.3 Broadcast without Sequence Numbers, 425

5.4 Flow Models, Optimal Routing, and Topological Design **433**

5.4.1 Overview of Topological Design Problems, 437
5.4.2 Subnet Design Problem, 439

Capacity assignment problem, 439
Heuristic methods for capacity assignment, 442
Network reliability issues, 445
Spanning tree topology design, 447

5.4.3 Local Access Network Design Problem, 448

5.5 Characterization of Optimal Routing **451**

5.6 Feasible Direction Methods for Optimal Routing **455**

5.6.1 The Frank–Wolfe (Flow Deviation) Method, 458

5.7 Projection Methods for Optimal Routing **464**

5.7.1 Unconstrained Nonlinear Optimization, 465
5.7.2 Nonlinear Optimization over the Positive Orthant, 467
5.7.3 Application to Optimal Routing, 468

5.8 Routing in the Codex Network **476**

Summary **477**

Notes, Sources, and Suggested Reading **478**

Problems **479**

6 FLOW CONTROL **493**

6.1 Introduction **493**

6.1.1 Means of Flow Control, 494
6.1.2 Main Objectives of Flow Control, 496

Limiting delay and buffer overflow, 496
Fairness, 498

6.2 Window Flow Control **500**

6.2.1 End-to-End Windows, 501

Limitations of end-to-end windows, 502

6.2.2 Node-by-Node Windows for Virtual Circuits, 506
6.2.3 The Isarithmic Method, 508
6.2.4 Window Flow Control at Higher Layers, 508
6.2.5 Dynamic Window Size Adjustment, 510

6.3 Rate Control Schemes **510**

Queueing analysis of the leaky bucket scheme, 513

6.4 Overview of Flow Control in Practice **515**

Flow control in the ARPANET, 515
Flow control in the TYMNET, 517
Flow control in SNA, 517
Flow control in a Codex network, 518
Flow control in the PARIS network, 518
Flow control in X.25, 519

6.5 Rate Adjustment Algorithms **519**

6.5.1 Combined Optimal Routing and Flow Control, 519
6.5.2 Max-Min Flow Control, 524

Summary **530**

Notes, Sources, and Suggested Reading **530**

Problems **531**

REFERENCES **537**

INDEX **552**

Preface

In the five years since the first edition of *Data Networks* appeared, the networking field has changed in a number of important ways. Perhaps the most fundamental has been the rapid development of optical fiber technology. This has created almost limitless opportunities for new digital networks of greatly enhanced capabilities. In the near term, the link capacities available for data networks, both wide area and local, are increasing by many orders of magnitude. In the longer term, public broadband integrated service networks that provide integrated data, voice, and video on a universal scale are now technologically feasible. These networks of the future appear at first to have almost nothing in common with the data networks of the last 20 years, but in fact, many of the underlying principles are the same. This edition is designed both to provide a fundamental understanding of these common principles and to provide insight into some of the new principles that are evolving for future networks.

Our approach to helping the reader understand the basic principles of networking is to provide a balance between the description of existing networks and the development of analytical tools. The descriptive material is used to illustrate the underlying concepts, and the analytical material is used to generate a deeper and more precise understanding of the concepts. Although the analytical material can be used to analyze the performance of various networks, we believe that its more important use is in sharpening one's conceptual and intuitive understanding of the field; that is, analysis should precede design rather than follow it.

The book is designed to be used at a number of levels, varying from a senior undergraduate elective, to a first year graduate course, to a more advanced graduate course, to a reference work for designers and researchers in the field. The material has been tested in a number of graduate courses at M.I.T. and in a number of short courses at

Samsung Exhibit 1015, Page 12 of 573

varying levels. The book assumes some background in elementary probability and some background in either electrical engineering or computer science, but aside from this, the material is self-contained.

Throughout the book, major concepts and principles are first explained in a simple non-mathematical way. This is followed by careful descriptions of modelling issues and then by mathematical analysis. Finally, the insights to be gained from the analysis are explained and examples are given to clarify the more subtle issues. Figures are liberally used throughout to illustrate the ideas. For lower-level courses, the analysis can be glossed over; this allows the beginning and intermediate-level student to grasp the basic ideas, while enabling the more advanced student to acquire deeper understanding and the ability to do research in the field.

Chapter 1 provides a broad introduction to the subject and also develops the layering concept. This layering allows the various issues of data networks to be developed in a largely independent fashion, thus making it possible to read the subsequent chapters in any desired depth (including omission) without seriously hindering the ability to understand other chapters.

Chapter 2 covers the two lowest layers of the above layering and also discusses a number of closely related aspects of the higher layers. The treatment of the lowest, or physical, layer provides a brief overview of how binary digits are transmitted over physical communication media. The effort here is to provide just enough material so that the student can relate the abstraction of digital transmission to physical phenomena. The next layer, data link control, allows packets to be transmitted reliably over communication links. This provides an introduction into the distributed algorithms, or protocols, that must be used at the ends of the link to provide the desired reliability. These protocols are less important in modern high speed networks than in older networks, but the concepts are used repeatedly at many layers in all kinds of networks. The remainder of the chapter focuses on other point to point protocols that allow the end points of a link, or the end points of a network session, to cooperate in providing some required service.

Chapter 3 develops the queueing theory used for performance analysis of multiaccess schemes (Chapter 4) and, to a lesser extent, routing algorithms (Chapter 5). Less analytical courses will probably omit most of this chapter, simply adopting the results on faith. Little's theorem and the Poisson process should be covered however, since they are simple and greatly enhance understanding of the subsequent chapters. This chapter is rich in results, often developed in a far simpler way than found in the queueing literature. This simplicity is achieved by considering only steady-state behavior and by sometimes sacrificing rigor for clarity and insight. Mathematically sophisticated readers will be able to supply the extra details for rigor by themselves, while for most readers the extra details would obscure the line of argument.

Chapter 4 develops the topic of multiaccess communication, including local area networks, metropolitan area networks, satellite networks, and radio networks. Less theoretical courses will probably skip the last half of section 4.2, all of section 4.3, and most of section 4.4, getting quickly to local area networks in section 4.5. Conceptually, one gains a great deal of insight into the nature of distributed algorithms in this chapter.

Samsung Exhibit 1015, Page 13 of 573

Preface

Chapter 5 develops the subject of routing. The material is graduated in order of increasing difficulty and depth, so readers can go as far as they are comfortable. Along with routing itself, which is treated in greater depth than elsewhere in the literature, further insights are gained into distributed algorithms. There is also a treatment of topological design and a section on recovery from link failures.

Chapter 6 deals with flow control (or congestion control as it is sometimes called). The first three sections are primarily descriptive, describing first the objectives and the problems in achieving these objectives, and then two general approaches, window flow control, and rate control. The fourth section describes the ways that flow control is handled in several existing networks. The last section is more advanced and analytical, describing various algorithms to select session rates in rate control schemes.

A topic that is not treated in any depth in the book is that of higher-layer protocols, namely the various processes required in the computers and devices using the network to communicate meaningfully with each other given the capability of reliable transport of packets through the network provided by the lower layers. This topic is different in nature than the other topics covered and would have doubled the size of the book if treated in depth.

We apologize in advance for the amount of jargon and acronyms in the book. We felt it was necessary to include at least the most commonly used acronyms in the field, both to allow readers to converse with other workers in the field and also for the reference value of being able to find out what these acronyms mean.

An extensive set of problems are given at the end of each chapter except the first. They range from simple exercises to gain familiarity with the basic concepts and techniques to advanced problems extending the results in the text. Solutions of the problems are given in a manual available to instructors from Prentice-Hall.

Each chapter also contains a brief section of sources and suggestions for further reading. Again, we apologize in advance to the many authors whose contributions have not been mentioned. The literature in the data network field is vast, and we limited ourselves to references that we found most useful, or that contain material supplementing the text.

The stimulating teaching and research environment at M.I.T. has been an ideal setting for the development of this book. In particular we are indebted to the many students who have used this material in courses. Their comments have helped greatly in clarifying the topics. We are equally indebted to the many colleagues and advanced graduate students who have provided detailed critiques of the various chapters. Special thanks go to our colleague Pierre Humblet whose advice, knowledge, and deep insight have been invaluable. In addition, Erdal Arikan, David Castanon, Robert Cooper, Tony Ephremides, Eli Gafni, Marianne Gardner, Inder Gopal, Paul Green, Ellen Hahne, Bruce Hajek, Michael Hluchyi, Robert Kennedy, John Spinelli, and John Tsitsiklis have all been very helpful. We are also grateful to Nancy Young for typing the many revisions. Our editors at Prentice-Hall have also been very helpful and cooperative in producing the final text under a very tight schedule. Finally we wish to acknowledge the research sup-

Samsung Exhibit 1015, Page 14 of 573

port of DARPA under grant ONR-N00014-84-K-0357, NSF under grants ECS-8310698, ECS-8217668, 8802991-NCR, and DDM-8903385, and ARO under grants DAAG 29-84-K-000 and DAAL03-86-K-0171.

Dimitri Bertsekas

Robert Gallager



Samsung Exhibit 1015, Page 15 of 573

Second Edition

Data Networks

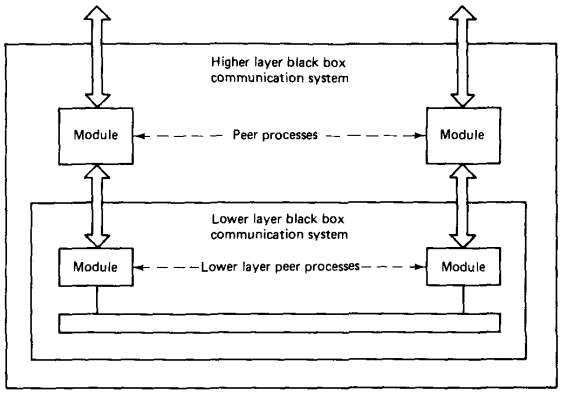
DIMITRI BERTSEKAS
Massachusetts Institute of Technology

ROBERT GALLAGER
Massachusetts Institute of Technology



PRENTICE HALL, Englewood Cliffs, New Jersey 07632

1



Introduction and Layered Network Architecture

1.1 HISTORICAL OVERVIEW

Primitive forms of data networks have a long history, including the smoke signals used by primitive societies, and certainly including nineteenth-century telegraphy. The messages in these systems were first manually encoded into strings of essentially binary symbols, and then manually transmitted and received. Where necessary, the messages were manually relayed at intermediate points.

A major development, in the early 1950s, was the use of communication links to connect central computers to remote terminals and other peripheral devices, such as printers and remote job entry points (RJEs) (see Fig. 1.1). The number of such peripheral devices expanded rapidly in the 1960s with the development of time-shared computer systems and with the increasing power of central computers. With the proliferation of remote peripheral devices, it became uneconomical to provide a separate long-distance communication link to each peripheral. Remote multiplexers or concentrators were developed to collect all the traffic from a set of peripherals in the same area and to send it on a single link to the central processor. Finally, to free the central processor from handling all this communication, special processors called *front ends* were developed to

Samsung Exhibit 1015, Page 17 of 573

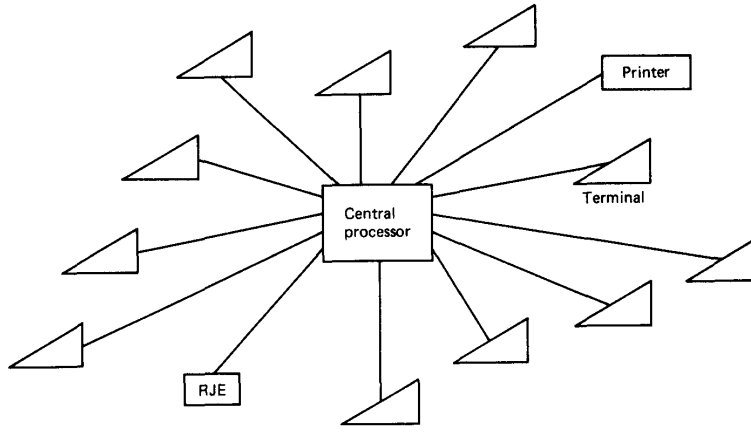


Figure 1.1 Network with one central processor and a separate communication link to each device.

control the communication to and from all the peripherals. This led to the more complex structure shown in Fig. 1.2. The communication is automated in such systems, in contrast to telegraphy, for example, but the control of the communication is centrally exercised at the computer. While it is perfectly appropriate and widely accepted to refer to such a system as a data network or computer communication network, it is simpler to view it as a computer with remote peripherals. Many of the interesting problems associated with data networks, such as the distributed control of the system, the relaying of messages over multiple communication links, and the sharing of communication links between many users and processes, do not arise in these centralized systems.

The ARPANET and TYMNET, introduced around 1970, were the first large-scale, general-purpose data networks connecting geographically distributed computer systems, users, and peripherals. Figure 1.3 shows such networks. Inside the "subnet" are a set of nodes, various pairs of which are connected by communication links. Outside the subnet are the various computers, data bases, terminals, and so on, that are connected via the subnet. Messages originate at these external devices, pass into the subnet, pass from node to node on the communication links, and finally pass out to the external recipient. The nodes of the subnet, usually computers in their own right, serve primarily to route the messages through the subnet. These nodes are sometimes called IMPs (interface message processors) and sometimes called switches. In some networks (*e.g.*, DECNET), nodes in the subnet might be physically implemented within the external computers using the network. It is helpful, however, to view the subnet nodes as being logically distinct from the external computers.

It is important to observe that in Figs. 1.1 and 1.2 the computer system is the center of the network, whereas in Fig. 1.3 the subnet (*i.e.*, the communication part of the network) is central. Keeping this picture of external devices around a communication

Samsung Exhibit 1015, Page 18 of 573

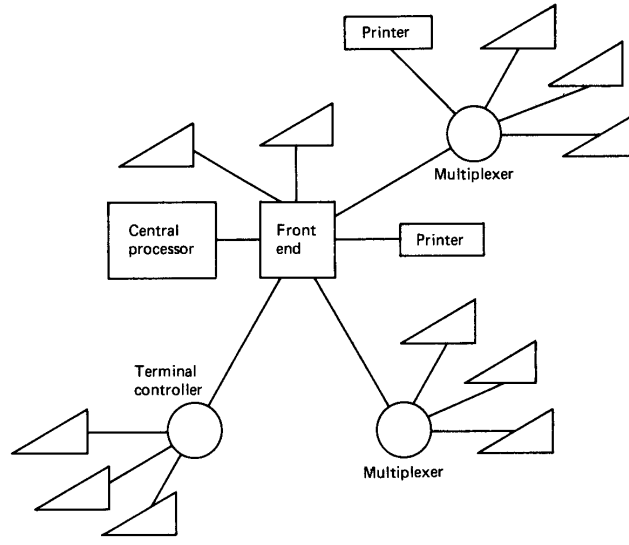


Figure 1.2 Network with one central processor but with shared communication links to devices.

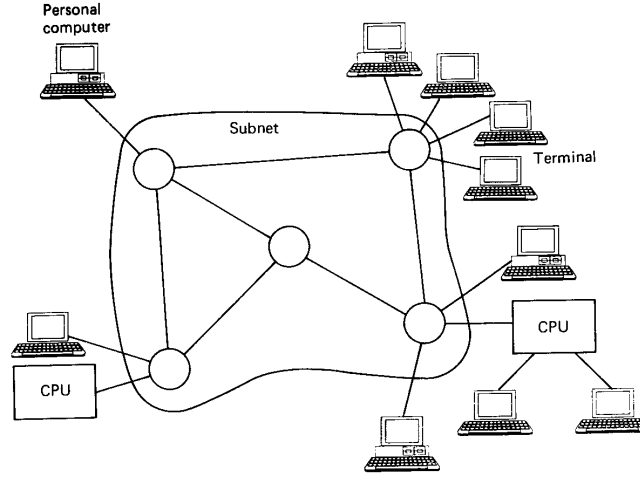


Figure 1.3 General network with a subnet of communication links and nodes. External devices are connected to the subnet via links to the subnet nodes.

Samsung Exhibit 1015, Page 19 of 573

subnet in mind will make it easier both to understand network layering later in this chapter and to understand the issues of distributed network control throughout the book.

The subnet shown in Fig. 1.3 contains a somewhat arbitrary placement of links between the subnet nodes. This arbitrary placement (or arbitrary topology as it is often called) is typical of wide area networks (*i.e.*, networks covering more than a metropolitan area). Local area networks (*i.e.*, networks covering on the order of a square kilometer or less) usually have a much more restricted topology, with the nodes typically distributed on a bus, a ring, or a star.

Since 1970 there has been an explosive growth in the number of wide area and local area networks. Many examples of these networks are discussed later, including as wide area networks, the seminal ARPANET and TYMNET, and as local area networks, Ethernets and token rings. For the moment, however, Fig. 1.3 provides a generic model for data networks.

With the multiplicity of different data networks in existence in the 1980s, more and more networks have been connected via gateways and bridges so as to allow users of one network to send data to users of other networks (see Fig. 1.4). At a fundamental level, one can regard such a network of networks as simply another network, as in Fig. 1.3, with each gateway, bridge, and subnet node of each constituent network being a subnet node of the overall network. From a more practical viewpoint, a network of networks is much more complex than a single network. The problem is that each constituent subnet has its own conventions and control algorithms (*i.e.*, protocols) for handling data, and the gateways and bridges must deal with this inhomogeneity. We discuss this problem later after developing some understanding of the functioning of individual subnets.

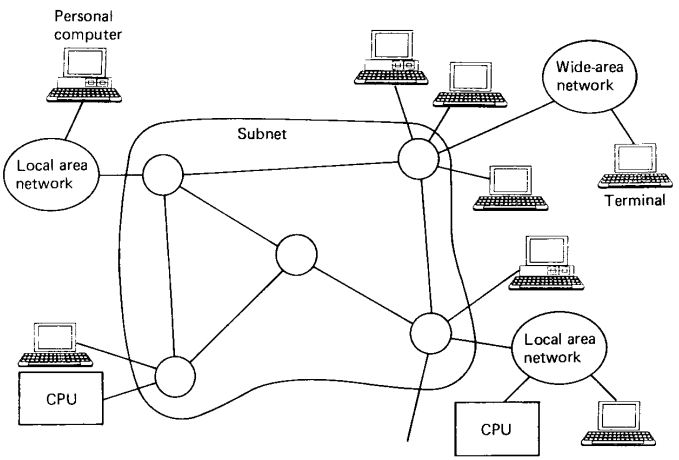


Figure 1.4 Network of interconnected networks. Individual wide area networks (WANs) and local networks (LANs) are connected via bridges and gateways.

In the future, it is likely that data networks, the voice network, and perhaps cable TV networks will be far more integrated than they are today. Data can be sent over the voice network today, as explained more fully in Section 2.2, and many of the links in data networks are leased from the voice network. Similarly, voice can be sent over data networks. What is envisioned for the future, however, is a single integrated network, called an integrated services digital network (ISDN), as ubiquitous as the present voice network. In this vision, offices and homes will each have an access point into the ISDN that will handle voice, current data applications, and new applications, all with far greater convenience and less expense than is currently possible. ISDN is currently available in some places, but it is not yet very convenient or inexpensive. Another possibility for the future is called broadband ISDN. Here the links will carry far greater data rates than ISDN and the network will carry video as well as voice and data. We discuss the pros and cons of this in Section 2.10.

1.1.1 Technological and Economic Background

Before data networks are examined, a brief overview will be given, first, of the technological and economic factors that have led to network development, and second, of the applications that require networks. The major driving force in the rapid advances in computers, communication, and data networks has been solid-state technology and in particular the development of very large scale integration (VLSI). In computers, this has led to faster, less expensive processors; faster, larger, less expensive primary memory; and faster, larger, less expensive bulk storage. The result has been the lowering of the cost of computation by roughly a factor of 2 every two years. This has led to a rapid increase in the number of cost effective applications for computers.

On the other hand, with the development of more and more powerful microprocessor chips, there has been a shift in cost effectiveness from large time-shared computer facilities to small but increasingly powerful personal computers and workstations. Thus, the primary growth in computation is in the number of computer systems rather than in the increasing power of a small number of very large computer systems.

This evolution toward many small but powerful computers has had several effects on data networks. First, since individual organizations use many computers, there is a need for them to share each other's data, thus leading to the network structures of Figs. 1.3 and 1.4. (If, instead, the evolution had been toward a small number of ever more powerful computer systems, the structure of Fig. 1.2 would still predominate.) Second, since subnet nodes are small computers, the cost of a given amount of processing within a subnet has been rapidly decreasing. Third, as workstations become more powerful, they deal with larger blocks of data (*e.g.*, high-resolution graphics) and the data rates of present-day wide area data networks are insufficient to transmit such blocks with acceptable delay.

The discussion of computational costs above neglects the cost of software. While the art of software design has been improving, the improvement is partly counterbalanced by the increasing cost of good software engineers. When software can be replicated, however, the cost per unit goes down inversely with the number of replicas. Thus,

Samsung Exhibit 1015, Page 21 of 573

even though software is a major cost of a new computer system, the increasing market decreases its unit cost. Each advance in solid-state technology decreases cost and increases the performance of computer systems; this leads to an increase in market, thus generating decreased unit software costs, leading, in a feedback loop, to further increases in market. Each new application, however, requires new specialized software which is initially expensive (until a market develops) and which requires a user learning curve. Thus, it is difficult to forecast the details of the growth of both the computer market and the data network market.

1.1.2 Communication Technology

The communication links for wide area networks are usually leased from the facilities of the voice telephone network. In Section 2.2 we explain how these physical links are used to transmit a fixed-rate stream of binary digits. The rate at which a link transmits binary digits is usually referred to as the data rate, capacity, or speed of the link, and these data rates come in standard sizes. Early networks typically used link data rates of 2.4, 4.8, 9.6, and 56 kilobits/sec, whereas newer networks often use 64 kilobits/sec, 1.5 megabits/sec, and even 45 megabits/sec. There are major economies of scale associated with higher link speeds; for example, the cost of a 1.5 megabit/sec link is about six times that of a 64 kilobit/sec link, but the data rate is 24 times higher. This makes it economically advantageous to concentrate network traffic on a relatively small set of high-speed links. (This effect is seen in Fig. 1.2 with the use of multiplexers or concentrators to share communication costs.)

One result of sharing high-speed (*i.e.*, high data rate) communication links is that the cost of sending data from one point to another increases less than linearly with the geographic separation of the points. This occurs because a user with a long communication path can share one or more high-speed links with other users (thus achieving low cost per unit data) over the bulk of the path, and use low-speed links (which have high cost per unit data) only for local access to the high-speed links.

Estimating the cost of transmission facilities is highly specialized and complex. The cost of a communication link depends on whether one owns the facility or leases it; with leasing, the cost depends on the current competitive and regulatory situation. The details of communication cost will be ignored in what follows, but there are several overall effects of these costs that are important.

First, for wide area data networks, cost has until recently been dominated by transmission costs. Thus, it has been desirable to use the communication links efficiently, perhaps at added computational costs. As will be shown in Section 1.3, the sporadic nature of most data communication, along with the high cost of idle communication links, led to the development of packet data networks.

Second, because of the gradual maturing of optical fiber technology, transmission costs, particularly for high data rate links, are dropping at an accelerating rate which is expected to continue well into the future. The capacity of a single optical fiber using today's technology is 10^9 to 10^{10} bits/sec, and in the future this could rise to 10^{14} or more. In contrast, all the voice and data traffic in the United States amounts to about

Samsung Exhibit 1015, Page 22 of 573

10^{12} bits/sec. Optical fiber is becoming widespread in use and is expected to be the dominant mode of transmission in the future. One consequence of this is that network costs are not expected to be dominated by transmission costs in the future. Another consequence is that network link capacities will increase dramatically; as discussed later, this will change the nature of network applications.

Third, for local area networks, the cost of a network has never been dominated by transmission costs. Coaxial cable and even a twisted pair of wires can achieve relatively high-speed communication at modest cost in a small geographic area. The use of such media and the desire to avoid relatively expensive switching have led to a local area network technology in which many nodes share a common high-speed communication medium on a shared multiaccess basis. This type of network structure is discussed in Chapter 4.

1.1.3 Applications of Data Networks

With the proliferation of computers referred to above, it is not difficult to imagine a growing need for data communication. A brief description of several applications requiring communication will help in understanding the basic problems that arise with data networks.

First, there are many applications centered on remote accessing of central storage facilities and of data bases. One common example is that of a local area network in which a number of workstations without disk storage use one or more common file servers to access files. Other examples are the information services and financial services available to personal computer users. More sophisticated examples, requiring many interactions between the remote site and the data base and its associated programs, include remote computerized medical diagnoses and remote computer-aided education. In some of these examples, there is a cost trade-off between maintaining the data base wherever it might be required and the communication cost of remotely accessing it as required. In other examples, in which the data base is rapidly changing, there is no alternative to communication between the remote sites and the central data base.

Next, there are many applications involving the remote updating of data bases, perhaps in addition to accessing the data. Airline reservation systems, automatic teller machines, inventory control systems, automated order entry systems, and word processing with a set of geographically distributed authors provide a number of examples. Weather tracking systems and military early warning systems are larger-scale examples. In general, for applications of this type, there are many geographically separated points at which data enter the system and often many geographically separated points at which outputs are required. Whether the inputs are processed and stored at one point (as in Figs. 1.1 and 1.2) or processed and stored at many points (as in Fig. 1.3), there is a need for a network to collect the inputs and disseminate the outputs. In any data base with multiple users there is a problem maintaining consistency (*e.g.*, two users of an airline reservation system might sell the same seat on some flight). In geographically distributed systems these problems are particularly acute because of the networking delays.

The communication requirements for accessing files and data bases have been increasing rapidly in recent years. Part of the reason for this is just the natural growth

Samsung Exhibit 1015, Page 23 of 573

of an expanding field. Another reason is that workstations are increasingly graphics oriented, and transmitting a high-resolution image requires millions of bits. Another somewhat related reason is that the link capacities available in local area networks have been much larger than those in wide area networks. As workstation users get used to sending images and large files over local area nets, they expect to do the same over wide area networks. Thus the need for increased link capacity for wide area networks is particularly pressing.

Another popular application is electronic mail between the human users of a network. Such mail can be printed, read, filed, forwarded to other individuals, perhaps with added comments, or read by the addressee at different locations. It is clear that such a service has many advantages over postal mail in terms of delivery speed and flexibility.

In comparison with facsimile, which has become very popular in recent years, electronic mail is more economical, has the flexibility advantages above, and is in principle more convenient for data already stored in a computer. Facsimile is far more convenient for data in hard-copy form (since the hard copy is fed directly into the facsimile machine). It appears clear, however, that the recent popularity of facsimile is due to the fact that it is relatively hassle-free, especially for the occasional or uninitiated user. Unfortunately, electronic mail, and more generally computer communication, despite all the cant about user friendliness, is full of hassles and pitfalls for the occasional or uninitiated user.

There is a similar comparison of electronic mail with voice telephone service. Voice service, in conjunction with an answering machine or voice mail service, in principle has most of the flexibility of electronic mail except for the ability to print a permanent record of a message. Voice, of course, has the additional advantage of immediate two-way interaction and of nonlinguistic communication via inflection and tone. Voice communication is more expensive, but requires only a telephone rather than a telephone plus computer.

As a final application, one might want to use a remote computer system for some computational task. This could happen as a means of load sharing if the local computer is overutilized. It could also arise if there is no local computer, if the local computer is inoperational, or the remote computer is better suited to the given task. Important special cases of the latter are very large problems that require supercomputers. These problems frequently require massive amounts of communication, particularly when the output is in high resolution graphic form. Present-day networks, with their limited link speeds, are often inadequate for these tasks. There are also "real-time" computational tasks in which the computer system must respond to inputs within some maximum delay. If such a task is too large for the local computer, it might be handled by a remote supercomputer or by a number of remote computers working together. Present-day networks are also often inadequate for the communication needs of these tasks.

It will be noted that all the applications above could be satisfied by a network with centralized computer facilities as in Fig. 1.1 or 1.2. To see this, simply visualize moving all the large computers, data bases, and subnet nodes in the network of Fig. 1.3 to one centralized location, maintaining links between all the nodes previously connected. The central facilities would then be connected by short communication lines rather than long, but aside from some changes in propagation delays, the overall network would be

Samsung Exhibit 1015, Page 24 of 573

unchanged. Such a geographically centralized but logically distributed structure would both allow for shared memory between the computers and for centralized repair. Why, then, are data networks with geographically distributed computational and data base facilities growing so quickly in importance? One major reason is the cost and delay of communication. With distributed computers, many computational tasks can be handled locally. Even for remote tasks, communication costs can often be reduced significantly by some local processing. Another reason is that organizations often acquire computers for local automation tasks, and only after this local automation takes place does the need for remote interactions arise. Finally, organizations often wish to have control of their own computer systems rather than be overly dependent on the pricing policies, software changes, and potential security violations of a computer utility shared with many organizations.

Another advantage often claimed for a network with distributed computational facilities is increased reliability. For the centralized system in Fig. 1.2 there is some truth to this claim, since the failure of a communication link could isolate a set of sites from all access to computation. For the geographically centralized but logically distributed network, especially if there are several disjoint paths between each pair of sites, the failure of a communication link is less critical and the question of reliability becomes more complex. If all the large computers and data bases in a network were centralized, the network could be destroyed by a catastrophe at the central site. Aside from this possibility, however, a central site can be more carefully protected and repairs can be made more quickly and easily than with distributed computational sites. Other than these effects, there appears to be no reason why geographically distributed computational facilities are inherently more or less reliable than geographically centralized (but logically distributed) facilities. At any rate, the main focus in what follows will be on networks as in Figs. 1.3 and 1.4, where the communication subnet is properly viewed as the center of the entire network.

1.2 MESSAGES AND SWITCHING

1.2.1 Messages and Packets

A message in a data network corresponds roughly to the everyday English usage of the word. For example, in an airline reservation system, we would regard a request for a reservation, including date, flight number, passenger names, and so on, as a message. In an electronic mail system, a message would be a single document from one user to another. If that same document is then forwarded to several other users, we would sometimes want to regard this forwarding as several new messages and sometimes as forwarding of the same message, depending on the context. In a file transfer system, a message would usually be regarded as a file. In an image transmission system (*i.e.*, pictures, figures, diagrams, etc.), we would regard a message as an image. In an application requiring interactive communication between two or more users, a message would be one unit of communication from one user to another. Thus, in an interactive transaction,

Samsung Exhibit 1015, Page 25 of 573

user 1 might send a message to user 2, user 2 might reply with a message to 1, who might then send another message to 2, and so forth until the completion of the overall transaction. The important characteristic of a message is that from the standpoint of the network users, it is a single unit of communication. If a recipient receives only part of a message, it is usually worthless.

It is sometimes necessary to make a distinction between a message and the representation of the message. Both in a subnet and in a computer, a message is usually represented as a string of binary symbols, 0 or 1. For brevity, a binary symbol will be referred to as a *bit*. When a message goes from sender to recipient, there can be several transformations on the string of bits used to represent the message. Such transformations are sometimes desirable for the sake of data compression and sometimes for the sake of facilitating the communication of the message through the network. A brief description of these two purposes follows.

The purpose of data compression is to reduce the length of the bit string representing the message. From the standpoint of information theory, a message is regarded as one of a collection of possible messages, with a probability distribution on the likelihood of different messages. Such probabilities can only be crudely estimated, either a priori or adaptively. The idea, then, is to assign shorter bit strings to more probable messages and longer bit strings to less probable messages, thus reducing the expected length of the representation. For example, with text, one can represent common letters in the alphabet (or common words in the dictionary) with a small number of bits and represent unusual letters or words with more bits. As another example, in an airline reservation system, the common messages have a very tightly constrained format (date, flight number, names, etc.) and thus can be very compactly represented, with longer strings for unusual types of situations. Data compression will be discussed more in Chapter 2 in the context of compressing control overhead. Data compression will not be treated in general here, since this topic is separable from that of data networks, and is properly studied in its own right, with applications both to storage and point-to-point communication.

Transforming message representations to facilitate communication, on the other hand, is a central topic for data networks. In subsequent chapters, there are many examples in which various kinds of control overhead must be added to messages to ensure reliable communication, to route the message to the correct destination, to control congestion, and so on. It will also be shown that transmitting very long messages as units in a subnet is harmful in several ways, including delay, buffer management, and congestion control. Thus, messages represented by long strings of bits are usually broken into shorter bit strings called *packets*. These packets can then be transmitted through the subnet as individual entities and reassembled into messages at the destination.

The purpose of a subnet, then, is to receive packets at the nodes from sites outside the subnet, then transmit these packets over some path of communication links and other nodes, and finally deliver them to the destination sites. The subnet must somehow obtain information about where the packet is going, but the meaning of the corresponding message is of no concern within the subnet. To the subnet, a packet is simply a string of bits that must be sent through the subnet reliably and quickly. We return to this issue in Section 1.3.

Samsung Exhibit 1015, Page 26 of 573

1.2.2 Sessions

Messages between two users usually occur as a sequence in some larger transaction; such a message sequence (or, equivalently, the larger transaction) is called a *session*. For example, updating a data base usually requires an interchange of several messages. Writing a program at a terminal for a remote computer usually requires many messages over a considerable time period. Typically, a setup procedure (similar to setting up a call in a voice network) is required to initiate a session between two users, and in this case a session is frequently called a *connection*. In other networks, no such setup is required and each message is treated independently; this is called a *connectionless* service. The reasons for these alternatives are discussed later.

From the standpoint of network users, the messages within a session are typically triggered by particular events. From the standpoint of the subnet, however, these message initiation times are somewhat arbitrary and unpredictable. It is often reasonable, for subnet purposes, to model the sequence of times at which messages or packets arrive for a given session as a random process. For simplicity, these arrivals will usually be modeled as occurring at random points in time, independently of each other and of the arrivals for other sessions. This type of arrival process is called a *Poisson* process and is defined and discussed in Section 3.3. This model is not entirely realistic for many types of sessions and ignores the interaction between the messages flowing in the two directions for a session. However, such simple models provide insight into the major trade-offs involved in network design, and these trade-offs are often obscured in more realistic and complex models.

Sometimes it will be more convenient to model message arrivals within a session by an on/off flow model. In such a model, a message is characterized by a sequence of bits flowing into the subnet at a given rate. Successive message arrivals are separated by random durations in which no flow enters the network. Such a model is appropriate, for example, for voice sessions and for real-time monitoring types of applications. When voice is digitized (see Section 2.2), there is no need to transmit when the voice is silent, so these silence periods correspond to the gaps in an on/off flow model. One might think that there is little fundamental difference between a model using point arrivals for messages and a model using on/off flow. The output from point message arrivals, followed by an access line of fixed rate, looks very much like an on/off flow (except for the possibilitity that one message might arrive while another is still being sent on the access line). The major difference between these models, however, is in the question of delay. For sessions naturally modeled by point message arrivals (*e.g.*, data base queries), one is usually interested in delay from message arrival to the delivery of the entire message (since the recipient will process the entire message as a unit). For sessions naturally modeled by flow (such as digitized voice), the concept of a message is somewhat artificial and one is usually interested in the delay experienced by the individual bits within the flow. It appears that the on/off flow model is growing in importance and is particularly appropriate for ISDN and broadband ISDN networks. Part of the reason for this growth is the prevalence of voice in ISDN and voice and video in broadband ISDN. An-

Samsung Exhibit 1015, Page 27 of 573

other reason, which will be more clear later, is that very long messages, which will be prevalent with ISDN, are probably better treated in the subnet as flows than as point arrivals.

To put this question of modeling message arrivals for a session in a more pragmatic way, note that networks, particularly wide area networks built around a subnet as in Fig. 1.3, generally handle multiple applications. Since the design and implementation of a subnet is a time-consuming process, and since applications are rapidly changing and expanding, subnets must be designed to handle a wide variety of applications, some of which are unknown and most of which are subject to change. Any complex model of message arrivals for sessions is likely to be invalid by the time the network is used. This point of view, that subnets must be designed to work independently of the fine details of applications, is discussed further in Section 1.3.

At this point we have a conceptual view, or model, of the function of a subnet. It will provide communication for a slowly varying set of sessions; within each session, messages of some random length distribution arrive at random times according to some random process. Since we will largely ignore the interaction between the two directions of message flow for a session, we shall usually model a two-way session as two one-way sessions, one corresponding to the message flow in one direction and the other in the opposite direction. In what follows we use the word *session* for such one-way sessions. In matters such as session initiation and end-to-end acknowledgment, distinctions are made between two-way and one-way sessions.

In principle a session could involve messages between more than two users. For example, one user could broadcast a sequence of messages to each of some set of other users, or the messages of each user in the set could be broadcast to each of the other users. Such sessions might become important in the future, especially for broadband ISDN, with applications such as video conferencing and television broadcast. We will not discuss such applications in any detail, but instead will simply model multiuser sessions as a multiplicity of one-way two-user sessions.

Although the detailed characteristics of different kinds of applications will not be examined, there are some gross characteristics of sessions that must be kept in mind. The most important are listed:

1. *Message arrival rate and variability of arrivals.* Typical arrival rates for sessions vary from zero to more than enough to saturate the network. Simple models for the variability of arrivals include Poisson arrivals, deterministic arrivals (*i.e.*, a fixed time interval from each message to the next message), and uniformly distributed arrivals (*i.e.*, the time interval between successive messages has a uniform probability density between some minimum and maximum interval).
2. *Session holding time.* Sometimes (as with electronic mail) a session is initiated for a single message. Other sessions last for a working day or even permanently.
3. *Expected message length and length distribution.* Typical message lengths vary roughly from a few bits to 10^9 bits, with file transfer applications at the high end and interactive sessions from a terminal to a computer at the low end. Simple models for length distribution include an exponentially decaying probability density, a uniform

Samsung Exhibit 1015, Page 28 of 573

probability density between some minimum and maximum, and fixed length. As mentioned above, long messages are becoming much more common because of graphics and long file transfers.

4. *Allowable delay.* The allowable expected delay varies from about 10 msec for some real-time control applications to 1 sec or less for interactive terminal to computer applications, to several minutes or more for some file transfer applications. In other applications, there is a maximum allowable delay (in contrast to expected delay). For example, with packetized voice, fixed-length segments of the incoming voice waveform are encoded into packets at the source. At the destination, these packets must be reconverted into waveform segments with some fixed overall delay; any packet not received by this time is simply discarded. As described above, delay is sometimes of interest on a message basis and sometimes, in the flow model, on a bit basis.
5. *Reliability.* For some applications, all messages must be delivered error-free. For example, in banking applications, in transmission of computer programs, or in file transfers, a single bit error in a message can have serious consequences. In other applications, such as electronic mail, all messages must be delivered, but an occasional bit error in a message can usually be visually corrected by the reader. Finally, in other applications, both occasional bit errors and occasional loss of entire packets or messages are allowable. For example, in distributed sensor systems, messages are sometimes noisy when transmitted, and occasional lost messages are soon replaced with more up-to-date messages. For packetized voice, the occasional loss (or late delivery) of a packet or an occasional bit error simply increases the noisiness of the received voice signal. It should be noted, however, that the use of data compression for packetized voice and other applications greatly increases the need for error-free communication.
6. *Message and packet ordering.* The packets within a message must either be maintained in the correct order going through the network or restored to the correct order at some point. For many applications (such as updating data bases), messages must also be delivered in the correct order, whereas for other applications, message order is unimportant. The question of where to handle reliability and message ordering (*i.e.*, at the external sites or within the subnet or both) is an important design issue. This is discussed in Section 2.8.

In keeping all these characteristics in mind, it is often helpful to focus on four types of applications which lie somewhat at the extreme points and which do not interact very well together in subnets. One is interactive terminal to computer sessions, in which messages are short, the message rate is low, the delay requirement is moderately stringent, and the need for reliability is high. Another is file transfer sessions, in which the messages are very long, the message arrival rate is typically low, the delay requirement is very relaxed, and the need for reliability is very high. The third is high-resolution graphics, in which the messages are again long, sometimes up to 10^9 bits, the delay requirement is stringent, and the arrival rate is low. The fourth is packetized voice. Here the concept of a message is not very useful, but the packets are short, the packet arrival rate is high,

Samsung Exhibit 1015, Page 29 of 573

the maximum delay requirement is stringent, and the need for reliability is rather low. A network that can handle all these applications together will probably not have too much difficulty with the other applications of interest.

1.2.3 Circuit Switching and Store-and-Forward Switching

There are two general approaches, known as circuit switching and store-and-forward switching, that can be used within a subnet to transmit the traffic for the various sessions. A brief overview will be given of the circuit switching approach, followed by the reason why this approach leads to inefficient utilization of the communication channels for many types of sessions. Next, an overview of the store-and-forward approach will be given, showing how it overcomes the above inefficiency.

For the circuit switching approach, when a session s is initiated, it is allocated a given transmission rate r_s in bits per second (this could be different in the two directions of a two-way session, but we focus on a one-way session here). A path is then created from the transmitting site through the subnet and to the destination site. Each communication link on this path then allocates a portion r_s of its total transmission capacity in the given direction for that session. This allocation of transmission rates to different sessions on a communication link is usually done by time-division multiplexing (TDM) or frequency-division multiplexing (FDM), but the details of that are explained in Section 2.1. What is important is that the sum of the rates for all the sessions using a link cannot exceed the total capacity of the link. Thus, if a communication link is fully allocated to existing sessions, a new session cannot use that link. If no path can be found using links with at least r_s bits/sec of unused rate, the new session must be rejected (*i.e.*, given a busy signal). The other important point is that once the session has been successfully initiated, it has a guaranteed transmission rate r_s through the network. The nodes then simply take the incoming bit stream for a given session off the incoming link and switch it to the allocated portion of the outgoing link. This type of switching is quite similar to the well-developed technology for switching in the telephone network. In the telephone network, however, each session is allocated the same transmission rate, whereas in a data network, the required transmission rates are different and vary over a wide range.

Circuit switching is rarely used for data networks. In the past, the reason for this has had nothing to do with the potential complexity of the switching, but rather, as we now explain, has been because of very inefficient use of the links. Typical data sessions tend to have short bursts of high activity followed by lengthy inactive periods; circuit switching wastes the allocated rate during these inactive periods. For a more quantitative view, let λ be the message arrival rate for a given session s. More precisely, $1/\lambda$ is the expected interarrival time between messages of s. Let $\overline{X}$ be the expected transmission time of a message over a given link in the path; that is, if $\overline{L}$ is the expected length (in bits) of messages from s, and r_s is the bit rate allocated to s, then $\overline{X} = \overline{L}/r_s$. Figure 1.5 illustrates these arrivals and transmission times.

Note from the figure that the fraction of time in which session s's portion of the link is actually transmitting messages is rather small; that portion of the link is otherwise

Samsung Exhibit 1015, Page 30 of 573

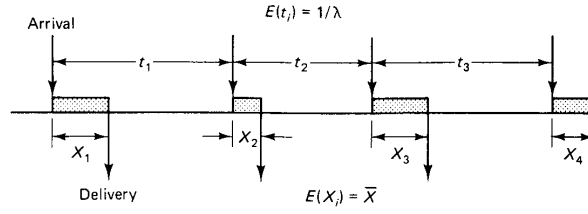


Figure 1.5 Link utilization. The expected transmission time of a message is $\overline{X}$. The expected interarrival period is $1/\lambda$. Thus, the link is used at most $\lambda\overline{X}$ of the time.

idle. It is intuitively plausible, since $1/\lambda$ is the expected interarrival time and $\overline{X}$ is the expected busy time between arrivals, that the ratio of $\overline{X}$ to $1/\lambda$ (*i.e.*, $\lambda\overline{X}$) is the fraction of time in which the portion of the link allocated to s is busy. This argument is made precise in Chapter 3. Our conclusion then is that if $\lambda\overline{X} \ll 1$, session s's portion of the link is idle most of the time (*i.e.*, inefficiently utilized).

To complete our argument about the inefficiency of circuit switching for data networks, we must relate $\overline{X}$ to the allowable expected delay T from message arrival at the source to delivery at the destination. Since $\overline{X}$ is the expected time until the last bit of the message has been sent on the first link, we must have $\overline{X} + P \leq T$, where P is the propagation delay through the network. Thus $\lambda\overline{X} < \lambda T$. If $\lambda T \ll 1$ (*i.e.*, the allowable delay is small relative to the message interarrival rate), the utilization $\lambda\overline{X}$ for the session is correspondingly small. In summary, the bit rate r_s allocated to a session must be large enough to allow message transmission within the required delay, and when $\lambda T \ll 1$, this implies inefficient utilization of the link. Sessions for which $\lambda T \ll 1$ are usually referred to as *bursty* sessions.

For many of the interactive terminal sessions carried by data networks, λT is on the order of 0.01 or less. Thus, with circuit switching, that fraction of a link allocated to such sessions is utilized at most 1% of the time. The conclusion we draw from this is that if link costs are a dominant part of the cost of a network and if bursty sessions require a dominant fraction of link capacity using circuit switching, then circuit switching is an unattractive choice for data networks. Up to the present, both the assumptions above have been valid, and for this reason, data networks have not used circuit switching. The argument above has ignored propagation delays, switching delays in the nodes, and queueing delays. (Queueing delay arises when a message from session s arrives while another message from s is in transmission.) Since these delays must be added to the link transmission time $\overline{X}$ in meeting the delay requirement T, $\overline{X}$ must often be substantially smaller than T, making circuit switching even more inefficient. While propagation and switching delays are often negligible, queueing delay is not, as shown in Chapter 3, particularly when λT is close to or exceeds 1.

In the future, it appears that link costs will become less important in the overall cost of a network. Also, with optical fiber, the marginal cost of link capacity is quite small, so that the wasted capacity of circuit switching will become less important. Finally, it appears that bursty interactive terminal traffic will grow considerably more slowly than

Samsung Exhibit 1015, Page 31 of 573

link capacities in the future (the reason for this is discussed later). Thus circuit switching is a feasible possibility (although not necessarily the best possibility) for networks of the future. Part of the issue here is that as link speeds increase, node processing speed must also increase, putting a premium on simple processing within the subnet. It is not yet clear whether circuit switching or store-and-forward allows simpler subnet processing at high link speeds, but store-and-forward techniques are currently receiving more attention.

In the store-and-forward approach to subnet design, each session is initiated without necessarily making any reserved allocation of transmission rate for the session. Similarly, there is no conventional multiplexing of the communication links. Rather, one packet or message at a time is transmitted on a communication link, using the full transmission rate of the link. The link is shared between the different sessions using that link, but the sharing is done on an as needed basis (*i.e.*, demand basis) rather than a fixed allocation basis. Thus, when a packet or message arrives at a switching node on its path to the destination site, it waits in a queue for its turn to be transmitted on the next link in its path.

Store-and-forward switching has the advantage over circuit switching that each communication link is fully utilized whenever it has any traffic to send. In Chapter 3, when queueing is studied, it will be shown that using communication links on a demand basis often markedly decreases the delay in the network relative to the circuit switching approach. Store-and-forward switching, however, has the disadvantage that the queueing delays in the nodes are hard to control. The packets queued at a node come from inputs at many different sites, and thus there is a need for control mechanisms to slow down those inputs when the queueing delay is excessive, or even worse, when the buffering capacity at the node is about to be exceeded. There is a feedback delay associated with any such control mechanism. First, the overloaded node must somehow send the offending inputs some control information (through the links of the network) telling them to slow down. Second, a considerable number of packets might already be in the subnet heading for the given node. This is the general topic of flow control and is discussed in Chapter 6. The reader should be aware, however, that this problem is caused by the store-and-forward approach and is largely nonexistent in the circuit switching approach.

There is a considerable taxonomy associated with store-and-forward switching. *Message switching* is store-and-forward switching in which messages are sent as unit entities rather than being segmented into packets. If message switching were to be used, there would have to be a maximum message size, which essentially would mean that the user would have to packetize messages rather than having packetization done elsewhere. *Packet switching* is store-and-forward switching in which messages are broken into packets, and from the discussion above, we see that store-and-forward switching and packet switching are essentially synonymous. *Virtual circuit routing* is store-and-forward switching in which a particular path is set up when a session is initiated and maintained during the life of the session. This is like circuit switching in the sense of using a fixed path, but it is virtual in the sense that the capacity of each link is shared by the sessions using that link on a demand basis rather than by fixed allocations. *Dynamic routing* (or *datagram routing*) is store-and-forward switching in which each packet finds its own path

Samsung Exhibit 1015, Page 32 of 573

through the network according to the current information available at the nodes visited. Virtual circuit routing is generally used in practice, although there are many interesting intermediate positions between virtual circuit routing and dynamic routing. The general issue of routing is treated in Chapter 5.

1.3 LAYERING

Layering, or layered architecture, is a form of hierarchical modularity that is central to data network design. The concept of modularity (although perhaps not the name) is as old as engineering. In what follows, the word *module* is used to refer either to a device or to a process within some computer system. What is important is that the module performs a given function in support of the overall function of the system. Such a function is often called the *service* provided by the module. The designers of a module will be intensely aware of the internal details and operation of that module. Someone who uses that module as a component in a larger system, however, will treat the module as a "black box." That is, the user will be uninterested in the internal workings of the module and will be concerned only with the inputs, the outputs, and, most important, the functional relation of outputs to inputs (*i.e.*, the service provided). Thus, a black box is a module viewed in terms of its input–output description. It can be used with other black boxes to construct a more complex module, which again will be viewed at higher levels as a bigger black box.

This approach to design leads naturally to a hierarchy of modules in which a module appears as a black box at one layer of the hierarchy, but appears as a system of lower-layer black boxes at the next lower layer of the hierarchy (see Fig. 1.6). At the overall system level (*i.e.*, at the highest layer of the hierarchy), one sees a small collection of top-layer modules, each viewed as black boxes providing some clear-cut service. At the next layer down, each top-layer module is viewed as a subsystem of lower-layer black boxes, and so forth, down to the lowest layer of the hierarchy. As shown in Fig. 1.6, each layer might contain not only black boxes made up of lower-layer modules but also simple modules that do not require division into yet simpler modules.

As an example of this hierarchical viewpoint, a computer system could be viewed as a set of processor modules, a set of memory modules, and a bus module. A processor module could, in turn, be viewed as a control unit, an arithmetic unit, an instruction fetching unit, and an input–output unit. Similarly, the arithmetic unit could be broken into adders, accumulators, and so on.

In most cases, a user of a black box does not need to know the detailed response of outputs to inputs. For example, precisely when an output changes in response to an input is not important as long as the output has changed by the time it is to be used. Thus, modules (*i.e.*, black boxes) can be specified in terms of tolerances rather than exact descriptions. This leads to standardized modules, which leads, in turn, to the possibility of using many identical, previously designed (*i.e.*, off-the-shelf) modules in the same system. In addition, such standardized modules can easily be replaced with new, functionally equivalent modules that are cheaper or more reliable.

Samsung Exhibit 1015, Page 33 of 573

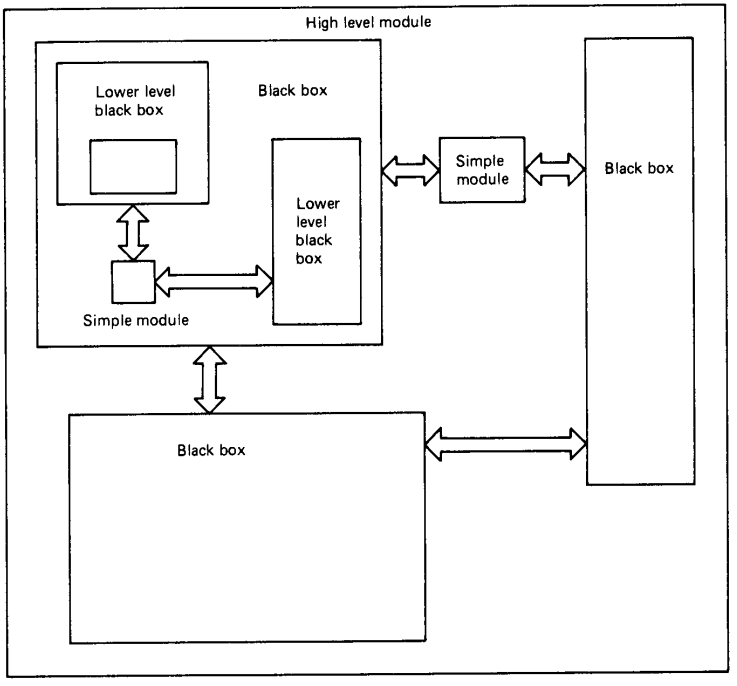


Figure 1.6 Hierarchy of nested black boxes. Each black box (except that at the lowest level) contains black boxes at a lower level, plus perhaps other modules.

All of these advantages of modularity (*i.e.*, simplicity of design; understandability; and standard, interchangeable, widely available modules) provide the motivation for a layered architecture in data networks. A layered architecture can be regarded as a hierarchy of nested modules or black boxes, as described above. Each given layer in the hierarchy regards the next lower layer as one or more black boxes which provide a specified service to the given higher layer.

What is unusual about the layered architecture for data networks is that the black boxes at the various layers are in fact distributed black boxes. The bottom layer of the hierarchy consists of the physical communication links, and at each higher layer, each black box consists of a lower-layer black box communication system plus a set of simple modules, one at each end of the lower-layer communication system. The simple modules associated with a black box at a given layer are called *peer processes* or *peer modules* (see Fig. 1.7).

In the simplest case, a black box consists of two peer processes, one at each of two nodes, and a lower-layer black box communication system connecting the two peer processes. One process communicates with its peer at the other node by placing a message

Samsung Exhibit 1015, Page 34 of 573

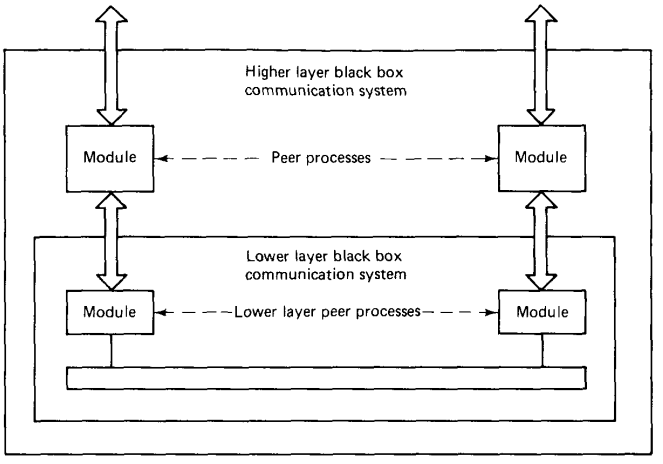


Figure 1.7 Peer processes within a black box communication system. The peer processes communicate through a lower-layer black box communication system that itself contains lower-layer peer processes.

into the lower-layer black box communication system. This lower-layer black box, as illustrated in Fig. 1.7, might in fact consist of two lower-layer peer processes, one at each of the two nodes, connected by a yet lower-layer black box communication system. As a familiar example, consider two heads of state who have no common language for communication. One head of state can then send a message to the peer head of state by a local translator, who communicates in a common language to a peer translator, who then delivers the message in the language of the peer head of state.

Note that there are two quite separate aspects to the communication between a module, say at layer n, and its layer n peer at another node. The first is the protocol (or distributed algorithm) that the peer modules use in exchanging messages or bit strings so as to provide the required functions or service to the next higher layer. The second is the specification of the precise interface between the layer n module at one node and the layer $n-1$ module at the same node through which the messages above are actually exchanged. The first aspect above is more important (and more interesting) for a conceptual understanding of the operation of a layered architecture, but the second is also vital in the actual design and standardization of the system. In terms of the previous example of communication between heads of state, the first aspect has to do with the negotiation between the heads of state, whereas the second has to do with each head of state ensuring that the translator can actually translate the messages faithfully.

Figure 1.8 illustrates such a layered architecture. The layers are those of the reference model of open systems interconnection (OSI) developed as an international standard for data networks by the International Standards Organization (ISO). Many existing networks, including SNA, DECNET, ARPANET, and TYMNET, have somewhat

Samsung Exhibit 1015, Page 35 of 573

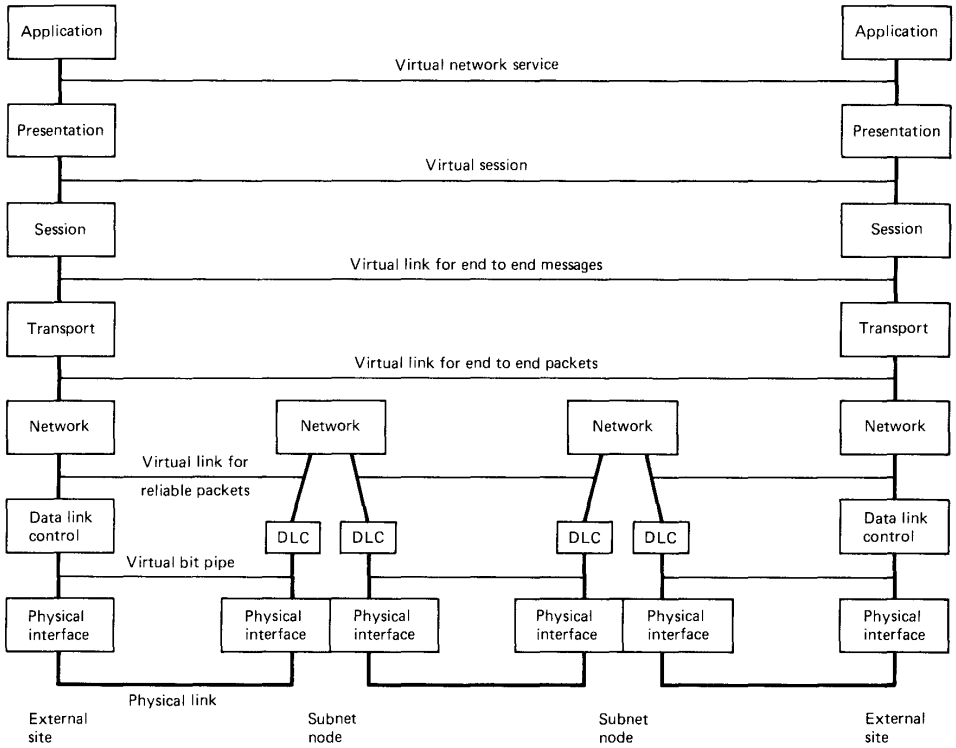


Figure 1.8 Seven-layer OSI network architecture. Each layer presents a virtual communication link with given properties to the next-higher layer.

different layers than this proposed standard. However, the OSI layers have a relatively clean structure that helps in understanding the concept of layering. Some of the variations used by these other networks are discussed later.

1.3.1 The Physical Layer

The function of the *physical layer* is to provide a virtual link for transmitting a sequence of bits between any pair of nodes (or any node and external site) joined by a physical communication channel. Such a virtual link is called a *virtual bit pipe*. To achieve this function, there is a physical interface module on each side of the communication channel whose function is to map the incoming bits from the next higher layer [*i.e.*, the data link control (DLC) layer] into signals appropriate for the channel, and at the receiving end, to map the signals back into bits. The physical interface module that

Samsung Exhibit 1015, Page 36 of 573

performs these mapping functions is often called a *modem* (digital data *mo*dulator and *dem*odulator). The term *modem* is used broadly here to refer to any module that performs the function above, whether or not modulation is involved; for example, if the physical communication channel is a digital link (see Section 2.2), there is nothing for the modem to do other than interface with the DLC module.

Modems and communication channels are discussed in Section 2.2. The modem designer must be aware of the detailed characteristics of the communication channel (and different modems must be designed for different types of channels). To the higher layers, however, the black box formed by the modem–channel–modem combination appears as a bit pipe with the complexities of the physical channel hidden. Even viewed as a bit pipe, however, there are a few issues that must be discussed.

The first issue has to do with the timing of the bit sequence entering the bit pipe. There are three common situations. The first is that of a *synchronous* bit pipe where bits are transmitted and received at regular intervals (*i.e.*, 1 bit per t second interval for some t). The higher-layer DLC module must supply bits at this synchronous rate whether or not it has any real data to send. The second situation is that of an *intermittent synchronous* bit pipe where the DLC module supplies bits at a synchronous rate when it has data to send and stops sending bits when there are no data to send. The third situation is that of *asynchronous characters*, usually used with personal computers and low-speed terminals. Here, keyboard characters and various control characters are mapped into fixed-length bit strings (usually, eight-bit strings according to a standard mapping from characters to bit strings known as ASCII code), and the individual character bit strings are transmitted asynchronously as they are generated.

The next issue is that of the interface between the DLC module and the modem. One would think that not many problems should exist in delivering a string of bits from one module to another, especially if they are physically close. Unfortunately, there are a number of annoying details about such an interface. For example, the module on one end of the interface might be temporarily inoperable, and when both become operable, some initialization is required to start the flow of bits. Also, for synchronous operation, one side or the other must provide timing. To make matters worse, many different manufacturers provide the modules on either end, so there is a need for standardizing the interface. In fact, there are many such standards, so many that one applauds the effort but questions the success. Two of the better known are RS-232-C and the physical layer of X.21.

The RS-232-C interface approaches the problem by providing a separate wire between the two modules for each type of control signal that might be required. These wires from the modules are joined in a standard 25-pin connector (although usually many fewer wires are required). In communication jargon, the interface is between a DCE (data communication equipment), which is the modem in this case, and a DTE (data terminal equipment), which is the DLC layer and higher layers in this case.

As an example of the interface use, suppose that the DTE wants to start sending data (either on initialization or with a new data sequence in intermittent synchronous transmission). The DTE then sends a signal to the DCE on a "request-to-send" wire. The DCE replies with a signal on the "clear-to-send" wire. The DCE also sends a signal

Samsung Exhibit 1015, Page 37 of 573

on the "DCE-ready" wire whenever it is operational and a signal on the "carrier detect" wire whenever it appears that the opposite modem and channel are operational. If the DTE receives all these signals (which are just level voltages), it starts to send data over the interface on the DTE-to-DCE data wire.

This interchange is a very simple example of a *protocol* or *distributed algorithm.* Each module performs operations based both on its own state and on the information received from the other module. Many less trivial protocols are developed in subsequent chapters. There are many other details in RS-232-C operation but no other new concepts.

It is sometimes helpful when focusing on the interface between the DLC module and the modem to view the wires between the modules as a physical channel and to view the DLC and modem as peer processes executing the interface protocol. To avoid confusion between the DLC module's major function as a peer process with the opposite DLC module and its lower-level function of interfacing with the modem, an extra dummy module is sometimes created (see Fig. 1.9) which exercises the interface protocol with the modem.

The X.21 physical layer interface is similar in function to RS-232-C, but it uses a smaller set of wires (eight wires are used, although there is a 15-pin connector) and is intended as an interface to a digital communication link. The idea is to avoid using a separate wire for each possible signal by doubling up on the use of wires by digital logic in the modules. The X.21 physical layer is used as the physical layer for the X.25 protocol, which is discussed in Chapter 2.

It should be clear from the above that there is a great conceptual advantage in removing the question of modem and modem interfaces from the higher-level aspects of networks. Note that this has already been done, in essence, in previous sections in referring to the number of bits per second that could be transmitted over communication links. It should also be noted, however, that modems cannot be totally segregated from network issues. For example, is it better to have a modem that transmits R bits/sec with an error rate of 10^{-6} or a modem that transmits $2R$ bits/sec with an error rate of 10^{-4}? This cannot be answered without some knowledge of how errors are eliminated at higher

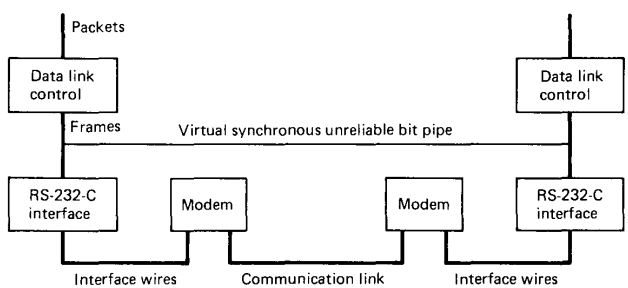


Figure 1.9 Layering with the interface between the DLC and the modem viewed as an interface over a physical medium consisting of a set of wires.

Samsung Exhibit 1015, Page 38 of 573

layers of the architecture. Conversely, decisions on how and where to eliminate errors at higher layers should depend on the error rate and error characteristics at the physical layer.

1.3.2 The Data Link Control Layer

The second layer in Fig. 1.8 is the *data link control (DLC) layer*. Each point-to-point communication link (*i.e.*, the two-way virtual bit pipe provided by layer 1) has data link control modules (as peer processes) at each end of the link. The customary purpose of data link control is to convert the unreliable bit pipe at layer 1 into a higher-level, virtual communication link for sending packets asynchronously but error-free in both directions over the link. From the standpoint of the DLC layer, a packet is simply a string of bits that comes from the next higher layer.

The communication at this layer is asynchronous in two ways. First, there is a variable delay between the entrance of a packet into the DLC module at one end of the link and its exit from the other end. This variability is due both to the need to correct the errors that occur at the physical layer and to the variable length of the packets. Second, the time between subsequent entries of packets into the DLC module at one end of the link is also variable. The latter variability is caused both because higher layers might have no packets to send at a given time and also because the DLC is unable to accept new packets when too many old packets are being retransmitted due to transmission errors.

Data link control is discussed in detail in Chapter 2. In essence, the sending DLC module places some overhead control bits called a *header* at the beginning of each packet and some more overhead bits called a *trailer* at the end of each packet, resulting in a longer string of bits called a *frame*. Some of these overhead bits determine if errors have occurred in the transmitted frames, some request retransmissions when errors occur, and some delineate the beginning and ending of frames. The algorithms (or protocols) for accomplishing these tasks are distributed between the peer DLC modules at the two ends of each link and are somewhat complex because the control bits themselves are subject to transmission errors.

The DLC layers in some networks do not retransmit packets in the presence of errors. In these networks, packets in error are simply dropped and retransmission is attempted on an end-to-end basis at the transport layer. The relative merits of this are discussed in Section 2.8.2. Typically, the DLC layer ensures that packets leave the receiving DLC in the same order in which they enter the transmitting DLC, but not all data link control strategies ensure this feature; the relative merits of ordering are also discussed in Section 2.8.2.

Our previous description of the physical layer and DLC was based on point-to-point communication links for which the received waveform at one end of the link is a noisy replica of the signal transmitted at the other end. In some networks, particularly local area networks, some or all of the communication takes place over multiaccess links. For these links, the signal received at one node is a function of the signals from a multiplicity of transmitting nodes, and the signal transmitted from one node might be

Samsung Exhibit 1015, Page 39 of 573

heard at a multiplicity of other nodes. This situation arises in satellite communication, radio communication, and communication over cables, optical fibers, and telephone lines with multiple taps. Multiaccess communication is treated in Chapter 4.

The MAC sublayer The appropriate layers for multiaccess communication are somewhat different from those in networks of point-to-point links. There is still the need for a DLC layer to provide a virtual error-free packet link to higher layers, and there is still the need for a physical layer to provide a bit pipe. However, there is also a need for an intermediate layer to manage the multiaccess link so that frames can be sent by each node without constant interference from the other nodes. This is called *medium access control* (MAC). It is usually considered as the lower sublayer of layer 2 with the conventional DLC considered as the higher sublayer. Figure 1.10 illustrates the relationship between these layers. The service provided by the MAC to the DLC is that of an intermittent synchronous bit pipe. The function of the MAC sublayer is to allocate the multiaccess channel so that each node can successfully transmit its frames without undue interference from the other nodes; see Chapter 4 for various ways of accomplishing this function.

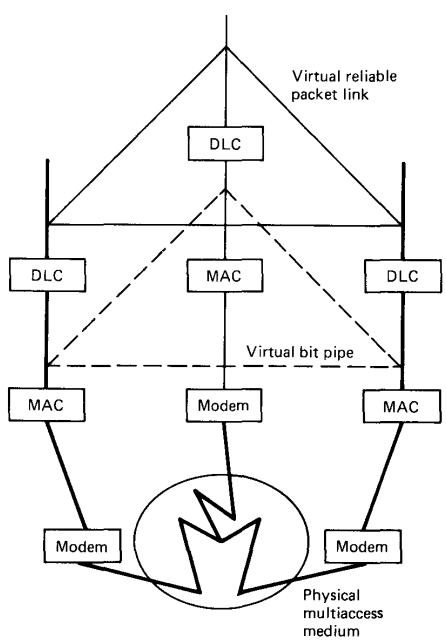


Figure 1.10 Layering for a multiaccess channel. The physical medium is accessed by all three users, each of whom hears the transmitted signals of the others. The DLC sublayer sees virtual point-to-point bit pipes below it. The MAC sublayer sees a multiaccess bit pipe, and the modems access the actual channel.

Samsung Exhibit 1015, Page 40 of 573

1.3.3 The Network Layer

The third layer in Fig. 1.8 is the *network layer*. There is one network layer process associated with each node and with each external site of the network. All these processes are peer processes and all work together in implementing routing and flow control for the network. When a frame enters a node or site from a communication link, the bits in that frame pass through the physical layer to the DLC layer. The DLC layer determines where the frame begins and ends, and if the frame is accepted as correct, the DLC strips off the DLC header and trailer from the frame and passes the resulting packet up to the network layer module (see Fig. 1.11). A packet consists of two parts, a packet header followed by the packet body (and thus a frame at the DLC layer contains first the DLC header, next the packet header, next the packet body, and then the DLC trailer). The network layer module uses the packet header of an incoming packet, along with stored information at the module, to accomplish its routing and flow control functions. Part of the principle of layering is that the DLC layer does not look at the packet header or packet body in performing its service function, which is to deliver the packet reliably to the network layer at the next node. Similarly, the network layer does not use any of the information in the DLC header or trailer in performing its functions of routing and flow control. The reason for this separation is to allow improvements, modifications, and replacements in the internal operation of one layer without forcing the other to be modified.

Newly generated messages from users at an external site are processed by the higher layers, broken into packet-sized pieces if need be, and passed down from the transport layer module to the network module. These packet-sized pieces constitute the packet body at the network layer. The transport layer also provides additional information about how to handle the packet (such as where the packet is supposed to go), but this information is passed to the network layer as a set of parameters in accordance with the interfacing protocol between transport and network layer. The network layer module uses

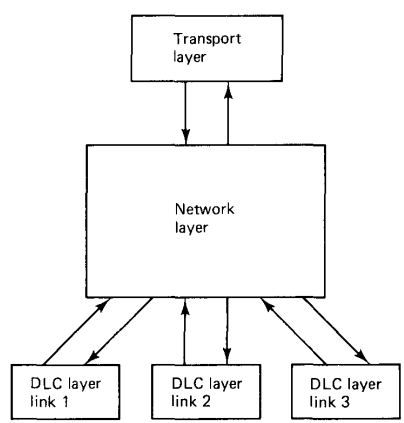


Figure 1.11 The network layer at a node or site can receive packets from a DLC layer for each incoming link and (in the case of a site) from the transport layer. It can send these packets out to the same set of modules.

Samsung Exhibit 1015, Page 41 of 573

these parameters, along with its own stored information, to generate the packet header in accordance with the protocol between peer network layer modules.

Along with the transit packets arriving at the network layer module from the lower layer, and the new packets arriving from the higher layer, the network layer can generate its own control packets. These control packets might be specific to a given session, dealing with initiating or tearing down the session, or might have a global function, informing other nodes of link congestion, link failures, and so on.

For networks using virtual circuit routing (*i.e.*, in which the route for a session is fixed over the life of the session), the routing function at the network layer module consists of two parts. The first is to select a route when the virtual circuit is being initiated, and the second is to ensure that each packet of the session follows the assigned route. The selection of a route could be carried out in a distributed way by all the nodes, or could be carried out by the source node or by some central node entrusted with this task. No matter how the job is allocated between the different nodes, however, there is need for considerable communication, via network control packets, concerning the operating characteristics and level of traffic and delay throughout the network. This subject is treated in considerable depth in Chapter 5. Ensuring that each packet follows the assigned route is accomplished by placing enough information in the packet header for the network layer module at each node of the route to be able to forward the packet on the correct outgoing link (or to pass the packet body up to the transport layer when the destination is reached). Ways of doing this are discussed in Section 2.8.

Datagram networks, on the other hand, do not have a connection phase in which a route for a session is determined. Rather, each packet is routed individually. This appears to be a very natural and simple approach, but as Chapter 5 shows, the dynamics of the traffic patterns in a network and the lack of timely knowledge about those patterns at the nodes make this much more difficult than one would think. Most wide area networks use virtual circuits for this reason.

It is necessary here to make a distinction between virtual circuit or datagram *operation* at the network layer and virtual circuit or datagram *service*. The discussion above concerned the *operation* of the network layer; the user of the network layer (usually the transport layer) is concerned only with the *service* offered. Since successive packets of a session, using datagram operation, might travel on different routes, they might appear at the destination out of order. Thus (assuming that the network layer module at the destination does not reorder the packets), the service offered by such a network layer allows packets to get out of order. Typically, with datagram operation, packets are sometimes dropped also. As a result, datagram *service* is usually taken to mean that the network layer can deliver packets out of order, can occasionally fail to deliver packets, and requires no connection phase at the initiation of a session. Conversely, virtual circuit *service* is taken to mean that all packets are delivered once, only once, and in order, but that a connection phase is required on session initiation. We will often use the term *connectionless service* in place of *datagram service* and *connection-oriented service* in place of *virtual circuit service*. We shall see that the difference between connectionless and connection-oriented service has as much to do with quality of service, flow control, and error recovery as it does with routing.

Samsung Exhibit 1015, Page 42 of 573

The other major function of the network layer, along with routing, is flow control, or congestion control. Some authors make a distinction between flow control and congestion control, viewing the first as avoiding sending data faster than the final destination can absorb it, and the second as avoiding congestion within the subnet. Actually, if the destination cannot absorb packets as fast as they are sent, those packets will remain in the subnet and cause congestion there. Similarly, if a link in the subnet is congested (*i.e.*, many packets are buffered in an adjacent node waiting for transmission on the link), then there are a number of mechanisms that cause the congestion to spread. Thus congestion is a global issue that involves both the subnet and the external sites, and at least at a conceptual level, it is preferable to treat it as a single problem.

Fundamentally, congestion occurs when the users of the network collectively demand more resources than the network (including the destination sites) has to offer. Good routing can help to alleviate this problem by spreading the sessions out over the available subnet resources. Good buffer management at the nodes can also help. Ultimately, however, the network layer must be able to control the flow of packets into the network, and this is what is meant by flow control (and why we use the term *flow control* in place of *congestion control*).

The control of packet flow into the network must be done in such a way as to prevent congestion and also to provide equitable service to the various sessions. Note that with connection-oriented service, it is possible for a session to negotiate its requirements from the network as a compromise between user desires and network utilization. Thus in some sense the network can guarantee the service as negotiated. With connectionless service, there is no such opportunity for negotiation, and equitable service between users does not have much meaning. This is another reason for the prevalence of connection-oriented service in wide area networks. In Chapter 6 we develop various distributed algorithms for performing the flow control function. As with routing, flow control requires considerable exchange of information between the nodes. Some of this exchange occurs through the packet headers, and some through control packets.

One might hope that the high link capacities that will be available in the future will make it possible to operate networks economically with low utilization, thus making flow control unnecessary. Unfortunately, this view appears overly simplistic. As link capacities increase, access rates into networks will also increase. Thus, even if the aggregate requirements for network service are small relative to the available capacity, a single malfunctioning user could dump enough data into the network quickly to cause serious congestion; if the network plays no regulatory role, this could easily lead to very chaotic service for other users.

The discussion of routing and flow control above has been oriented primarily toward wide area networks. Most local area networks can be viewed as using a single multiaccess channel for communication, and consequently any node is capable of receiving any packet. Thus routing is not a major problem for local area networks. There is a possibility of congestion in local area networks, but this must be dealt with in the MAC sublayer. Thus, in a sense, the major functions of the network layer are accomplished in the MAC sublayer, and the network layer is not of great importance in local area networks. For

Samsung Exhibit 1015, Page 43 of 573

this reason, the arguments for virtual circuit operation and connection oriented service in the network layer do not apply to local area networks, and connectionless service is common there.

The network layer is conceptually the most complex of the layered hierarchy since *all* the peer processes at this layer must work together. For the lower layers (except for the MAC sublayer for multiaccess), the peer processes are paired, one at each side of a communication link. For the higher layers, the peer processes are again paired, one at each end of a session. Thus, the network layer and the MAC sublayer are the only layers in which the overall algorithms are distributed between many geographically separated processes.

Acquiring the ability to design and understand such distributed algorithms is one of the basic objectives of this book. Chapter 2 covers the simpler forms of distributed algorithms involving just two peer processes at opposite ends of a link. In Chapter 4 we treat distributed algorithms involving many peer processes in the context of the MAC sublayer, and Chapters 5 and 6 deal with distributed algorithms involving many peer processes at the network layer.

When the network layer and lower layers at all nodes and sites are regarded as one black box, a packet entering the network layer from the next higher layer at a site reappears at some later time at the interface between the network layer and the next higher layer at the destination site. Thus, the network layer appears as a virtual, packet-carrying, end-to-end link from origin site to destination site. Depending on the design of the network layer, this virtual link might be reliable, delivering every packet, once and only once, without errors, or might be unreliable, failing to deliver some packets and delivering some packets with errors. The higher layers then might have to recover from these errors. The network layer might also deliver all packets for each session in order or might deliver them out of order. The relative merits of these alternatives are discussed further in Section 2.8.

The internet sublayer Despite all efforts at standardization, different networks use different algorithms for routing and flow control at the network layer. We have seen some of the reasons for this variety in our discussion of wide area versus local area networks. Since these network layer algorithms are distributed and require close coordination between the various nodes, it is not surprising that one cannot simply connect different subnetworks together. The accepted solution to this problem is to create a new sublayer called the *internet sublayer*. This is usually regarded as being the top part of the network layer. Several subnets can be combined by creating special nodes called gateways between them. A gateway connecting two subnets will interface with each subnet through a network layer module appropriate for that subnet. From the standpoint of the subnet, then, a gateway looks like an external site.

Each gateway will have an internet sublayer module sitting on top of the network layer modules for the individual subnets. When a packet arrives at a gateway from one subnet, the corresponding network layer module passes the packet body and subsidiary information about the packet to the internet module (which thus acts like a transport layer module to the network layer module). This packet body and subsidiary information is

Samsung Exhibit 1015, Page 44 of 573

then passed down to the other network layer module for forwarding on through the other subnet.

The internet modules also must play a role in routing and flow control. There is not a great deal of understanding in the field yet as to the appropriate ways for the internet sublayer and the various network layers to work together on routing and flow control. From a practical standpoint, the problem is exacerbated by the fact that the network layers for the subnets are usually in place, designed without the intention of later being used in a network of networks. Thus the internet layer must of necessity be somewhat ad hoc.

When combining local area networks, where routing and flow control are exercised at the MAC sublayer, it is often possible to replace the gateway between subnets with a bridge. Bridges interface different subnets at the DLC layer rather than at the network layer; for local area networks, this is possible because the routing and flow control are done in the MAC sublayer. In Chapter 5 we discuss gateways and bridges in greater detail, particularly with respect to routing.

1.3.4 The Transport Layer

The fourth layer in Fig. 1.8 is the *transport layer*. Here, for each virtual end-to-end link provided by the network layer (or internet sublayer), there is a pair of peer processes, one at each end of the virtual end-to-end link. The transport layer has a number of functions, not all of which are necessarily required in any given network.

First, the transport layer breaks messages into packets at the transmitting end and reassembles packets into messages at the receiving end. This reassembly function is relatively simple if the transport layer process has plenty of buffer space available, but can be quite tricky if there is limited buffer space that must be shared between many virtual end-to-end links. If the network layer delivers packets out of order, this reassembly problem becomes even more difficult.

Second, the transport layer might multiplex several low-rate sessions, all from the same source site and all going to the same destination site, into one session at the network layer. Since the subnet sees only one session in this case, the number of sessions in the subnet and the attendant overhead is reduced. Often this is carried to the extreme in which all sessions with a common source site and common destination site are multiplexed into the same session. In this case, the addressing at the network layer need only specify the source and destination sites; the process within the source site and destination site are then specified in a transport layer header.

Third, the transport layer might split one high-rate session into multiple sessions at the network layer. This might be desirable if the flow control at the network layer is incapable of providing higher-rate service to some sessions than others, but clearly a better solution to this problem would be for the network layer to adjust the rate to the session requirement.

Fourth, if the network layer is unreliable, the transport layer might be required to achieve reliable end-to-end communication for those sessions requiring it. Even when the network layer is designed to provide reliable communication, the transport layer has to be

Samsung Exhibit 1015, Page 45 of 573

involved when one or the other end site fails or when the network becomes disconnected due to communication link failures. These failure issues are discussed further in Section 2.8 and in Chapters 5 and 6.

Fifth, end-to-end flow control is often done at the transport layer. There is little difference between end-to-end flow control at the transport layer and network layer (or internet sublayer if it exists). End-to-end flow control at the transport layer is common in practice but makes an integrated approach to avoiding congestion somewhat difficult. This is discussed further in Section 2.9.4 and in Chapter 6.

A header is usually required at the transport layer; this transport header, combined with the data being transported, serves as the packet body passed on to the network layer. Thus the actual body of data is encapsulated in a sequence of headers with the lowest layers on the outside (see Fig. 1.12). At the destination, these layer headers are peeled off in passing up through the various layers. In ISO terminology, the body of data shown in the figure is referred to as a transport service data unit (T-SDU). This data unit, along with the transport header, is referred to as a transport protocol data unit (T-PDU). This unit is also the body of the packet at the network layer, which is sometimes referred to as a network service data unit (N-SDU). Similarly, the packet body plus packet header is referred to as a network protocol data unit (N-PDU). Similarly, each layer in the hierarchy has an SDU, as the unit coming in from the higher layer, and a PDU as the unit going down to the next-lower layer. It is difficult to know where to take a stand against acronymitis in the network field, but we will continue to use the more descriptive terminology of messages, packets, and frames.

1.3.5 The Session Layer

The *session layer* is the next layer above the transport layer in the OSI hierarchy of Fig. 1.8. One function of the session layer is akin to the directory assistance service in the telephone network. That is, if a user wants an available service in the network

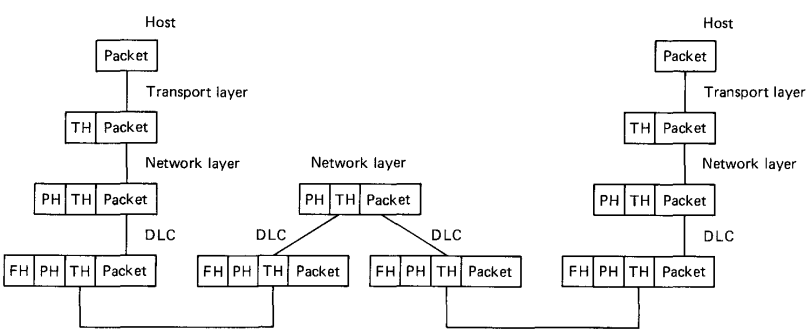


Figure 1.12 Illustration of various headers on a frame. Note that each layer looks only at its own header.

Samsung Exhibit 1015, Page 46 of 573

but does not know where to access that service, this layer provides the transport layer with the information needed to establish the session. For example, this layer would be an appropriate place to achieve load sharing between many processors that are sharing computational tasks within a network.

The session layer also deals with *access rights* in setting up sessions. For example, if a corporation uses a public network to exchange records between branch offices, those records should not be accessible to unauthorized users. Similarly, when a user accesses a service, the session layer helps deal with the question of who pays for the service.

In essence, the session layer handles the interactions between the two end points in setting up a session, whereas the network layer handles the subnet aspects of setting up a session. The way that session initiation is divided between session layer, transport layer, and network layer varies from network to network, and many networks do not have these three layers as separate entities.

1.3.6 The Presentation Layer

The major functions of the *presentation layer* are data encryption, data compression, and code conversion. The need for encryption in military organizations is obvious, but in addition, corporations and individual users often must send messages that should only be read by the intended recipient. Although data networks should be designed to prevent messages from getting to the wrong recipients, one must expect occasional malfunctions both at the external sites and within the subnet; this leads to the need for encryption of critical messages.

The desirability of data compression in reducing the number of bits to be communicated has already been mentioned. This function could be performed at any of the layers, but there is some advantage in compressing the data for each session separately, in the sense that different sessions have different types of redundancy in their messages. In particular, data compression must be done (if at all) before encryption, since encrypted data will not have any easily detectable redundancy.

Finally, code conversion is sometimes necessary because of incompatible terminals, printers, graphics terminals, file systems, and so on. For example, some terminals use the ASCII code to represent characters as 8-bit bytes, whereas other terminals use the EBCDIC code. Messages using one code must be converted to the other code to be readable by a terminal using the other code.

1.3.7 The Application Layer

The *application layer* is simply what is left over after the other layers have performed their functions. Each application requires its own software (*i.e.*, peer processes) to perform the desired application. The lower layers perform those parts of the overall task that are required for many different applications, while the application layer does that part of the task specific to the particular application.

At this point, the merits of a layered approach should be clear, but there is some question about which functions should be performed at each layer. Many networks omit

Samsung Exhibit 1015, Page 47 of 573

the session and presentation layers, and as we have seen, the lower layers are now divided into sublayers. An even more serious issue is that in an effort to achieve agreement on the standards, a number of alternatives have been introduced which allow major functions to be either performed or not at various layers. For example, error recovery is sometimes done at the DLC layer and sometimes not, and because of this, the higher layers cannot necessarily count on reliable packet transfer. Thus, even within the class of networks that conform to the OSI reference model, there is considerable variation in the services offered by the various layers. Many of the existing networks described later do not conform to the OSI reference model, and thus introduce even more variation in layer services. Broadband ISDN networks, for example, do routing and flow control at the physical layer (in a desire to simplify switch design), thus making the network look like an end-to-end bit pipe from the origin to destination.

Even with all these problems, there is a strong desire for standardization of the interfaces and functions provided by each layer, even if the standard is slightly inappropriate. This desire is particularly strong for international networks and particularly strong among smaller equipment manufacturers who must design equipment to operate correctly with other manufacturers' equipment. On the other hand, standardization is an impediment to innovation, and this impediment is particularly important in a new field, such as data networks, that is not yet well understood. Fortunately, there is a constant evolution of network standards. For example, the asynchronous transfer node (ATM) protocol of broadband ISDN circumvents the ISO network layer standard by performing the function of the network layer at the physical layer (see Section 2.10).

One particular difficulty with the seven layers is that each message must pass through seven processes before even entering the subnet, and all of this might generate considerable delay. This text neither argues for this particular set of layers nor proposes another set. Rather, the objective is to create an increased level of understanding of the functions that must be accomplished, with the hope that this will contribute to standardization. The existing networks examined in subsequent chapters do not, in fact, have layers that quite correspond to the OSI layers.

1.4 A SIMPLE DISTRIBUTED ALGORITHM PROBLEM

All of the layers discussed in Section 1.3 have much in common. All contain peer processes, one at each of two or more geographically separated points, and the peer processes communicate via the communication facility provided by the next lower layer. The peer processes in a given layer have a common objective (*i.e.*, task or function) to be performed jointly, and that objective is achieved by some combination of processing and interchanging information. The algorithm to achieve the objective is a *distributed algorithm* or a *protocol*. The distributed algorithm is broken down into a set of *local algorithms*, one of which is performed by each peer process. The local algorithm performed by one process in a set of peers consists of carrying out various operations on the available data, and at various points in the algorithm, either sending data to one or more other peer processes or reading (or waiting for) data sent by another peer process.

Samsung Exhibit 1015, Page 48 of 573

In the simplest distributed algorithm, the order in which operations are carried out by the various local algorithms is completely determined. For example, one local algorithm might perform several operations and then reliably send some data to the other local algorithm, which then carries out some operations and returns some data. Only one local algorithm operates at a time and the distributed algorithm is similar to a centralized algorithm that performs all operations sequentially at one location.

In more complex cases, several local algorithms might operate concurrently, but each still waits at predetermined points in the local algorithm for predetermined messages from specific other local algorithms. In this case, the overall distributed algorithm still operates in a deterministic fashion (given the input data to the peer processes), but the lockstep ordering of operations between different local algorithms is removed.

In the most complex case (which is of most interest), the order in which a local algorithm performs its operations depends on the order in which data arrive (either from the next higher layer or from a peer process). Also, if the underlying communication facility is unreliable, data sent by a peer process might never arrive or might arrive with errors.

Most people are very familiar with the situation above, since people must often perform tasks requiring interacting with others, often with unreliable communication. In these situations, however, people deal with problems as they arise rather than thinking through all possible eventualities ahead of time, as must be done with a distributed algorithm.

To gain some familiarity with distributed algorithms, a very simple problem is presented, involving unreliable communication, which in fact has no solution. Analogous situations arise frequently in the study of data networks, so it is well to understand such a problem in its simplest context.

There are three armies, two colored magenta and one lavender. The lavender army separates the two magenta armies, and if the magenta armies can attack simultaneously, they win, but if they attack separately, the lavender army wins. The only communication between the magenta armies is by sending messengers through the lavender army lines, but there is a possibility that any such messenger will be captured, causing the message to go undelivered (see Fig. 1.13). The magenta armies would like to synchronize their attack at some given time, but each is unwilling to attack unless assured with certainty that the other will also attack. Thus, the first army might send a message saying, "Let's attack on Saturday at noon; please acknowledge if you agree."

The second army, hearing such a message, might send a return message saying, "We agree; send an acknowledgment if you receive our message." It is not hard to see that this strategy leads to an infinite sequence of messages, with the last army to send a message being unwilling to attack until obtaining a commitment from the other side.

What is more surprising is that no strategy exists for allowing the two armies to synchronize. To see this, assume that each army is initially in state 0 and stays in this state if it receives no messages. If an army commits itself to attack, it goes to state 1, but it will not go to state 1 unless it is certain that the other army will go to state 1. We also assume that an army can change state only at the time that it receives a message (this assumption in essence prevents side information other than messages from synchronizing the armies). Now consider any ordering in which the two armies receive messages. The

Samsung Exhibit 1015, Page 49 of 573

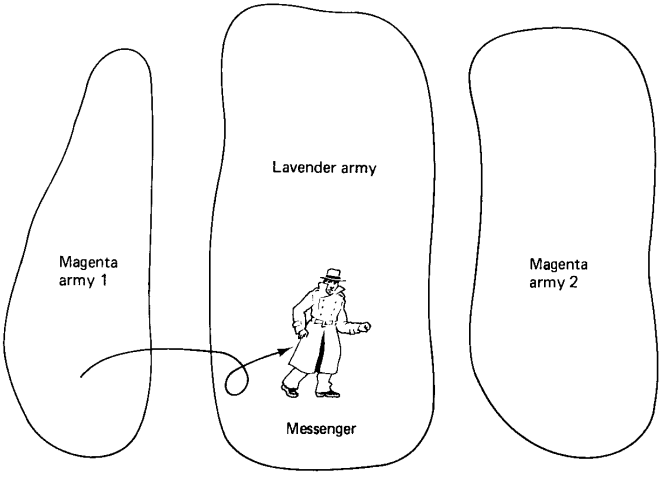


Figure 1.13 A messenger carries a message through enemy lines from magenta army 1 to magenta army 2. If the messenger is caught, the message is undelivered. Magenta army 1 is unaware of capture and magenta army 2 is unaware of existence of message.

first army to receive a message cannot go to state 1, since it has no assurance that any message will be received by the other army. The second army to receive a message also cannot go to state 1 since it is not assured that the other side will receive subsequent messages, and even if it knows that the other side received a first message, it knows that the other side is not currently in state 1. Proceeding in this way (or more formally by induction), neither army can ever go to state 1.

What is surprising about this argument is the difficulty in convincing oneself that it is correct. The difficulty does not lie with the induction argument, but rather with the question of whether the model fairly represents the situation described. It appears that the problem is that we are not used to dealing with distributed questions in a precise way; classical engineering problems do not deal with situations in which distributed decisions based on distributed information must be made.

If the conditions above are relaxed somewhat so as to require only a high probability of simultaneous attack, the problem can be solved. The first army simply decides to attack at a certain time and sends many messengers simultaneously to the other side. The first army is then assured with high probability that the second army will get the message, and the second army is assured that the first army will attack.

Fortunately, most of the problems of communication between peer processes that are experienced in data networks do not require this simultaneous agreement. Typically, what is required is for one process to enter a given state with the assurance that the peer process will *eventually* enter a corresponding state. The first process might be required to wait for a confirmation of this eventuality, but the deadlock situation of the three-army problem, in which neither process can act until after the other has acted, is avoided.

Samsung Exhibit 1015, Page 50 of 573

NOTES AND SUGGESTED READING

The introductory textbooks by Tanenbaum [Tan88], Stallings [Sta85], and Schwartz [Sch87] provide alternative treatments of the material in this chapter. Tanenbaum's text is highly readable and contains several chapters on the higher levels of the OSI architecture. Stalling's text contains a wealth of practical detail on current network practice. Schwartz's text also includes several chapters on circuit switching. Some perspectives on the historical evolution of data networks are given in [Gre84].

New developments in technology and applications are critically important in both network design and use. There are frequent articles in the *IEEE Spectrum*, *IEEE Communications Magazine*, and *IEEE Computer* that monitor these new developments. *Silicon Dreams: Information, Man and Machine* by Lucky [Luc90] provides an excellent overview of these areas. A good reference on layered architecture is [Gre82], and some interesting commentary on future standardization of layers is given in [Gre86].

PROBLEMS

1.1. A high quality image requires a spatial resolution of about 0.002 inch, which means that about 500 pixels (*i.e.* samples) per inch are needed in a digital representation. Assuming 24 bits per pixel for a color image of size 8.5 by 11 inches, find the total number of bits required for such an image representation.

1.2. **(a)** Suppose a city of one million inhabitants builds a data network. Suppose that each inhabitant, during the busiest hour of the day, is involved in an average of 4 transactions per hour that use this network (such as withdrawing money from a cash machine, buying some item in a store and thus generating an inventory control message, etc.). Suppose that each transaction, on the average, causes 4 packets of 1000 bits each to be sent. What is the aggregate average number of bits per second carried by the network? How many 64 kbit/sec voice telephone links are required to carry this traffic assuming that each packet travels over an average of 3 links?

(b) Suppose that the inhabitants use their telephones an average of 10% of the time during the busy hour. How many voice telephone links are required for this, assuming that all calls are within the city and travel over an average of three links?

1.3. Suppose packets can get dropped or arbitrarily delayed inside a packet network. Suppose two users are communicating in a session and want to terminate the session. We would like a protocol that exchanges packets in such a way that both users know that they can terminate with the knowledge that no further packets will arrive from the other user. Can such a protocol be designed? What is the relation between this problem and the three-army problem of Section 1.10?

Samsung Exhibit 1015, Page 51 of 573

Second Edition

Data Networks

DIMITRI BERTSEKAS
Massachusetts Institute of Technology

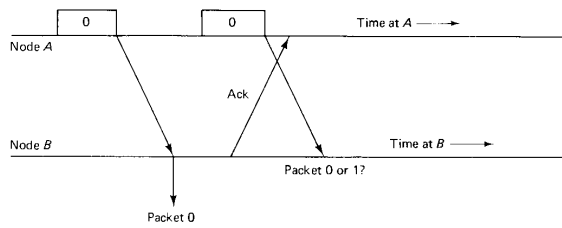
ROBERT GALLAGER
Massachusetts Institute of Technology



PRENTICE HALL, Englewood Cliffs, New Jersey 07632

Samsung Exhibit 1015, Page 52 of 573

2



Point-to-Point Protocols and Links

2.1 INTRODUCTION

This chapter first provides an introduction to the physical communication links that constitute the building blocks of data networks. The major focus of the chapter is then data link control (*i.e.*, the point-to-point protocols needed to control the passage of data over a communication link). Finally, a number of point-to-point protocols at the network, transport, and physical layers are discussed. There are many similarities between the point-to-point protocols at these different layers, and it is desirable to discuss them together before addressing the more complex network-wide protocols for routing, flow control, and multiaccess control.

The treatment of physical links in Section 2.2 is a brief introduction to a very large topic. The reason for the brevity is not that the subject lacks importance or inherent interest, but rather, that a thorough understanding requires a background in linear system theory, random processes, and modern communication theory. In this section we provide a sufficient overview for those lacking this background and provide a review and perspective for those with more background.

Samsung Exhibit 1015, Page 53 of 573

In dealing with the physical layer in Section 2.2, we discuss both the actual communication channels used by the network and whatever interface modules are required at the ends of the channels to transmit and receive digital data (see Fig 2.1). We refer to these modules as modems (digital data modulators and demodulators), although in many cases no modulation or demodulation is required. In sending data, the modem converts the incoming binary data into a signal suitable for the channel. In receiving, the modem converts the received signal back into binary data. To an extent, the combination of modem, physical link, modem can be ignored; one simply recognizes that the combination appears to higher layers as a virtual bit pipe.

There are several different types of virtual bit pipes that might be implemented by the physical layer. One is the *synchronous* bit pipe, where the sending side of the data link control (DLC) module supplies bits to the sending side of the modem at a synchronous rate (*i.e.*, one bit each T seconds for some fixed T). If the DLC module temporarily has no data to send, it must continue to send dummy bits, called *idle fill*, until it again has data. The receiving modem recovers these bits synchronously (with delay and occasional errors) and releases the bits, including idle fill, to the corresponding DLC module.

The *intermittent synchronous* bit pipe is another form of bit pipe in which the sending DLC module supplies bits synchronously to the modem when it has data to send, but supplies nothing when it has no data. The sending modem sends no signal during these idle intervals, and the receiving modem detects the idle intervals and releases nothing to the receiving DLC module. This somewhat complicates the receiving modem, since it must distinguish between 0, 1, and idle in each time interval, and it must also regain synchronization at the end of an idle period. In Chapter 4 it will be seen that the capability to transmit nothing is very important for multiaccess channels.

A final form of virtual bit pipe is the *asynchronous character* pipe, where the bits within a character are sent at a fixed rate, but successive characters can be separated by

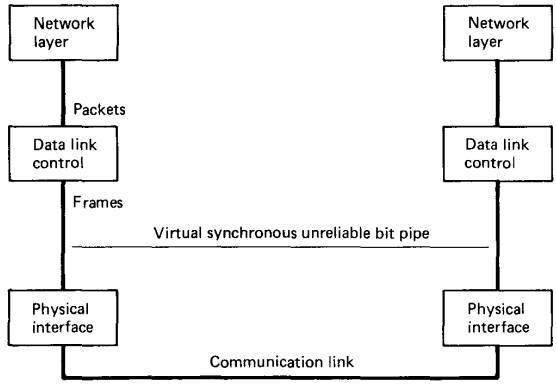


Figure 2.1 Data link control (DLC) layer with interfaces to adjacent layers.

Samsung Exhibit 1015, Page 54 of 573

variable delays, subject to a given minimum. As will be seen in the next section, this is used only when high data rates are not an important consideration.

In Sections 2.3 to 2.7 we treat the DLC layer, which is the primary focus of this chapter. For each point-to-point link, there are two DLC peer modules, one module at each end of the link. For traffic in a given direction, the sending DLC module receives packets from the network layer module at that node. The peer DLC modules employ a distributed algorithm, or protocol, to transfer these packets to the receiving DLC and thence to the network layer module at that node. In most cases, the objective is to deliver the packets in the order of arrival, with neither repetitions nor errors. In accomplishing this task, the DLC modules make use of the virtual bit pipe (with errors) provided by the physical layer.

One major problem in accomplishing the objective above is that of correcting the bit errors that occur on the virtual bit pipe. This objective is generally accomplished by a technique called ARQ (*a*utomatic *r*epeat re*q*uest). In this technique, errors are first detected at the receiving DLC module and then repetitions are requested from the transmitting DLC module. Both the detection of errors and the requests for retransmission require a certain amount of control overhead to be communicated on the bit pipe. This overhead is provided by adding a given number of bits (called a *header*) to the front of each packet and adding an additional number of bits (called a *trailer*) to the rear (see Fig. 2.2). A packet, extended by this header and trailer, is called a *frame*. From the standpoint of the DLC, a packet is simply a string of bits provided as a unit by the network layer; examples arise later where single "packets," in this sense, from the network layer contain pieces of information from many different sessions, but this makes no difference to the DLC layer.

Section 2.3 treats the problem of error detection and shows how a set of redundant bits in the trailer can be used for error detection. Section 2.4 then deals with retransmission requests. This is not as easy as it appears, first because the requests must be embedded into the data traveling in the opposite direction, and second because the opposite direction is also subject to errors. This provides our first exposure to a real distributed algorithm, which is of particular conceptual interest since it must operate in the presence of errors.

In Section 2.5 we discuss framing. The issue here is for the receiving DLC module to detect the beginning and end of each successive frame. For a synchronous bit pipe, the bits within a frame must contain the information to distinguish the end of the frame; also the idle fill between frames must be uniquely recognizable. This problem becomes even more interesting in the presence of errors.

A widely accepted standard of data link control is the HDLC protocol. This is discussed in Section 2.6, followed in Section 2.7 by a general discussion of how to initialize and to disconnect DLC protocols. This topic might appear to be trivial, but on closer inspection, it requires careful thought.

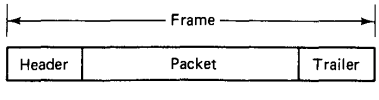


Figure 2.2 Frame structure. A packet from the network layer is extended in the DLC layer with control bits in front and in back of the packet.

Samsung Exhibit 1015, Page 55 of 573

Section 2.8 then treats a number of network layer issues, starting with addressing. End-to-end error recovery, which can be done at either the network or transport layer, is discussed next, along with a general discussion of why error recovery should or should not be done at more than one layer. The section ends with a discussion first of the X.25 network layer standard and next of the Internet Protocol (IP). IP was originally developed to connect the many local area networks in academic and research institutions to the ARPANET and is now a defacto standard for the internet sublayer.

A discussion of the transport layer is then presented in Section 2.9. This focuses on the transport protocol, TCP, used on top of IP. The combined use of TCP and IP (usually called TCP/IP) gives us an opportunity to explore some of the practical consequences of particular choices of protocols.

The chapter ends with an introduction to broadband integrated service data networks. In these networks, the physical layer is being implemented as a packet switching network called ATM, an abbreviation for "asynchronous transfer mode." It is interesting to compare how ATM performs its various functions with the traditional layers of conventional data networks.

2.2 THE PHYSICAL LAYER: CHANNELS AND MODEMS

As discussed in Section 2.1, the virtual channel seen by the data link control (DLC) layer has the function of transporting bits or characters from the DLC module at one end of the link to the module at the other end (see Fig. 2.1). In this section we survey how communication channels can be used to accomplish this function. We focus on point-to-point channels (*i.e.*, channels that connect just two nodes), and postpone consideration of multiaccess channels (*i.e.*, channels connecting more than two nodes) to Chapter 4. We also focus on communication in one direction, thus ignoring any potential interference between simultaneous transmission in both directions.

There are two broad classes of point-to-point channels: digital channels and analog channels. From a black box point of view, a digital channel is simply a bit pipe, with a bit stream as input and output. An analog channel, on the other hand, accepts a waveform (*i.e.*, an arbitrary function of time) as input and produces a waveform as output. We discuss analog channels first since digital channels are usually implemented on top of an underlying analog structure.

A module is required at the input of an analog channel to map the digital data from the DLC module into the waveform sent over the channel. Similarly, a module is required at the receiver to map the received waveform back into digital data. These modules will be referred to as modems (digital data *mo*dulator and *dem*odulator). The term *modem* is used broadly here, not necessarily implying any modulation but simply referring to the required mapping operations.

Let $s(t)$ denote the analog channel input as a function of time; $s(t)$ could represent a voltage or current waveform. Similarly, let $r(t)$ represent the voltage or current waveform at the output of the analog channel. The output $r(t)$ is a distorted, delayed, and attenuated version of $s(t)$, and our objective is to gain some intuitive appreciation of how to map

Samsung Exhibit 1015, Page 56 of 573

the digital data into $s(t)$ so as to minimize the deleterious effects of this distortion. Note that a black box viewpoint of the physical channel is taken here, considering the input–output relation rather than the internal details of the analog channel. For example, if an ordinary voice telephone circuit (usually called a voice-grade circuit) is used as the analog channel, the physical path of the channel will typically pass through multiple switches, multiplexers, demultiplexers, modulators, and demodulators. For dial-up lines, this path will change on each subsequent call, although the specification of tolerances on the input–output characterization remain unchanged.

2.2.1 Filtering

One of the most important distorting effects on most analog channels is linear time-invariant filtering. Filtering occurs not only from filters inserted by the channel designer but also from the inherent behavior of the propagation medium. One effect of filtering is to "smooth out" the transmitted signal $s(t)$. Figure 2.3 shows two examples in which $s(t)$ is first a single rectangular pulse and then a sequence of rectangular pulses. The defining properties of linear time-invariant filters are as follows:

1. If input $s(t)$ yields output $r(t)$, then for any τ, input $s(t-\tau)$ yields output $r(t-\tau)$.
2. If $s(t)$ yields $r(t)$, then for any real number α, $\alpha s(t)$ yields $\alpha r(t)$.
3. If $s_1(t)$ yields $r_1(t)$ and $s_2(t)$ yields $r_2(t)$, then $s_1(t)+s_2(t)$ yields $r_1(t)+r_2(t)$.

The first property above expresses the time invariance; namely, if an input is delayed by τ, the corresponding output remains the same except for the delay of τ. The second and third properties express the linearity. That is, if an input is scaled by α, the corresponding output remains the same except for being scaled by α. Also, the

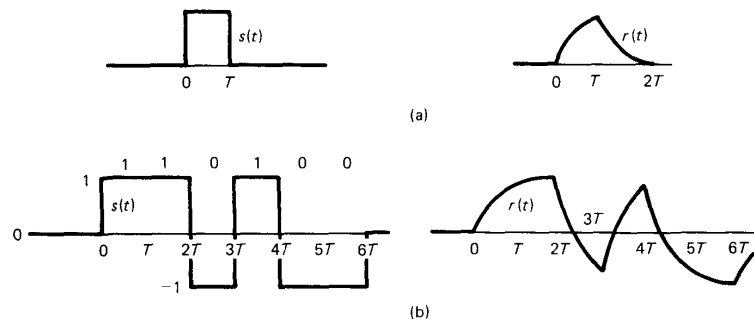


Figure 2.3 Relation of input and output waveforms for a communication channel with filtering. Part (a) shows the response $r(t)$ to an input $s(t)$ consisting of a rectangular pulse, and part (b) shows the response to a sequence of pulses. Part (b) also illustrates the NRZ code in which a sequence of binary inputs (1 1 0 1 0 0) is mapped into rectangular pulses. The duration of each pulse is equal to the time between binary inputs.

Samsung Exhibit 1015, Page 57 of 573

output due to the sum of two inputs is simply the sum of the corresponding outputs. As an example of these properties, the output for Fig. 2.3(b) can be calculated from the output in Fig. 2.3(a). In particular, the response to a negative-amplitude rectangular pulse is the negative of that from a positive-amplitude pulse (property 2), the output from a delayed pulse is delayed from the original output (property 1), and the output from the sum of the pulses is the sum of the outputs for the original pulses (property 3).

Figure 2.3 also illustrates a simple way to map incoming bits into an analog waveform $s(t)$. The virtual channel accepts a new bit from the DLC module each T seconds; the bit value 1 is mapped into a rectangular pulse of amplitude +1, and the value 0 into amplitude -1. Thus, Fig. 2.3(b) represents $s(t)$ for the data sequence 110100. This mapping scheme is often referred to as a nonreturn to zero (NRZ) code. The notation NRZ arises because of an alternative scheme in which the pulses in $s(t)$ have a shorter duration than the bit time T, resulting in $s(t)$ returning to 0 level for some interval between each pulse. The merits of these schemes will become clearer later.

Next, consider changing the rate at which bits enter the virtual channel. Figure 2.4 shows the effect of increasing the rate by a factor of 4 (*i.e.*, the signaling interval is reduced from T to $T/4$). It can be seen that output $r(t)$ is more distorted than before. The problem is that the response to a single pulse lasts much longer than a pulse time, so that the output at a given t depends significantly on the polarity of several input pulses; this phenomenon is called *intersymbol interference*.

From a more general viewpoint, suppose that $h(t)$ is the channel output corresponding to an infinitesimally narrow pulse of unit area at time 0. We call $h(t)$ the impulse response of the channel. Think of an arbitrary input waveform $s(t)$ as a superposition of very narrow pulses as shown in Fig. 2.5. The pulse from $s(\tau)$ to $s(\tau + \delta)$ can be viewed as a small impulse of area $\delta s(\tau)$ at time τ; this gives rise to the output $\delta s(\tau)h(t - \tau)$ at

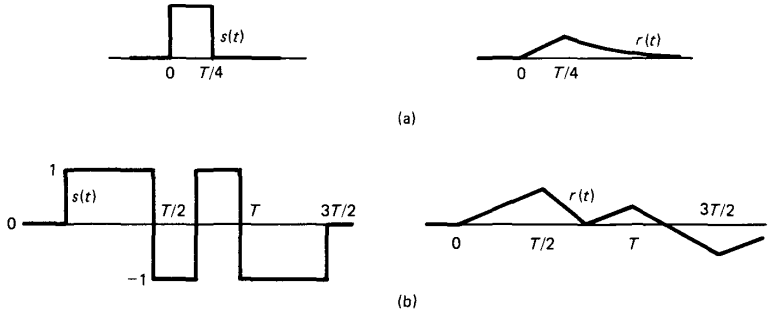


Figure 2.4 Relation of input and output waveforms for the same channel as in Fig. 2.3. Here the binary digits enter at 4 times the rate of Fig. 2.3, and the rectangular pulses last one-fourth as long. Note that the output $r(t)$ is more distorted and more attenuated than that in Fig. 2.3.

Samsung Exhibit 1015, Page 58 of 573

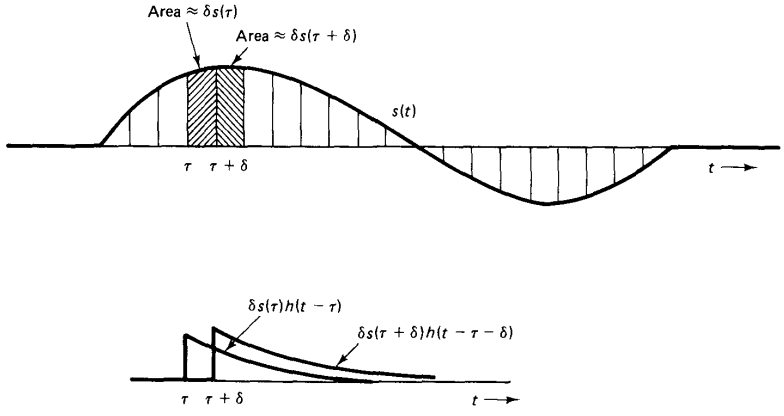


Figure 2.5 Graphical interpretation of the convolution equation. Input $s(t)$ is viewed as the superposition of narrow pulses of width δ. Each such pulse yields an output $\delta s(\tau)h(t-\tau)$. The overall output is the sum of these pulse responses.

time t. Adding the responses at time t from each of these input pulses and going to the limit $\delta \to 0$ shows that

$$r(t) = \int_{-\infty}^{+\infty} s(\tau)h(t-\tau)d\tau \tag{2.1}$$

This formula is called the *convolution integral*, and $r(t)$ is referred to as the convolution of $s(t)$ and $h(t)$. Note that this formula asserts that the filtering aspects of a channel are completely characterized by the impulse response $h(t)$; given $h(t)$, the output $r(t)$ for any input $s(t)$ can be determined. For the example in Fig. 2.3, it has been assumed that $h(t)$ is 0 for $t < 0$ and $\alpha e^{-\alpha t}$ for $t \geq 0$, where $\alpha = 2/T$. Given this, it is easy to calculate the responses in the figure.

From Eq. (2.1), note that the output at a given time t depends significantly on the input $s(\tau)$ over the interval where $h(t-\tau)$ is significantly nonzero. As a result, if this interval is much larger than the signaling interval T between successive input pulses, significant amounts of intersymbol interference will occur.

Physically, a channel cannot respond to an input before the input occurs, and therefore $h(t)$ should be 0 for $t < 0$; thus, the upper limit of integration in Eq. (2.1) could be taken as t. It is often useful, however, to employ a different time reference at the receiver than at the transmitter, thus eliminating the effect of propagation delay. In this case, $h(t)$ could be nonzero for $t < 0$, which is the reason for the infinite upper limit of integration.

2.2.2 Frequency Response

To gain more insight into the effects of filtering, it is necessary to look at the frequency domain as well as the time domain; that is, one wants to find the effect of filtering

Samsung Exhibit 1015, Page 59 of 573

on sinusoids of different frequencies. It is convenient analytically to take a broader viewpoint here and allow the channel input $s(t)$ to be a complex function of t; that is, $s(t) = \text{Re}[s(t)] + j\ \text{Im}[s(t)]$, where $j = \sqrt{-1}$. The actual input to a channel is always real, of course. However, if $s(t)$ is allowed to be complex in Eq. (2.1), $r(t)$ will also be complex, but the output corresponding to $\text{Re}[s(t)]$ is simply $\text{Re}[r(t)]$ [assuming that $h(t)$ is real], and the output corresponding to $\text{Im}[s(t)]$ is $\text{Im}[r(t)]$. For a given frequency f, let $s(\tau)$ in Eq. (2.1) be the complex sinusoid $e^{j2\pi f\tau} = \cos(2\pi f\tau) + j\sin(2\pi f\tau)$. Integrating Eq. (2.1) (see Problem 2.3) yields

$$r(t) = H(f)e^{j2\pi ft} \tag{2.2}$$

where

$$H(f) = \int_{-\infty}^{\infty} h(\tau)e^{-j2\pi f\tau}d\tau \tag{2.3}$$

Thus, the response to a complex sinusoid of frequency f is a complex sinusoid of the same frequency, scaled by the factor $H(f)$. $H(f)$ is a complex function of the frequency f and is called the *frequency response* of the channel. It is defined for both positive and negative f. Let $|H(f)|$ be the magnitude and $\angle H(f)$ the phase of $H(f)$ [*i.e.*, $H(f) = |H(f)|e^{j\angle H(f)}$]. The response $r_1(t)$ to the real sinusoid $\cos(2\pi ft)$ is given by the real part of Eq. (2.2), or

$$r_1(t) = |H(f)|\cos[2\pi ft + \angle H(f)] \tag{2.4}$$

Thus, a real sinusoidal input at frequency f gives rise to a real sinusoidal output at the same freqency; $|H(f)|$ gives the amplitude and $\angle H(f)$ the phase of that output relative to the phase of the input. As an example of the frequency response, if $h(t) = \alpha e^{-\alpha t}$ for $t \geq 0$, then integrating Eq. (2.3) yields

$$H(f) = \frac{\alpha}{\alpha + j2\pi f} \tag{2.5}$$

Equation (2.3) maps an arbitrary time function $h(t)$ into a frequency function $H(f)$; mathematically, $H(f)$ is the *Fourier transform* of $h(t)$. It can be shown that the time function $h(t)$ can be recovered from $H(f)$ by the *inverse Fourier transform*

$$h(t) = \int_{-\infty}^{\infty} H(f)e^{j2\pi ft}df \tag{2.6}$$

Equation (2.6) has an interesting interpretation; it says that an (essentially) arbitrary function of time $h(t)$ can be represented as a superposition of an infinite set of infinitesimal complex sinusoids, where the amount of each sinusoid per unit frequency is $H(f)$, as given by Eq. (2.3). Thus, the channel input $s(t)$ (at least over any finite interval) can also be represented as a frequency function by

$$S(f) = \int_{-\infty}^{\infty} s(t)e^{-j2\pi ft}dt \tag{2.7}$$

$$s(t) = \int_{-\infty}^{\infty} S(f)e^{j2\pi ft}df \tag{2.8}$$

Samsung Exhibit 1015, Page 60 of 573

The channel output $r(t)$ and its frequency function $R(f)$ are related in the same way. Finally, since Eq. (2.2) expresses the output for a unit complex sinusoid at frequency f, and since $s(t)$ is a superposition of complex sinusoids as given by Eq. (2.8), it follows from the linearity of the channel that the response to $s(t)$ is

$$r(t) = \int_{-\infty}^{\infty} H(f)S(f)e^{j2\pi ft}df \tag{2.9}$$

Since $r(t)$ is also the inverse Fourier transform of $R(f)$, the input–output relation in terms of frequency is simply

$$R(f) = H(f)S(f) \tag{2.10}$$

Thus, the convolution of $h(t)$ and $s(t)$ in the time domain corresponds to the multiplication of $H(f)$ and $S(f)$ in the frequency domain. One sees from Eq. (2.10) why the frequency domain provides insight into filtering.

If $H(f) = 1$ over some range of frequencies and $S(f)$ is nonzero only over those frequencies, then $R(f) = S(f)$, so that $r(t) = s(t)$. One is not usually so fortunate as to have $H(f) = 1$ over a desired range of frequencies, but one could filter the received signal by an additional filter of frequency response $H^{-1}(f)$ over the desired range; this additional filtering would yield a final filtered output satisfying $r(t) = s(t)$. Such filters can be closely approximated in practice subject to additional delay [*i.e.*, satisfying $r(t) = s(t - \tau)$, for some delay τ]. Such filters can even be made to adapt automatically to the actual channel response $H(f)$; filters of this type are usually implemented digitally, operating on a sampled version of the channel output, and are called *adaptive equalizers*. These filters can and often are used to overcome the effects of intersymbol interference.

The question now arises as to what frequency range should be used by the signal $s(t)$; it appears at this point that any frequency range could be used as long as $H(f)$ is nonzero. The difficulty with this argument is that the additive noise that is always present on a channel has been ignored. Noise is added to the signal at various points along the propagation path, including at the receiver. Thus, the noise is not filtered in the channel in the same way as the signal. If $|H(f)|$ is very small in some interval of frequencies, the signal is greatly attenuated at those frequencies, but typically the noise is not attenuated. Thus, if the received signal is filtered at the receiver by $H^{-1}(f)$, the signal is restored to its proper level, but the noise is greatly amplified.

The conclusion from this argument (assuming that the noise is uniformly distributed over the frequency band) is that the digital data should be mapped into $s(t)$ in such a way that $|S(f)|$ is large over those frequencies where $|H(f)|$ is large and $|S(f)|$ is small (ideally zero) elsewhere. The cutoff point between large and small depends on the noise level and signal level and is not of great relevance here, particularly since the argument above is qualitative and oversimplified. Coming back to the example where $h(t) = \alpha e^{-\alpha t}$ for $t \geq 0$, the frequency response was given in Eq. (2.5) as $H(f) = \alpha/(\alpha + j2\pi f)$. It is seen that $|H(f)|$ is approximately 1 for small f and decreases as $1/f$ for large f. We shall not attempt to calculate $S(f)$ for the NRZ code of Fig. 2.3, partly because $s(t)$ and $S(f)$ depend on the particular data sequence being encoded, but the effect of changing the signaling interval T can easily be found. If T is decreased, then $s(t)$ is compressed

Samsung Exhibit 1015, Page 61 of 573

in time, and this correspondingly expands $S(f)$ in frequency (see Problem 2.5). The equalization at the receiver would then be required either to equalize $H(f)$ over a broader band of frequencies, thus increasing the noise, or to allow more intersymbol interference.

2.2.3 The Sampling Theorem

A more precise way of looking at the question of signaling rates comes from the sampling theorem. This theorem states that if a waveform $s(t)$ is low-pass limited to frequencies at most W [*i.e.*, $S(f) = 0$, for $|f| > W$], then [assuming that $S(f)$ does not contain an impulse at $f = W$], $s(t)$ is completely determined by its values each $1/(2W)$ seconds; in particular,

$$s(t) = \sum_{i=-\infty}^{\infty} s\left(\frac{i}{2W}\right) \frac{\sin[2\pi W(t - i/(2W))]}{2\pi W[t - i/(2W)]} \tag{2.11}$$

Also, for any choice of sample values at intervals 1/(2W), there is a low-pass waveform, given by Eq. (2.11), with those sample values. Figure 2.6 illustrates this result.

The impact of this result, for our purposes, is that incoming digital data can be mapped into sample values at a spacing of $1/(2W)$ and used to create a waveform of the given sample values that is limited to $|f| \leq W$. If this waveform is then passed through an ideal low-pass filter with $H(f) = 1$ for $|f| \leq W$ and $H(f) = 0$ elsewhere, the received waveform will be identical to the transmitted waveform (in the absence of noise); thus, its samples can be used to recreate the original digital data.

The NRZ code can be viewed as mapping incoming bits into sample values of $s(t)$, but the samples are sent as rectangular pulses rather than the ideal $(\sin x)/x$ pulse shape of Eq. (2.11). In a sense, the pulse shape used at the transmitter is not critically important

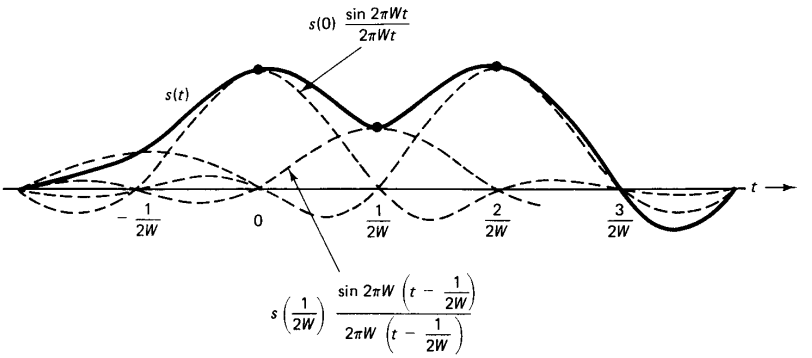


Figure 2.6 Sampling theorem, showing a function $s(t)$ that is low-pass limited to frequencies at most W. The function is represented as a superposition of $(\sin x)/x$ functions. For each sample, there is one such function, centered at the sample and with a scale factor equal to the sample value.

Samsung Exhibit 1015, Page 62 of 573

since the pulse shape can be viewed as arising from filtering sample impulses. If the combination of the pulse-shaping filter at the input with the channel filter and equalizer filter has the $(\sin x)/x$ shape of Eq. (2.11) (or, equivalently, a combined frequency response equal to the ideal low-pass filter response), the output samples will re-create the input samples.

The ideal low-pass filter and the $(\sin x)/x$ pulse shape suggested by the sampling theorem are nice theoretically but not very practical. What is required in practice is a more realistic filter for which the sample values at the output replicate those at the input (*i.e.*, no intersymbol interference) while the high-frequency noise is filtered out.

An elegant solution to this problem was given in a classic paper by Nyquist [Nyq28]. He showed that intersymbol interference is avoided by using a filter $H'(f)$ with odd symmetry at the band edge; that is, $H'(f+W) = 1 - H'(f-W)$ for $|f| \leq W$, and $H'(f) = 0$ for $|f| > 2W$ (see Fig. 2.7). The filter $H'(f)$ here is the composite of the pulse-shaping filter at the transmitter, the channel filter, and the equalizer filter. In practice, such a filter usually cuts off rather sharply around $f = W$ to avoid high-frequency noise, but ideal filtering is not required, thereby allowing considerable design flexibility.

It is important to recognize that the sampling theorem specifies the number of samples per second that can be utilized on a low-pass channel, but it does not specify how many bits can be mapped into one sample. For example, two bits per sample could be achieved by the mapping $11 \rightarrow 3, 10 \rightarrow 1, 00 \rightarrow -1$, and $01 \rightarrow -3$. As discussed later, it is the noise that limits the number of bits per sample.

2.2.4 Bandpass Channels

So far, we have considered only low-pass channels for which $|H(f)|$ is large only for a frequency band around $f = 0$. Most physical channels do not fall into this category, and instead have the property that $|(H(f)|$ is significantly nonzero only within some frequency band $f_1 \leq |f| \leq f_2$, where $f_1 > 0$. These channels are called bandpass channels and many of them have the property that $H(0) = 0$. A channel or waveform with $H(0) = 0$ is said to have no dc component. From Eq. (2.3), it can be seen that this implies that $\int_{-\infty}^{\infty} h(t)dt = 0$. The impulse response for these channels fluctuates

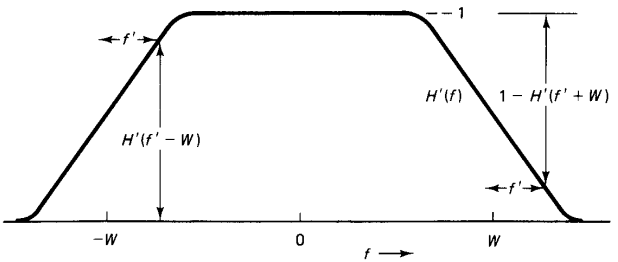


Figure 2.7 Frequency response $H'(f)$ that satisfies the Nyquist criterion for no intersymbol interference. Note that the response has odd symmetry around the point $f = W$.

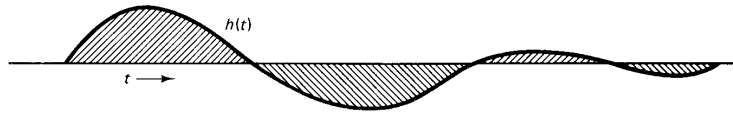


Figure 2.8 Impulse response $h(t)$ for which $H(f) = 0$ for $f = 0$. Note that the area over which $h(t)$ is positive is equal to that over which it is negative.

around 0, as illustrated in Fig. 2.8; this phenomenon is often called ringing. This type of impulse response suggests that the NRZ code is not very promising for a bandpass channel.

To avoid the foregoing problems, most modems for bandpass channels either directly encode digital data into signals with no dc component or else use modulation techniques. The best known direct encoding of this type is Manchester coding (see Fig. 2.9). As seen from the figure, the signals used to represent 0 and 1 each have no dc component and also have a transition (either 1 to -1 or -1 to 1) in the middle of each signaling interval. These transitions simplify the problem of timing recovery at the receiver (note that with the NRZ code, timing recovery could be difficult for a long run of 0's or 1's even in the absence of filtering and noise). The price of eliminating dc components in this way is the use of a much broader band of frequencies than required. Manchester coding is widely used in practice, particularly in the Ethernet system and the corresponding IEEE 802.3 standard described in Section 4.5. There are many other direct encodings, all of a somewhat ad hoc nature, with varying frequency characteristics and no dc component.

2.2.5 Modulation

One of the simplest modulation techniques is amplitude modulation (AM). Here a signal waveform $s(t)$ (called a baseband signal) is generated from the digital data as before, say by the NRZ code. This is then multiplied by a sinusoidal carrier, say $\cos(2\pi f_0 t)$, to generate a modulated signal $s(t)\cos(2\pi f_0 t)$. It is shown in Problem 2.6 that the frequency representation of this modulated signal is $[S(f - f_0) + S(f + f_0)]/2$ (see Fig. 2.10). At the receiver, the modulated signal is again multiplied by $\cos(2\pi f_0 t)$, yielding a received signal

$$\begin{aligned} r(t) &= s(t)\cos^2(2\pi f_0 t) \\ &= \frac{s(t)}{2} + \frac{s(t)\cos(4\pi f_0 t)}{2} \end{aligned} \tag{2.12}$$

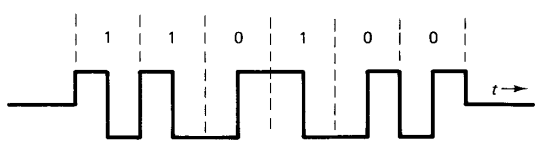


Figure 2.9 Manchester coding. A binary 1 is mapped into a positive pulse followed by a negative pulse, and a binary 0 is mapped into a negative pulse followed by a positive pulse. Note the transition in the middle of each signal interval.

Samsung Exhibit 1015, Page 64 of 573

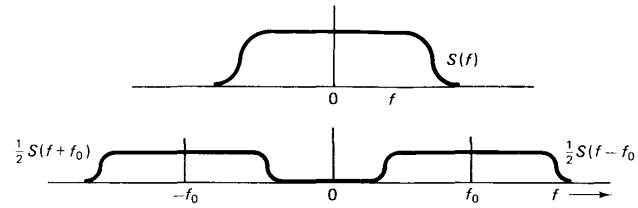


Figure 2.10 Amplitude modulation. The frequency characteristic of the waveform $s(t)$ is shifted up and down by f_0 in frequency.

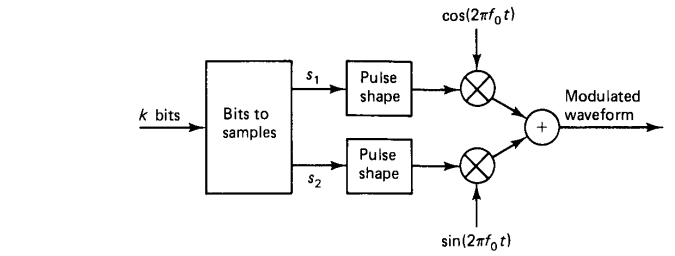
The high-frequency component, at $2f_0$, is then filtered out at the receiver, leaving the demodulated signal $s(t)/2$, which is then converted back to digital data. In practice, the incoming bits are mapped into shorter pulses than those of NRZ and then filtered somewhat to remove high-frequency components. It is interesting to note that Manchester coding can be viewed as AM in which the NRZ code is multiplied by a square wave. Although a square wave is not a sinusoid, it can be represented as a sinusoid plus odd harmonics; the harmonics serve no useful function and are often partly filtered out by the channel.

AM is rather sensitive to the receiver knowing the correct phase of the carrier. For example, if the modulated signal $s(t)\cos(2\pi f_0 t)$ were multiplied by $\sin(2\pi f_0 t)$, the low-frequency demodulated waveform would disappear. This suggests, however, the possibility of transmitting twice as many bits by a technique known as quadrature amplitude modulation (QAM). Here, the incoming bits are mapped into two baseband signals, $s_1(t)$ and $s_2(t)$. Then $s_1(t)$ is multiplied by $\cos(2\pi f_0 t)$ and $s_2(t)$ by $\sin(2\pi f_0 t)$; the sum of these products forms the transmitted QAM signal (see Fig. 2.11). The received waveform is separately multiplied by $\cos(2\pi f_0 t)$ and $\sin(2\pi f_0 t)$ The first multiplication (after filtering out the high-frequency component) yields $s_1(t)/2$, and the second yields $s_2(t)/2$.

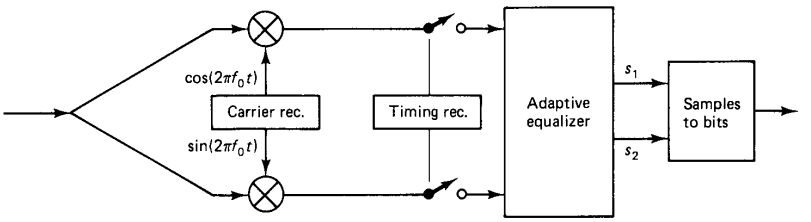
QAM is widely used in high-speed modems for voice-grade telephone channels. These channels have a useful bandwidth from about 500 to 2900 Hz (hertz, *i.e.*, cycles per second), and the carrier frequency is typically about 1700 Hz. The signals $s_1(t)$ and $s_2(t)$ are usually low-pass limited to 1200 Hz, thus allowing 2400 sample inputs per second for each baseband waveform. Adaptive equalization is usually used at the receiver (either before or after demodulation) to compensate for variations in the channel frequency response.

The next question is how to map the incoming bits into the low-pass waveforms $s_1(t)$ and $s_2(t)$. The pulse shape used in the mapping is not of central importance; it will be filtered anyway, and the equalizer will compensate for its shape automatically in attempting to eliminate intersymbol interference. Thus, what is important is mapping the input bits into sample amplitudes of $s_1(t)$ and $s_2(t)$. One could map one bit into a sample of s_1, mapping 1 into +1 and 0 into −1, and similarly map a second bit into a sample of s_2. This is shown in Fig. 2.12(a), viewed as a mapping from two bits into two amplitudes.

Samsung Exhibit 1015, Page 65 of 573



(a) Modulator



(b) Demodulator

Figure 2.11 Quadrature amplitude modulation. (a) Each sample period, k bits enter the modulator, are converted into quadrature amplitudes, are then modulated by sine and cosine functions, respectively, and are then added. (b) The reverse operations take place at the demodulator.

Similarly, for any given integer k, one can map k bits into two amplitudes. Each of the 2^k combinations of k bits map into a different amplitude pair. This set of 2^k amplitude pairs in a mapping is called a signal constellation (see Fig. 2.12 for a number of examples). The first two constellations in Fig. 2.12 are also referred to as phase-shift keying (PSK), since they can be viewed as simply changing the phase of the carrier rather than changing its amplitude. For voice-grade modems, where $W = 2400$ Hz, $k = 2$ [as in Fig. 2.12(a)] yields 4800 bits per second (bps), whereas $k = 4$ [as in Fig. 2.12(d)] yields 9600 bps.

To reduce the likelihood of noise causing one constellation point to be mistaken for another, it is desirable to place the signal points as far from each other as possible subject to a constraint on signal power (*i.e.*, on the mean-squared distance from origin to constellation point, taken over all constellation points). It can be shown that the constellation in Fig. 2.12(c) is preferable in this sense to that in Fig. 2.12(b) for $k = 3$.

Modems for voice-grade circuits are also available at rates of 14,400 and 19,200 bps. These use correspondingly larger signal constellations and typically also use error-correction coding. It is reasonable to ask at this point what kind of limit is imposed on data rate by channels subject to bandwidth constraints and noise. Shannon [Sha48] has

Samsung Exhibit 1015, Page 66 of 573

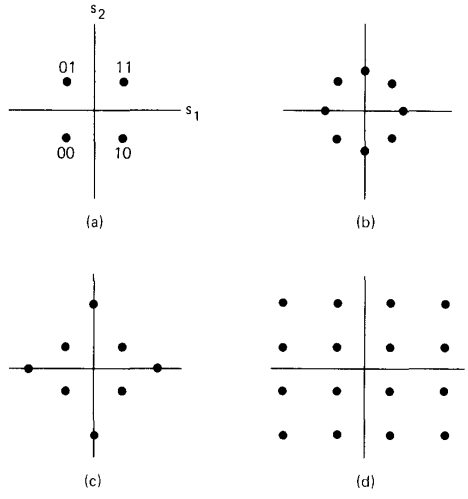


Figure 2.12 Signal constellations for QAM. Part (a) maps two binary digits into a quadrature amplitude sample. Parts (b) and (c) each map three binary digits, and part (d) maps four binary digits. Parts (a) and (b) also can be regarded as phase-shift keying.

shown that the capacity (*i.e.*, the maximum reliable data rate in bps) of such a channel is given by

$$C = W \log_2 \left(1 + \frac{S}{N_0 W}\right) \tag{2.13}$$

where W is the available bandwidth of the channel (essentially 2400 Hz for voice-grade telephone), S is the allowable signal power as seen by the receiver, and $N_0 W$ is the noise power within the bandwidth W seen by the receiver (*i.e.*, N_0 is the noise power per unit bandwidth, assumed uniformly distributed over W).

This is a deep and subtle result, and the machinery has not been developed here even to state it precisely, let alone to justify its validity. The description of QAM, however, at least provides some intuitive justification for the form of the result. The available sampling rate $1/T$ is equal to W by the sampling theorem [recall that W is twice the low-pass bandwidth of $s_1(t)$ and $s_2(t)$]. Since k bits are transmitted per sample, Wk bps can be sent. Next note that some minimum area must be provided around each point in the signal constellation to allow for noise, and this area is proportional to $N_0 W$. Also, the total area occupied by the entire constellation is proportional to the signal power S. Thus, 2^k, the number of points in the constellation, should be proportional to $S/(N_0 W)$, making k proportional to $\log_2[S/(N_0 W)]$.

Communication engineers usually express the signal-to-noise ratio [*i.e.*, $S/(N_0 W)$] in terms of decibels (dB), where the number of dB is defined as $10 \log_{10}[S/(N_0 W)]$. Thus, k, the number of bits per quadrature sample, is proportional to the signal-to-noise ratio in dB. The additional 1 in Eq. (2.13) is harder to explain; note, however, that it is required to keep the capacity positive when the signal-to-noise ratio is very small. The intuitive argument above would make it appear that the error probability depends on the spacing of points in the constellation and thus on k. The beauty of Shannon's theorem,

Samsung Exhibit 1015, Page 67 of 573

on the other hand, is that it asserts that with the use of error-correction coding, any rate less than C can be achieved with arbitrarily small error probability. The capacity of voice-grade telephone channels is generally estimated to be about 25,000 bps, indicating that the data rates of voice-grade modems are remarkably close to the theoretical limit.

High-speed modems generally maintain signal timing, carrier phase, and adaptive equalization by slowly updating these quantities from the received signal. Thus, it is essential for these modems to operate synchronously; each T seconds, k bits must enter the modem, and the DLC unit must provide idle fill when there are no data to send. For low-speed modems, on the other hand, it is permissible for the physical channel to become idle in between frames or even between individual characters. What is regarded as high speed here depends on the bandwidth of the channel. For modems on 3 kHz voice-grade channels, for example, data rates of 2400 bps or more are usually synchronous, whereas lower rates might be intermittent synchronous or character asynchronous. For a coaxial cable, on the other hand, the bandwidth is so large that intermittent synchronous data rates of 10 megabits per second (Mbps) or more can be used. As mentioned before, this is essential for multiaccess systems such as Ethernet.

2.2.6 Frequency- and Time-Division Multiplexing

In previous subsections, bandwidth constraints imposed by filtering on the physical channel and by noise have been discussed. Often, however, a physical channel is shared by multiple signals each constrained to a different portion of the available bandwidth; this is called frequency-division multiplexing (FDM). The most common examples of this are AM, FM, and TV broadcasting, in which each station uses a different frequency band. As another common example, voice-grade channels are often frequency multiplexed together in the telephone network. In these examples, each multiplexed signal is constrained to its own allocated frequency band to avoid interference (sometimes called crosstalk) with the other signals. The modulation techniques of the preceding subsection are equally applicable whether a bandwidth constraint is imposed by FDM or by channel filtering.

FDM can be viewed as a technique for splitting a big channel into many little channels. Suppose that a physical channel has a usable bandwidth of W hertz and we wish to split it into m equal FDM subchannels. Then, each subchannel has W/m hertz available, and thus W/m available quadrature samples per second for sending data (in practice, the bandwidth per subchannel must be somewhat smaller to provide guard bands between the subchannels, but that is ignored here for simplicity). In terms of Eq. (2.13), each subchannel is allotted $1/m$ of the overall available signal power and is subject to $1/m$ of the overall noise, so each subchannel has $1/m$ of the total channel capacity.

Time-division multiplexing (TDM) is an alternative technique for splitting a big channel into many little channels. Here, one modem would be used for the overall bandwidth W. Given m equal rate streams of binary data to be transmitted, the m bit streams would be multiplexed together into one bit stream. Typically, this is done by sending the data in successive frames. Each frame contains m slots, one for each bit stream to be multiplexed; a slot contains a fixed number of bits, sometimes 1, sometimes 8, and sometimes more. Typically, each frame also contains extra bits to help the

Samsung Exhibit 1015, Page 68 of 573

receiver maintain frame synchronization. For example, T1 carrier, which is in widespread telephone network use in the United States and Japan, multiplexes 24 data streams into slots of eight bits each with one extra bit per frame for synchronization. The overall data rate is 1.544 Mbps, with 64,000 bps for each of the multiplexed streams. In Europe, there is a similar system with 32 data streams and an overall data rate of 2.048 Mbps.

One way to look at FDM and TDM is that in each case one selects W quadrature samples per second (or more generally, $2W$ individual samples) for transmission; in one case, the samples are distributed in frequency and in the other they are distributed in time.

2.2.7 Other Channel Impairments

The effects of filtering and noise have been discussed in preceding subsections. There are a number of other possible impairments on physical channels. Sometimes there are multiple stages of modulation and demodulation internal to the physical channel. These can cause small amounts of phase jitter and carrier frequency offset in the received waveform. Amplification of the signals, within the physical channel and in the modems, can cause nonnegligible nonlinear distortion. Impulses of noise can occur due to lightning and switching effects. Repair personnel can short out or disconnect the channel for periods of time. Finally, there can be crosstalk from other frequency bands and from nearby wires.

To a certain extent, these effects can all be considered as extra noise sources. However, these noise sources have different statistical properties than the noise assumed by Shannon's theorem. (Technically, that noise is known as additive white Gaussian noise.) The principal difference is that the errors caused by these extra noise sources tend to occur in bursts of arbitrarily long length. As a result, one cannot, in practice, achieve the arbitrarily low error probabilities promised by Shannon's theorem. One must also use error detection and retransmission at the DLC layer; this is treated in Sections 2.3 and 2.4.

2.2.8 Digital Channels

In many cases, physical channels are designed to carry digital data directly, and the DLC unit can interface almost directly to the digital channel rather than to a modem mapping digital data into analog signals. To a certain extent, this is simply the question of whether the channel supplier or the channel user supplies the modem. This is an oversimplified view, however. A channel designed to carry digital data directly (such as the T1 carrier mentioned earlier) is often capable of achieving higher data rates at lower error probabilities than one carrying analog signals.

A major reason for this improved performance is the type of repeater used for digital channels. Repeaters are basically amplifiers inserted at various points in the propagation path to overcome attenuation. For a channel designed for analog signals, both the signal and the noise must be amplified at each repeater. Thus, the noise at the final receiver is an accumulation of noise over each stage of the path. For a digital

Samsung Exhibit 1015, Page 69 of 573

channel, on the other hand, the digital signal can be recovered at each repeater. This means that the noise does not accumulate from stage to stage, and an error occurs only if the noise in a single stage is sufficient to cause an error. In effect, the noise is largely suppressed at each stage. Because of this noise suppression, and also because of the low cost of digital processing, it is increasingly common to use digital channels (such as the T1 carrier system) for the transmission of analog signals such as voice. Analog signals are sampled (typically at 8000 samples per second for voice) and then quantized (typically at eight bits per sample) to provide a digital input for a digital channel.

The telephone network can be separated into two parts—the local loops, which go from subscribers to local offices, and the internal network connecting local offices, central offices, and toll switches. Over the years, the internal network has become mostly digital, using TDM systems such as the T1 carrier just described and a higher-speed carrier system called T3, at 44.736 Mbps, that will multiplex 28 T1 signals.

There are two consequences to this increasing digitization. First, the telephone companies are anxious to digitize the local loops, thus allowing 64 kilobit per second (kbps) digital service over a single voice circuit. An added benefit of digitizing the local loops is that more than a single voice circuit could be provided to each subscriber. Such an extension of the voice network in which both voice and data are simultaneously available in an integrated way is called an *integrated services digital network* (ISDN). The other consequence is that data network designers can now lease T1 lines (called DS1 service) and T3 lines (called DS3 service) at modest cost and thus construct networks with much higher link capacities than are available on older networks. These higher-speed links are making it possible for wide area networks to send data at speeds comparable to local area nets.

With the increasing use of optical fiber, still higher link speeds are being standardized. *SONET* (Synchronous Optical Network) is the name for a standard family of interfaces for high-speed optical links. These start at 51.84 Mbps (called STS-1), and have various higher speeds of the form n times 51.84 Mbps (called STS-n) for $n = 1, 3, 9, 12, 18, 24, 36, 48$. Like the T1 and T3 line speeds, each of these speeds has a 125 μs frame structure which can be broken down into a very large number of 64 kbps voice circuits, each with one byte per frame. What is ingenious and interesting about these standards is the way that the links carrying these frames can be joined together, in the presence of slight clock drifts, without losing or gaining bits for any of the voice circuits. What is relevant for our purposes, however, is that 1.5 and 45 Mbps link speeds are now economically available, and much higher speeds will be economically available in the future. The other relevant fact is that optical fibers tend to be almost noise-free, with error rates of less than 10^{-10}.

ISDN As outlined briefly above, an integrated services digital network (ISDN) is a telephone network in which both the internal network and local loops are digital. The "integrated service" part of the name refers to the inherent capability of such a network to carry both voice and data in an integrated fashion. A telephone network with analog voice lines for local loops also allows both data and voice to be carried, but not easily at the same time and certainly not with much speed or convenience. There has been

Samsung Exhibit 1015, Page 70 of 573

considerable effort to standardize the service provided by ISDN. Two standard types of service are available. The first, called *basic service*, provides two 64 kbps channels plus one 16 kbps channel to the user. Each 64 kbps channel can be used as an ordinary voice channel or as a point-to-point data channel. The 16 kbps channel is connected to the internal signaling network of the telephone system (*i.e.*, the network used internally for setting up and disconnecting calls, managing the network, etc.). Thus, the 16 kbps channel can be used to set up and control the two 64 kbps channels, but it could also be used for various low-data-rate services such as alarm and security systems.

In ISDN jargon, the 64 kbps channels are called B channels and the 16 kbps channel is called a D channel; the overall basic service is thus referred to as $2B + D$. With these facilities, a subscriber could conduct two telephone conversations simultaneously, or have one data session (at 64 kbps) plus one voice conversation. The latter capability appears to be rather useful to someone who works at home with a terminal but does not want to be cut off from incoming telephone calls. Most terminal users, of course, would be quite happy, after getting used to 300 bps terminals, to have the data capabilities of the 16 kbps D channel, and it is likely that subscribers will be offered a cheaper service consisting of just one B and one D channel.

The 64 kbps channels can be used in two ways: to set up a direct connection to some destination (thus using ISDN as a circuit switched network), and as an access line into a node of a packet switching network. On the one hand, a 64 kbps access into a packet switched network appears almost extravagant to one used to conventional networks. On the other hand, if one is trying to transfer a high-resolution graphics image of 10^9 bits, one must wait for over 4 hours using a 64 kbps access line. This illustrates a rather peculiar phenomenon. There are a very large number of applications of data networks that can be accomplished comfortably at very low data rates, but there are others (usually involving images) that require very high data rates.

Optical fiber will gradually be introduced into the local loops of the telephone network, and this will allow ISDN to operate at much higher data rates than those described above. Broadband ISDN (BISDN) is the name given to such high-speed ISDN networks. There has already been considerable standardization on broadband ISDN, including a standard user access rate of 155 Mbps (*i.e.*, the SONET STS-3 rate). These high data rates will allow for high-resolution TV as well as for fast image transmission, high-speed interconnection of supercomputers, and video conferencing. There are many interesting questions about how to build data networks handling such high data rates; one such strategy, called asynchronous transfer mode (ATM), is discussed briefly in Section 2.10.

The $2B + D$ basic service above is appropriate (or even lavish, depending on one's point of view) for the home or a very small office, but is inappropriate for larger offices used to using a PBX with a number of outgoing telephone lines. What ISDN (as proposed) offers here is something called *primary service* as opposed to basic service. This consists of 24 channels at 64 kbps each (in the United States and Japan) or 31 channels at 64 kbps each (in Europe). One of these channels is designated as the D channel and the others as B channels. The D channel is the one used for signaling and call setup and has a higher data rate here than in the basic service to handle the higher traffic levels. Subscribers can also obtain higher rate channels than the 64 kbps

Samsung Exhibit 1015, Page 71 of 573

B channels above. These higher rate channels are called H channels and come in 384, 1536, and 1920 kbps flavors. Such higher rate channels partially satisfy the need for the high data rate file transfers discussed above.

One of the interesting technical issues with ISDN is how to provide the required data rates over local loops consisting of a twisted wire pair. The approach being taken, for basic service, is to use time-division multiplexing of the B and D channels together with extra bits for synchronization and frequency shaping, thus avoiding a dc component. The most interesting part of this is that data must travel over the local loop in both directions, and that a node trying to receive the attenuated signal from the other end is frustrated by the higher-level signal being generated at the local end. For voice circuits, this problem has been solved in the past by a type of circuit called a hybrid which isolates the signal going out on the line from that coming in. At these high data rates, however, this circuit is not adequate and adaptive echo cancelers are required to get rid of the remaining echo from the local transmitted signal at the receiver.

2.2.9 Propagation Media for Physical Channels

The most common media for physical channels are twisted pair (*i.e.*, two wires twisted around each other so as to partially cancel out the effects of electromagnetic radiation from other sources), coaxial cable, optical fiber, radio, microwave, and satellite. For the first three, the propagated signal power decays exponentially with distance (*i.e.*, the attenuation in dB is linear with distance). Because of the attenuation, repeaters are used every few kilometers or so. The rate of attenuation varies with frequency, and thus as repeaters are spaced more closely, the useful frequency band increases, yielding a trade-off between data rate and the cost of repeaters. Despite this trade-off, it is helpful to have a ballpark estimate of typical data rates for channels using these media.

Twisted pair is widely used in the telephone network between subscribers and local stations and is increasingly used for data. One Mbps is a typical data rate for paths on the order of 1 km or less. Coaxial cable is widely used for local area networks, cable TV, and high-speed point-to-point links. Typical data rates are from 10 to several hundred Mbps. For optical fiber, data rates of 1000 Mbps or more are possible. Optical fiber is growing rapidly in importance, and the major problems lie in the generation, reception, amplification, and switching of such massive amounts of data.

Radio, microwave, and satellite channels use electromagnetic propagation in open space. The attenuation with distance is typically much slower than with wire channels, so repeaters can either be eliminated or spaced much farther apart than for wire lines. Frequencies below 1000 MHz are usually referred to as radio frequencies, and higher frequencies are referred to as microwave.

Radio frequencies are further divided at 30 MHz. Above 30 MHz, the ionosphere is transparent to electromagnetic waves, whereas below 30 MHz, the waves can be reflected by the ionosphere. Thus, above 30 MHz, propagation is on line-of-sight paths. The antennas for such propagation are frequently placed on towers or hills to increase the length of these line-of-sight paths, but the length of a path is still somewhat limited and repeaters are often necessary. This frequency range, from 30 to 1000 MHz, is used

Samsung Exhibit 1015, Page 72 of 573

for UHF and VHF TV broadcast, for FM broadcast, and many specialized applications. Packet radio networks, discussed in Section 4.6, use this frequency band. Typical data rates in this band are highly variable; the DARPA (U.S. Department of Defense Advanced Research Projects Agency) packet radio network, for example, uses 100,000 and 400,000 bps.

Below 30 MHz, long-distance propagation beyond line-of-sight is possible by reflection from the ionosphere. Ironically, the 3 to 30 MHz band is called the high-frequency (HF) band, the terminology coming from the early days of radio. This band is very noisy, heavily used (*e.g.*, by ham radio), and subject to fading. Fading can be viewed as a channel filtering phenomenon with the frequency response changing relatively rapidly in time; this is caused by time-varying multiple propagation paths from source to destination. Typical data rates in this band are 2400 bps and less.

Microwave links (above 1000 MHz) must use line-of-sight paths. The antennas (usually highly directional dishes) yield typical path lengths of 10 to 200 km. Longer paths than this can be achieved by the use of repeaters. These links can carry 1000 Mbps or so and are usually multiplexed between long-distance telephony, TV program distribution, and data.

Satellite links use microwave frequencies with a satellite as a repeater. They have similar data rates and uses as microwave links. One satellite repeater can receive signals from many ground stations and broadcast back in another frequency band to all those ground stations. The satellite can contain multiple antenna beams, allowing it to act as a switch for multiple microwave links. In Chapter 4, multiaccess techniques for sharing individual frequency bands between different ground stations are studied.

This section has provided a brief introduction to physical channels and their use in data transmission. A link in a subnet might use any of these physical channels, or might share such a channel on a TDM or FDM basis with many other uses. Despite the complexity of this subject (which we have barely touched), these links can be regarded simply as unreliable bit pipes by higher layers.

2.3 ERROR DETECTION

The subject of the next four sections is data link control. This section treats the detection of transmission errors, and the next section treats retransmission requests. Assume initially that the receiving data link control (DLC) module knows where frames begin and end. The problem then is to determine which of those frames contain errors. From the layering viewpoint, the packets entering the DLC are arbitrary bit strings (*i.e.*, the function of the DLC layer is to provide error-free packets to the next layer up, no matter what the packet bit strings are). Thus, at the receiving DLC, any bit string is acceptable as a packet and errors cannot be detected by analysis of the packet itself. Note that a transformation on packets of K bits into some other representation of length K cannot help; there are 2^K possible packets and all possible bit strings of length K must be used to represent all possible packets. The conclusion is that extra bits must be appended to a packet to detect errors.

2.3.1 Single Parity Checks

The simplest example of error detection is to append a single bit, called a *parity check*, to a string of data bits. This parity check bit has the value 1 if the number of 1's in the bit string is odd, and has the value 0 otherwise (see Fig. 2.13). In other words, the parity check bit is the sum, modulo 2, of the bit values in the original bit string (k modulo j, for integer k and positive integer j, is the integer m, $0 \leq m < j$, such that $k - m$ is divisible by j).

In the ASCII character code, characters are mapped into strings of seven bits and then a parity check is appended as an eighth bit. One can also visualize appending a parity check to the end of a packet, but it will soon be apparent that this is not a sufficiently reliable way to detect errors.

Note that the total number of 1's in an encoded string (*i.e.*, the original bit string plus the appended parity check) is always even. If an encoded string is transmitted and a single error occurs in transmission, then, whether a 1 is changed to a 0 or a 0 to a 1, the resulting number of 1's in the string is odd and the error can be detected at the receiver. Note that the receiver cannot tell which bit is in error, nor how many errors occurred; it simply knows that errors occurred because of the odd number of 1's.

It is rather remarkable that for bit strings of any length, a single parity check enables the detection of any single error in the encoded string. Unfortunately, two errors in an encoded string always leave the number of 1's even so that the errors cannot be detected. In general, any odd number of errors are detected and any even number are undetected.

Despite the appealing simplicity of the single parity check, it is inadequate for reliable detection of errors; in many situations, it only detects errors in about half of the encoded strings where errors occur. There are two reasons for this poor behavior. The first is that many modems map several bits into a single sample of the physical channel input (see Section 2.2.5), and an error in the reception of such a sample typically causes several bit errors. The second reason is that many kinds of noise, such as lightning and temporarily broken connections, cause long bursts of errors. For both these reasons, when one or more errors occur in an encoded string, an even number of errors is almost as likely as an odd number and a single parity check is ineffective.

2.3.2 Horizontal and Vertical Parity Checks

Another simple and intuitive approach to error detection is to arrange a string of data bits in a two-dimensional array (see Fig. 2.14) with one parity check for each row and one for each column. The parity check in the lower right corner can be viewed as a parity check on the row parity checks, on the column parity checks, or on the data array. If an even number of errors are confined to a single row, each of them can be detected by the

s_1	s_2	s_3	s_4	s_5	s_6	s_7	c
1	0	1	1	0	0	0	1

Figure 2.13 Single parity check. Final bit c is the modulo 2 sum of s_1 to s_k, where $k = 7$ here.

Samsung Exhibit 1015, Page 74 of 573

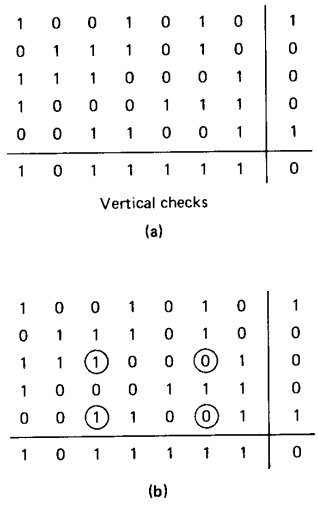


Figure 2.14 Horizontal and vertical parity checks. Each horizontal parity check checks its own row, and each column parity check checks its own column. Note, in part (b), that if each circled bit is changed, all parity checks are still satisfied.

corresponding column parity checks; similarly, errors in a single column can be detected by the row parity checks. Unfortunately, any pattern of four errors confined to two rows and two columns [*i.e.*, forming a rectangle as indicated in Fig. 2.14(b)] is undetectable.

The most common use of this scheme is where the input is a string of ASCII encoded characters. Each encoded character can be visualized as a row in the array of Fig. 2.14; the row parity check is then simply the last bit of the encoded character. The column parity checks can be trivially computed by software or hardware. The major weakness of this scheme is that it can fail to detect rather short bursts of errors (*e.g.*, two adjacent bits in each of two adjacent rows). Since adjacent errors are quite likely in practice, the likelihood of such failures is undesirably high.

2.3.3 Parity Check Codes

The nicest feature about horizontal and vertical parity checks is that the underlying idea generalizes immediately to arbitrary parity check codes. The underlying idea is to start with a bit string (the array of data bits in Fig. 2.14) and to generate parity checks on various subsets of the bits (the rows and columns in Fig. 2.14). The transformation from the string of data bits to the string of data bits and parity checks is called a *parity check code* or *linear code*. An example of a parity check code (other than the horizontal and vertical case) is given in Fig. 2.15. A parity check code is defined by the particular collection of subsets used to generate parity checks. Note that the word *code* refers to the transformation itself; we refer to an encoded bit string (data plus parity checks) as a *code word*.

Let K be the length of the data string for a given parity check code and let L be the number of parity checks. For the frame structure in Fig. 2.2, one can view the data

Samsung Exhibit 1015, Page 75 of 573

s_1	s_2	s_3	c_1	c_2	c_3	c_4	
1	0	0	1	1	1	0	
0	1	0	0	1	1	1	$c_1 = s_1 + s_3$
0	0	1	1	1	0	1	$c_2 = s_1 + s_2 + s_3$
1	1	0	1	0	0	1	$c_3 = s_1 + s_2$
1	0	1	0	0	1	1	$c_4 = s_2 + s_3$
1	1	1	0	1	0	0	
0	0	0	0	0	0	0	
0	1	1	1	0	1	0	

Figure 2.15 Example of a parity check code. Code words are listed on the left, and the rule for generating the parity checks is given on the right.

string as the header and packet, and view the set of parity checks as the trailer. Note that it is important to detect errors in the control bits of the header as well as to detect errors in the packets themselves. Thus, $K + L$ is the frame length, which for the present is regarded as fixed. For a given code, each of the possible 2^K data strings of length K is mapped into a frame (*i.e.*, code word) of length $K + L$. In an error-detection system, the frame is transmitted and the receiving DLC module determines if each of the parity checks is still the modulo 2 sum of the corresponding subset of data bits. If so, the frame is regarded by the receiver as error-free, and if not, the presence of errors is detected. If errors on the link convert one code word into another, the frame is regarded by the receiver as error-free, and undetectable errors are said to have occurred in the frame.

Given any particular code, one would like to be able to predict the probability of undetectable errors in a frame for a particular link. Unfortunately, this is very difficult. First, errors on links tend to be dependent and to occur in bursts; there are no good models for the length or intensity of these bursts, which vary widely among links of the same type. Second, for any reasonable code, the frequency of undetectable errors is very small and is thus both difficult to measure experimentally and dependent on rare, difficult-to-model events. The literature contains many calculations of the probability of undetectable errors, but these calculations are usually based on the assumption of independent errors; the results are usually orders of magnitude away from reality.

As a result of these difficulties, the effectiveness of a code for error detection is usually measured by three parameters: (1) the minimum distance of the code, (2) the burst-detecting capability, and (3) the probability that a completely random string will be accepted as error-free. The minimum distance of a code is defined as the smallest number of errors that can convert one code word into another. As we have seen, the minimum distance of a code using a single parity check is 2, and the minimum distance of a code with horizontal and vertical parity checks is 4.

The length of a burst of errors in a frame is the number of bits from the first error to the last, inclusive. The burst-detecting capability of a code is defined as the largest integer B such that a code can detect all bursts of length B or less. The burst-detecting capability of the single parity check code is 1, whereas the burst-detecting capability of

Samsung Exhibit 1015, Page 76 of 573

a code with horizontal and vertical parity checks is 1 plus the length of a row (assuming that rows are sent one after the other).

By a completely random string of length $K + L$ is meant that each such string is received with probability 2^{-K-L}. Since there are 2^K code words, the probability of an undetected error is the probability that the random string is one of the code words; this occurs with probability 2^{-L} (the possibility that the received random string happens to be the same as the transmitted frame is ignored). This is usually a good estimate of the probability of undetectable errors given that both the minimum distance and the burst-detecting capability of the code are greatly exceeded by the set of errors on a received frame.

Parity check codes can be used for error correction rather than just for error detection. For example, with horizontal and vertical parity checks, any single error can be corrected simply by finding the row and column with odd parity. It is shown in Problem 2.10 that a code with minimum distance d can be used to correct any combination of fewer than $d/2$ errors. Parity check codes (and convolutional codes, which are closely related but lack the frame structure) are widely used for error correction at the physical layer. The modern approach to error correction is to view it as part of the modulation and demodulation process, with the objective of creating a virtual bit pipe with relatively low error rate.

Error correction is generally not used at the DLC layer, since the performance of an error correction code is heavily dependent on the physical characteristics of the channel. One needs error detection at the DLC layer, however, to detect rare residual errors from long noisy periods.

2.3.4 Cyclic Redundancy Checks

The parity check codes used for error detection in most DLCs today are cyclic redundancy check (CRC) codes. The parity check bits are called the CRC. Again, let L be the length of the CRC (*i.e.*, the number of check bits) and let K be the length of the string of data bits (*i.e.*, the header and packet of a frame). It is convenient to denote the data bits as $s_{K-1}, s_{K-2}, \ldots, s_1, s_0$, and to represent the string as a polynomial $s(D)$ with coefficients $s_{K-1}, \ldots, s_0$,

$$s(D) = s_{K-1}D^{K-1} + s_{K-2}D^{K-2} + \cdots + s_0 \tag{2.14}$$

The powers of the indeterminate D can be thought of as keeping track of which bit is which; high-order terms are viewed as being transmitted first. The CRC is represented as another polynomial,

$$c(D) = c_{L-1}D^{L-1} + \cdots + c_1 D + c_0 \tag{2.15}$$

The entire frame of transmitted information and CRC can then be represented as $x(D) = s(D)D^L + c(D)$, that is, as

$$x(D) = s_{K-1}D^{L+K-1} + \cdots + s_0 D^L + c_{L-1}D^{L-1} + \cdots + c_0 \tag{2.16}$$

Samsung Exhibit 1015, Page 77 of 573

The CRC polynomial $c(D)$ is a function of the information polynomial $s(D)$, defined in terms of a *generator polynomial* $g(D)$; this is a polynomial of degree L with binary coefficients that specifies the particular CRC code to be used.

$$g(D) = D^L + g_{L-1}D^{L-1} + \cdots + g_1 D + 1 \tag{2.17}$$

For a given $g(D)$, the mapping from the information polynomial to the CRC polynomial $c(D)$ is given by

$$c(D) = \text{ Remainder } \left[\frac{s(D)D^L}{g(D)}\right] \tag{2.18}$$

The polynomial division above is just ordinary long division of one polynomial by another, except that the coefficients are restricted to be binary and the arithmetic on coefficients is performed modulo 2. Thus, for example, $(1+1)$ modulo $2 = 0$ and $(0-1)$ modulo $2 = 1$. Note that subtraction using modulo 2 arithmetic is the same as addition. As an example of the operation in Eq. (2.18),

$$\begin{array}{r} D^2 + D \\ D^3 + D^2 + 1 \overline{)\, D^5 + \quad\; D^3} \\ \underline{D^5 + D^4 + \quad\; D^2} \\ D^4 + D^3 + D^2 \\ \underline{D^4 + D^3 + \quad\; D} \\ D^2 + D = \text{ Remainder} \end{array}$$

Since $g(D)$ is a polynomial of degree at most L, the remainder is of degree at most $L-1$. If the degree of $c(D)$ is less than $L-1$, the corresponding leading coefficients $c_{L-1}, \ldots$, in Eq. (2.18) are taken as zero.

This long division can be implemented easily in hardware by the feedback shift register circuit shown in Fig. 2.16. By comparing the circuit with the long division above, it can be seen that the successive contents of the shift register cells are just the coefficients of the partial remainders in the long division. In practice, CRCs are usually calculated by VLSI chips, which often perform the other functions of DLC as well.

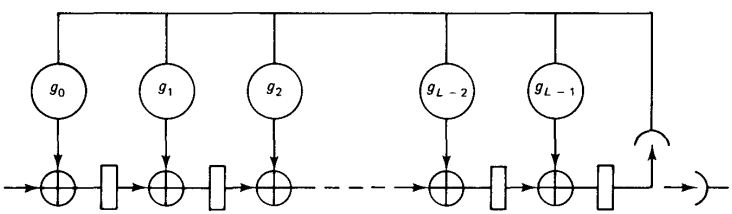


Figure 2.16 Shift register circuit for dividing polynomials and finding the remainder. Each rectangle indicates a storage cell for a single bit and the preceding circles denote modulo 2 adders. The large circles at the top indicate multiplication by the value of g_i. Initially, the register is loaded with the first L bits of $s(D)$ with s_{K-1} at the right. On each clock pulse, a new bit of $s(D)$ comes in at the left and the register reads in the corresponding modulo 2 sum of feedback plus the contents of the previous stage. After K shifts, the switch at the right moves to the horizontal position and the CRC is read out.

Samsung Exhibit 1015, Page 78 of 573

Let $z(D)$ be the quotient resulting from dividing $s(D)D^L$ by $g(D)$. Then, $c(D)$ can be represented as

$$s(D)D^L = g(D)z(D) + c(D) \tag{2.19}$$

Subtracting $c(D)$ (modulo 2) from both sides of this equation and recognizing that modulo 2 subtraction and addition are the same, we obtain

$$x(D) = s(D)D^L + c(D) = g(D)z(D) \tag{2.20}$$

Thus, all code words are divisible by $g(D)$, and all polynomials divisible by $g(D)$ are code words. It has not yet been shown that the mapping from $s(D)$ to $x(D)$ corresponds to a parity check code. This is demonstrated in Problem 2.15 but is not necessary for the subsequent development.

Now suppose that $x(D)$ is transmitted and that the received sequence is represented by a polynomial $y(D)$, where $x(D)$ and $y(D)$ differ because of the errors on the communication link. If the error sequence is represented as a polynomial $e(D)$, then $y(D) = x(D) + e(D)$, where, as throughout this section, + means modulo 2 addition; each error in the frame corresponds to a nonzero coefficient in $e(D)$ [*i.e.*, a coefficient in which $y(D)$ and $x(D)$ differ]. At the receiver, Remainder$[y(D)/g(D)]$ can be calculated by essentially the same circuit as that above. Since it has been shown that $x(D)$ is divisible by $g(D)$,

$$\text{Remainder}\left[\frac{y(D)}{g(D)}\right] = \text{Remainder}\left[\frac{e(D)}{g(D)}\right] \tag{2.21}$$

If no errors occur, then $e(D) = 0$ and the remainder above will be 0. The rule followed by the receiver is to decide that the frame is error- free if this remainder is 0 and to decide that there are errors otherwise. When errors actually occur [*i.e.*, $e(D) \neq 0$], the receiver fails to detect the errors only if this remainder is 0; this occurs only if $e(D)$ is itself some code word. In other words, $e(D) \neq 0$ is undetectable if and only if

$$e(D) = g(D)z(D) \tag{2.22}$$

for some nonzero polynomial $z(D)$. We now explore the conditions under which undetected errors can occur.

First, suppose that a single error occurs, say $e_i = 1$, so that $e(D) = D^i$. Since $g(D)$ has at least two nonzero terms (*i.e.*, D^L and 1), $g(D)z(D)$ must also have at least two nonzero terms for any nonzero $z(D)$ (see Problem 2.13). Thus $g(D)z(D)$ cannot equal D^i; since this is true for all i, all single errors are detectable. By the same type of argument, since the highest-order and lowest-order terms in $g(D)$ (*i.e.*, D^L and 1, respectively) differ by L, the highest-order and lowest-order terms in $g(D)z(D)$ differ by at least L for all nonzero $z(D)$. Thus, if $e(D)$ is a code word, the burst length of the errors is at least $L + 1$ (the +1 arises from the definition of burst length as the number of positions from the first error to the last error *inclusive*).

Next, suppose that a double error occurs, say in positions i and j, so that

$$e(D) = D^i + D^j = D^j(D^{i-j} + 1), \quad i > j \tag{2.23}$$

Samsung Exhibit 1015, Page 79 of 573

From the argument above, D^j is not divisible by $g(D)$ or by any factor of $g(D)$; thus, $e(D)$ fails to be detected only if $D^{i-j}+1$ is divisible by $g(D)$. For any binary polynomial $g(D)$ of degree L, there is some smallest n for which $D^n + 1$ is divisible by $g(D)$. It is known from the theory of finite fields that this smallest n can be no larger than $2^L - 1$; moreover, for all $L > 0$, there are special L-degree polynomials, called *primitive polynomials*, such that this smallest n is equal to $2^L - 1$. Thus, if $g(D)$ is chosen to be such a primitive polynomial of degree L, and if the frame length is restricted to be at most $2^L - 1$, then $D^{i-j} + 1$ cannot be divisible by $g(D)$; thus, all double errors are detected.

In practice, the generator polynomial $g(D)$ is usually chosen to be the product of a primitive polynomial of degree $L-1$ times the polynomial $D+1$. It is shown in Problem 2.14 that a polynomial $e(D)$ is divisible by $D + 1$ if and only if $e(D)$ contains an even number of nonzero coefficients. This ensures that all odd numbers of errors are detected, and the primitive polynomial ensures that all double errors are detected (as long as the block length is less than 2^{L-1}). Thus, any code of this form has a minimum distance of at least 4, a burst-detecting capability of at least L, and a probability of failing to detect errors in completely random strings of 2^{-L}. There are two standard CRCs with length $L = 16$ (denoted CRC-16 and CRC-CCITT). Each of these CRCs is the product of $D+1$ times a primitive $(L-1)$- degree polynomial, and thus both have the foregoing properties. There is also a standard CRC with $L = 32$. It is a 32-degree primitive polynomial, and has been shown to have a minimum distance of 5 for block lengths less than 3007 and 4 for block lengths less than 12,145 [FKL86]. These polynomials are as follows:

$$g(D) = D^{16} + D^{15} + D^2 + 1 \qquad \text{for CRC-16}$$

$$g(D = D^{16} + D^{12} + D^5 + 1 \qquad \text{for CRC-CCITT}$$

$$g(D) = D^{32} + D^{26} + D^{23} + D^{22} + D^{16} + D^{12} + D^{11} + D^{10} + D^8 + D^7 + D^5 + D^4 + D^2 + D^1 + 1$$

2.4 ARQ: RETRANSMISSION STRATEGIES

The general concept of *a*utomatic *r*epeat re*q*uest (ARQ) is to detect frames with errors at the receiving DLC module and then to request the transmitting DLC module to repeat the information in those erroneous frames. Error detection was discussed in the preceding section, and the problem of requesting retransmissions is treated in this section. There are two quite different aspects of retransmission algorithms or protocols. The first is that of correctness: Does the protocol succeed in releasing each packet, once and only once, without errors, from the receiving DLC? The second is that of efficiency: How much of the bit-transmitting capability of the bit pipe is wasted by unnecessary waiting and by sending unnecessary retransmissions? First, several classes of protocols are developed and shown to be correct (in a sense to be defined more precisely later). Later, the effect that the various parameters in these classes have on efficiency is considered.

Samsung Exhibit 1015, Page 80 of 573

Recall from Fig. 2.2 that packets enter the DLC layer from the network layer. The DLC module appends a header and trailer to each packet to form a frame, and the frames are transmitted on the virtual bit pipe (*i.e.*, are sent to the physical layer for transmission). When errors are detected in a frame, a new frame containing the old packet is transmitted. Thus, the first transmitted frame might contain the first packet, the next frame the second packet, the third frame a repetition of the first packet, and so forth. When a packet is repeated, the frame header and trailer might or might not be the same as in the earlier version.

Since framing will not be discussed until the next section, we continue to assume that the receiving DLC knows when frames start and end; thus a CRC (or any other technique) may be used for detecting errors. We also assume, somewhat unrealistically, that *all* frames containing transmission errors are detected. The reason for this is that we want to prove that ARQ works correctly except when errors are undetected. This is the best that can be hoped for, since error detection cannot work with perfect reliability and bounded delay; in particular, any code word can be changed into another code word by some string of transmission errors. This can cause erroneous data to leave the DLC or, perhaps worse, can cause some control bits to be changed. In what follows, we refer to frames without transmission errors as *error-free frames* and those with transmission errors as *error frames*. We are assuming, then, that the receiver can always disinguish error frames from error-free frames.

Finally, we need some assumptions about the bit pipes over which these frames are traveling. The reason for these assumptions will be clearer when framing is studied; in effect, these assumptions will allow us to relax the assumption that the receiving DLC has framing information. Assume first that each transmitted frame is delayed by an arbitrary and variable time before arriving at the receiver, and assume that some frames might be "lost" and never arrive. Those frames that arrive, however, are assumed to arrive in the same order as transmitted, with or without errors. Figure 2.17 illustrates this behavior.

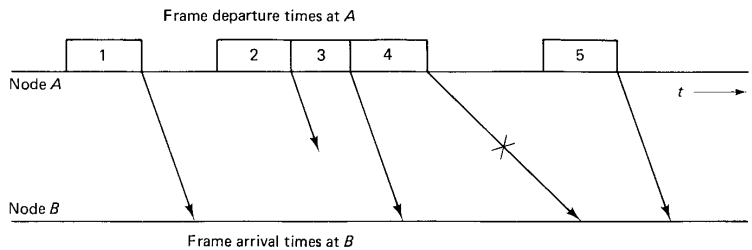


Figure 2.17 Model of frame transmissions: frame 2 is lost and never arrives; frame 4 contains errors; the frames have variable transmission delay, but those that arrive do so in the order transmitted. The rectangles at the top of the figure indicate the duration over which the frame is being transmitted. Each arrow indicates the propagation of a frame from A to B, and the arrowhead indicates the time at which the frame is completely received; at this time the CRC can be recomputed and the frame accepted or not.

Samsung Exhibit 1015, Page 81 of 573

2.4.1 Stop-and-Wait ARQ

The simplest type of retransmission protocol is called *stop-and-wait*. The basic idea is to ensure that each packet has been received correctly before initiating transmission of the next packet. Thus, in transmitting packets from point A to B, the first packet is transmitted in the first frame, and then the sending DLC waits. If the frame is error-free, B sends an acknowledgment (called an ack) back to A; if the frame is an error frame, B sends a negative acknowledgment (called a nak) back to A. Since errors can occur from B to A as well as from A to B, the ack or nak is protected with a CRC.

If an error-free frame is received at B, and the corresponding ack frame to A is error-free, then A can start to send the next packet in a new frame. Alternatively, detected errors can occur either in the transmission of the frame or the return ack or nak, and in either case A resends the old packet in a new frame. Finally, if either the frame or the return ack or nak is lost, A must eventually time-out and resend the old packet.

Figure 2.18 illustrates a potential malfunction in such a strategy. Since delays are arbitrary, it is possible for node A to time-out and resend a packet when the first transmission and/or the corresponding ack is abnormally delayed. If B receives both transmissions of the given packet correctly, B has no way of knowing whether the second transmission is a new packet or a repetition of the old packet. One might think that B could simply compare the packets to resolve this issue, but as far as the DLC layer is concerned, packets are arbitrary bit strings and the first and second packets could be identical; it would be a violation of the principle of layering for the DLC layer to rely on higher layers to ensure that successive packets are different.

The simplest solution to this problem is for the sending DLC module (at A) to use a sequence number in the frame header to identify successive packets. Unfortunately, even the use of sequence numbers from A to B is not quite enough to ensure correct operation. The problem is that acks can get lost on the return channel, and thus when B gets the same packet correctly twice in a row, it has to send a new ack for the second reception (see Fig. 2.19). After transmitting the packet twice but receiving only one ack,

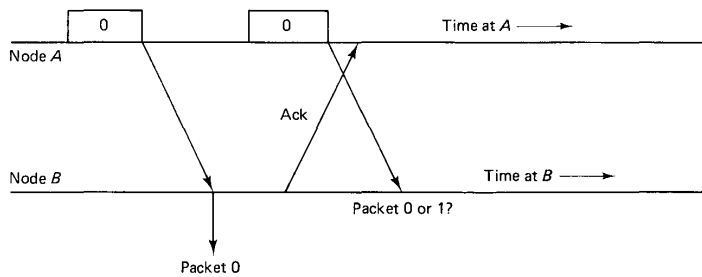


Figure 2.18 The trouble with unnumbered packets. If the transmitter at A times-out and sends packet 0 twice, the receiver at B cannot tell whether the second frame is a retransmission of packet 0 or the first transmission of packet 1.

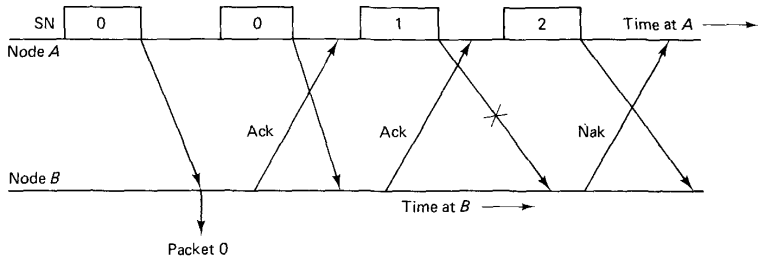


Figure 2.19 The trouble with unnumbered acks. If the transmitter at A times-out and sends packet 0 twice, node B can use the sequence numbers to recognize that packet 0 is being repeated. It must send an ack for both copies, however, and (since acks can be lost) the transmitter cannot tell whether the second ack is for packet 0 or 1.

node A could transmit the next packet in sequence, and then on receiving the second ack, could interpret that as an ack for the new packet, leading to a potential failure of the system.

To avoid this type of problem, the receiving DLC (at B), instead of returning ack or nak on the reverse link, returns the number of the next packet awaited. This provides all the information of the ack/nak, but avoids ambiguities about which frame is being acked. An equivalent convention would be to return the number of the packet just accepted, but this is not customary. Node B can request this next awaited packet upon the receipt of each packet, at periodic intervals, or at an arbitrary selection of times. In many applications, there is another stream of data from B to A, and in this case, the frames from B to A carrying requests for new A to B packets must be interspersed with data frames carrying data from B to A. It is also possible to "piggyback" these requests for new packets into the headers of the data frames from B to A (see Fig. 2.20), but as shown in Problem 2.38, this is often counterproductive for stop-and-wait ARQ. Aside from its effect on the timing of the requests from node B to A, the traffic from B to A does not affect the stop-and-wait strategy from A to B; thus, in what follows, we ignore this reverse traffic (except for the recognition that requests might be delayed). Figure 2.21 illustrates the flow of data from A to B and the flow of requests in the opposite direction.

We next specify the stop-and-wait strategy more precisely and then show that it works correctly. The correctness might appear to be self-evident, but the methodology of the demonstration will be helpful in understanding subsequent distributed algorithms. It

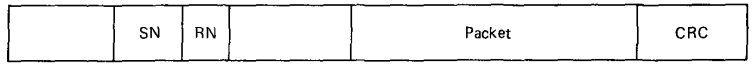


Figure 2.20 The header of a frame contains a field carrying the sequence number, SN, of the packet being transmitted. If piggybacking is being used, it also contains a field carrying the request number, RN, of the next packet awaited in the opposite direction.

Samsung Exhibit 1015, Page 83 of 573

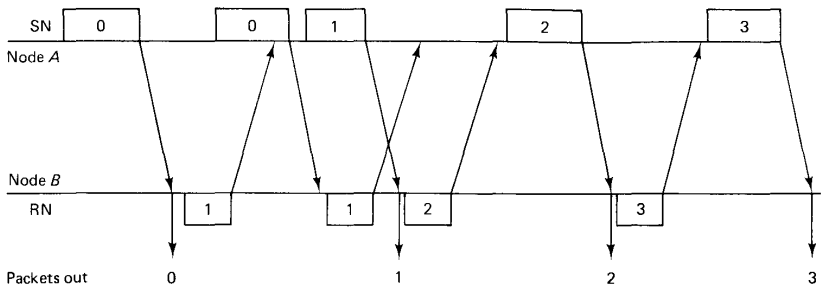


Figure 2.21 Example of use of sequence and request numbers for stop-and-wait transmission from A to B. Note that packet 0 gets repeated, presumably because node A times-out too soon. Note also that node A delays repeating packet 1 on the second request for it. This has no effect on the correctness of the protocol, but avoids unnecessary retransmissions.

will be seen that what is specified is not a single algorithm, but rather a class of algorithms in which the timing of frame transmissions is left unspecified. Showing that all algorithms in such a class are correct then allows one to specify the timing as a compromise between simplicity and efficiency without worrying further about correctness.

Assume that when the strategy is first started on the link, nodes A and B are correctly initialized in the sense that no frames are in transit on the link and that the receiver at B is looking for a frame with the same sequence number as the first frame to be transmitted from A. It makes no difference what this initial sequence number SN is as long as A and B agree on it, so we assume that $SN = 0$, since this is the conventional initial value.

The algorithm at node A for A-to-B transmission:

1. Set the integer variable SN to 0.
2. Accept a packet from the next higher layer at A; if no packet is available, wait until it is; assign number SN to the new packet.
3. Transmit the SNth packet in a frame containing SN in the sequence number field.
4. If an error-free frame is received from B containing a request number RN greater than SN, increase SN to RN and go to step 2. If no such frame is received within some finite delay, go to step 3.

The algorithm at node B for A-to-B transmission:

1. Set the integer variable RN to 0 and then repeat steps 2 and 3 forever.
2. Whenever an error-free frame is received from A containing a sequence number SN equal to RN, release the received packet to the higher layer and increment RN.
3. At arbitrary times, but within bounded delay after receiving any error-free data frame from A, transmit a frame to A containing RN in the request number field.

Samsung Exhibit 1015, Page 84 of 573

There are a number of conventional ways to handle the arbitrary delays between subsequent transmissions in the algorithm above. The usual procedure for node A (recall that we are discussing only the stop-and-wait strategy for A-to-B traffic) is to set a timer when a frame transmission begins. If the timer expires before receipt of a request for the next packet from B, the timer is reset and the packet is resent in a new frame. If a request for a new packet is received from B before the timer expires, the timer is disabled until A transmits the next packet. The same type of timer control can be used at node B. Alternatively, node B could send a frame containing a request for the awaited packet each time that it receives a frame from A. Also, if B is piggybacking its request numbers on data frames, it could simply send the current value of *RN* in each such frame. Note that this is not sufficient in itself, since communication from A to B would then halt in the absence of traffic from B to A; thus, node B must send nondata frames containing the *RN* values when there is no B-to-A data traffic. The important point is that whatever timing strategy is being used, it must guarantee that the intervening interval between repetitions in each direction is bounded.

Correctness of stop and wait We now go through an informal proof that this class of algorithms is correct in the sense that a never-ending stream of packets can be accepted from the higher layer at A and delivered to the higher layer at B in order and without repetitions or deletions. We continue to assume that all error frames are detected by the CRC. We also assume that there is some $q > 0$ such that each frame is received error-free with probability at least q. Finally, we recall the assumption that the link is initially empty, that the first packet from A has $SN = 0$, and that node B is initially awaiting the packet with $SN = 0$.

Proofs of this type usually break into two parts, characterized as *safety* and *liveness*. An algorithm is safe if it never produces an incorrect result, which in this case means never releasing a packet out of the correct order to the higher layer at B. An algorithm is live if it can continue forever to produce results (*i.e.*, if it can never enter a deadlock condition from which no further progress is possible). In this case liveness means the capability to continue forever to accept new packets at A and release them at B.

The safety property is self-evident for this algorithm; that is, node B is initially awaiting packet 0, and the only packet that can be released is packet 0. Subsequently (using induction if one wants to be formal), node B has released all the packets in order, up to, but not including, the current value of RN; packet RN is the only packet that it can next accept and release. When an error-free frame containing packet RN is received and released, the value of RN is incremented and the situation above is repeated with the new value of RN.

To see that the liveness property is satisfied, assume that node A first starts to transmit a given packet i at time t_1 (see Fig. 2.22). Let t_2 be the time at which this packet is received error-free and released to the higher layer at node B; let $t_2 = \infty$ if this event never occurs. Similarly, let t_3 be the time at which the sequence number at A is increased to $i + 1$, and let $t_3 = \infty$ if this never occurs. We will show that $t_1 < t_2 < t_3$ and that t_3 is finite. This is sufficient to demonstrate liveness, since using the argument repeatedly for $i = 0$, then $i = 1$, and so on, shows that each packet is transmitted with

Samsung Exhibit 1015, Page 85 of 573

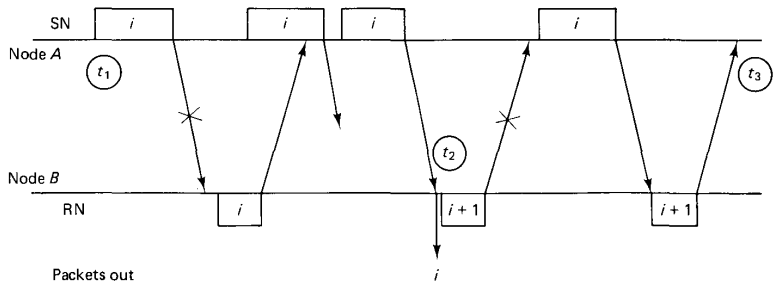


Figure 2.22 Times t_1, t_2, and t_3 when a packet is first placed in a frame for transmission at A, first received and released to the higher layer at B, and first acknowledged at A.

finite delay. Note that we cannot guarantee that the higher layer at A will always supply packets within finite delay, so that the notion of liveness here can only require finite delay given available packets to send.

Let $RN(t)$ be the value of the variable RN at node B as a function of time t and let $SN(t)$ be the corresponding value of SN at node A. It is seen directly from the algorithm statement that $SN(t)$ and $RN(t)$ are nondecreasing in t. Also, since $SN(t)$ is the largest request number received from B up to time t, $SN(t) \leq RN(t)$. By assumption, packet i has never been transmitted before time t_1, so (using the safety property) $RN(t_1) \leq i$. Since $SN(t_1) = i$, it follows that $SN(t_1) = RN(t_1) = i$. By definition of t_2 and t_3, $RN(t)$ is incremented to $i + 1$ at t_2 and $SN(t)$ is incremented to $i + 1$ at t_3. Using the fact that $SN(t) \leq RN(t)$, it follows that $t_2 < t_3$.

We have seen that node A transmits packet i repeatedly, with finite delay between successive transmissions, from t_1 until it is first received error-free at t_2. Since there is a probability $q > 0$ that each retransmission is received correctly, and retransmissions occur within finite intervals, an error-free reception eventually occurs and t_2 is finite. Node B then transmits frames carrying $RN = i+1$ from time t_2 until received error-free at time t_3. Since node A is also transmitting frames in this interval, the delay between subsequent transmissions from B is finite, and, since $q > 0$, t_3 eventually occurs; thus the interval from t_1 to t_3 is finite and the algorithm is live.

One trouble with the stop-and-wait strategy developed above is that the sequence and request numbers become arbitrarily large with increasing time. Although one could simply use a very large field for sending these numbers, it is nicer to send these numbers modulo some integer. Given our assumption that frames travel in order on the link, it turns out that a modulus of 2 is sufficient.

To understand why it is sufficient to send sequence numbers modulo 2, we first look more carefully at what happens in Fig. 2.22 when ordinary integers are used. Note that after node B receives packet i at time t_2, the subsequent frames received at B must all have sequence numbers i or greater (since frames sent before t_1 cannot arrive after t_2). Similarly, while B is waiting for packet $i + 1$, no packet greater than $i + 1$ can be sent [since $SN(t) \leq RN(t)$]. Thus, in the interval while $RN(t) = i + 1$, the received

Samsung Exhibit 1015, Page 86 of 573

frames all carry sequence numbers equal to i or $i + 1$. Sending the sequence number modulo 2 is sufficient to resolve this ambiguity. The same argument applies for all i. By the same argument, in the interval t_1 to t_3, while $SN(t)$ is equal to i, the request numbers received at A must be either i or $i + 1$, so again sending *RN* modulo 2 is sufficient to resolve the ambiguity. Finally, since *SN* and *RN* need be transmitted only modulo 2, it is sufficient to keep track of them at the nodes only modulo 2.

Using modulo 2 values for *SN* and *RN*, we can view nodes A and B as each having two states (for purposes of A to B traffic), corresponding to the binary value of SN at node A and RN at node B. Thus, A starts in state 0; a transition to state 1 occurs upon receipt of an error-free request for packet 1 modulo 2. Note that A has to keep track of more information than just this state, such as the contents of the current packet and the time until time-out, but the binary state above is all that is of concern here.

Node B similarly is regarded as having two possible states, 0 and 1, corresponding to the number modulo 2 of the packet currently awaited. When a packet of the desired number modulo 2 is received, the DLC at B releases that packet to the higher layer and changes state, awaiting the next packet (see Fig. 2.23). The combined state of A and B is then initially (0,0); when the first packet is received error-free, the state of B changes to 1, yielding a combined state (0, 1). When A receives the new *RN* value (*i.e.*, 1), the state of A changes to 1 and the combined state becomes (1,1). Note that there is a fixed sequence for these combined states, (0,0), (0,1), (1,1), (1,0), (0,0), and so on, and that A and B alternate in changing states. It is interesting that at the instant of the transition from (0,0) to (0,1), B knows the combined state, but subsequently, up until the transition later from (1,1) to (1,0), it does not know the combined state (*i.e.*, B never knows that A has received the ack information until the next packet is received). Similarly, A knows the combined state at the instant of the transition from (0,1) to (1,1) and of the transition from (1,0) to (0,0). The combined state is always unknown to either A or B, and is frequently unknown to both. The situation here is very similar to that in the three-army problem discussed in Section 1.4. Here, however, information is transmitted even though the combined state information is never jointly known, whereas in the three-army problem, it is impossible to coordinate an attack because of the impossibility of obtaining this joint knowledge.

The stop-and-wait strategy is not very useful for modern data networks because of its highly inefficient use of communication links. In particular, it should be possible to do something else while waiting for an ack. There are three common strategies for extending the basic idea of stop-and-wait ARQ so as to achieve higher efficiency: go back n ARQ, selective repeat ARQ, and finally, the ARPANET ARQ.

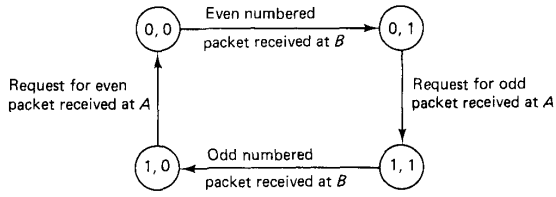


Figure 2.23 State transition diagram for stop-and-wait ARQ. The state is (*SN* mod 2 at A, *RN* mod 2 at B).

Samsung Exhibit 1015, Page 87 of 573

2.4.2 Go Back *n* ARQ

Go back n ARQ is the most widely used type of ARQ protocol; it appears in the various standard DLC protocols, such as HDLC, SDLC, ADCCP, and LAPB. It is not elucidating to know the meaning of these acronyms, but in fact, these standards are almost the same. They are discussed in Section 2.6, and some of their differences are mentioned there. Go back n is also the basis of most error recovery procedures at the transport layer.

The basic idea of go back n is very simple. Incoming packets to a transmitting DLC module for a link from A to B are numbered sequentially, and this sequence number SN is sent in the header of the frame containing the packet. In contrast to stop-and-wait ARQ, several successive packets can be sent without waiting for the next packet to be requested. The receiving DLC at B operates in essentially the same way as stop-and-wait ARQ. It accepts packets only in the correct order and sends request numbers RN back to A; the effect of a given request RN is to acknowledge all packets prior to RN and to request transmission of packet RN.

The go back number $n \geq 1$ in a go back n protocol is a parameter that determines how many successive packets can be sent in the absence of a request for a new packet. Specifically, node A is not allowed to send packet $i+n$ before i has been acknowledged (*i.e.*, before $i+1$ has been requested). Thus, if i is the most recently received request from node B, there is a "window" of n packets, from i to $i+n-1$, that the transmitter is allowed to send. As successively higher-numbered requests are received from B, this window slides upward; thus go back n protocols are often called *sliding window* ARQ protocols.

Figure 2.24 illustrates the operation of go back 7 ARQ when piggybacking of request numbers is being used and when there are no errors and a constant supply of traffic. Although the figure portrays data traffic in both directions, the flow of sequence numbers is shown in one direction and request numbers in the other. Note that when the first frame from A (containing packet 0) is received at B, node B is already in the middle of transmitting its second frame. The piggybacked request number at node B is traditionally in the frame header, and the frame is traditionally completely assembled before transmission starts. Thus when packet 0 is received from A, it is too late for node B to request packet 1 in the second frame, so that $RN = 1$ does not appear until the third frame from B. When this frame is completely received at A, the window at A "slides up" from $[0, 6]$ to $[1, 7]$.

Note that even in the absence of transmission errors, there are several sources of delay between the time that a packet is first assembled into a frame at A and the time when A receives an acknowledgment of the packet. First there is the transmission time of the frame, then the propagation delay, then the wait until the frame in transmission at B is completed, then the transmission time of the frame carrying the acknowledgment, and finally, the B-to-A propagation delay; the effect of these delays is discussed later.

Figure 2.25 shows the effect of error frames on go back 4 ARQ. The second frame from A, carrying packet 1, is received in error at node B. Node B continues to look for packet 1 and to transmit $RN = 1$ in frames from B to A. Packets 2, 3, and 4 from A

Samsung Exhibit 1015, Page 88 of 573

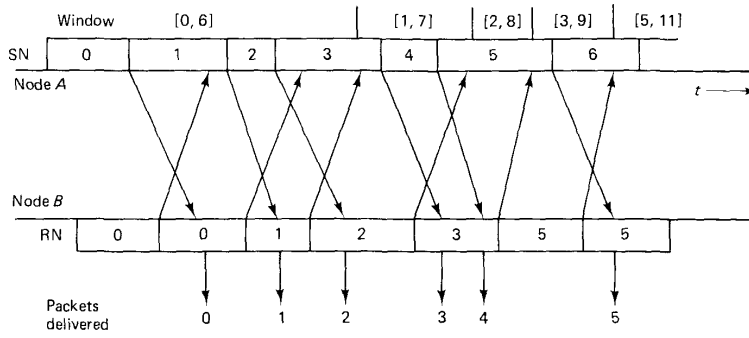


Figure 2.24 Example of go back 7 protocol for A-to-B traffic. Both nodes are sending data, but only the sequence numbers are shown for the frames transmitted from A and only the request numbers are shown for the frames from B. When packet 0 is completely received at B, it is delivered to the higher layer, as indicated at the lower left of the figure. At this point, node B wants to request packet 1, so it sends $RN = 1$ in the next outgoing frame. When that outgoing frame is completely received at A, node A updates its window from [0,6] to [1,7]. Note that when packets 3 and 4 are both received at B during the same frame transmission at B, node B awaits packet 5 and uses $RN = 5$ in the next frame from B to acknowledge both packets 3 and 4.

all arrive at B in error-free frames but are not accepted since node B is looking only for packet 1. One might think that it would be more efficient for node B to buffer packets 2, 3, and 4, thus avoiding the necessity for A to retransmit them after packet 1 is finally retransmitted. Such a buffering strategy is indeed possible and is called *selective repeat* ARQ; this is discussed in Section 2.4.3, but, by definition, go back n does not include this possibility.

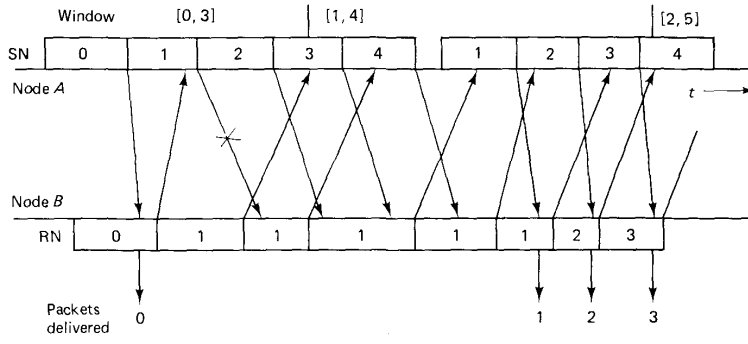


Figure 2.25 Effect of a transmission error on go back 4. Packet 1 is received in error at B, and node B continues to request packet 1 in each reverse frame until node A transmits its entire window, times-out, and goes back to packet 1.

Samsung Exhibit 1015, Page 89 of 573

After node A has completed transmission of the frame containing packet 4 (see Fig. 2.25), it has exhausted its window, which is still [1, 4]; node A then goes back and retransmits packet 1. The figure shows the retransmission of packet 1 after a time-out. There are many possible strategies for the timing of retransmissions within the go back n protocol, and for this reason, go back n is really an entire class of algorithms (just like stop and wait). In go back n, however, not only is the timing of transmissions unspecified, but also the selection of a packet within the window. The class of algorithms will be specified precisely after going through several more examples.

Figure 2.26 illustrates the effect of error frames in the reverse direction (from node B to A). Such an error frame need not slow down transmission in the A-to-B direction, since a subsequent error-free frame from B to A can acknowledge the packet and perhaps some subsequent packets (*i.e.*, the third frame from B to A in the figure acknowledges both packets 1 and 2). On the other hand, with a small window and long frames from B to A, an error frame from B to A can delay acknowledgments until after all the packets in the window are transmitted, thus causing A to either wait or to go back. Note that when the delayed acknowledgments get through, node A can jump forward again (from packet 3 to 5 in Fig. 2.26).

Finally, Fig. 2.27 shows that retransmissions can occur even in the absence of any transmission errors. This happens particularly in the case of short frames in one direction and long frames in the other. We discuss the impact of this phenomenon on the choice of window size and frame length later.

Rules followed by transmitter and receiver in go back *n* We now specify the precise rules followed by the class of go back n protocols and then, in the next

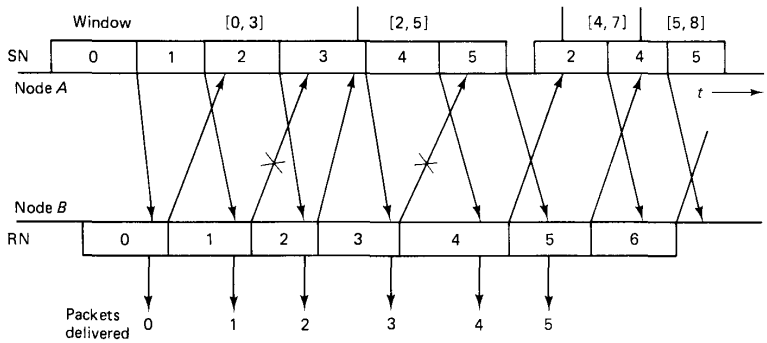


Figure 2.26 Effect of transmission errors in the reverse direction for go back 4. The first error frame, carrying $RN = 1$, causes no problem since it is followed by an error-free frame carrying $RN = 2$ and this frame reaches A before packet number 3, (*i.e.*, the last packet in the current window at A) has completed transmission and before a time-out occurs. The second error frame, carrying $RN = 3$, causes retransmissions since the following reverse frame is delayed until after node A sends its entire window and times-out. This causes A to go back and retransmit packet 2.

Samsung Exhibit 1015, Page 90 of 573

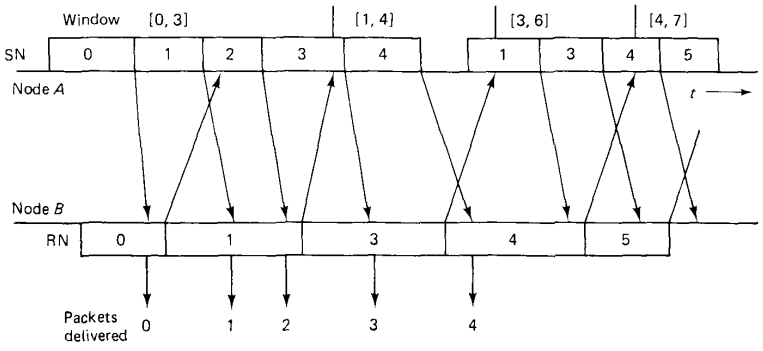


Figure 2.27 Effect of delayed feedback for go back 4. The frames in the B-to-A direction are much longer than those in the A-to-B direction, thus delaying the request numbers from getting back to A. The request for packet 1 arrives in time to allow packet 4 to be sent, but after sending packet 4, node A times-out and goes back to packet 1.

subsection, demonstrate that the algorithms in the class work correctly. We assume here that the sequence numbers and request numbers are integers that can increase without bound. The more practical case in which SN and RN are taken modulo some integer m is considered later.

The rules given here do not treat the initialization of the protocol. We simply assume that initially there are no frames in transit on the link, that node A starts with the transmission of packet number 0, and that node B is initially awaiting packet number 0. How to achieve such an initialization is discussed in Section 2.7.

The transmitter uses two integer variables, SN_{min} and SN_{max}, to keep track of its operations. SN_{min} denotes the smallest-numbered packet that has not yet been acknowledged (*i.e.*, the lower end of the window). SN_{max} denotes the number of the next packet to be accepted from the higher layer. Thus the DLC layer is attempting to transmit packets SN_{min} to $SN_{max} - 1$. Conceptually we can visualize the DLC layer as storing these packets, but it makes no difference where they are stored physically as long as they are maintained somewhere at the node for potential retransmission.

The go back n algorithm at node A for A-to-B transmission:

1. Set the integer variables SN_{min} and SN_{max} to 0.
2. Do steps 3, 4, and 5 repeatedly in any order. There can be an arbitrary but bounded delay between the time when the conditions for a step are satisfied and when the step is executed.
3. If $SN_{max} < SN_{min} + n$, and if a packet is available from the higher layer, accept a new packet into the DLC, assign number SN_{max} to it, and increment SN_{max}.
4. If an error-free frame is received from B containing a request number RN greater than SN_{min}, increase SN_{min} to RN.

Samsung Exhibit 1015, Page 91 of 573

5. If $SN_{min} < SN_{max}$ and no frame is currently in transmission, choose some number SN, $SN_{min} \leq SN < SN_{max}$; transmit the SNth packet in a frame containing SN in the sequence number field. At most a bounded delay is allowed between successive transmissions of packet SN_{min} over intervals when SN_{min} does not change.

The go back n algorithm at node B for A-to-B transmission:

1. Set the integer variable RN to 0 and repeat steps 2 and 3 forever.
2. Whenever an error-free frame is received from A containing a sequence number SN equal to RN, release the received packet to the higher layer and increment RN.
3. At arbitrary times, but within bounded delay after receiving any error-free data frame from A, transmit a frame to A containing RN in the request number field.

There are many conventional ways of handling the timing and ordering of the various operations in the algorithm above. Perhaps the simplest is for node A to set a timer whenever a packet is transmitted. If the timer expires before that packet is acknowledged (*i.e.*, before SN_{min} increases beyond that packet number), the packet is retransmitted. Sometimes when this approach is used, the transmitter, after going back and retransmitting SN_{min}, simply retransmits subsequent packets in order up to $SN_{max} - 1$, whether or not subsequent request numbers are received. For example, at the right-hand edge of Fig. 2.26, the transmitter might have followed packet 3 with 4 rather than 5. In terms of the algorithm as stated, this corresponds to the transmitter delaying the execution of step 4 while in the process of retransmitting a window of packets. Another possibility is for node A to cycle back whenever all the available packets in the window have been transmitted. Also, A might respond to a specific request from B for retransmission; such extra communication between A and B can be considered as part of this class of protocols in the sense that it simply guides the available choices within the algorithm.

Perhaps the simplest approach to timing at node B is to piggyback the current value of RN in each data frame going from B to A. When there are no data currently going from B to A, a nondata frame containing RN should be sent from B to A whenever a data frame is received from A.

Correctness of go back *n* We first demonstrate the correctness of this class of algorithms under our current assumptions that SN and RN are integers; we then show that correctness is maintained if SN and RN are integers modulo m, for m strictly greater than the go back number n. The correctness demonstration when SN and RN are integers is almost the same as the demonstration in Section 2.4.1 for stop and wait. In particular, we start by assuming that all frames with transmission errors are detected by the CRC, that there is some $q > 0$ such that each frame is received error-free with probability at least q, and that the system is correctly initialized in the sense that there

Samsung Exhibit 1015, Page 92 of 573

are no frames on the link and that nodes A and B both start at step 1 of their respective algorithms.

The safety property of the go back n algorithm is exactly the same as for stop and wait. In particular, node B releases packets to the higher layer in order, using the variable RN to track the next packet awaited. To verify the liveness property, assume that i is the value of SN_{min} at node A at a given time t_1 (see Fig. 2.28). Let t_2 be the time at which packet i is received error-free and released to the higher layer at node B; let $t_2 = \infty$ if this event never occurs. Similarly, let t_3 be the time at which SN_{min} is increased beyond i and let $t_3 = \infty$ if this never occurs. We will show that t_3 is finite and that $t_1 < t_3$ and $t_2 < t_3$. This is sufficient to demonstrate liveness, since using the argument for each successive value of SN_{min} shows that each packet is transmitted with finite delay.

Let $RN(t)$ be the value of the variable RN at node B as a function of time t and let $SN_{min}(t)$ be the corresponding value of SN_{min} at node A. It is seen directly from the algorithm statement that $SN_{min}(t)$ and $RN(t)$ are nondecreasing in t. Also, since $SN_{min}(t)$ is the largest request number (if any) received from B up to time t, $SN_{min}(t) \leq RN(t)$. By definition of t_2 and t_3, $RN(t)$ is incremented to $i + 1$ at t_2 and $SN_{min}(t)$ is increased beyond i at t_3. Using the fact that $SN_{min}(t) \leq RN(t)$, it follows that $t_2 < t_3$. Note that it is possible that $t_2 < t_1$, since packet i might have been received error-free and released at B before time t_1 and even before SN_{min} became equal to i.

From the algorithm statement, node A transmits packet i repeatedly, with finite delay between successive transmissions, from t_1 until t_3. If $t_1 < t_2$, then $RN(t) = i$ for $t_1 \leq t \leq t_2$, so the first error-free reception of packet i after t_1 will be accepted and released to the higher layer at B. Since $t_2 < t_3$, node A will retransmit packet i until this happens. Since there is a probability $q > 0$ that each retransmission is received correctly, and retransmissions occur within finite intervals, the time from t_1 to t_2 is finite. Node

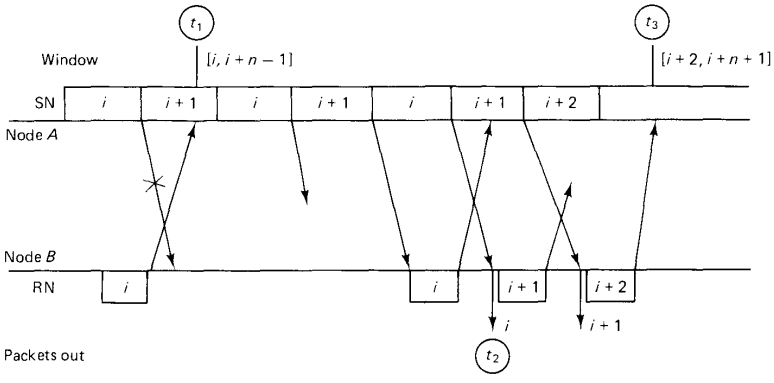


Figure 2.28 SN_{min} is i at time t_1. Packet i is then released to the higher layer at B at time t_2 and the window is increased at A at time t_3.

Samsung Exhibit 1015, Page 93 of 573

B (whether $t_1 < t_2$, or vice versa) transmits frames carrying $RN \geq i + 1$ from time t_2 until some such frame is received error-free at A at time t_3. Since node A is also transmitting frames in this interval, the delay between subsequent transmissions from B is finite, and, since $q > 0$, the interval from t_2 to t_3 is finite. Thus the interval from t_1 to t_3 is finite and the algorithm is live.

It will be observed that no assumption was made in the demonstration above about the frames traveling in order on the links, and the algorithm above operates correctly even when frames get out of order. When we look at error recovery at the transport layer, the role of a link will be played by a subnetwork and the role of a frame will be played by a packet. For datagram networks, packets can get out of order, so this generality will be useful.

Go back *n* with modulus *m* > *n*. It will now be shown that if the sequence number SN and the request number RN are sent modulo m, for some m strictly greater than the go back number n, the correctness of go back n is maintained as long as we reimpose the condition that frames do not get out of order on the links. To demonstrate this correctness, we first look more carefully at the ordering of events when ordinary integers are used for SN and RN.

Consider the transmission of an arbitrary frame from node A to B. Suppose that the frame is generated at time t_1 and received at t_2 (see Fig. 2.29). The sequence number SN of the frame must lie in node A's window at time t_1, so

$$SN_{min}(t_1) \leq SN \leq SN_{min}(t_1) + n - 1 \tag{2.24}$$

Also, as shown in the figure,

$$SN_{min}(t_1) \leq RN(t_2) \leq SN_{min}(t_1) + n \tag{2.25}$$

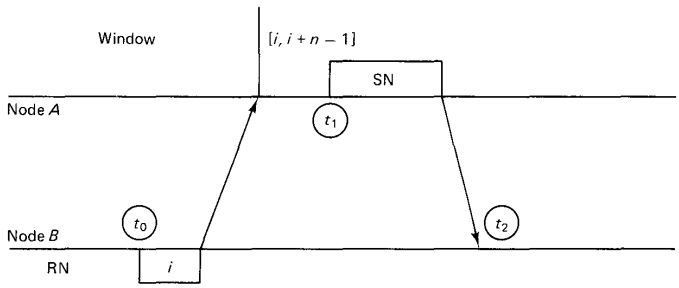


Figure 2.29 Let t_1 and t_2 be the times at which a given frame is generated at A and received at B respectively. The sequence number in the frame satisfies $SN_{min}(t_1) \leq SN \leq SN_{min}(t_1) + n - 1$. The value i of $SN_{min}(t_1)$ is equal to the last received value of RN, which is $RN(t_0) \leq RN(t_2)$. Thus $SN_{min}(t_1) \leq RN(t_2)$, which is the left side of Eq. (2.25). Conversely, no frame with sequence number $SN_{min}(t_1) + n$ can have been sent before t_1 since this value is beyond the upper limit of the window. Since frames travel in order on the link, no frame with this number has arrived at B before t_2, and $RN(t_2) \leq SN_{min}(t_1) + n$, which is the right side of Eq. (2.25).

Samsung Exhibit 1015, Page 94 of 573

We see from Eqs. (2.24) and (2.25) that SN and $RN(t_2)$ are both contained in the interval from $SN_{min}(t_1)$ to $SN_{min}(t_1) + n$, and thus must satisfy

$$|RN(t_2) - SN| \leq n \tag{2.26}$$

Now suppose that when packet number SN is sent, the accompanying sequence number is sent modulo m, and let sn denote SN mod m. Step 3 of the algorithm at node B must then be modified to: If an error-free frame is received from A containing a sequence number sn equal to RN mod m, release the received packet to the higher layer and increment RN. Since $m > n$ by assumption, we see from Eq. (2.26) that $sn = RN$ mod m will be satisfied if and only if the packet number SN is equal to RN; thus, the algorithm still works correctly.

Next consider the ordering of arriving request numbers (using ordinary integers) relative to the window at node A. From Fig. 2.30, we see that

$$SN_{min}(t_2) \leq RN \leq SN_{min}(t_2) + n \tag{2.27}$$

Now suppose that RN is sent modulo m, and let $rn = RN$ mod m. Step 4 of the algorithm at node A must then be modified to: If an error-free frame is received from B containing $rn \neq SN_{min}$ mod m, then increment SN_{min} until $rn = SN_{min}$ mod m. Because of the range of RN in Eq. (2.27), we see that this new rule is equivalent to the old rule, and it is sufficient to send request numbers modulo m. At this point, however, we see that it is unnecessary for SN_{min}, SN_{max}, and RN to be saved at nodes A and B as ordinary integers; everything can be numbered modulo m, and the algorithm has been demonstrated to work correctly for $m > n$.

For completeness, we restate the algorithm for operation modulo m; since all numbers are taken modulo m, we use capital letters for these numbers.

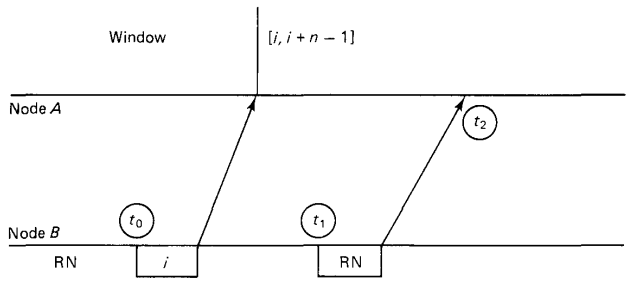


Figure 2.30 Let t_1 and t_2 be the times at which a given frame with request number RN is generated at B and received at A, respectively. Let $SN_{min}(t_2) = i$ be the lower edge of the window at t_2 and let t_0 be the generation time of the frame from B that caused the window at A to move to $[i, i + n - 1]$. Since the frames travel in order on the link, $t_0 < t_1$, so $i \leq RN$ and thus $SN_{min}(t_2) \leq RN$. Similarly, node A cannot have sent packet $i + n$ before t_2, so it certainly cannot have been received before t_1. Thus $RN \leq SN_{min}(t_2) + n$.

Samsung Exhibit 1015, Page 95 of 573

The go back n algorithm at node A for modulo m operation, $m > n$:

1. Set the modulo m variables SN_{min} and SN_{max} to 0.
2. Do steps 3, 4, and 5 repeatedly in any order. There can be an arbitrary but bounded delay between the time when the conditions for a step are satisfied and when the step is executed.
3. If $(SN_{max} - SN_{min}) \bmod m < n$, and if a packet is available from the higher layer, accept a new packet into the DLC, assign number SN_{max} to it, and increment SN_{max} to $(SN_{max} + 1) \bmod m$.
4. If an error-free frame is received from B containing a request number RN, and $(RN - SN_{min}) \bmod m \leq (SN_{max} - SN_{min}) \bmod m$, set SN_{min} to equal RN.
5. If $SN_{min} \neq SN_{max}$ and no frame is currently in transmission, choose some number SN such that $(SN - SN_{min}) \bmod m < (SN_{max} - SN_{min}) \bmod m$; transmit packet SN in a frame containing SN in the sequence number field.

The go back n algorithm at node B for modulo m operation, $m > n$:

1. Set the modulo m variable RN to 0.
2. Whenever an error-free frame is received from A containing a sequence number SN equal to RN, release the received packet to the higher layer and increment RN to $(RN + 1) \bmod m$.
3. At arbitrary times, but within bounded delay after receiving any error-free data frame from A, transmit a frame to A containing RN in the request number field.

Efficiency of go back *n* implementations Retransmissions, or delays waiting for time-outs, occur in go back n ARQ for the following three reasons: first, errors in the forward direction, second, errors in the feedback direction, and third, longer frames in the feedback than in the forward direction. These will be discussed in reverse order.

The likelihood of retransmissions caused by long reverse frames can be reduced by increasing the go back number n. Unfortunately, the normal value of modulus in the standard protocols is $m = 8$, which constrains n to be at most 7. Figure 2.31 illustrates that even in the absence of propagation delay, reverse frames more than three times the length of forward frames can cause retransmissions for $n = 7$. Problem 2.23 also shows that if frames have lengths that are exponentially distributed, with the same distribution

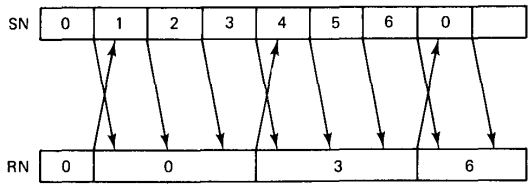


Figure 2.31 Go back 7 ARQ with long frames in the reverse direction. Note that the ack for packet 1 has not arrived at the sending side by the time packet 6 finishes transmission, thereby causing a retransmission of packet 0.

Samsung Exhibit 1015, Page 96 of 573

in each direction, the probability p that a frame is not acked by the time the window is exhausted is given by

$$p = (1 + n)2^{-n} \tag{2.28}$$

For $n = 7$, p is equal to 1/16. Frames normally have a maximum permissible length (and a minimum length because of the control overhead), so in practice p is somewhat smaller. However, links sometimes carry longer frames in one direction than in the other, and in this case, there might be considerable waste in link utilization for $n = 7$. When propagation delay is large relative to frame length (as, for example, on high-speed links and satellite links), this loss of utilization can be quite serious. Fortunately, the standard protocols have an alternative choice of modulus as $m = 128$.

When errors occur in a reverse frame, the acknowledgment information is postponed for an additional reverse frame. This loss of line utilization in one direction due to errors in the other can also be avoided by choosing n sufficiently large.

Finally, consider the effect of errors in the forward direction. If n is large enough to avoid retransmissions or delays due to large propagation delay and long frames or errors in the reverse direction, and if the sending DLC waits to exhaust its window of n packets before retransmitting, a large number of packets are retransmitted for each forward error. The customary solution to this problem is the use of time-outs. In its simplest version, if a packet is not acknowledged within a fixed time-out period after transmission, it is retransmitted. This time-out should be chosen long enough to include round-trip propagation and processing delay plus transmission time for two maximum-length packets in the reverse direction (one for the frame in transit when a packet is received, and one to carry the new RN). In a more sophisticated version, the sending DLC, with knowledge of propagation and processing delays, can determine which reverse frame should carry the ack for a given packet; it can go back if that frame is error free and fails to deliver the ack.

In a more fundamental sense, increasing link utilization and decreasing delay is achieved by going back quickly when a forward error occurs, but avoiding retransmissions caused by long frames and errors in the reverse direction. One possibility here is for the receiving DLC to send back a short supervisory frame upon receiving a frame in error. This allows the sending side to go back much sooner than if RN were simply piggybacked on a longer reverse data frame. Another approach is to insert RN in the trailer of the reverse frame, inserting it before the CRC and inserting it at the last moment, after the packet part of the frame has been sent. This cannot be done in the standard DLC protocols, but would have the effect of reducing the feedback delay by almost one frame length.

It is not particularly difficult to invent new ways of reducing both feedback delay and control overhead in ARQ strategies. One should be aware, however, that it is not trivial to ensure the correctness of such strategies. Also, except in special applications, improvements must be substantial to outweigh the advantages of standardization.

2.4.3 Selective Repeat ARQ

Even if unnecessary retransmissions are avoided, go back n protocols must retransmit at least one round-trip-delay worth of frames when a single error occurs in an awaited

Samsung Exhibit 1015, Page 97 of 573

packet. In many situations, the probability of one or more errors in a frame is 10^{-4} or less, and in this case, retransmitting many packets for each frame in error has little effect on efficiency. There are some communication links, however, for which small error probabilities per frame are very difficult to achieve, even with error correction in the modems. For other links (*e.g.*, high-speed links and satellite links), the number of frames transmitted in a round- trip delay time is very large. In both these cases, selective repeat ARQ can be used to increase efficiency.

The basic idea of selective repeat ARQ for data on a link from A to B is to accept out-of-order packets and to request retransmissions from A only for those packets that are not correctly received. There is still a go back number, or window size, n, specifying how far A can get ahead of RN, the lowest-numbered packet not yet correctly received at B.

Note that whatever ARQ protocol is used, only error-free frames can deliver packets to B, and thus, if p is the probability of frame error, the expected number η of packets delivered to B per frame from A to B is bounded by

$$\eta \leq 1 - p \tag{2.29}$$

As discussed later, this bound can be approached (in principle) by selective repeat ARQ; thus $1 - p$ is sometimes called the throughput of ideal selective repeat. Ideal go back n ARQ can similarly be defined as a protocol that retransmits the packets in one round-trip delay each time that a frame carrying the packet awaited by the receiving DLC is corrupted by errors. The throughput of this ideal is shown in Problem 2.26 to be

$$\eta \leq \frac{1 - p}{1 + p\beta} \tag{2.30}$$

where β is the expected number of frames in a round-trip delay interval. This indicates that the increase in throughput available with selective repeat is significant only when $p\beta$ is appreciable relative to 1.

The selective repeat algorithm at node A, for traffic from A to B, is the same as the go back n algorithm, except that (for reasons soon to be made clear) the modulus m must satisfy $m \geq 2n$. At node B, the variable RN has the same significance as in go back n; namely, it is the lowest-numbered packet (or the lowest-numbered packet modulo m) not yet correctly received. In the selective repeat algorithm, node B accepts packets anywhere in the range RN to $RN + n - 1$. The value of RN must be sent back to A as in go back n, either piggybacked on data frames or sent in separate frames. Usually, as discussed later, the feedback from node B to A includes not only the value of RN but also additional information about which packets beyond RN have been correctly received. In principle, the DLC layer still releases packets to the higher layer in order, so that the accepted out-of-order packets are saved until the earlier packets are accepted and released.

We now assume again that frames do not get out of order on the links and proceed to see why a larger modulus is required for selective repeat than for go back n. Assume that a frame is received at node B at some given time t_2 and that the frame was generated at node A at time t_1. If SN and RN are considered as integers, Eqs. (2.24) and (2.25)

Samsung Exhibit 1015, Page 98 of 573

are still valid, and we can conclude from them that the sequence number SN in the received frame must satisfy

$$RN(t_2) - n \leq SN \leq RN(t_2) + n - 1 \tag{2.31}$$

If sequence numbers are sent mod m, and if packets are accepted in the range $RN(t_2)$ to $RN(t_2) + n - 1$, it is necessary for node B to distinguish values of SN in the entire range of Eq. (2.31). This means that the modulus m must satisfy

$$m \geq 2n, \qquad \text{for selective repeat} \tag{2.32}$$

With this change, the correctness of this class of protocols follows as before. The real issue with selective repeat, however, is using it efficiently to achieve throughputs relatively close to the ideal $1 - p$. Note first that using RN alone to provide acknowledgment information is not very efficient, since if several frame errors occur in one round-trip delay period, node A does not find out about the second frame error until one round-trip delay after the first error is retransmitted. There are several ways of providing the additional acknowledgment information required by A. One is for B to send back the lowest j packet numbers that it has not yet received; j should be larger than $p\beta$ (the expected number of frame errors in a round-trip delay time), but is limited by the overhead required by the feedback. Another possibility is to send RN plus an additional k bits (for some constant k), one bit giving the ack/nak status of each of the k packets after RN.

Assume now that the return frames carry sufficient information for A to determine, after an expected delay of β frames, whether or not a packet was successfully received. The typical algorithm for A is then to repeat packets as soon as it is clear that the previous transmission contained errors; if A discovers multiple errors simultaneously, it retransmits them in the order of their packet numbers. When there are no requested retransmissions, A continues to send new packets, up to $SN_{max} - 1$. At this limit, node A can wait for some time-out or immediately go back to SN_{min} to transmit successive unacknowledged packets.

Node A acts like an ideal selective repeat system until it is forced to wait or go back from $SN_{max} - 1$. When this happens, however, the packet numbered SN_{min} must have been transmitted unsuccessfully about n/β times. Thus, by making n large enough, the probability of such a go back can be reduced to negligible value. There are two difficulties with very large values of n (assuming that packets must be reordered at the receiving DLC). The first is that storage must be provided at B for all of the accepted packets beyond RN. The second is that the large number of stored packets are all delayed waiting for RN.

The amount of storage provided can be reduced to $n - \beta$ without much harm, since whenever a go back from $SN_{max} - 1$ occurs, node A can send no new packets beyond $SN_{max} - 1$ for a round-trip delay; thus, it might as well resend the yet-unacknowledged packets from to $SN_{max} - \beta$ to $SN_{max} - 1$; this means that B need not save these packets. The value of n can also be reduced, without increasing the probability of go back, by retransmitting a packet several times in succession if the previous transmission contained errors. For example, Fig. 2.32 compares double retransmissions with single retransmissions for $n = 2\beta + 2$. Single retransmissions fail with probability p and cause β extra retrans-

Samsung Exhibit 1015, Page 99 of 573

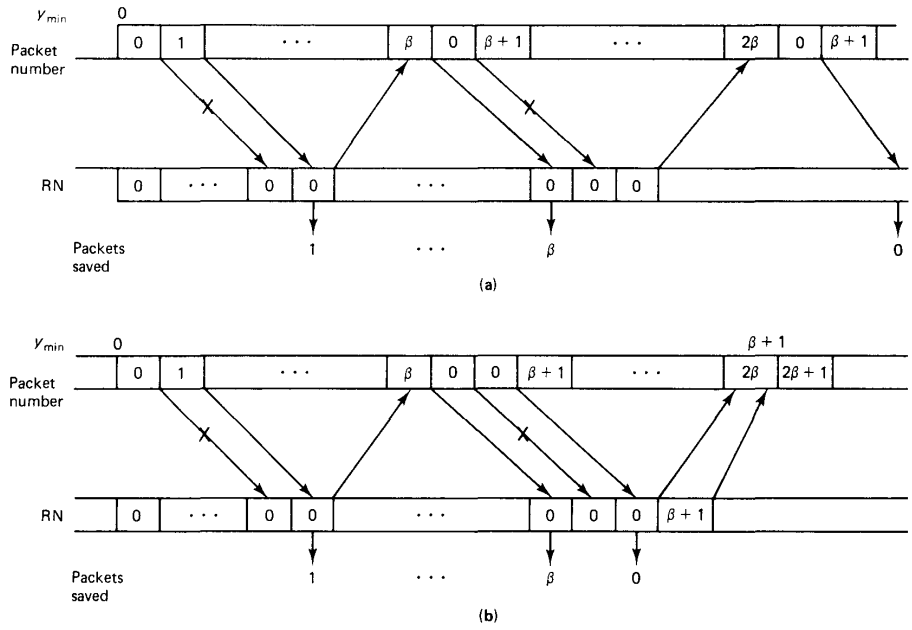


Figure 2.32 Selective repeat ARQ with $n = 2\beta + 2$ and receiver storage for $\beta + 1$ packets. (a) Note the wasted transmissions if a given packet (0) is transmitted twice with errors. (b) Note that this problem is cured, at the cost of one extra frame, if the second transmission of packet 0 is doubled. Feedback contains not only RN but additional information on accepted packets.

missions; double retransmissions rarely fail, but always require one extra retransmission. Thus, double retransmissions can increase throughput if $pn > 1$, but might be desirable in other cases to reduce the variability of delay. (See [Wel82] for further analysis.)

2.4.4 ARPANET ARQ

The ARPANET achieves efficiency by using eight stop-and-wait strategies in parallel, multiplexing the bit pipe between the eight. That is, each incoming packet is assigned to one of eight virtual channels, assuming that one of the eight is idle; if all the virtual channels are busy, the incoming packet waits outside the DLC (see Fig. 2.33). The busy virtual channels are multiplexed on the bit pipe in the sense that frames for the different virtual channels are sent one after the other on the link. The particular order in which frames are sent is not very important, but a simple approach is to send them in round-robin order. If a virtual channel's turn for transmission comes up before an ack has been received for that virtual channel, the packet is sent again, so that the multiplexing removes the need for any time-outs. (The actual ARPANET protocol, however, does use

Samsung Exhibit 1015, Page 100 of 573

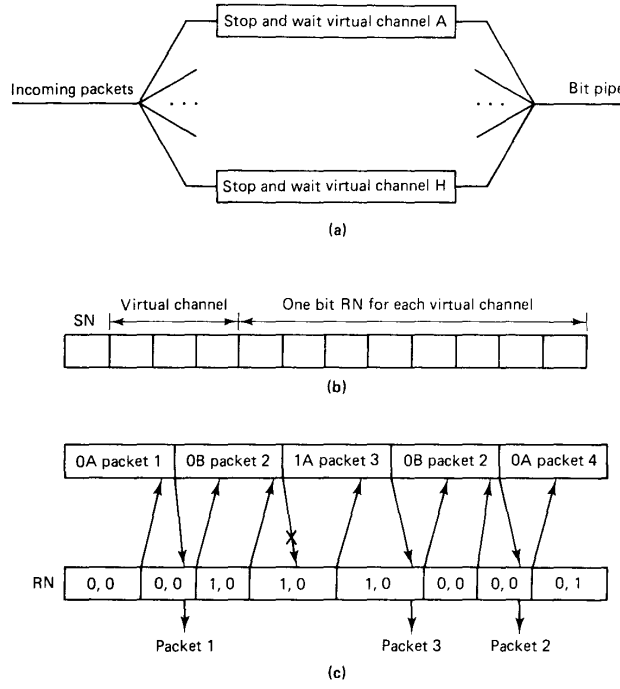


Figure 2.33 ARPANET ARQ. (a) Eight multiplexed, stop-and-wait virtual channels. (b) Bits in the header for ARQ control. (c) Operation of multiplexed stop and wait for two virtual channels. Top-to-bottom frames show SN and the channel number, and bottom-to-top frames show RN for both channels. The third frame from bottom to top acks packet 1 on the A channel.

time-outs.) When an ack is received for a frame on a given virtual channel, that virtual channel becomes idle and can accept a new packet from the higher layer.

Somewhat more overhead is required here than in the basic stop-and-wait protocol. In particular, each frame carries both the virtual channel number (requiring three bits) and the sequence number modulo 2 (*i.e.*, one bit) of the packet on that virtual channel. The acknowledgment information is piggybacked onto the frames going in the opposite direction. Each such frame, in fact, carries information for all eight virtual channels. In particular, an eight-bit field in the header of each return frame gives the number modulo 2 of the awaited packet for each virtual channel.

One of the desirable features of this strategy is that the ack information is repeated so often (*i.e.*, for all virtual channels in each return frame) that relatively few retransmissions are required because of transmission errors in the reverse direction. Typically, only one retransmission is required for each frame in error in the forward direction. The

Samsung Exhibit 1015, Page 101 of 573

undesirable feature of the ARPANET protocol is that packets are released to the higher layer at the receiving DLC in a different order from that of arrival at the sending DLC. The DLC layer could, in principle, reorder the packets, but since a packet on one virtual channel could be arbitrarily delayed, an arbitrarily large number of later packets might have to be stored. The ARPANET makes no effort to reorder packets on individual links, so this protocol is not a problem for ARPANET. We shall see later that the lack of ordering on links generates a number of problems at higher layers. Most modern networks maintain packet ordering for this reason, and consequently do not use this protocol despite its high efficiency and low overhead. For very poor communication links, where efficiency and overhead are very important, it is a reasonable choice.

2.5 FRAMING

The problem of framing is that of deciding, at the receiving DLC, where successive frames start and stop. In the case of a synchronous bit pipe, there is sometimes a period of idle fill between successive frames, so that it is also necessary to separate the idle fill from the frames. For an intermittent synchronous bit pipe, the idle fill is replaced by dead periods when no bits at all arrive. This does not simplify the problem since, first, successive frames are often transmitted with no dead periods in between, and second, after a dead period, the modems at the physical layer usually require some idle fill to reacquire synchronization.

There are three types of framing used in practice. The first, *character-based framing*, uses special communication control characters for idle fill and to indicate the beginning and ending of frames. The second, *bit-oriented framing with flags*, uses a special string of bits called a flag both for idle fill and to indicate the beginning and ending of frames. The third, *length counts*, gives the frame length in a field of the header. The following three subsections explain these three techniques, and the third also gives a more fundamental view of the problem. These subsections, except for a few comments, ignore the possibility of errors on the bit pipe. Section 2.5.4 then treats the joint problems of ARQ and framing in the presence of errors. Finally, Section 2.5.5 explains the trade-offs involved in the choice of frame length.

2.5.1 Character-Based Framing

Character codes such as ASCII generally provide binary representations not only for keyboard characters and terminal control characters, but also for various communication control characters. In ASCII, all these binary representations are seven bits long, usually with an extra parity bit which might or might not be stripped off in communication [since a cyclic redundancy check (CRC) can be used more effectively to detect errors in frames].

SYN (synchronous idle) is one of these communication control characters; a string of SYN characters provides idle fill between frames when a sending DLC has no data to send but a synchronous modem requires bits. SYN can also be used within frames,

Samsung Exhibit 1015, Page 102 of 573

sometimes for synchronization of older modems, and sometimes to bridge delays in supplying data characters. STX (start of text) and ETX (end of text) are two other communication control characters used to indicate the beginning and end of a frame, as shown in Fig. 2.34.

The character-oriented communication protocols used in practice, such as the IBM binary synchronous communication system (known as Bisynch or BSC), are far more complex than this, but our purpose here is simply to illustrate that framing presents no insurmountable problems. There is a slight problem in the example above in that either the header or the CRC might, through chance, contain a communication control character. Since these always appear in known positions after STX or ETX, this causes no problem for the receiver. If the packet to be transmitted is an arbitrary binary string, however, rather than a string of ASCII keyboard characters, serious problems arise; the packet might contain the ETX character, for example, which could be interpreted as ending the frame. Character-oriented protocols use a special mode of transmission, called *transparent mode*, to send such data.

The transparent mode uses a special control character called DLE (data link escape). A DLE character is inserted before the STX character to indicate the start of a frame in transparent mode. It is also inserted before intentional uses of communication control characters within such a frame. The DLE is not inserted before the possible appearances of these characters as part of the binary data. There is still a problem if the DLE character itself appears in the data, and this is solved by inserting an extra DLE before each appearance of DLE in the data proper. The receiving DLC then strips off one DLE from each arriving pair of DLEs, and interprets each STX or ETX preceded by an unpaired DLE as an actual start or stop of a frame. Thus, for example, DLE ETX (preceded by something other than DLE) would be interpreted as an end of frame, whereas DLE DLE ETX (preceded by something other than DLE) would be interpreted as an appearance of the bits corresponding to DLE ETX within the binary data.

With this type of protocol, the frame structure would appear as shown in Fig. 2.35. This frame structure is used in the ARPANET. It has two disadvantages, the first of which is the somewhat excessive use of framing overhead (*i.e.*, DLE STX precedes each frame, DLE ETX follows each frame, and two SYN characters separate each frame for a total of six framing characters per frame). The second disadvantage is that each frame must consist of an integral number of characters.

Let us briefly consider what happens in this protocol in the presence of errors. The CRC checks the header and packet of a frame, and thus will normally detect errors

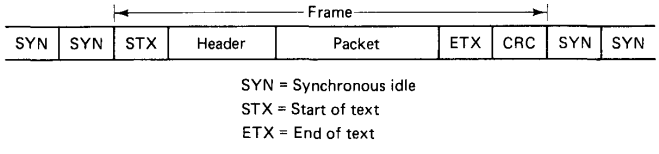


Figure 2.34 Simplified frame structure with character-based framing.

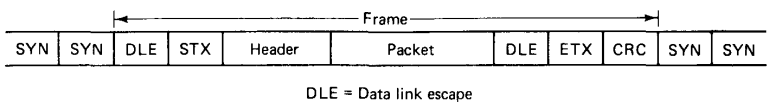


Figure 2.35 Character-based framing in a transparent mode as used in the ARPANET.

there. If an error occurs in the DLE ETX ending a frame, however, the receiver will not detect the end of the frame and thus will not check the CRC; this is why in the preceding section we assumed that frames could get lost. A similar problem is that errors can cause the appearance of DLE ETX in the data itself; the receiver would interpret this as the end of the frame and interpret the following bits as a CRC. Thus, we have an essentially random set of bits interpreted as a CRC, and the preceding data will be accepted as a packet with probability 2^{-L}, where L is the length of the CRC. The same problems occur in the bit-oriented framing protocols to be studied next and are discussed in greater detail in Section 2.5.4.

2.5.2 Bit-Oriented Framing: Flags

In the transparent mode for character-based framing, the special character pair DLE ETX indicated the end of a frame and was avoided within the frame by doubling each DLE character. Here we look at another approach, using a *flag* at the end of the frame. A flag is simply a known bit string, such as DLE ETX, that indicates the end of a frame. Similar to the technique of doubling DLEs, a technique called *bit stuffing* is used to avoid confusion between possible appearances of the flag as a bit string within the frame and the actual flag indicating the end of the frame. One important difference between bit-oriented and character based framing is that a bit-oriented frame can have any length (subject to some minimum and maximum) rather than being restricted to an integral number of characters. Thus, we must guard against appearances of the flag bit pattern starting in any bit position rather than just on character boundaries.

In practice, the flag is the bit string 01^60, where the notation 1^j means a string of j 1's. The rule used for bit stuffing is to insert (stuff) a 0 into the data string of the frame proper after each successive appearance of five 1's (see Fig. 2.36). Thus, the frame, after stuffing, never contains more than five consecutive 1's, and the flag at the end of the frame is uniquely recognizable. At the receiving DLC, the first 0 after each string of five consecutive 1's is deleted; if, instead, a string of five 1's is followed by a 1, the frame is declared to be finished.

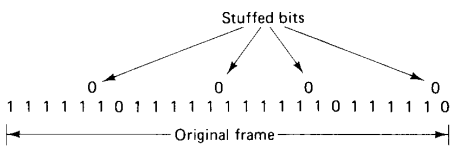


Figure 2.36 Bit stuffing. A 0 is stuffed after each consecutive five 1's in the original frame. A flag, 01111110, without stuffing, is sent at the end of the frame.

Samsung Exhibit 1015, Page 104 of 573

Bit stuffing has a number of purposes beyond eliminating flags within the frame. Standard DLCs have an abort capability in which a frame can be aborted by sending seven or more 1's in a row; in addition, a link is regarded as idle if 15 or more 1's in a row are received. What this means is that 01^6 is really the string denoting the end of a frame. If 01^6 is followed by a 0, it is the flag, indicating normal frame termination; if followed by a 1, it indicates abnormal termination. Bit stuffing is best viewed as preventing the appearance of 01^6 within the frame. One other minor purpose of bit stuffing is to break up long strings of 1's, which cause some older modems to lose synchronization.

It is easy to see that the bit stuffing rule above avoids the appearance of 01^6 within the frame, but it is less clear that so much bit stuffing is necessary. For example, consider the first stuffed bit in Fig. 2.36. Since the frame starts with six 1's (following a distinguishable flag), this could not be logically mistaken for 01^6. Thus, stuffing is not logically necessary after five 1's at the beginning of the frame (provided that the receiver's rule for deleting stuffed bits is changed accordingly).

The second stuffed bit in the figure is clearly necessary to avoid the appearance of 01^6. From a strictly logical standpoint, the third stuffed bit could be eliminated (except for the synchronization problem in older modems). Problem 2.31 shows how the receiver rule could be appropriately modified. Note that the reduction in overhead, by modifying the stuffing rules as above, is almost negligible; the purpose here is to understand the rules rather than to suggest changes.

The fourth stuffed bit in the figure is definitely required, although the reason is somewhat subtle. The original string 01^50 surrounding this stuffed bit could not be misinterpreted as 01^6, but the receiving DLC needs a rule to eliminate stuffed bits; it cannot distinguish a stuffed 0 following 01^5 from a data 0 following 01^5. Problem 2.32 develops this argument in detail.

There is nothing particularly magical about the string 01^6 as the bit string used to signal the termination of a frame (except for its use in preventing long strings of 1's on the link), and in fact, any bit string could be used with bit stuffing after the next-to-last bit of the string (see Problem 2.33). Such strings, with bit stuffing, are often useful in data communication to signal the occurrence of some rare event.

Consider the overhead incurred by using a flag to indicate the end of a frame. Assume that a frame (before bit stuffing and flag addition) consists of independent, identically distributed, random binary variables, with equal probability of 0 or 1. Assume for increased generality that the terminating signal for a frame is 01^j for some j (with 01^j0 being the flag and 01^{j+1} indicating abnormal termination); thus, $j = 6$ for the standard flag. An insertion will occur at (*i.e.*, immediately following) the i^{th} bit of the original frame (for $i \geq j$) if the string from $i - j + 1$ to i is 01^{j-1}; the probability of this is 2^{-j}. An insertion will also occur (for $i \geq 2j - 1$) if the string from $i - 2j + 2$ to i is 01^{2j-2}; the probability of this is 2^{-2j+1}. We ignore this term and the probability of insertions due to yet longer strings of 1's: first, because these probabilities are practically negligible, and second, because these insertions are used primarily to avoid long strings of 1's rather than to provide framing. Bit $j - 1$ in the frame is somewhat different than the other bits, since an insertion here occurs with probability 2^{-j+1} (*i.e.*, if the first $j - 1$ bits of the frame are all 1's).

Samsung Exhibit 1015, Page 105 of 573

Recall that the expected value of a sum of random variables is equal to the sum of the expected values (whether or not the variables are independent). Thus, the expected number of insertions in a frame of original length K is the sum, over i, of the expected number of insertions at each bit i of the frame. The expected number of insertions at a given bit, however, is just the probability of insertion there. Thus, the expected number of insertions in a string of length $K \geq j - 1$ is

$$(K - j + 3)2^{-j}$$

Taking the expected value of this over frame lengths K (with the assumption that all frames are longer than $j - 1$) and adding the $j + 1$ bits in the termination string, the expected overhead for framing becomes

$$E\{OV\} = (E\{K\} - j + 3)2^{-j} + j + 1 \tag{2.33}$$

Since $E\{K\}$ is typically very much larger than j, we have the approximation and upper bound (for $j \geq 3$)

$$E\{OV\} \leq E\{K\}2^{-j} + j + 1 \tag{2.34}$$

One additional bit is needed to distinguish a normal from an abnormal end of frame.

It will be interesting to find the integer value of j that minimizes this expression for a given value of expected frame length. As j increases from 1, the quantity on the right-hand side of Eq. (2.34) first decreases and then increases. Thus, the minimizing j is the smallest integer j for which the right-hand side is less than the same quantity with j increased by 1, that is, the smallest j for which

$$E\{K\}2^{-j} + j + 1 < E\{K\}2^{-j-1} + j + 2 \tag{2.35}$$

This inequality simplifies to $E\{K\}2^{-j-1} < 1$, and the smallest j that satisfies this is

$$j = \lfloor \log_2 E\{K\} \rfloor \tag{2.36}$$

where $\lfloor x \rfloor$ means the integer part of x. It is shown in Problem 2.34 that for this optimal value of j,

$$E\{OV\} \leq \log_2 E\{K\} + 2 \tag{2.37}$$

For example, with an expected frame length of 1000 bits, the optimal j is 9 and the expected framing overhead is less than 12 bits. For the standard flag, with $j = 6$, the expected overhead is about 23 bits (hardly cause for wanting to change the standard).

2.5.3 Length Fields

The basic problem in framing is to inform the receiving DLC where each idle fill string ends and where each frame ends. In principle, the problem of determining the end of an idle fill string is trivial; idle fill is represented by some fixed string (*e.g.*, repeated SYN characters or repeated flags) and idle fill stops whenever this fixed pattern is broken. In principle, one bit inverted from the pattern is sufficient, although in practice, idle fill is usually stopped at a boundary between flags or SYN characters.

Samsung Exhibit 1015, Page 106 of 573

Since a frame consists of an arbitrary and unknown bit string, it is somewhat harder to indicate where it ends. A simple alternative to flags or special characters is to include a length field in the frame header. DECNET, for example, uses this framing technique. Assuming no transmission errors, the receiving DLC simply reads this length in the header and then knows where the frame ends. If the length is represented by ordinary binary numbers, the number of bits in the length field has to be at least $\lfloor \log_2 K_{\max} \rfloor + 1$, where $K_{\max}$ is the maximum frame size. This is the overhead required for framing in this technique; comparing it with Eq. (2.37) for flags, we see that the two techniques require similar overhead.

Could any other method of encoding frame lengths require a smaller expected number of bits? This question is answered by information theory. Given any probability assignment $P(K)$ on frame lengths, the source coding theorem of information theory states that the minimum expected number of bits that can encode such a length is at least the entropy of that distribution, given by

$$H = \sum_K P(K) \log_2 \frac{1}{P(K)} \tag{2.38}$$

According to the theorem, at least this many bits of framing overhead, on the average, must be sent over the link per frame for the receiver to know where each frame ends. If $P(K) = 1/K_{\max}$, for $1 \leq K \leq K_{\max}$, then H is easily calculated to be $\log_2 K_{\max}$. Similarly, for a geometric distribution on lengths, with given $E\{K\}$, the entropy of the length distribution is approximately $\log_2 E\{K\} + \log_2 e$, for large $E\{K\}$. This is about 1/2 bit below the expression in Eq. (2.37). Thus, for the geometric distribution on lengths, the overhead using flags for framing is essentially minimum. The geometric distribution has an interesting extremal property; it can be shown to have the largest entropy of any probability distribution over the positive integers with given $E\{K\}$ (*i.e.*, it requires more bits than any other distribution).

The general idea of source coding is to map the more likely values of K into short bit strings and less likely values into long bit strings; more precisely, one would like to map a given K into about $\log_2[1/P(K)]$ bits. If one does this for a geometric distribution, one gets an interesting encoding known as the unary–binary encoding. In particular, for a given j, the frame length K is represented as

$$K = i2^j + r; \quad 0 \leq r < 2^j \tag{2.39}$$

The encoding for K is then i 0's followed by a 1 (this is known as a unary encoding of i) followed by the ordinary binary encoding of r (using j bits). For example, if $j = 2$ and $K = 7$, K is represented by $i = 1$, $r = 3$, which encodes into 0111 (where 01 is the unary encoding of $i = 1$ and 11 is the binary encoding of $r = 3$). Note that different values of K are encoded into different numbers of bits, but the end of the encoding can always be recognized as occurring j bits after the first 1.

In general, with this encoding, a given K maps into a bit string of length $\lfloor K/2^j \rfloor + 1 + j$. If the integer value above is neglected and the expected value over K is taken, then

$$E\{OV\} = E\{K\}2^{-j} + 1 + j \tag{2.40}$$

Samsung Exhibit 1015, Page 107 of 573

Note that this is the same as the flag overhead in Eq. (2.34). This is again minimized by choosing $j = \lfloor \log_2 E\{K\} \rfloor$. Thus, this unary–binary length encoding and flag framing both require essentially the minimum possible framing overhead for the geometric distribution, and no more overhead for any other distribution of given $E\{K\}$.

2.5.4 Framing with Errors

Several peculiar problems arise when errors corrupt the framing information on the communication link. First, consider the flag technique. If an error occurs in the flag at the end of a frame, the receiver will not detect the end of frame and will not check the cyclic redundancy check (CRC). In this case, when the next flag is detected, the receiver assumes the CRC to be in the position preceding the flag. This perceived CRC might be the actual CRC for the following frame, but the receiver interprets it as checking what was transmitted as two frames. Alternatively, if some idle fill follows the frame in which the flag was lost, the perceived CRC could include the error-corrupted flag. In any case, the perceived CRC is essentially a random bit string in relation to the perceived preceding frame, and the receiver fails to detect the errors with a probability essentially 2^{-L}, where L is the length of the CRC.

An alternative scenario is for an error within the frame to change a bit string into the flag, as shown for the flag 01^60:

$$0\,1\,0\,0\,1\,1\,0\,1\,1\,1\,0\,0\,1\ \ldots \qquad \text{(sent)}$$

$$0\,1\,0\,0\,1\,1\,1\,1\,1\,1\,0\,0\,1\ \ldots \qquad \text{(received)}$$

It is shown in Problem 2.35 that the probability of this happening somewhere in a frame of K independent equiprobable binary digits is approximately $(1/32)Kp$, where p is the probability of a bit error. In this scenario, as before, the bits before the perceived flag are interpreted by the receiver as a CRC, and the probability of accepting a false frame, given this occurrence, is 2^{-L}. This problem is often called the data sensitivity problem of DLC, since even though the CRC is capable of detecting any combination of three or fewer errors, a single error that creates or destroys a flag, plus a special combination of data bits to satisfy the perceived preceding CRC, causes an undetectable error.

If a length field in the header provides framing, an error in this length field again causes the receiver to look for the CRC in the wrong place, and again an incorrect frame is accepted with probability about 2^{-L}. The probability of such an error is smaller using a length count than using a flag (since errors can create false flags within the frame); however, after an error occurs in a length field, the receiver does not know where to look for any subsequent frames. Thus, if a length field is used for framing, some synchronizing string must be used at the beginning of a frame whenever the sending DLC goes back to retransmit. (Alternatively, synchronization could be used at the start of every frame, but this would make the length field almost redundant.)

There are several partial solutions to these problems, but none are without disadvantages. DECNET uses a fixed-length header for each frame and places the length of the frame in that header; in addition, the header has its own CRC. Thus, if an error

Samsung Exhibit 1015, Page 108 of 573

occurs in the length field of the header, the receiver can detect it by the header CRC, which is in a known position. One difficulty with this strategy is that the transmitter must still resynchronize after such an error, since even though the error is detected, the receiver will not know when the next frame starts. The other difficulty is that two CRCs must be used in place of one, which is somewhat inefficient.

A similar approach is to put the length field of one frame into the trailer of the preceding frame. This avoids the inefficiency of the DECNET approach, but still requires a special synchronizing sequence after each detected error. This also requires a special header frame to be sent whenever the length of the next frame is unknown when a given frame is transmitted.

Another approach, for any framing technique, is to use a longer CRC. This at least reduces the probability of falsely accepting a frame if framing errors occur. It appears that this is the most likely alternative to be adopted in practice; a standard 32 bit CRC exists as an option in standard DLCs.

A final approach is to regard framing as being at a higher layer than ARQ. In such a system, packets would be separated by flags, and the resulting sequence of packets and flags would be divided into fixed-length frames. Thus, frame boundaries and packet boundaries would bear no relation. If a packet ended in the middle of a frame and no further packets were available, the frame would be completed with idle fill. These frames would then enter the ARQ system, and because of the fixed-length frames, the CRC would always be in a known place. One disadvantage of this strategy is delay; a packet could not be accepted until the entire frame containing the end of the packet was accepted. This extra delay would occur on each link of the packet's path.

2.5.5 Maximum Frame Size

The choice of a maximum frame length, or maximum packet length, in a data network depends on many factors. We first discuss these factors for networks with variable packet lengths and then discuss some additional factors for networks using fixed packet lengths. Most existing packet networks use variable packet lengths, but the planners of broadband ISDN are attempting to standardize on a form of packet switching called asynchronous transfer mode (ATM) which uses very short frames (called cells in ATM) with a fixed length of 53 bytes. The essential reason for the fixed-length is to simplify the hardware for high-speed switching. There are also some advantages of fixed-length frames for multiaccess systems, as discussed in Chapter 4.

Variable frame length Assume that each frame contains a fixed number V of overhead bits, including frame header and trailer, and let K_{max} denote the maximum length of a packet. Assume, for the time being, that each message is broken up into as many maximum-length packets as possible, with the last packet containing what is left over. That is, a message of length M would be broken into $\lceil M/K_{max} \rceil$ packets, where $\lceil x \rceil$ is the smallest integer greater than or equal to x. The first $\lceil M/K_{max} \rceil - 1$ of these packets each contain K_{max} bits and the final packet contains between 1 and K_{max} bits.

Samsung Exhibit 1015, Page 109 of 573

The total number of bits in the resulting frames is then

$$\text{total bits} = M + \left\lceil \frac{M}{K_{max}} \right\rceil V \tag{2.41}$$

We see from this that as K_{max} decreases, the number of frames increases and thus the total overhead in the message, $\lceil M/K_{max} \rceil V$, increases. In the limit of very long messages, a fraction $V/(V + K_{max})$ of the transmitted bits are overhead bits. For shorter messages, the fraction of overhead bits is typically somewhat larger because of the reduced length of the final packet.

A closely related factor is that the nodes and external sites must do a certain amount of processing on a frame basis; as the maximum frame length decreases, the number of frames, and thus this processing load, increase. With the enormous increase in data rates available from optical fiber, it will become increasingly difficult to carry out this processing for small frame lengths. In summary, transmission and processing overhead both argue for a large maximum frame size.

We next discuss the many factors that argue for small frame size. The first of these other factors is the pipelining effect illustrated in Fig. 2.37. Assume that a packet must be completely received over one link before starting transmission over the next. If an entire message is sent as one packet, the delay in the network is the sum of the message transmission times over each link. If the message is broken into several packets, however, the earlier packets may proceed along the path while the later packets are still being transmitted on the first link, thus reducing overall message delay.

Since delay could be reduced considerably by starting the transmission of a packet on one link before receiving it completely on the preceding link, we should understand why this is not customarily done. First, if the DLC is using some form of ARQ, the CRC must be checked before releasing the packet to the next link. This same argument holds even if a CRC is used to discard packets in error rather than to retransmit them. Finally, if the links on a path have different data rates, the timing required to supply incoming bits to the outgoing link becomes very awkward; the interface between the DLC layer and the network layer also becomes timing dependent.

Let us investigate the combined effect of overhead and pipelining on message delay, assuming that each packet is completely received over one link before starting transmission on the next. Suppose that a message of length M is broken into maximum-length packets, with a final packet of typically shorter length. Suppose that the message must be transmitted over j equal-capacity links and that the network is lightly loaded, so that waiting at the nodes for other traffic can be ignored. Also, ignore errors on the links (which will be discussed later), and ignore propagation delays (which are independent of maximum packet length). The total time T required to transmit the message to the destination is the time it takes the first packet to travel over the first $j - 1$ links, plus the time it takes the entire message to travel over the final link (*i.e.*, when a nonfinal frame finishes traversing a link, the next frame is always ready to start traversing the link). Let C be the capacity of each link in bits per second, so that TC is the number of bit transmission times required for message delivery. Then, assuming that $M \geq K_{max}$,

Samsung Exhibit 1015, Page 110 of 573

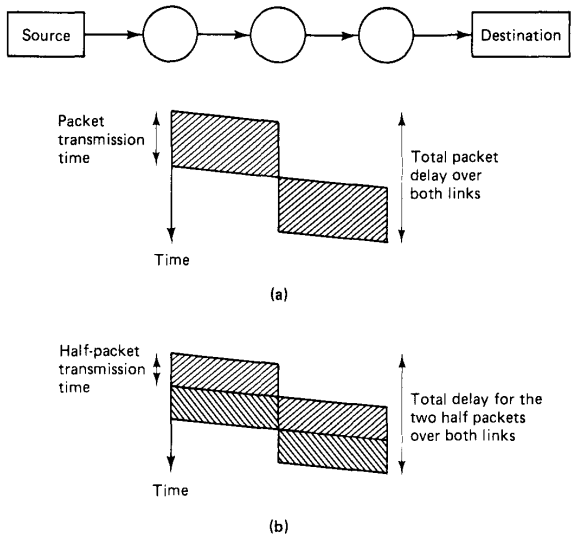


Figure 2.37 Decreasing delay by shortening packets to take advantage of pipelining. (a) The total packet delay over two empty links equals twice the packet transmission time on a link plus the overall propagation delay. (b) When each packet is split in two, a pipelining effect occurs. The total delay for the two half packets equals 1.5 times the original packet transmission time on a link plus the overall propagation delay.

$$TC = (K_{max} + V)(j - 1) + M + \left\lceil \frac{M}{K_{max}} \right\rceil V \tag{2.42}$$

In order to find the expected value of this over message lengths M, we make the approximation that $E\{\lceil M/K_{max} \rceil\} = E\{M/K_{max}\} + \frac{1}{2}$ (this is reasonable if the distribution of M is reasonably uniform over spans of K_{max} bits). Then

$$E\{TC\} \approx (K_{max} + V)(j - 1) + E\{M\} + \frac{E\{M\}V}{K_{max}} + \frac{V}{2} \tag{2.43}$$

We can differentiate this with respect to K_{max} (ignoring the integer constraint) to find the value of K_{max} that minimizes $E\{TC\}$. The result is

$$K_{max} \approx \sqrt{\frac{E\{M\}V}{j - 1}} \tag{2.44}$$

This shows the trade-off between overhead and pipelining. As the overhead V increases, K_{max} should be increased, and as the path length j increases, K_{max} should be reduced. As a practical detail, recall that delay is often less important for file transfers than for other messages, so file transfers should probably be weighted less than other

Samsung Exhibit 1015, Page 111 of 573

messages in the estimation of $E\{M\}$, thus arguing for a somewhat smaller K_{max} than otherwise.

As the loading in a network increases, the pipelining effect remains, although packets typically will have to queue up at the nodes before being forwarded. The effect of overhead becomes more important at heavier loads, however, because of the increased number of bits that must be transmitted with small packet sizes. On the other hand, there are several other effects at heavier loads that argue for small packet sizes. One is the "slow truck" effect. If many packets of widely varying lengths are traveling over the same path, the short packets will pile up behind the long packets because of the high transmission delay of the long packets on each link; this is analogous to the pileup of cars behind a slow truck on a single-lane road. This effect is analyzed for a single link in Section 3.5. The effect is more pronounced over a path, but is mathematically intractable.

Delay for stream-type traffic (such as voice) is quite different from delay for data messages. For stream-type traffic, one is interested in the delay from when a given bit enters the network until that bit leaves, whereas for message traffic, one is interested in the delay from arrival of the message to delivery of the complete message. Consider the case of light loading again and assume an arrival rate of R and a packet length K. The first bit in a packet is then held up for a time K/R waiting for the packet to be assembled. Assuming that the links along the path have capacities $C_1, C_2, \ldots$, each exceeding R, and assuming V bits of framing overhead, a given packet is delayed by $(K+V)/C_i$ on the ith link. When a given packet is completely received at the last node of the network, the first bit of the packet can be released immediately, yielding a total delay

$$T = \frac{K}{R} + (K+V)\sum_i \frac{1}{C_i} \tag{2.45}$$

Assuming that the received data stream is played out at rate R, all received bits have the same delay, which is thus given by Eq. (2.45). We have tacitly assumed in deriving this equation that $(K+V)/C_i \leq K/R_i$ for each link i (*i.e.*, that each link can transmit frames as fast as they are generated). If this is violated, the queueing delay becomes infinite, even with no other traffic in the network. We see then from Eq. (2.45) that T decreases as K decreases until $(K+V)/C_i = K/R$ for some link, and this yields the minimum possible delay. Packet lengths for stream traffic are usually chosen much larger than this minimum because of the other traffic that is expected on the links. As link speeds increase, however, the dominant delay term in Eq. (2.45) is K/R, which is unaffected by other traffic. For 64 kbps voice traffic, for example, packets usually contain on the order of 500 bits or less, since the delay from the K/R term starts to become objectionable for longer lengths.

Note that under light loading, the delay of a session (either message or stream) is controlled by the packet length of that session. Under heavy-loading conditions, however, the use of long packets by some users generally increases delay for all users. Thus a maximum packet length should be set by the network rather than left to the users.

Several effects of high variability in frame lengths on go back n ARQ systems were discussed in Section 2.4.2. High variability either increases the number of packets that must be retransmitted or increases waiting time. This again argues for small maximum

Samsung Exhibit 1015, Page 112 of 573

packet size. Finally, there is the effect of transmission errors. Large frames have a somewhat higher probability of error than small frames (although since errors are usually correlated, this effect is not as pronounced as one would think). For most links in use (with the notable exception of radio links), the probability of error on reasonable-sized frames is on the order of 10^{-4} or less, so that this effect is typically less important than the other effects discussed. Unfortunately, there are many analyses of optimal maximum frame length in the literature that focus only on this effect. Thus, these analyses are relevant only in those special cases where error probabilities are very high.

In practice, typical maximum frame lengths for wide area networks are on the order of 1 to a few thousand bits. Local area networks usually have much longer maximum frame lengths, since the path usually consists of a single multiaccess link. Also, delay and congestion are typically less important there and long frames allow most messages to be sent as a single packet.

Fixed frame length When all frames (and thus all packets) are required to be the same length, message lengths are not necessarily an integer multiple of the packet length, and the last packet of a message will have to contain some extra bits, called fill, to bring it up to the required length. Distinguishing fill from data at the end of a packet is conceptually the same problem as determining the end of a frame for variable-length frames. One effect of the fill in the last packet is an additional loss of efficiency, especially if the fixed packet length is much longer than many messages. Problem 2.39 uses this effect to repeat the optimization of Eq. (2.44), and as expected, the resulting optimal packet length is somewhat smaller with fixed frame lengths than with variable frame lengths. A considerably more important practical effect comes from the need to achieve a small delay for stream-type traffic. As was pointed out earlier, 64 kbps voice traffic must use packets on the order or 500 bits or less, and this requirement, for fixed frame length, forces *all* packets to have such short lengths. This is the primary reason why ATM (see Section 2.10) uses 53 byte frames even though very much longer frames would be desirable for most of the other traffic.

2.6 STANDARD DLCs

There are a number of similar standards for data link control, namely HDLC, ADCCP, LAPB, and SDLC. These standards (like most standards in the network field) are universally known by their acronyms, and the words for which the acronyms stand are virtually as cryptic as the acronyms themselves. HDLC was developed by the International Standards Organization (ISO), ADCCP by the American National Standards Institute (ANSI), LAPB by the International Consultative Committee on Telegraphy and Telephony (CCITT), and SDLC by IBM. HDLC and ADCCP are virtually identical and are described here. They have a wide variety of different options and modes of operation. LAPB is the DLC layer for X.25, which is the primary standard for connecting an external site to a subnet; LAPB is, in essence, a restricted subset of the HDLC options and modes, as will be described later. Similarly, SDLC, which was the precursor of HDLC and ADCCP, is essentially another subset of options and modes.

Samsung Exhibit 1015, Page 113 of 573

HDLC and ADCCP are designed for a variety of link types, including both multiaccess or point-to-point links and both full-duplex links (*i.e.*, both stations can transmit at once) or half-duplex links (*i.e.*, only one station can transmit at a time). There are three possible *modes* of operation, one of which is selected when the link is initialized for use.

The first mode, the *normal response mode* (NRM), is for a master–slave type of link use such as is typical between a computer and one or more peripherals (the word "normal" is of historical significance only). There is a primary or master station (the computer) and one or more secondary or slave stations. The secondary stations can send frames only in response to polling commands from the primary station. This mode is reasonable in multiaccess situations where the polling commands ensure that only one station transmits at a time (see Chapter 4 for a general discussion of multiaccess communication). NRM is also reasonable for half-duplex links or for communication with peripherals incapable of acting in other modes.

The second mode, the *asynchronous response mode* (ARM), is also a master–slave mode in which the secondary nodes are not so tightly restricted. It is not widely used and will not be discussed further.

The third mode, the *asynchronous balanced mode* (ABM), is for full-duplex point-to-point links between stations that take equal responsibility for link control. This is the mode of greatest interest for point-to-point links in data networks. It is also used, in slightly modified form, for multiaccess communication in local area networks. LAPB uses only this mode, whereas SDLC uses only the first two modes.

We first describe the frame structure common to all three modes and all four standards. Then the operation of the normal response mode (NRM) and the asynchronous balanced mode (ABM) are described in more detail. Although ABM is our major interest, the peculiarities of ABM will be understood more easily once NRM is understood.

Figure 2.38 shows the frame structure. The flag, 01^60, is as described in Section 2.5.2, with bit stuffing used within the frame (the flag is not considered as part of the frame proper). One or more flags must separate each pair of frames. The CRC uses the CRC-CCITT generator polynomial $g(D) = D^{16} + D^{12} + D^5 + 1$ as explained in Section 2.3.4; it checks the entire frame (not including the flag). One modification is that the first 16 bits of the frame are inverted in the calculation of the CRC (although not inverted in transmission). Also, the remainder is inverted to form the CRC. At the receiving DLC, the first 16 bits of the frame are similarly inverted for recalculation of the remainder, which is then inverted for comparison with the received CRC. One reason for this is to avoid having an all-zero string (which is common if the link or modem is malfunctioning) satisfy the CRC. Another reason is to avoid satisfying the CRC if a few

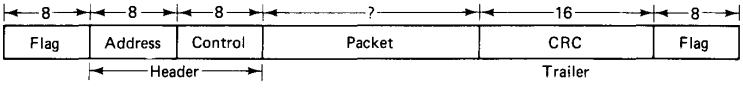


Figure 2.38 Frame structure of standard DLCs. The address, control, and CRC fields can each be optionally extended in length.

Samsung Exhibit 1015, Page 114 of 573

zeros are added or deleted from the beginning or end of the frame. As an option, the 32-bit CRC discussed in Section 2.3.4 can be used. This optional 32-bit CRC is used in the IEEE 802 standard for local area networks.

The address field normally consists of one byte (eight bits). For the normal response mode (NRM), the address is always that of the secondary station; that is, when the primary sends a frame to a secondary, it uses the secondary's address, and when the secondary responds, it uses its own address. Note that this address is not intended as a destination for a session, but simply to distinguish stations for multiaccess use. For point-to-point links, the address field has no natural function, although we shall see later that it has some rather peculiar uses in ABM. There is an option in which the address field is extendable to an arbitrary number of 1 or more bytes; the first bit in a byte is 1 if that byte is the last address byte, and otherwise it is 0.

The layout of the control field is shown in Fig. 2.39. There are three different frame formats: *information*, *supervisory*, and *unnumbered*. The information frames are the actual packet-carrying frames; they normally use go back n ARQ with a modulus of 8. As shown in the figure, the sequence and request numbers SN and RN are part of the control field. The supervisory frames are used to return ARQ information (*i.e.*, RN) either when there are no data packets to send or when a speedy ack or nak is needed. Finally, the unnumbered frames are used for the initiation and termination of a link and for sending various kinds of supplementary control information.

There is an option under which the control field is two bytes long and in which SN and RN are each seven bits long and each use a modulus of 128. Aside from this lengthening of SN and RN, this option is essentially the same as the normal single-byte control field, which is assumed in the following description.

The first bit of the control field distinguishes the information format from the supervisory and unnumbered formats. Similarly, for noninformation formats, the second bit distinguishes supervisory from unnumbered formats. The fifth bit of the control field is the *poll final* (P/F) bit. For the normal response mode, the primary sends a 1 in this bit position to poll the secondary. The secondary must then respond with one or more frames, and sends a 1 in this bit position to denote the final frame of the response. This polling and response function of the P/F bit in NRM is used for supervisory and unnumbered frames as well as information frames. The P/F bit is used in a rather strange way for the asynchronous balanced mode (ABM), as described later.

There are four types of supervisory frames: *receive-ready* (RR), *receive-not-ready* (RNR), *reject* (REJ), and *selective-reject* (SREJ). The type is encoded in the third and

1	2	3	4	5	6	7	8	
0	SN			P/F	RN			Information
1	0	type		P/F	RN			Supervisory
1	1	type		P/F	type			Unnumbered

Figure 2.39 Structure of the control field for the information, supervisory, and unnumbered frame formats.

fourth bits of the control field. All of these have the basic function of returning ARQ information (*i.e.*, all packets with numbers less than RN are acknowledged) and consequently they contain neither a sequence number nor an information field. RR is the normal response if there is no data packet on which to piggyback RN. RNR means in addition that the station is temporarily unable to accept further packets; this allows a rudimentary form of link flow control—that is, if the buffers at one end of the link are filling up with received packets, the other end of the link can be inhibited from further transmission.

The REJ supervisory frame is sent to indicate either a received frame in error or a received frame with other than the awaited sequence number. Its purpose is to improve the efficiency of go back n ARQ, as explained in Section 2.4.2. Similarly, SREJ provides a primitive form of selective repeat. It acks all packets before RN and requests retransmission of RN itself followed by new packets; one cannot provide supplementary information about multiple required retransmissions. This facility can be useful, however, on satellite links, where there is a very small probability of more than one frame error in a round-trip delay. The implementation of REJ and SREJ is optional, whereas RR and RNR are required.

The unnumbered frames carry no sequence numbers and are used for initializing the link for communication, disconnecting the link, and special control purposes. There are five bits in the control field available for distinguishing different types of unnumbered frames, but not all are used. Six of these types will be referred to generically as *set mode* commands. There is one such command for each mode (NRM, ARM, and ABM), and one for each mode using the extended control field. This command is sent by a primary (for NRM or ARM) and by either node (for ABM) to initialize communication on a link. Upon sending or receiving this command, a station sets its sequence and request numbers to zero. The recipient of the command must acknowledge it with an unnumbered frame, either an *unnumbered ack* which agrees to start using the link, or a *disconnected mode* which refuses to start using the link. Similarly, there is a disconnect command which is used to stop using the link; this also requires an unnumbered ack in response.

Figure 2.40 shows the operation of the NRM mode for a primary and two secondaries. Note that the primary initiates communication with each separately and disconnects from each separately. Note also that the P/F bit is set to 1 on each unnumbered frame that requires a response. After a given unnumbered frame with $P = 1$ is sent to a particular secondary, no new unnumbered frame with $P = 1$ may be sent to that secondary until the given frame is acknowledged by a frame with $F = 1$; naturally the given frame may be repeated after a time-out waiting for the ack. This alternation of unnumbered frames with acknowledgments is similar to the stop-and-wait protocol. Section 2.7 explains this similarity carefully and shows that the acknowledgment strategy here contains the same defect as stop-and-wait strategies without numbered acknowledgments.

In the asynchronous balanced mode, the decision was made, in designing these protocols, to have each station act as both a primary and a secondary (thus replacing a simple problem with a more familiar but more difficult problem). When a station sends a primary frame (in its role as a primary station), it uses the other station's address in the address field, and the P/F bit is interpreted as a poll bit. When a station sends

Samsung Exhibit 1015, Page 116 of 573

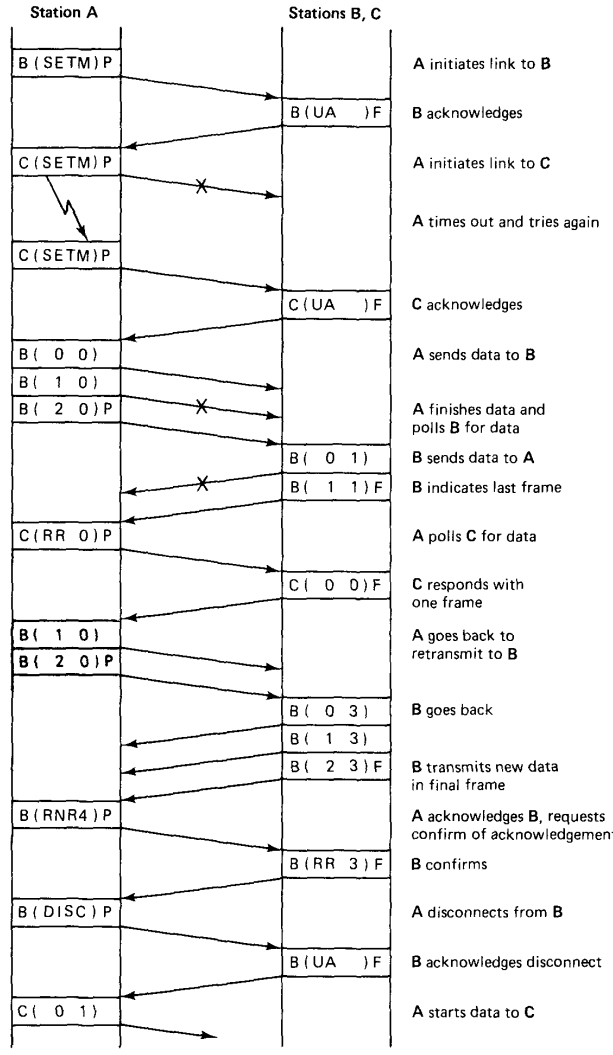


Figure 2.40 Normal response mode (NRM) operation of primary A with two secondaries, B and C. SETM refers to the command setting, the NRM mode with $SN = 0$ and $RN = 0$ at both stations. The address of each frame is given first, followed by (SN,RN) for information frames, (type,RN) for supervisory frames, and (type) for unnumbered frames; this is followed by a P/F bit if set to 1. Note that A uses different SN and RN values for B and C.

Samsung Exhibit 1015, Page 117 of 573

a secondary frame (in its role as a secondary), it uses its own address, and the P/F bit is interpreted as a final bit. In customary operation, all information frames and all unnumbered frames requiring acks are sent as primary frames; all unnumbered frames carrying acks or responses, and most supervisory frames (which ack primary data frames), are sent as secondary frames.

What this means in practice is that the P/F bit is either P or F, depending on the address field. This bit, in conjunction with the address field, has three possible values: $P = 1$, requiring a response from the other station acting in a secondary role; $F = 1$, providing the secondary response, and neither. Given the somewhat inadequate performance of this poll/final alternation, this modified use of the P/F bit was an unfortunate choice.

Figure 2.41 shows a typical operation of ABM. The most interesting part of this is the next-to-last frame sent by station A; this is a supervisory frame sent with $P = 1$, thus requiring an acknowledgment. Since A has signified unwillingness to accept any more packets, a response from B signifies that B knows that packets up to three inclusive have been accepted and that no more will be accepted. Thus, upon receiving the acknowledgment of this, A can disconnect, knowing that B knows exactly which packets from B have been accepted; A also knows which of its packets have been accepted, so the disconnect leaves no uncertainty about the final condition of any packets at either A

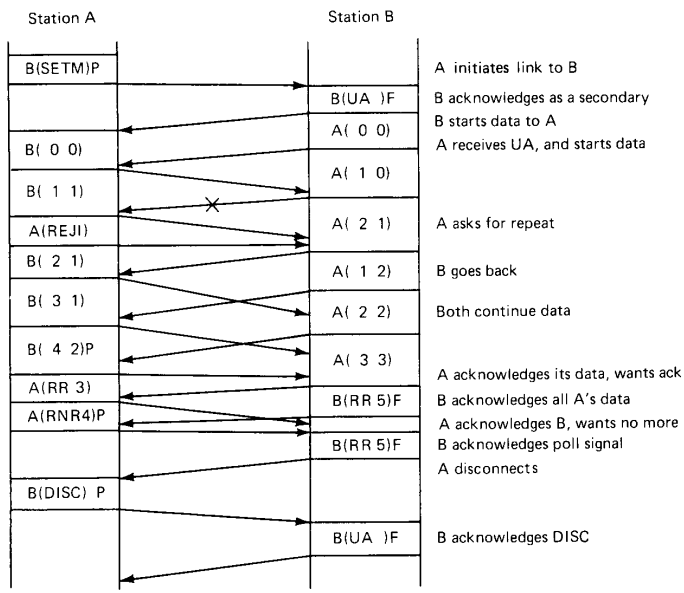


Figure 2.41 Asynchronous balanced mode (ABM) of operation. SETM refers to the set ABM command, which initiates the link with SN and RN at both sides equal to 0. Note the use of addresses in different types of frames.

Samsung Exhibit 1015, Page 118 of 573

or B. Note that this "ideal" disconnect requires some previous agreement (not specified by the standard) between A and B; in particular, A must send RNR, with $P = 1$ and get an ack before disconnecting—this lets B know that A did not go back to accepting packets again temporarily before the disconnect. This "ideal" disconnect does not work, of course, if the line is being disconnected because of a malfunction; in this case, the idea of an orderly sequence of RNR, ack, disconnect, and ack is unrealistic.

Another type of unnumbered frame is "frame reject." This is sent in response to a received frame with a valid CRC but otherwise impossible value (such as exceeding the maximum frame length, being shorter than the minimum frame length, having an RN value not between $SN_{\min}$ and $SN_{\max} - 1$, or having an impossible control field). Thus, either an undetectable error has occurred or one of the stations has malfunctioned. Such an error is outside the realm of what the ARQ is designed for, and it should be reported at each station. The usual response to a "frame reject" is for the opposite station either to reinitialize the link with a set mode command or send an unnumbered "reset" command. The first resets RN and SN at each station to 0, and the second sets SN to 0 at the station that sent the faulty frame and RN to 0 at the receiving station. Both stations will again start to send packets (starting from the first packet not acked), but because of the resetting, some packets might arrive more than once and out of order. This is not a fault of the protocol, but a recognition that node malfunctions and undetectable errors cannot be dealt with.

2.7 INITIALIZATION AND DISCONNECT FOR ARQ PROTOCOLS

The discussion of ARQ in Section 2.4 assumed throughout that the protocol was correctly initialized with no frames traveling on the link and with sequence and request numbers equal to 0 at both ends of the link. Initialization is a general problem for protocols involving communication between two nodes, and arises, for example, in transport protocols and many special-purpose protocols. We shall soon see that initialization and disconnection are intimately related.

One might think that the problem of initializing an ARQ protocol is trivial—one might imagine simply initializing the DLC at each node when the link is first physically connected and then using the ARQ protocol forever. Unfortunately, if the link fails or if one or both nodes fail, initialization is more complicated. We first discuss the problem in the presence of link failures alone and then consider the case in which nodes can fail also.

2.7.1 Initialization in the Presence of Link Failures

It was shown in Section 2.4 that standard ARQ protocols operate correctly (assuming correct initialization) in the presence of arbitrary link delays and errors. Thus, even if the link fails for some period of time, the ARQ protocol could continue attempting packet transmission, and those packets not received correctly before the failure would be received after the failure. The difficulty with this point of view is that when a link fails for a protracted period of time, it is necessary for the transport and/or network layers to

Samsung Exhibit 1015, Page 119 of 573

take over the problem of reliable end-to-end delivery, using alternative paths for those packets that were not delivered over the failed link. When the failed link eventually returns to operation, the higher layers set up new paths through the link, thus presenting the link with a completely new stream of packets.

We see then that the operation of a link protocol over time is properly viewed as being segmented into an alternation of up and down periods. The DLC must be properly initialized at the beginning of each up period, whereas at the end of an up period, there may be packets still in the process of communication that have been accepted into the DLC for transmission at one node but not yet received and released at the opposite node. Stated more mathematically, correct operation of a DLC means that in each up period, the string of packets released to the higher layer at one node must be a prefix of the string of packets accepted for transmission from the higher layer at the opposite node.

2.7.2 Master–Slave Protocol for Link Initialization

The careful reader will have observed that there is a problem in defining an up period carefully; in particular, the nodes at opposite ends of a link might not agree at any given instant about whether the link is up or down. Rather than resolving this issue in the abstract, we present a simple protocol for initializing and disconnecting the link. The protocol is a master–slave protocol where one node, say A, is in charge of determining when the link should be up or down. Node A then informs node B at the other end of the link whenever it determines that the link has changed from down to up or up to down.

The sequence of messages (for this protocol) that go from A to B is then simply an alternating sequence of initialize and disconnect messages (see Fig. 2.42). In this discussion we ignore the data messages that are sent during up periods and simply focus on the messages of the initialization protocol. We view these messages as using a stop-and-wait protocol, using modulo 2 sequence numbering. Thus, the initialize messages (denoted INIT) correspond to frames with $SN = 1$, and the disconnect messages (denoted DISC) correspond to frames with $SN = 0$. The acknowledgments sent by node B will be denoted ACKI and ACKD. In terms of stop-and-wait ARQ, ACKI corresponds to $RN = 0$ [*i.e.*, the INIT message ($SN = 1$) has been correctly received and DISC ($SN = 0$) can be sent when desired by A].

When node A sends INIT, it waits for the acknowledgment ACKI and continues to send INIT on successive time-outs until receiving ACKI. Node A is then initialized (as described below) and can send and receive data packets until it decides to disconnect, at which time it sends DISC (repeatedly if need be) until receiving ACKD. The decision to disconnect (after A is initialized and up) corresponds to the acceptance of a new packet from the higher layer in ordinary stop and wait. The decision to initialize (after A has received ACKD and is down) has the same interpretation. Note that these decisions to try to initialize or disconnect are not part of the initialization protocol, but rather should be viewed either as coming from a higher layer or as coming from measurements of the link. The decisions to actually send the messages INIT or DISC are part of the protocol, however, and rely on having already received an acknowledgment for the previous message.

Samsung Exhibit 1015, Page 120 of 573

From our analysis of the stop and wait protocol, we know that each message from A is accepted once and only once in order. There is also a strict alternation of events between A and B (*i.e.*, B accepts INIT, A accepts ACKI, B accepts DISC, A accepts ACKD, etc.). Now assume that A regards the link as up from the time that it receives ACKI from B until it sends DISC to B (see Fig. 2.42). Similarly, B regards the link as up from the time that it receives INIT from A until it receives DISC from A. Each node sends and accepts data packets only when it regards the link as up. Assuming that frames stay in order on the links, we see that the link from A to B must be free of data when a DISC message reaches B, and it must stay free until the next up period starts at A. Similarly, the link from B to A must be free of data when ACKD reaches A and must stay free until the next up period starts at B (see Fig. 2.43). It follows that the link is free of data in both directions from the time when an ACKD message reaches A until the next INIT message reaches B. Assuming that A initializes its ARQ protocol on sending the INIT message and that B initializes on receiving the INIT message, we see that the ARQ is correctly initialized.

A peculiar feature of this protocol is that once node A starts to initialize, there is no way to abort the initialization before receiving ACKI from node B (see Problem 2.40). Naturally, node A can start to disconnect as soon as ACKI is received, thus sending no data, but A must continue to send INIT until receiving ACKI. The protocol could be complicated to include a message to abort an attempted initialization, but the example described in Fig. 2.44 should make the reader wary of ad hoc additions to protocols. A similar issue is that node B might wish to refuse to initialize (which would be handled by the "disconnected mode" message in HDLC). The cleanest way to handle this is for node B to acknowledge A's INIT message, but to use an extra control field to request A to disconnect immediately without sending data.

The initialization protocol used in the normal response mode of HDLC (see Fig. 2.40) behaves in almost the same way as the initialization protocol above. The SETM unnumbered frame corresponds to INIT, the DISC unnumbered frame corresponds to

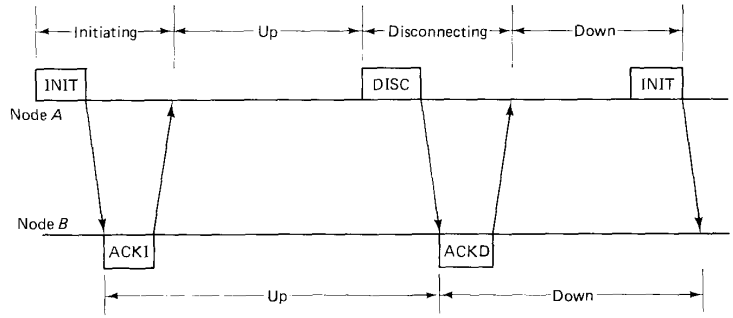


Figure 2.42 Initialization and disconnection protocol. Data are transmitted only in up periods and the ARQ is initialized at the beginning of an up period. Note that node A cycles between periods of initiating, being up, disconnecting, and being down.

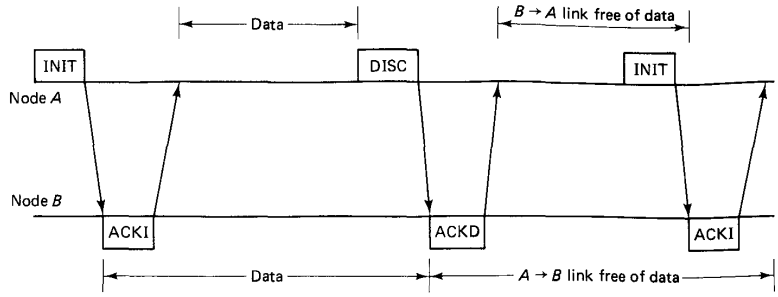


Figure 2.43 Illustration of periods when link is free of data in each direction. Note that the link is free in both directions both when A sends INIT and when B receives it.

DISC, and the ACK unnumbered frame corresponds to both ACKI and ACKD; that is, the unnumbered acks in HDLC fail to indicate whether they are acking initialize or disconnect frames. Figure 2.44 illustrates how this can cause incorrect operation. Node A times-out twice trying to initialize a link between A and B. Finally, on receiving an ack, it initializes the ARQ and sends data packet D with $SN = 0$. After timing out waiting for an acknowledgment, node A decides that the link is not operational, sends a disconnect, and the higher layers send D over some other path. An ack from one of the earlier SETM messages then arrives at A and is viewed as an ack to DISC, so A tries to initialize the link again. The third old ack then arrives, A sends a new data packet D', again initialized with $SN = 0$; D' is lost on the link, but an ack for the old packet D arrives; packet D' then never reaches node B and the ARQ system has failed.

This conceptual error in HDLC does not seem to have caused many problems over the years. The reason for this appears to be that in most networks, link delays are not arbitrary and are easily determined. Thus it is easy to set time-outs (for retransmission of unnumbered frames) large enough to avoid this type of failure. Even if the time-outs were set very poorly, it would be highly unlikely for a node to decide to disconnect and

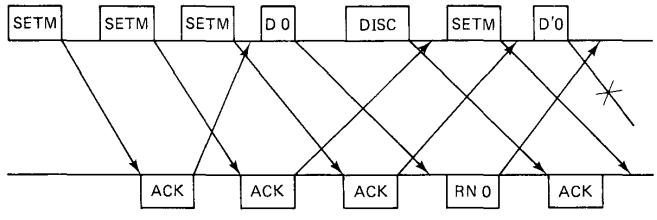


Figure 2.44 Example of HDLC in which node A attempts to initialize the link three times, due to delayed acknowledgments from B. The delayed acks are falsely interpreted as being acks for a subsequent disconnect and initialization. This in turn causes a failure in the data packets.

Samsung Exhibit 1015, Page 122 of 573

then to try to reinitialize over a time span comparable to the link delay. Despite the lack of practical problems caused by this error, however, this is a good example of the danger of designing protocols in an ad hoc fashion.

In the initialization and disconnect protocol just discussed, the process of disconnecting is a critical part of the protocol, and can be viewed as the first part of a new initialization of the link. In fact, if the link fails, the message DISC cannot be sent over the link until the link has recovered; it is this message, however, that guarantees that the next INIT message is not an earlier copy of an earlier attempt to initialize. As seen in Section 2.6, initialization and disconnection are often used on a multidrop line to set up and terminate communication with a given secondary node in the absence of any link failures, and in these applications, the DISC message is really a signal to disconnect rather than part of the initialization process.

Some readers may have noticed that the stop-and-wait protocol that we are using for initialization itself requires initialization. This is true, but this protocol does not require reinitialization on each link failure; it requires initialization only once when the link is physically set up between the two nodes. Recall that the ARQ protocol requires reinitialization on each link failure because of the need to break the packet stream between successive up periods on the link.

2.7.3 A Balanced Protocol for Link Initialization

It is often preferable to have an initialization protocol that is the same at both ends of the link. At the very least, there is a conceptual simplification in not having to decide which node is master and ensuring that each node uses the appropriate protocol. The essential idea of a balanced initialization protocol is to use two master–slave protocols, with node A playing the master for one and node B playing the master for the other. The only new requirement is to synchronize the two protocols so that they have corresponding up and down periods.

To simplify this synchronization somewhat, we assume that each INIT or DISC message from a node (acting as master) also contains a piggybacked ACKI or ACKD for the master–slave protocol in the opposite direction. Thus a single received message can affect both master–slave protocols, and we regard the ACK as being acted on first. ACK messages can also be sent as stand alone messages when there is no need to send an INIT or DISC but an INIT or DISC is received.

We recall that in the master–slave initialization protocol from A to B, the link was considered up at A from the receipt of ACKI until an external decision to disconnect (see Fig. 2.42). The link was regarded as up at B from the receipt of INIT until the receipt of the next DISC. For the balanced protocol, a node regards the link as *up* if it is up according to both the $A \rightarrow B$ protocol and the $B \rightarrow A$ protocol. A node initializes its ARQ at the beginning of an up period and sends and acknowledges data during this period. Similarly, a node regards the link as down if it is down according to both the $A \rightarrow B$ and $B \rightarrow A$ protocols.

If the link is regarded as up at node A (according to both master–slave protocols), either an external decision at A or a DISC message from B causes A to leave the up

Samsung Exhibit 1015, Page 123 of 573

period and start to disconnect by sending a DISC message. As a special case of this, if a DISC message from B carries a piggybacked ACKI, the ACKI can cause node A to regard the link as up, in which case the DISC message immediately ends the up period (see Problem 2.40). Similarly, if the link is regarded as down at node A, either an external decision or an INIT message from B causes A to leave the down period and start to initialize by sending an INIT message. Again, if an INIT message from B is accompanied by a piggybacked ACKD, the ACKD can cause A to regard the link as down, and the INIT message immediately ends the down period.

Figure 2.45 illustrates the operation of this balanced protocol. Each of the master–slave protocols operate as described in the preceding section, and in particular, each node, in its role as slave, acks each INIT and DISC message from the other node; also each node, in its role as master, continues to retransmit INIT and DISC messages until acknowledged. The synchronizing rules of the preceding paragraph simply limit when external decisions can be made about starting to initialize and disconnect. For this reason, the safety of the balanced protocol follows immediately from the correctness of the master–slave protocols. The safety of the protocol can be verified, using the same assumptions as in Section 2.4, simply by verifying that when one node starts to initialize (or disconnect), the other node will eventually start to do the same (if it did not start earlier), and both nodes will eventually reach the up state (or the down state in the case of disconnect). See Problem 2.40 for details.

This protocol has the same peculiar problems as the master-slave protocol in that a node might be sending INIT messages with no intention of transferring data; this could occur either because of a link failure while in the process of initializing or because of sending an INIT message in response to the opposite node sending an INIT message. These problems are best treated by using extra fields for control information and recognizing that the protocol here is primarily designed for synchronizing the two nodes.

The initialization part of this balanced protocol (*i.e.*, sending INIT, waiting for both ACKI and INIT from the other side, and responding to INIT with ACKI) is often

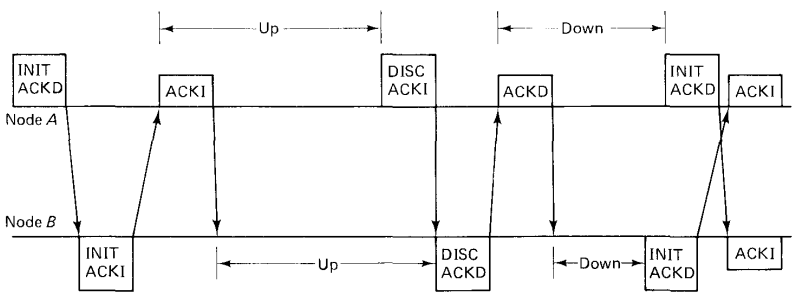


Figure 2.45 Example of balanced initialization and disconnect protocol. Either node can start to initialize from the down state or start to disconnect from the up state and the other node will follow.

called a three-way handshake. The reason for this terminology is that the ACKI from the second node is usually piggybacked on the INIT from the second node, making a three-message interchange. It is always possible that both sides start to initialize independently, however, and then all four messages must be sent separately (as seen on the right-hand side of Fig. 2.45). We shall see a closely related type of three-way handshake in transport protocols, but the problem is more complex there because of the possibility of packets traveling out of order.

2.7.4 Link Initialization in the Presence of Node Failures

From the viewpoint of a network, a node failure is similar to (but, as we shall see, not the same as) the failures of all the adjacent links to that node; in both cases, no communication is possible passing through that node or going to or from that node. For our purposes, a node is modeled as either working correctly or being in a failure state; in the failure state, no input is accepted, no output is produced, and no operations occur (except for a possible loss of state information). We view each node as alternating in time between correct operation and the failure state (typically, with very long periods between failures). Thus the question of interest here is that of reinitialization after a node failure. Another interesting class of problems, which we do not address here, is that of intermittent faulty operation, or even malicious operation, at a node. This is really a network layer problem and is treated in [Per88].

If a node maintains its state information during a failure period, we view the data at the node as being lost or retransmitted over other paths. Thus when the node returns to correct operation, the adjacent links must be reinitialized and the problem is really the same as if all the adjacent links had failed. If a node loses its state information during a failure period, the problem of reinitialization is much more difficult than that of adjacent link failures.

To illustrate the problem of initialization after loss of state information, consider the master–slave initialization protocol of Section 2.7.2 and suppose that node A, the master node, fails and recovers several times in rapid succession. Assume also that when A recovers, the initialization protocol starts in the down state (*i.e.*, in the state for which ACKD was the last received message); Problem 2.41 looks at the opposite case where it starts in the up state. Each time node A recovers, it receives a new sequence of packets to transmit to node B after initializing the (A, B) link.

Figure 2.46 illustrates what can happen if the time between failure and recovery is on the order of the round-trip delay. Node B has no way of knowing that A has failed (other than the messages of the initialization protocol, which are supposed to initialize the link after a failure). In the figure, A sends just the INIT message and then fails. When it recovers, it sends INIT again, and receives the ack from the previous INIT; this completes the initialization and allows A to send data packet D with sequence number 0. Node A then fails again, and on recovery gets the responses from B from the previous period when A was functioning.

The failure to initialize correctly here is not due to the particular protocol and is not due to the particular starting state. In fact, incorrect operation can occur for any

Samsung Exhibit 1015, Page 125 of 573

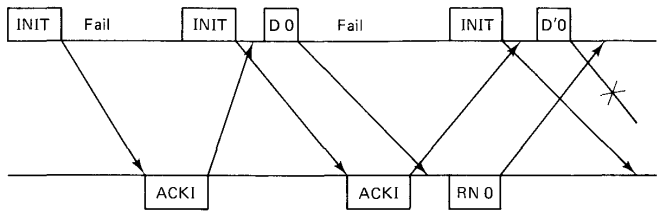


Figure 2.46 Example of a sequence of node failures, with loss of state, that causes a failure in correct initialization. Each time the node recovers from a failure, it assumes that ACKD was the last protocol message received, thus allowing acknowledgments from earlier initialization periods to get confused with later periods.

protocol and any starting state, assuming that there is no upper bound on link delay (see [Spi89] and [LMF88]). Problem 2.41 illustrates this phenomenon.

There are several solutions to the problem of initialization after a node failure. The first is to provide nodes with a small amount of nonvolatile storage. For example, the initialization protocols in this section can operate correctly with only a single bit of nonvolatile storage per adjacent link. If the links have a maximum propagation delay, another solution is simply to use sufficiently long time-outs in the initialization protocol to avoid the problems above (see [Spi89]). Yet another type of solution, which we explore later in terms of transport protocols, is to use a pseudo-random-number generator to avoid confusion between different periods of correct operation. We shall see that such protocols are very robust, although they have the disadvantage that there is a small probability of incorrect operation.

2.8 POINT-TO-POINT PROTOCOLS AT THE NETWORK LAYER

The major conceptual issues at the network layer, namely routing and flow control, involve algorithms that are distributed between all the nodes of the network; these issues are treated in Chapters 5 and 6, respectively. There are a number of smaller issues, however, that simply involve interactions between a pair of nodes. One of these issues is the transfer of packets between adjacent nodes or sites. At the network layer, this involves being able to distinguish packets of one session from another and being able to distinguish different packets within the same session. These issues are of particular concern between a source–destination site and the adjacent subnet node because of the need for standardization between users of the network and the subnet.

In the following subsections, we first treat the problem of addressing and session identification from a general viewpoint and give several examples to illustrate the range of possibilities. We then discuss packet numbering with a brief description of how this relates to flow control and to error recovery at the network layer. Finally, we discuss the X.25 network layer standard.

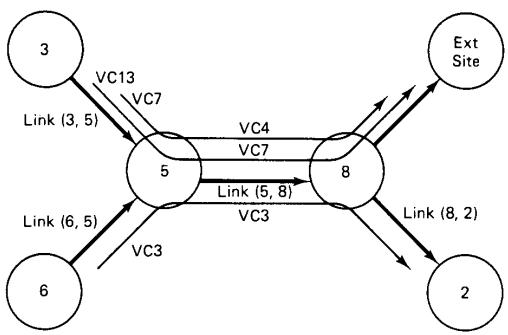
Samsung Exhibit 1015, Page 126 of 573

2.8.1 Session Identification and Addressing

The issue of session identification is relevant at the network layer. A packet received by a node must contain enough information for the node to determine how to forward it toward its destination. The brute-force approach to this problem is for the header of each packet to contain identification numbers for both the source and destination sites and additional identification numbers to identify the session within each site. This approach has great generality, since it allows different packets of the same session to travel over different paths in the subnet; it also requires considerable overhead.

If the network uses virtual circuits, far less overhead is required in the packet headers. With virtual circuits, each (one way) session using the network at a given time has a given path through the network, and thus there is a certain set of sessions using each link. It is helpful to visualize each link as being shared by a set of "virtual channels," distinguished by numbers. When a new session is set up, a path is established by assigning, on each link of the path, one unused virtual channel to that session. Each node then maintains a table mapping each busy incoming virtual channel on each link onto the corresponding outgoing virtual channel and link for the corresponding session (see Fig. 2.47). For example, if a given session has a path from node 3 to 5 to 8, using virtual channel 13 on link (3,5) and virtual channel 7 on link (5,8), then node 5 would map [link (3,5), virtual channel 13] to [link (5,8), virtual channel 7], and node 8 would map [link (5,8), virtual channel 7] to the access link to the appropriate destination site with a virtual channel number for that link. The nice thing about these virtual channels is that each link has its own set of virtual channels, so that no coordination is required between the links with respect to numbering.

With this virtual channel strategy, the only thing that must be transmitted in the packet header on a link is an encoding of the virtual channel number. The simplest way



Node 5 table
(3, 5) VC13 → (5, 8) VC7
(3, 5) VC7 → (5, 8) VC4
(6, 5) VC3 → (5, 8) VC3

Figure 2.47 Use of virtual channel (VC) numbers to establish virtual circuits in a network. Each packet carries its VC number on each link and the node changes that number (by a table representing the virtual circuit) for the next link on the path.

Samsung Exhibit 1015, Page 127 of 573

to do this, of course, is to represent the virtual channel number as a binary number in a fixed field of the header. Because of the limited capacity of the link, there is some natural upper bound on the number of sessions a link can handle, so there is no problem representing the virtual channel number by a fixed number of binary digits.

Later, other ways of representing virtual channel numbers will be discussed, but for now note that any method of distinguishing sessions must do something equivalent to representing virtual channel numbers. In other words, a given link carries data for a number of different sessions, and the transmitted bits themselves must distinguish which data belong to which session. The packet header might supply a great deal of additional information, such as source and destination identities, but the bare minimum required is to distinguish the sessions of a given link. We now look at how this representation is handled in TYMNET.

Session identification in TYMNET TYMNET is a network developed in 1970 primarily to provide access for a time-sharing computer service, but expanded rapidly to allow users to communicate with their own computers. Since the terminals of that period usually transmitted only one character at a time and usually waited for an echo of that character to come back before transmitting the next character, the protocols were designed to work well with very short messages consisting of one or several characters. Although the network never received much academic publicity, many of the ideas first implemented there are now standard network practice. The frames are very short, with a maximum length of 66 bytes; the frames have the format shown in Fig. 2.48.

The frame header of 16 bits is broken into five bits for a synchronization string, five bits for a word count, three bits for SN, and three bits for RN. The word count gives the number of 16 bit words in the frame, not counting the header and trailer. Since

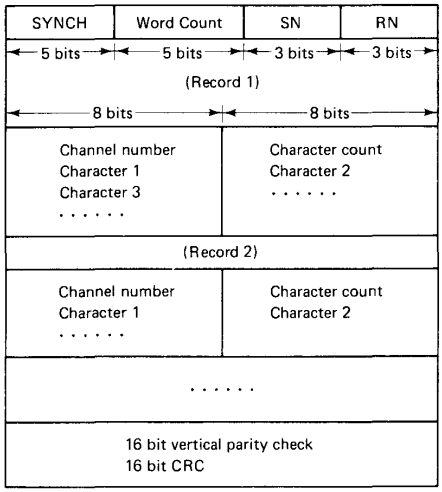


Figure 2.48 Frame structure in TYMNET. The first 16 bits is a frame header, and the last 32 bits is a frame trailer. Note that data from multiple sessions are included in a single frame.

Samsung Exhibit 1015, Page 128 of 573

the nodes were implemented with 16 bit minicomputers, all the fields of the frame are organized in 16 bit (two byte) increments. Note that the 5 bit word count can represent up to 31 words of two bytes each, or 62 bytes. Thus, restricting the frames to be multiples of 16 bits serves to reduce (slightly) the overhead required for framing. The SN and RN fields provide conventional go back n ARQ with a modulus of $m = 8$. The last four bytes (the frame trailer) provide error detection, using both a 16 bit CRC and 16 vertical parity checks. Thus, somewhat more error detection is provided than with the standard 16 bit CRC, but slightly less than with a 32 bit CRC.

Each record within a frame corresponds to data from a different session. A record starts with a byte giving the virtual channel number, which identifies the session. The next byte tells how many characters of data are contained in the record. If the number is odd, the last byte of the record is just a fill character, so that the next record can start on a word boundary. A record here can be thought of as a very short packet, so that the channel number and character count correspond to the packet header. Thus, the packets are so short that a number of packets are normally grouped into one frame so as to avoid excessive frame overhead. These packets are somewhat different from the usual concept of a packet in that they do not necessarily contain the same number of characters from one link to the next on their path.

This type of strategy is usually referred to as *statistical multiplexing*. This means that the link is multiplexed between the virtual channels, but multiplexed on a demand basis. Conventional packet switching, in which there is just one packet per frame, is another form of statistical multiplexing, but conventionally, one refers to that as packet switching rather than statistical multiplexing. When a frame is assembled for transmission at a node, the sessions are served in round-robin order, taking some number of characters from each session that has characters waiting.

Session identification in the Codex networks The Codex networks use quite a similar statistical multiplexing strategy, again with multiple sessions transmitting characters within the same frame. Here, however, a more efficient encoding is used to represent the virtual channel number and the amount of data for that session. In assembling a frame, the sessions are again served in sequential order, starting with virtual channel 1. Instead of sending the virtual channel number, however, an encoding is sent of the gap (*i.e.*, the number of idle virtual channels) between the number of the session being served and the previous session served (see Fig. 2.49). Thus, when all sessions have characters to send, these gaps are all zero, and if many but not all of the sessions have characters, most of the gaps are small. Thus, by encoding small gaps into a small number of bits, and large gaps into more bits, very high efficiency is maintained when large numbers of sessions are active. On the other hand, if only a small number of sessions are active, but have a large amount of data each, then only a small number of records will be in a frame, and again high efficiency is maintained.

Because of implementation issues, all of the fields in the Codex frames are multiples of four bit "nibbles." Thus, the smallest number of bits that can be used to represent a gap for identifying a session is one nibble. The method that identifies the number of characters in a record uses a special start record character, 0000, to start

Samsung Exhibit 1015, Page 129 of 573

Unary-binary gap encoding:	$0 \rightarrow 10$
	$1 \rightarrow 11$
	$2 \rightarrow 010$
	$3 \rightarrow 011$
	$4 \rightarrow 0010$
	.....
Start record character: 0000	
Sample encoding of data:	
(0000)(10)(x_1)(x_2)(x_3)(x_4)(0000)(10)(x_1)(x_2)	
(0000)(11)(x_1)(0000)(0010)(x_1)(x_2)(x_3)...	

Figure 2.49 Efficient encoding for virtual channels and lengths of records. In the example shown, there are four characters x_1, x_2, x_3, x_4 for VC1, two characters for VC2, one for VC4, and three for VC9. Each record starts with 0000 to indicate the start of new record (where 0000 cannot occur in data characters); this is followed by the encoding of the gap between virtual channels with data. The Codex protocols are similar, but gaps are encoded into multiples of four bits. Parentheses are for visual clarity.

each new record. The characters for each session first go through a data compressor that maps highly probable characters into one nibble, less likely characters into two nibbles, and least likely characters into three nibbles. This is done adaptively, thus automatically using the current character statistics for the session. The 0000 nibble is reserved to indicate the start of a new record. Thus, the "packet header" consists of first a nibble to indicate a new packet (record), and next one or more nibbles representing the new session number. Under heavy loading, the packet header is typically just one byte long.

This strategy of encoding the difference between successive virtual channel numbers can also be used in conventional packet switched networks. There are two advantages in this. First, the packet overhead is reduced under heavy loading, and second, serving the sessions on a link in round-robin order tends to promote fairness in the service that each session receives.

2.8.2 Packet Numbering, Window Flow Control, and Error Recovery

In datagram networks, packets might be dropped and might get out of order relative to other packets in the same session. Thus, to achieve reliable communication, it is necessary to provide some way to identify successive packets in a session. The most natural and usual way of doing this is to number successive packets modulo 2^k for some k (*i.e.*, to provide a k-bit sequence number). This sequence number is placed in the packet header if the network layer modules at the source and destination have the responsibility for reordering the packets and retransmitting dropped packets. In many networks, for reasons we discuss later, the transport layer takes the responsibility for reordering and retransmission, and in that case the sequence number above is placed in the transport header. Sometimes sequence numbers are used at the network layer, at the internet sublayer, and at the transport layer.

For virtual circuit networks, there is also a need to number the packets in a session, but the reason is less obvious than for datagram networks. In our discussion of data link control, we saw a number of ways in which packets might get lost or arrive at the destination containing errors. First, a sequence of channel errors might (with very low

Samsung Exhibit 1015, Page 130 of 573

probability) lead to an incorrect frame that still satisfies the CRC at the DLC layer. Second, errors made while data packets pass through a node are not checked by the CRC at the DLC layer. Third, if a link on the session path fails, there is uncertainty about how many packets were successfully transmitted before the failure. Finally, if a node fails, the packets stored at that node are typically lost. The error events described above are presumably quite rare, but for some sessions, correct transmission is important enough that error recovery at the network or transport layer is necessary.

Error recovery From a conceptual standpoint, error recovery at either the network or transport layer is quite similar to ARQ at the DLC layer. Each packet within a given session is numbered modulo $m = 2^k$ at the source site, and this sequence number SN is sent in the packet header at the network or transport layer. The destination site receives these packets and at various intervals sends acknowledgment packets back to the source site containing a request number RN equal (mod m) to the lowest-numbered yet-unreceived packet for the session. The receiver can operate either on a go back n basis or on a selective repeat basis.

It is important to understand both the similarity and the difference between the problem of end-to-end error recovery and that of error detection and retransmission at the DLC layer. End-to-end error recovery involves two sites and a subnet in between, whereas at the DLC layer, there are two nodes with a link in between. The sequence numbering for end-to-end error recovery involves only the packets (or bytes, messages, etc.) of a given session, whereas at the DLC layer, all packets using the link are sequentially numbered. At the DLC layer, we assumed that frames could get lost on the link, and similarly here we are allowing for the possibility of packets being thrown away or lost. At the DLC layer, we assumed variable delay for frame transmission, and here there is certainly widely variable delay. (Note that variable delay on a link is somewhat unnatural, but we assumed this so as to avoid worrying about processing delays.) The only real difference (and it is an important difference) is that frames must stay in order on a link, whereas packets might arrive out of order in a network.

Recall that datagram networks typically make no attempt to keep packets of the same session on the same path, so that out-of-order packets pose a generic problem for datagram networks. One would think that virtual circuit networks would be guaranteed to keep packets of the same session in order, but there are some subtle problems even here concerned with setting up new paths. If a link fails and a new path is set up for a session, the last packets on the old path might arrive out of order relative to the first packets on the new path. This problem can be avoided by a well-designed network layer, but for simplicity, we simply assume here that we are dealing with a datagram network and that packets can get out of order arbitrarily on the network.

Since packets can get lost, packets must sometimes be retransmitted end-to-end, and since packets can be delayed arbitrarily, a packet can be retransmitted while the earlier copy of the packet is lurking in the network; that earlier copy can then remain in the network until the packet numbering wraps around, and the packet can then cause an uncorrectable error. This scenario is perhaps highly unlikely, but it serves to show that end-to-end error recovery at the network or transport layer cannot be guaranteed to

Samsung Exhibit 1015, Page 131 of 573

work correctly if packets are sometimes lost, sometimes get out of order, and sometimes become arbitrarily delayed.

There are three possible ways out of the dilemma described above: first, insist that the network layer use virtual circuits and maintain ordering even if the path for a session is changed; second, make the modulus for numbering packets so large that errors of the foregoing type are acceptably unlikely; and third, create a mechanism whereby packets are destroyed after a maximum lifetime in the network, using a modulus large enough that wraparound cannot occur in that time. It is shown in Problem 2.43 that the required modulus, using selective repeat, must be twice the window size plus the maximum number of packets of a session that can be transmitted while a given packet is alive. In Section 2.9 we discuss how the transport protocol TCP uses a combination of the last two approaches described above.

End-to-end error recovery also requires some sort of end-to-end check sequence such as a supplementary CRC to detect errors either made at the subnet nodes or undetected by the CRC at the link layer. In TCP, for example, this check sequence is 16 bits long and is the one's complement of the one's complement sum of the 16 bit words making up the packet. This type of check sequence is not as good at detecting errors as a CRC, but is easier to implement in software for typical processors. One interesting problem that occurs with end-to-end check sequences is that packets frequently change somewhat when passing from node to node. The primary example of this is the changing virtual channel numbers in the virtual channel addressing scheme discussed in Section 2.8.1. One certainly does not want to recompute the end-to-end check sequence at each node, since this would fail in the objective of detecting node errors.

The solution to this problem is to calculate the end-to-end check sequence before placing values in those fields that can vary from node to node. At the destination site, the check sequence is recomputed, again ignoring the values in those varying fields. This solution leaves the varying fields unprotected except by the link layer. For the example of virtual channel addressing, however, there is a good way to restore this protection. Both end sites for the session know the universal address of the other (from the setup of the session). Thus these addresses can be included in the check sequence, thus protecting against packets going to the wrong destination. Other parameters known to both end sites can also be included in the check sequence. As one example, we discussed the large end-to-end sequence number fields required to number packets in datagram networks. Only a few of the least significant bits are required to reorder the packets over a window, whereas the rest of the bits are required to ensure that very old packets (and very old acknowledgments) get rejected. By transmitting only the least significant bits of the sequence and request numbers, and including the more significant bits in the check sequence, one achieves greater efficiency at no cost in reliability.

Flow control The use of either go back n or selective repeat ARQ for end-to-end error recovery provides a rather effective form of flow control as an added bonus. If we ignore errors and retransmissions, we see that with a window size of n, at most n packets for a given session can be outstanding in the network at a given time. A new

Samsung Exhibit 1015, Page 132 of 573

packet j can be sent only after an acknowledgment is received for packet $j - n$. The effect of this is that as congestion builds up and delay increases, acknowledgments are delayed and the source is slowed down. This is called end-to-end window flow control and is analyzed in Chapter 6. If the destination wishes to receive packets less rapidly, it can simply delay sending the current request number RN; thus this type of flow control can be effective in reducing congestion both in the subnet and at the destination site.

In the discussion of data link control we did not mention this flow control aspect of ARQ. The reason is that a link simply carries a fixed number of bits per second and there is no reason to limit this flow (other than the ability of the receiving node to handle the incoming data). Recall that standard DLCs provide a receive-not-ready supervisory packet as a way to protect the receiving node from too many incoming packets. This mechanism allows speedy acknowledgments, thus preventing unnecessary retransmissions due to time-outs, but also permits the receiving node to exert some flow control over the sending node.

Combining end-to-end acknowledgments with flow control has some limitations. If an acknowledgment does not arrive because of an error event, the packet should be retransmitted, whereas if the ack is delayed due to network congestion, the packet should not be retransmitted, since it would only add to the congestion. Since delays in the subnet are highly variable, it is difficult for time-outs to distinguish between these cases. To make matters worse, the destination might be overloaded and postpone sending acks as a way of preventing further transmissions from the source; this again might cause unnecessary retransmissions due to time-outs.

One partial solution to these problems is to provide the destination with a way to slow down the source without delaying acknowledgments. One way of doing this is by a technique such as the receive-not-ready supervisory frames in the standard data link controls. Such a frame both provides a speedy ack and prevents further transmission. This same technique can be used at the network layer (and is used in the X.25 network layer standard to be discussed shortly). A *permit* scheme is a more refined technique for the same objective. The destination sends two feedback numbers back to the source rather than one. The first, the usual RN, provides the ack function, and the second, a permit, tells the source how many additional packets the destination is prepared to receive. Thus if the permit number is j, the source is permitted to send packets from number RN to $RN + j - 1$ inclusive. In effect, this allows the destination to change the window size with each acknowledgment to the source at will. It allows the destination to send permits for only as many packets as will fit in the available buffers. This permit scheme could be provided in DLCs as well, but it is less useful there because the buffering requirements at a node are relatively modest because of the fixed number of links at a node.

Error recovery at the transport layer versus the network layer We have discussed end-to-end error recovery and its relationship to end-to-end window flow control. We now consider whether error recovery belongs at the transport layer or the network layer. From a practical standpoint, the answer now and in the near future is that error recovery belongs at the transport layer. The ISO transport layer standard known as

Samsung Exhibit 1015, Page 133 of 573

TP4 and the de facto standard TCP both provide for error recovery at the transport layer. Part of the reason for this is the prevalence of internetworking and the large number of networks that do not provide reliable service. The Internet Protocol, which is designed to work with TCP and which connects many universities and research labs, is itself an unreliable protocol, as will be discussed shortly.

The major disadvantage of having unreliable network and internet protocols and error recovery at the transport layer is that the transport layer has no way to distinguish between acknowledgments that are slow because of congestion and acknowledgments that will never arrive because the packet was thrown away. This problem is particularly severe for datagram networks. As a result, when the network becomes congested, packets start to be retransmitted, leading to still greater congestion, more dropped packets, and thus unstable behavior. There are a number of ad hoc approaches to reducing the severity of this problem, but it appears that error recovery at the transport layer will always lead to problems unless such recovery is required only rarely . This in turn will require almost reliable operation in the internet layer and in the various network layers.

2.8.3 The X.25 Network Layer Standard

The X.25 standard was developed by CCITT to provide a standard interface between external sites and subnet nodes. Recall that the physical layer, X.21, of this standard was discussed briefly in Section 2.2, and the DLC layer, LAPB, was discussed in Section 2.6. In X.25 terminology, the external site is referred to as DTE (data terminal equipment) and the subnet node as DCE (data communication equipment), but we shall continue to refer to them as sites and nodes.

The structure of an X.25 data packet is shown in Fig. 2.50. Recall that the DLC layer adds a frame header and trailer to this packet structure before transmission on the link. The third byte of the packet header is very similar to the control byte of the standard DLCs (except that SN and RN here refer to sequence numbers within the session, as discussed above). The control bit, C, is 0 for data packets and 1 for control packets (to be discussed later). This control bit is different from the control bit in the frame header; at the DLC layer, all packets, data and control, are treated indistinguishably, using 0 as the DLC layer control bit. Finally, the "more" bit, M, in the packet header is 1 for nonfinal packets in a message and 0 for the final packet.

The last half of the first byte and the second byte of the packet header is a 12-bit virtual channel number, as discussed in Section 2.7. This number can be different at the two ends of the virtual circuit for a given session. Many subnets use this same packet structure within the subnet, and the virtual channel number can then change from link to link, as described in Section 2.7.

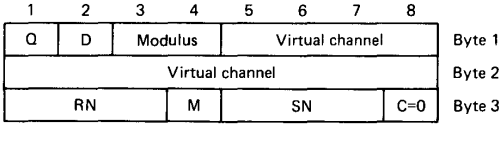


Figure 2.50 Packet header for data in X.25. For a control packet the C bit is 1.

Samsung Exhibit 1015, Page 134 of 573

The first bit, Q, in the packet header (if used at all) has the value 1 for control packets at the transport and higher layers and has the value 0 for control packets at the network layer and for data packets. The next bit, D, indicates whether the acknowledgment (RN) has end-to-end significance or just link significance. If $D = 1$, then RN signifies that the destination site has received all packets for the session previous to RN, that RN is awaited, and that the source is free to send a window of packets from RN on. If $D = 0$, RN signifies that the local subnet node or site has received packets previous to RN, that RN is awaited, and the other site or node is free to send a window of packets from RN on. It is possible for some sessions to use end-to-end significance and others, single-link significance. As discussed above, end-to-end acks are very helpful in providing error recovery. Some sessions, however, do not require error recovery, and some provide it at a higher layer, for example, by using a query–response interaction between the sites. For single-link significance, the acknowledgment function of RN is redundant (the DLC layer guarantees that the packet is received). Thus, RN is used strictly for flow control, and the receiving node delays sending RN until it is prepared to accept a window of packets beyond RN.

The third and fourth bits of the header indicate whether the modulus is 8 or 128. If the modulus is 128, the third byte is extended to two bytes to provide seven bits for SN and seven for RN.

The control packets in X.25 are quite similar to the supervisory frames and unnumbered frames in the standard DLCs. The third byte of the packet header indicates the type of control packet, with the final bit (control bit) always 1 to distinguish it from a data packet. The virtual channel number is the same as in a data packet, so that these control packets always refer to a particular session. There are receive-ready, receive-not-ready, and reject control packets, each of which carry RN; these are used for acknowledgments in the same way as in DLCs. The receive-not-ready packet provides some ability to acknowledge while asking for no more packets, but there is no ready-made facility for permits.

One interesting type of control packet is the call-request packet. After the three-byte header, this packet contains an address field, containing the length of the calling address and called address in the first byte, followed by the addresses themselves. This is used to set up a virtual circuit for a new session in the network. Note that the virtual channel number is of no help in finding the destination; it simply specifies the virtual channel number to be used after the virtual circuit is established. The subnet node, on receiving a call-request packet, sets up a path to the destination and forwards the packet to the destination site. If the recipient at the destination site is willing and able to take part in the session, it sends back a call-accepted packet; when this reaches the originating site, the session is established.

Both the call-request and call-accepted packet contain a facilities field; this contains data such as the maximum packet length for the session, the window size, who is going to pay for the session, and so on. The default maximum packet length is 128 bytes, and the default window size is 2. If the call-request packet specifies different values for maximum packet length or window size, the recipient can specify values closer to the default values, which must then be used. This is a sensible procedure for the external

Samsung Exhibit 1015, Page 135 of 573

sites, but it is somewhat awkward for the subnet. The subnet can specify a maximum packet length and maximum window size to be used by the sessions using the subnet, but it is also desirable to have the ability to reduce window sizes under heavy loading, and this cannot be done naturally with X.25.

If the subnet is unable to set up the virtual circuit (because of heavy loading, for example), or if the recipient is unwilling to accept the call, a clear-request control packet goes back to the initiating site, explaining why the session could not be established.

2.8.4 The Internet Protocol

The Internet is a collection of thousands of networks interconnected by gateways. It was developed under the U.S. Department of Defense Advanced Research Projects Agency (DARPA) support as an effort to connect the myriad of local area networks at U.S. universities to the ARPANET. It now provides almost universal data access within the academic and research communities. The protocol developed to interconnect the various networks was called the Internet Protocol, usually referred to as IP. The backbone functions have since been taken over by the U.S. National Science Foundation (NSF), and the current topology is quite chaotic and still growing rapidly.

The individual networks comprising the Internet are joined together by gateways. A gateway appears as an external site to an adjacent network, but appears as a node to the Internet as a whole. In terms of layering, the internet sublayer is viewed as sitting on top of the various network layers. The internet sublayer packets are called datagrams, and each datagram can pass through a path of gateways and individual networks on the way to its final destination. When a gateway sends a datagram into an individual network, the datagram becomes the body of the packet at the network layer (see Fig. 2.51). Thus the individual network sends the datagram on to the next gateway or to the final destination (if on that network) in the same way that it handles other packets. Assuming that the individual network succeeds in getting the datagram to the next gateway or destination, the individual network looks like a link to the internet layer.

One of the goals of the Internet project was to make it as easy as possible for networks to join the Internet. Arbitrary networks were allowed to join the Internet, even if they allowed datagrams to be delayed arbitrarily, lost, duplicated, corrupted by noise, or reversed in order. This lack of reliability underneath was one of the reasons for making IP a datagram protocol; the problem of error recovery was left for the transport layer. Note that if an individual network happens to be a virtual circuit network, this poses no problem. Most virtual circuit networks can also handle datagrams, and alternatively, permanent virtual circuits could be established between gateways and sites on a given network to carry the internet datagrams between those gateways and sites. It would also have been possible to develop a virtual circuit internet protocol. This would have involved using error recovery at each gateway, thus making the individual networks appear like virtual circuit networks. We discuss this later.

Two transport layer protocols were developed along with IP. The first, the transmission control protocol, known as TCP, uses error recovery and reordering to present

Samsung Exhibit 1015, Page 136 of 573

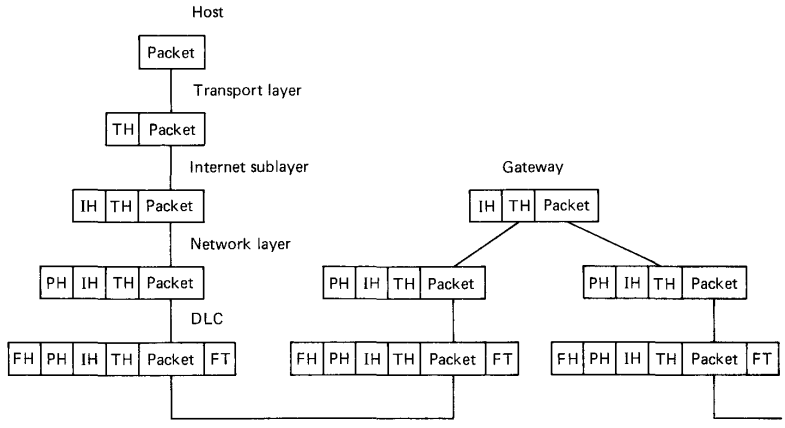


Figure 2.51 Illustration of a packet as it passes through a gateway. Within any given network, only the packet header and frame header are used, whereas at a gateway, the internet header is used to select the next network and to construct a new packet header appropriate to the new network.

the user with reliable virtual circuit service. The second, the user datagram protocol, simply deals with the question of where to send datagrams within a given destination.

The primary functions of IP are, first, to route datagrams through the Internet; second, to provide addressing information to identify the source and destination sites; and third, to fragment datagrams into smaller pieces when necessary because of size restrictions in the various networks. We shall not discuss routing here, since that is a topic for Chapter 5. We first discuss addressing and then fragmentation.

Since datagrams are being used, it is necessary for each datagram to contain identifications of the source and destination sites that are universally recognizable throughout the Internet (*i.e.*, the virtual channel addressing scheme of Section 2.8.1 is not applicable to datagram networks). Assigning and keeping track of these addresses is difficult because of the size and rapidly changing nature of the Internet. The addressing scheme used for IP is hierarchical in the sense that each site is identified first by a network identifier and then by an identifier for the site within the network. One peculiar consequence of this is that a site that directly accesses several networks has several addresses, and the address used determines which of those networks a datagram will arrive on.

The addresses in IP are all 32 bits in length, with a variable number of bits allocated to the network ID and the rest to the site (or host as it is called in IP terminology). In particular, the networks are divided into three classes, A, B, and C, according to size. The largest networks, class A, have a 7 bit network ID and a 24 bit host ID. Class B networks have 14 bits for network ID and 16 for host, and class C have 22 bits for network and 8 for host. The remaining one or two bits out of the 32 bit field are used to encode which class is being used. These addresses indicate only the source and destination sites. The

Samsung Exhibit 1015, Page 137 of 573

intended process or user at the site is contained in additional information in the transport layer header discussed in Section 2.9.

The original length of a datagram is selected primarily for user convenience, but since individual networks have limitations on packet lengths, the datagrams must sometimes be fragmented into smaller pieces for passage through these networks. These fragments are datagrams in their own right (and can be further fragmented if need be) and the fragments are not combined into the original datagram until reaching the final destination. If one or more fragments are lost in the Internet, then, after some time-out, the ones that arrive at the destination are also thrown away, and (if TCP is being used at the transport layer) the entire datagram is eventually retransmitted by the source site. This appears to be quite inefficient, but the alternative would have been to do error recovery on the fragments in the internet layer.

When a datagram is fragmented, most of the header is simply copied to each fragment. The fragment length field (see Fig. 2.52), however, is modified to give the number of bytes in the given fragment. Fragmentation is always done on eight byte boundaries, and the fragment offset field gives the starting point of the given fragment in eight byte increments. One of the three flag bits in the header is used to indicate whether or not a fragment is the last fragment of the datagram. The final piece of information needed at the destination to reconstruct a datagram from its fragments is the datagram ID. The source site numbers successive datagrams, and this sequence number is used as the datagram ID; this is copied to each fragment and allows the destination to know which fragments are part of the same datagram. It is not difficult to see that the combination of fragment length, offset, last fragment bit, and datagram ID provided for each fragment allows the original datagram to be reconstructed.

We now give a quick summary of what the other fields in the IP datagram header do. The field "Vers" tells which version of the protocol is being used. This is useful when a new version of the protocol is issued, since it is virtually impossible to synchronize a changeover from one version to another. The field "HL" gives the length of the header (including options) in 32 bit increments. "Service" allows users to specify what kind of service they would like, but there is no assurance that the gateways pay any attention to this.

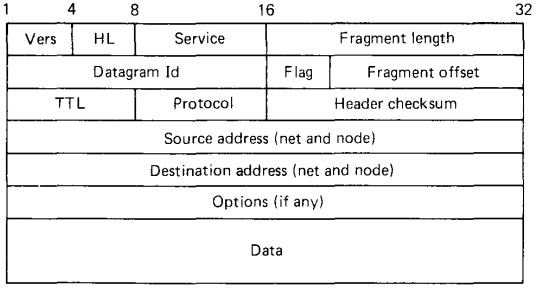


Figure 2.52 IP datagram header format in 32 bit words. Note that the minimum-size header is 20 bytes.

Samsung Exhibit 1015, Page 138 of 573

The time to live field, "TTL," is decremented each time the datagram reaches a new gateway, and the datagram is thrown away when the count reaches zero. This is supposed to prevent old datagrams from existing in the Internet too long, but unfortunately, there is no mechanism to prevent a datagram from remaining in one of the individual networks for an arbitrarily long period. The field "protocol" indicates which higher-layer protocol is to operate on the datagram. This is necessary since the information about where the datagram is to go at the destination site is in the transport header, and the format of this header depends on which transport protocol is being used. Finally, the header check sum checks on the header. This seems a little peculiar for a datagram, but as a practical matter, one does not want datagrams in a large internet going to some random destination in the event of an error.

2.9 THE TRANSPORT LAYER

The transport layer is the fourth layer in the OSI layered architecture and has the functions, (1) of breaking messages into packets, (2) of performing error recovery if the lower layers are not adequately error free, (3) of doing flow control if not done adequately at the network layer, and (4) of multiplexing and demultiplexing sessions together. The first function, breaking messages into packets, is relatively straightforward, aside from the question of selecting the maximum packet length for a network. As discussed in the preceding subsection, even the selection of packet size is less important for TCP combined with IP, since IP fragments packets when necessary; thus, the transport layer packets going from TCP to IP can really be viewed as messages with a transport layer header.

We have already discussed the general problem of end-to-end error recovery and the reasons why it is normally done at the transport layer. Rather than discussing the other functions of the transport layer in general, we shall discuss them in terms of TCP and the ISO standards.

2.9.1 Transport Layer Standards

One has a wide choice of standard transport protocols. The International Standards Organization (ISO), in its OSI layered architecture, has five different classes of transport protocol (denoted TP), numbered 0 to 4. TP class 0 is a relatively simple protocol that splits messages into packets and reassembles them at the destination but provides neither for multiplexing of sessions nor for error recovery. Class 1 allows for some error recovery but relies on the network layer and consistency checks for detection of the need for retransmission. Class 2 provides multiplexing but no error recovery, and class 3 provides the error recovery of class 1 and the multiplexing of class 2. Finally, class 4 provides multiplexing plus a more complete form of error recovery using both detection of errors and retransmission. The class 4 protocol is designed for use with datagram networks and other networks that do not adequately deliver each packet once, only once, correctly, and in order.

Samsung Exhibit 1015, Page 139 of 573

The U.S. National Institute for Standards and Technology has also developed a U.S. Federal Information Processing Standard (FIPS) for the transport layer; this is essentially the same as TP classes 2 and 4 of the OSI standard except for some additional options.

The U.S. Department of Defense has standardized TCP (transmission control protocol). As discussed in the preceding subsection, this protocol was originally developed for the ARPANET and was designed to operate on top of IP, thus allowing internetworking between the ARPANET and other networks. TCP was developed before the OSI TP protocols, and TP class 4 uses many of the features of TCP. In what follows, we discuss TCP and then contrast it briefly with TP class 4.

2.9.2 Addressing and Multiplexing in TCP

In Section 2.8.4 we discussed addressing in the Internet and pointed out that the IP header must contain sufficient information to route the given packet through the Internet, namely a network ID and a host ID. One of the basic tenets of layering is that the header (and perhaps trailer) of a packet at a given layer contains all the information necessary for the module at that layer to do its job; the body of the packet at that layer (which contains the headers for the higher layers) should not be consulted by the module at the given layer. When a packet is passed from TCP to IP at a given node, there are extra parameters passed along with the packet that provide IP with addressing information. Similarly, when the datagram arrives at its destination, the IP header is stripped off, but the addressing information and other header information can be passed up as parameters.

In TCP, the entire address of a source (or destination) is called a *socket*. It is organized hierarchically as, first, network ID, then host ID, and finally, the user or process ID within that node. The user or process ID within a host is called a *port* in TCP (*i.e.*, the transport layer is viewed as having many ports accessing the next higher layer, one for each session). The port ID, for both source and destination, is included in the transport header, whereas the network and node IDs appear in the IP header. This means that all sessions going from a given source host to a given destination host will normally have the same source and destination address in the IP header and will be distinguished only in the transport header. Thus all sessions between the same pair of hosts could be viewed as being multiplexed together at the transport layer into a common lower-layer session. The fact that IP is a datagram protocol is irrelevant from this standpoint; TCP could also operate with an X.25 form of network layer, and then the network layer would have to set up a session and map the host address into an address in the X.25 packet header.

It is possible to have multiple logical addresses for the same host in TCP. Thus one could avoid multiplexing by using a separate address in the IP header for each transport layer session. The general point to be learned from this, however, is that transport headers, internet headers, and network headers all have address fields. Sometimes there is redundancy between the information in these various fields, but the information in the network header must be sufficient for routing and flow control in an individual network. Similarly, the internet header plus the network header must be sufficient for routing between gateways. Finally, the transport header must be sufficient to distinguish all sessions multiplexed together.

Samsung Exhibit 1015, Page 140 of 573

The size of the address fields at different layers in the various standards is somewhat bizarre. TCP uses 16 bit port numbers, and TP class 4 similarly uses 16 bits for a reference number that performs the same role. IP uses 32 bit addresses with 8, 16, or 24 bits representing the node and the rest representing the component network (along with an encoding of whether 8, 16, or 24 bits represent the node). On the other hand, X.25 uses only 12 bits as a virtual circuit number. The use of virtual circuits allows part of this efficiency at the network layer, but one still wonders why the standard network layer is so stingy with address bits and the transport layer and internet sublayer are so profligate. This same concern for efficiency at the network (and DLC) layer and lack of concern at higher layers runs through all the standard protocols. One might have hoped that with the enormous flexibility in these protocols, some flexibility allowing a little efficiency would have been included (such as variable-length address fields), but the designers of transport protocols and internet protocols seem dedicated to the principle that bigger is better. The large size of address fields at the transport and IP levels and the small size at the network layer sometimes inhibit maintaining session identities (and thus providing different types of service, such as delay and error recovery to different sessions) at the network layer.

2.9.3 Error Recovery in TCP

Error recovery in TCP follows the general pattern described in Section 2.8.2; the strategy is selective repeat ARQ. The transport layer packets each contain both a 32 bit sequence number, SN, and a 32 bit request number, RN, in the packet header. The number SN is used in a rather peculiar way in TCP, since it counts successive bytes in the session data rather than successive packets. In particular, SN is the number of the first data byte in the given transport layer packet. Thus if a given transport layer packet has a sequence number $SN = m$ and contains n bytes in the packet body (*i.e.*, the part of the packet labeled "data" in Fig. 2.53), then the next transport layer packet for that session has $SN = m + n$. There appears to be no good reason for this, but it does not do any harm, other than adding complexity and overhead. One might think that this type of numbering would make it easier to fragment packets into smaller pieces for internetworking, but as we have seen, IP has its own mechanisms for handling fragmentation. The request number RN, as one might expect, denotes the next desired

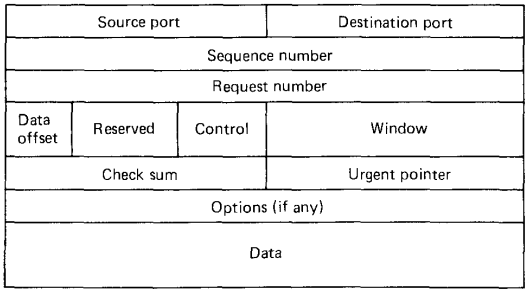


Figure 2.53 TCP header format in 32 bit words; note that the minimum-sized header is 20 bytes.

packet in the opposite direction by giving the number of the first data byte in that packet. As with most ARQ schemes, the RN field allows acknowledgments to be piggybacked on data for the given session in the reverse direction, and in the absence of such data, a packet with just the header can carry the RN information.

As discussed earlier, errors are detected in TCP by a 16 bit check sum calculated from both the transport layer packet and some parameters (such as source and destination address and packet length) that are passed to the next lower layer. These addresses are also checked in IP with a header check sum, but TCP is supposed to work with other lower layers, so this provides worthwhile extra protection.

From the description above, it should be clear that error recovery for packets within an established connection at the transport layer is not a logically complicated matter . The same principles apply as at the DLC layer, and the only new feature is that of enlarging the modulus to allow for out-of-order packet arrivals. There are several complications, however, that remain to be discussed.

The first such complication is that of retransmission time-outs. At the DLC layer, retransmission time-outs provide an interesting issue to think through in understanding how distributed algorithms work, but the issue is not very critical. At the transport layer, the issue is quite serious, since high congestion in the network causes large delays, which trigger retransmissions from the transport layer, which further increase delays. When the congestion becomes bad enough, packets will also be discarded in the network. This triggers more retranmissions, and eventually throughput is reduced to almost zero. One advantage of doing error recovery at the network layer is that it is possible to provide better estimates on delay, and thus avoid retransmission under high-delay conditions.

The final complication is the problem of error recovery when setting up and tearing down connections. What makes this problem tricky is that one does not want to save information about connections after they have been torn down. Thus there is the possibility of a packet from an old connection entering a node after a new connection has been set up between the same processes. There is a similar possibility of confusion between the packets used to initiate or tear down a connection, and finally there is a possibility of confusion when one node crashes and loses track of connections that still exist at other nodes.

In TCP, the major mechanism used to deal with the initiation and disconnection of sessions is the synchronization of the sequence number and request number at the initiation of a session. Each node has a clock counter (not synchronized between nodes) that increments once each 4 μsec. When a new connection is attempted, this clock counter is used as the initial sequence number for the first packet in an initiation handshake. The opposite node responds by acknowledging this sequence number in its request number (as usual) and using its own clock counter for its own initial sequence number. Since the clocks are incrementing faster than the sequence numbers of any established connection can increment (at least for data nets at today's speeds) and since the 32 bit clock counter wraps around less than once in 4 hours, this mechanism solves most of the problems noted above. With this type of protocol, the responding node cannot start to send data (beyond this initial packet and its potential repetitions) until it receives a second packet from the initiating node acking the responding node's first packet. Because of this third

Samsung Exhibit 1015, Page 142 of 573

required packet, this type of initiation is called a three-way handshake. We discussed this type of handshake in Section 2.7. What is new here is that avoiding confusion between old and new connections is achieved by a pseudorandom choice of sequence numbers rather than by using an alternation of connection and disconnection. The strategy here has the disadvantage of potential malfunction under arbitrary delays, but the use of a 32 bit number makes this possibility remote.

The major type of problem for this kind of handshake is that it is possible for one node to believe that a given connection is open while the other node believes that it is closed. This can happen due to one node crashing, the network disconnecting temporarily, or incomplete handshakes on opening or closing. If the node that believes the connection is open sends a packet, and then the other node starts to initiate a connection, it is possible for the new connection to get set up and then for the old packet to arrive. If the old connection had been in existence for a very long time, it is possible for both the sequence and request numbers to be in the appropriate windows, causing an error. This type of situation appears very unlikely, especially considering the large amount of redundancy in the rest of the header.

For completeness, the format of the header in TCP is shown below as a sequence of 32 bit words. We have discussed the source port, destination port, sequence number, request number, and checksum. The data offset gives the number of 32 bit words in the header, the urgent pointer has to do with high-priority data and need not concern us, and the options also need not concern us. Each of the six control bits has a particular function. For example, the fifth control bit is the SYN bit, used to indicate an initiation packet that is setting the sequence number field. The second control bit is the ACK bit, used to distinguish the first initiation packet (which does not carry an ACK) from the second and subsequent packets, which do carry an ACK.

2.9.4 Flow Control in TCP/IP

TCP uses a permit scheme to allow the destination to control the flow of data from the source host. This is implemented by a 16 bit field called a *window* stating how many bytes beyond the request number can be accepted for the given connection by the given node. This provides some rationale for TCP to count bytes (or some fixed-length unit of data), since there is such high potential variability in datagram lengths. For a given size of window, the level of congestion in the Internet (or network in the case of no internetworking) controls how fast data flow for the given connection. Thus this scheme both protects the destination from being saturated and slows down the sources in the event of congestion.

It is also possible for a source to monitor the round-trip delay from datagram transmission to acknowledgment receipt. This can provide an explicit measure of congestion and there are many strategies that use such measurements to constrain data flow into the Internet or subnet. There are some inherent difficulties with strategies that use such measurements to decrease source flow. The first is that such strategies are subject to instability, since there is a delay between the onset of congestion and its measurement. This is made worse by throwing away packets under congestion. The next problem is

Samsung Exhibit 1015, Page 143 of 573

that "good citizen" sessions that cut back under congestion are treated unfairly relative to sessions that do not cut back. Since the Internet uses datagrams, the Internet cannot distinguish "good citizen" sessions from bad, and thus datagrams are thrown away without regard to session.

It is important to recognize that these are inherent problems associated with any datagram network. The network resources are shared among many sessions, and if the resources are to be shared fairly, the network has to have some individual control over the sessions. The session that a datagram belongs to, in principle, is not known at the network layer (note from Fig. 2.52 that the IP protocol has no header field to indicate session number), and thus the datagram network or internet cannot monitor individual sessions. A corollary of this is that datagram networks cannot provide guaranteed types of service to given sessions and cannot negotiate data rate. A datagram network might give different classes of service to different kinds of datagrams, but the choice of service class comes from the transport layer and is not controlled by the network. If some sessions place very heavy demands on the network, all sessions suffer.

TCP/IP has been enormously successful in bringing wide area networking to a broad set of users. The goal of bringing in a wide variety of networks precluded the possibility of any effective network or internet-oriented flow control, and thus precluded the possibility of guaranteed classes of service. How to provide such guarantees and still maintain internetworking among a wide variety of networks is a challenge for the future.

2.9.5 TP Class 4

The ISO transport protocol, TP class 4, is very similar to TCP, with the following major differences. First, sequence numbers and request numbers count packets rather than bytes (in the jargon of the standards, packets at the transport level are called transport protocol data units or TPDUs). Next, acknowledgments are not piggybacked on packets going the other way, but rather, separate acknowledgment packets are sent (using the same selective repeat strategy as before). These ack packets are sent periodically after some time-out even with no data coming the other way, and this serves as a way to terminate connections at one node that are already terminated at the opposite node.

Another difference is that sequence numbers are not initiated from a clock counter, but rather start from 0. The mechanism used in TP class 4 to avoid confusion between old and new connections is a destination reference number. This is similar to the destination port number in TCP, but here the reference number maps into the port number and can essentially be chosen randomly or sequentially to avoid confusion between old and new connections.

2.10 BROADBAND ISDN AND THE ASYNCHRONOUS TRANSFER MODE

In Sections 2.8 and 2.9 we have described some of the point-to-point aspects of network layer and transport layer protocols. In this section we describe a new physical layer protocol called asynchronous transfer mode (ATM). From a conceptual viewpoint, ATM

Samsung Exhibit 1015, Page 144 of 573

performs the functions of a network layer protocol, and thus we must discuss why network layer functions might be useful at the physical layer. Since ATM was developed for use in broadband ISDN systems, however, we start with some background on broadband ISDN.

A broadband integrated services digital network (broadband ISDN) is a network designed to carry data, voice, images, and video. The applications for such networks are expected to expand rapidly after such networks are available. Recall that in our discussion of ISDN in Section 2.2, we pointed out that a high-resolution image represented by 10^9 bits would require over 4 hours for transmission on a 64 kbit/sec circuit and would require 11 minutes on a 1.5 Mbit/sec circuit. Broadband ISDN is an effort to provide, among other things, data rates that are high enough to comfortably handle image data in the future. The planned access rate for broadband ISDN is 150 Mbit/sec, which is adequate for image traffic and also allows for the interconnection of high-speed local area networks (see Section 4.5.5). This access rate also allows video broadcast traffic, video conferencing, and many potential new applications.

The technology is currently available for building such networks, and the potential for new applications is clearly very exciting. However, the evolution from the networks of today to a full-fledged broadband ISDN will be very difficult, and it is debatable whether this is the right time to standardize the architectural details of such a network. Some of the arguments for rapid standardization are as follows. There are current needs for these high data rates, including image transmission, accessing high-speed local area networks, and accessing supercomputers; the rapid development of such a network would meet these needs and also encourage new applications. Many people also argue that the cost of networks is dominated by the "local loop" (*i.e.*, the access link to the user); optical fiber will increasingly be installed here, and economy dictates combining all the telecommunication services to any given user on a single fiber rather than having multiple networks with separate fibers. The difficulty with this argument is that it is not difficult to multiplex the access to several networks on a single fiber. Another popular argument for the rapid deployment of broadband ISDN is that there are economies of scale in having a single network provide all telecommunication services.

One of the arguments against rapid standardization is that there are also diseconomies of scale involved with a single all-purpose network. These diseconomies come from trying to force very different types of communication requirements, such as video and conventional low-speed data, into a common mold. Another argument is that there is no large-scale engineering experience with such networks and little knowledge of how the potential applications will evolve. This is exacerbated by the rapid development of optical fiber technology, which is rapidly changing the range of choices available in the backbone of a broadband network. Finally, the evolution of broadband ISDN is central to the future of the telecommunication industry, and thus political, legal, and regulatory issues, as well as corporate competitive positioning, will undoubtedly play a large and uncertain role in the evolution and structure of future networks. In what follows we look only at technological issues, but the reader should be aware that this is only part of the story.

Broadband ISDN can be contrasted both with data networks and with the voice network. In contrast with data networks, there is an increase of over three orders of magnitude in the link speeds envisioned for broadband ISDN. Because of this, broadband

Samsung Exhibit 1015, Page 145 of 573

ISDN designers are more concerned with computational complexity and less concerned with efficient link utilization than are designers of conventional data networks. Because of the speed constraints, the protocols in broadband ISDN should be simple and amenable to both parallel and VLSI implementation.

Another important contrast with data networks is that individual user requirements are critically important in broadband ISDN. Part of the reason for this is that voice and video normally require both a fixed bit rate and fixed delay, whereas packet-switched data networks generally have variable delays and often use flow control to reduce data rates. In the past, data applications that required fixed rate and/or fixed delay usually avoided data networks and instead used dedicated leased lines from the voice network. In other words, traditional data networks provide a limited range of service, which users can take or leave. Broadband ISDN, on the other hand, is intended to supply all necessary telecommunication services, and thus must provide those services, since there will be no additional voice network on which to lease lines.

Next consider the contrast of broadband ISDN with the voice network. All voice sessions on a voice network are fundamentally the same. Except on the local loops, voice is usually sent digitally at 64 kbits/sec, with one eight bit sample each 125 μsec. In Section 2.2 we described briefly how these sessions are time-division multiplexed together. The important point here is that all of these multiplexing systems, including the SONET standard, use a 125 μsec frame broken up into a large number of one byte slots. Switching is accomplished at a node by mapping the slots in the incoming frames into the appropriate slots of the outgoing frames. In contrast, broadband ISDN must cope with potential user rates from less than 1 bit/sec to hundreds of megabits per second. It must also cope with both bursty traffic and constant-rate traffic.

Despite the enormous difference between broadband ISDN and the voice network, broadband ISDN will probably evolve out of the present-day voice network if and when it is built. The reason for this is that the bulk of transmission and switching facilities currently exist within the current voice network, and current wide area data networks typically lease their transmission facilities from the voice network. Because of this, the early thinking about broadband ISDN focused on using the underlying synchronous 125-μsec frame structure of the voice network to provide the physical layer transmission.

Within the frame structure of the voice network, it is possible to group multiple slots per frame into a single circuit whose rate is a multiple of 64 kbits/sec. These higher-rate circuits can be leased either for dedicated data channels or for use as links in data networks. They come in standard multiples of 64 kbits/sec (*e.g.*, DS1 service is 24 times 64 kbits/sec and DS3 service is 28 times the DS1 speed). These higher-rate circuits could also be used in broadband ISDN for high-rate services such as video. In this sense, broadband ISDN was initially visualized as an evolution from (narrowband) ISDN, maintaining the emphasis both on 64 kbit/sec circuit switched sessions and on various standard multiples of the 64 kbit/sec speed. The early objective for broadband ISDN was simply to increase data rates to allow for video, high-resolution images, and interconnection of high-speed local area networks.

As interest in broadband ISDN increased and as standardization efforts started, an increasing number of people questioned the underlying reliance on the synchronous frame

Samsung Exhibit 1015, Page 146 of 573

structure of the voice network and proposed using an asynchronous packet structure for multiplexing at the physical layer. As this debate quickened, use of the conventional synchronous frame structure became known as STM (synchronous transfer mode), and the use of a packet structure became known as ATM (asynchronous transfer mode). Fundamentally, therefore, the choice between STM and ATM is the choice between circuit switching and packet switching.

As discussed in Section 1.2.3, conventional data networks use packet switching principally because of the burstiness of the data, but burstiness is far less important for broadband ISDN. In the first place, voice traffic totally outweighs conventional data traffic and will probably continue to do so (see Problem 1.2). Next, circuit switching is already used in the voice network, is ideal for voice traffic, and is well suited for video traffic. High-resolution images would not fit well into circuit switching, but could be handled (along with conventional data) on a packet network using the transmission resources of a circuit-switched net as a physical layer.

A more important aspect of the choice between STM and ATM is the question of the flexibility of STM for handling a wide variety of different data rates. STM would require each circuit-switched application to use some standard multiple of 64 kbits/sec. This would lead to great switching complexity if the number of standard rates were large, and to great inefficiency otherwise. In addition, the proponents of ATM argue that the use of a small set of standard rates would inhibit the development of new applications and the development of improved data compression techniques.

This lack of flexibility in STM is particularly troublesome at the local loop. Here, given a small set of sessions, each would have to be assigned one of the standard rates. Any attempt to multiplex several sessions (particularly bursty data sessions) on one standard rate, or to use several standard rates for one session, would lead to considerable complexity. In comparison with these problems of matching sessions to standard rates, the ATM solution of simply packetizing all the data looks very simple.

While the relative merits of STM and ATM were being debated, there was great activity in developing high-speed packet-switching techniques. These techniques were highly parallelizable and amenable to VLSI implementation. As a result of both the flexibility of ATM for local access and the promise of simple implementation, the CCITT study group on broadband ISDN selected ATM for standardization.

Before describing ATM in more detail, we should understand how it fits into the OSI seven-layer architecture. In the original view of ISDN, the synchronous frame structure of STM was quite properly viewed as the physical layer. Thus, when ATM replaced STM, it was reasonable to consider ATM as the physical layer for data transmission. This had the added advantage of relieving ATM from any responsibility to conform to the OSI standard network layer. Another consequence of using ATM in place of STM, however, is that there is no longer a need for a packet-switched network on top of ATM; ATM deals with bursty data directly as a packet-switching technique in its own right. We will see later that ATM, as a packet-switching layer, has an *adaptation layer* on top of it. This adaptation layer is much like the transport layer of OSI; it breaks up the incoming messages or bit streams into fixed-length packets (called cells in ATMese), and it is the responsibility of the adaptation layer, using the ATM layer, to reconstitute the messages or

Samsung Exhibit 1015, Page 147 of 573

data stream at the other end. What comes in could be OSI layer 2 frames; the adaptation layer will view this as a user message, break it into "cells," and reconstitute the user message (*i.e.*, the frame) at the destination. Thus, as far as the OSI layer 2 is concerned, the broadband ISDN looks like a bit pipe. Naturally, there is no need for a user to go through all the OSI layers, since data can be presented directly to the broadband ISDN.

2.10.1 Asynchronous Transfer Mode (ATM)

ATM packets (or cells) all have the same length, 53 bytes, consisting of 5 bytes of header information followed by 48 bytes of data (see Fig. 2.54). The reason for choosing a fixed length was that implementation of fast packet switches seems to be somewhat simpler if packets arrive synchronously at the switch. One might question being so restrictive in a long-term standard because of a short-term implementation issue, but the standards committee clearly felt some pressure to develop a standard that would allow early implementation.

The major reason for choosing 48 bytes of data for this standard length was because of the packetization delay of voice. Standard 64 kbit/sec voice is generated by an eight bit sample of the voice waveform each 125 μsec. Thus, the time required to collect 48 bytes of voice data in a cell is 6 msec. Even if a cell is transmitted with no delay, the reconstruction of the analog voice signal at the destination must include this 6 msec delay. The problem now is that this reconstructed voice can be partially reflected at the destination and be added to the voice in the return direction. There is another 6 msec packetization delay for this echo on the return path, and thus an overall delay of 12 msec in the returning echo. A delayed echo is quite objectionable in speech, so it should either be kept small or be avoided by echo cancellation. In the absence of echo cancellation everywhere, it is desirable to keep the delay small.

There are several reasons why a larger cell length would have been desirable. One is that there is a fixed amount of computation that must be done on each cell in an ATM switch, and thus, the computational load on an ATM switch is inversely proportional to the amount of data in a cell. The other relates to the communication inefficiency due to the cell header (and as we shall see later, the header in the adaptation layer). The length of 48 bytes was a compromise between voice packetization delay and these reasons for wanting larger cells.

The format of the header for ATM cells has two slightly different forms, one for the access from user to subnet and the other within the subnet (see Fig. 2.55). The only difference between the header at the user–subnet interface and that inside the network

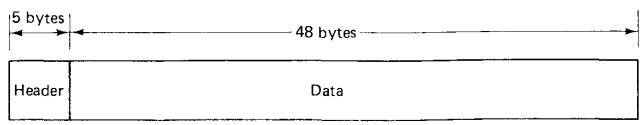


Figure 2.54 Packet (cell) format in ATM. Note that the cell size is fixed.

Samsung Exhibit 1015, Page 148 of 573

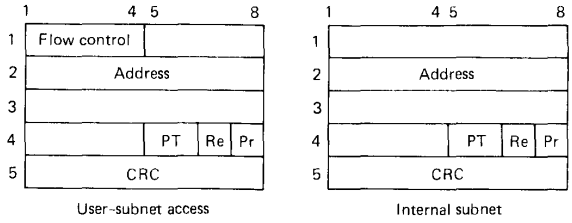


Figure 2.55 ATM header format for the user–subnet interface and for the internal subnet. PT stands for payload type, Re for reserved, and Pr for priority bit.

is a four bit "generic flow control" field provided at the user–subnet interface. This is intended to allow the user to mediate between the demands of different applications at the user site. The use of this field is not standardized and is used only to help the user to statistically multiplex the cells from different applications onto the access link to the network. The question of how to accomplish flow control within the broadband ISDN network is discussed in Section 2.10.3, but there is no field in the ATM header (other than the priority bit) to help accomplish this function.

The contents of the address field in the header can change as the cell travels from link to link, and aside from a detail to be discussed later, this address is used in the same way as the virtual channel numbers described for addressing in Section 2.8. That is, ATM uses virtual circuits, and when a virtual circuit is set up for a session, a virtual channel number is assigned to that session on each link along the session's path. Since the address field contains 24 bits for the user–network interface, over 16 million sessions can access the network from one user site. Since the "user" might be a corporation or a local area network, for example, this is not quite as extravagant as it sounds. Within the network, the address field contains 28 bits, allowing for over 268 million sessions to share a link.

There is an additional quirk about addressing in ATM. The address field is divided into two subfields, called the virtual channel identifier (VCI) and the virtual path identifier (VPI) in ATMese. The reason for this is to allow sessions sharing the same path (or at least sharing the same path over some portion of their entire paths) to be assigned the same virtual path identifier and to be switched together. For example, two office buildings of a given corporation might require a large number of virtual circuits between the two offices, but they could all use the same virtual path identifiers and be switched together within the network. The VCI subfield contains 16 bits, and the VPI subfield contains 8 bits at the user–subnet interface and 12 bits within the subnet.

After the address field comes the "payload type" field, the "reserved field," which is not used, and the "priority bit" field, which is discussed in Section 2.10.3. The payload type field is used to distinguish between cells carrying user data and cells containing network control information.

Finally, at the end of the header is an eight bit CRC checking on the ATM header. The generator polynomial of this CRC is $g(D) = D^8 + D^2 + D + 1$. This is the product of a primitive polynomial of degree 7 times $D + 1$, and, as explained in Section 2.3.4,

Samsung Exhibit 1015, Page 149 of 573

the header thus has a minimum distance of 4. Note that this CRC checks only on the cell header, not on the data. Thus errors in the data are delivered to the destination and some other means is required to achieve error-free transmission of user information. The reason for checking the header is to reduce the probability of having data for one user get delivered to another user.

One very unusual feature of this CRC is that it is sometimes used to *correct* errors rather than just to detect them. The reason for this is that broadband ISDN is intended for use on optical fiber channels, and at least with current modulation techniques, errors are very improbable (the error rate is less than 10^{-9}) and more or less independent. Thus single error correction might actually remove most header errors. In an attempt to guard against bursts of errors due to equipment problems, the receiving node switches between single error correction and pure error detection as illustrated in Fig. 2.56. After either correcting an error or detecting more than a single error, the decoder switches to an error-detecting mode and stays there until the next header is received without detected errors. This is quite a novel approach to improving the reliability of optical channels in a simple way. The only problem is that optical technology is still rather fluid, and this approach is very dependent both on a very low channel error probability and on an extremely low probability of multiple errors.

We next investigate how ATM interfaces to the physical channel underneath. Since ATM performs the functions of a network layer, we would expect to find some sort of data link control layer beneath it. However, in this case neither ARQ nor framing is required, so there is no need for DLC. There is no need for ARQ for two reasons: first, the error probability is very low on the links, and second, a large part of the traffic is expected to be voice or video, where retransmission is inappropriate. Thus, error recovery is more appropriate on an end-to-end basis as required. In North America and Japan, ATM will use SONET as a lower layer and there are control bytes within the SONET frame structure that can specify the beginning of an ATM cell within the SONET frame. This of course explains why the user access rate for ATM was standardized at the SONET STS-3 rate. Since all cells are of the same length, framing information is not needed on each cell. Some European countries are not planning to use SONET as a lower layer to ATM, so they will require some other means of framing.

Since framing information is not provided on each cell, it is necessary to be careful about what to send when no data are available. One solution is to send "unassigned cells" which carry no data but fill up a cell slot. A special reserved virtual circuit number is used to identify this type of cell so that it can be discarded at each ATM switching point.

The use of SONET underneath ATM has a number of other advantages. One advantage is that very-high-data-rate applications could have the possibility of using SONET directly rather than being broken down into tiny ATM cells that must be individually

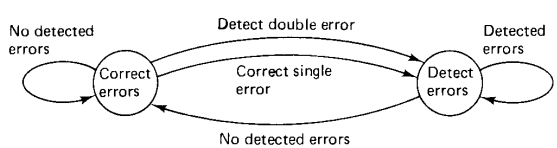


Figure 2.56 Error correction and detection on the headers of ATM cells. A single error is corrected only if the preceding cell header contained no detected errors.

switched at each node. Another is that SONET circuits carrying ATM traffic could be switched directly through intermediate nodes rather than being broken down at each switch into ATM cells. If one takes the ATM user access rate of 150 mbits/sec seriously, one can see that backbone links might be required to carry enormous data rates, and switches that operate on 53 bytes at a time are not attractive. If one looks further into the future and visualizes optical fibers carrying 10^{12} or 10^{13} bits/sec, the thought of ATM switches at every node is somewhat bizarre.

2.10.2 The Adaptation Layer

The adaptation layer of broadband ISDN sits on top of the ATM layer and has the function of breaking incoming source data into small enough segments to fit into the data portions of ATM cells. In the sense that ATM is performing the functions of a network layer, the adaptation layer performs the function of a transport layer. Thus the adaptation layer is normally used only on entry and exit from the network. Since the adaptation layer interfaces with incoming voice, video, and message data, its functions must depend on the type of input. The adaptation layer is not yet well defined within the standards bodies and is in a state of flux. Thus we give only a brief introduction to the way in which it accomplishes its functions.

Different types of input are separated into the following four major classes within the adaptation layer:

Class 1: Constant-bit-rate traffic. Examples of this are 64 kbit/sec voice, fixed-rate video, and leased lines for private data networks.

Class 2: Variable-bit-rate packetized data that must be delivered with fixed delay. Examples of this are packetized voice or video; the packetization allows for data compression, but the fixed delay is necessary to reconstruct the actual voice or video signal at the receiver.

Class 3: Connection-oriented data. Examples of this include all the conventional applications of data networks described in Section 1.1.3 (assuming that a connection is set up before transferring data). This class includes the signaling used within the subnet for network control and establishment of connections. Finally, it includes very-low-rate stream-type data.

Class 4: Connectionless data. Examples include the conventional applications of data networks in cases where no connection is set up (*i.e.*, datagram traffic).

For classes 2, 3, and 4, the source data coming into the adaptation layer are in the form of frames or messages. The adaptation layer must break these frames into segments that fit into the ATM cells and provide enough extra information for the destination to be able to reconstruct the original frames. Along with this, the different classes (and different applications within the classes) have differing requirements for treating the problems of dropped or misdirected ATM cells. We describe the approach taken for class 3 traffic and then briefly describe the differences for the other classes.

Samsung Exhibit 1015, Page 151 of 573

Class 3 (connection-oriented) traffic The adaptation layer is split into two sublayers, called the convergence sublayer and the segmentation and reassembly sublayer. The first treats the user frames as units and is concerned with flow control and error recovery. The second treats the segments of a frame as units and is concerned with reassemblying the segments into frames. As usual, peer communication between nodes at each sublayer takes place via headers and trailers as illustrated in Fig. 2.57.

Current plans call for a two byte header and two byte trailer in the segmentation and reassembly sublayer. The header consists of the following fields:

Segment type (2 bits). This distinguishes whether a segment is the beginning segment of a convergence sublayer frame, an intermediate segment, a final segment, or a segment that contains an entire frame.

Sequence number (4 bits). This numbers segments within a convergence sublayer frame; this helps check on dropped or misdirected cells.

Reserved (10 bits). This might be used to multiplex multiple-user sessions on a single virtual circuit.

The trailer consists of the following fields:

Length indicator (6 bits). This tells how many bytes from the convergence sublayer frame are contained in the given segment. This is useful primarily for the last segment of a frame.

CRC (10 bits). The CRC polynomial is $D^{10} + D^9 + D^5 + D^4 + D + 1$. This is a primitive polynomial times $D+1$ and has the properties given in Section 2.3.4. It is used to check on the entire segment, including the header and the length indicator of the trailer.

It should be clear that the header and trailer above provide enough data to reconstruct the convergence sublayer frames from the received segments. The sequence

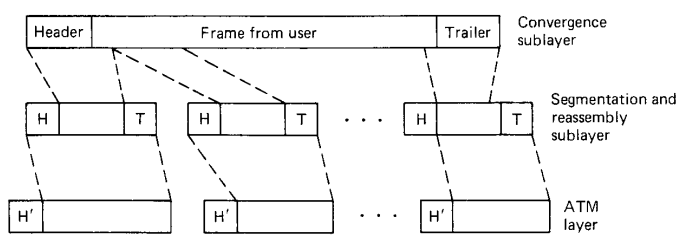


Figure 2.57 Headers and trailers at the convergence sublayer and segmentation and reassembly sublayer of the adaptation layer. Note that the user frame and its convergence sublayer header and trailer need not completely fill the data area at the segmentation and reassembly sublayer. The data plus header and trailer at the segmentation and reassembly sublayer must be exactly 48 bytes in length, however, to fit into the ATM data area.

Samsung Exhibit 1015, Page 152 of 573

number and CRC also provide some protection against bit errors and dropped or misdirected cells, but the approach is ad hoc.

The header and trailer for the convergence sublayer have not yet been defined. It is surprising that a CRC is used at the segmentation and reassembly sublayer rather than at the convergence sublayer. The longer block length at the convergence sublayer would have allowed greater error detection (or correction) power and higher efficiency. It would also have made it more convenient to use a CRC only when required by the user.

Class 4 (connectionless) traffic The headers and trailers at the segmentation and reassembly sublayer are expected to be the same for class 4 and class 3 traffic. The major new problem is that of connectionless traffic in a network where the cells are routed by virtual circuits. As discussed in Section 2.10.3, the most promising approach to this traffic is to view the routing as taking place at a higher layer than the adaptation layer; essentially this means viewing the datagram traffic as a higher-layer network using ATM virtual circuits as links (see Fig. 2.58). Thus, whereas connection-oriented traffic enters the adaptation layer only at origin and destination, connectionless traffic passes through the adaptation layer at each node used as a datagram switch. Between any two datagram switches, the traffic would use permanent virtual circuits assigned to carry all the connectionless traffic between the two switches. An interesting question is whether to view the switching of connectionless traffic as taking place in the convergence sublayer or strictly at a higher layer.

Class 1 and 2 traffic Class 1 traffic is constant-bit-rate traffic without any framing structure on the input to the adaptation layer. Thus a user data stream simply has to be broken into segments. The current plan is to use a one byte header and no trailer at the segmentation and reassembly sublayer (thus providing 47 bytes of user data per ATM cell). The header would consist of a four bit segment number plus a four bit CRC on the segment number. The reason for this is that occasional errors in this type

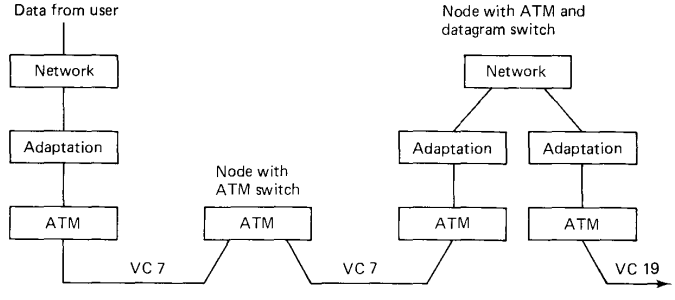


Figure 2.58 Datagram traffic can use an ATM network by being routed from one datagram switch to another. The virtual link between neighboring datagram switches is a virtual circuit in the ATM network dedicated to the aggregate of all the datagram traffic traveling between the two datagram switches.

Samsung Exhibit 1015, Page 153 of 573

of data are permissible, but to maintain framing, it is important to know when data have been dropped or misinserted, and the segment numbers provide this protection. Class 2 traffic has the same resilience to occasional errors as class 1 and the same need to know about dropped cells. At the same time, it has an incoming frame structure, requiring some of the features of the class 3 protocol for reassembling the frames.

2.10.3 Congestion

One would think that congestion could not be a problem in a network with the enormous link capacities of a broadband ISDN network. Unfortunately, this first impression is naive. Video and image applications can use large data rates, and marginal increases in quality will always appear to be possible by further increases in rate. Also, given a broadband network, there will be vested interests in ensuring that it is reasonably loaded, and reasonable loading will typically lead in time to heavy loading.

There are three mechanisms in broadband ISDN for limiting congestion. The first is for the network and user to agree to the required rate of the user at connection setup time. This agreement is expected to involve other quantities, such as the burstiness of the source and the quality of service required. The second mechanism is for the network to monitor each connection (within the convergence sublayer of the adaptation layer) to ensure that the user is complying with the agreed-upon rate and burstiness. Finally, the third mechanism is the priority bit in the ATM header (see Fig. 2.55). We explain briefly how these mechanisms work and interact. Chapter 6 develops the background required for a more thorough understanding of these mechanisms.

The ability to negotiate an agreed-upon rate between user and network is one of the primary reasons for a connection-oriented network. Without such negotiation, it would be impossible for the network to guarantee data rates and limited delay to users. It is easy to determine the effect of a new constant-bit-rate user, since such a session simply uses part of the available capacity on each link of the path without otherwise contributing to queueing delays. For class 2 and class 3 traffic, it is more difficult to determine the effect of a new session, since the burstiness of the traffic can cause queueing delays which can delay existing users as well as the user attempting to set up a connection. Relatively little is known about the interactions of delay between users of widely differing characteristics, and even less is known about the impact of different service disciplines at the switching nodes on these delays.

Since there is no negotiation of rates between the network and connectionless users, these users must be carefully controlled to avoid generating excessive delays for the connection-oriented sessions for which guaranteed rates and delays have been negotiated. This is why it is desirable to have a higher-layer network handling the connectionless traffic. This allows rates to be allocated for the aggregate of all connectionless traffic on given links, thus allowing statistical multiplexing between the connectionless users without having those users interfere with the connection-oriented traffic.

Along with negotiating rates when virtual circuits are established, it is necessary for the network to monitor and regulate the incoming traffic to ensure that it satisfies the agreed-upon limits. There are a number of mechanisms, such as the leaky bucket mech-

Samsung Exhibit 1015, Page 154 of 573

anism, for monitoring and regulating incoming traffic, and these are discussed in Chapter 6. The problem of interest here is what to do with traffic that exceeds the agreed-upon limits. The simplest approach is simply to discard such traffic, but this appears to be foolish when the network is lightly loaded. Another approach is to allow the excessive traffic into the network but to mark it so that it can be discarded later at any congested node.

The priority bit in the ATM header could be used to mark cells that exceed the negotiated rate limits of a user. This priority bit could also be set by a user to indicate lower-priority traffic that could be dropped by the network. Such a capability is useful in some video compression and voice compression schemes. It appears at first that using a single bit to carry all priority information in such a large and diverse network severely limits flexibility in applying flow control to the network. For example, suppose that the priority bit were used both by users to send relatively unimportant data and by the network to mark excessive data from a user. Then it would be possible for the network to be flooded by marked but unimportant video traffic; this could cause the network to discard important bursty traffic that was marked by the network because of exceeding the negotiated burstiness.

One possibility for an increased differentiation of priorities comes from the fact that individual users could negotiate priority while setting up a connection, and the subnet nodes could use these user priorities along with the priority bit in making discard decisions. It is also possible in principle to establish priorities concerning delay while setting up a connection, thus enabling the subnet switches to serve those virtual circuits requiring small delay before other circuits. Designing switches that are simple and yet account flexibly for both delay and discarding requirements is clearly a challenge, but the lack of information in the cell header does not seem to be a critical drawback.

Connectionless traffic poses a particular challenge to ATM networks. In the preceding section it was suggested that this traffic should be routed through a packet network sitting on top of the adaptation layer (or perhaps within the convergence sublayer). One benefit of this is that rate could be allocated to virtual circuits used as links in that higher-layer net. Assuming buffering in these higher-layer packet switches, the aggregate connectionless traffic could stay within its allocated limit and thus not interfere with any of the other classes of users. On the other hand, since the connectionless traffic is aggregated on these virtual circuits, individual connectionless users would obtain the benefits of statistical multiplexing. One could also adjust the rates of the virtual circuits over time to provide yet better service for connectionless traffic over periods of light loading from other classes.

SUMMARY

In this chapter, an introductory survey of the physical layer, a fairly complete treatment of the data link control layer, and a few closely related topics at other layers were presented. From a conceptual standpoint, one important issue has been distributed algorithms between two stations in the presence of errors and arbitrary delays. This has

been developed in the context of ARQ, framing, and error recovery at higher layers. The correctness proof of the go back n class of protocols is important here since it shows how to reason carefully about such algorithms. There are many ad hoc algorithms used in data networks that contain subtle bugs, and it is important for designers to learn how to think about such algorithms in a more fundamental and structured way.

Another conceptual issue has been the encoding of control information. This was developed in the context of framing and session identification. Although the emphasis here was on efficiency of representation, the most important issue is to focus on the information actually conveyed; for example, it is important to recognize that flags with bit stuffing, length count fields, and special communication control characters all convey the same information but in quite different ways.

From a more practical standpoint, discussions were given on how modems work, how CRCs work (and fail), how ARQ works and how it can be made to work more efficiently with carefully chosen time-outs and control feedback, how various kinds of framing techniques work (and fail), how session identification can be accomplished, and how error recovery at different layers can be effected. Many existing networks have been used to illustrate these ideas.

Various standards have been discussed, but not in enough detail for one actually to implement them. Unfortunately, if one wishes to implement a protocol in conformance with a standard, one must read the documentation of the standard, which is long and tedious. The purpose here was to give some conceptual understanding of the standard so as to see what the standard accomplishes and how. There are many special-purpose networks in which standard protocols are inappropriate, and it is helpful to know why.

NOTES, SOURCES, AND SUGGESTED READING

Section 2.2. For more information on linear filters and their representation in the time and frequency domain, see any standard undergraduate text (*e.g.*, [Sie86]) on linear systems. For an extensive and up-to-date treatment of modulation and demodulation of digital signals, see [Pro83]. Also, [Qur85] provides a complete and beautifully organized treatment of adaptive equalization. For the information-theoretic aspects of this chapter (*i.e.*, Shannon's theorem, entropy, error detection and correction coding, and source coding), see [Gal68]. Finally, for an elementary treatment of the topics in this section oriented toward current practice, see [Sta85].

Section 2.3. Treatments of parity check codes and cyclic codes (CRC) can be found in [Bla83], [ClC81], and [Gal68].

Sections 2.4 and 2.5. A classic early paper on ARQ and framing is [Gra72]. An entire subdiscipline has developed to prove the correctness of point-to-point protocols. Typically (see, *e.g.*, [BoS82]) those approaches use a combination of exhaustive computer search plus analytic tools to reduce the search space. More information on highly efficient ARQ and framing is given in [Gal81]. Selective repeat and the advantages of sending

Samsung Exhibit 1015, Page 156 of 573

packets multiple times after errors is discussed in [Wel82]. A more complete analysis of choice of frame lengths is given in [Alt86].

Sections 2.6 and 2.7. An excellent description of the standard DLCs is given in [Car80]. Initialization and disconnection are treated in [BaS83] and [Spi89].

Sections 2.8 and 2.9. TYMNET is discussed in [Rin76] and [Tym81]. The Codex networks are discussed in [HuS86]. See [Ryb80] for more discussion of the X.25 standard and [Com88] for discussion of TCP/IP.

Section 2.10. Basic tutorials on ATM and SONET are given in [Min89] and [BaC89], respectively. The CCITT and T1S1 working documents on ATM are surprisingly readable for an in-depth treatment.

PROBLEMS

2.1 Suppose that the output in Fig. 2.3(a) is $1 - e^{-2t/T}$ for $0 \leq t \leq T$, $(e^2 - 1)e^{-2t/T}$ for $t > T$, and zero for $t < 0$. Find the output in Fig. 2.3(b) using the linearity and time invariance of the filter.

2.2 Let $s(t) = 1$, for $0 \leq t \leq T$, and $s(t) = 0$ elsewhere. Let $h(t) = \alpha e^{-\alpha t}$ for $t \geq 0$ and $h(t) = 0$ for $t < 0$. Use the convolution equation to find the output when $s(t)$ is passed through a filter of impulse response $h(t)$.

2.3 Integrate Eq. (2.1) for $s(\tau) = e^{j2\pi f\tau}$ to get Eqs. (2.2) and (2.3). *Hint*: Transform the variable of integration by $\tau' = t - \tau$.

2.4 Suppose that a channel has the ideal low-pass frequency response $H(f) = 1$ for $-f_0 \leq f \leq f_0$ and $H(f) = 0$ elsewhere. Find the impulse response of the channel. [Use the inverse Fourier transform, Eq. (2.6).]

2.5 Let $s(t)$ be a given waveform and let $s_1(t) = s(\beta t)$ for some given positive β. Sketch the relationship between $s(t)$ and $s_1(t)$ for $\beta = 2$. Let $S(f)$ be the Fourier transform of $s(t)$. Find $S_1(f)$, the Fourier transform of $s_1(t)$, in terms of $S(f)$. Sketch the relationship for $\beta = 2$ [assume $S(f)$ real for your sketch].

2.6 Let $S(f)$ be the Fourier transform of $s(t)$.
- **(a)** Show that the Fourier transform of $s(t)\cos(2\pi f_0 t)$ is $[S(f - f_0) + S(f + f_0)]/2$.
- **(b)** Find the Fourier transform of $s(t)\cos^2(2\pi f_0 t)$ in terms of $S(f)$.

2.7 Suppose that the expected frame length on a link is 1000 bits and the standard deviation is 500 bits.
- **(a)** Find the expected time to transmit a frame on a 9600-bps link and on a 50,000-bps link.
- **(b)** Find the expected time and standard deviation of the time required to transmit 10^6 frames on each of the links above (assume that all frame lengths are statistically independent of each other).
- **(c)** The *rate* at which frames can be transmitted is generally defined as the reciprocal of the expected transmission time. Using the result in part (b), discuss whether this definition is reasonable.

Samsung Exhibit 1015, Page 157 of 573

2.8 Show that the final parity check in a horizontal and vertical parity check code, if taken as the modulo 2 sum of all the data bits, is equal to the modulo 2 sum of the horizontal parity checks and also equal to the modulo 2 sum of the vertical parity checks.

2.9 **(a)** Find an example of a pattern of six errors that cannot be detected by the use of horizontal and vertical parity checks. *Hint*: Each row with errors and each column with errors will contain exactly two errors.

(b) Find the number of different patterns of four errors that will not be detected by a horizontal and vertical parity check code; assume that the array is K bits per row and J bits per column.

2.10 Suppose that a parity check code has minimum distance d. Define the distance between a code word and a received string (of the same length) as the number of bit positions in which the two are different. Show that if the distance between one code word and a given string is less than $d/2$, the distance between any other code word and the given string must exceed $d/2$. Show that if a decoder maps a given received string into a code word at smallest distance from the string, all combinations of fewer than $d/2$ errors will be corrected.

2.11 Consider a parity check code with three data bits and four parity checks. Suppose that three of the code words are 1001011, 0101101, and 0011110. Find the rule for generating each of the parity checks and find the set of all eight code words. What is the minimum distance of this code?

2.12 Let $g(D) = D^4 + D^2 + D + 1$, and let $s(D) = D^3 + D + 1$. Find the remainder when $D^4 s(D)$ is divided by $g(D)$, using modulo 2 arithmetic.

2.13 Let $g(D) = D^L + g_{L-1}D^{L-1} + \cdots + g_1 D + 1$ (assume that $L > 0$). Let $z(D)$ be a nonzero polynomial with highest and lowest order terms of degree j and i, respectively; that is, $z(D) = D^j + z_{j-1}D^{j-1} + \cdots + D^i$ [or $z(D) = D^j$, if $i = j$]. Show that $g(D)z(D)$ has at least two nonzero terms. *Hint*: Look at the coefficients of D^{L+j} and of D^i.

2.14 Show that if $g(D)$ contains the factor $1 + D$, then all error sequences with an odd number of errors are detected. *Hint*: Recall that a nonzero error polynomial $e(D)$ is detected unless $e(D) = g(D)z(D)$ for some polynomial $z(D)$. Look at what happens if 1 is substituted for D in this equation.

2.15 For a given generator polynomial $g(D)$ of degree L and given data length K, let $c^{(i)}(D) = c_{L-1}^{(i)} D^{L-1} + \cdots + c_1^{(i)} D + c_0^{(i)}$ be the CRC resulting from the data string with a single 1 in position i [*i.e.*, $s(D) = D^i$ for $0 \le i \le K - 1$].

(a) For an arbitrary data polynomial $s(D)$, show that the CRC polynomial is $c(D) = \sum_{i=0}^{K-1} s_i c^{(i)}(D)$ (using modulo 2 arithmetic).

(b) Letting $c(D) = c_{L-1}D^{L-1} + \cdots + c_1 D + c_0$, show that

$$c_j = \sum_{i=0}^{K-1} s_i c_j^{(i)}; \qquad 0 \le j < L$$

This shows that each c_j is a parity check and that a cyclic redundancy check code is a parity check code.

2.16 **(a)** Consider a stop-and-wait ARQ strategy in which, rather than using a sequence number for successive packets, the sending DLC sends the number of times the given packet has been retransmitted. Thus, the format of the transmitted frames is $\boxed{j}\boxed{\text{packet}}\boxed{\text{CRC}}$, where j is 0 the first time a packet is transmitted, 1 on the first retransmission, and so on. The receiving DLC returns an ack or nak (without any request number) for each

Samsung Exhibit 1015, Page 158 of 573

frame it receives. Show by example that this strategy does not work correctly no matter what rule the receiving DLC uses for accepting packets. (Use the assumptions preceding Section 2.4.1.)

(b) Redo part (a) with the following two changes: (1) the number j above is 0 on the initial transmission of a packet and 1 on *all* subsequent retransmissions of that packet, and (2) frames have a fixed delay in each direction, are always recognized as frames, and the sending DLC never times-out, simply waiting for an ack or either a nak or error.

2.17 (a) Let T_t be the expected transmission time for sending a data frame. Let T_f be the feedback transmisson time for sending an ack or nak frame. Let T_d be the expected propagation and processing delay in one direction (the same in each direction). Find the expected time T between successive frame transmissions in a stop-and-wait system (assume that there are no other delays and that no frames get lost).

(b) Let p_t be the probability of frame error in a data frame (independent of frame length), and p_f the probability of error in a feedback ack or nak frame. Find the probability q that a data packet is correctly received and acked on a given transmission. Show that $1/q$ is the expected number of times a packet must be transmitted for a stop-and-wait system (assume independent errors on all frames).

(c) Combining parts (a) and (b), find the expected time required per packet. Evaluate for $T_t = 1$, $T_f = T_d = 0.1$, and $p_t = p_f = 10^{-3}$.

2.18 Redraw Figs. 2.25 and 2.26 with the same frame lengths and the same set of frames in error, but focusing on the packets from B to A. That is, show SN and the window for node B, and show RN and packets out for node A (assume that all frames are carrying packets).

2.19 Give an example in which go back n will deadlock if receiving DLCs ignore the request number RN in each frame not carrying the awaited packet. *Hint*: Construct a pattern of delays leading to a situation where RN at node B is n plus SN_{min} at node A, and similarly for A and B reversed.

2.20 Give an example in which go back n ARQ fails if the modulus m is equal to n. For $m > n$, give an example where the right hand inequality in Eq. (2.24) is satisfied with equality and an example where the right hand inequality in Eq. (2.25) is satisfied with equality.

2.21 Assume that at time t, node A has a given SN_{min} as the smallest packet number not yet acknowledged and $SN_{max} - 1$ as the largest number yet sent. Let T_m be the minimum packet transmission time on a link and T_d be the propagation delay. Show that RN at node B must lie in the range from SN_{min} to SN_{max} for times in the range from $t - T_m - T_d$ to $t + T_m + T_d$.

2.22 (a) Let T_{min} be the minimum transmission time for data frames and T_d be the propagation and processing delay in each direction. Find the maximum allowable value T_{max} for frame transmission time such that a go back n ARQ system (of given n) will never have to go back or wait in the absence of transmission errors or lost frames. *Hint*: Look at Fig. 2.31.

(b) Redo part (a) with the added possibility that isolated errors can occur in the feedback direction.

2.23 Assume that frame transmission times τ are exponentially distributed with, for example, probability density $p(\tau) = e^{-\tau}$, for $\tau \geq 0$. Assume that a new frame always starts in each direction as soon as the previous one in that direction terminates. Assume that transmission times are all independent and that processing and propagation delays are negligible. Show that the probability q that a given frame is not acked before $n - 1$ additional frames have been sent (*i.e.*, the window is exhausted) is $q = (1 + n)2^{-n}$. *Hint*: Note that the set of

Samsung Exhibit 1015, Page 159 of 573

times at which frame transmissions terminate at either node is a Poisson process and that a point in this process is equally likely to be a termination at A or B, with independence between successive terminations.

2.24 Show that if an isolated error in the feedback direction occurs on the ack of the given packet, then q in Problem 2.23 becomes $q = [2 + n + (n + 1)n/2]2^{-n-1}$.

2.25 Assume that frame transmission times τ have an Erlang distribution with probability density $p(\tau) = \tau e^{-\tau}$. Find the corresponding value of q in Problem 2.23. *Hint*: The termination times at A can be regarded as alternative points in a Poisson process and the same is true for B.

2.26 Let γ be the expected number of transmitted frames from A to B per successfully accepted packet at B. Let β be the expected number of transmitted frames from A to B between the transmission of a given frame and the reception of feedback about that frame (including the frame in transmission when the feedback arrives). Let p be the probability that a frame arriving at B contains errors (with successive frames assumed independent). Assume that A is always busy transmitting frames, that n is large enough that A never goes back in the absence of feedback, and that A always goes back on the next frame after hearing that the awaited frame contained errors. Show that γ satisfies $\gamma = 1 + p(\beta + \gamma)$. Define the efficiency η as $1/\gamma$, and find η as a function of β and p.

2.27 Assume that a selective repeat system has the property that the sending DLC always finds out whether a given transmitted packet was successfully received during or before the β^{th} frame transmission following the frame in which the given packet was transmitted. Give a careful demonstration that the sending DLC need never save more than $\beta + 1$ unacknowledged packets.

2.28 Find the efficiency of the ARPANET ARQ system under the assumptions that all eight virtual channels are always busy sending packets and that feedback always arrives about the fate of a given packet during or before the seventh frame transmission following the frame carrying the given packet.

2.29 Consider a generalized selective repeat ARQ system that operates in the following way: as in conventional go back n and selective repeat, let RN be the smallest numbered packet not yet correctly received, and let SN_{min} be the transmitter's estimate of RN. Let y_{top} be the largest-numbered packet correctly received and accepted (thus, for go back n, y_{top} would be $RN - 1$, whereas for conventional selective repeat, y_{top} could be as large as $RN + n - 1$). The rule at the transmitter is the same as for conventional go back n or selective repeat: The number z of the packet transmitted must satisfy

$$SN_{min} \leq z \leq SN_{min} + n - 1$$

The rule at the receiver, for some given positive number k, is that a correctly received packet with number z is accepted if

$$RN \leq z \leq y_{\text{top}} + k \tag{2.46}$$

(a) Assume initially that z (rather than $SN = z \bmod m$) is sent with each packet and that RN (rather than $RN \bmod m$) is sent with each packet in the reverse direction. Show that for each received packet at the receiver,

$$z \leq RN + n - 1$$

$$z \geq y_{\text{top}} - n + 1 \tag{2.47}$$

Samsung Exhibit 1015, Page 160 of 573

(b) Now assume that $SN = z \bmod m$ and RN is sent mod m. When z is received Eq. (2.47) is satisfied, but erroneous operation will result if $z - m$ or $z + m$ lie in the window specified by Eq. (2.46). How large need m be to guarantee that $z + m > y_{\text{top}} + k$ [*i.e.*, that $z + m$ is outside the window of Eq. (2.46)]?

(c) Is the value of m determined in part (b) large enough to ensure that $z - m < RN$?

(d) How large need m be (as a function of n and k) to ensure correct operation?

(e) Interpret the special cases $k = 1$ and $k = n$.

2.30 **(a)** Apply the bit stuffing rules of Section 2.5.2 to the following frame:
0 1 1 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 0

(b) Suppose that the following string of bits is received:
0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 0 0 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 0 1 1
0 0 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 0
Remove the stuffed bits and show where the actual flags are.

2.31 Suppose that the bit stuffing rule of Section 2.5.2 is modified to stuff a 0 only after the appearance of 01^5 in the original data. Carefully describe how the destuffing rule at the receiver must be modified to work with this change. Show how your rule would destuff the following string:
0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 0
If your destuffing rule is correct, you should remove only two 0's and find only one actual flag.

2.32 Bit stuffing must avoid the appearance of 01^6 within the transmitted frame, so we accept as given that an original string 01^6 will always be converted into 01^501. Use this, plus the necessity to destuff correctly at the receiver, to show that a 0 must always be stuffed after 01^5. *Hint*: Consider the data string $01^501x_1x_2\ldots$. If a 0 is not stuffed after 01^5, the receiver cannot distinguish this from $01^6x_1x_2\ldots$ after stuffing, so stuffing is required in this case. Extend this argument to $01^50^k1x_1x_2\ldots$ for any $k > 1$.

2.33 Suppose that the string 0101 is used as the bit string to indicate the end of a frame and the bit stuffing rule is to insert a 0 after each appearance of 010 in the original data; thus, 010101 would be modified by stuffing to 01001001. In addition, if the frame proper ends in 01, a 0 would be stuffed after the first 0 in the actual terminating string 0101. Show how the string 11011010010101011101 would be modified by this rule. Describe the destuffing rule required at the receiver. How would the string 11010001001001100101 be destuffed?

2.34 Let $A = E\{K\}2^{-j} + j + 1$ be the upper bound on overhead in Eq. (2.34), and let $j = \lfloor \log_2 E\{K\} \rfloor$. Show that A satisfies

$$1.914\ldots + \log_2 E\{K\} \leq A \leq 2 + \log_2 E\{K\}$$

Find the analytic expression for $1.914\ldots$. *Hint*: Define γ as $\log_2 E\{K\} - j$ and find the minimum and maximum of $A - \log_2 E\{K\}$ as γ varies between 0 and 1.

2.35 Show that the probability that errors will cause a flag 01^60 to appear within a transmitted frame is approximately $(1/32)Kp$, where K is the number of bits in the frame proper before stuffing and p is the probability of bit error. Assume that p is very small and that the probability of multiple errors in a frame is negligible. Assume that the bits of the original frame are IID (independent, identically distributed) with equal probability of 0 and 1. *Hint*: This is trickier than it appears, since the bits after stuffing are no longer IID. First, find the probability that a stuffed bit in error causes a flag (not an abort) to appear and use the analysis of Section 2.5.2 to approximate the expected number of stuffed bits as $K2^{-6}$. Then find the probability of a flag appearing due to errors in the original bits of the frame.

Samsung Exhibit 1015, Page 161 of 573

2.36 A certain virtual circuit network is designed for the case where all sessions carry heavy, almost steady traffic. It is decided to serve the various sessions using each given link in round-robin order, first sending one packet from the first session, then one from the second, and so on, up to the last, and then starting over with the first, thus avoiding the necessity of sending a session number with each packet. To handle the unlikely possibility that a session has no packet to send on a link at its turn, we place a k-bit flag, 01^{k-1}, in front of the information packet for the next session that has a packet. This flag is followed by a unary encoding of the number of sessions in order with nothing to send, 1 for one session, 01 for two, 001 for three, and so on. Thus, for example, if a packet for the second session has just been sent, sessions 3 and 4 have no packets, and session 5 has the packet $\overline{x}_5$, we send $01^{k-1}01\overline{x}_5$ as the next packet. In addition, an insertion is used if an information packet happens to start with 01^{k-2}. Otherwise, insertions are not required in information packets, and information packets carry their own length information.

(a) Assume that p is the probability that a session has no packet when it is that session's turn; this event is assumed independent of other sessions and of past history. Find the expected number of overhead bits per transmitted packet for flags, unary code, and insertions. Ignore, as negligible, any problems of what to do if no session has a packet to send.

(b) Suppose that the packets above, as modified by flags, unary code, and insertions, are the inputs to a data link control system using ARQ for error control and flags to indicate the end of packets. Explain if any problems arise due to the use of flags both on the DLC system and in the higher-level scheme for identifying packets with sessions.

2.37 Consider a stop-and-wait data link control protocol. Each transmitted frame contains the sequence number, modulo 2, of the contained packet. When a frame arrives at the receiver, the sequence number of the packet is compared with the packet number awaited by the receiver; if the numbers are the same modulo 2 and the CRC is satisfied, the packet is accepted and the awaited packet number is incremented. An ack containing the new awaited packet number modulo 2 is sent back to the transmitter. If a frame is received with correct CRC but the wrong sequence number, an ack (containing the awaited packet number modulo 2) is also sent back to the transmitter. The new twist here is that frames can go out of order on the channel. More precisely, there is a known maximum delay T on the channel. If a packet is transmitted at some time t, it is either not received at all or is received at an arbitrary time in the interval from t to $t + T$, independently of other transmissions. The return channel for the acks behaves the same way with the same maximum delay T. Assume that the receiver sends acks instantaneously upon receiving packets with valid CRCs.

(a) Describe rules for the transmitter to follow so as to ensure that each packet is eventually accepted, once and in order, by the receiver. You are not to put any extra protocol information in the packets nor to change the receiver's operation. You should try to leave as much freedom as possible to the transmitter's operation subject to the restriction of correct operation.

(b) Explain why it is impossible to achieve correct operation if there is no bound on delay.

2.38 Consider a stop-and-wait system with two-way traffic between nodes A and B. Assume that data frames are all of the same length and require D seconds for transmission. Acknowledgment frames require R seconds for transmission and there is a propagation delay P on the link. Assume that A and B both have an unending sequence of packets to send, assume that no transmission errors occur, and assume that the time-out interval at which a

Samsung Exhibit 1015, Page 162 of 573

node resends a previously transmitted packet is very large. Finally, assume that each node sends new data packets and acknowledgments as fast as possible, subject to the rules of the stop-and-wait protocol.

(a) Show that the rate at which packets are transmitted in each direction is $(D+R+2P)^{-1}$. Show that this is true whether or not the starting time between A and B is synchronized.

(b) Assume, in addition, that whenever a node has both an acknowledgment and a data packet to send, the acknowledgment is piggybacked onto the data frame (which still requires D seconds). Assume that node B does not start transmitting data until the instant that it receives the first frame from A (*i.e.*, B's first frame contains a piggybacked ack). Show that all subsequent frames in each direction are data frames with a piggybacked ack. Show that the rate in each direction is now $(2D+2P)^{-1}$. Note that if $D > R$, this rate is less than that in part (a).

2.39 **(a)** Consider a data link control using fixed-length packets of K bits each. Each message is broken into packets using fill in the last packet of the message as required; for example, a message of 150 bits is broken into two packets of 100 bits each, using 50 bits of fill. Modify Eq. (2.42) to include fill for fixed-length packets.

(b) Find $E\{TC\}$ using the approximation $E\{\lceil M/K \rceil\} \approx E\{M/K\} + 1/2$; find the value of K that minimizes this approximation to $E\{TC\}$.

(c) Note that for $j = 1$ the value of K_{max} in Eq. (2.44) is infinite, whereas the minimizing fixed length K is finite. Explain why this is reasonable.

2.40 **(a)** Assume that nodes A and B both regard the link between them as up according to the balanced initialization and disconnect protocol of Section 2.7.3. Assume that at time t node A starts to disconnect (*i.e.*, transmits DISC). Show, using the same assumptions as in Section 2.4, that each node eventually regards the link as down.

(b) Give an example for the situation in part (a), where node A regards the link as down and then starts to initialize the link before node B regards the link as down; check your proof for part (a) in light of this example. Show that B regards the link as down before A again regards the link as up.

(c) Now suppose that node A regards the link as up and node B is initializing the link but has not yet received ACKI from A. Suppose that A starts to disconnect. Show that B eventually regards the link as up and then starts to disconnect and that after that A and B both eventually regard the link as down.

2.41 **(a)** Create an example of incorrect operation, analogous to that of Fig. 2.46, for the master–slave initialization protocol under the assumption that a node, after recovering from a node failure, starts in the up state and initializes by first sending DISC and then INIT. *Hint:* Use the same structure as Fig. 2.46; that is, assume that after each failure, node A receives the messages from B that were generated by B in response to the messages in the earlier period at A.

(b) Now suppose that an arbitrary protocol is being used for initialization. It is known that node A, on recovery from a failure, will send message x. Node B, on recovering from failure and receiving message x, will send message y, and node A, upon receiving y under these circumstances, will be correctly initialized. Construct a sequence of failures and delays such that the protocol fails, assuming only that the nodes operate as above under the above conditions. *Hint:* Follow the hint in part (a), but also assume a sequence of failures at B, with B receiving responses in one period to messages in the preceding period.

Samsung Exhibit 1015, Page 163 of 573

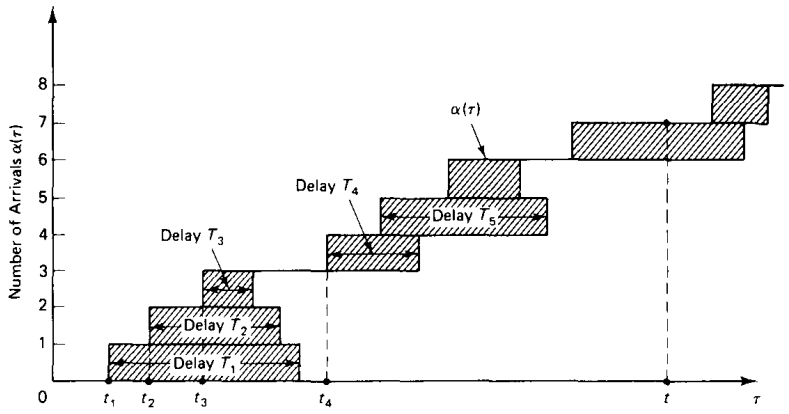
2.42 The balanced initialization protocol of Section 2.7.3 piggybacks acks on each INIT or DISC message. Explain why it is a bad idea to include also either an INIT or DISC message every time that an ACKI or ACKD must be sent.

2.43 Consider a datagram network and assume that M is the maximum number of packets that can be sent by a session while a given packet (in either direction) still exists within the network. Assume that selective repeat ARQ is used for the session with a window size of n. Show that the modulus m must satisfy

$$m \geq 2n + M$$

Samsung Exhibit 1015, Page 164 of 573

3



Delay Models in Data Networks

3.1 INTRODUCTION

One of the most important performance measures of a data network is the average delay required to deliver a packet from origin to destination. Furthermore, delay considerations strongly influence the choice and performance of network algorithms, such as routing and flow control. For these reasons, it is important to understand the nature and mechanism of delay, and the manner in which it depends on the characteristics of the network.

Queueing theory is the primary methodological framework for analyzing network delay. Its use often requires simplifying assumptions since, unfortunately, more realistic assumptions make meaningful analysis extremely difficult. For this reason, it is sometimes impossible to obtain accurate quantitative delay predictions on the basis of queueing models. Nevertheless, these models often provide a basis for adequate delay approximations, as well as valuable qualitative results and worthwhile insights.

In what follows, we will mostly focus on packet delay within the communication subnet (*i.e.*, the network layer). This delay is the sum of delays on each subnet link traversed by the packet. Each link delay in turn consists of four components.

Samsung Exhibit 1015, Page 165 of 573

1. The *processing delay* between the time the packet is correctly received at the head node of the link and the time the packet is assigned to an outgoing link queue for transmission. (In some systems, we must add to this delay some additional processing time at the DLC and physical layers.)
2. The *queueing delay* between the time the packet is assigned to a queue for transmission and the time it starts being transmitted. During this time, the packet waits while other packets in the transmission queue are transmitted.
3. The *transmission delay* between the times that the first and last bits of the packet are transmitted.
4. The *propagation delay* between the time the last bit is transmitted at the head node of the link and the time the last bit is received at the tail node. This is proportional to the physical distance between transmitter and receiver; it can be relatively substantial, particularly for a satellite link or a very high speed link.

This accounting neglects the possibility that a packet may require retransmission on a link due to transmission errors or various other causes. For most links in practice, other than multiaccess links to be considered in Chapter 4, retransmissions are rare and will be neglected. The propagation delay depends on the physical characteristics of the link and is independent of the traffic carried by the link. The processing delay is also independent of the amount of traffic handled by the corresponding node if computation power is not a limiting resource. This will be assumed in our discussion. Otherwise, a separate processing queue must be introduced prior to the transmission queues. Most of our subsequent analysis focuses on the queueing and transmission delays. We first consider a single transmission line and analyze some classical queueing models. We then take up the network case and discuss the type of approximations involved in deriving analytical delay models.

While our primary emphasis is on packet-switched network models, some of the models developed are useful in a circuit-switched network context. Indeed, queueing theory was developed extensively in response to the need for performance models in telephony.

3.1.1 Multiplexing of Traffic on a Communication Link

The communication link considered is viewed as a bit pipe over which a given number of bits per second can be transmitted. This number is called the *transmission capacity* of the link. It depends on both the physical channel and the interface (*e.g.*, modems), and is simply the rate at which the interface accepts bits. The link capacity may serve several traffic streams (*e.g.*, virtual circuits or groups of virtual circuits) multiplexed on the link. The manner of allocation of capacity among these traffic streams has a profound effect on packet delay.

In the most common scheme, *statistical multiplexing*, the packets of all traffic streams are merged into a single queue and transmitted on a first-come first-serve basis. A variation of this scheme, which has roughly the same average delay per packet, maintains

a separate queue for each traffic stream and serves the queues in sequence one packet at a time. However, if the queue of a traffic stream is empty, the next traffic stream is served and no communication resource is wasted. Since the entire transmission capacity C (bits/sec) is allocated to a single packet at a time, it takes L/C seconds to transmit a packet that is L bits long.

In *time-division* (TDM) *and frequency-division multiplexing* (FDM) with m traffic streams, the link capacity is essentially subdivided into m portions—one per traffic stream. In FDM, the channel bandwidth W is subdivided into m channels each with bandwidth W/m (actually slightly less because of the need for guard bands between channels). The transmission capacity of each channel is roughly C/m, where C is the capacity that would be obtained if the entire bandwidth were allocated to a single channel. The transmission time of a packet that is L bits long is Lm/C, or m times larger than in the corresponding statistical multiplexing scheme. In TDM, allocation is done by dividing the time axis into slots of fixed length (*e.g.*, one bit or one byte long, or perhaps one packet long for fixed length packets). Again, conceptually, we may view the communication link as consisting of m separate links with capacity C/m. In the case where the slots are short relative to packet length, we may again regard the transmission time of a packet L bits long as Lm/C. In the case where the slots are of packet length, the transmission time of an L bit packet is L/C, but there is a wait of $(m-1)$ packet transmission times between packets of the same stream.

One of the themes that will emerge from our queueing analysis is that statistical multiplexing has smaller average delay per packet than either TDM or FDM. This is particularly true when the traffic streams multiplexed have a relatively low duty cycle. The main reason for the poor delay performance of TDM and FDM is that communication resources are wasted when allocated to a traffic stream with a momentarily empty queue, while other traffic streams have packets waiting in their queue. For a traffic analogy, consider an m-lane highway and two cases. In one case, cars are not allowed to cross over to other lanes (this corresponds to TDM or FDM), while in the other case, cars can change lanes (this corresponds roughly to statistical multiplexing). Restricting crossover increases travel time for the same reason that the delay characteristics of TDM or FDM are poor: namely, some system resources (highway lanes or communication channels) may not be utilized, while others are momentarily stressed.

Under certain circumstances, TDM or FDM may have an advantage. Suppose that each traffic stream has a "regular" character (*i.e.*, all packets arrive sufficiently apart so that no packet has to wait while the preceding packet is transmitted.) If these traffic streams are merged into a single queue, it can be shown that the average delay per packet will decrease, but the variance of waiting time in queue will generally become positive (for an illustration, see Prob. 3.7). Therefore, if maintaining a small variability of delay is more important than decreasing delay, it may be preferable to use TDM or FDM. Another advantage of TDM and FDM is that there is no need to include identification of the traffic stream on each packet, thereby saving some overhead and simplifying packet processing at the nodes. Note also that when overhead is negligible, one can afford to make packets very small, thereby reducing delay through pipelining (cf. Fig. 2.37).

Samsung Exhibit 1015, Page 167 of 573

3.2 QUEUEING MODELS—LITTLE'S THEOREM

We consider queueing systems where customers arrive at random times to obtain service. In the context of a data network, customers represent packets assigned to a communication link for transmission. Service time corresponds to the packet transmission time and is equal to L/C, where L is the packet length in bits and C is the link transmission capacity in bits/sec. In this chapter it is convenient to ignore the layer 2 distinction between packets and frames; thus packet lengths are taken to include frame headers and trailers. In a somewhat different context (which we will not emphasize very much), customers represent ongoing conversations (or virtual circuits) between points in a network and service time corresponds to the duration of a conversation. In a related context, customers represent active calls in a telephone or circuit switched network and again service time corresponds to the duration of the call.

We shall be typically interested in estimating quantities such as:

1. The average number of customers in the system (*i.e.*, the "typical" number of customers either waiting in queue or undergoing service)
2. The average delay per customer (*i.e.*, the "typical" time a customer spends waiting in queue plus the service time).

These quantities will be estimated in terms of known information such as:

1. The customer arrival rate (*i.e.*, the "typical" number of customers entering the system per unit time)
2. The customer service rate (*i.e.*, the "typical" number of customers the system serves per unit time when it is constantly busy)

In many cases the customer arrival and service rates are not sufficient to determine the delay characteristics of the system. For example, if customers tend to arrive in groups, the average customer delay will tend to be larger than when their arrival times are regularly spaced apart. Thus to predict average delay, we will typically need more detailed (statistical) information about the customer interarrival and service times. In this section, however, we will largely ignore the availability of such information and see how far we can go without it.

3.2.1 Little's Theorem

We proceed to clarify the meaning of the terms "average" and "typical" that we used somewhat liberally above in connection with the number of customers in the system, the customer delay, and so on. In doing so we will derive an important result known as *Little's Theorem*.

Suppose that we observe a sample history of the system from time $t = 0$ to the indefinite future and we record the values of various quantities of interest as time

progresses. In particular, let

$N(t)$ = Number of customers in the system at time t

$\alpha(t)$ = Number of customers who arrived in the interval $[0, t]$

T_i = Time spent in the system by the i^{th} arriving customer

Our intuitive notion of the "typical" number of customers in the system observed up to time t is

$$N_t = \frac{1}{t} \int_0^t N(\tau)\, d\tau$$

which we call the *time average of* $N(\tau)$ *up to time* t. Naturally, N_t changes with the time t, but in many systems of interest, N_t tends to a steady-state N as t increases, that is,

$$N = \lim_{t \to \infty} N_t$$

In this case, we call N the *steady-state time average* (or simply time average) of $N(\tau)$. It is also natural to view

$$\lambda_t = \frac{\alpha(t)}{t}$$

as the *time average arrival rate* over the interval $[0, t]$. The *steady-state arrival rate* is defined as

$$\lambda = \lim_{t \to \infty} \lambda_t$$

(assuming that the limit exists). The *time average of the customer delay up to time* t is similarly defined as

$$T_t = \frac{\sum_{i=0}^{\alpha(t)} T_i}{\alpha(t)} \tag{3.1}$$

that is, the average time spent in the system per customer up to time t. The *steady-state time average customer delay* is defined as

$$T = \lim_{t \to \infty} T_t$$

(assuming that the limit exists).

It turns out that the quantities N, λ, and T above are related by a simple formula that makes it possible to determine one given the other. This result, known as Little's Theorem, has the form

$$N = \lambda T$$

Little's Theorem expresses the natural idea that crowded systems (large N) are associated with long customer delays (large T) and reversely. For example, on a rainy day, traffic on a rush hour moves slower than average (large T), while the streets are more crowded (large N). Similarly, a fast-food restaurant (small T) needs a smaller waiting room (small N) than a regular restaurant for the same customer arrival rate.

Samsung Exhibit 1015, Page 169 of 573

The theorem is really an accounting identity and its derivation is very simple. We will give a graphical proof under some simplifying assumptions. Suppose that the system is initially empty [$N(0) = 0$] and that customers depart from the system in the order they arrive. Then the number of arrivals $\alpha(t)$ and departures $\beta(t)$ up to time t form a staircase graph as shown in Fig. 3.1. The difference $\alpha(t) - \beta(t)$ is the number in the system $N(t)$ at time t. The shaded area between the graphs of $\alpha(\tau)$ and $\beta(\tau)$ can be expressed as

$$\int_0^t N(\tau)\, d\tau$$

and if t is any time for which the system is empty [$N(t) = 0$], the shaded area is also equal to

$$\sum_{i=1}^{\alpha(t)} T_i$$

Dividing both expressions above with t, we obtain

$$\frac{1}{t}\int_0^t N(\tau)\, d\tau = \frac{1}{t}\sum_{i=1}^{\alpha(t)} T_i = \frac{\alpha(t)}{t}\frac{\sum_{i=1}^{\alpha(t)} T_i}{\alpha(t)}$$

or equivalently,

$$N_t = \lambda_t T_t \tag{3.2}$$

Little's Theorem is obtained assuming that

$$N_t \to N,\ \lambda_t \to \lambda,\ T_t \to T$$

and that the system becomes empty infinitely often at arbitrarily large times. With a minor modification in the preceding argument, the latter assumption becomes unnecessary. To see this, note that the shaded area in Fig. 3.1 lies between $\sum_{i=1}^{\alpha(t)} T_i$ and $\sum_{i=1}^{\beta(t)} T_i$, so we obtain

$$\frac{\beta(t)}{t}\frac{\sum_{i=1}^{\beta(t)} T_i}{\beta(t)} \le N_t \le \lambda_t T_t$$

Assuming that $N_t \to N$, $\lambda_t \to \lambda$, $T_t \to T$, and that the departure rate $\beta(t)/t$ up to time t tends to the steady-state arrival rate λ, we obtain Little's Theorem.

The simplifying assumptions used in the preceding graphical proof can be relaxed considerably, and one can construct an analytical proof that requires only that the limits $\lambda = \lim_{t\to\infty} \alpha(t)/t$, $\delta = \lim_{t\to\infty} \beta(t)/t$, and $T = \lim_{t\to\infty} T_t$ exist, and that $\lambda = \delta$. In particular, it is not necessary that customers are served in the order they arrive, and that the system is initially empty (see Problem 3.41). Figure 3.2 explains why the order of customer service is not essential for the validity of Little's Theorem.

3.2.2 Probabilistic Form of Little's Theorem

Little's Theorem admits also a probabilistic interpretation provided that we can replace time averages with statistical or ensemble averages, as we now discuss. Our preceding

Samsung Exhibit 1015, Page 170 of 573

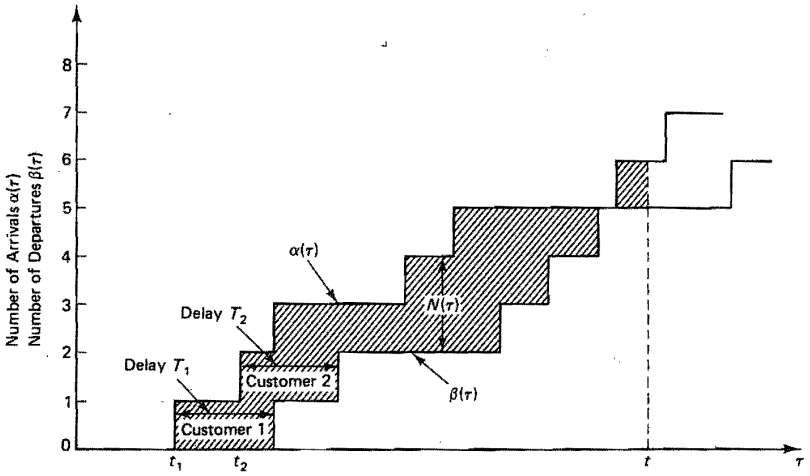


Figure 3.1 Proof of Little's Theorem. If the system is empty at time t $[N(t) = 0]$, the shaded area can be expressed both as $\int_0^t N(\tau)\,d\tau$ and as $\sum_{i=1}^{\alpha(t)} T_i$. Dividing both expressions by t, equating them, and taking the limit as $t \to \infty$ gives Little's Theorem. If $N(t) > 0$, we have

$$\sum_{i=1}^{\beta(t)} T_i \leq \int_0^t N(\tau)\,d\tau \leq \sum_{i=1}^{\alpha(t)} T_i$$

and assuming that the departure rate $\beta(t)/t$ up to time t tends to the steady-state arrival rate λ, the same argument applies.

analysis deals with a single sample function; now we will look at the probabilities of many sample functions and other events.

We first need to clarify the meaning of an ensemble average. Let us denote

$$p_n(t) = \text{Probability of } n \text{ customers in the system at time } t$$

(waiting in queue or under service)

In a typical situation we are given the initial probabilities $p_n(0)$ at time 0, together with enough statistical information to determine, at least in principle, the probabilities $p_n(t)$ for all times t. For example, the probability distribution of the time between two successive arrivals (the interarrival time), and the probability distribution of the customers' service time at various parts of the queueing system may be given. Then the average number in the system at time t is given by

$$\overline{N}(t) = \sum_{n=0}^{\infty} n p_n(t)$$

Note that both $\overline{N}(t)$ and $p_n(t)$ depend on t as well as the initial probability distribution

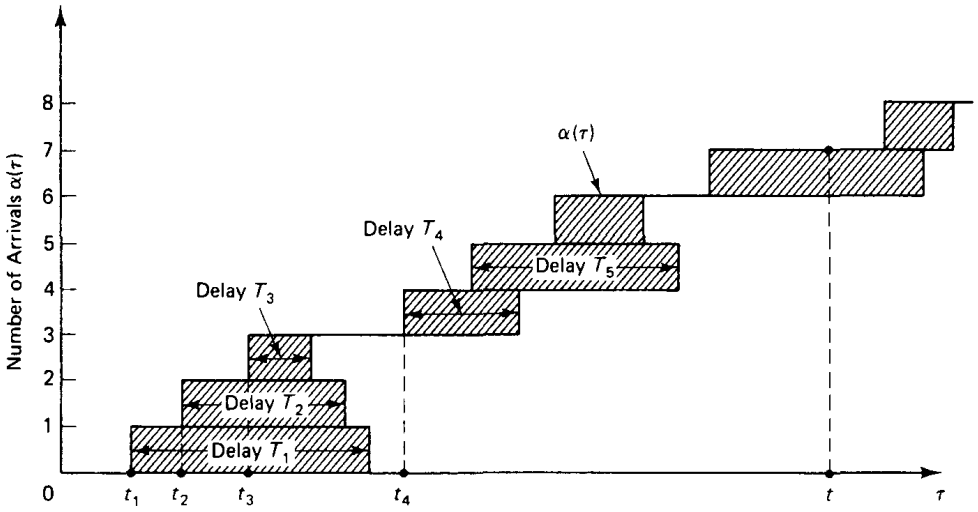


Figure 3.2 Informal justification of Little's Theorem without assuming first-in first-out customer service. The shaded area can be expressed both as $\int_0^t N(\tau)\,d\tau$ and as $\sum_{i\in D(t)} T_i + \sum_{i\in \overline{D}(t)}(t - t_i)$, where $D(t)$ is the set of customers that have departed up to time t, $\overline{D}(t)$ is the set of customers that are still in the system at time t, and t_i is the time of arrival of the i^{th} customer. Dividing both expressions by t, equating them, and taking the limit as $t \to \infty$ gives Little's Theorem.

$\{p_0(0),\ p_1(0), \ldots\}$. However, the queueing systems that we will consider typically reach a steady-state in the sense that for some p_n (independent of the initial distribution), we have

$$\lim_{t\to\infty} p_n(t) = p_n\,, \quad n = 0, 1, \ldots.$$

The average number in the system at steady-state is given by

$$\overline{N} = \sum_{n=0}^{\infty} n p_n$$

and we typically have

$$\overline{N} = \lim_{t\to\infty} \overline{N}(t)$$

Regarding average delay per customer, we are typically given enough statistical information to determine in principle the probability distribution of delay of each individual customer (*i.e.*, the first, second, etc.). From this, we can determine the average delay of each customer. The average delay of the k^{th} customer, denoted $\overline{T}_k$, typically converges as $k \to \infty$ to a steady-state value

$$\overline{T} = \lim_{k\to\infty} \overline{T}_k$$

To make the connection with time averages, we note that almost every system of interest to us is *ergodic* in the sense that the time average, $N = \lim_{t\to\infty} N_t$, of a sample

function is, with probability 1, equal to the steady-state average $\overline{N} = \lim_{t\to\infty} \overline{N}(t)$, that is,

$$N = \lim_{t\to\infty} N_t = \lim_{t\to\infty} \overline{N}(t) = \overline{N}$$

Similarly, for the systems of interest to us, the time average of customer delay T is also equal (with probability 1) to the steady-state average delay $\overline{T}$, that is,

$$T = \lim_{k\to\infty} \frac{1}{k} \sum_{i=1}^{k} T_i = \lim_{k\to\infty} \overline{T}_k = \overline{T}$$

Under these circumstances, Little's formula, $N = \lambda T$, holds with N and T being stochastic averages and with λ given by

$$\lambda = \lim_{t\to\infty} \frac{\text{Expected number of arrivals in the interval } [0, t]}{t}$$

The equality of long term time and ensemble averages of various stochastic processes will often be accepted in this chapter on intuitive grounds. This equality can often be shown by appealing to general results from the theory of Markov chains (see Appendix A, at the end of this chapter, which states these results without proof). In other cases, this equality, though highly plausible, requires a specialized mathematical proof. Such a proof is typically straightforward for an expert in stochastic processes but requires background that is beyond what is assumed in this book. In what follows we will generally use the time average notation T and N in place of the ensemble average notation $\overline{T}$ and $\overline{N}$, respectively, implicitly assuming the equality of the corresponding time and ensemble averages.

3.2.3 Applications of Little's Theorem

The significance of Little's Theorem is due in large measure to its generality. It holds for almost every queueing system that reaches a steady-state. The system need not consist of just a single queue. Indeed, with appropriate interpretation of the terms N, λ, and T, the theorem holds for many complex arrival–departure systems. The following examples illustrate its broad applicability.

Example 3.1

If λ is the arrival rate in a transmission line, N_Q is the average number of packets waiting in queue (but not under transmission), and W is the average time spent by a packet waiting in queue (not including the transmission time), Little's Theorem gives

$$N_Q = \lambda W$$

Furthermore, if $\overline{X}$ is the average transmission time, then Little's Theorem gives the average number of packets under transmission as

$$\rho = \lambda \overline{X}$$

Since at most one packet can be under transmission, ρ is also the line's *utilization factor*, (*i.e.*, the proportion of time that the line is busy transmitting a packet).

Example 3.2

Consider a network of transmission lines where packets arrive at n different nodes with corresponding rates $\lambda_1, \ldots, \lambda_n$. If N is the average total number of packets inside the network, then (regardless of the packet length distribution and method for routing packets) the average delay per packet is

$$T = \frac{N}{\sum_{i=1}^{n} \lambda_i}$$

Furthermore, Little's Theorem also yields $N_i = \lambda_i T_i$, where N_i and T_i are the average number in the system and average delay of packets arriving at node i, respectively.

Example 3.3

A packet arrives at a transmission line every K seconds with the first packet arriving at time 0. All packets have equal length and require αK seconds for transmission where $\alpha < 1$. The processing and propagation delay per packet is P seconds. The arrival rate here is $\lambda = 1/K$. Because packets arrive at a regular rate (equal interarrival times), there is no delay for queueing, so the time T a packet spends in the system (including the propagation delay) is

$$T = \alpha K + P$$

According to Little's Theorem, we have

$$N = \lambda T = \alpha + \frac{P}{K}$$

Here the number in the system $N(t)$ is a deterministic function of time. Its form is shown in Fig. 3.3 for the case where $K < \alpha K + P < 2K$, and it can be seen that $N(t)$ does not converge to any value (the system never reaches statistical equilibrium). However, Little's Theorem holds with N viewed as a time average.

Example 3.4

Consider a window flow control system (as described in Section 2.8.1) with a window of size W for each session. Since the number of packets in the system per session is always no more than W, Little's Theorem asserts that the arrival rate λ of packets into the system for each session, and the average packet delay are related by $W \geq \lambda T$. Thus, if congestion builds up in the network and T increases, λ must eventually decrease. Next, suppose that the network is congested and capable of delivering only λ packets per unit time for each session. Assuming that acknowledgment delays are negligible relative to the forward packet delays, we have $W \simeq \lambda T$. Then, increasing the window size W for all sessions merely serves to increase the delay T without appreciably changing λ.

Example 3.5

Consider a queueing system with K servers, and with room for at most $N \geq K$ customers (either in queue or in service). The system is always full; we assume that it starts with N customers and that a departing customer is immediately replaced by a new customer. (Queueing systems of this type are called *closed* and are discussed in detail in Section 3.8.) Suppose that the average customer service time is $\overline{X}$. We want to find the average

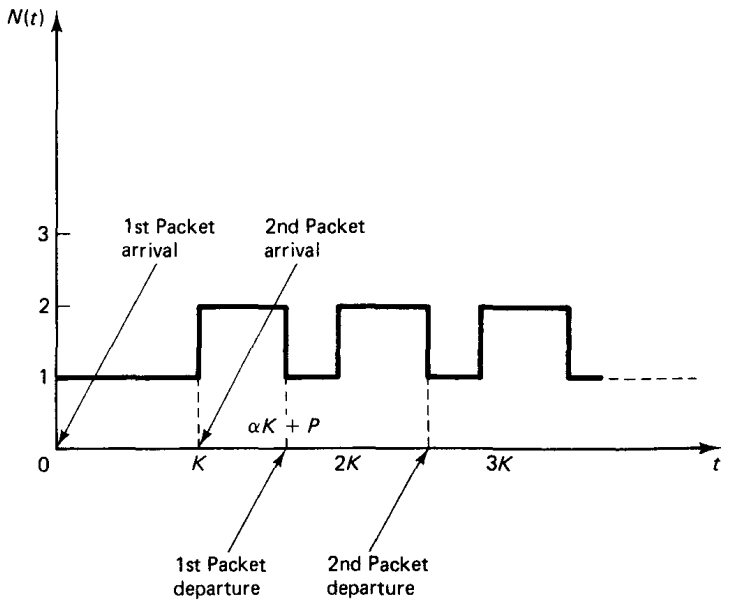


Figure 3.3 The number in the system in Example 3.3, $N(t)$, is deterministic and does not converge as $t \to \infty$. Little's Theorem holds with N, λ, and T viewed as time averages.

customer time in the system T. We apply Little's Theorem twice, first for the entire system, obtaining $N = \lambda T$, and then for the service portion of the system, obtaining $K = \lambda \overline{X}$ (since all servers are constantly busy). By eliminating λ in these two relations we have

$$T = \frac{N\overline{X}}{K}$$

Consider also the same system but under different customer arrival assumptions. In particular, assume that customers arrive at a rate λ but are blocked (and lost) from the system if they find the system full. Then the number of servers that are busy may be less than K. Let $\overline{K}$ be the average number of busy servers and let β be the proportion of customers that are blocked from entering the system. Applying Little's Theorem to the service portion of the system, we obtain

$$\overline{K} = (1 - \beta)\lambda\overline{X}$$

from which

$$\beta = 1 - \frac{\overline{K}}{\lambda\overline{X}}$$

Since $\overline{K} \leq K$, we obtain a lower bound on the blocking probability, namely,

$$\beta \geq 1 - \frac{K}{\lambda\overline{X}}$$

Example 3.6

A transmission line serves m packet streams, also called users, in round-robin cycles. In each cycle, some packets of user 1 are transmitted, followed by some packets of user 2, and

so on, until finally, some packets of user m are transmitted. An overhead period of average length A_i precedes the transmission of the packets of user i in each cycle. Systems of this type are called *polling systems* and are discussed in detail in Section 3.5.2.

The arrival rate and the average transmission time of the packets of user i are λ_i and $\overline{X}_i$, respectively. From Little's theorem we know that the fraction of time the transmission line is busy transmitting packets of user i is $\lambda_i \overline{X}_i$. Consider now the time intervals used for overhead of user i. We can view these intervals as "packets" with average transmission time A_i. The arrival rate of these "packets" is $1/L$, where L is the average cycle length, and as before, we may use Little's theorem to assert that the fraction of time used for transmission of these "packets" is A/L, where $A = A_1 + A_2 + \cdots + A_m$. Therefore, we have

$$1 = \frac{A}{L} + \sum_{i=1}^{m} \lambda_i \overline{X}_i$$

which yields the average cycle length

$$L = \frac{A}{1 - \sum_{i=1}^{m} \lambda_i \overline{X}_i}$$

Example 3.7 Estimating Throughput in a Time-Sharing System

Little's Theorem can sometimes be used to provide bounds on the attainable system throughput λ. In particular, known bounds on N and T can be translated into throughput bounds via $\lambda = N/T$. As an example, consider a time-sharing computer system with N terminals. A user logs into the system through a terminal, and after an initial reflection period of average length R, submits a job that requires an average processing time P at the computer. Jobs queue up inside the computer and are served by a single CPU according to some unspecified priority or time-sharing rule.

We would like to get estimates of the throughput sustainable by the system (in jobs per unit time), and corresponding estimates of the average delay of a user. Since we are interested in maximum attainable throughput, we assume that there is always a user ready to take the place of a departing user, so the number of users in the system is always N. For this reason, it is appropriate to adopt a model whereby a departing user immediately reenters the system as shown in Fig. 3.4.

Applying Little's Theorem to the portion of the system between the entry to the terminals and the exit of the system (points A and C in Fig. 3.4), we have

$$\lambda = \frac{N}{T} \tag{3.3}$$

where T is the average time a user spends in the system. We have

$$T = R + D \tag{3.4}$$

where D is the average delay between the time a job is submitted to the computer and the time its execution is completed. Since D can vary between P (case where the user's job does not have to wait for other jobs to be completed) and NP (case where the user's job has to wait for the jobs of all the other users; compare with Example 3.5), we have

$$R + P \leq T \leq R + NP \tag{3.5}$$

Samsung Exhibit 1015, Page 176 of 573

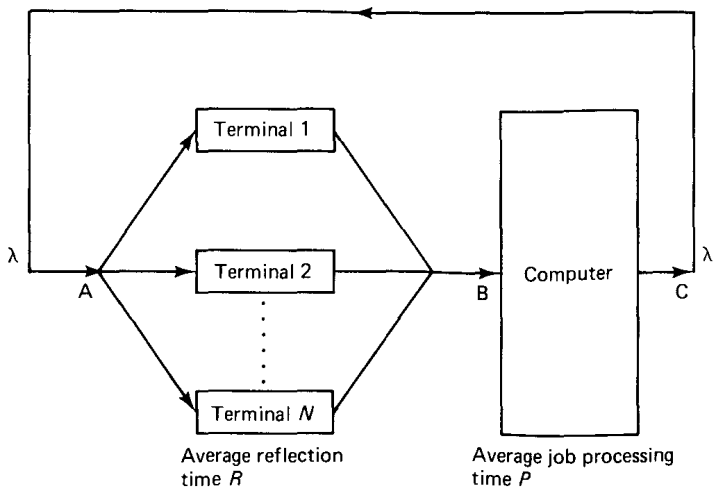


Figure 3.4 N terminals connected with a time-sharing computer system. To estimate maximum attainable throughput, we assume that a departing user immediately reenters the system or, equivalently, is immediately replaced by a new user.

Combining this relation with $\lambda = N/T$ [cf. Eq. (3.3)], we obtain

$$\frac{N}{R+NP} \leq \lambda \leq \frac{N}{R+P} \tag{3.6}$$

The throughput λ is also bounded above by the processing capacity of the computer. In particular, since the execution time of a job is P units on the average, it follows that the computer cannot process in the long run more than $1/P$ jobs per unit time, that is,

$$\lambda \leq \frac{1}{P} \tag{3.7}$$

(This conclusion can also be reached by applying Little's Theorem between the entry and exit points of the computer's CPU.)

By combining the preceding two relations, we obtain the bounds

$$\frac{N}{R+NP} \leq \lambda \leq \min\left\{\frac{1}{P}, \frac{N}{R+P}\right\} \tag{3.8}$$

for the throughput λ. By using $T = N/\lambda$, we also obtain bounds for the average user delay when the system is fully loaded:

$$\max\{NP,\ R+P\} \leq T \leq R+NP \tag{3.9}$$

These relations are illustrated in Fig. 3.5.

It can be seen that as the number of terminals N increases, the throughput approaches the maximum $1/P$, while the average user delay rises essentially in direct proportion with N. The number of terminals becomes a throughput bottleneck when $N < 1 + R/P$, in which case the computer resource stays idle for a substantial portion of the time while all users are engaged in reflection. In contrast, the limited processing power of the computer becomes the bottleneck when $N > 1 + R/P$. It is interesting to note that while the exact maximum attainable throughput depends on system parameters, such as the statistics of the reflection and processing times, and the manner in which jobs are served by the CPU, the

Samsung Exhibit 1015, Page 177 of 573

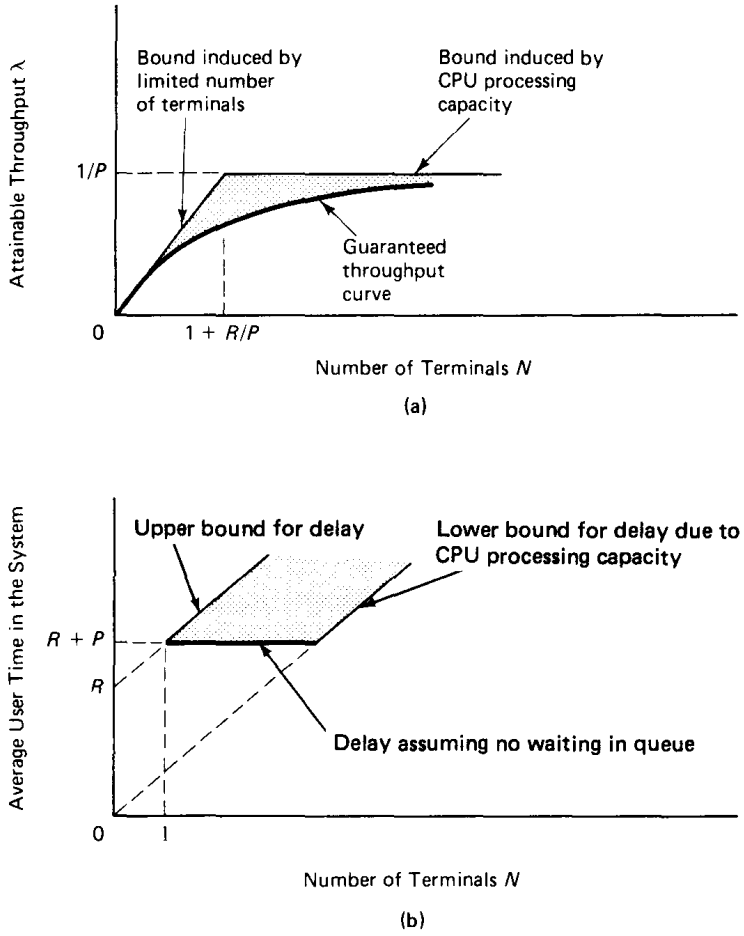


Figure 3.5 Bounds on throughput and average user delay in a time-sharing system. (a) Bounds on attainable throughput [Eq. (3.8)]. (b) Bounds on average user time in a fully loaded system [Eq. (3.9)]. The time increases essentially in proportion with the number of terminals N.

bounds obtained are independent of these parameters. We owe this convenient situation to the generality of Little's Theorem.

3.3 THE *M*/*M*/1 QUEUEING SYSTEM

The $M/M/1$ queueing system consists of a single queueing station with a single server (in a communication context, a single transmission line). Customers arrive according to a Poisson process with rate λ, and the probability distribution of the service time is exponential with mean $1/\mu$ sec. We will explain the meaning of these terms shortly. The name $M/M/1$ reflects standard queueing theory nomenclature whereby:

1. The first letter indicates the nature of the arrival process [*e.g.*, M stands for memoryless, which here means a Poisson process (*i.e.*, exponentially distributed inter-

arrival times), G stands for a general distribution of interarrival times, D stands for deterministic interarrival times].

2. The second letter indicates the nature of the probability distribution of the service times (*e.g.*, M, G, and D stand for exponential, general, and deterministic distributions, respectively). In all cases, successive interarrival times and service times are assumed to be statistically independent of each other.
3. The last number indicates the number of servers.

We have already established, via Little's Theorem, the relations

$$N = \lambda T, \qquad N_Q = \lambda W$$

between the basic quantities,

$N =$ Average number of customers in the system

$T =$ Average customer time in the system

$N_Q =$ Average number of customers waiting in queue

$W =$ Average customer waiting time in queue

However, N, T, N_Q, and W cannot be specified further unless we know something more about the statistics of the system. Given these statistics, we will be able to derive the steady-state probabilities

$p_n =$ Probability of n customers in the system, $n = 0, 1, \ldots$.

From these probabilities, we can get

$$N = \sum_{n=0}^{\infty} n p_n$$

and using Little's Theorem,

$$T = \frac{N}{\lambda}$$

Similar formulas exist for N_Q and W. Appendix B provides a summary of the results for the $M/M/1$ system and the other major systems analyzed later.

The analysis of the $M/M/1$ system as well as several other related systems, such as the $M/M/m$ or $M/M/\infty$ systems, is based on the theory of Markov chains summarized in Appendix A. An alternative approach is to use simple graphical arguments based on the concept of mean residual time introduced in Section 3.5. This approach does not require that the service times are exponentially distributed (*i.e.*, it applies to the $M/G/1$ system). The price paid for this generality is that the characterization of the steady-state probabilities is more complicated than for the $M/M/1$ system. The reader wishing to circumvent the Markov chain analysis may start directly with the $M/G/1$ system in Section 3.5 after a reading of the preliminary facts on the Poisson process given in Sections 3.3.1 and 3.3.2.

Samsung Exhibit 1015, Page 179 of 573

3.3.1 Main Results

We first introduce our assumptions on the arrival and service statistics of the $M/M/1$ system.

Arrival statistics—the Poisson process. In the $M/M/1$ system, customers arrive according to a Poisson process which we now define:

A stochastic process $\{A(t) \mid t \geq 0\}$ taking nonnegative integer values is said to be a *Poisson process* with rate λ if

1. $A(t)$ is a counting process that represents the total number of arrivals that have occurred from 0 to time t [*i.e.*, $A(0) = 0$], and for $s < t$, $A(t) - A(s)$ equals the numbers of arrivals in the interval $(s, t]$.
2. The numbers of arrivals that occur in disjoint time intervals are independent.
3. The number of arrivals in any interval of length τ is Poisson distributed with parameter $\lambda\tau$. That is, for all t, $\tau > 0$,

$$P\{A(t+\tau) - A(t) = n\} = e^{-\lambda\tau}\frac{(\lambda\tau)^n}{n!}, \qquad n = 0, 1, \ldots \tag{3.10}$$

The average number of arrivals within an interval of length τ is $\lambda\tau$ (based on the mean of the Poisson distribution). This leads to the interpretation of λ as an arrival rate (average number of arrivals per unit time).

We list some of the properties of the Poisson process that will be of interest:

1. Interarrival times are independent and exponentially distributed with parameter λ; that is, if t_n denotes the time of the n^{th} arrival, the intervals $\tau_n = t_{n+1} - t_n$ have the probability distribution

$$P\{\tau_n \leq s\} = 1 - e^{-\lambda s}, \qquad s \geq 0 \tag{3.11}$$

and are mutually independent. [The corresponding probability density function is $p(\tau_n) = \lambda e^{-\lambda\tau_n}$. The mean and variance of τ_n are $1/\lambda$ and $1/\lambda^2$, respectively.] For a proof of this property, see [Ros83], p. 35.

2. For every $t \geq 0$ and $\delta \geq 0$,

$$P\{A(t+\delta) - A(t) = 0\} = 1 - \lambda\delta + o(\delta) \tag{3.12}$$

$$P\{A(t+\delta) - A(t) = 1\} = \lambda\delta + o(\delta) \tag{3.13}$$

$$P\{A(t+\delta) - A(t) \geq 2\} = o(\delta) \tag{3.14}$$

where we generically denote by $o(\delta)$ a function of δ such that

$$\lim_{\delta \to 0} \frac{o(\delta)}{\delta} = 0$$

These equations can be verified by expanding the Poisson distribution on the number of arrivals in an interval of length δ [Eq. (3.10)] in a Taylor series [or equivalently, by writing $e^{-\lambda\delta} = 1 - \lambda\delta + (\lambda\delta)^2/2 - \cdots$].

3. If two or more independent Poisson processes $A_1, \ldots, A_k$ are merged into a single process $A = A_1 + A_2 + \cdots + A_k$, the latter process is Poisson with a rate equal to the sum of the rates of its components (see Problem 3.10).
4. If a Poisson process is split into two other processes by independently assigning each arrival to the first (second) of these processes with probability p ($1 - p$, respectively), the two arrival processes thus obtained are Poisson (see Problem 3.11). (For this it is essential that the assignment of each arrival be independent of the assignment of other arrivals. If, for example, the assignment is done by alternation, with even-numbered arrivals assigned to one process and odd-numbered arrivals assigned to the other, the two generated processes are not Poisson. This will prove to be significant in the context of data networks; see Example 3.17 in Section 3.6.)

A Poisson process is generally considered to be a good model for the aggregate traffic of a large number of similar and independent users. In particular, suppose that we merge n independent and identically distributed packet arrival processes. Each process has arrival rate λ/n, so that the aggregate process has arrival rate λ. The interarrival times τ between packets of the same process have a given distribution $F(s) = P\{\tau \leq s\}$ and are independent [$F(s)$ need not be an exponential distribution]. Then under relatively mild conditions on F [*e.g.*, $F(0) = 0$, $dF(0)/ds > 0$], the aggregate arrival process can be approximated well by a Poisson process with rate λ as $n \to \infty$ (see [KaT75], p. 221).

Service statistics. Our assumption regarding the service process is that *the customer service times have an exponential distribution with parameter* μ, that is, if s_n is the service time of the n^{th} customer,

$$P\{s_n \leq s\} = 1 - e^{-\mu s}, \qquad s \geq 0$$

[The probability density function of s_n is $p(s_n) = \mu e^{-\mu s_n}$, and its mean and variance are $1/\mu$ and $1/\mu^2$, respectively.] Furthermore, *the service times* s_n *are mutually independent and also independent of all interarrival times.* The parameter μ is called the *service rate* and represents the rate (in customers served per unit time) at which the server operates when busy. In the context of a packet transmission system, the independence of interarrival and service times implies, among other things, that the length of an arriving packet does not affect the arrival time of the next packet. It will be seen in Section 3.6 that this condition is often violated in practice, particularly when the arriving packets have just departed from another queue.

An important fact regarding the exponential distribution is its *memoryless* character, which can be expressed as

$$P\{\tau_n > r + t \mid \tau_n > t\} = P\{\tau_n > r\}, \qquad \text{for } r, t \geq 0$$

$$P\{s_n > r + t \mid s_n > t\} = P\{s_n > r\}, \qquad \text{for } r, t \geq 0$$

Samsung Exhibit 1015, Page 181 of 573

for the interarrival and service times τ_n and s_n, respectively. This means that the additional time needed to complete a customer's service in progress is independent of when the service started. Similarly, the time up to the next arrival is independent of when the previous arrival occurred. Verification of the memoryless property follows from the calculation

$$P\{\tau_n > r + t \mid \tau_n > t\} = \frac{P\{\tau_n > r + t\}}{P\{\tau_n > t\}} = \frac{e^{-\lambda(r+t)}}{e^{-\lambda t}} = e^{-\lambda r} = P\{\tau_n > r\}$$

Markov chain formulation. An important consequence of the memoryless property is that it allows the use of the theory of Markov chains. Indeed, this property, together with our earlier independence assumptions on interarrival and service times, imply that once we know the number $N(t)$ of customers in the system at time t, the times at which customers will arrive or complete service in the future are independent of the arrival times of the customers presently in the system and of how much service the customer currently in service (if any) has already received. This means that the future numbers of customers depend on past numbers only through the present number; that is, $\{N(t) \mid t \geq 0\}$ *is a continuous-time Markov chain.*

We could analyze the process $N(t)$ in terms of continuous-time Markov chain methodology; most of the queueing literature follows this line of analysis (see also Problem 3.12). It is sufficient, however, for our purposes in this section to use the simpler theory of discrete-time Markov chains (briefly summarized in Appendix A).

Let us focus attention at the times

$$0,\ \delta,\ 2\delta, \ldots, k\delta, \ldots$$

where δ is a small positive number. We denote

$$N_k = \text{ Number of customers in the system at time } k\delta$$

Since $N_k = N(k\delta)$ and, as discussed, $N(t)$ is a continuous-time Markov chain, we see that $\{N_k \mid k = 0, 1, \ldots\}$ is a discrete-time Markov chain with steady-state occupancy probabilities equal to those of the continuous chain. Let P_{ij} denote the corresponding transition probabilities

$$P_{ij} = P\{N_{k+1} = j \mid N_k = i\}$$

Note that P_{ij} depends on δ, but to keep notation simple, we do not show this dependence. By using Eqs. (3.12) through (3.14), one can show that

$$P_{oo} = 1 - \lambda\delta + o(\delta) \tag{3.15}$$

$$P_{ii} = 1 - \lambda\delta - \mu\delta + o(\delta), \qquad i \geq 1 \tag{3.16}$$

$$P_{i,i+1} = \lambda\delta + o(\delta), \qquad i \geq 0 \tag{3.17}$$

$$P_{i,i-1} = \mu\delta + o(\delta), \qquad i \geq 1 \tag{3.18}$$

$$P_{ij} = o(\delta). \qquad i \text{ and } j \neq i,\ i+1,\ i-1$$

Samsung Exhibit 1015, Page 182 of 573

To see how these equations are verified, note that when at a state $i \geq 1$, the probability of 0 arrivals and 0 departures in a δ-interval $I_k = (k\delta, (k+1)\delta]$ is $(e^{-\lambda\delta})(e^{-\mu\delta})$; this is because the number of arrivals and the number of departures are Poisson distributed and independent of each other. Expanding this in a power series in δ,

$$P\{0 \text{ customers arrive and } 0 \text{ depart in } I_k\} = 1 - \lambda\delta - \mu\delta + o(\delta) \tag{3.19}$$

The probability of 0 arrivals and 1 departure in the interval I_k is $e^{-\lambda\delta}(1 - e^{-\mu\delta})$ if $i = 1$ (since $1 - e^{-\mu\delta}$ is the probability that the customer in service will complete its service within I_k), and $e^{-\lambda\delta}(\mu\delta e^{-\mu\delta})$ if $i > 1$ (since $\mu\delta e^{-\mu\delta}$ is the probability that within the interval I_k, the customer in service will complete its service while the subsequent customer will not). In both cases we have

$$P\{0 \text{ customers arrive and } 1 \text{ departs in } I_k\} = \mu\delta + o(\delta)$$

Similarly, the probability of 1 arrival and 0 departures in I_k is $(\lambda\delta e^{-\lambda\delta})e^{-\mu\delta}$, so

$$P\{1 \text{ customer arrives and } 0 \text{ depart in } I_k\} = \lambda\delta + o(\delta)$$

These probabilities add up to 1 plus $o(\delta)$. Thus, the probability of more than one arrival or departure is negligible for δ small. It follows that for $i \geq 1$, P_{ii}, which is the probability of an equal number of arrivals and departures in I_k, is within $o(\delta)$ of the value in Eq. (3.19); this verifies Eq. (3.16). Equations (3.15), (3.17), and (3.18) are verified in the same way.

The state transition diagram for the Markov chain $\{N_k\}$ is shown in Fig. 3.6, where we have omitted the terms $o(\delta)$.

Derivation of the stationary distribution. Consider now the steady-state probabilities

$$p_n = \lim_{k\to\infty} P\{N_k = n\} = \lim_{t\to\infty} P\{N(t) = n\}$$

Note that during any time interval, the total number of transitions from state n to $n+1$ must differ from the total number of transitions from $n+1$ to n by at most 1. Thus asymptotically, the frequency of transitions from n to $n+1$ is equal to the frequency of transitions from $n+1$ to n. Equivalently, the probability that the system is in state n and makes a transition to $n+1$ in the next transition interval is the same as the probability that the system is in state $n+1$ and makes a transition to n, that is,

$$p_n\lambda\delta + o(\delta) = p_{n+1}\mu\delta + o(\delta)$$

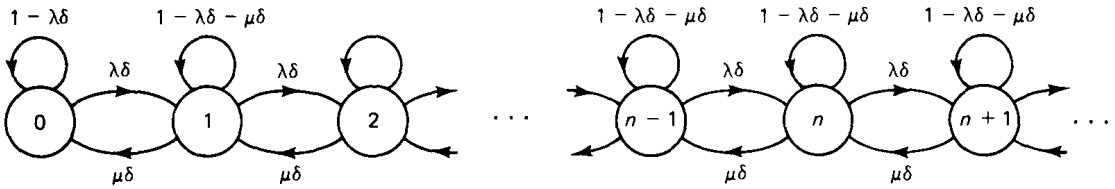


Figure 3.6 Discrete-time Markov chain for the $M/M/1$ system. The state n corresponds to n customers in the system. Transition probabilities shown are correct up to an $o(\delta)$ term.

Samsung Exhibit 1015, Page 183 of 573

By taking the limit in this equation as $\delta \to 0$, we obtain

$$p_n\lambda = p_{n+1}\mu \tag{3.20}$$

(The preceding equations are called *global balance equations*, corresponding to the set of states $\{0, 1, \ldots, n\}$ and $\{n+1, n+2, \ldots\}$. See Appendix A for a more general statement of these equations and for an interpretation that parallels the argument given above.) These equations can also be written as

$$p_{n+1} = \rho p_n, \qquad n = 0, 1, \ldots$$

where

$$\rho = \frac{\lambda}{\mu}$$

It follows that

$$p_{n+1} = \rho^{n+1} p_0, \quad n = 0, 1, \ldots \tag{3.21}$$

If $\rho < 1$ (service rate exceeds arrival rate), the probabilities p_n are all positive and add up to unity, so

$$1 = \sum_{n=0}^{\infty} p_n = \sum_{n=0}^{\infty} \rho^n p_0 = \frac{p_0}{1-\rho} \tag{3.22}$$

Combining the last two equations, we finally obtain

$$p_n = \rho^n(1-\rho), \qquad n = 0, 1, \ldots \tag{3.23}$$

We can now calculate the average number of customers in the system in steady-state:

$$N = \lim_{t\to\infty} E\{N(t)\} = \sum_{n=0}^{\infty} np_n = \sum_{n=0}^{\infty} n\rho^n(1-\rho)$$

$$= \rho(1-\rho)\sum_{n=0}^{\infty} n\rho^{n-1} = \rho(1-\rho)\frac{\partial}{\partial\rho}\left(\sum_{n=0}^{\infty}\rho^n\right)$$

$$= \rho(1-\rho)\frac{\partial}{\partial\rho}\left(\frac{1}{1-\rho}\right) = \rho(1-\rho)\frac{1}{(1-\rho)^2}$$

and finally, using $\rho = \lambda/\mu$, we have

$$N = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu-\lambda} \tag{3.24}$$

The graph of this equation is shown in Fig. 3.7. As ρ increases, so does N, and as $\rho \to 1$, we have $N \to \infty$. The graph is valid for $\rho < 1$. If $\rho > 1$, the server cannot keep up with the arrival rate and the queue length increases without bound. In the context of a packet transmission system, $\rho > 1$ means that $\lambda L > C$, where λ is the arrival rate in packets/sec, L is the average packet length in bits, and C is the transmission capacity in bits/sec.

Samsung Exhibit 1015, Page 184 of 573

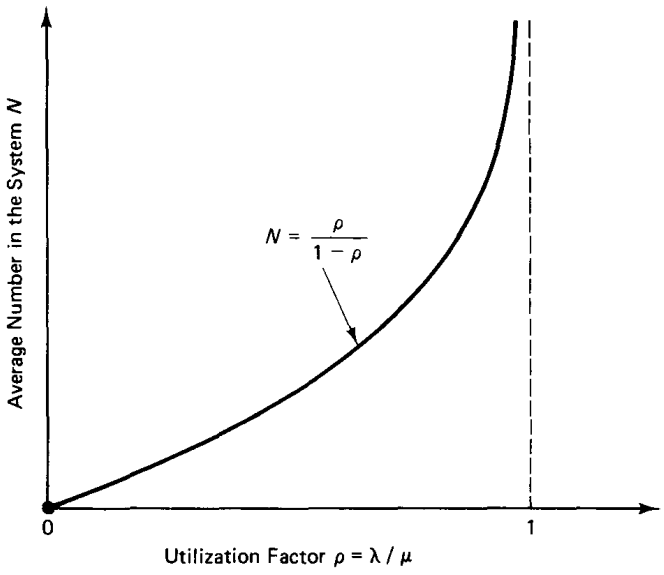


Figure 3.7 Average number in the system versus the utilization factor in the $M/M/1$ system. As $\rho \to 1$, $N \to \infty$.

The average delay per customer (waiting time in queue plus service time) is given by Little's Theorem,

$$T = \frac{N}{\lambda} = \frac{\rho}{\lambda(1-\rho)} \tag{3.25}$$

Using $\rho = \lambda/\mu$, this becomes

$$T = \frac{1}{\mu - \lambda} \tag{3.26}$$

We note that it is actually possible to show that the customer delay is exponentially distributed in steady-state [see Problem 3.11(b)].

The average waiting time in queue, W, is the average delay T less the average service time $1/\mu$, so

$$W = \frac{1}{\mu - \lambda} - \frac{1}{\mu} = \frac{\rho}{\mu - \lambda}$$

By Little's Theorem, the average number of customers in queue is

$$N_Q = \lambda W = \frac{\rho^2}{1-\rho}$$

A very useful interpretation is to view the quantity ρ as the *utilization factor* of the queueing system, (*i.e.*, the long-term proportion of time the server is busy). We showed this earlier in a broader context by using Little's Theorem (Example 3.1). Based on this interpretation, it follows that $\rho = 1 - p_0$, where p_0 is the probability of having no customers in the system, and we obtain an alternative verification of the formula derived for p_0 [Eq. (3.22)].

We illustrate these results by means of some examples from data networks.

Example 3.8 Increasing the Arrival and Transmission Rates by the Same Factor

Consider a packet transmission system whose arrival rate (in packets/sec) is increased from λ to $K\lambda$, where $K > 1$ is some scalar factor. The packet length distribution remains the same but the transmission capacity is increased by a factor of K, so the average packet transmission time is now $1/(K\mu)$ instead of $1/\mu$. It follows that the utilization factor ρ, and therefore the average number of packets in the system, remain the same:

$$N = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu - \lambda}$$

However, the average delay per packet is now $T = N/(K\lambda)$ and is therefore decreased by a factor of K. In other words, *a transmission line K times as fast will accommodate K times as many packets/sec at K times smaller average delay per packet.* This result is quite general, even applying to networks of queues. What is happening, as illustrated in Fig. 3.8, is that by increasing arrival rate and service rate by a factor K, the statistical characteristics of the queueing process are unaffected except for a change in time scale—the process is speeded up by a factor K. Thus, when a packet arrives, it will see ahead of it statistically the same number of packets as with a slower transmission line. However, the packets ahead of it will be moving K times faster.

Example 3.9 Statistical Multiplexing Compared with Time- and Frequency-Division Multiplexing

Assume that m statistically identical and independent Poisson packet streams each with an arrival rate of λ/m packets/sec are transmitted over a communication line. The packet lengths for all streams are independent and exponentially distributed. The average transmission time is $1/\mu$. If the streams are merged into a single Poisson stream, with rate λ, as in statistical multiplexing, the average delay per packet is

$$T = \frac{1}{\mu - \lambda}$$

If, instead, the transmission capacity is divided into m equal portions, one per packet stream as in time- and frequency-division multiplexing, each portion behaves like an $M/M/1$ queue with arrival rate λ/m and average service rate μ/m. Therefore, the average delay per packet is

$$T = \frac{m}{\mu - \lambda}$$

that is, m times larger than for statistical multiplexing.

The preceding argument indicates that multiplexing a large number of traffic streams on separate channels in a transmission line performs very poorly in terms of delay. The performance is even poorer if the capacity of the channels is not allocated in direct proportion to the arrival rates of the corresponding streams—something that cannot be done (at least in the scheme considered here) if these arrival rates change over time. This is precisely why data networks, which most of the time serve many low duty cycle traffic streams, are typically organized on the basis of some form of statistical multiplexing. An argument in favor of time- and frequency-division multiplexing arises when each traffic stream is "regular" (as opposed to Poisson) in the sense that no packet arrives while another is transmitted, and thus there is no waiting in queue if that stream is transmitted on a dedicated transmission line. If several streams of this type are statistically multiplexed on a single transmission line, the

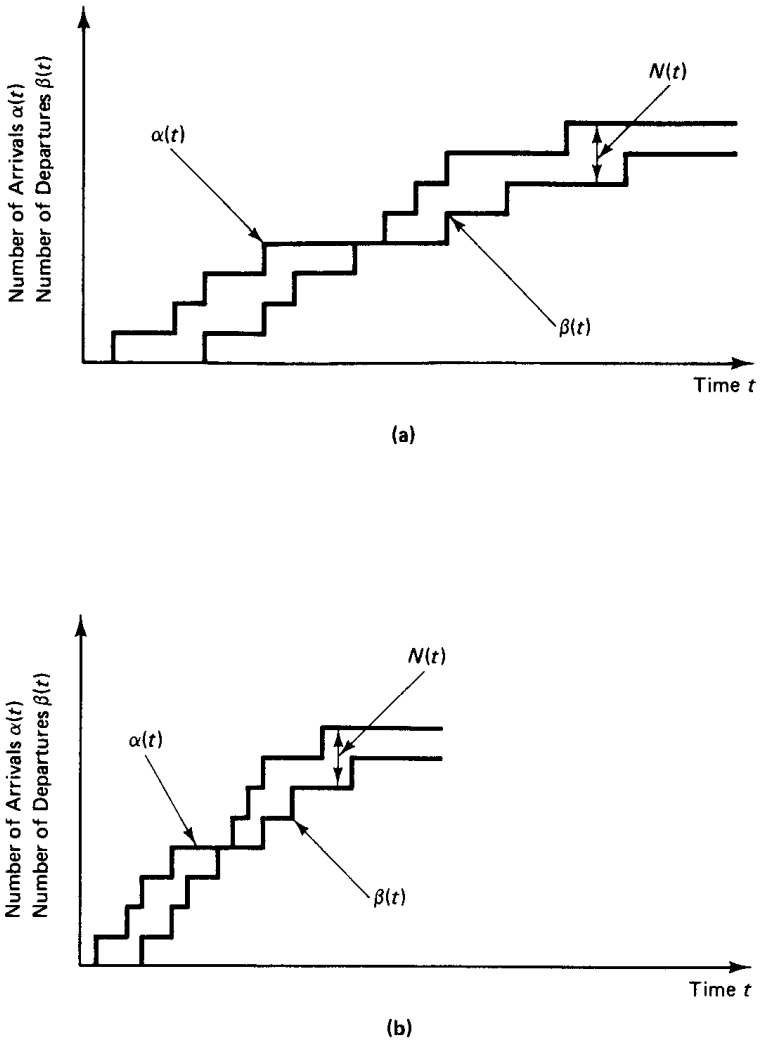


Figure 3.8 Increasing the arrival rate and the service rate by the same factor (see Example 3.8). (a) Sample paths of number of arrivals $\alpha(t)$ and departures $\beta(t)$ in the original system. (b) Corresponding sample paths of number of arrivals $\alpha(t)$ and departures $\beta(t)$ in the "speeded up" system, where the arrival rate and the service rate have been increased by a factor of 2. The average number in the system is the same as before, but the average delay is reduced by a factor of 2 since customers are moving twice as fast.

average delay per packet will decrease, but the average waiting time in queue will become positive and the variance of delay will also become positive. Thus in telephony, where each traffic stream is a voice conversation that is regular in the sense above and small variability of delay is critical, time- and frequency-division multiplexing are still used widely.

3.3.2 Occupancy Distribution upon Arrival

In our subsequent development, there are several situations where we will need a probabilistic characterization of a queueing system as seen by an arriving customer. It is

possible that the times of customer arrivals are in some sense nontypical, so that the steady-state occupancy probabilities upon arrival,

$$a_n = \lim_{t\to\infty} P\{N(t) = n \mid \text{an arrival occurred just after time } t\} \tag{3.27}$$

need not be equal to the corresponding unconditional steady-state probabilities,

$$p_n = \lim_{t\to\infty} P\{N(t) = n\} \tag{3.28}$$

It turns out, however, that for the $M/M/1$ system, we have

$$p_n = a_n, \qquad n = 0, 1, \ldots \tag{3.29}$$

so that an arriving customer finds the system in a "typical" state. Indeed, *this holds under very general conditions for queueing systems with Poisson arrivals regardless of the distribution of the service times.* The only additional requirement we need is that future arrivals are independent of the current number in the system. More precisely, *we assume that for every time t and increment $\delta > 0$, the number of arrivals in the interval $(t, t+\delta)$ is independent of the number in the system at time t.* Given the Poisson hypothesis, essentially this amounts to assuming that, at any time, the service times of previously arrived customers and the future interarrival times are independent—something that is reasonable for packet transmission systems. In particular, the assumption holds if the arrival process is Poisson and interarrival times and service times are independent.

For a formal proof of the equality $a_n = p_n$ under the preceding assumption, let $A(t, t+\delta)$ be the event that an arrival occurs in the interval $(t, t+\delta)$. Let

$$p_n(t) = P\{N(t) = n\} \tag{3.30}$$

$$a_n(t) = P\{N(t) = n \mid \text{an arrival occurred just after time } t\} \tag{3.31}$$

We have, using Bayes' rule,

$$\begin{aligned} a_n(t) &= \lim_{\delta\to 0} P\{N(t) = n \mid A(t, t+\delta)\} \\ &= \lim_{\delta\to 0} \frac{P\{N(t) = n,\ A(t, t+\delta)\}}{P\{A(t, t+\delta)\}} \\ &= \lim_{\delta\to 0} \frac{P\{A(t, t+\delta) \mid N(t) = n\}\ P\{N(t) = n\}}{P\{A(t, t+\delta)\}} \end{aligned} \tag{3.32}$$

By assumption, the event $A(t, t+\delta)$ is independent of the number in the system at time t. Therefore,

$$P\{A(t, t+\delta) \mid N(t) = n\} = P\{A(t, t+\delta)\}$$

and we obtain from Eq. (3.32)

$$a_n(t) = P\{N(t) = n\} = p_n(t)$$

Taking the limit as $t \to \infty$, we obtain $a_n = p_n$.

As an example of what can happen if the arrival process is not Poisson, suppose that interarrival times are independent and uniformly distributed between 2 and 4 sec,

while customer service times are all equal to 1 sec. Then an arriving customer always finds an empty system. On the other hand, the average number in the system as seen by an outside observer looking at a system at a random time is $1/3$. (The time in the system of each customer is 1 sec, so by Little's Theorem, N is equal to the arrival rate λ, which is $1/3$ since the expected time between arrivals is 3.)

For a similar example where the arrival process is Poisson but the service times of customers in the system and the future arrival times are correlated, consider a packet transmission system where packets arrive according to a Poisson process. The transmission time of the n^{th} packet equals one half the interarrival time between packets n and $n+1$. Upon arrival, a packet finds the system empty. However, the average number in the system, as seen by an outside observer, is easily seen to be $1/2$.

3.3.3 Occupancy Distribution upon Departure

Let us consider the distribution of the number of customers in the system just after a departure has occurred, that is, the probabilities

$$d_n(t) = P\{N(t) = n \mid \text{a departure occurred just before time } t\}$$

The corresponding steady-state values are denoted

$$d_n = \lim_{t\to\infty} d_n(t), \qquad n = 0, 1, \ldots$$

It turns out that

$$d_n = a_n, \qquad n = 0, 1, \ldots$$

under very general assumptions—the only requirement essentially is that the system reaches a steady-state with all n having positive steady-state probabilities, and that $N(t)$ changes in unit increments. [These assumptions certainly hold for a stable $M/M/1$ system ($\rho < 1$), but they also hold for most stable single-queue systems of interest.] For any sample path of the system and for every n, the number in the system will be n infinitely often (with probability 1). This means that for each time the number in the system increases from n to $n+1$ due to an arrival, there will be a corresponding future decrease from $n+1$ to n due to a departure. Therefore, in the long run, the frequency of transitions from n to $n+1$ out of transitions from any k to $k+1$ equals the frequency of transitions from $n+1$ to n out of transitions from any $k+1$ to k, which implies that $d_n = a_n$. Therefore, *in steady-state, the system appears statistically identical to an arriving and a departing customer. When arrivals are Poisson*, we saw earlier that $a_n = p_n$; so, in this case, *both an arriving and a departing customer in steady-state see a system that is statistically identical to the one seen by an observer looking at the system at an arbitrary time.*

3.4 THE $M/M/m$, $M/M/\infty$, $M/M/m/m$, AND OTHER MARKOV SYSTEMS

We consider now a number of queueing systems that are similar to $M/M/1$ in that the arrival process is Poisson and the service times are independent, exponentially dis-

tributed, and independent of the interarrival times. Because of these assumptions, these systems can be modeled with continuous- or discrete-time Markov chains. From the corresponding state transition diagram, we can derive a set of equations that can be solved for the steady-state occupancy probabilities. Application of Little's Theorem then yields the average delay per customer.

3.4.1 $M/M/m$: The m-Server Case

The $M/M/m$ queueing system is identical to the $M/M/1$ system except that there are m servers (or channels of a transmission line in a data communication context). A customer at the head of the queue is routed to any server that is available. The corresponding state transition diagram is shown in Fig. 3.9.

By writing down the global balance equations for the steady-state probabilities p_n and taking $\delta \to 0$, we obtain

$$\begin{aligned} \lambda p_{n-1} &= n\mu p_n, \qquad n \le m \\ \lambda p_{n-1} &= m\mu p_n, \qquad n > m \end{aligned} \tag{3.33}$$

From these equations we obtain

$$p_n = \begin{cases} p_0 \dfrac{(m\rho)^n}{n!}, & n \le m \\ p_0 \dfrac{m^m \rho^n}{m!}, & n > m \end{cases} \tag{3.34}$$

where ρ is given by

$$\rho = \frac{\lambda}{m\mu} < 1$$

We can calculate p_0 using Eq. (3.34) and the condition $\sum_{n=0}^{\infty} p_n = 1$. We obtain

$$p_0 = \left[1 + \sum_{n=1}^{m-1} \frac{(m\rho)^n}{n!} + \sum_{n=m}^{\infty} \frac{(m\rho)^n}{m!} \frac{1}{m^{n-m}} \right]^{-1}$$

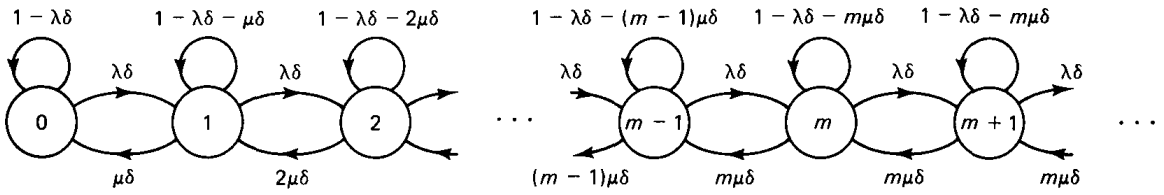


Figure 3.9 Discrete-time Markov chain for the $M/M/m$ system.

and finally,

$$p_0 = \left[\sum_{n=0}^{m-1} \frac{(m\rho)^n}{n!} + \frac{(m\rho)^m}{m!(1-\rho)}\right]^{-1} \tag{3.35}$$

The probability that an arrival will find all servers busy and will be forced to wait in queue is an important measure of performance of the $M/M/m$ system. Since an arriving customer finds the system in "typical" state (see Section 3.3.2), we have

$$P\{\text{Queueing}\} = \sum_{n=m}^{\infty} p_n = \sum_{n=m}^{\infty} \frac{p_0 m^m \rho^n}{m!} = \frac{p_0 (m\rho)^m}{m!} \sum_{n=m}^{\infty} \rho^{n-m}$$

and, finally,

$$P_Q \triangleq P\{\text{Queueing}\} = \frac{p_0 (m\rho)^m}{m!(1-\rho)} \tag{3.36}$$

where p_0 is given by Eq. (3.35). This is known as the *Erlang C formula*, honoring Denmark's A. K. Erlang, the foremost pioneer of queueing theory. This equation is often used in telephony (and more generally in circuit switching systems) to estimate the probability of a call request finding all of the m circuits of a transmission line busy. In an $M/M/m$ model it is assumed that such a call request "remains in queue," that is, continuously attempts to find a free circuit. The alternative model where such a call departs from the system and never returns is discussed in the context of the $M/M/m/m$ system in Section 3.4.3.

The expected number of customers waiting in queue (not in service) is given by

$$N_Q = \sum_{n=0}^{\infty} n p_{m+n}$$

Using the expression for p_{m+n} [Eq. (3.34)], we obtain

$$N_Q = \sum_{n=0}^{\infty} n p_0 \frac{m^m \rho^{m+n}}{m!} = \frac{p_0 (m\rho)^m}{m!} \sum_{n=0}^{\infty} n\rho^n$$

Using the Erlang C formula of Eq. (3.36) to express p_0 in terms of P_Q, and the equation $(1-\rho)\sum_{n=0}^{\infty} n\rho^n = \rho/(1-\rho)$ encountered in the $M/M/1$ system analysis, we finally obtain

$$N_Q = P_Q \frac{\rho}{1-\rho} \tag{3.37}$$

Note that

$$\frac{N_Q}{P_Q} = \frac{\rho}{1-\rho}$$

represents the expected number found in queue by an arriving customer conditioned on the fact that he is forced to wait in queue, and is independent of the number of servers for a given $\rho = \lambda/m\mu$. This suggests in particular that as long as there are customers waiting in queue, the queue size of the $M/M/m$ system behaves identically as in an $M/M/1$

system with service rate $m\mu$—the aggregate rate of the m servers. Some thought shows that indeed this is true in view of the memoryless property of the exponential distribution.

Using Little's Theorem and the expression (3.37) for N_Q, we obtain the average time W a customer has to wait in queue:

$$W = \frac{N_Q}{\lambda} = \frac{\rho P_Q}{\lambda(1-\rho)}$$

The average delay per customer is, therefore,

$$T = \frac{1}{\mu} + W = \frac{1}{\mu} + \frac{\rho P_Q}{\lambda(1-\rho)}$$

and using $\rho = \lambda/m\mu$, we obtain

$$T = \frac{1}{\mu} + W = \frac{1}{\mu} + \frac{P_Q}{m\mu - \lambda} \tag{3.38}$$

Using Little's Theorem again, the average number of customers in the system is

$$N = \lambda T = \frac{\lambda}{\mu} + \frac{\lambda P_Q}{m\mu - \lambda}$$

and using $\rho = \lambda/m\mu$, we finally obtain

$$N = m\rho + \frac{\rho P_Q}{1-\rho}$$

Example 3.10 Using One vs. Using Multiple Channels in Statistical Multiplexing

Consider a communication link serving m independent Poisson traffic streams with overall rate λ. Suppose that the link is divided into m separate channels with one channel assigned to each traffic stream. However, if a traffic stream has no packet awaiting transmission, its corresponding channel is used to transmit a packet of another traffic stream. The transmission times of packets on each of the channels are exponentially distributed with mean $1/\mu$. The system can be modeled by the same Markov chain as the $M/M/m$ queue. Let us compare the average delays per packet of this system, and an $M/M/1$ system with the same arrival rate λ and service rate $m\mu$ (statistical multiplexing with one channel having m times larger capacity). In the former case, the average delay per packet is given by the $M/M/m$ average delay expression (3.38)

$$T = \frac{1}{\mu} + \frac{P_Q}{m\mu - \lambda}$$

while in the latter case, the average delay per packet is

$$\hat{T} = \frac{1}{m\mu} + \frac{\hat{P}_Q}{m\mu - \lambda}$$

where P_Q and $\hat{P}_Q$ denote the queueing probability in each case. When $\rho \ll 1$ (lightly loaded system) we have $P_Q \cong 0$, $\hat{P}_Q \cong 0$, and

$$\frac{T}{\hat{T}} \cong m$$

When ρ is only slightly less than 1, we have $P_Q \cong 1$, $\hat{P}_Q \cong 1$, $1/\mu \ll 1/(m\mu - \lambda)$, and

$$\frac{T}{\hat{T}} \cong 1$$

Therefore, for a light load, statistical multiplexing with m channels produces a delay almost m times larger than the delay of statistical multiplexing with the m channels combined in one (about the same as time- and frequency-division multiplexing). For a heavy load, the ratio of the two delays is close to 1.

3.4.2 $M/M/\infty$: The Infinite-Server Case

In the limiting case where $m = \infty$ in the $M/M/m$ system, we obtain from the global balance equations (3.33)

$$\lambda p_{n-1} = n\mu p_n\,, \qquad n = 1, 2, \ldots$$

so

$$p_n = p_0 \left(\frac{\lambda}{\mu}\right)^n \frac{1}{n!}\,, \qquad n = 1, 2, \ldots$$

From the condition $\sum_{n=0}^{\infty} p_n = 1$, we obtain

$$p_0 = \left[1 + \sum_{n=1}^{\infty} \left(\frac{\lambda}{\mu}\right)^n \frac{1}{n!}\right]^{-1} = e^{-\lambda/\mu}$$

so finally,

$$p_n = \left(\frac{\lambda}{\mu}\right)^n \frac{e^{-\lambda/\mu}}{n!}\,, \qquad n = 0, 1, \ldots$$

Therefore, in steady-state, *the number in the system is Poisson distributed with parameter* λ/μ. The average number in the system is

$$N = \frac{\lambda}{\mu}$$

By Little's Theorem, the average delay is N/λ or

$$T = \frac{1}{\mu}$$

This last equation can also be obtained simply by arguing that in an $M/M/\infty$ system, there is no waiting in queue, so T equals the average service time $1/\mu$. It can be shown that the number in the system is Poisson distributed even if the service time distribution is not exponential (*i.e.*, in the $M/G/\infty$ system; see Problem 3.47).

Example 3.11 The Quasistatic Assumption

It is often convenient to assume that the external packet traffic entering a subnet node and destined for some other subnet node can be modeled by a stationary stochastic process that has a constant bit arrival rate (average bits/sec). This approximates a situation where the

arrival rate changes slowly with time and constitutes what we refer to as the quasistatic assumption.

When there are only a few active sessions (*i.e.*, user pairs) for the given origin–destination pair, this assumption is seriously violated since the addition or termination of a single session can change the total bit arrival rate by a substantial factor. When, however, there are many active sessions, each with a bit arrival rate that is small relative to the total, it seems plausible that the quasistatic assumption is approximately valid. The reason is that session additions are statistically counterbalanced by session terminations, with variations in the total rate being relatively small. For an analytical substantiation, assume that sessions are generated according to a Poisson process with rate λ, and terminate after a time which is exponentially distributed with mean $1/\mu$. Then the number of active sessions n evolves like the number of customers in an $M/M/\infty$ system (*i.e.*, is Poisson distributed with parameter λ/μ in steady-state). In particular, the mean and standard deviation of n are

$$N = E\{n\} = \frac{\lambda}{\mu}$$

$$\sigma_n = \left[E\left\{(n-N)^2\right\}\right]^{1/2} = \left(\frac{\lambda}{\mu}\right)^{1/2}$$

Suppose the i^{th} active session generates traffic according to a stationary stochastic process having a bit arrival rate γ_i bits/sec. Assume that the rates γ_i are independent random variables with common mean $E\{\gamma_i\} = \Gamma$ and second moment $s_\gamma^2 = E\{\gamma_i^2\}$. Then the total bit arrival rate for n active sessions is the random variable $f = \sum_{i=1}^{n} \gamma_i$, which has mean

$$F = E\{f\} = \frac{\lambda}{\mu}\Gamma$$

The standard deviation of f, denoted σ_f, can be obtained by writing

$$\sigma_f^2 = E\left\{\left(\sum_{i=1}^{n}\gamma_i\right)^2\right\} - F^2$$

and carrying out the corresponding calculations (Problem 3.28). The result is

$$\sigma_f = \left(\frac{\lambda}{\mu}\right)^{1/2} s_\gamma$$

Therefore, we have

$$\frac{\sigma_f}{F} = \left(\frac{s_\gamma}{\Gamma}\right)\left(\frac{\mu}{\lambda}\right)^{1/2}$$

Suppose now that the average bit rate Γ of a session is small relative to the total F; that is, a "many-small-sessions assumption" holds. Then, since $\Gamma/F = \mu/\lambda$, we have that μ/λ is small. If we reasonably assume that s_γ/Γ has a moderate value, it follows from the equation above that σ_f/F is small. Therefore, the total arrival rate f is approximately constant, thereby justifying the quasistatic assumption.

3.4.3 *M/M/m/m*: The *m*-Server Loss System

Consider a system which is identical to the $M/M/m$ system except that if an arrival finds all m servers busy, it does not enter the system and is lost instead; the last m in the

$M/M/m/m$ notation indicates the limit on the number of customers in the system. This model is in wide use in telephony (and also, more generally, in circuit switched networks). In this context, customers in the system correspond to active telephone conversations and the m servers represent a single transmission line consisting of m circuits. The average service time $1/\mu$ is the average duration of a telephone conversation. The principal quantity of interest here is the *blocking probability*, that is, the steady-state probability that all circuits are busy, in which case an arriving call is refused service. Note that in an $M/M/m/m$-based model, the assumption is that blocked calls are lost (not reattempted). This is in contrast with an $M/M/m$-based model, where the assumption is that blocked calls continuously reattempt admission into service. In data networks, the $M/M/m/m$ system can be used as a model where arrivals correspond to requests for virtual circuit connections between two nodes and the maximum number of virtual circuits allowed is m.

The corresponding state transition diagram is shown in Fig. 3.10. We have

$$\lambda p_{n-1} = n\mu p_n\,, \qquad n = 1, 2, \ldots, m$$

so

$$p_n = p_0 \left(\frac{\lambda}{\mu}\right)^n \frac{1}{n!}\,, \qquad n = 1, 2, \ldots, m$$

Solving for p_0 in the equation $\sum_{n=0}^{m} p_n = 1$, we obtain

$$p_0 = \left[\sum_{n=0}^{m} \left(\frac{\lambda}{\mu}\right)^n \frac{1}{n!}\right]^{-1}$$

The probability that an arrival will find all m servers busy and will therefore be lost is

$$p_m = \frac{(\lambda/\mu)^m/m!}{\sum_{n=0}^{m}(\lambda/\mu)^n/n!}$$

This equation is known as the *Erlang B formula* and finds wide use in evaluating the blocking probability of telephone systems. It can be shown to hold even if the service time has mean $1/\mu$ but arbitrary probability distribution (*i.e.*, for an $M/G/m/m$ system; see [Ros83], p. 170).

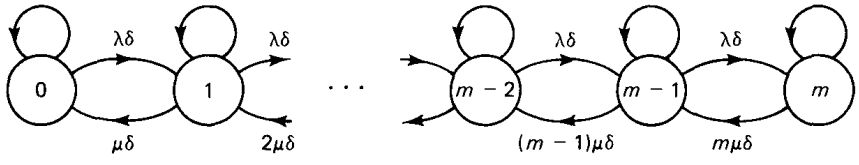


Figure 3.10 Discrete-time Markov chain for the $M/M/m/m$ system.

Samsung Exhibit 1015, Page 195 of 573

3.4.4 Multidimensional Markov Chains—Applications in Circuit Switching

We have considered so far queueing systems with a single type of customer where the state can be described by the number of customers in the system. In some important systems there are several classes of customers, each with its own statistical characteristics for arrival and service, which cannot be lumped into a single class for the purpose of analysis. Here are some examples:

Example 3.12 Two Session Classes in a Circuit Switching System

Consider a transmission line consisting of m independent circuits of equal capacity. There are two types of sessions arriving with Poisson rates λ_1 and λ_2, respectively. A session is blocked and lost for the system if all circuits are busy upon arrival, and is otherwise routed to any free circuit. The durations (or holding times) of the sessions of the two types are exponentially distributed with means $1/\mu_1$ and $1/\mu_2$. We are interested in finding the steady-state blocking probability for this system.

We first note that if $\mu_1 = \mu_2$, the two session types are indistinguishable for queueing purposes and the system can be modeled by an $M/M/m/m$ queue with arrival rate $\lambda_1 + \lambda_2$ and state equal to the total number of busy circuits. The desired blocking probability p_m is then given by the Erlang B formula of the preceding subsection. If, however, $\mu_1 \neq \mu_2$, then the total number of busy circuits does not fully specify the future statistical behavior of the queue; the number of each session type is also important since the duration of a session depends statistically on its type. Thus, the appropriate Markov chain model involves the two-dimensional state (n_1, n_2), where n_i is the number of circuits occupied by a session of type i, for $i = 1, 2$. The transition probability diagram for this chain is shown in Fig. 3.11. Generally, for multidimensional chains one may write the global balance equations for the stationary distribution

$$P(n_1, n_2), \qquad n_1 \geq 0,\ n_2 \geq 0,\ n_1 + n_2 \leq m$$

and try to solve them numerically. For this example, however, a closed-form expression is possible. We will demonstrate this shortly, once we develop the appropriate methodology.

Example 3.13 Two-Class System with Preferential Treatment for One Class

Consider the system of the preceding example with the difference that there is a limit $k < m$ on the number of circuits that can be used by sessions of the second type, so there are always $m - k$ circuits for use by sessions of the first type. The corresponding two-dimensional Markov chain is shown in Fig. 3.12. Note that here we should distinguish between the blocking probability for the first type of session, which is

$$\sum_{\{(n_1,n_2)|m-k\leq n_1\leq m, n_2=m-n_1\}} P(n_1, n_2)$$

and the blocking probability for the second type of session, which is

$$\sum_{\{(n_1,n_2)|0\leq n_1\leq m, n_2=\min\{k,m-n_1\}\}} P(n_1, n_2)$$

Again, it turns out that there is a closed-form expression for $P(n_1, n_2)$, as will be seen shortly.

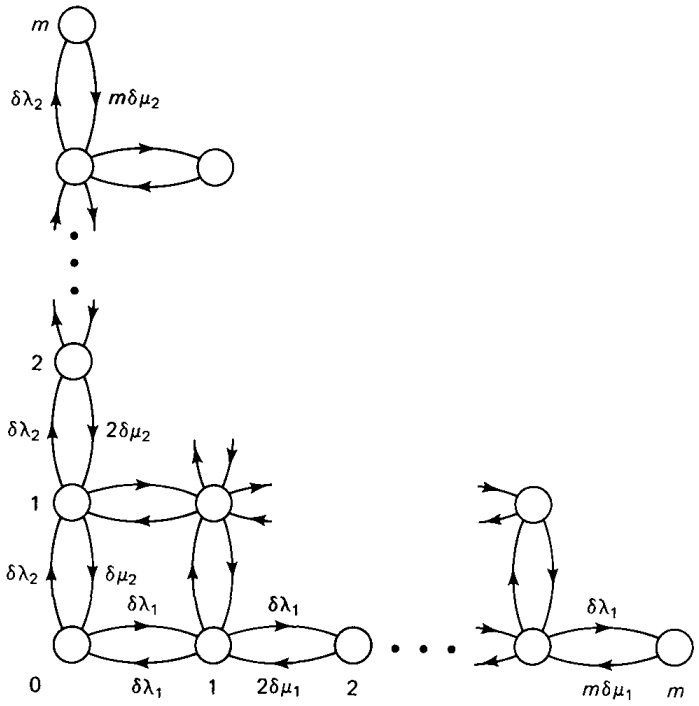


Figure 3.11 Markov chain for the two-class queue of Example 3.12. To simplify the diagram, we do not show self-transitions and $o(\delta)$ transitions.

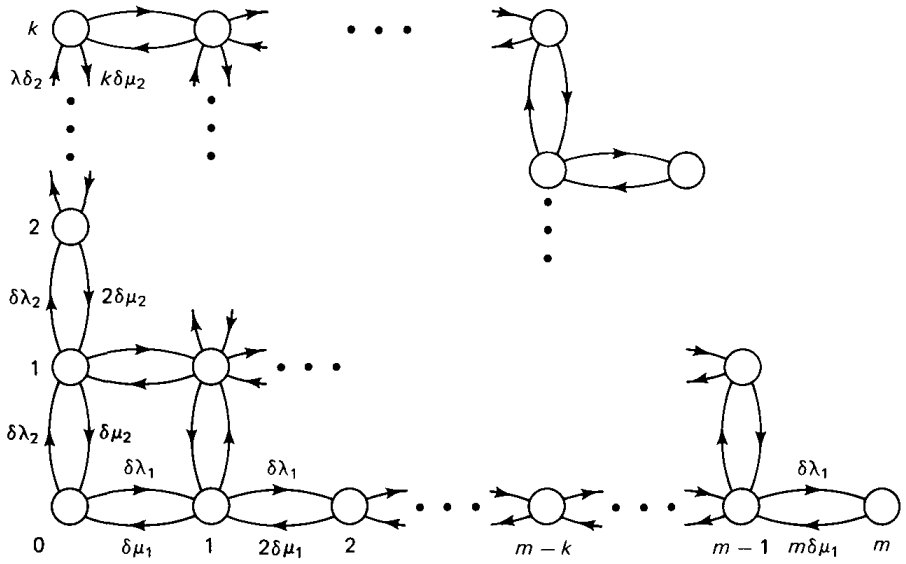


Figure 3.12 Markov chain for the two class queue with preferential treatment for one class (cf. Example 3.13). Self-transitions and $o(\delta)$ transitions are not shown.

Multidimensional Markov chains usually involve K customer types. Their states are of the form $(n_1, n_2, \ldots, n_K)$, where n_i is the number of customers of type i in the system. Such chains are usually harder to analyze than their one-dimensional counterparts, but in many interesting special cases one can obtain a closed-form solution for the

Samsung Exhibit 1015, Page 197 of 573

stationary distribution $P(n_1, n_2, \ldots, n_K)$. Important examples of properties that make this possible are:

1. The *detailed balance equations*

$$\lambda_i P(n_1, \ldots, n_{i-1}, n_i, n_{i+1}, \ldots, n_K) = \mu_i P(n_1, \ldots, n_{i-1}, n_i + 1, n_{i+1}, \ldots, n_K)$$

hold for all pairs of adjacent states

$$(n_1, \ldots, n_{i-1}, n_i, n_{i+1}, \ldots, n_K) \quad \text{and} \quad (n_1, \ldots, n_{i-1}, n_i + 1, n_{i+1}, \ldots, n_K)$$

where λ_i and μ_i are the arrival rate and service rate, respectively, of the customers of type i. These equations imply that the frequency of transitions between any two adjacent states is the same in both directions (see Appendix A). We will explain in Section 3.7 that chains for which these equations hold are statistically indistinguishable when looked in forward and in reverse time, and for this reason they will be called *reversible*. Note that these equations hold for all the single-customer class systems discussed so far.

2. The stationary distribution can be expressed in *product form*, that is,

$$P(n_1, n_2, \ldots, n_K) = P_1(n_1) P_2(n_2) \cdots P_K(n_K)$$

where for each i, $P_i(n_i)$ is an expression depending only on the number n_i of customers of type i. Several important types of networks of queues admit product form solutions, as will be seen in Section 3.8.

In this section we restrict ourselves to a class of multidimensional Markov chains, constructed from single-customer class systems using a process called *truncation*, for which we will see that both of the properties above hold.

Truncation of independent single-class systems. For a trivial example of a multidimensional Markov chain that admits a product form solution, consider K independent $M/M/1$ queues. The number of customers in the i^{th} queue has distribution

$$P_i(n_i) = \rho_i^{n_i}(1 - \rho_i)$$

where

$$\rho_i = \frac{\lambda_i}{\mu_i}$$

λ_i and μ_i are the corresponding arrival and service rates, respectively, and we assume that $\rho_i < 1$ for all i. Since the K queues are independent, we have the product form

$$P(n_1, n_2, \ldots, n_K) = P_1(n_1) P_2(n_2) \cdots P_K(n_K)$$

Note that the reasoning above would also apply if each of the $M/M/1$ queues were replaced by a birth–death type of queue (two successive states can differ only by a single unit: for example, the $M/M/m$, $M/M/\infty$, and $M/M/m/m$ queues). The only requirement is that the queues are independent.

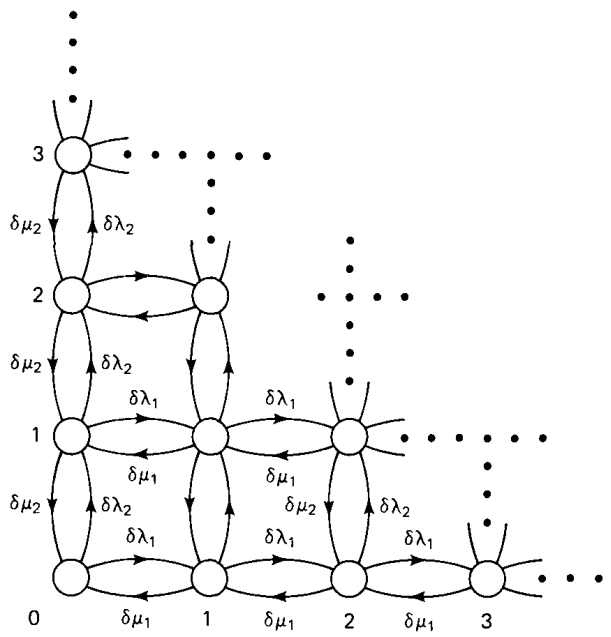


Figure 3.13 Markov chain for a K independent $M/M/1$ queues system. Self-transitions and $o(\delta)$ transitions are not shown ($K = 2$ in the figure).

Consider now the transition probability diagram of the K independent $M/M/1$ queues system, shown in Fig. 3.13. A *truncation* of this system is a Markov chain having the same transition probability diagram with the only difference that some states have been eliminated, while transitions between all other pairs of states, together with their corresponding transition probabilities, have been left unchanged except for $O(\delta)$ terms (see Fig. 3.14). We require that the truncation is an irreducible Markov chain, that is, all states communicate with each other (see Appendix A).

We claim that the stationary distribution of this truncated system has the product form

$$P(n_1, n_2, \ldots, n_K) = \frac{\rho_1^{n_1}\rho_2^{n_2}\cdots\rho_K^{n_K}}{G} \tag{3.39}$$

where G is a normalization constant guaranteeing that $P(n_1, n_2, \ldots, n_K)$ is a probability distribution, that is,

$$G = \sum_{(n_1, n_2, \ldots, n_K)\in S} \rho_1^{n_1}\rho_2^{n_2}\cdots\rho_K^{n_K} \tag{3.40}$$

where S is the set of states of the truncated system.

To show this, we consider the detailed balance equations

$$\lambda_i P(n_1, \ldots, n_{i-1}, n_i, n_{i+1}, \ldots, n_K) = \mu_i P(n_1, \ldots, n_{i-1}, n_i + 1, n_{i+1}, \ldots, n_K)$$

By substituting the probabilities (3.39) in these equations, we obtain

$$\lambda_i \frac{\rho_1^{n_1}\cdots\rho_{i-1}^{n_{i-1}}\rho_i^{n_i}\rho_{i+1}^{n_{i+1}}\cdots\rho_K^{n_K}}{G} = \mu_i \frac{\rho_1^{n_1}\cdots\rho_{i-1}^{n_{i-1}}\rho_i^{n_i+1}\rho_{i+1}^{n_{i+1}}\cdots\rho_K^{n_K}}{G}$$

which holds as an identity in view of the definition $\rho_i = \lambda_i/\mu_i$. Therefore, the probability

Samsung Exhibit 1015, Page 199 of 573

distribution given by the expression (3.39) satisfies the detailed balance equations for the truncated chain, so it must be its unique stationary distribution (see Appendix A).

It should be noted here that there is a generic difficulty with product form solutions. To obtain the stationary distribution, one needs to compute the normalization constant G of Eq. (3.40). For some systems, this involves a large amount of computation. An alternative to computing G directly from Eq. (3.40) is to approximate it using Monte Carlo simulation. Here, a fairly large number of independent samples of $(n_1, \ldots, n_K)$ are generated using the distribution

$$P(n_1, \ldots, n_K) = \prod_{i=1}^{K} (1 - \rho_i)\rho_i^{n_i}$$

and G is approximated by the proportion of samples that belong to the truncated space S. We will return to the computation of normalization constants in Section 3.8 in the context of queueing networks; see also Problem 3.51.

The reasoning above can also be used to show that there is a product form solution for any truncation of a system consisting of K independent queues each described by a birth–death Markov chain, such as the $M/M/m$, $M/M/\infty$, and $M/M/m/m$ systems. For example, it is straightforward to verify that the stationary distribution of the K independent $M/M/\infty$ queues system is given by

$$P(n_1, n_2, \ldots, n_K) = \frac{\frac{\rho_1^{n_1}}{n_1!}\frac{\rho_2^{n_2}}{n_2!}\cdots\frac{\rho_K^{n_K}}{n_K!}}{G}$$

where G is a normalization constant,

$$G = \sum_{(n_1, n_2, \ldots, n_K) \in S} \frac{\rho_1^{n_1}}{n_1!}\frac{\rho_2^{n_2}}{n_2!}\cdots\frac{\rho_K^{n_K}}{n_K!}$$

and S is the set of states of the truncated chain.

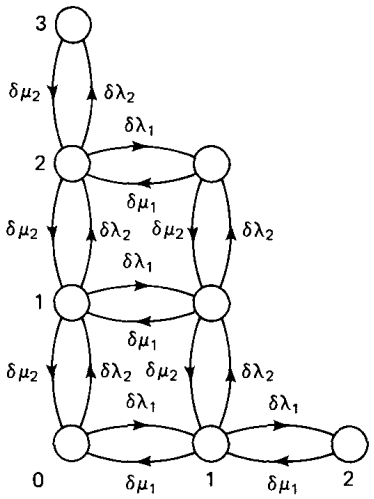


Figure 3.14 Example of a Markov chain which is a truncation of the K independent $M/M/1$ queues system ($K = 2$ in the figure).

Blocking probabilities for circuit switching systems. Using the product form solution (3.39), it is straightforward to write closed-form expressions for the blocking probabilities of the circuit switching systems with two session classes of Examples 3.12 and 3.13. The two-dimensional chains of these examples are truncations of the two independent $M/M/\infty$ queues system (Figs. 3.11 to 3.13). Thus, in the case of Example 3.13, the blocking probability for the first type of session is

$$\sum_{\{(n_1,n_2)|m-k\le n_1\le m, n_2=m-n_1\}} P(n_1,n_2)$$

$$= \frac{\sum_{n_1=k}^{m} \frac{\rho_1^{n_1}}{n_1!}\frac{\rho_2^{m-n_1}}{m-n_1!}}{\sum_{n_1=0}^{k}\sum_{n_2=0}^{k}\frac{\rho_1^{n_1}}{n_1!}\frac{\rho_2^{n_2}}{n_2!} + \sum_{n_1=k+1}^{m}\sum_{n_2=0}^{m-n_1}\frac{\rho_1^{n_1}}{n_1!}\frac{\rho_2^{n_2}}{n_2!}}$$

The blocking probability for the second type of session is

$$\sum_{\{(n_1,n_2)|0\le n_1\le m, n_2=\min\{k,m-n_1\}\}} P(n_1,n_2)$$

$$= \frac{\sum_{n_1=0}^{k} \frac{\rho_1^{n_1}}{n_1!}\frac{\rho_2^{k-n_1}}{m-n_1!} + \sum_{n_1=k+1}^{m}\frac{\rho_1^{n_1}}{n_1!}\frac{\rho_2^{m-n_1}}{m-n_1!}}{\sum_{n_1=0}^{k}\sum_{n_2=0}^{k}\frac{\rho_1^{n_1}}{n_1!}\frac{\rho_2^{n_2}}{n_2!} + \sum_{n_1=k+1}^{m}\sum_{n_2=0}^{m-n_1}\frac{\rho_1^{n_1}}{n_1!}\frac{\rho_2^{n_2}}{n_2!}}$$

The following important example illustrates the wide applicability of product form solutions in circuit switching networks.

Example 3.14 Circuit Switching Networks with Fixed Routing

Consider a network of transmission lines shared by sessions of K different types (see Fig. 3.15). Sessions of type i arrive according to a Poisson rate λ_i and have an exponentially distributed holding time with mean $1/\mu_i$.

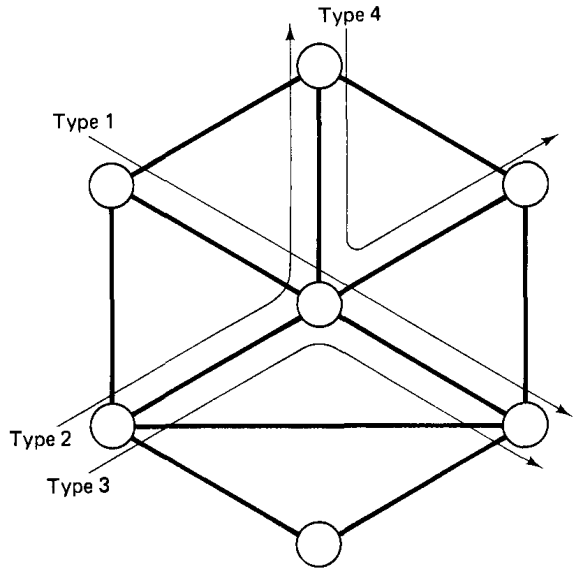


Figure 3.15 Model of a circuit switching network. There are K different session types. All sessions of the same type go over the same path and reserve the same amount of transmission capacity on each link of their path. A session is blocked if some link on its path is loaded to the point that it cannot accommodate the transmission capacity of the session.

Samsung Exhibit 1015, Page 201 of 573

We assume that all sessions of a given type i traverse the same set of network links (fixed routing) and reserve a fixed amount b_i of transmission capacity at each link. Thus if C_j is the transmission capacity of a link j and $I(j)$ is the set of session types using this link, we must have

$$\sum_{i \in I(j)} b_i n_i \leq C_j$$

where n_i is the number of sessions of type i in the network. A session of a given type m is blocked from entering the network (and is assumed lost to the system) if upon arrival it finds that it cannot be accommodated due to insufficient link capacity, that is,

$$b_m + \sum_{i \in I(j)} b_i n_i > C_j$$

for some link j that the session must traverse.

The quality of service of this system may be described by the blocking probabilities for the different session types. To obtain these probabilities, we model the system as a truncation of the K independent $M/M/\infty$ queues system. The truncated chain is the same as for the latter system, except that all states $(n_1, n_2, \ldots, n_K)$ for which the inequality $\sum_{i \in I(j)} b_i n_i \leq C_j$ is violated for some link j have been eliminated. The stationary distribution has a product form, which yields the desired blocking probabilities.

A remarkable fact about the product form solution of this example is that it is valid for a broad class of holding time distributions that includes the exponential as a special case (see [BLL84] and [Kau81]).

3.5 THE $M/G/1$ SYSTEM

Consider a single-server queueing system where customers arrive according to a Poisson process with rate λ, but the customer service times have a general distribution—not necessarily exponential as in the $M/M/1$ system. Suppose that customers are served in the order they arrive and that X_i is the service time of the i^{th} arrival. We assume that the random variables $(X_1, X_2, \ldots)$ are identically distributed, mutually independent, and independent of the interarrival times.

Let

$$\overline{X} = E\{X\} = \frac{1}{\mu} = \text{ Average service time}$$

$$\overline{X^2} = E\{X^2\} = \text{ Second moment of service time}$$

Our objective is to derive and understand the *Pollaczek–Khinchin (P-K) formula:*

$$W = \frac{\lambda \overline{X^2}}{2(1-\rho)} \tag{3.41}$$

where W is the expected customer waiting time in queue and $\rho = \lambda/\mu = \lambda \overline{X}$. Given the P-K formula (3.41), the total waiting time, in queue and in service, is

$$T = \overline{X} + \frac{\lambda \overline{X^2}}{2(1-\rho)} \tag{3.42}$$

Applying Little's formula to W and T, we get the expected number of customers in the queue N_Q and the expected number in the system N:

$$N_Q = \frac{\lambda^2 \overline{X^2}}{2(1-\rho)} \tag{3.43}$$

$$N = \rho + \frac{\lambda^2 \overline{X^2}}{2(1-\rho)} \tag{3.44}$$

For example, when service times are exponentially distributed, as in the $M/M/1$ system, we have $\overline{X^2} = 2/\mu^2$, and the P-K formula (3.41) reduces to the equation (see Section 3.3.2)

$$W = \frac{\rho}{\mu(1-\rho)} \qquad (M/M/1)$$

When service times are identical for all customers (the $M/D/1$ system, where D means deterministic), we have $\overline{X^2} = 1/\mu^2$, and

$$W = \frac{\rho}{2\mu(1-\rho)} \qquad (M/D/1) \tag{3.45}$$

Since the $M/D/1$ case yields the minimum possible value of $\overline{X^2}$ for given μ, it follows that the values of W, T, N_Q, and N for an $M/D/1$ queue are lower bounds to the corresponding quantities for an $M/G/1$ queue of the same λ and μ. It is interesting to note that W and N_Q for the $M/D/1$ queue are exactly one half their values for the $M/M/1$ queue of the same λ and μ. The values of T and N for $M/D/1$, on the other hand, range from the same as $M/M/1$ for small ρ to one half of $M/M/1$ as ρ approaches 1. The reason is that the expected service time is the same in the two cases, and for ρ small, most of the waiting occurs in service, whereas for ρ large, most of the waiting occurs in the queue.

We provide a proof of the Pollaczek–Khinchin formula based on the concept of the *mean residual service time*. This same concept will prove useful in a number of subsequent developments. One example is $M/G/1$ queues with priorities. Another is reservation systems where part of the service time is occupied with sending packets (*i.e.*, serving customers), and part with sending control information or making reservations for sending the packets.

Denote

W_i = Waiting time in queue of the ith customer

R_i = Residual service time seen by the ith customer. By this we mean that if customer j is already being served when i arrives, R_i is the remaining time until customer j's service time is complete. If no customer is in service (*i.e.*, the system is empty when i arrives), then R_i is zero

X_i = Service time of the ith customer

N_i = Number of customers found waiting in queue by the ith customer upon arrival

We have

$$W_i = R_i + \sum_{j=i-N_i}^{i-1} X_j$$

By taking expectations and using the independence of the random variables N_i and $X_{i-1}, \ldots, X_{i-N_i}$, we have

$$E\{W_i\} = E\{R_i\} + E\left\{\sum_{j=i-N_i}^{i-1} E\{X_j \mid N_i\}\right\} = E\{R_i\} + \overline{X}E\{N_i\}$$

Taking the limit as $i \to \infty$, we obtain

$$W = R + \frac{1}{\mu} N_Q \tag{3.46}$$

where

$$R = \text{ Mean residual time, defined as } R = \lim_{i\to\infty} E\{R_i\}.$$

In Eq. (3.46) (and throughout this section) all long-term average quantities should be viewed as limits when time or customer index increases to infinity. Thus, W, R, and N_Q are limits (as $i \to \infty$) of the average waiting time, residual time, and number found in queue, respectively, corresponding to the i^{th} customer. We assume that these limits exist, and this is true of almost all systems of interest to us provided that $\lambda < \mu$. Note that in the waiting time equation (3.46), the average number in queue N_Q and the mean residual time R as seen by an arriving customer are also equal to the average number in queue and mean residual time seen by an outside observer at a random time. This is due to the Poisson character of the arrival process, which implies that the occupancy distribution upon arrival is typical (see Section 3.3.2).

By Little's Theorem, we have

$$N_Q = \lambda W$$

and by substitution in the waiting time formula (3.46), we obtain

$$W = R + \rho W \tag{3.47}$$

where $\rho = \lambda/\mu$ is the utilization factor; so, finally,

$$W = \frac{R}{1-\rho} \tag{3.48}$$

We can calculate R by a graphical argument. In Fig. 3.16 we plot the residual service time $r(\tau)$ (*i.e.*, the remaining time for completion of the customer in service at time τ) as a function of τ. Note that when a new service of duration X begins, $r(\tau)$ starts at X and decays linearly for X time units. Consider a time t for which $r(t) = 0$. The time average of $r(\tau)$ in the interval $[0, t]$ is

$$\frac{1}{t}\int_0^t r(\tau)\, d\tau = \frac{1}{t}\sum_{i=1}^{M(t)} \frac{1}{2} X_i^2 \tag{3.49}$$

where $M(t)$ is the number of service completions within $[0, t]$, and X_i is the service time of the i^{th} customer. We can also write this equation as

$$\frac{1}{t}\int_0^t r(\tau)\,d\tau = \frac{1}{2}\frac{M(t)}{t}\frac{\sum_{i=1}^{M(t)} X_i^2}{M(t)} \tag{3.50}$$

and assuming the limits below exist, we obtain

$$\lim_{t\to\infty}\frac{1}{t}\int_0^t r(\tau)\,d\tau = \frac{1}{2}\lim_{t\to\infty}\frac{M(t)}{t}\cdot\lim_{t\to\infty}\frac{\sum_{i=1}^{M(t)} X_i^2}{M(t)} \tag{3.51}$$

The two limits on the right are the time averages of the departure rate (which equals the arrival rate) and the second moment of the service time, respectively, while the limit on the left is the time average of the residual time. Assuming that time averages can be replaced by ensemble averages, we obtain

$$R = \frac{1}{2}\lambda\overline{X^2} \tag{3.52}$$

The P-K formula,

$$W = \frac{\lambda\overline{X^2}}{2(1-\rho)} \tag{3.53}$$

now follows by substituting the expression obtained for R [cf. Eq. (3.52)] into the formula $W = R/(1-\rho)$ [cf. Eq. (3.48)].

Note that our derivation was based on two assumptions:

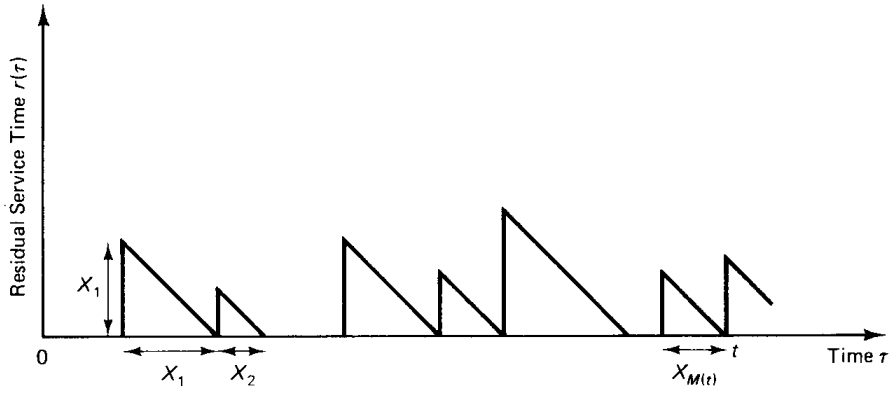


Figure 3.16 Derivation of the mean residual service time. During period $[0, t]$, the time average of the residual service time $r(\tau)$ is

$$\frac{1}{t}\int_0^t r(\tau)\;\;d\tau = \frac{1}{t}\sum_{i=1}^{M(t)}\frac{1}{2}X_i^2 = \frac{1}{2}\frac{M(t)}{t}\frac{\sum_{i=1}^{M(t)} X_i^2}{M(t)}$$

where X_i is the service time of the i^{th} customer, and $M(t)$ is the number of service completions in $[0, t]$. Taking the limit as $t \to \infty$ and equating time and ensemble averages, we obtain the mean residual time $R = (1/2)\lambda\overline{X^2}$.

1. The existence of the steady-state averages W, R, and N_Q
2. The equality (with probability one) of the long-term time averages appearing in Eq. (3.51) with the corresponding ensemble averages

These assumptions can be justified by careful applications of the law of large numbers, but the details are beyond the scope of this book. However, these are natural assumptions for the systems of interest to us, and we will base similar derivations on graphical arguments and interchange of time averages with ensemble averages without further discussion.

One curious feature of the P-K formula (3.53) is that an $M/G/1$ queue can have $\rho < 1$ but infinite W if the second moment $\overline{X^2}$ is ∞. What is happening in this case is that a small fraction of customers have incredibly long service times. When one of these customers is served, an incredible number of arrivals are queued and delayed by a significant fraction of that long service time. Thus, the contribution to W is proportional to the square of the service time, leading to an infinite W if $\overline{X^2}$ is infinite.

The derivation of the P-K formula above assumed that customers were served in order of arrival, that is, that the number of customers served between the i^{th} arrival and service is just the number in queue at the i^{th} arrival. It turns out, however, that this formula is valid for any order of servicing customers as long as the order is determined independently of the required service time. To see this, suppose that the i^{th} and j^{th} customers are both in the queue and that they exchange places. The expected queueing time of customer i (over the service times of the customers in queue) will then be exchanged with that for customer j, but the average, over all customers, is unchanged. Since any service order can be considered as a sequence of reversals in queue position, the P-K formula remains valid (see also Problem 3.32).

To see why the P-K formula is invalid if the service order can depend on service time, consider a queue with two customers requiring 10 and 1 units of service time, respectively. Assuming that the server becomes available at time 0, serving the first customer first results in one customer starting service at time 0 and the other at time 10. Serving the second customer first results in one customer starting at time 0 and the other at time 1. Thus, the average queueing time over the two customers is 5 in the first case and 0.5 in the second case. Clearly, queueing time is reduced by serving customers with small service time first. For this situation, the derivation of the P-K formula breaks down at Eq. (3.46) since the customers that will be served before a newly arriving customer no longer have a mean service time equal to $1/\mu$.

Example 3.15 Delay Analysis of an ARQ System

Consider a go back n ARQ system such as the one discussed in Section 2.4. Assume that packets are transmitted in frames that are one time unit long, and there is a maximum wait for an acknowledgment of $n-1$ frames before a packet is retransmitted (see Fig. 3.17). In this system packets are retransmitted for two reasons:

1. A given packet transmitted in frame i might be rejected at the receiver due to errors, in which case the transmitter will transmit packets in frames $i+1, i+2, \ldots, i+n-1$, (if any are available), and then go back to retransmit the given packet in frame $i+n$.

2. A packet transmitted in frame i might be accepted at the receiver, but the corresponding acknowledgment (in the form of the receive number) might not have arrived at the transmitter by the time the transmission of packet $i + n - 1$ is completed. This can happen due to errors in the return channel, large propagation delays, long return frames relative to the size of the goback number n, or a combination thereof.

We will assume (somewhat unrealistically) that retransmissions occur only due to reason 1, and that a packet is rejected at the receiver with probability p independently of other packets.

Consider the case where packets arrive at the transmitter according to a Poisson process with rate λ. It follows that the time interval between start of the first transmission of a given packet after the last transmission of the previous packet and end of the last transmission of the given packet is $1 + kn$ time units with probability $(1 - p)p^k$. (This corresponds to k retransmissions following the last transmission of the previous packet; see Fig. 3.17.) Thus, the transmitter's queue behaves like an $M/G/1$ queue with service time distribution given by

$$P\{X = 1 + kn\} = (1 - p)p^k, \quad k = 0, 1, \ldots$$

The first two moments of the service time are

$$\overline{X} = \sum_{k=0}^{\infty}(1 + kn)(1 - p)p^k = (1 - p)\left(\sum_{k=0}^{\infty} p^k + n\sum_{k=0}^{\infty} kp^k\right)$$

$$\overline{X^2} = \sum_{k=0}^{\infty}(1 + kn)^2(1 - p)p^k = (1 - p)\left(\sum_{k=0}^{\infty} p^k + 2n\sum_{k=0}^{\infty} kp^k + n^2\sum_{k=0}^{\infty} k^2p^k\right)$$

We now note that

$$\sum_{k=0}^{\infty} p^k = \frac{1}{1 - p}, \qquad \sum_{k=0}^{\infty} kp^k = \frac{p}{(1 - p)^2}, \qquad \sum_{k=0}^{\infty} k^2p^k = \frac{p + p^2}{(1 - p)^3}$$

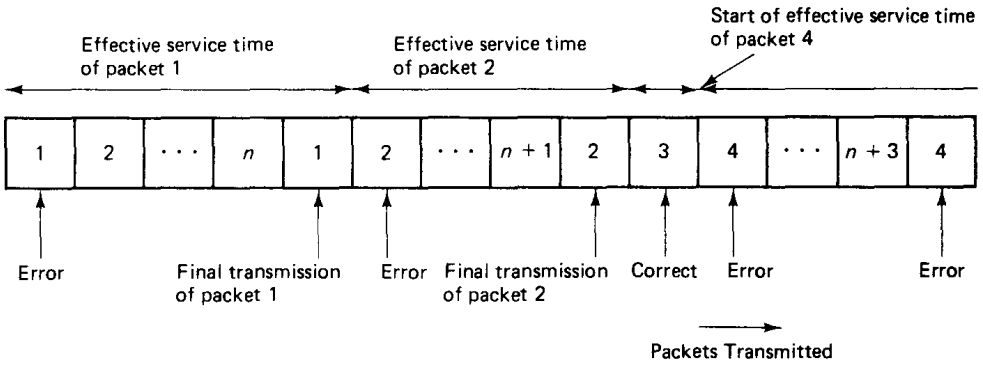


Figure 3.17 Illustration of the effective service times of packets in the ARQ system of Example 3.15. For example, packet 2 has an effective service time of $n + 1$ because there was an error in the first attempt to transmit it following the last transmission of packet 1, but no error in the second attempt.

Samsung Exhibit 1015, Page 207 of 573

(The first sum is the usual geometric series sum, while the other two sums are obtained by differentiating the first sum twice.) Using these formulas in the equations for $\overline{X}$ and $\overline{X^2}$ above, we obtain

$$\overline{X} = 1 + \frac{np}{1-p}$$

$$\overline{X^2} = 1 + \frac{2np}{1-p} + \frac{n^2(p+p^2)}{(1-p)^2}$$

The P-K formula gives the average packet time in queue and in the system (up to the end of the last transmission):

$$W = \frac{\lambda \overline{X^2}}{2(1-\lambda \overline{X})}$$

$$T = \overline{X} + W$$

3.5.1 *M*/*G*/1 Queues with Vacations

Suppose that at the end of each busy period, the server goes on "vacation" for some random interval of time. Thus, a new arrival to an idle system, rather than going into service immediately, waits for the end of the vacation period (see Fig. 3.18). If the system is still idle at the completion of a vacation, a new vacation starts immediately. For data networks, vacations correspond to the transmission of various kinds of control and record-keeping packets when there is a lull in the data traffic; other applications will become apparent later.

Let $V_1, V_2, \ldots$ be the durations of the successive vacations taken by the server. We assume that $V_1, V_2, \ldots$ are independent and identically distributed (IID) random variables, also independent of the customer interarrival times and service times. As before, the arrivals are Poisson and the service times are IID with a general distribution. A new arrival to the system has to wait in the queue for the completion of the current service or vacation and then for the service of all the customers waiting before it. Thus, the waiting time formula $W = R/(1-\rho)$ is still valid [cf. Eq. (3.48)], where now R is the mean residual time for completion of the service *or* vacation in process when the i^{th} customer arrives.

The analysis of this new system is the same as that of the P-K formula except that vacations must be included in the graph of residual service times $r(\tau)$ (see Fig. 3.19). Let $M(t)$ be the number of services completed by time t and $L(t)$ be the number of vacations completed by time t. Then [as in Eq. (3.49)], for any t where a service or vacation is just completed, we have

$$\frac{1}{t}\int_0^t r(\tau)\,d\tau = \frac{1}{t}\sum_{i=1}^{M(t)} \frac{1}{2}X_i^2 + \frac{1}{t}\sum_{i=1}^{L(t)} \frac{1}{2}V_i^2 = \frac{M(t)}{t}\frac{\sum_{i=1}^{M(t)} \frac{1}{2}X_i^2}{M(t)} + \frac{L(t)}{t}\frac{\sum_{i=1}^{L(t)} \frac{1}{2}V_i^2}{L(t)} \tag{3.54}$$

As before, assuming that a steady-state exists, $M(t)/t$ approaches λ with increasing t, and the first term on the right side of Eq. (3.54) approaches $\lambda \overline{X^2}/2$ as in the derivation

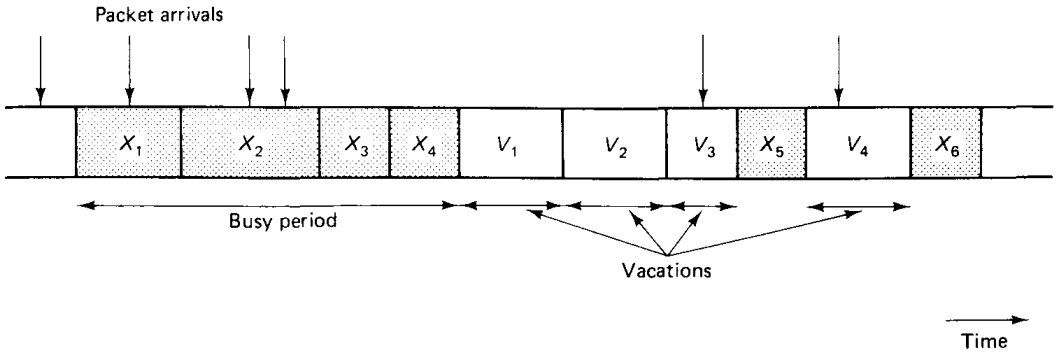


Figure 3.18 $M/G/1$ system with vacations. At the end of a busy period, the server goes on vacation for time V with first and second moments $\overline{V}$ and $\overline{V^2}$, respectively. If the system is empty at the end of a vacation, the server takes a new vacation. An arriving customer to an empty system must wait until the end of the current vacation to get service.

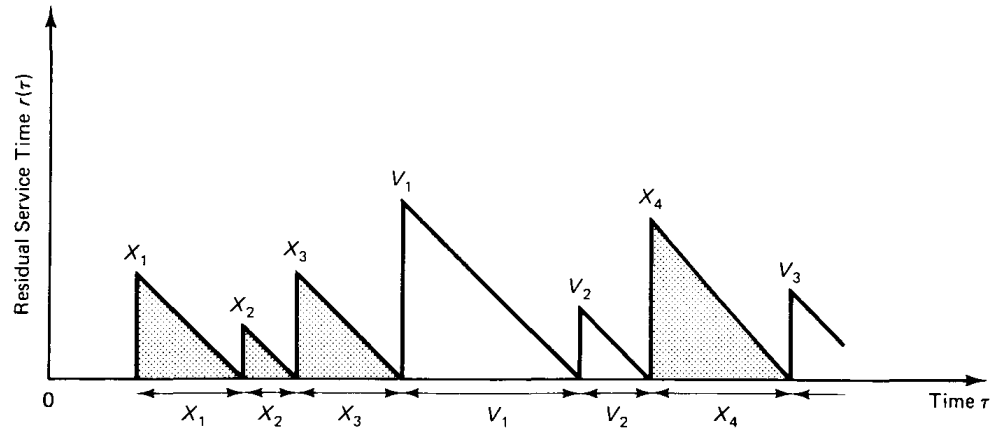


Figure 3.19 Residual service times for an $M/G/1$ system with vacations. Busy periods alternate with vacation periods. If $M(t)$ and $L(t)$ are the numbers of services and vacations completed by time t, respectively, and t is a time of completion of a service or a vacation, we have

$$\frac{1}{t}\int_0^t r(\tau)\,d\tau = \frac{1}{t}\sum_{i=1}^{M(t)}\frac{1}{2}X_i^2 + \frac{1}{t}\sum_{i=1}^{L(t)}\frac{1}{2}V_i^2 = \frac{M(t)}{t}\frac{\sum_{i=1}^{M(t)}\frac{1}{2}X_i^2}{M(t)} + \frac{L(t)}{t}\frac{\sum_{i=1}^{L(t)}\frac{1}{2}V_i^2}{L(t)}$$

Taking limit as $t \to \infty$ and arguing that $M(t)/t \to \lambda$ and $L(t)/t \to (1-\rho)/\overline{V}$, we obtain the mean residual time $R = \frac{\lambda\overline{X^2}}{2} + \frac{(1-\rho)\overline{V^2}}{2\overline{V}}$

of the P-K formula [cf. Eq. (3.52)]. For the second term, note that as $t \to \infty$, the fraction of time spent serving customers approaches ρ, and thus the fraction of time occupied with vacations is $1 - \rho$. Assuming that time averages can be replaced by ensemble averages, we have $t(1-\rho)/L(t) \to \overline{V}$ with increasing t, and thus the second term in Eq. (3.54) approaches $(1-\rho)\overline{V^2}/(2\overline{V})$, where $\overline{V}$and $\overline{V^2}$ are the first and second moments of the vacation interval, respectively. Combining this with $W = R/(1-\rho)$, and assuming

equality of the time and ensemble averages of R, we get

$$W = \frac{\lambda \overline{X^2}}{2(1-\rho)} + \frac{\overline{V^2}}{2\overline{V}} \tag{3.55}$$

as the expected waiting time in queue for an $M/G/1$ system with vacations.

If we look carefully at the derivation of the preceding equation, we see that the mutual independence of the vacation intervals is not required (although the time and ensemble averages of the vacation intervals must still be equal) and the length of a vacation interval need not be independent of the customer arrival and service times. Naturally, with this kind of dependence, it becomes more difficult to calculate $\overline{V}$ and $\overline{V^2}$, as these quantities might be functions of the underlying $M/G/1$ process.

Example 3.16 Frequency- and Time-Division Multiplexing on a Slot Basis

We have m traffic streams of equal-length packets arriving according to a Poisson process with rate λ/m each. If the traffic streams are frequency-division multiplexed on m subchannels of an available channel, the transmission time of each packet is m time units. Then, each subchannel can be represented by an $M/D/1$ queueing system and the $M/D/1$ formula $W = \rho/(2\mu(1-\rho))$ [cf. Eq. (3.45)] with $\rho = \lambda$, $\mu = 1/m$, gives the average queueing delay per packet,

$$W_{\text{FDM}} = \frac{\lambda m}{2(1-\lambda)} \tag{3.56}$$

Consider the same FDM scheme with the difference that packet transmissions can start only at times $m, 2m, 3m, \ldots$ (*i.e.*, at the beginning of a slot of m time units). We call this scheme *slotted frequency-division multiplexing* (SFDM), and note that it can be viewed as an $M/D/1$ queue with vacations. When there are no packets in the queue for a given stream at the beginning of a slot, the server takes a vacation for one slot, or m time units. Thus, $\overline{V} = m$, $\overline{V^2} = m^2$, and the vacation system waiting time formula (3.55) becomes

$$W_{\text{SFDM}} = W_{\text{FDM}} + \frac{m}{2} \tag{3.57}$$

Finally, consider the case where the m traffic streams are time-division multiplexed in a scheme whereby the time axis is divided in m-slot frames with one slot dedicated to each traffic stream (see Fig. 3.20). Each slot is one time unit long and can carry a single packet. Then, if we compare this TDM scheme with the SFDM scheme, we see that the queue for a given stream in TDM is precisely the same as the queue for SFDM, and

$$W_{\text{TDM}} = W_{\text{SFDM}} = W_{\text{FDM}} + \frac{m}{2} = \frac{m}{2(1-\lambda)} \tag{3.58}$$

If we now look at the total delay for TDM, we get a different picture, since the service time is 1 unit of time rather than m units as in SFDM. By adding the service times to the queueing delays, we obtain

$$\begin{aligned} T_{\text{FDM}} &= m + \frac{\lambda m}{2(1-\lambda)} \\ T_{\text{SFDM}} &= T_{\text{FDM}} + \frac{m}{2} \\ T_{\text{TDM}} &= 1 + \frac{m}{2(1-\lambda)} = T_{\text{FDM}} - \left(\frac{m}{2} - 1\right) \end{aligned} \tag{3.59}$$

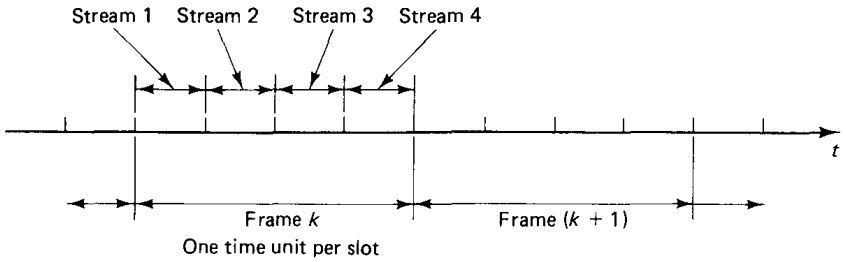


Figure 3.20 TDM with $m = 4$ traffic streams.

Thus, the customer's average total delay is more favorable in TDM than in FDM (assuming that $m > 2$). The longer average waiting time in queue for TDM is more than compensated by the faster service time. Contrast this with the Example 3.9, which treats TDM with slots that are a very small portion of the packet size. Problem 3.33 outlines an alternative approach for deriving the TDM average delay.

3.5.2 Reservations and Polling

Organizing transmissions from several packet streams into a statistical multiplexing system requires some form of scheduling. In some cases, this scheduling is naturally and easily accomplished; in other cases, however, some form of reservation or polling system is required.

Situations of this type arise often in multiaccess channels, which will be treated extensively in Chapter 4. For a typical example, consider a communication channel that can be accessed by several spatially separated users; however, only one user can transmit successfully on the channel at any one time. The communication resource of the channel can be divided over time into a portion used for packet transmissions and another portion used for reservation or polling messages that coordinate the packet transmissions. In other words, the time axis is divided into *data intervals*, where actual data are transmitted, and *reservation intervals*, used for scheduling future data. For uniform presentation, we use the term "reservation" even though "polling" may be more appropriate to the practical situation.

We will consider m traffic streams (also called users) and assume that each data interval contains packets of a *single* user. Reservations for these packets are made in the immediately preceding reservation interval. All users are taken up in cyclic order (see Fig. 3.21). There are several versions of this system differing in the rule for deciding which packets are transmitted during the data interval of each user. In the *gated* system, the rule is that only those packets that arrived prior to the user's preceding reservation interval are transmitted. By contrast, in the *exhaustive* system, the rule is that all available packets of a user are transmitted during the corresponding data interval, including those that arrived in this data interval or the preceding reservation interval. An intermediate version, which we call the *partially gated* system, results when the packets transmitted in a user's data interval are those that arrived up to the time this data interval began (and the corresponding reservation interval ended). A typical example of such reservation systems

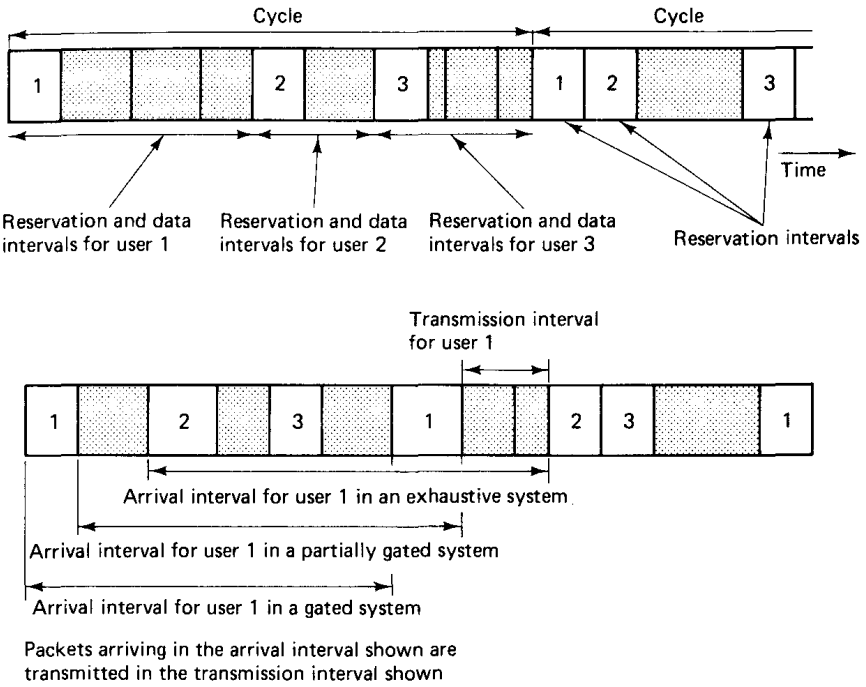


Figure 3.21 Reservation or polling system with three users. In the exhaustive version, a packet of a user that arrives during the user's reservation or data interval is transmitted in the same data interval. In the partially gated version, a packet of a user arriving during the user's data interval must wait for an entire cycle and be transmitted during the next data interval of the user. In the fully gated version, packets arriving during the user's reservation interval must also wait for an entire cycle. The figure shows, for the three systems, the association between the interval in which a packet arrives and the interval in which the packet is transmitted.

is one of the most common local area networks, the token ring. The users are connected by cable in a unidirectional loop. Each user transmits the current packet backlog, then gives the opportunity to a neighbor to transmit, and the process is repeated. (A more detailed description of the token ring is given in Chapter 4.)

We assume that the arrival processes of all users are independent Poisson with rate λ/m, and that the first and second moments of the packet transmission times are $\overline{X} = 1/\mu$ and $\overline{X^2}$, respectively. The utilization factor is $\rho = \lambda/\mu$. Interarrival times and transmission times are, as usual, assumed independent. While we assume that all users have identical arrival and service statistics, we allow the reservation intervals of different users to have different statistics.

Single-user system. Our general line of analysis of reservation systems can be better understood in terms of the special case where $m = 1$, so we consider this case first. We may also view this as a system where all users share reservation and data intervals. Let V_ℓ be the duration of the ℓ^{th} reservation interval and assume that successive reservation intervals are independent and identically distributed random variables with first and second moments $\overline{V}$ and $\overline{V^2}$, respectively. We consider a gated system and

assume that the reservation intervals are statistically independent of the arrival times and service durations. Finally, for convenience of exposition, we assume that packets are transmitted in the order of their arrival. As in our derivation of the P-K formula, expected delays and queue lengths are independent of service order as long as service order is independent of service requirement (*i.e.*, packet length).

Consider the i^{th} data packet arriving at the system. This packet must wait in queue for the residual time R_i until the end of the current packet transmission or reservation interval. It must also wait for the transmission of the N_i packets currently in the queue (this includes both packets for which reservations were already made in the last reservation interval and earlier arrivals waiting to make a reservation). Finally, the packet must wait during the next reservation interval $V_{\ell(i)}$, say, in which its reservation will be made (see Fig. 3.22). Thus, the expected queueing delay for the i^{th} packet is given by

$$E\{W_i\} = E\{R_i\} + \frac{E\{N_i\}}{\mu} + E\{V_{\ell(i)}\} \tag{3.60}$$

The similarity of this reservation system to the $M/G/1$ queue with vacations should be noted. The only difference is that in the gated reservation system, a reservation interval starts when all packets that arrived prior to the start of the preceding reservation interval have been served, whereas in the vacation system, a vacation interval starts when all arrivals up to the current time have been served and the system is empty. (Thus in the gated reservation system, every packet has to wait in queue for a full reservation interval, while in the vacation system, only the packets that find the system empty upon arrival have to wait for part of a vacation interval. Note that the exhaustive version of this reservation system is equivalent to the vacation system.) The time-average mean residual time for the two systems is the same (the calculation based on Fig. 3.19 still applies) and is given by $\lambda\overline{X^2}/2 + (1-\rho)\overline{V^2}/2\overline{V}$. The value of $\lim_{i\to\infty} E\{N_i\}/\mu$ is ρW in both systems, and finally the value of $\lim_{i\to\infty} E\{V_{\ell(i)}\}$ is just $\overline{V}$. Thus, from

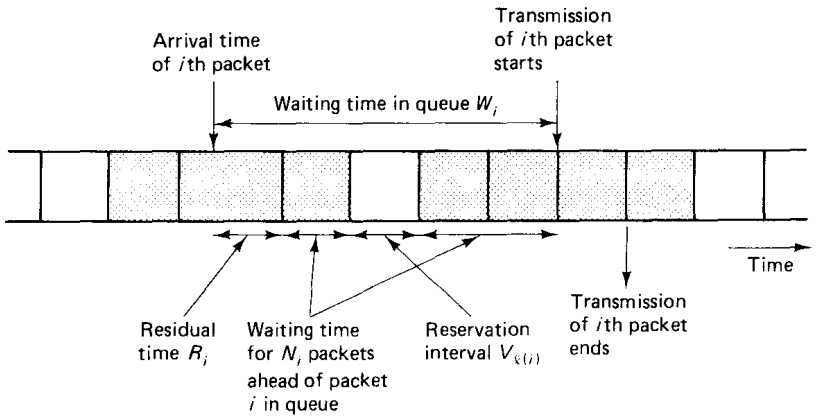


Figure 3.22 Calculation of the average waiting time in the single-user gated system. The expected waiting time $E\{W_i\}$ of the i^{th} packet is

$$E\{W_i\} = E\{R_i\} + \frac{E\{N_i\}}{\mu} + E\{V_{\ell(i)}\}$$

Samsung Exhibit 1015, Page 213 of 573

Eq. (3.60) the expected time in queue for the single-user reservation system is

$$W = \frac{\lambda \overline{X^2}}{2(1-\rho)} + \frac{\overline{V^2}}{2\overline{V}} + \frac{\overline{V}}{1-\rho} \qquad \text{(single user, gated)} \tag{3.61}$$

In the common situation where the reservation interval is a constant A, this simplifies to

$$W = \frac{\lambda \overline{X^2}}{2(1-\rho)} + \frac{A}{2}\left(\frac{3-\rho}{1-\rho}\right) \tag{3.62}$$

There is an interesting paradox associated with the waiting time formula (3.61). We have seen that a fraction $1-\rho$ of time is used on reservations. Since there is one reservation interval of mean duration $\overline{V}$ per cycle, we can conclude that the expected cycle length must be $\overline{V}/(1-\rho)$ (see also Example 3.6). The mean queueing delay in Eq. (3.61) can be an arbitrarily large multiple of this mean cycle length, which seems paradoxical since each packet is transmitted on the cycle following its arrival. The explanation of this is that more packets tend to arrive in long cycles than in short cycles, and thus mean cycle length is not representative of the cycle lengths seen by arriving packets; this is the same phenomenon that makes the mean residual service time used in the P-K formula derivation larger than one might think (see also Problem 3.31).

Multiuser system. Suppose that the system has m users, each with independent Poisson arrivals of rate λ/m. Again $\overline{X}$ and $\overline{X^2}$ are the first two moments of the service time for each user's packets. We denote by $\overline{V_i}$ and $\overline{V_i^2}$, respectively, the first two moments of the reservation intervals of user i. The service times and reservation intervals are all independent. We number the users $0, 1, \ldots, m-1$ and assume that the ℓ^{th} reservation interval is used to make reservations for user $\ell \bmod m$ and the subsequent (ℓ^{th}) data interval is used to send the packets corresponding to those reservations.

Consider the i^{th} packet arrival into the system (counting packets in order of arrival, regardless of user). As before, the expected delay for this packet consists of three terms: first, the mean residual time for the packet or reservation in progress; second, the expected time to transmit the number N_i of packets that must be transmitted before packet i; and third, the expected duration of reservation intervals (see Fig. 3.23). Thus,

$$E\{W_i\} = E\{R_i\} + \frac{E\{N_i\}}{\mu} + E\{Y_i\} \tag{3.63}$$

where Y_i is the duration of all the whole reservation intervals during which packet i must wait before being transmitted. The time average mean residual time is calculated as before, and is given by

$$R = \frac{\lambda \overline{X^2}}{2} + \frac{(1-\rho)\sum_{\ell=0}^{m-1} \overline{V_\ell^2}}{2\sum_{\ell=0}^{m-1} \overline{V_\ell}} \tag{3.64}$$

The number of packets N_i that i must wait for is not equal to the number already in queue, but the order of serving packets is independent of packet service time; thus, each packet served before i still has a mean transmission time $1/\mu$ as indicated in Eq. (3.63) and by Little's formula, the value of $\lim_{i\to\infty} E\{N_i\}/\mu$ is ρW as before.

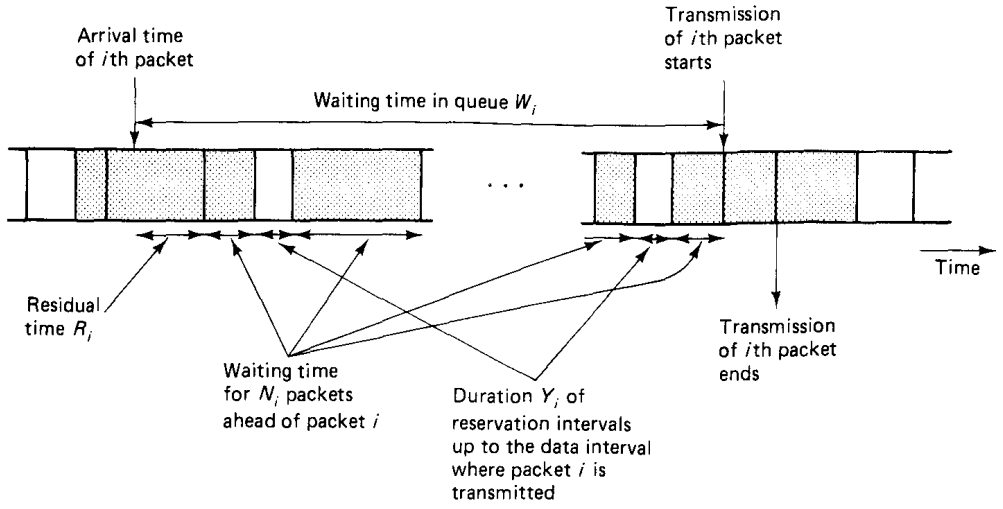


Figure 3.23 Calculation of the average waiting time in the multiuser system. The expected waiting time $E\{W_i\}$ of the i^{th} packet is

$$E\{W_i\} = E\{R_i\} + \frac{E\{N_i\}}{\mu} + E\{Y_i\}$$

Letting $Y = \lim_{i\to\infty} E\{Y_i\}$, we can thus write the steady-state version of Eq. (3.63):

$$W = R + \rho W + Y$$

or, equivalently,

$$W = \frac{R+Y}{1-\rho} \tag{3.65}$$

We first calculate Y for an exhaustive system. Denote

$$\alpha_{\ell j} = E\{Y_i \mid \text{ packet } i \text{ arrives in user } \ell\text{'s reservation or data interval and belongs to user } (\ell + j) \bmod m\}$$

We have

$$\alpha_{\ell j} = \begin{cases} 0, & j = 0 \\ \overline{V}_{(\ell+1) \bmod m} + \cdots + \overline{V}_{(\ell+j) \bmod m}, & j > 0 \end{cases}$$

Since packet i belongs to any user with equal probability $1/m$, we have

$$E\{Y_i \mid \text{ packet } i \text{ arrives in user } \ell\text{'s reservation or data interval}\}$$

$$= \frac{1}{m}\sum_{j=1}^{m-1} \alpha_{\ell j} = \sum_{j=1}^{m-1} \frac{m-j}{m} \overline{V}_{(\ell+j) \bmod m} \tag{3.66}$$

Since all users have equal data rate, the data intervals of all users have equal average length in steady-state. Therefore, in steady-state, a packet will arrive during user ℓ's data interval with probability ρ/m, and during user ℓ's reservation interval with probability

$(1-\rho)\overline{V}_\ell / \left(\sum_{k=0}^{m-1} \overline{V}_k\right)$. Using this fact in Eq. (3.66), we obtain the following equation for $Y = \lim_{i\to\infty} E\{Y_i\}$:

$$Y = \sum_{\ell=0}^{m-1} \left(\frac{\rho}{m} + \frac{(1-\rho)\overline{V}_\ell}{\sum_{k=0}^{m-1} \overline{V}_k} \right) \sum_{j=1}^{m-1} \frac{m-j}{m} \overline{V}_{(\ell+j) \bmod m}$$

$$= \frac{\rho}{m} \sum_{j=1}^{m-1} \frac{m-j}{m} \left(\sum_{\ell=0}^{m-1} \overline{V}_\ell \right) + \frac{1-\rho}{\sum_{k=0}^{m-1} \overline{V}_k} \sum_{\ell=0}^{m-1} \sum_{j=1}^{m-1} \frac{m-j}{m} \overline{V}_\ell \overline{V}_{(\ell+j) \bmod m} \quad (3.67)$$

The last sum above can be written

$$\sum_{\ell=0}^{m-1} \sum_{j=1}^{m-1} \frac{m-j}{m} \overline{V}_\ell \overline{V}_{(\ell+j) \bmod m} = \frac{1}{2} \left[\left(\sum_{\ell=0}^{m-1} \overline{V}_\ell \right)^2 - \sum_{\ell=0}^{m-1} \overline{V}_\ell^2 \right]$$

(To see this, note that the right side above is the sum of all possible products $\overline{V}_\ell \overline{V}_{\ell'}$ for $\ell \neq \ell'$. The left side is the sum of all possible terms $(j/m)\overline{V}_\ell \overline{V}_{\ell'}$ and $[(m-j)/m]\overline{V}_\ell \overline{V}_{\ell'}$, where $j = |\ell - \ell'|$ and $\ell \neq \ell'$.) Using this expression, and denoting

$$\overline{V} = \frac{1}{m} \sum_{\ell=0}^{m-1} \overline{V}_\ell$$

as the reservation interval averaged over all users, we can write Eq. (3.67) as

$$Y = \frac{\rho \overline{V}(m-1)}{2} + \frac{(1-\rho)m\overline{V}}{2} - \frac{(1-\rho)\sum_{\ell=0}^{m-1} \overline{V}_\ell^2}{2m\overline{V}}$$

$$= \frac{(m-\rho)\overline{V}}{2} - \frac{(1-\rho)\sum_{\ell=0}^{m-1} \overline{V}_\ell^2}{2m\overline{V}} \quad (3.68)$$

Combining Eqs. (3.64), (3.65), and (3.68), we obtain

$$W = \frac{\lambda \overline{X^2}}{2(1-\rho)} + \frac{(m-\rho)\overline{V}}{2(1-\rho)} + \frac{\sum_{\ell=0}^{m-1} \left(\overline{V_\ell^2} - \overline{V}_\ell^2 \right)}{2m\overline{V}}$$

Denoting

$$\sigma_V^2 = \frac{\sum_{\ell=0}^{m-1} \left(\overline{V_\ell^2} - \overline{V}_\ell^2 \right)}{m}$$

as the variance of the reservation intervals averaged over all users, we finally obtain

$$W = \frac{\lambda \overline{X^2}}{2(1-\rho)} + \frac{(m-\rho)\overline{V}}{2(1-\rho)} + \frac{\sigma_V^2}{2\overline{V}} \quad \text{(exhaustive)} \quad (3.69)$$

The partially gated system is the same as the exhaustive except that if a packet of a user arrives during a user's own data interval (an event of probability ρ/m in steady-state), it is delayed by an additional $m\overline{V}$, the average sum of reservation intervals in a

cycle. Thus, Y is increased by $\rho\overline{V}$ in the preceding calculation, and we obtain

$$W = \frac{\lambda\overline{X^2}}{2(1-\rho)} + \frac{(m+\rho)\overline{V}}{2(1-\rho)} + \frac{\sigma_V^2}{2\overline{V}} \qquad \text{(partially gated)} \tag{3.70}$$

Consider, finally, the fully gated system. This is the same as the partially gated system except that if a packet of a user arrives during a user's own reservation interval [an event of probability $(1-\rho)/m$ in steady-state], it is delayed by an additional $m\overline{V}$. This increases Y by an additional $(1-\rho)\overline{V}$ and results in the equation

$$W = \frac{\lambda\overline{X^2}}{2(1-\rho)} + \frac{(m+2-\rho)\overline{V}}{2(1-\rho)} + \frac{\sigma_V^2}{2\overline{V}} \qquad \text{(gated)} \tag{3.71}$$

In comparing these results with the single-user system, consider the case where the reservation interval is a constant A/m. Thus, A is the overhead or reservation time for an entire cycle of reservations for each user, which is usually the appropriate parameter to compare with A in the single-user waiting-time formula (3.62). We then have ($\overline{V} = A/m$, $\sigma_V^2 = 0$)

$$W = \frac{\lambda\overline{X^2}}{2(1-\rho)} + \frac{A}{2}\left(\frac{1-\rho/m}{1-\rho}\right) \qquad \text{(exhaustive)} \tag{3.72}$$

$$W = \frac{\lambda\overline{X^2}}{2(1-\rho)} + \frac{A}{2}\left(\frac{1+\rho/m}{1-\rho}\right) \qquad \text{(partially gated)} \tag{3.73}$$

$$W = \frac{\lambda\overline{X^2}}{2(1-\rho)} + \frac{A}{2}\left(\frac{1+(2-\rho)/m}{1-\rho}\right) \qquad \text{(gated)} \tag{3.74}$$

It can be seen that delay is somewhat reduced in the multiuser case; essentially, packets are delayed by roughly the same amount until the reservation time in all cases, but delay is quite small after the reservation in the multiuser case.

Limited service systems. We now consider a variation of the multiuser system whereby, in each user's data interval, *only the first* packet of the user waiting in queue (if any) is transmitted (rather than *all* waiting packets). We concentrate on the gated and partially gated versions of this system, since an exhaustive version does not make sense. As before, we have

$$E\{W_i\} = E\{R_i\} + \frac{E\{N_i\}}{\mu} + E\{Y_i\}$$

and by taking the limit as $i \to \infty$, we obtain

$$W = R + \rho W + Y \tag{3.75}$$

Here R is given by Eq. (3.64) as before. To calculate the new formula for Y for the partially gated system, we argue as follows. A packet arriving during user ℓ's data or reservation interval will belong to any one of the users with equal probability $1/m$. Therefore, in steady-state, the expected number of packets waiting in the queue of the user that owns the arriving packet, averaged over all users, is $\lim_{i\to\infty} E\{N_i\}/m = \lambda W/m$.

Each of these packets causes an extra cycle of reservations $m\overline{V}$, so Y is increased by an amount $\lambda W\overline{V}$. Using this fact in Eq. (3.75), we see that

$$W = \frac{R + \tilde{Y}}{1 - \rho - \lambda\overline{V}}$$

where $\tilde{Y}$ is the value of Y obtained earlier for the partially gated system without the single-packet-per-data-interval restriction. Equivalently, we see from Eq. (3.65) that *the single-packet-per-data-interval restriction results in an increase of the average waiting time for the partially gated system by a factor*

$$\frac{1 - \rho}{1 - \rho - \lambda\overline{V}}$$

Using this fact in Eq. (3.70), we obtain

$$W = \frac{\lambda\overline{X^2}}{2(1 - \rho - \lambda\overline{V})} + \frac{(m + \rho)\overline{V}}{2(1 - \rho - \lambda\overline{V})} + \frac{\sigma_V^2(1 - \rho)}{2\overline{V}(1 - \rho - \lambda\overline{V})}$$

(limited service, partially gated) (3.76)

Consider now the gated version. Y_i is the same as for the partially gated system except for an additional cycle of reservation intervals of average length $m\overline{V}$ associated with the event where packet i arrives during the reservation interval of its owner, and the subsequent data interval is empty. It is easily verified (Problem 3.34) that the latter event occurs with steady-state probability $(1 - \rho - \lambda\overline{V})/m$. Therefore, for the gated system Y equals the corresponding value for the partially gated system plus $(1 - \rho - \lambda\overline{V})\overline{V}$. This adds $\overline{V}$ to the value of W for the partially gated system, and the average waiting time now is

$$W = \frac{\lambda\overline{X^2}}{2(1 - \rho - \lambda\overline{V})} + \frac{(m + 2 - \rho - 2\lambda\overline{V})\overline{V}}{2(1 - \rho - \lambda\overline{V})} + \frac{\sigma_V^2(1 - \rho)}{2\overline{V}(1 - \rho - \lambda\overline{V})}$$

(limited service, gated) (3.77)

Note that it is not enough that $\rho = \lambda/\mu < 1$ for W to be bounded; rather, $\rho + \lambda\overline{V} < 1$ is required or, equivalently,

$$\lambda\left(\frac{1}{\mu} + \overline{V}\right) < 1$$

This is due to the fact that each packet requires a separate reservation interval of average length $\overline{V}$, thereby effectively increasing the average transmission time from $1/\mu$ to $1/\mu + \overline{V}$.

As a final remark, consider the case of a very large number of users m and a very small average reservation interval $\overline{V}$. An examination of the equation given for the average waiting time W of every multiuser system considered so far shows that as $m \to \infty$, $\overline{V} \to 0$, $\sigma_V^2/\overline{V} \to 0$, and $m\overline{V} \to A$, where A is a constant, we have

$$W \to \frac{\lambda\overline{X^2}}{2(1 - \rho)} + \frac{A}{2(1 - \rho)}$$

It can be shown (see Example 3.6) that $A/(1-\rho)$ is the average length of a cycle (m successive reservation and data intervals). Thus, W approaches the $M/G/1$ average waiting time plus one half the average cycle length.

3.5.3 Priority Queueing

Consider the $M/G/1$ system with the difference that arriving customers are divided into n different priority classes. Class 1 has the highest priority, class 2 has the second highest, and so on. The arrival rate and the first two moments of service time of each class k are denoted λ_k, $\overline{X}_k = 1/\mu_k$, and $\overline{X_k^2}$, respectively. The arrival processes of all classes are assumed independent, Poisson, and independent of the service times.

Nonpreemptive priority. We first consider the nonpreemptive priority rule whereby a customer undergoing service is allowed to complete service without interruption even if a customer of higher priority arrives in the meantime. A separate queue is maintained for each priority class. When the server becomes free, the first customer of the highest nonempty priority queue enters service. This priority rule is one of the most appropriate for modeling packet transmission systems.

We will develop an equation for average delay for each priority class, which is similar to the P-K formula and admits a similar derivation. Denote

$$N_Q^k = \text{Average number in queue for priority } k$$

$$W_k = \text{Average queueing time for priority } k$$

$$\rho_k = \lambda_k/\mu_k = \text{System utilization for priority } k$$

$$R = \text{Mean residual service time}$$

We assume that the overall system utilization is less than 1, that is,

$$\rho_1 + \rho_2 + \cdots + \rho_n < 1$$

When this assumption is not satisfied, there will be some priority class k such that the average delay of customers of priority k and lower will be infinite while the average delay of customers of priority higher than k will be finite. Problem 3.39 takes a closer look at this situation.

As in the derivation of the P-K formula given earlier, we have for the highest-priority class,

$$W_1 = R + \frac{1}{\mu_1} N_Q^1$$

Eliminating N_Q^1 from this equation using Little's Theorem,

$$N_Q^1 = \lambda_1 W_1$$

we obtain

$$W_1 = R + \rho_1 W_1$$

and finally,

$$W_1 = \frac{R}{1 - \rho_1} \tag{3.78}$$

For the second priority class, we have a similar expression for the queueing delay W_2 except that we have to count the additional queueing delay due to customers of higher priority that arrive while a customer is waiting in queue. This is the meaning of the last term in the formula

$$W_2 = R + \frac{1}{\mu_1} N_Q^1 + \frac{1}{\mu_2} N_Q^2 + \frac{1}{\mu_1} \lambda_1 W_2$$

Using Little's Theorem ($N_Q^k = \lambda_k W_k$), we obtain

$$W_2 = R + \rho_1 W_1 + \rho_2 W_2 + \rho_1 W_2$$

which yields

$$W_2 = \frac{R + \rho_1 W_1}{1 - \rho_1 - \rho_2}$$

Using the expression $W_1 = R/(1 - \rho_1)$ obtained earlier, we finally have

$$W_2 = \frac{R}{(1 - \rho_1)(1 - \rho_1 - \rho_2)}$$

The derivation is similar for all priority classes $k > 1$. The formula for the waiting time in queue is

$$W_k = \frac{R}{(1 - \rho_1 - \cdots - \rho_{k-1})(1 - \rho_1 - \cdots - \rho_k)} \tag{3.79}$$

The average delay per customer of class k is

$$T_k = \frac{1}{\mu_k} + W_k \tag{3.80}$$

The mean residual service time R can be derived as for the P-K formula (compare with Fig. 3.16). We have

$$R = \frac{1}{2} \sum_{i=1}^{n} \lambda_i \overline{X_i^2} \tag{3.81}$$

The average waiting time in queue and the average delay per customer for each class is obtained by combining Eqs. (3.79) to (3.81):

$$W_k = \frac{\sum_{i=1}^{n} \lambda_i \overline{X_i^2}}{2(1 - \rho_1 - \cdots - \rho_{k-1})(1 - \rho_1 - \cdots - \rho_k)} \tag{3.82}$$

$$T_k = \frac{1}{\mu_k} + W_k \tag{3.83}$$

The analysis given above does not extend easily to the case of multiple servers, primarily because there is no simple formula for the mean residual time R. If, however,

the service times of all priority classes are identically and exponentially distributed, there is a convenient characterization of R. Equation (3.79) then yields a closed-form expression for the average waiting times W_k (see Problem 3.38).

Note that it is possible to affect the average delay per customer by choosing the priority classes appropriately. It is generally true that average delay tends to be reduced when customers with short service times are given higher priority. (For an example from common experience, consider the supermarket practice of having special checkout counters for customers with few items. A similar situation can be seen in copying machine waiting lines, where people often give priority to others who need to make just a few copies.) For an analytical substantiation, consider a nonpreemptive system and two customer classes A and B, with respective arrival and service rates λ_A, μ_A, and λ_B, μ_B. A straightforward calculation using the formulas above shows that if $\mu_A > \mu_B$, then the average delay per customer (averaged over both classes)

$$T = \frac{\lambda_A T_A + \lambda_B T_B}{\lambda_A + \lambda_B}$$

is smaller when A is given priority over B than when B is given priority over A. For related results, see Problem 3.40.

Preemptive resume priority. One of the features of the nonpreemptive priority rule is that the average delay of a priority class depends on the arrival rate of lower-priority classes. This is evident from Eq. (3.82), which gives the average waiting times W_k, and is due to the fact that a high-priority customer must wait for a lower-priority customer already in service. This dependence is not present in the *preemptive resume priority discipline*, whereby service of a customer is interrupted when a higher-priority customer arrives and is resumed from the point of interruption once all customers of higher priority have been served.

As an example of an (approximation to) such a system, consider a transmission line serving several Poisson packet streams of different priorities. The packets of each stream are subdivided into many small "subpackets" (*e.g.*, ATM cells), which in the absence of packets of higher priority, are contiguously transmitted on the line. The transmission of the subpackets of a given packet is halted when a packet of higher priority arrives and is resumed when no subpackets of higher priority packets are left in the system.

As we consider the calculation of T_k, the average time in the system of priority k customers, we should keep in mind that the presence of customers of priorities $k+1$ through n does not affect this calculation. Therefore, we can treat each priority class as if it were the lowest in the system. The system time T_k consists of three terms:

1. The customer's average service time $1/\mu_k$.
2. The average time required, upon arrival of a priority k customer, to service customers of priority 1 to k already in the system (*i.e.*, the average unfinished work corresponding to priorities 1 through k). It can be seen that this time is equal to the average waiting time in the corresponding, ordinary $M/G/1$ system (without priorities), where the customers of priorities $k+1$ through n are neglected, that is

[cf. Eq. (3.48)],

$$\frac{R_k}{1-\rho_1-\cdots-\rho_k}$$

where R_k is the mean residual time

$$R_k = \frac{\sum_{i=1}^{k} \lambda_i \overline{X_i^2}}{2} \tag{3.84}$$

The reason is that at all times, the unfinished work (sum of remaining service times of all customers in the system) of an $M/G/1$-type system is independent of the priority discipline of the system. This is true for any system where the server is always busy while the system is nonempty, and customers leave the system only after receiving their required service. (An example of a system that does not have this property is the vacation system of Section 3.5.1.)

3. The average waiting time for customers of priorities 1 through $k-1$ who arrive while the customer of class k is in the system. This term is

$$\sum_{i=1}^{k-1} \frac{1}{\mu_i}\lambda_i T_k = \sum_{i=1}^{k-1} \rho_i T_k$$

for $k > 1$, and is zero for $k = 1$.

Collecting the three terms above, we obtain the equation

$$T_k = \frac{1}{\mu_k} + \frac{R_k}{1-\rho_1-\cdots-\rho_k} + \left(\sum_{i=1}^{k-1} \rho_i\right) T_k \tag{3.85}$$

The final result is, for $k = 1$,

$$T_1 = \frac{(1/\mu_1)(1-\rho_1)+R_1}{1-\rho_1} \tag{3.86}$$

and for $k > 1$,

$$T_k = \frac{(1/\mu_k)(1-\rho_1-\cdots-\rho_k)+R_k}{(1-\rho_1-\cdots-\rho_{k-1})(1-\rho_1-\cdots-\rho_k)} \tag{3.87}$$

where R_k is given by Eq. (3.84). As for the nonpreemptive system, there is no easy extension of this formula to the case of multiple servers unless the service times of all priority classes are identically and exponentially distributed (see Problem 3.38).

3.5.4 An Upper Bound for the $G/G/1$ System

Consider the $G/G/1$ system, which is the same as $M/G/1$ except that the interarrival times have a general rather than exponential distribution. We continue to assume that the interarrival times and service times are all independent. We want to show that the average waiting time in queue satisfies

$$W \le \frac{\lambda(\sigma_a^2+\sigma_b^2)}{2(1-\rho)} \tag{3.88}$$

where

σ_a^2 = Variance of the interarrival times

σ_b^2 = Variance of the service times

λ = Average interarrival time

ρ = Utilization factor λ/μ, where $1/\mu$ is the average service time

The upper bound (3.88) becomes exact asymptotically as $\rho \to 1$, that is, as the system becomes heavily loaded.

Let us denote

W_k = Waiting time of the k^{th} customer

X_k = Service time of the k^{th} customer

τ_k = Interarrival time between the k^{th} and $(k+1)^{\text{st}}$ customer

From Fig. 3.24 we see that

$$W_{k+1} = \max\{0, W_k + X_k - \tau_k\} \tag{3.89}$$

To simplify the analysis, we will use the following notation for any random variable Y:

$$Y^+ = \max\{0, Y\}, \qquad Y^- = -\min\{0, Y\}$$

$$\overline{Y} = E\{Y\}, \qquad \sigma_Y^2 = E\{Y^2 - \overline{Y}^2\}$$

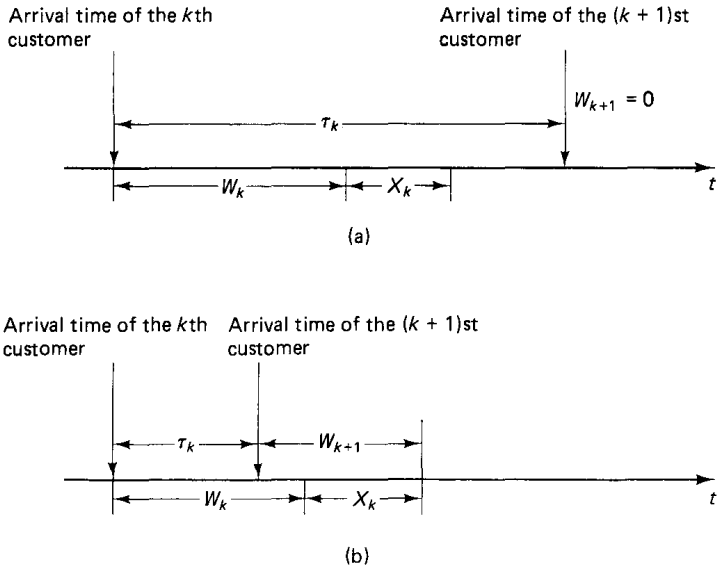


Figure 3.24 Expressing the waiting time W_{k+1} of the $(k+1)^{\text{st}}$ customer in terms of the waiting time W_k, the service time X_k, and the interarrival time τ_k of the k^{th} customer. If the k^{th} customer has departed before the $(k+1)^{\text{st}}$ customer's arrival, which is equivalent to $W_k + X_k - \tau_k \leq 0$, then $W_{k+1} = 0$ [case (a)]. Otherwise, we have $W_{k+1} = W_k + X_k - \tau_k$ [case (b)].

Note that we have

$$Y = Y^+ - Y^-, \qquad Y^+ \cdot Y^- = 0$$

from which we see that

$$\overline{Y} = \overline{Y^+} - \overline{Y^-}, \qquad \sigma_Y^2 = \sigma_{Y^+}^2 + \sigma_{Y^-}^2 + 2\overline{Y^+} \cdot \overline{Y^-} \tag{3.90}$$

Let us now write the expression (3.89) as

$$W_{k+1} = (W_k + V_k)^+ \tag{3.91}$$

where

$$V_k = X_k - \tau_k \tag{3.92}$$

Let us also denote

$$I_k = (W_k + V_k)^- \tag{3.93}$$

From Fig. 3.24 we see that I_k is the length of the idle period between the arrival of the k^{th} and the arrival of the $(k+1)^{\text{st}}$ customer.

We have, using Eq. (3.90),

$$\sigma_{(W_k+V_k)}^2 = \sigma_{(W_k+V_k)^+}^2 + \sigma_{(W_k+V_k)^-}^2 + 2\overline{(W_k+V_k)^+} \cdot \overline{(W_k+V_k)^-} \tag{3.94}$$

Since W_k and V_k are independent, we also have

$$\sigma_{(W_k+V_k)}^2 = \sigma_{W_k}^2 + \sigma_{V_k}^2 = \sigma_{W_k}^2 + \sigma_a^2 + \sigma_b^2 \tag{3.95}$$

Combining Eqs. (3.91) to (3.95), we obtain

$$\sigma_{W_k}^2 + \sigma_a^2 + \sigma_b^2 = \sigma_{W_{k+1}}^2 + \sigma_{I_k}^2 + 2\overline{W}_{k+1}\overline{I}_k$$

We now take the limit as $k \to \infty$, assuming that steady-state values exist, that is,

$$\overline{W}_k \to W, \qquad \sigma_{W_k}^2 \to \sigma_W^2, \qquad \overline{I}_k \to I, \qquad \sigma_{I_k}^2 \to \sigma_I^2$$

We obtain

$$W = \frac{\sigma_a^2 + \sigma_b^2}{2I} - \frac{\sigma_I^2}{2I} \tag{3.96}$$

The average idle time I between two successive arrivals is equal to $(1-\rho)$ (the fraction of time the system is idle) multiplied by the average interarrival time $1/\lambda$, that is, $I = (1-\rho)/\lambda$. Thus, we can write Eq. (3.96) as

$$W = \frac{\lambda(\sigma_a^2 + \sigma_b^2)}{2(1-\rho)} - \frac{\lambda\sigma_I^2}{2(1-\rho)} \tag{3.97}$$

Since $\sigma_I^2 \geq 0$, we obtain the inequality

$$W \leq \frac{\lambda(\sigma_a^2 + \sigma_b^2)}{2(1-\rho)} \tag{3.98}$$

Samsung Exhibit 1015, Page 224 of 573

which is the desired result. Note that as the system becomes more heavily loaded, the average idle time I_k tends to diminish and so does the variance σ_I^2, thereby making the upper bound increasingly accurate.

As an example, consider the $M/G/1$ queue. By the Pollatcheck–Khinchin formula we have

$$W = \frac{\lambda \overline{X^2}}{2(1-\rho)} = \frac{\lambda(\sigma_b^2 + 1/\mu^2)}{2(1-\rho)} \tag{3.99}$$

Since for the Poisson arrival process with rate λ we have $\sigma_a^2 = 1/\lambda^2$, by comparing Eqs. (3.97) and (3.99) we see that

$$\sigma_I^2 = \frac{1}{\lambda^2} - \frac{1}{\mu^2}$$

Thus the term that was neglected to derive the upper bound (3.98) for W is equal to

$$\frac{\lambda \sigma_I^2}{2(1-\rho)} = \frac{\lambda}{2(1-\lambda/\mu)} \left(\frac{1}{\lambda^2} - \frac{1}{\mu^2} \right) = \frac{1}{2} \left(\frac{1}{\lambda} + \frac{1}{\mu} \right)$$

This term is always less than the average interarrival time $1/\lambda$ when $\rho < 1$. As $\rho \to 1$, it approaches $1/\mu$ and is negligible relative to the upper bound of Eq. (3.98).

We finally note that several other bounds and approximations for the $G/G/1$ queue have been obtained. A particularly simple improvement to the one we have given here is

$$W \leq \frac{\lambda(\sigma_a^2 + \sigma_b^2)}{2(1-\rho)} - \frac{\lambda(1-\rho)\sigma_a^2}{2}$$

Its derivation is outlined in Problem 3.48.

3.6 NETWORKS OF TRANSMISSION LINES

In a data network, there are many transmission queues that interact in the sense that a traffic stream departing from one queue enters one or more other queues, perhaps after merging with portions of other traffic streams departing from yet other queues. Analytically, this has the unfortunate effect of complicating the character of the arrival processes at downstream queues. The difficulty is that the packet interarrival times become strongly correlated with packet lengths once packets have traveled beyond their entry queue. As a result it is impossible to carry out a precise and effective analysis comparable to the one for the $M/M/1$ and $M/G/1$ systems.

As an illustration of the phenomena that complicate the analysis, consider two transmission lines of equal capacity in tandem, as shown in Fig. 3.25. Assume that Poisson arrivals of rate λ packets/sec enter the first queue, and that all packets have *equal* length. Therefore, the first queue is $M/D/1$ and the average packet delay there is given by the Pollaczek–Khinchin formula. However, at the second queue the interarrival times must be greater than or equal to $1/\mu$ (the packet transmission time). Furthermore, because the packet transmission times are equal at both queues, each packet arriving at

Samsung Exhibit 1015, Page 225 of 573

the second queue will complete transmission at or before the time the next packet arrives, so there is *no waiting at the second queue*. Therefore, a delay model based on Poisson assumptions is totally inappropriate for the second queue.

Consider next the case of the two tandem transmission lines where packet lengths are exponentially distributed and are independent of each other as well as of the interarrival times at the first queue. Then the first queue is $M/M/1$. The second queue, however, *cannot* be modeled as $M/M/1$. The reason is, again, that *the interarrival times at the second queue are strongly correlated with the packet lengths*. In particular, the interarrival time of two packets at the second queue is greater than or equal to the transmission time of the second packet at the first queue (see Fig. 3.26). As a result, long packets will typically wait less time at the second queue than short packets, since their transmission at the first queue takes longer, thereby giving the second queue more time to empty out. For a traffic analogy, consider a slow truck traveling on a busy narrow street together with several faster cars. The truck will typically see empty space ahead of it while being closely followed by the faster cars.

As an indication of the difficulty of analyzing queueing network problems involving dependent interarrival and service times, no analytical solution is known for even the simple tandem queueing problem of Fig. 3.25 involving Poisson arrivals and exponentially distributed service times. In the real situation where packet lengths and interarrival times are correlated, a simulation has shown that under heavy traffic conditions, average delay per packet is smaller than in the idealized situation where there is no such correlation. The reverse is true under light traffic conditions. It is not known whether and in what form this result can be extended to more general networks.



Figure 3.25 Two equal-capacity transmission lines in tandem. If all packets have equal length, there is no queueing delay in the second queue.

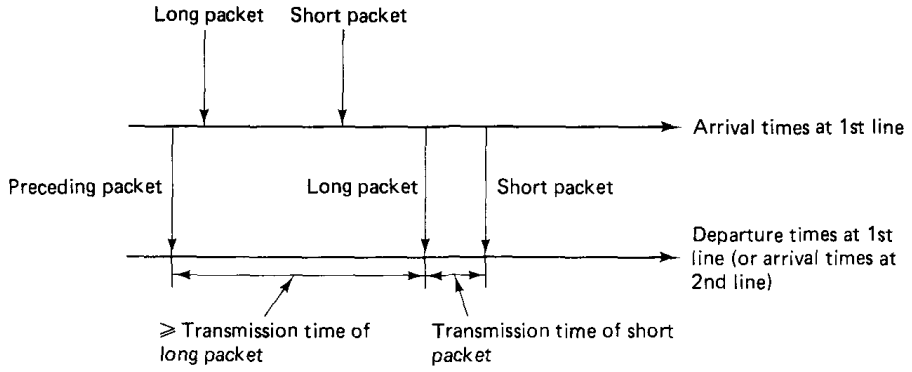


Figure 3.26 Timing diagram of packet arrivals and departures completions in a system of two transmission lines in tandem. The interarrival time of two packets at the second queue is greater or equal to the transmission time of the second packet. (It is greater if and only if the second packet finds the first queue empty upon arrival.) Hence the interarrival times at the second queue are correlated with the packet lengths.

3.6.1 The Kleinrock Independence Approximation

We now formulate a framework for approximation of average delay per packet in data networks. Consider a network of communication links as shown in Fig. 3.27. Assume that there are several packet streams, each following a unique path that consists of a sequence of links through the network. Let x_s, in packets/sec, be the arrival rate of the packet stream s. Then the total arrival rate at link (i, j) is

$$\lambda_{ij} = \sum_{\substack{\text{all packet streams } s \\ \text{crossing link } (i,j)}} x_s$$

The preceding network model is well suited for virtual circuit networks, with each packet stream modeling a separate virtual circuit. For datagram networks, it is sometimes necessary to use a more general model that allows bifurcation of the traffic of a packet stream. Here there are again several packet streams, each having a unique origin and destination. However, there may be several paths followed by the packets of a stream (see Fig. 3.28). Assume that no packets travel in a loop, let x_s denote the arrival rate of packet stream s, and let $f_{ij}(s)$ denote the fraction of the packets of stream s that go through link (i, j). Then the total arrival rate at link (i, j) is

$$\lambda_{ij} = \sum_{\substack{\text{all packet streams } s \\ \text{crossing link } (i,j)}} f_{ij}(s) x_s$$

We have seen from the special case of two tandem queues that even if the packet streams are Poisson with independent packet lengths at their point of entry into the network, this property is lost after the first transmission line. To resolve the dilemma, it was suggested by Kleinrock [Kle64] that merging several packet streams on a transmission line has an effect akin to restoring the independence of interarrival times and packet lengths. For example, if the second transmission line in the preceding tandem queue case were to receive a substantial amount of additional external Poisson traffic,

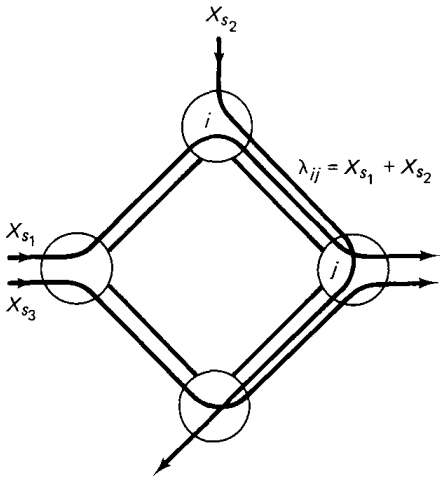


Figure 3.27 Model suitable for virtual circuit networks. There are several packet streams, each using a single path. The total arrival rate λ_{ij} at a link (i, j) is equal to the sum of the arrival rates x_s of all packet streams s traversing the link.

Samsung Exhibit 1015, Page 227 of 573

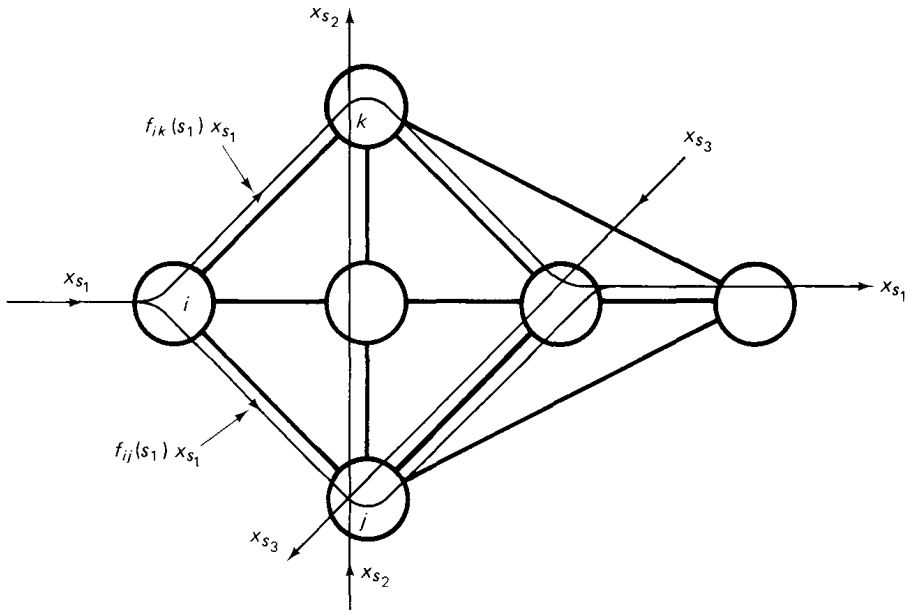


Figure 3.28 Model suitable for datagram networks. There are several packet streams, each associated with a unique origin–destination pair. However, packets of the same stream may follow one of several paths. The total arrival rate λ_{ij} at a link (i, j) is equal to the sum of the fractions $f_{ij}(s)x_s$ of the arrival rates of all packet streams s traversing the link.

the dependence of interarrival and service times displayed in Fig. 3.26 would be weakened considerably. It was concluded that it is often appropriate to adopt an $M/M/1$ queueing model for each communication link regardless of the interaction of traffic on this link with traffic on other links. (See also the discussion preceding Jackson's theorem in Section 3.8.) This is known as the *Kleinrock independence approximation* and seems to be a reasonably good approximation for systems involving Poisson stream arrivals at the entry points, packet lengths that are nearly exponentially distributed, a densely connected network, and moderate-to-heavy traffic loads. Based on this $M/M/1$ model, the average number of packets in queue or service at (i, j) is

$$N_{ij} = \frac{\lambda_{ij}}{\mu_{ij} - \lambda_{ij}} \tag{3.100}$$

where $1/\mu_{ij}$ is the average packet transmission time on link (i, j). The average number of packets summed over all queues is

$$N = \sum_{(i,j)} \frac{\lambda_{ij}}{\mu_{ij} - \lambda_{ij}} \tag{3.101}$$

so by Little's Theorem, the average delay per packet (neglecting processing and propagation delays) is

$$T = \frac{1}{\gamma} \sum_{(i,j)} \frac{\lambda_{ij}}{\mu_{ij} - \lambda_{ij}} \tag{3.102}$$

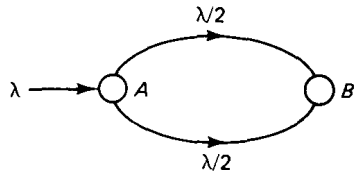


Figure 3.29 Poisson process with rate λ divided among two links. If division is done by randomization, each link behaves like an $M/M/1$ queue. If division is done by metering, the whole system behaves like an $M/M/2$ queue.

where $\gamma = \sum_s x_s$ is the total arrival rate in the system. If the average processing and propagation delay d_{ij} at link (i,j) is not negligible, this formula should be adjusted to

$$T = \frac{1}{\gamma} \sum_{(i,j)} \left(\frac{\lambda_{ij}}{\mu_{ij} - \lambda_{ij}} + \lambda_{ij} d_{ij} \right) \tag{3.103}$$

Finally, the average delay per packet of a traffic stream traversing a path p is given by

$$T_p = \sum_{\substack{\text{all } (i,j) \\ \text{on path } p}} \left(\frac{\lambda_{ij}}{\mu_{ij}(\mu_{ij} - \lambda_{ij})} + \frac{1}{\mu_{ij}} + d_{ij} \right) \tag{3.104}$$

where the three terms in the sum above represent average waiting time in queue, average transmission time, and processing and propagation delay, respectively.

In many networks, the assumption of exponentially distributed packet lengths is not appropriate. Given a different type of probability distribution of the packet lengths, one may keep the approximation of independence between queues but use the P-K formula for average number in the system in place of the $M/M/1$ formula (3.100). Equations (3.101) to (3.104) for average delay would then be modified in an obvious way.

For virtual circuit networks (cf. Fig. 3.27), the main approximation involved in the $M/M/1$ formula (3.101) is due to the correlation of the packet lengths and the packet interarrival times at the various queues in the network. If somehow this correlation was not present (*e.g.*, if a packet upon departure from a transmission line was assigned a new length drawn from an exponential distribution), then the average number of packets in the system would be given indeed by the formula

$$N = \sum_{(i,j)} \frac{\lambda_{ij}}{\mu_{ij} - \lambda_{ij}}$$

This fact (by no means obvious) is a consequence of Jackson's Theorem, which will be discussed in Section 3.8.

In datagram networks that involve multiple path routing for some origin–destination pairs (cf. Fig. 3.28), the accuracy of the $M/M/1$ approximation deteriorates for another reason, which is best illustrated by an example.

Example 3.17

Suppose that node A sends traffic to node B along two links with service rate μ in the network of Fig. 3.29. Packets arrive at A according to a Poisson process with rate λ packets/sec. Packet transmission times are exponentially distributed and independent of interarrival times as in the $M/M/1$ system. Assume that the arriving traffic is to be divided equally among the two links. However, how should this division be implemented? Consider the following possibilities.

1. *Randomization.* Here each packet is assigned upon arrival at A to one of the two links based on the outcome of a fair coin flip. It is then possible to show that the arrival process on each of the two queues is Poisson and independent of the packet lengths (see Problem 3.11). Therefore, each of the two queues behaves like an $M/M/1$ queue with arrival rate $\lambda/2$ and average delay per packet

$$T_R = \frac{1}{\mu - \lambda/2} = \frac{2}{2\mu - \lambda} \tag{3.105}$$

which is consistent with the Kleinrock independence approximation.

2. *Metering.* Here each arriving packet is assigned to a queue that currently has the smallest total backlog in bits and will therefore empty out first. An equivalent system maintains a common queue for the two links and routes the packet at the head of the queue to the link that becomes idle first. This works like an $M/M/2$ system with arrival rate λ and with each link playing the role of a server. Using the result of Section 3.4.1, the average delay per packet can be calculated to be

$$T_M = \frac{2}{(2\mu - \lambda)(1 + \rho)} \tag{3.106}$$

where $\rho = \lambda/2\mu$.

Comparing the average delay expressions (3.105) and (3.106), we see that metering performs better than randomization in terms of delay by a factor $1/(1+\rho)$. This is basically the same advantage that statistical multiplexing with multiple channels holds over time-division multiplexing as discussed in Example 3.10. Generally, it is preferable to use some form of metering rather than randomization when dividing traffic among alternative routes. However, in contrast with randomization, metering destroys the Poisson character of the arrival process at the point of division. In our example, when metering is used, the interarrival times at each link are neither exponentially distributed nor independent of preceding packet lengths. Therefore, the use of metering (which is recommended for performance reasons) tends to degrade the accuracy of the $M/M/1$ approximation.

We finally mention an alternative approach for approximating average delay in a network of transmission lines. This approach uses $G/G/1$ approximations in place of $M/M/1$ or $M/G/1$ approximations. The key idea is that given the first two moments of the interarrival and service times of each of the external packet streams, one may approximate reasonably well the first two moments of the interarrival and service times of the total packet arrival stream at each queue (see [Whi83a], [Whi83b], and the references quoted there). Then the average delay at each queue can be estimated using $G/G/1$ bounds and approximations of the type discussed in Section 3.5.4.

3.7 TIME REVERSIBILITY—BURKE'S THEOREM

The analysis of the $M/M/1$, $M/M/m$, $M/M/\infty$, and $M/M/m/m$ systems was based on the equality of the steady-state frequency of transitions from j to $j+1$, that is, $p_j P_{j(j+1)}$, with the steady-state frequency of transitions from $j+1$ to j, that is, $p_{j+1} P_{(j+1)j}$. These relations, called *detailed balance equations*, are valid for any Markov

chain with integer states in which transitions can occur only between neighboring states (*i.e.*, from j to $j-1$, j, or $j+1$); these Markov chains are called *birth–death* processes. The detailed balance equations lead to an important property called time reversibility, as we now explain.

Consider an irreducible, aperiodic, discrete-time Markov chain $X_n, X_{n+1}, \ldots$ having transition probabilities P_{ij} and stationary distribution $\{p_j \mid j \geq 0\}$ with $p_j > 0$ for all j. Suppose that the chain is in steady-state, that is,

$$P\{X_n = j\} = p_j, \qquad \text{for all } n$$

(This occurs if the initial state is chosen according to the stationary distribution, and is equivalent to imagining that the process began at time $-\infty$.)

Suppose that we trace the sequence of states going backward in time. That is, starting at some n, consider the sequence of states $X_n, X_{n-1}, \ldots$. This sequence is itself a Markov chain, as seen by the following calculation:

$$\begin{aligned}
&P\{X_m = j \mid X_{m+1} = i,\ X_{m+2} = i_2, \ldots, X_{m+k} = i_k\} \\
&= \frac{P\{X_m = j, X_{m+1} = i,\ X_{m+2} = i_2, \ldots, X_{m+k} = i_k\}}{P\{X_{m+1} = i,\ X_{m+2} = i_2, \ldots, X_{m+k} = i_k\}} \\
&= \frac{P\{X_m = j, X_{m+1} = i\}\, P\{X_{m+2} = i_2, \ldots, X_{m+k} = i_k \mid X_m = j, X_{m+1} = i\}}{P\{X_{m+1} = i\} P\{X_{m+2} = i_2, \ldots, X_{m+k} = i_k \mid X_{m+1} = i\}} \\
&= \frac{P\{X_m = j, X_{m+1} = i\}}{P\{X_{m+1} = i\}} \\
&= \frac{P\{X_m = j\}\, P\{X_{m+1} = i \mid X_m = j\}}{P\{X_{m+1} = i\}} \\
&= \frac{p_j P_{ji}}{p_i}
\end{aligned}$$

where the third equality follows from the Markov property of the chain $X_n, X_{n+1}, \ldots$ Thus, conditional on the state at time $m+1$, the state at time m is independent of that at times $m+2, m+3, \ldots$. The backward transition probabilities are given by

$$P^*_{ij} = P\{X_m = j \mid X_{m+1} = i\} = \frac{p_j P_{ji}}{p_i}, \qquad i, j \geq 0 \tag{3.107}$$

If $P^*_{ij} = P_{ij}$ for all i, j (*i.e.*, the transition probabilities of the forward and reversed chain are identical), we say that the chain is *time reversible.*

We list some properties of the reversed chain:

1. The reversed chain is irreducible, aperiodic, and has the same stationary distribution as the forward chain. [This property can be shown either by elementary reasoning using the definition of the reversed chain, or by verifying the equality $p_j = \sum_{i=0}^{\infty} p_i P^*_{ij}$ using Eq. (3.107).] The intuitive idea here is that the reversed chain corresponds to the same process, looked at in the reversed time direction. Thus, if the steady-state probabilities are viewed as proportions of time the process

Samsung Exhibit 1015, Page 231 of 573

visits the states, then the steady-state occupancy distributions of the forward and the reverse chains are equal. Note that in view of this equality, the form of the transition probabilities of the reversed chain $P^*_{ij} = p_j P_{ji}/p_i$ [cf. Eq. (3.107)] can be intuitively explained. It expresses the fact that (with probability 1) the proportion of transitions from j to i out of all transitions in the forward chain (which is $p_j P_{ji}$) equals the proportion of transitions from i to j out of all transitions in the reversed chain (which is $p_i P^*_{ij}$).

2. If we can find positive numbers p_i, $i \geq 0$, summing to unity and such that the scalars

$$P^*_{ij} = \frac{p_j P_{ji}}{p_i}, \qquad i, j \geq 0 \tag{3.108}$$

form a transition probability matrix, that is,

$$\sum_{j=0}^{\infty} P^*_{ij} = 1, \qquad i = 0, 1, \ldots$$

then $\{p_i \mid i \geq 0\}$ is the stationary distribution and P^*_{ij} are the transition probabilities of the reversed chain. [To see this, note that by multiplying with p_i Eq. (3.108) and adding over j, we obtain

$$\sum_{j=0}^{\infty} p_j P_{ji} = p_i \sum_{j=0}^{\infty} P^*_{ij} = p_i$$

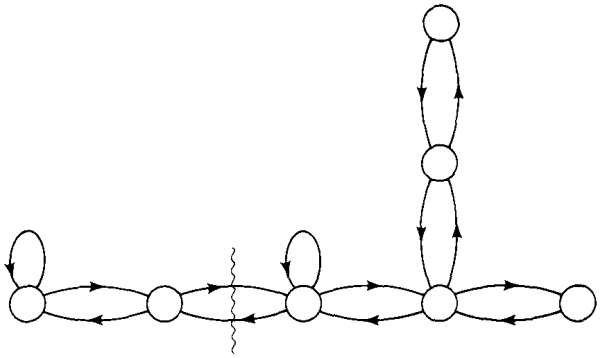
which is the global balance equation and implies that $\{p_i \mid i \geq 0\}$ is the stationary distribution.] This property, which holds regardless of whether the chain is time reversible, is useful if through an intelligent guess, we can verify Eq. (3.108), thereby obtaining both the p_j and P^*_{ij}; for examples of such applications, see Section 3.8.

3. A chain is time reversible if and only if the detailed balance equations hold:

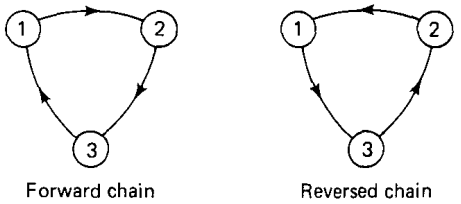
$$p_i P_{ij} = p_j P_{ji}, \qquad i, j \geq 0$$

This follows from the equality $p_i P^*_{ij} = p_j P_{ji}$ [cf. Eq. (3.107)] and the definition of time reversibility. In other words, a system is time reversible if in a typical system history, transitions from i to j occur with the same frequency as transitions from j to i (and therefore also with the same frequency as transitions from i to j when this system history is reversed in time). In particular, the chains corresponding to the queueing systems $M/M/1$, $M/M/m$, $M/M/\infty$, and $M/M/m/m$ discussed in Sections 3.3 and 3.4 are time reversible (in the limit as $\delta \to 0$). More generally, chains corresponding to birth–death processes ($P_{ij} = 0$ if $|i - j| > 1$) are time reversible. Figure 3.30 gives some additional examples of reversible and nonreversible systems.

The idea of time reversibility extends in a straightforward manner to irreducible continuous-time Markov chains. The corresponding analysis can be carried out either directly or by discretizing time in intervals of length δ, considering the corresponding



(a)



(b)

Figure 3.30 (a) Example of a time reversible chain. To see this, note that by splitting the state space in two subsets as shown we obtain global balance equations which are identical with the detailed balance equations. (b) Example of a chain which is not time reversible. The states in the forward and the reversed systems move in the clockwise and counterclockwise directions, respectively.

discrete-time chain, and passing back to the continuous chain by taking the limit as $\delta \to 0$. All results regarding the reversed chain carry over almost verbatim from their discrete-time counterparts by replacing transition probabilities with transition rates. In particular, if the continuous-time chain has transition rates q_{ij} and a stationary distribution $\{p_j \mid j \geq 0\}$ with $p_j > 0$ for all j, then:

1. The reversed chain is a continuous-time Markov chain with the same stationary distribution as the forward chain and with transition rates

$$q_{ij}^* = \frac{p_j q_{ji}}{p_i}, \qquad i, j \geq 0 \tag{3.109}$$

2. If we can find positive numbers p_i, $i \geq 0$, summing to unity and such that the scalars

$$q_{ij}^* = \frac{p_j q_{ji}}{p_i}, \qquad i, j \geq 0 \tag{3.110}$$

satisfy for all $i \geq 0$

$$\sum_{j=0}^{\infty} q_{ij} = \sum_{j=0}^{\infty} q_{ij}^* \tag{3.111}$$

then $\{p_i \mid i \geq 0\}$ is the stationary distribution of both the forward and the reversed chain, and q_{ij}^* are the transition rates of the reversed chain. The relation $\sum_{j=0}^{\infty} q_{ij} = \sum_{j=0}^{\infty} q_{ij}^*$ equates, for every state i, the total rate out of i in the forward and the reversed chains, and by taking into account also the relation $q_{ij}^* = p_j q_{ji}/p_i$, it can

be seen to be equivalent to the global balance equation

$$p_i \sum_{j=0}^{\infty} q_{ij} = \sum_{j=0}^{\infty} p_j q_{ji}$$

[cf. Eq. (3A.10) of Appendix A].

3. The forward chain is time reversible if and only if its stationary distribution and transition rates satisfy the detailed balanced equations

$$p_i q_{ij} = p_j q_{ji}\,, \qquad i, j \geq 0$$

Consider now the $M/M/1$, $M/M/m$, and $M/M/\infty$ queueing systems. We assume that the initial state is chosen according to the stationary distribution so that the queueing systems are in steady-state at all times. The reversed process can be represented by another queueing system where departures correspond to arrivals of the original system and arrivals correspond to departures of the original system (see Fig. 3.31). Because time reversibility holds for all these systems as discussed above, the forward and reversed systems are statistically indistinguishable in steady-state. In particular by using the fact that the departure process of the forward system corresponds to the arrival process of the reversed system, we obtain the following result:

Burke's Theorem. Consider an $M/M/1$, $M/M/m$, or $M/M/\infty$ system with arrival rate λ. Suppose that the system starts in steady-state. Then the following hold true:

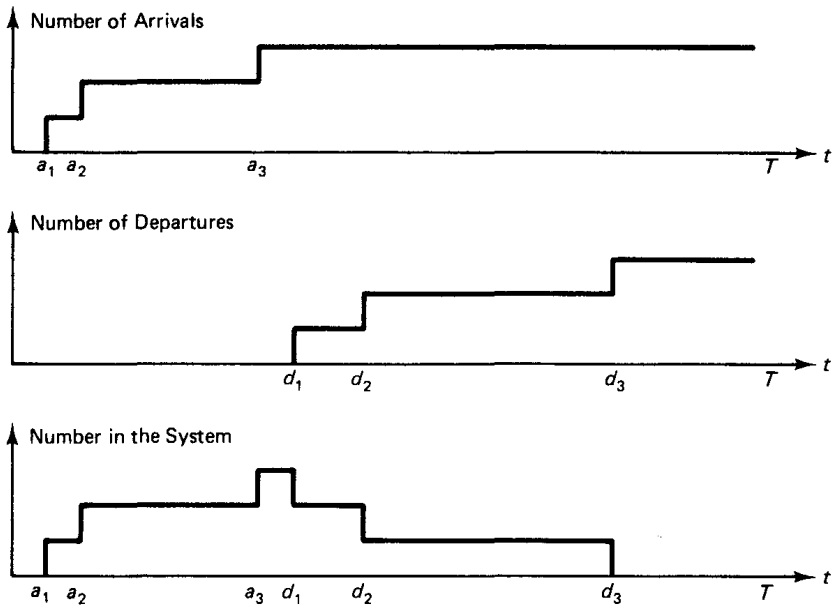
(a) The departure process is Poisson with rate λ.

(b) At each time t, the number of customers in the system is independent of the sequence of departure times prior to t.

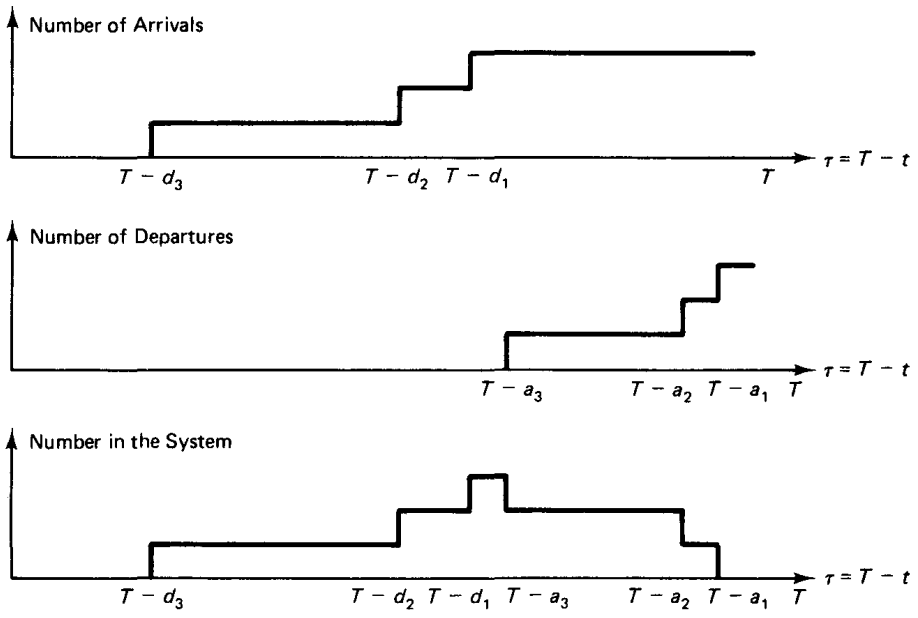
Proof: (a) This follows from the fact that the forward and reversed systems are statistically indistinguishable in steady-state, and the departure process in the forward system is the arrival process in the reversed system.

(b) As shown in Fig. 3.32, for a fixed time t, the departures prior to t in the forward process are also the arrivals after t in the reversed process. The arrival process in the reversed system is independent Poisson, so the future arrival process does not depend on the current number in the system, which in forward system terms means that the past departure process does not depend on the current number in the system. **Q.E.D.**

Note that part (b) of Burke's Theorem is quite counterintuitive. One would expect that a recent stream of closely spaced departures suggests a busy system with an atypically large number of customers in queue. Yet Burke's Theorem shows that this is not so. Note, however, that Burke's Theorem says nothing about the state of the system *before* a stream of closely spaced departures. Such a state would tend to have abnormally many customers in queue, in accordance with intuition.



(a)



(b)

Figure 3.31 (a) Forward system number of arrivals, number of departures, and occupancy during $[0, T]$. (b) Reversed system number of arrivals, number of departures, and occupancy during $[0, T]$.

Samsung Exhibit 1015, Page 235 of 573

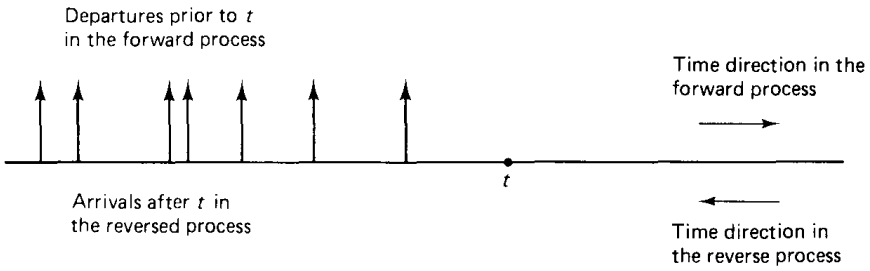


Figure 3.32 Customer departures *prior* to time t in the forward system become customer arrivals *after* time t in the reversed system.

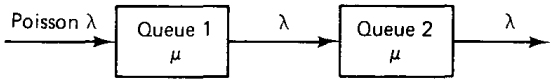


Figure 3.33 Two queues in tandem. The service times at the two queues are exponentially distributed and mutually independent. Using Burke's Theorem, we can show that the number of customers in queues 1 and 2 are independent at a given time and

$$P\{n \text{ at queue 1, } m \text{ at queue 2}\} = \rho_1^n(1-\rho_1)\rho_2^m(1-\rho_2)$$

that is, the two queues behave as if they are independent $M/M/1$ queues in isolation.

Example 3.18 Two $M/M/1$ Queues in Tandem

Consider a queueing network involving Poisson arrivals and two queues in tandem with exponential service times (see Fig. 3.33). There is a major difference between this system and the one discussed in Section 3.6 in that here we assume that the service times of a customer at the first and second queues are mutually independent as well as independent of the arrival process. As a result of this assumption, we will see that the occupancy distribution in the two queues is the same as if they were independent $M/M/1$ queues in isolation. This fact will also be shown in a more general context in the next section.

Let the rate of the Poisson arrival process be λ, and let the mean service times at queues 1 and 2 be $1/\mu_1$ and $1/\mu_2$, respectively. Let $\rho_1 = \lambda/\mu_1$ and $\rho_2 = \lambda/\mu_2$ be the corresponding utilization factors, and assume that $\rho_1 < 1$ and $\rho_2 < 1$. We will show that under steady-state conditions the number of customers at queue 1 and at queue 2 at any given time are independent. Furthermore,

$$P\{n \text{ at queue 1, } m \text{ at queue 2}\} = \rho_1^n(1-\rho_1)\rho_2^m(1-\rho_2) \tag{3.112}$$

To prove this we first note that queue 1 is an $M/M/1$ queue, so by part (a) of Burke's Theorem, the departure process from queue 1 is Poisson. By assumption, it is also independent of the service times at queue 2. Therefore, queue 2, viewed in isolation, is an $M/M/1$ queue. Thus, from the results of Section 3.1,

$$\begin{aligned} P\{n \text{ at queue 1}\} &= \rho_1^n(1-\rho_1) \\ P\{m \text{ at queue 2}\} &= \rho_2^m(1-\rho_2) \end{aligned} \tag{3.113}$$

From part (b) of Burke's Theorem it follows that the number of customers presently in queue 1 is independent of the sequence of earlier arrivals at queue 2 and therefore also of

Samsung Exhibit 1015, Page 236 of 573

the number of customers presently in queue 2. This implies that

$$P\{n \text{ at queue 1, } m \text{ at queue 2}\} = P\{n \text{ at queue 1}\} \cdot P\{m \text{ at queue 2}\}$$

and using Eq. (3.113) we obtain the desired product form (3.112).

We note that, by part (a) of Burke's Theorem, the arrival and the departure processes at both queues of the preceding example are Poisson. This fact can be similarly shown for a much broader class of queueing networks with Poisson arrivals and independent, exponentially distributed service times. We call such networks *acyclic* and define them as follows. We say that queue j is a *downstream neighbor* of queue i if there is a positive probability that a departing customer from queue i will next enter queue j. We say that queue j *lies downstream* of queue i if there is a sequence of queues starting from i and ending at j such that each queue after i in the sequence is a downstream neighbor of its predecessor. A queueing network is called acyclic if it is impossible to find two queues i and j such that j lies downstream of i, and i lies downstream of j. Having an acyclic network is essential for the Poisson character of the arrival and departure processes at each queue to be maintained (see Section 3.8). However, the product form (3.112) of the occupancy distribution generalizes in a natural way to networks that are not acyclic, as we show in the next section.

3.8 NETWORKS OF QUEUES—JACKSON'S THEOREM

As discussed in Section 3.6, the main difficulty with analysis of networks of transmission lines is that the packet interarrival times after traversing the first queue are correlated with their lengths. It turns out that if somehow this correlation were eliminated (which is the premise of the Kleinrock independence approximation) and randomization is used to divide traffic among different routes, then the average number of packets in the system can be derived as if each queue in the network were $M/M/1$. This is an important result known as Jackson's Theorem. In this section we derive a simple version of this theorem and some of its extensions.

Consider a network of K first-come first-serve, single-server queues in which customers arrive from outside the network at each queue i in accordance with independent Poisson processes at rate r_i. We allow the possibility that $r_i = 0$, in which case there are no external arrivals at queue i, but we require that $r_i > 0$ for at least one i. Once a customer is served at queue i, it proceeds to join each queue j with probability P_{ij} or to exit the network with probability $1 - \sum_{j=1}^{K} P_{ij}$.

The routing probabilities P_{ij} together with the external input rates r_j can be used to determine the total arrival rate of customers λ_j at each queue j, that is, the sum of r_j and the arrival rate of customers coming from other queues. Calculating λ_j is fairly easy when the network is of the acyclic type discussed at the end of Section 3.7. If there is a positive probability that a customer may visit the same queue twice, a more complex

Samsung Exhibit 1015, Page 237 of 573

computation is necessary, based on the equations

$$\lambda_j = r_j + \sum_{i=1}^{K} \lambda_i P_{ij}, \qquad j = 1, \ldots, K \tag{3.114}$$

These equations represent a linear system in which the rates λ_j, $j = 1, \ldots, K$, constitute a set of K unknowns. To guarantee that they can be solved uniquely to yield λ_j, $j = 1, \ldots, K$ in terms of r_j, P_{ij}, $i, j = 1, \ldots, K$, we make a fairly natural assumption that essentially asserts that each customer will eventually exit the system with probability 1. This assumption is that for every queue i_1, there is a queue i with $1 - \sum_{j=1}^{K} P_{ij} > 0$ and a sequence $i_1, i_2, \ldots, i_k, i$ such that $P_{i_1 i_2} > 0, \ldots, P_{i_k i} > 0$.*

The service times of customers at the j^{th} queue are assumed exponentially distributed with mean $1/\mu_j$ and are assumed mutually independent and independent of the arrival process at the queue. The utilization factor of each queue is denoted

$$\rho_j = \frac{\lambda_j}{\mu_j}, \qquad j = 1, \ldots, K \tag{3.115}$$

and we assume that $\rho_j < 1$ for all j.

In order to model a packet network such as the one considered in Section 3.6 within the framework described above, it is necessary to accept several simplifying conditions in addition to assuming Poisson arrivals and exponentially distributed packet lengths. The first is the independence of packet lengths and interarrival times discussed earlier. The second is relevant to datagram networks, and has to do with the assumption that bifurcation of traffic at a network node can be modeled reasonably well by a randomization process whereby each departing packet from queue i joins queue j with probability P_{ij}—this need not be true, as discussed in Section 3.6. Still a packet network differs from the model of this section because it involves several traffic streams which may have different routing probabilities at each node, and which maintain their identity as they travel along different routes (see the virtual circuit and datagram network models of Figs. 3.27 and 3.28). This difficulty can be partially addressed by using an extension of Jackson's Theorem that applies to a network with multiple classes of customers. Within this more general framework, we can model traffic streams corresponding to different origin–destination pairs as different classes of customers. If all traffic streams have the same average packet length, it turns out that Jackson's Theorem as stated below is valid assuming the simplifying conditions mentioned earlier; see the analysis in the next subsection.

*For a brief explanation aimed at the advanced reader, consider the Markov chain with states $0, 1, \ldots, K$ and transition probabilities from states $i \neq 0$ to states $j \neq 0$ equal to P_{ij}, and transition probabilities to state 0 equal to $P_{00} = 1$, $P_{i0} = 1 - \sum_{j=1}^{K} P_{ij}$ for $i \neq 0$. (Thus state 0 is an absorbing state that corresponds to exit of a customer from the system.) Let P be the $K \times K$ matrix with elements P_{ij}. The sum of the i^{th} row elements of the matrix P^m (P to the m^{th} power) is the probability that the Markov chain has not arrived at state 0 after m transitions starting from state i. Our hypothesis on P_{ij} implies that the chain will eventually (with probability 1) arrive at state 0 regardless of the initial state. It follows that $\lim_{m \to \infty} P^m = 0$, so unity is not an eigenvalue of P. Therefore, $I - P$ is nonsingular, where I is the identity matrix, from which it can be seen that the system of equations (3.114) has a unique solution.

Samsung Exhibit 1015, Page 238 of 573

For an analysis, we view the system as a continuous-time Markov chain in which the state n is the vector $(n_1, n_2, \ldots, n_K)$, where n_i denotes the number of customers at queue i. At a given state

$$n = (n_1, n_2, \ldots, \ldots, n_K)$$

the possible successor states correspond to a single customer arrival and/or departure. In particular, the transition from n to state

$$n(j^+) = (n_1, \ldots, n_{j-1}, n_j + 1, n_{j+1}, \ldots, n_K)$$

corresponding to an external arrival at queue j, has transition rate

$$q_{nn(j^+)} = r_j$$

The transition from n to state

$$n(j^-) = (n_1, \ldots, n_{j-1}, n_j - 1, n_{j+1}, \ldots, n_K)$$

corresponding to a departure from queue j to the outside, has transition rate

$$q_{nn(j^-)} = \mu_j \left(1 - \sum_i P_{ji}\right)$$

The transition from n to state

$$n(i^+, j^-) = (n_1, \ldots, n_{i-1}, n_i + 1, n_{i+1}, \ldots, n_{j-1}, n_j - 1, n_{j+1}, \ldots, n_K)$$

corresponding to a customer moving from queue j to queue i, has transition rate

$$q_{nn(i^+, j^-)} = \mu_j P_{ji}$$

Let $P(n_1, \ldots, n_K)$ denote the stationary distribution of the chain. We have:

Jackson's Theorem. Assuming that $\rho_j < 1$, $j = 1, \ldots, K$, we have for all $n_1, \ldots, n_K \geq 0$,

$$P(n) = P_1(n_1) P_2(n_2) \cdots P_K(n_K) \tag{3.116}$$

where $n = (n_1, \ldots, n_K)$ and

$$P_j(n_j) = \rho_j^{n_j}(1 - \rho_j), \qquad n_j \geq 0 \tag{3.117}$$

Proof: In our proof we will assume that $\lambda_j > 0$ for all j. There is no loss of generality in doing so because every queue j with $\lambda_j = 0$ is empty in steady-state, so we have $P_j(0) = 1$ and $P_j(n_j) = 0$ for $n_j > 0$, and queue j can be ignored in deriving the stationary distribution of Eqs. (3.116) and (3.117). It can be verified that the condition $\lambda_j > 0$ for all j together with the assumption made earlier to guarantee the uniqueness of solution of Eq. (3.114) imply that the Markov chain with states $n = (n_1, \ldots, n_K)$ describing the system is irreducible; we leave the proof of this for the reader. We will use a technique outlined in Section 3.7 whereby we guess at the transition rates of the reversed process and verify that, together with the probability distribution of Eqs. (3.116) and (3.117), they satisfy the total departure rate equation (3.111). (The Markov chain is

Samsung Exhibit 1015, Page 239 of 573

not time reversible here. Nonetheless, the use of the reversed process is both analytically convenient and conceptually useful.)

For any two state vectors n and n', let $q_{nn'}$ be the corresponding transition rate. Jackson's Theorem will be proved if the rates $q^*_{nn'}$ defined for all n, n' by the equation

$$q^*_{nn'} = \frac{P(n')q_{n'n}}{P(n)} \tag{3.118}$$

satisfy, for all n, the total rate equation

$$\sum_m q_{nm} = \sum_m q^*_{nm} \tag{3.119}$$

which as mentioned in Section 3.7, is equivalent to the global balance equations.

For transitions between states n, $n(j^+)$, and $n(j^-)$, we have

$$q_{nn(j^+)} = r_j \tag{3.120}$$

$$q_{nn(j^-)} = \mu_j \left(1 - \sum_i P_{ji}\right) \tag{3.121}$$

The rates $q^*_{nn(j^+)}$ and $q^*_{nn(j^-)}$ are defined by Eqs. (3.118), (3.120), and (3.121). Using the fact $P\big(n(j^+)\big) = \rho_j P(n) = \lambda_j P(n)/\mu_j$ [cf. Eqs. (3.115)–(3.117)], we obtain

$$q^*_{nn(j^+)} = \lambda_j \left(1 - \textstyle\sum_i P_{ji}\right) \tag{3.122}$$

$$q^*_{nn(j^-)} = \frac{\mu_j r_j}{\lambda_j} \tag{3.123}$$

Next consider transitions between states n and $n(i^+, j^-)$ corresponding to a customer moving from queue j to queue i. We have

$$q_{nn(i^+,j^-)} = \mu_j P_{ji} \tag{3.124}$$

and using the fact that $P\big(n(i^+, j^-)\big) = \rho_i P(n)/\rho_j = \lambda_i \mu_j P(n)/(\lambda_j \mu_i)$, we obtain $q^*_{n(i^+,j^-)n}$ from Eq. (3.118) as

$$q^*_{n(i^+,j^-)n} = \frac{\mu_i \lambda_j P_{ji}}{\lambda_i} \tag{3.125}$$

Since for all other types of pairs of state vectors n, n', we have

$$q_{nn'} = 0 \tag{3.126}$$

it follows from Eq. (3.118) that

$$q^*_{n'n} = 0 \tag{3.127}$$

There remains to verify that the rates q_{nm} and q^*_{nm} satisfy the total rate equation $\sum_m q_{nm} = \sum_m q^*_{nm}$. We have for the forward system, using Eqs. (3.120), (3.121), and (3.124),

$$\sum_m q_{nm} = \sum_{j=1}^{K} q_{nn(j^+)} + \sum_{\{(j,i)|n_j>0\}} q_{nn(i^+,j^-)} + \sum_{\{j|n_j>0\}} q_{nn(j^-)}$$

Samsung Exhibit 1015, Page 240 of 573

$$= \sum_{j=1}^{K} r_j + \sum_{\{(j,i)|n_j>0\}} \mu_j P_{ji} + \sum_{\{j|n_j>0\}} \mu_j \left(1 - \sum_{i=1}^{K} P_{ji}\right)$$

$$= \sum_{j=1}^{K} r_j + \sum_{\{j|n_j>0\}} \mu_j \tag{3.128}$$

Similarly, using Eqs. (3.122), (3.123), (3.125), and (3.114), we obtain for the reversed system

$$\begin{aligned}\sum_m q^*_{nm} &= \sum_{j=1}^{K} q^*_{nn(j^+)} + \sum_{\{(j,i)|n_j>0\}} q^*_{nn(i^+,j^-)} + \sum_{\{j|n_j>0\}} q^*_{nn(j^-)} \\ &= \sum_{j=1}^{K} \lambda_j \left(1 - \sum_{i=1}^{K} P_{ji}\right) + \sum_{\{(j,i)|n_j>0\}} \frac{\mu_j \lambda_i P_{ij}}{\lambda_j} + \sum_{\{j|n_j>0\}} \frac{\mu_j r_j}{\lambda_j} \\ &= \sum_{j=1}^{K} \lambda_j \left(1 - \sum_{i=1}^{K} P_{ji}\right) + \sum_{\{j|n_j>0\}} \frac{\mu_j (r_j + \sum_{i=1}^{K} \lambda_i P_{ij})}{\lambda_j} \\ &= \sum_{j=1}^{K} \lambda_j \left(1 - \sum_{i=1}^{K} P_{ji}\right) + \sum_{\{j|n_j>0\}} \mu_j \end{aligned} \tag{3.129}$$

By writing Eq. (3.114) as $r_j = \lambda_j - \sum_{i=1}^{K} \lambda_i P_{ij}$ and adding over $j = 1, \ldots, K$, we obtain

$$\sum_{j=1}^{K} r_j = \sum_{j=1}^{K} \lambda_j \left(1 - \sum_{i=1}^{K} P_{ji}\right) \tag{3.130}$$

By combining the last three equations, we see that the total rate equation $\sum_m q_{nm} = \sum_m q^*_{nm}$ is satisfied. **Q.E.D.**

Note that the transition rates $q^*_{nn'}$ defined by Eqs. (3.122), (3.123), (3.125), and (3.127) are those of the reversed process. It can be seen that the reversed process corresponds to a network of queues where traffic arrives at queue i from outside the network according to a Poisson process with rate $\lambda_i \left(1 - \sum_j P_{ij}\right)$ [cf. Eq. (3.122)]. The routing probability from queue i to queue j in the reversed process is

$$\frac{\lambda_j P_{ji}}{r_i + \sum_k \lambda_k P_{ki}}$$

[cf. Eqs. (3.123) and (3.125)]. This is also the probability that an arriving customer at queue i just departed from queue j in the forward process. Note that the processes of departure out of the forward system are the exogenous arrival processes of the reversed system, which suggests that the processes of departure out of the system are independent

Samsung Exhibit 1015, Page 241 of 573

Poisson. Indeed, this can be proved by observing that the interarrival times in the reversed system are independent and exponentially distributed.

Example 3.19 Computer System with Feedback Loop for I/O

Consider a model of a computer CPU connected to an I/O device as shown in Fig. 3.34(a). Jobs enter the system according to a Poisson process with rate λ, and use the CPU for an exponentially distributed time interval with mean $1/\mu_1$. Upon exiting the CPU, a job with probability p_1 exits the system, and with probability p_2 $(= 1 - p_1)$ uses the I/O device for a time which is exponentially distributed with mean $1/\mu_2$. Upon exit from the I/O device, a job again joins the CPU queue. We assume that all service times, including successive service times of the same job at the CPU or the I/O device, are independent.

We first calculate the arrival rates λ_1 and λ_2 at the CPU and I/O device queues, respectively. We have (cf. Fig. 3.34)

$$\lambda_1 = \lambda + \lambda_2, \qquad \lambda_2 = p_2\lambda_1$$

[These are Eqs. (3.114) specialized to this example.] By solving for λ_1 and λ_2 we obtain

$$\lambda_1 = \frac{\lambda}{p_1}, \qquad \lambda_2 = \frac{\lambda p_2}{p_1} \tag{3.131}$$

Let

$$\rho_1 = \frac{\lambda_1}{\mu_1}, \qquad \rho_2 = \frac{\lambda_2}{\mu_2} \tag{3.132}$$

The steady-state probability distribution of the system is given by Jackson's Theorem

$$P(n_1, n_2) = \rho_1^{n_1}(1 - \rho_1)\rho_2^{n_2}(1 - \rho_2)$$

The average number of jobs N_i in the i^{th} queue is the same as for an $M/M/1$ system with utilization factor ρ_i, that is,

$$N_1 = \frac{\rho_1}{1 - \rho_1}, \qquad N_2 = \frac{\rho_2}{1 - \rho_2}$$

The total number in the system is

$$N = N_1 + N_2 = \frac{\rho_1}{1 - \rho_1} + \frac{\rho_2}{1 - \rho_2}$$

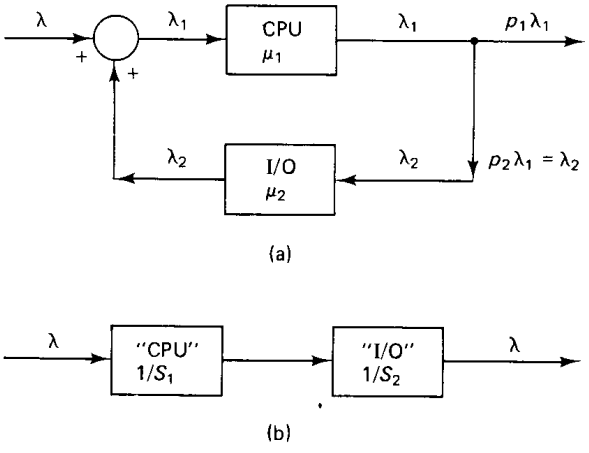


Figure 3.34 (a) Feedback model of a CPU and an I/O device (cf. Example 3.19). (b) "Equivalent" tandem model of a CPU and an I/O queue which has the same occupancy distribution as the feedback model.

and the average time in the system is

$$T = \frac{N}{\lambda} = \frac{\rho_1}{\lambda(1-\rho_1)} + \frac{\rho_2}{\lambda(1-\rho_2)}$$

Using Eqs. (3.131) and (3.132) we can write this relation as

$$T = \frac{\lambda_1/\mu_1}{\lambda(1-\lambda_1/\mu_1)} + \frac{\lambda_2/\mu_2}{\lambda(1-\lambda_2/\mu_2)} = \frac{\lambda/(\mu_1 p_1)}{\lambda\big(1-\lambda/(\mu_1 p_1)\big)} + \frac{\lambda p_2/(\mu_2 p_1)}{\lambda\big(1-\lambda p_2/(\mu_2 p_1)\big)}$$

$$= \frac{S_1}{1-\lambda S_1} + \frac{S_2}{1-\lambda S_2} \tag{3.133}$$

where

$$S_1 = \frac{1}{\mu_1 p_1}, \qquad S_2 = \frac{p_2}{\mu_2 p_1} \tag{3.134}$$

Since the utilization factor of the CPU queue is $\rho_1 = \lambda_1/\mu_1 = \lambda/(\mu_1 p_1)$, while the arrival rate of *new* job arrivals at the CPU (as opposed to feedback arrivals) is λ, we see from Little's Theorem that S_1 is the total CPU time a job requires on the average (this includes all visits of the job to the CPU). Similarly, S_2 is the total I/O time a job requires on the average.

An interesting interpretation of Eqs. (3.133) and (3.134) is that the average number of jobs and time in the system are the same as in an "equivalent" tandem model of CPU and I/O queues with service rates $1/S_1$ and $1/S_2$, respectively, as shown in Fig. 3.34(b). However, the probability density function of the time in the system is not the same in the feedback and tandem systems. To get some idea of this fact, suppose that $p_1 = p_2 = 1/2$ and that the CPU service rate is much faster than the I/O service rate ($\mu_1 >> \mu_2$). Then half the jobs in the feedback system do not require any I/O service and their average time in the system is much smaller than the average time of the other half. This is not so in the tandem system where the average job time in the "CPU" queue is very small and the system time is distributed approximately as in the "I/O" queue, that is, as in an $M/M/1$ queue with Poisson rate λ and service rate $1/S_1$.

Jackson's Theorem says in effect that the numbers of customers in the system's queues are distributed as if each queue is $M/M/1$ and is independent of the other queues [compare Eq. (3.117) and the corresponding equations in Section 3.3]. Despite this fact, *the total arrival process at each queue need not be Poisson*. As an example (see Fig. 3.35), suppose that there is a single queue with a service rate which is very large relative to the arrival rate from the outside. Suppose also that with probability p near unity, a customer upon completion of service is fed back into the queue. Hence, when an arrival occurs at the queue, there is a large probability of another arrival at the queue in a short time (namely, the feedback arrival), whereas at an arbitrary time point, there will be only a very slight chance of an arrival occurring shortly since λ is small. In other words, queue arrivals tend to occur in bursts triggered by the arrival of a single customer from the outside. Hence, the queue arrival process does not have independent interarrival times and cannot be Poisson.

Heuristic explanation of Jackson's Theorem. Our proof of Jackson's theorem is based on algebraic manipulation and gives little insight as to why this remarkable result holds. For this reason we provide a heuristic explanation for the case of the feed-

Samsung Exhibit 1015, Page 243 of 573

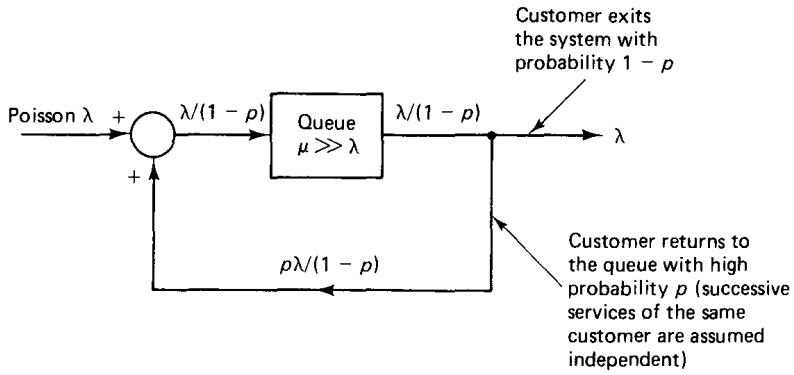


Figure 3.35 Example of a queue within a network where the external arrival process is Poisson but the total arrival process at the queue is not Poisson. An external arrival is typically processed fast (since μ is much larger than λ) and with high probability returns to the queue through the feedback loop. As a result, the total queue arrival process typically consists of bursts of arrivals, with each burst triggered by the arrival of a single customer from the outside.

back network of Fig. 3.35. This explanation can be generalized and made rigorous albeit at the expense of a great deal of technical complications (see [Wal83]).

Suppose that we introduce a delay Δ in the feedback loop of the single-queue network discussed above (see Fig. 3.36). Let us denote by $n(t)$ the number in the queue at time t, and by $f_\Delta(t)$ the content of the delay line at time t. The interpretation here is that $f_\Delta(t)$ is a function of time that specifies the customer output of the delay line in the subsequent Δ interval $(t, t+\Delta]$. Suppose that the initial distribution $n(0)$ of the queue state at time 0, is equal to the steady-state distribution of an $M/M/1$ queue, that is,

$$P\{n(0) = n\} = \rho^n(1-\rho) \tag{3.135}$$

where $\rho = \lambda/(\mu(1-p))$ is the utilization factor. Suppose also that $f_\Delta(0)$ is a portion of a Poisson arrival process with rate λ. The customers in $f_\Delta(0)$ have service times that are independent, exponentially distributed with parameter μ. We assume that $n(0)$ and $f_\Delta(0)$ are independent. Then, the input to the queue over the interval $[0, \Delta)$ will

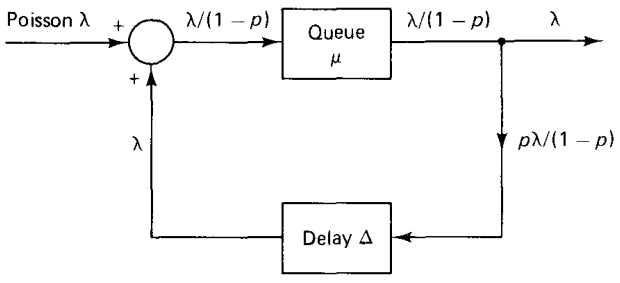


Figure 3.36 Heuristic explanation of Jackson's Theorem. Consider the introduction of an arbitrarily small positive delay Δ in the feedback loop of the network of Fig. 3.35. An occupancy distribution of the queue that equals the $M/M/1$ equilibrium, and a content of the delay line that is an independent Δ segment of a Poisson process form an equilibrium distribution of the overall system. Therefore, the $M/M/1$ equilibrium distribution is an equilibrium for the queue as suggested by Jackson's Theorem even though the total arrival process to the queue is not Poisson.

be the sum of two independent Poisson streams which are independent of the number in queue at time 0. It follows that the queue will behave in the interval $[0, \Delta)$ like an $M/M/1$ queue in equilibrium. Therefore, $n(\Delta)$ will be distributed according to the $M/M/1$ steady-state distribution of Eq. (3.135), and by part (b) of Burke's theorem, $n(\Delta)$ will be independent of the departure process from the queue in the interval $[0, \Delta)$, or, equivalently, of $f_\Delta(\Delta)$ —the delay line content at time Δ. Furthermore, by part (a) of Burke's Theorem, $f_\Delta(\Delta)$ will be Poisson. Thus, to summarize, we started out with independent initial conditions $n(0)$ and $f_\Delta(0)$ which had the equilibrium distribution of an $M/M/1$ queue and the statistics of a Poisson process, respectively, and Δ seconds later we obtained corresponding quantities $n(\Delta)$ and $f_\Delta(\Delta)$ with the same properties. Using the same reasoning, we can show that for all t which are multiples of Δ, $n(t)$ and $f_\Delta(t)$ have the same properties. It follows that the $M/M/1$ steady-state distribution of Eq. (3.135) is an equilibrium distribution for the queueing system for an arbitrary positive value of the feedback delay Δ, and this strongly suggests the validity of Jackson's Theorem. Note that this argument does not suggest that the feedback process, and therefore also the total arrival process to the queue, are Poisson. Indeed, it can be seen that successive Δ portions of the feedback arrival stream are correlated since, with probability p, a departing customer from the queue appears as an arrival Δ seconds later. Therefore, over the interval $[0, \infty)$, the feedback process is not Poisson. This is consistent with our earlier observations regarding the example of Fig. 3.35.

3.8.1 Extensions of Jackson's Theorem

There are a number of interesting extensions and variations of Jackson's Theorem, and in this and the next subsections we will describe a few of them.

State-dependent service rates. The model for Jackson's Theorem assumed so far requires that all queues have a single server. An extension to the multiserver case can be obtained by allowing the service rate at each queue to depend on the number of customers at that queue. Thus the model is the same as before but the service time at the j^{th} queue is exponentially distributed with mean $1/\mu_j(m)$, where m is the number of customers in the j^{th} queue just before the customer's departure (m includes the customer). The single-queue version of this model includes as special cases the $M/M/m$ and $M/M/\infty$ queues, and can be analyzed by means of a Markov chain (see Problem 3.16). The corresponding network of queues model can also be analyzed by means of a Markov chain, and is characterized by a product form structure for the stationary distribution.

Let us define

$$\rho_j(m) = \frac{\lambda_j}{\mu_j(m)}, \qquad j = 1, \ldots, K, \ m = 1, 2, \ldots \tag{3.136}$$

where λ_j is the total arrival rate at the j^{th} queue determined by Eq. (3.114). Let us also define

$$\hat{P}_j(n_j) = \begin{cases} 1, & \text{if } n_j = 0 \\ \rho_j(1)\rho_j(2)\cdots\rho_j(n_j), & \text{if } n_j > 0 \end{cases} \tag{3.137}$$

Samsung Exhibit 1015, Page 245 of 573

We have:

Jackson's Theorem for State-Dependent Service Rates. We have for all states $n = (n_1, \ldots, n_K)$

$$P(n) = \frac{\hat{P}_1(n_1) \cdots \hat{P}_K(n_K)}{G} \tag{3.138}$$

assuming that $0 < G < \infty$, where the normalizing constant G is given by

$$G = \sum_{n_1=0}^{\infty} \cdots \sum_{n_K=0}^{\infty} \hat{P}_1(n_1) \cdots \hat{P}_K(n_K) \tag{3.139}$$

Proof: Note that the formula for G guarantees that $P(n)$ is a probability distribution, that is, the sum of all $P(n)$ is unity. Using this fact, the proof is obtained by repeating the steps of the earlier proof of Jackson's Theorem, substituting the state-dependent service rates $\mu_j(m)$ in place of the rates μ_j at the appropriate points, and is left for the reader. **Q.E.D.**

Multiple classes of customers. In many interesting networks of queues the routing probabilities P_{ij} are not the same for all customers. Typical examples arise in data networks where the transmission queue joined by a packet at each intermediate node depends on the packet's destination and possibly its origin. It is therefore necessary to distinguish between customers of different types or classes. We will show that the product form expressions derived so far remain valid provided that the service time distribution at each queue is the same for all customer classes.

Let the customer classes be $c = 1, 2, \ldots, C$, let $r_j(c)$ be the rate of the external Poisson arrival process of class c at queue j, and let $P_{ij}(c)$ be the routing probabilities of class c. The assumptions made for an open Jackson network with a single customer class are replicated for each customer class, so that the equations

$$\lambda_j(c) = r_j(c) + \sum_{i=1}^{K} \lambda_i(c) P_{ij}(c), \qquad j = 1, \ldots, K, \quad c = 1, 2, \ldots, C \tag{3.140}$$

can be solved uniquely to give the total arrival rate $\lambda_j(c)$ at each queue j and for each customer class c. We assume that the service times at queue j are exponentially distributed with a common mean $1/\mu_j(m)$ for all customer classes, which depends on m, the total number of customers in the queue. As earlier, customers are served on a first-come first-serve basis.

The state of each queue is characterized not just by the total number of customers present in the queue, but also by the class of the customers and the relative order of arrival of the customers of different classes. Thus, we define the *composition of the* j^{th} *queue* at a given time as

$$z_j = (c_1, c_2, \ldots, c_{n_j})$$

where n_j is the total number of customers in the queue and c_i is the class of the customer in the i^{th} queue position.

The state of the queueing network at a given time is

$$z = (z_1, z_2, \ldots, z_K)$$

where z_j is the composition of the j^{th} queue at that time. It can be viewed as the state of a Markov chain the transition probabilities of which can be described in terms of the given quantities $\lambda_j(c)$, $\mu_j(m)$, and $P_{ij}(c)$. To state the appropriate form of Jackson's Theorem, define

$$\hat{\rho}_j(c, m) = \frac{\lambda_j(c)}{\mu_j(m)}, \qquad j = 1, \ldots, K, \quad c = 1, 2, \ldots, C \tag{3.141}$$

$$\hat{P}_j(z_j) = \begin{cases} 1, & \text{if } n_j = 0 \\ \hat{\rho}_j(c_1, 1)\hat{\rho}_j(c_2, 2) \cdots \hat{\rho}_j(c_{n_j}, n_j), & \text{if } n_j > 0 \end{cases} \tag{3.142}$$

$$G = \sum_{(z_1, \ldots, z_K)} \prod_{j=1}^{K} \hat{P}_j(z_j) \tag{3.143}$$

The proof of the following theorem follows the same pattern as the corresponding proof for the single customer class case, and is left for the reader.

Jackson's Theorem for Multiple Classes of Customers. Assuming that $0 < G < \infty$, the steady-state probability $\hat{P}(z)$ of state $z = (z_1, z_2, \ldots, z_K)$ is given by

$$\hat{P}(z) = \frac{\hat{P}_1(z_1) \cdots \hat{P}_K(z_K)}{G} \tag{3.144}$$

The steady-state probability $P(n) = P(n_1, \ldots, n_K)$ of having a total of n_j customers at queue $j = 1, \ldots, K$ (irrespective of class) is given by

$$P(n) = \sum_{z \in Z(n)} \hat{P}(z)$$

where $Z(n)$ is the set of states for which there is a total of n_j customers in queue j. By adding the expression (3.144) over $z \in Z(n)$, it is straighforward to verify that when the service rate at each queue is the same for all customer classes and is independent of the queue size, we have

$$P(n) = \prod_{j=1}^{K} \rho_j^{n_j}(1 - \rho_j) \tag{3.145}$$

where

$$\rho_j = \frac{\sum_{c=1}^{C} \lambda_j(c)}{\mu_j} \tag{3.146}$$

and μ_j is the service rate at queue j. In other words, the expression for $P(n)$ is the same as when there is a single customer class with total arrival rate at each queue j equal to the sum of the arrival rates of all customer classes $\sum_{j=1}^{C} \lambda_j(c)$.

Samsung Exhibit 1015, Page 247 of 573

We note that when the service rates at the queues are state dependent (but identical for all classes) the steady-state probabilities $P(n)$ can be shown to be given by the (single class) formulas (3.136) to (3.139). (See the references cited at the end of the chapter.)

The following example addresses the first data network model discussed in Section 3.6 (cf. Fig. 3.27). A similar analysis can be used for the datagram network model of Fig. 3.28.

Example 3.20 Virtual Circuit Network

Consider the network of communication links discussed in Section 3.5 (cf. Fig. 3.27). There are several traffic streams (or virtual circuits) denoted $c = 1, 2, \ldots, C$. Virtual circuit c uses a path p_c and has a Poisson arrival rate x_c. The total arrival rate of each link (i, j) is

$$\lambda_{ij} = \sum_{\{c|(i,j) \text{ lies on the path } p_c\}} x_c$$

Assume that the transmission times of all packets at link (i, j) are exponentially distributed with mean $1/\mu_{ij}$, which is the same for all virtual circuits. Assume also that the transmission times of all packets are independent, including the transmission times of the same packet at two different links (this is the essence of the Kleinrock independence approximation). Then the multiple-class model of this subsection applies and based on Eq. (3.145), the average number of packets in the system, N, is the same as if each link were an $M/M/1$ queue in isolation, that is,

$$N = \sum_{(i,j)} \frac{\lambda_{ij}}{\mu_{ij} - \lambda_{ij}}$$

Example 3.20 shows how multiple customer classes can be used to model data network situations where the route used by a packet depends on its origin and destination. There is still an unrealistic assumption in this example, namely that the transmission times of the same packet at two different links are independent. Furthermore, the assumption that all packet transmission times are exponentially distributed with common mean is often violated in practice. For a more realistic model, we would like to be able to assume more general transmission time distributions (*e.g.*, deterministic transmission times). It turns out that the product form of Eqs. (3.141) to (3.144) holds even when the service time distributions belong to a broad class of "phase-type" distributions, which can approximate arbitrarily closely deterministic service times (see [GeP87] and [Wal88]). For this, however, we need to assume that the service discipline at each queue is either *processor sharing* or *last-come first-serve* instead of first-come first-serve. Processor sharing refers to a situation where all customers in the queue are simultaneously served at the same rate (which is μ/n when μ is the total service rate and n is the number of customers). Last-come first-serve refers to the situation where upon arrival at a queue, a customer goes immediately into service, replacing the customer who is in service at the time (if any) on a preemptive-resume basis. While processor sharing or last-come first-serve may not be reasonable models for most data networks, the validity of the product form expression (3.141) to (3.144) under a variety of different assumptions is reassuring. It suggests that product forms provide a good first approximation in many practical situations where their use cannot be rigorously justified. Current practice and

Samsung Exhibit 1015, Page 248 of 573

experience seems to be supporting this view. We note, however, that for special types of priority disciplines, there are queueing networks that are unstable (some queue lengths grow indefinitely) even though the arrival rate is smaller than the service rate at each queue [KuS89]. We refer to the sources given at the end of the chapter for more details and discussion on the subject.

3.8.2 Closed Queueing Networks

Many interesting queueing problems involve a network of queues where the total number of customers is fixed because no customers are allowed to arrive or depart. Networks of this type are called *closed*, emphasizing the distinction from the earlier networks in this section which are called *open*. Examples 3.5 and 3.7 illustrate applications of closed networks. In both examples the fixed number of customers in the network depends on some limited resource, and the main purpose of analysis is to understand how the availability of this resource affects performance characteristics such as system throughput.

Closed networks can also be analyzed using Markov chains and it can be shown that the steady-state occupancy distribution has a product form under assumptions similar to those used earlier for open networks. For simplicity, we assume a single customer class, but extensions involving multiple customer classes are possible. Let M be the fixed number of customers in the system and let P_{ij} be the routing probability that a customer that departs from queue i will next visit queue j. Note that because no customer can exit the system, we have

$$\sum_{j=1}^{K} P_{ij} = 1, \qquad i = 1, \ldots, K$$

Let also $\mu_j(m)$ be the service rate at the j^{th} queue when the number of customers at that queue is m.

An important difference from the open network case is that the total arrival rates, denoted $\lambda_j(M)$, at the queues $j = 1, \ldots, K$ are not easily determined. We still have the equations

$$\lambda_j = \sum_{i=1}^{K} \lambda_i P_{ij}, \qquad j = 1, \ldots, K \tag{3.147}$$

obtained by setting to zero the external arrival rates r_j in Eq. (3.114). These equations do not have a unique solution anymore, but under some fairly natural assumptions, they determine the arrival rates $\lambda_j(M)$ up to a multiplicative constant. In particular, let us assume that the Markov chain with states $1, \ldots, K$ and transition probabilities P_{ij} is irreducible (see Appendix A). Then it can be shown that all solutions λ_j, $j = 1, \ldots, K$, of Eq. (3.147) are of the form*

*For a brief explanation, fix λ_1 at some positive value a and consider the system of equations $\lambda_j = aP_{1j} + \sum_{i=2}^{K} \lambda_i P_{ij}$, $j = 2, \ldots, K$. Because of the irreducibility assumption, this system has a unique solution. [See the explanation given in connection with the uniqueness of solution of the corresponding open network equation (3.114).] This unique solution is proportional to a and it can be shown to have positive elements.

Samsung Exhibit 1015, Page 249 of 573

$$\lambda_j = \alpha\overline{\lambda}_j, \qquad j = 1,\ldots,K$$

where α is a scalar and $\overline{\lambda}_j$, $j = 1,\ldots,K$ is a particular solution with $\overline{\lambda}_j > 0$ for all j. Thus the true arrival rates are given by

$$\lambda_j(M) = \alpha(M)\overline{\lambda}_j, \qquad j = 1,\ldots,K \tag{3.148}$$

where $\alpha(M)$ is the constant of proportionality corresponding to M. Note that while $\overline{\lambda}_j$ can be chosen to be independent of M, both $\alpha(M)$ and the true total arrival rates $\lambda_j(M)$ increase with M. In the case where the queue service rates μ_j are independent of the number of customers, $\alpha(M)$ tends asymptotically to the value that makes the maximum utilization factor $\max\{\lambda_1(M)/\mu_1,\ldots,\lambda_K(M)/\mu_K\}$ equal to one.

We now describe the form of Jackson's Theorem for closed networks. Let

$$\rho_j(m) = \frac{\overline{\lambda}_j}{\mu_j(m)}, \qquad j = 1,\ldots,K, \quad m = 1,2,\ldots \tag{3.149}$$

where $\{\overline{\lambda}_j \mid j = 1,\ldots,K\}$ is some positive solution of the system of equations (3.147). Denote

$$\hat{P}_j(n_j) = \begin{cases} 1, & \text{if } n_j = 0 \\ \rho_j(1)\rho_j(2)\cdots\rho_j(n_j), & \text{if } n_j > 0 \end{cases} \tag{3.150}$$

$$G(M) = \sum_{\{(n_1,\ldots,n_K)\mid n_1+\cdots+n_K=M\}} \hat{P}_1(n_1)\cdots\hat{P}_K(n_K) \tag{3.151}$$

We have:

Jackson's Theorem for Closed Networks. Under the preceding assumptions, we have for all states $n = (n_1,\ldots,n_K)$ with $n_1 + \cdots + n_K = M$

$$P(n) = \frac{\hat{P}_1(n_1)\cdots\hat{P}_K(n_K)}{G(M)} \tag{3.152}$$

[Note that because all solutions of Eq. (3.147) are scalar multiples of each other, the expression (3.152) for the probabilities $P(n)$ is not affected by the choice of the solution as long as this solution is nonzero. Note also that $G(M)$ is a normalization constant that ensures that $P(n)$ is a probability distribution.]

Proof: The proof is similar to the proof of Jackson's Theorem for open networks. We consider state vectors n and n' of the form

$$n = (n_1,\ldots,n_i,\ldots,n_j,\ldots,n_K)$$

$$n' = (n_1,\ldots,n_{i-1},n_i+1,n_{i+1},\ldots,n_{j-1},n_j-1,n_{j+1},\ldots,n_K)$$

Let $q_{nn'}$ be the corresponding transition rate. Jackson's Theorem will be proved if the rates $q^*_{nn'}$ defined for all n, n' by the equation

$$q^*_{nn'} = \frac{P(n')q_{n'n}}{P(n)} \tag{3.153}$$

satisfy, for all states n, the total rate equation

$$\sum_m q_{nm} = \sum_m q^*_{nm} \tag{3.154}$$

Indeed, let us assume for the purpose of the proof that the particular solutions $\bar{\lambda}_j$ are taken to be equal to the true arrival rates $\lambda_j(M)$, and for convenience let us denote both $\bar{\lambda}_j$ and $\lambda_j(M)$ as λ_j. Then we have [cf. Eqs. (3.124) and (3.125)]

$$q_{nn'} = \mu_j(n_j)P_{ji}, \qquad q^*_{nn'} = \frac{\mu_j(n_j)\lambda_i P_{ij}}{\lambda_j}$$

and the total rate equation (3.154) is written as

$$\sum_{\{(j,i)|n_j>0\}} \mu_j(n_j)P_{ji} = \sum_{\{(j,i)|n_j>0\}} \frac{\mu_j(n_j)\lambda_i P_{ij}}{\lambda_j}$$

We have

$$\sum_{\{(j,i)|n_j>0\}} \mu_j(n_j)P_{ji} = \sum_{i=1}^{K} \sum_{\{j|n_j>0\}} \mu_j(n_j)P_{ji} = \sum_{\{j|n_j>0\}} \mu_j(n_j) \sum_{i=1}^{K} P_{ji}$$

$$= \sum_{\{j|n_j>0\}} \mu_j(n_j) \tag{3.155}$$

We also have

$$\sum_{\{(j,i)|n_j>0\}} \frac{\mu_j(n_j)\lambda_i P_{ij}}{\lambda_j} = \sum_{i=1}^{K} \sum_{\{j|n_j>0\}} \frac{\mu_j(n_j)\lambda_i P_{ij}}{\lambda_j} = \sum_{\{j|n_j>0\}} \frac{\mu_j(n_j)}{\lambda_j} \sum_{i=1}^{K} \lambda_i P_{ij}$$

$$= \sum_{\{j|n_j>0\}} \mu_j(n_j) \tag{3.156}$$

From Eqs. (3.155) and (3.156) we see that the total rate equation (3.154) holds. **Q.E.D.**

Example 3.21 Closed Computer System with Feedback Loop for I/O

Consider a model of a computer CPU connected to an I/O device as shown in Fig. 3.37. This is a similar model to the one discussed in Example 3.19. The difference is that here

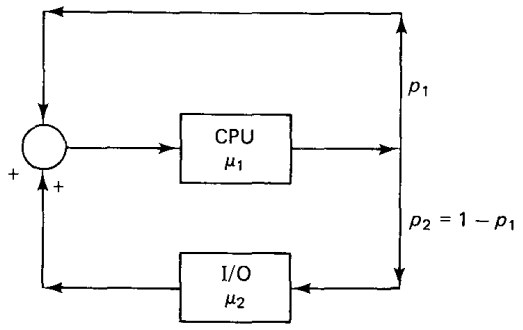


Figure 3.37 Closed network model of a feedback system of a CPU and an I/O device.

we have a closed network with each job reentering the CPU directly (with probability p_1) or after using the I/O device (with probability $p_2 = 1 - p_1$). There are M jobs in the system. We select

$$\overline{\lambda}_1 = \mu_1, \qquad \overline{\lambda}_2 = p_2\mu_1$$

as the particular solution of the system $\lambda_j = \sum_{i=1}^{2} \lambda_i P_{ij}$, $j = 1, 2$. With this choice we have

$$\rho_1 = 1, \qquad \rho_2 = \frac{p_2\mu_1}{\mu_2}$$

and the steady-state distribution of the system is given by [cf. Eqs. (3.149) to (3.151)]

$$P(M - n, n) = \frac{\rho_2^n}{G(M)}, \qquad n = 0, 1, \ldots, M$$

where the normalizing constant $G(M)$ is given by

$$G(M) = \sum_{n=0}^{M} \rho_2^n$$

The CPU utilization factor is given by

$$U(M) = 1 - P(0, M) = 1 - \frac{\rho_2^M}{G(M)} = \frac{G(M-1)}{G(M)}$$

and from Little's Theorem we obtain the arrival rate at the CPU as $\lambda_1(M) = U(M)\mu_1$. The expression above for the utilization factor $U(M)$ is a special case of a more general formula (see Problem 3.65).

Example 3.22 Throughput of a Time-Sharing System

Consider the time-sharing computer system with N terminals discussed in Example 3.7 [cf. Fig. 3.38(a)]. We will make detailed statistical assumptions on the times spent by jobs at the terminals and the CPU. We will consequently be able to obtain a closed-form expression for the throughput of the system in place of the upper and lower bounds obtained in Section 3.2.

In particular, let us assume that the reflection time of a job at a terminal is exponentially distributed with mean R and the processing time of a job at the CPU is exponentially distributed with mean P. All reflection and processing times are assumed independent. Then the terminal and CPU queues, viewed in isolation, can be modeled as an $M/M/\infty$ and an $M/M/1$ queue, respectively [see Fig. 3.38(b)]. Let $\overline{\lambda} = 1/P$ be the particular solution of the arrival rate equation for the system [cf. Eq. (3.147)]. With this choice we have

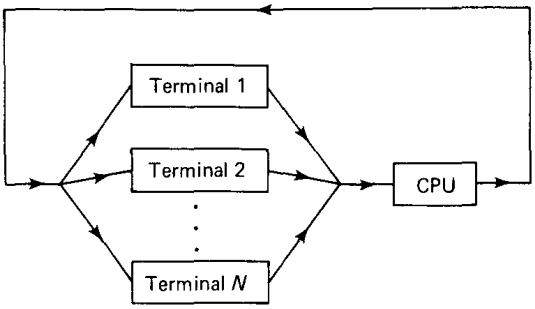
$$\rho_1 = \frac{R}{P}, \qquad \rho_2 = 1$$

The steady-state probability distribution is given by [cf. Eqs. (3.149) to (3.151)]

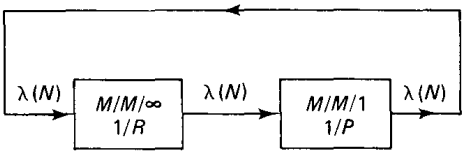
$$P(n, N - n) = \frac{(R/P)^n}{n!G(N)}$$

where the normalizing constant $G(N)$ is given by

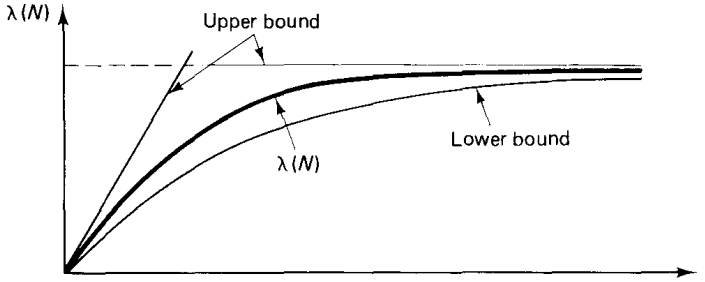
$$G(N) = 1 + (R/P) + \frac{(R/P)^2}{2!} + \cdots + \frac{(R/P)^N}{N!}$$



(a)



(b)



(c)

Figure 3.38 (a) Closed network model of a time-sharing system consisting of N terminals and a CPU. (b) Network of queues model of the system. There are at N jobs in the system at all times. (c) Throughput $\lambda(N)$ as a function of the number of terminals compared with the upper and lower bounds

$$\frac{N}{R+NP} \leq \lambda(N) \leq \min\left\{\frac{1}{P}, \frac{N}{R+P}\right\}$$

derived in Example 3.7.

The CPU utilization factor is

$$U(N) = 1 - \frac{P(N,0)}{G(N)} = \frac{(R/P)^N}{N!G(N)} = \frac{G(N-1)}{G(N)}$$

and by Little's Theorem, it is also equal to $\lambda(N)P$, where $\lambda(N)$ is the system throughput. Therefore, we have $\lambda(N) = U(N)/P$, or

$$\lambda(N) = \frac{1}{P}\frac{G(N-1)}{G(N)}$$

This expression for $\lambda(N)$ is shown in Fig. 3.38(c) and is contrasted with the upper and lower bounds

$$\frac{N}{R+NP} \leq \lambda(N) \leq \min\left\{\frac{1}{P}, \frac{N}{R+P}\right\} \tag{3.157}$$

obtained in Section 3.2.

3.8.3 Computational Aspects—Mean Value Analysis

Given a closed queueing network with M customers, one is typically interested in calculating

$N_j(M) =$ Average number of customers in the j^{th} queue

$T_j(M) =$ Average customer time spent per visit in the j^{th} queue

From these one can obtain the arrival rate at the j^{th} queue given by Little's Theorem as

$$\lambda_j(M) = \frac{N_j(M)}{T_j(M)} \tag{3.158}$$

One possibility is to calculate first the normalizing constant $G(M)$ of Eq. (3.151) and then to use the steady-state distribution $P(n)$ of Eq. (3.152) to obtain all quantities of interest. Several different algorithms can be used for this computation, which is often nontrivial when M is large. We will describe an alternative approach, known as *mean value analysis*, which calculates $N_j(M)$ and $T_j(M)$ directly. The normalizing constant $G(M)$ can then be obtained from these quantities and the arrival rates of Eq. (3.158). [See Problem 3.65 for the case where the service rates $\mu_j(m)$ do not depend on the number of customers m.]

Let us assume for simplicity that the service rate at the j^{th} queue is μ_j and does not depend on the number of customers in the queue. The main idea in mean value analysis is to start with the known quantities

$$T_j(0) = N_j(0) = 0, \qquad j = 1, \ldots, K \tag{3.159}$$

(corresponding to an empty system) and then calculate $T_j(1)$ and $N_j(1)$ (corresponding to one customer in the system), then calculate $T_j(2)$ and $N_j(2)$, and so on until the desired quantities $T_j(M)$ and $N_j(M)$ are obtained. This calculation is based on the equation (to be justified shortly)

$$T_j(s) = \frac{1}{\mu_j}\big(1 + N_j(s-1)\big), \qquad j = 1, \ldots, K, \quad s = 1, \ldots, M \tag{3.160}$$

which obtains $T_j(s)$ from $N_j(s-1)$ for all j. Then $N_j(s)$ is calculated from $T_j(s)$ for all j, using the equation (which is in effect Little's Theorem, as will be seen shortly)

$$N_j(s) = s\frac{\overline{\lambda}_j T_j(s)}{\sum_{i=1}^{K} \overline{\lambda}_i T_i(s)}, \qquad j = 1, \ldots, K, \quad s = 1, \ldots, M \tag{3.161}$$

where $\overline{\lambda}_j$, $j = 1, \ldots, K$ is a positive solution of the system of equations $\lambda_j = \sum_{i=1}^{K} \lambda_i P_{ij}$, $j = 1, \ldots, K$ [cf. Eq. (3.147)].

We proceed to derive Eqs. (3.160) and (3.161). Since we have for all j, $\lambda_j(s) = \alpha(s)\overline{\lambda}_j$ for some scalar $\alpha(s) > 0$, Eq. (3.161) can be written as

$$N_j(s) = s\frac{\lambda_j(s)T_j(s)}{\sum_{i=1}^{K}\lambda_i(s)T_i(s)}$$

and becomes evident once we observe that we have $\lambda_i(s)T_i(s) = N_i(s)$ for all i (by Little's Theorem) and $s = \sum_{i=1}^{K} N_i(s)$ [by the definition of $N_i(s)$].

To derive Eq. (3.160), we need an important result known as the *Arrival Theorem*. It states that the occupancy distribution found by a customer upon arrival at the j^{th} queue is the same as the steady-state distribution of the j^{th} queue in a closed network with the arriving customer removed. Thus, in an s-customer closed network, the average number of customers found upon arrival by a customer at the j^{th} queue is equal to $N_j(s-1)$, the average number seen by a random observer in the $(s-1)$-customer closed network. This explains the form of Eq. (3.160).

An intuitive explanation of the Arrival Theorem is given in Problem 3.59. For an analytical justification, assume that the s-customer closed network is in steady-state at time t and let $x(t)$ denote the state at that time. For each state $n = (n_1, \ldots, n_K)$ with $n_i > 0$, we want to calculate

$$\alpha_{ij}(n) = P\{x(t) = n \mid \text{a customer moved from queue } i \text{ to queue } j \text{ just after time } t\} \tag{3.162}$$

Let us denote by $M_{ij}(t)$ the event of a customer move from queue i to queue j just after time t, and let us denote by $M_i(t)$ the event of a customer move from queue i just after time t. Then Eq. (3.162) can be written as

$$\begin{aligned}\alpha_{ij}(n) &= \frac{P\{x(t) = n, M_{ij}(t) \mid M_i(t)\}}{P\{M_{ij}(t) \mid M_i(t)\}} \\ &= \frac{P\{x(t) = n \mid M_i(t)\}P\{M_{ij}(t) \mid x(t) = n,\ M_i(t)\}}{P\{M_{ij}(t) \mid M_i(t)\}} \\ &= \frac{P(n)P_{ij}}{\sum_{\{n'=(n'_1,\ldots,n'_K)\mid n'_i>0\}} P(n')P_{ij}}\end{aligned}$$

and finally, using Eqs. (3.149) to (3.152) for the steady-state probabilities $P(n)$,

$$\alpha_{ij}(n) = \frac{\hat{P}_1(n_1)\cdots\hat{P}_K(n_K)}{\sum_{\{(n'_1,\ldots,n'_K)\mid n'_1+\cdots+n'_K=s,\ n'_i>0\}} \hat{P}_1(n'_1)\cdots\hat{P}_K(n'_K)} \tag{3.163}$$

The numerator and the denominator of this equation contain a common factor ρ_i because $n_i > 0$ in the numerator and $n'_i > 0$ in each term of the denominator. By dividing with ρ_i, and by using the expression (3.150) for $\hat{P}_j(n_j)$, we obtain

$$\alpha_{ij}(n) = \frac{\hat{P}_1(n_1)\cdots\hat{P}_{i-1}(n_{i-1})\hat{P}_i(n_i-1)\hat{P}_{i+1}(n_{i+1})\cdots\hat{P}_K(n_K)}{\sum_{\{(n'_1,\ldots,n'_K)\mid n'_1+\cdots+n'_K=s-1\}} \hat{P}_1(n'_1)\cdots\hat{P}_K(n'_K)}$$

Therefore, $\alpha_{ij}(n)$ is equal to the steady-state probability of state $(n_1, \ldots, n_{i-1}, n_i - 1, n_{i+1}, \ldots, n_K)$ in the $(s-1)$-customer closed network, as stated by the Arrival Theorem.

Samsung Exhibit 1015, Page 255 of 573

We note that the Arrival Theorem holds also in some cases where there are multiple classes of customers and where the queues have multiple servers. Mean value analysis can also be used in these cases, with Eq. (3.160) replaced by the appropriate formula.

Finally, a number of approximate methods based on mean value analysis have been proposed. As an example, suppose that an approximate relation of the form

$$N_j(M-1) = f_j\big(N_j(M)\big)$$

is hypothesized; for large M, one reasonable possibility is

$$N_j(M-1) = \frac{M-1}{M} N_j(M)$$

Then Eqs. (3.160) and (3.161) yield the system of nonlinear equations

$$T_j(M) = \frac{1}{\mu_j}\Big(1 + f_j\big(N_j(M)\big)\Big), \qquad j = 1, \ldots, K$$

$$N_j(M) = M \frac{\overline{\lambda}_j T_j(M)}{\sum_{i=1}^{K} \overline{\lambda}_i T_i(M)}, \qquad j = 1, \ldots, K$$

which can be solved by iterative methods to yield approximate values for $T_j(M)$ and $N_j(M)$.

SUMMARY

Queueing models provide qualitative insights on the performance of data networks, and quantitative predictions of average packet delay. An example of the former is the comparison of time-division and statistical multiplexing, while an example of the latter is the delay analysis of reservation systems.

To obtain tractable queueing models for data networks, it is frequently necessary to make simplifying assumptions. A prime example is the Kleinrock independence approximation discussed in Section 3.6. Delay predictions based on this approximation are adequate for many uses. A more accurate alternative is simulation, which, however, can be slow, expensive, and lacking in insight.

Little's Theorem is a simple but extremely useful result since it holds under very general conditions. To proceed beyond this theorem we assumed Poisson arrivals and independent interarrival and service times. This led to the $M/G/1$ system and its extensions in reservation and priority queueing systems. We analyzed a surprisingly large number of important delay models using simple graphical arguments. An alternative analysis was based on the use of Markov chain models and led to the derivation of the occupancy probability distribution of the $M/M/1$, $M/M/m$, and related systems.

Reversibility is an important notion that helps to prove and understand Jackson's Theorem and provides a taste of advanced queueing topics.

NOTES, SOURCES, AND SUGGESTED READING

Section 3.2. Little's Theorem was formalized in [Lit61]. Rigorous proofs under various assumptions are given in [Sti72] and [Sti74]. Several applications in finding performance bounds of computer systems are described in [StA85].

Section 3.3. For a general background on the Poisson process, Markov chains, and related topics, see [Ros80], [Ros83], and [KaT75]. Standard texts on queueing theory include [Coo81], [GrH85], [HeS82], [Kle75], and [Wol89]. A reference for the fact that Poisson arrivals see a typical occupancy distribution (Section 3.3.2) is [Wol82a].

Section 3.4. Queueing systems that admit analysis via Markov chain theory include those where the service times have an Erlang distribution; see [Kle76], Chap. 4. For extensions to more general models and computational methods, see [Kei79], [Neu81], [Haj82], and [Twe82]. For methods to calculate the blocking probability in circuit switching networks (Example 3.14), see [Kau81], [Kel86], [LeG89], [RoT90], and [TsR90].

Section 3.5. The P-K formula is often derived by using z-transforms; see [Kle75]. This derivation is not very insightful, but gives the probability distribution of the system occupancy (not just the mean that we obtained via our much simpler analysis). For more on delay analysis of ARQ systems, see [AnP86] and [ToW79].

The results on polling and reservation systems are fairly recent; see [Coo70], [Eis79], [FeA85], [FuC85], [IEE86], and [Kue79]. The original references that are closest to our analysis are [Has72] for unlimited service systems, [NoT78] for limited service systems, and [Hum78] for nonsymmetric polling systems. Reference [Tak86] is a monograph devoted to polling. There are two main reservation and polling systems considered in the literature: the symmetric case, where all users have identical arrival and reservation interval statistics, and the nonsymmetric case, where these statistics are user dependent. The former case admits simple expressions for the mean waiting times while the latter does not. We have considered the partially symmetric case, where all users have identical arrival statistics but different reservation interval statistics. The fact that simple expressions hold for this case has not been known earlier, and in this respect, our formulas are original. Our treatment in terms of simple graphical arguments is also original. Approximate formulas for nonsymmetric polling systems are given in [BoM86] and [IbC89]. The result of Problem 3.35 on limited service systems with shared reservation and data intervals is new.

An extensive treatment of priority queueing systems is [Jai68]. A simpler, less comprehensive exposition is given in [Kle75].

The material on the $G/G/1$ queue is due to [Kin62]. For further material, see Chapter 11 of [Wol89], [Whi83a], [Whi83b], and the references quoted there.

Section 3.6. Delay analysis for data networks in terms of $M/M/1$ approximations was introduced in [Kle64]. References [Wol82b] and [PiW82] study via analysis and simulation the behavior of two queues in tandem when the service times of a customer at the two queues are dependent. The special issue [IEE86] provides a view of recent work on the subject.

Section 3.7. The notion of reversibility was used in Markov chain analysis by Kolmogorov [Kol36], and was explored in depth in [Kel79] and [Wal88].

Section 3.8. There is an extensive literature on product form solutions of queueing networks following Jackson's original paper [Jac57]. The survey [DiK85] lists 314 references. There are also several books on the subject: [Kel79], [BrB80], [GeP87], [Wal88], and [CoG89]. The heuristic explanation of Jackson's theorem is due to [Wal83].

PROBLEMS

3.1 Customers arrive at a fast-food restaurant at a rate of five per minute and wait to receive their order for an average of 5 minutes. Customers eat in the restaurant with probability 0.5 and carry out their order without eating with probability 0.5. A meal requires an average of 20 minutes. What is the average number of customers in the restaurant? (Answer: 75.)

3.2 Two communication nodes 1 and 2 send files to another node 3. Files from 1 and 2 require on the average R_1 and R_2 time units for transmission, respectively. Node 3 processes a file of node i ($i = 1, 2$) in an average of P_i time units and then requests another file from either node 1 or node 2 (the rule of choice is left unspecified). If λ_i is the throughput of node i in files sent per unit time, what is the region of all feasible throughput pairs (λ_1, λ_2) for this system?

3.3 A machine shop consists of N machines that occasionally fail and get repaired by one of the shop's m repairpersons. A machine will fail after an average of R time units following its previous repair and requires an average of P time units to get repaired. Obtain upper and lower bounds (functions of R, N, P, and m) on the number of machine failures per unit time and on the average time between repairs of the same machine.

3.4 The average time T a car spends in a certain traffic system is related to the average number of cars N in the system by a relation of the form $T = \alpha + \beta N^2$, where $\alpha > 0$, $\beta > 0$ are given scalars.

(a) What is the maximal car arrival rate λ^* that the system can sustain?

(b) When the car arrival rate is less than λ^*, what is the average time a car spends in the system assuming that the system reaches a statistical steady state? Is there a unique answer? Try to argue against the validity of the statistical steady-state assumption.

3.5 An absent-minded professor schedules two student appointments for the same time. The appointment durations are independent and exponentially distributed with mean 30 minutes. The first student arrives on time, but the second student arrives 5 minutes late. What is the expected time between the arrival of the first student and the departure of the second student? (Answer: 60.394 minutes.)

3.6 A person enters a bank and finds all of the four clerks busy serving customers. There are no other customers in the bank, so the person will start service as soon as one of the customers in service leaves. Customers have independent, identical, exponential distribution of service time.

(a) What is the probability that the person will be the last to leave the bank assuming that no other customers arrive?

(b) If the average service time is 1 minute, what is the average time the person will spend in the bank?

(c) Will the answer in part (a) change if there are some additional customers waiting in a common queue and customers begin service in the order of their arrival?

3.7 A communication line is divided in two identical channels each of which will serve a packet traffic stream where all packets have equal transmission time T and equal interarrival time $R > T$. Consider, alternatively, statistical multiplexing of the two traffic streams by combining the two channels into a single channel with transmission time $T/2$ for each packet. Show that the average system time of a packet will be decreased from T to something between $T/2$ and $3T/4$, while the variance of waiting time in queue will be increased from 0 to as much as $T^2/16$.

3.8 Consider a packet stream whereby packets arrive according to a Poisson process with rate 10 packets/sec. If the interarrival time between any two packets is less than the transmission time of the first to arrive, the two packets are said to collide. (This notion will be made more meaningful in Chapter 4 when we discuss multiaccess schemes.) Find the probabilities that a packet does not collide with either its predecessor or its successor, and that a packet does not collide with another packet assuming:

(a) All packets have a transmission time of 20 msec. (Answer: Both probabilities are equal to 0.67.)

(b) Packets have independent, exponentially distributed transmission times with mean 20 msec. (This part requires the $M/M/\infty$ results.) (Answer: The probability of no collision with predecessor or successor is 0.694. The probability of no collision is 0.682.)

3.9 A communication line capable of transmitting at a rate of 50 Kbits/sec will be used to accommodate 10 sessions each generating Poisson traffic at a rate 150 packets/min. Packet lengths are exponentially distributed with mean 1000 bits.

(a) For each session, find the average number of packets in queue, the average number in the system, and the average delay per packet when the line is allocated to the sessions by using:

(1) 10 equal-capacity time-division multiplexed channels. (Answer: $N_Q = 5$, $N = 10$, $T = 0.4$ sec.)

(2) Statistical multiplexing. (Answer: $N_Q = 0.5$, $N = 1$, $T = 0.04$ sec.)

(b) Repeat part (a) for the case where five of the sessions transmit at a rate of 250 packets/min while the other five transmit at a rate of 50 packets/min. (Answer: $N_Q = 21$, $N = 26$, $T = 1.038$ sec.)

3.10 This problem deals with some of the basic properties of the Poisson process.

(a) Derive Eqs. (3.11) to (3.14).

(b) Show that if the arrivals in two disjoint time intervals are independent and Poisson distributed with parameters $\lambda\tau_1$, $\lambda\tau_2$, then the number of arrivals in the union of the intervals is Poisson distributed with parameter $\lambda(\tau_1 + \tau_2)$. (This shows in particular that the Poisson distribution of the number of arrivals in any interval [cf. Eq. (3.10)] is consistent with the independence requirement in the definition of the Poisson process.) *Hint*: Verify the correctness of the following calculation, where N_1 and N_2 are the number of arrivals in the two disjoint intervals:

Samsung Exhibit 1015, Page 259 of 573

$$P\{N_1 + N_2 = n\} = \sum_{k=0}^{n} P\{N_1 = k\}P\{N_2 = n-k\}$$

$$= e^{-\lambda(\tau_1+\tau_2)} \sum_{k=0}^{n} \frac{(\lambda\tau_1)^k(\lambda\tau_2)^{n-k}}{k!(n-k)!}$$

$$= e^{-\lambda(\tau_1+\tau_2)} \frac{(\lambda\tau_1 + \lambda\tau_2)^n}{n!}$$

(c) Show that if k independent Poisson processes $A_1, \ldots, A_k$ are combined into a single process $A = A_1 + A_2 + \cdots + A_k$, then A is Poisson with rate λ equal to the sum of the rates $\lambda_1, \ldots \lambda_k$ of $A_1, \ldots, A_k$. Show also that the probability that the first arrival of the combined process comes from A_1 is λ_1/λ independently of the time of arrival. *Hint*: For $k = 2$ write

$$P\{A_1(t+\tau) + A_2(t+\tau) - A_1(t) - A_2(t) = n\}$$

$$= \sum_{m=0}^{n} P\{A_1(t+\tau) - A_1(t) = m\}P\{A_2(t+\tau) - A_2(t) = n-m\}$$

and continue as in the hint for part (b). Also write for any t

$$P\{1 \text{ arrival from } A_1 \text{ prior to } t \mid 1 \text{ occurred}\}$$

$$= \frac{P\{1 \text{ arrival from } A_1 \text{ prior to } t,\ 0 \text{ from } A_2\}}{P\{1 \text{ occurred}\}}$$

$$= \frac{\lambda_1 t e^{-\lambda_1 t} e^{-\lambda_2 t}}{\lambda t e^{-\lambda t}} = \frac{\lambda_1}{\lambda}$$

(d) Suppose we know that in an interval $[t_1, t_2]$ only one arrival of a Poisson process has occurred. Show that, conditional on this knowledge, the time of this arrival is uniformly distributed in $[t_1, t_2]$. *Hint*: Verify that if t is the time of arrival, we have for all $s \in [t_1, t_2]$,

$$P\{t < s \mid 1 \text{ arrival occurred in } [t_1, t_2]\}$$

$$= \frac{P\{1 \text{ arrival occurred in } [t_1, s),\ 0 \text{ arrivals occurred in } [s, t_2]\}}{P\{1 \text{ arrival occurred}\}}$$

$$= \frac{s - t_1}{t_2 - t_1}$$

3.11 Packets arrive at a transmission facility according to a Poisson process with rate λ. Each packet is independently routed with probability p to one of two transmission lines and with probability $(1-p)$ to the other.

(a) Show that the arrival processes at the two transmission lines are Poisson with rates λp and $\lambda(1-p)$, respectively. Furthermore, the two processes are independent. *Hint*: Let $N_1(t)$ and $N_2(t)$ be the number of arrivals in $[0, t]$ in lines 1 and 2, respectively. Verify the correctness of the following calculation:

$$P\{N_1(t) = n, N_2(t) = m\}$$

$$= P\{N_1(t) = n, N_2(t) = m \mid N(t) = n + m\} \frac{e^{-\lambda t}(\lambda t)^{n+m}}{(n+m)!}$$

$$= \binom{n+m}{n} p^n (1-p)^m \frac{e^{-\lambda t}(\lambda t)^{n+m}}{(n+m)!}$$

$$= \frac{e^{-\lambda tp}(\lambda tp)^n}{n!} \frac{e^{-\lambda t(1-p)}(\lambda t(1-p))^m}{m!}$$

$$P\{N_1(t) = n\} = \sum_{m=0}^{\infty} P\{N_1(t) = n, N_2(t) = m\} = \frac{e^{-\lambda tp}(\lambda tp)^n}{n!}$$

(b) Use the result of part (a) to show that the probability distribution of the customer delay in a (first-come first-serve) $M/M/1$ queue with arrival rate λ and service rate μ is exponential, that is, in steady-state we have

$$P\{T_i \geq \tau\} = e^{-(\mu-\lambda)\tau}$$

where T_i is the delay of the i^{th} customer. *Hint*: Consider a Poisson process A with arrival rate μ, which is split into two processes, A_1 and A_2, by randomization according to a probability $\rho = \lambda/\mu$; that is, each arrival of A is an arrival of A_1 with probability ρ and an arrival of A_2 with probability $(1-\rho)$, independently of other arrivals. Show that the interarrival times of A_2 have the same distribution as T_i.

3.12 Let τ_1 and τ_2 be two exponentially distributed, independent random variables with means $1/\lambda_1$ and $1/\lambda_2$. Show that the random variable $\min\{\tau_1, \tau_2\}$ is exponentially distributed with mean $1/(\lambda_1 + \lambda_2)$ and that $P\{\tau_1 < \tau_2\} = \lambda_1/(\lambda_1 + \lambda_2)$. Use these facts to show that the $M/M/1$ queue can be described by a continuous-time Markov chain with transition rates $q_{n(n+1)} = \lambda$, $q_{(n+1)n} = \mu$, $n = 0, 1, \ldots$. (See Appendix A for material on continuous-time Markov chains.)

3.13 Persons arrive at a taxi stand with room for W taxis according to a Poisson process with rate λ. A person boards a taxi upon arrival if one is available and otherwise waits in a line. Taxis arrive at the stand according to a Poisson process with rate μ. An arriving taxi that finds the stand full departs immediately; otherwise, it picks up a customer if at least one is waiting, or else joins the queue of waiting taxis.

(a) Use an $M/M/1$ queue formulation to obtain the steady-state distribution of the person's queue. What is the steady-state probability distribution of the taxi queue size when $W = 5$ and λ and μ are equal to 1 and 2 per minute, respectively? (Answer: Let p_i = Probability of i taxis waiting. Then $p_0 = 1/32$, $p_1 = 1/32$, $p_2 = 1/16$, $p_3 = 1/8$, $p_4 = 1/4$, $p_5 = 1/2$.)

(b) In the leaky bucket flow control scheme to be discussed in Chapter 6, packets arrive at a network entry point and must wait in a queue to obtain a permit before entering the network. Assume that permits are generated by a Poisson process with given rate and can be stored up to a given maximum number; permits generated while the maximum number of permits is available are discarded. Assume also that packets arrive according to a Poisson process with given rate. Show how to obtain the occupancy distribution of the queue of packets waiting for permits. *Hint*: This is the same system as the one of part (a).

Samsung Exhibit 1015, Page 261 of 573

(c) Consider the flow control system of part (b) with the difference that permits are not generated according to a Poisson process but are instead generated periodically at a given rate. (This is a more realistic assumption.) Formulate the problem of finding the occupancy distribution of the packet queue as an $M/D/1$ problem.

3.14 A communication node A receives Poisson packet traffic from two other nodes, 1 and 2, at rates λ_1 and λ_2, respectively, and transmits it, on a first-come first-serve basis, using a link with capacity C bits/sec. The two input streams are assumed independent and their packet lengths are identically and exponentially distributed with mean L bits. A packet from node 1 is always accepted by A. A packet from node 2 is accepted only if the number of packets in A (in queue or under transmission) is less than a given number $K > 0$; otherwise, it is assumed lost.

(a) What is the range of values of λ_1 and λ_2 for which the expected number of packets in A will stay bounded as time increases?

(b) For λ_1 and λ_2 in the range of part (a) find the steady-state probability of having n packets in A ($0 \leq n < \infty$). Find the average time needed by a packet from source 1 to clear A once it enters A, and the average number of packets in A from source 1. Repeat for packets from source 2.

3.15 Consider a system that is identical to $M/M/1$ except that when the system empties out, service does not begin again until k customers are present in the system (k is given). Once service begins it proceeds normally until the system becomes empty again. Find the steady-state probabilities of the number in the system, the average number in the system, and the average delay per customer. [Answer: The average number in the system is $N = \rho/(1-\rho) + (k-1)/2$.]

3.16 *M/M/1-Like System with State-Dependent Arrival and Service Rate.* Consider a system which is the same as $M/M/1$ except that the rate λ_n and service rate μ_n when there are n customers in the system depend on n. Show that

$$p_{n+1} = (\rho_0 \cdots \rho_n) p_0$$

where $\rho_k = \lambda_k/\mu_{k+1}$ and

$$p_0 = \left[1 + \sum_{k=0}^{\infty} (\rho_0 \cdots \rho_k)\right]^{-1}$$

3.17 *Discrete-Time Version of the M/M/1 System.* Consider a queueing system where interarrival and service times are integer valued, so customer arrivals and departures occur at integer times. Let λ be the probability that an arrival occurs at any time k, and assume that at most one arrival can occur. Also let μ be the probability that a customer who was in service at time k will complete service at time $k+1$. Find the occupancy distribution p_n in terms of λ and μ.

3.18 Empty taxis pass by a street corner at a Poisson rate of 2 per minute and pick up a passenger if one is waiting there. Passengers arrive at the street corner at a Poisson rate of 1 per minute and wait for a taxi only if there are fewer than four persons waiting; otherwise, they leave and never return. Find the average waiting time of a passenger who joins the queue. (Answer: 13/15 min.)

3.19 A telephone company establishes a direct connection between two cities expecting Poisson traffic with rate 30 calls/min. The durations of calls are independent and exponentially distributed with mean 3 min. Interarrival times are independent of call durations. How many circuits should the company provide to ensure that an attempted call is blocked (because all

Samsung Exhibit 1015, Page 262 of 573

circuits are busy) with probability less than 0.01? It is assumed that blocked calls are lost (*i.e.*, a blocked call is not attempted again).

3.20 A mail-order company receives calls at a Poisson rate of one per 2 min and the duration of the calls is exponentially distributed with mean 3 min. A caller who finds all telephone operators busy patiently waits until one becomes available. Write a computer program to determine how many operators the company should use so that the average waiting time of a customer is half a minute or less?

3.21 Consider the $M/M/1/m$ system which is the same as $M/M/1$ except that there can be no more than m customers in the system and customers arriving when the system is full are lost. Show that the steady-state occupancy probabilities are given by

$$p_n = \frac{\rho^n(1-\rho)}{1-\rho^{m+1}}, \qquad 0 \le n \le m$$

3.22 An athletic facility has five tennis courts. Players arrive at the courts at a Poisson rate of one pair per 10 min and use a court for an exponentially distributed time with mean 40 min.

(a) Suppose that a pair of players arrives and finds all courts busy and k other pairs waiting in queue. How long will they have to wait to get a court on the average?

(b) What is the average waiting time in queue for players who find all courts busy on arrival?

3.23 Consider an $M/M/\infty$ queue with servers numbered $1, 2, \ldots$ There is an additional restriction that upon arrival a customer will choose the lowest-numbered server that is idle at the time. Find the fraction of time that each server is busy. Will the answer change if the number of servers is finite? *Hint*: Argue that in steady-state the probability that all of the first m servers are busy is given by the Erlang B formula of the $M/M/m/m$ system. Find the total arrival rate to servers $(m+1)$ and higher, and from this, the arrival rate to each server.

3.24 $M/M/1$ *Shared Service System.* Consider a system which is the same as $M/M/1$ except that whenever there are n customers in the system they are all served simultaneously at an equal rate $1/n$ per unit time. Argue that the steady-state occupancy distribution is the same as for the $M/M/1$ system. *Note*: It can be shown that the steady-state occupancy distribution is the same as for $M/M/1$ even if the service time distribution is not exponential (*i.e.*, for an $M/G/1$ type of system) ([Ros83], p. 171).

3.25 *Blocking Probability for Single-Cell Radio Systems ([BaA81] and [BaA82]).* A cellular radiotelephone system serves a given geographical area with m radiotelephone channels connected to a single switching center. There are two types of calls: radio-to-radio calls, which occur with a Poisson rate λ_1 and require two radiochannels per call, and radio-to-nonradio calls, which occur with a Poisson rate λ_2 and require one radiochannel per call. The duration of all calls is exponentially distributed with mean $1/\mu$. Calls that cannot be accommodated by the system are blocked. Give formulas for the blocking probability of the two types of calls.

3.26 A facility of m identical machines is sharing a single repairperson. The time to repair a failed machine is exponentially distributed with mean $1/\lambda$. A machine, once operational, fails after a time that is exponentially distributed with mean $1/\mu$. All failure and repair times are independent. What is the steady-state proportion of time where there is no operational machine?

3.27 $M/M/2$ *System with Heterogeneous Servers.* Derive the stationary distribution of an $M/M/2$ system where the two servers have different service rates. A customer that arrives when the system is empty is routed to the faster server.

Samsung Exhibit 1015, Page 263 of 573

3.28 In Example 3.11, verify the formula $\sigma_f = (\lambda/\mu)^{1/2} s_\gamma$. *Hint*: Write

$$E\{f^2\} = E\left\{\left(\sum_{i=1}^{n} \gamma_i\right)^2\right\} = E\left\{E\left\{\left(\sum_{i=1}^{n} \gamma_i\right)^2 \mid n\right\}\right\},$$

and use the fact that n is Poisson distributed.

3.29 Customers arrive at a grocery store's checkout counter according to a Poisson process with rate 1 per minute. Each customer carries a number of items that is uniformly distibuted between 1 and 40. The store has two checkout counters, each capable of processing items at a rate of 15 per minute. To reduce the customer waiting time in queue, the store manager considers dedicating one of the two counters to customers with x items or less and dedicating the other counter to customers with more than x items. Write a small computer program to find the value of x that minimizes the average customer waiting time.

3.30 In the $M/G/1$ system, show that

$$P\{\text{the system is empty}\} = 1 - \lambda\overline{X}$$

$$\text{Average length of time between busy periods} = \frac{1}{\lambda}$$

$$\text{Average length of busy period} = \frac{\overline{X}}{1 - \lambda\overline{X}}$$

$$\text{Average number of customers served in a busy period} = \frac{1}{1 - \lambda\overline{X}}$$

3.31 Consider the following argument in the $M/G/1$ system: When a customer arrives, the probability that another customer is being served is $\lambda\overline{X}$. Since the served customer has mean service time $\overline{X}$, the average time to complete the service is $\overline{X}/2$. Therefore, the mean residual service time is $\lambda\overline{X}^2/2$. What is wrong with this argument?

3.32 *$M/G/1$ System with Arbitrary Order of Service.* Consider the $M/G/1$ system with the difference that customers are not served in the order they arrive. Instead, upon completion of a customer's service, one of the waiting customers in queue is chosen according to some rule, and is served next. Show that the P-K formula for the average waiting time in queue W remains valid provided that the relative order of arrival of the customer chosen is independent of the service times of the customers waiting in queue. *Hint*: Argue that the independence hypothesis above implies that at any time t, the number $N_Q(t)$ of customers waiting in queue is independent of the service times of these customers. Show that this in turn implies that $U = R + \rho W$, where R is the mean residual time and U is the average steady-state unfinished work in the system (total remaining service time of the customers in the system). Argue that U and R are independent of the order of customer service.

3.33 Show that Eq. (3.59) for the average delay of time-division multiplexing on a slot basis can be obtained as a special case of the results for the limited service reservation system. *Hint*: Consider the gated system with zero packet length.

3.34 Consider the limited service reservation system. Show that for both the gated and the partially gated versions:

(a) The steady-state probability of arrival of a packet during a reservation interval is $1 - \rho$.

(b) The steady-state probability of a reservation interval being followed by an empty data interval is $(1 - \rho - \lambda\overline{V})/(1 - \rho)$. *Hint:* If p is the required probability, argue that the ratio of the times used for data intervals and for reservation intervals is $(1 - p)\overline{X}/\overline{V}$.

3.35 *Limited Service Reservation System with Shared Reservation and Data Intervals.* Consider the gated version of the limited service reservation system with the difference that the m users share reservation and data intervals, (*i.e.*, all users make reservations in the same interval and transmit at most one packet each in the subsequent data interval). Show that

$$W = \frac{\lambda \overline{X^2}}{2(1-\rho-\lambda\overline{V}/m)} + \frac{(1-\rho)\overline{V^2}}{2(1-\rho-\lambda\overline{V}/m)\overline{V}} + \frac{(1-\rho\alpha-\lambda\overline{V}/m)\overline{V}}{1-\rho-\lambda\overline{V}/m}$$

where $\overline{V}$ and $\overline{V^2}$ are the first two moments of the reservation interval, and α satisfies

$$\frac{\overline{K}+(\hat{K}-1)(2\overline{K}-\hat{K})}{2m\overline{K}} - \frac{1}{2m} \leq \alpha \leq \frac{1}{2} - \frac{1}{2m}$$

where

$$\overline{K} = \frac{\lambda\overline{V}}{1-\rho}$$

is the average number of packets per data interval, and $\hat{K}$ is the smallest integer which is larger than $\overline{K}$. Verify that the formula for W becomes exact as $\rho \to 0$ (light load) and as $\rho \to 1 - \lambda\overline{V}/m$ (heavy load). *Hint:* Verify that

$$W = R + \lambda W + \left(1 + \frac{\lambda W}{m} - S\right)\overline{V}$$

where $S = \lim_{i\to\infty} E\{S_i\}$ and S_i is the number (0 or 1) of packets of the owner of packet i that will start transmission between the time of arrival of packet i and the end of the cycle in which packet i arrives. Try to obtain bounds for S by considering separately the cases where packet i arrives in a reservation and in a data interval.

3.36 Repeat part (a) of Problem 3.9 for the case where packet lengths are not exponentially distributed, but 10% of the packets are 100 bits long and the rest are 1500 bits long. Repeat the problem for the case where the short packets are given nonpreemptive priority over the long packets. (Answer: $N_Q = 0.791$, $N = 1.47$, $T = 0.588$ sec.)

3.37 Persons arrive at a Xerox machine according to a Poisson process with rate one per minute. The number of copies to be made by each person is uniformly distributed between 1 and 10. Each copy requires 3 sec. Find the average waiting time in queue when:

(a) Each person uses the machine on a first-come first-serve basis. (Answer: $W = 3.98$.)

(b) Persons with no more than 2 copies to make are given nonpreemptive priority over other persons.

3.38 *Priority Systems with Multiple Servers.* Consider the priority systems of Section 3.5.3 assuming that there are m servers and that all priority classes have exponentially distributed service times with common mean $1/\mu$.

(a) Consider the nonpreemptive system. Show that Eq. (3.79) yields the average queueing times with the mean residual time R given by

$$R = \frac{P_Q}{m\mu}$$

where P_Q is the steady-state probability of queueing given by the Erlang C formula of Eq. (3.36). [Here $\rho_i = \lambda_i/(m\mu)$ and $\rho = \sum_{i=1}^{n} \rho_i$.]

(b) Consider the preemptive resume system. Argue that $W_{(k)}$, defined as the average time in queue averaged over the first k priority classes, is the same as for an $M/M/m$ system with arrival rate $\lambda_1 + \cdots + \lambda_k$ and mean service time $1/\mu$. Use Little's Theorem to

Samsung Exhibit 1015, Page 265 of 573

show that the average time in queue of a k^{th} priority class customer can be obtained recursively from

$$W_1 = W_{(1)}$$

$$W_k = \frac{1}{\lambda_k}\left[W_{(k)}\sum_{i=1}^{k}\lambda_i - W_{(k-1)}\sum_{i=1}^{k-1}\lambda_i\right], \qquad k = 2, 3, \ldots, n$$

3.39 Consider the nonpreemptive priority queueing system of Section 3.5.3 for the case where the available capacity is sufficient to handle the highest-priority traffic but cannot handle the traffic of all priorities, that is,

$$\rho_1 < 1 < \rho_1 + \rho_2 + \cdots + \rho_n$$

Find the average delay per customer of each priority class. *Hint*: Determine the departure rate of the highest-priority class that will experience infinite average delay and the mean residual service time.

3.40 *Optimization of Class Ordering in a Nonpreemptive System.* Consider an n-class, nonpreemptive priority system:

(a) Show that the sum $\sum_{k=1}^{n}\rho_k W_k$ is independent of the priority order of classes, and in fact

$$\sum_{k=1}^{n}\rho_k W_k = \frac{R\rho}{1-\rho}$$

where $\rho = \rho_1 + \rho_2 + \cdots + \rho_n$. (This is known as the $M/G/1$ conservation law [Kle64].) *Hint:* Use Eq. (3.79). Alternatively, argue that $U = R + \sum_{k=1}^{n}\rho_k W_k$, where U is the average steady-state unfinished work in the system (total remaining service time of customers in the system), and U and R are independent of the priority order of the classes.

(b) Suppose that there is a cost c_k per unit time for each class k customer that waits in queue. Show that cost is minimized when classes are ordered so that

$$\frac{\overline{X_1}}{c_1} \le \frac{\overline{X_2}}{c_2} \le \cdots \le \frac{\overline{X_n}}{c_n}$$

Hint: Express the cost as $\sum_{k=1}^{n}(c_k/\overline{X}_k)(\rho_k W_k)$ and use part (a). Also use the fact that interchanging the order of any two adjacent classes leaves the waiting time of all other classes unchanged.

3.41 *Little's Theorem for Arbitrary Order of Service; Analytical Proof [Sti74].* Consider the analysis of Little's Theorem in Section 3.2 and the notation introduced there. We allow the possibility that the initial number in the system is positive [*i.e.*, $N(0) > 0$]. Assume that the time-average arrival and departure rates exist and are equal:

$$\lambda = \lim_{t\to\infty}\frac{\alpha(t)}{t} = \lim_{t\to\infty}\frac{\beta(t)}{t}$$

and that the following limit defining the time-average system time exists:

$$T = \lim_{k\to\infty}\frac{1}{N(0)+\alpha(t)}\left(\sum_{i\in D(t)}T_i + \sum_{i\in\overline{D}(t)}(t-t_i)\right)$$

where $D(t)$ is the set of customers departed by time t and $\overline{D}(t)$ is the set of customers that are in the system at time t. (For all customers that are initially in the system, the time T_i is counted starting at time 0.) Show that regardless of the order in which customers are served, Little's Theorem ($N = \lambda T$) holds with

$$N = \lim_{t \to \infty} \frac{1}{t} \int_0^t N(\tau)\, d\tau$$

Show also that

$$T = \lim_{k \to \infty} \frac{1}{k} \sum_{i=1}^{k} T_i$$

Hint: Take $t \to \infty$ below:

$$\frac{1}{t} \sum_{i \in D(t)} T_i \leq \frac{1}{t} \int_0^t N(\tau)\, d\tau \leq \frac{1}{t} \sum_{i \in D(t) \cup \overline{D}(t)} T_i$$

3.42 *A Generalization of Little's Theorem.* Consider an arrival–departure system with arrival rate λ, where entering customers are forced to pay money to the system according to some rule.

(a) Argue that the following identity holds:

$$\text{Average rate at which the system earns} = \lambda \cdot (\text{Average amount a customer pays})$$

(b) Show that Little's Theorem is a special case.

(c) Consider the $M/G/1$ system and the following cost rule: Each customer pays at a rate of y per unit time when its remaining service time is y, whether in queue or in service. Show that the formula in (a) can be written as

$$W = \lambda \left(\overline{X} W + \frac{\overline{X^2}}{2} \right)$$

which is the Pollaczek–Khinchin formula.

3.43 $M/G/1$ *Queue with Random-Sized Batch Arrivals.* Consider the $M/G/1$ system with the difference that customers are arriving in batches according to a Poisson process with rate λ. Each batch has n customers, where n has a given distribution and is independent of customer service times. Adapt the proof of Section 3.5 to show that the waiting time in queue is given by

$$W = \frac{\lambda \overline{n} \overline{X^2}}{2(1-\rho)} + \frac{\overline{X}(\overline{n^2} - \overline{n})}{2\overline{n}(1-\rho)}$$

Hint: Use the equation $W = R + \rho W + W_B$, where W_B is the average waiting time of a customer for other customers who arrived in the same batch.

3.44 $M/G/1$ *Queue with Overhead for Each Busy Period.* Consider the $M/G/1$ queue with the difference that the service of the first customer in each busy period requires an increment Δ over the ordinary service time of the customer. We assume that Δ has a given distribution and is independent of all other random variables in the model. Let $\rho = \lambda \overline{X}$ be the utilization factor. Show that

(a) $p_0 = P\{\text{the system is empty}\} = (1 - \rho)/(1 + \lambda \overline{\Delta})$.

(b) Average length of busy period $= (\overline{X} + \overline{\Delta})/(1 - \rho)$.

(c) The average waiting time in queue is

Samsung Exhibit 1015, Page 267 of 573

$$W = \frac{\lambda \overline{X^2}}{2(1-\rho)} + \frac{\lambda[\overline{(X+\Delta)^2} - \overline{X^2}]}{2(1+\lambda\overline{\Delta})}$$

(d) Parts (a), (b), and (c) also hold in the case where Δ may depend on the interarrival and service time of the first customer in the corresponding busy period.

3.45 Consider a system that is identical to $M/G/1$ except that when the system empties out, service does not begin again until k customers are present in the system (k is given). Once service begins, it proceeds normally until the system becomes empty again. Show that:

(a) In steady-state:

$$P\{\text{system empty}\} = \frac{1-\rho}{k}$$

$$P\{\text{system nonempty and waiting}\} = \frac{(k-1)(1-\rho)}{k}$$

$$P\{\text{system nonempty and serving}\} = \rho$$

(b) The average length of a busy period is

$$\frac{\rho + k - 1}{\lambda(1-\rho)}$$

Verify that this average length is equal to the average time between arrival and start of service of the first customer in a busy period, plus k times the average length of a busy period for the corresponding $M/G/1$ system ($k = 1$).

(c) Suppose that we divide a busy period into a busy/waiting portion and a busy/serving portion. Show that the average number in the system during a busy/waiting portion is $k/2$ and the average number in the system during a busy/serving portion is

$$\frac{N_{M/G/1}}{\rho} + \frac{k-1}{2}$$

where $N_{M/G/1}$ is the average number in the system for the corresponding $M/G/1$ system ($k = 1$). *Hint*: Relate a busy/serving portion of a busy period with k independent busy periods of the corresponding $M/G/1$ system where $k = 1$.

(d) The average number in the system is

$$N_{M/G/1} + \frac{k-1}{2}$$

3.46 *Single-Vacation $M/G/1$ System.* Consider the $M/G/1$ system with the difference that each busy period is followed by a single vacation interval. Once this vacation is over, an arriving customer to an empty system starts service immediately. Assume that vacation intervals are independent, identically distributed, and independent of the customer interarrival and service times. Prove that the average waiting time in queue is

$$W = \frac{\lambda \overline{X^2}}{2(1-\rho)} + \frac{\overline{V^2}}{2I}$$

where I is the average length of an idle period, and show how to calculate I.

3.47 *The $M/G/\infty$ System.* Consider a queueing system with Poisson arrivals at rate λ. There are an infinite number of servers, so that each arrival starts service at an idle server immediately on arrival. Each server has a general service time distribution and $F_X(x) = P\{X \leq x\}$

Samsung Exhibit 1015, Page 268 of 573

denotes the probability that a service starting at any given time τ is completed by time $\tau + x$ [$F_X(x) = 0$ for $x \leq 0$]. The servers have independent and identical service time distributions.

(a) For x and δ ($0 < \delta < x$) very small, find the probability that there was an arrival in the interval $[\tau - x, \tau - x + \delta]$ *and* that this arrival is still being served at time τ.

(b) Show that the mean service time for any arrival is given by

$$\overline{X} = \int_0^\infty [1 - F_X(x)]\, dx$$

Hint: Use a graphical argument.

(c) Use parts (a) and (b) to verify that the number in the system is Poisson distributed with mean $\lambda \overline{X}$.

3.48 *An Improved Bound for the $G/G/1$ Queue.*

(a) Let r be a nonnegative random variable and let x be a nonnegative scalar. Show that

$$\frac{\overline{\big(\max\{0, r - x\}\big)^2}}{\overline{\big(\max\{0, r - x\}\big)}^2} \geq \frac{\overline{r^2}}{(\overline{r})^2}$$

where overbar denotes expected value. *Hint*: Prove that the left-hand expression is monotonically nondecreasing as a function of x.

(b) Using the notation of Section 3.5.4, show that

$$\sigma_I^2 \geq (1 - \rho)^2 \sigma_a^2$$

and that

$$W \leq \frac{\lambda(\sigma_a^2 + \sigma_b^2)}{2(1 - \rho)} - \frac{\lambda(1 - \rho)\sigma_a^2}{2}$$

Hint: Use part (a) with r being the customer interarrival time and x equal to the time in the system [cf. Eq. (3.93)].

3.49 *Last-Come First-Serve $M/G/1$ System.* Consider an $M/G/1$ system with the difference that upon arrival at the queue, a customer goes immediately into service, replacing the customer who is in service at the time (if any) on a preemptive-resume basis. When a customer completes service, the customer most recently preempted resumes service. Show that:

(a) The expected length of a busy period, denoted $E\{B\}$, is the same as in the ordinary $M/G/1$ queue.

(b) Show that the expected time in the system of a customer is equal to $E\{B\}$. *Hint*: Argue that a customer who starts a busy period stays in the system for the entire duration of the busy period.

(c) Let C be the average time in the system of a customer requiring one unit of service time. Argue that the average time in the system of a customer requiring X units of service time is XC. *Hint*: Argue that a customer requiring two units of service time is "equivalent" to two customers with one unit service time each, and with the second customer arriving at the time that the first departs.

(d) Show that

$$C = \frac{E\{B\}}{E\{X\}} = \frac{1}{1 - \rho}$$

Samsung Exhibit 1015, Page 269 of 573

3.50 *Truncation of Queues.* This problem illustrates one way to use simple queues to obtain results about more complicated queues.

(a) Consider a continuous-time Markov chain with state space S, stationary distribution $\{p_j\}$, and transition rates q_{ij}. Suppose that we have a truncated version of this chain, that is, a new chain with space $\overline{S}$, which is a subset of S and has the same transition rates q_{ij} between states i and j of $\overline{S}$. Assume that for all $j \in \overline{S}$, we have

$$p_j \sum_{i \notin \overline{S}} q_{ji} = \sum_{i \notin \overline{S}} p_i q_{ij}$$

Show that if the truncated chain is irreducible, then its stationary distribution $\{\overline{p}_j\}$ satisfies $\overline{p}_j = p_j / \sum_{i \in \overline{S}} p_i$ for all $j \in \overline{S}$. (Note that $\overline{p}_j$ is the conditional probability for the state of the original chain to be j conditioned on the fact that it lies within $\overline{S}$.)

(b) Show that the condition of part (a) on the stationary distribution $\{p_j\}$ and the transition rates $\{q_{ij}\}$ is satisfied if the original chain is time reversible, and that in this case, the truncated chain is also time reversible.

(c) Consider two queues with independent Poisson arrivals and independent exponentially distributed service times. The arrival and service rates are denoted λ_i, μ_i, for $i = 1, 2$, respectively. The two queues share a waiting room with finite capacity B (including customers in service). Arriving customers that find the waiting room full are lost. Use part (b) to show that the system is reversible and that for $m + n \leq B$, the steady-state probabilities are

$$P\{m \text{ in queue } 1,\ n \text{ in queue } 2\} = \frac{\rho_1^m \rho_2^n}{G}$$

where $\rho_i = \lambda_i / \mu_i$, $i = 1, 2$, and G is a normalizing constant.

3.51 *Decomposition/Aggregation of Reversible Chains.* Consider a time reversible continuous-time Markov chain in equilibrium, with state space S, transition rates q_{ij}, and stationary probabilities p_j. Let $S = \cup_{k=1}^{K} S_k$ be a partition of S in mutually disjoint sets, and denote for all k and $j \in S_k$:

u_k = Probability of the state being in S_k (*i.e.*, $u_k = \sum_{j \in S_k} p_j$)

π_j = Probability of the state being equal to j conditioned on the fact that the state belongs to S_k (*i.e.*, $\pi_j = P\{X_n = j \mid X_n \in S_k\} = p_j / u_k$)

Assume that all states in S_k communicate with all other states in S_k.

(a) Show that $\{\pi_j \mid j \in S_k\}$ is the stationary distribution of the truncated chain with state space S_k (cf. Problem 3.50).

(b) Show that $\{u_k \mid k = 1, \ldots, K\}$ is the stationary distribution of the so-called *aggregate chain*, which is the Markov chain with states $k = 1, \ldots, K$ and transition rates

$$\tilde{q}_{km} = \sum_{j \in S_k,\ i \in S_m} \pi_j q_{ji}, \qquad k, m = 1, \ldots, K$$

Show also that the aggregate chain is reversible. (Note that the aggregate chain corresponds to a fictitious process; the actual process, corresponding to transitions between sets of states, need not be Markov.)

(c) Outline a divide-and-conquer solution method that first solves for the distributions of the truncated chains and then solves for the distribution of the aggregate chain. Apply this method to Examples 3.12 and 3.13.

Samsung Exhibit 1015, Page 270 of 573

(d) Suppose that the truncated chains are reversible but the original chain is not. Show that the results of parts (a) and (b) hold except that the aggregate chain need not be reversible.

3.52 *An Extension of Burke's Theorem.* Consider an $M/M/1$ system in steady state where customers are served in the order that they arrive. Show that given that a customer departs at time t, the arrival time of that customer is independent of the departure process prior to t. *Hint*: Consider a customer arriving at time t_1 and departing at time t_2. In reversed system terms, the arrival process is independent Poisson, so the arrival process to the left of t_2 is independent of the times spent in the system of customers that arrived at or to the right of t_2.

3.53 Consider the model of two queues in tandem of Section 3.7 and assume that customers are served at each queue in the order they arrive.

(a) Show that the times (including service) spent by a customer in queue 1 and in queue 2 are mutually independent, and independent of the departure process from queue 2 prior to the customer's departure from the system. *Hint:* By Burke's Theorem, the time spent by a customer in queue 1 is independent of the sequence of arrival times at queue 2 prior to the customer's arrival at queue 2. These arrival times (together with the corresponding independent service times) determine the time the customer spends at queue 2 as well as the departure process from queue 2 prior to the customer's departure from the system.

(b) Argue by example that the times a customer spends waiting *before entering service* at the two queues are *not* independent.

3.54 Use reversibility to characterize the departure process of the $M/M/1/m$ queue.

3.55 Consider the feedback model of a CPU and I/O device of Example 3.19 with the difference that the CPU consists of m identical parallel processors. The service time of a job at each parallel processor is exponentially distributed with mean $1/\mu_1$. Derive the stationary distribution of the system.

3.56 Consider the discrete-time approximation to the $M/M/1$ queue of Fig. 3.6. Let X_n be the state of the system at time $n\Delta$ and let D_n be a random variable taking on the value 1 if a departure occurs between $n\Delta$ and $(n+1)\Delta$, and the value 0 if no departure occurs. Assume that the system is in steady-state at time $n\Delta$. Answer the following without using reversibility.

(a) Find $P\{X_n = i, D_n = j\}$ for $i \geq 0$, $j = 0, 1$.

(b) Find $P\{D_n = 1\}$.

(c) Find $P\{X_n = i, D_n = 1\}$ for $i \geq 0$.

(d) Find $P\{X_{n+1} = i, D_n = 1\}$ and show that X_{n+1} is statistically independent of D_n. *Hint*: Use part (c); also show that $P\{X_{n+1} = i\} = P\{X_{n+1} = i \mid D_n = 1\}$ for all $i \geq 0$ is sufficient to show independence.

(e) Find $P\{X_{n+k} = i, D_{n+1} = j \mid D_n\}$ and show that the pair of variables (X_{n+1}, D_{n+1}) is statistically independent of D_n.

(f) For each $k > 1$, find $P\{X_{n+k} = i, D_{n+k} = j \mid D_{n+k-1}, D_{n+k-2}, \ldots, D_n\}$ and show that the pair (X_{n+k}, D_{n+k}) is statistically independent of $(D_{n+k-1}, D_{n+k-2}, \ldots, D_n)$. *Hint*: Use induction on k.

(g) Deduce a discrete-time analog to Burke's Theorem.

3.57 Consider the network in Fig. 3.39. There are four sessions: ACE, ADE, BCEF, and BDEF sending Poisson traffic at rates 100, 200, 500, and 600 packets/min, respectively. Packet lengths are exponentially distributed with mean 1000 bits. All transmission lines have capac-

ity 50 kbits/sec, and there is a propagation delay of 2 msec on each line. Using the Kleinrock independence approximation, find the average number of packets in the system, the average delay per packet (regardless of session), and the average delay per packet of each session.

3.58 *Jackson Networks with a Limit on the Total Number of Customers.* Consider an open Jackson network as described in the beginning of Section 3.8, with the difference that all customers who arrive when there are a total of M customers in the network are blocked from entering and are lost for the system. Derive the stationary distribution. *Hint*: Convert the system into a closed network with M customers by introducing an additional queue $K+1$ with service rate equal to $\sum_{j=1}^{K} r_j$. A customer exiting queue $i \in \{1, \ldots, K\}$ enters queue $K+1$ with probability $1 - \sum_j P_{ij}$, and a customer exiting queue $K+1$ enters queue $i \in \{1, \ldots, K\}$ with probability $r_i / \sum_{j=1}^{K} r_j$.

3.59 Justify the Arrival Theorem for closed networks by inserting a very fast $M/M/1$ queue between every pair of queues. Argue that conditioning on a customer moving from one queue to another is essentially equivalent to conditioning on a single customer being in the fast $M/M/1$ queue that lies between the two queues.

3.60 Consider a closed Jackson network where the service time at each queue is independent of the number of customers at the queue. Suppose that for a given number of customers, the utilization factor of one of the queues, say queue 1, is strictly larger than the utilization factors of the other queues. Show that as the number of customers increases, the proportion of time that a customer spends in queue 1 approaches unity.

3.61 Consider a model of a computer CPU connected to m I/O devices as shown in Fig. 3.40. Jobs enter the system according to a Poisson process with rate λ, use the CPU and with

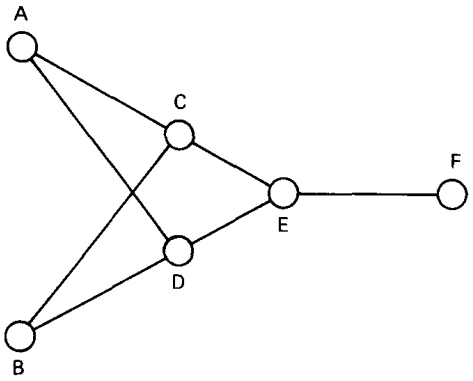


Figure 3.39 Network of transmission lines for Problem 3.57.

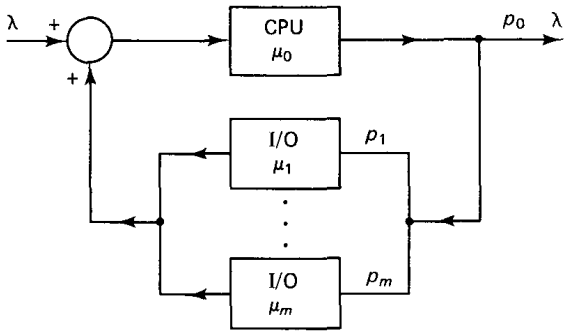


Figure 3.40 Model of a computer CPU connected to m I/O devices for Problem 3.61.

probability p_i, $i = 1, \ldots, m$, are routed to the i^{th} I/O device, while with probability p_0 they exit the system. The service time of a job at the CPU (or the i^{th} I/O device) is exponentially distributed with mean $1/\mu_0$ (or $1/\mu_i$, respectively). We assume that all job service times at all queues are independent (including the times of successive visits to the CPU and I/O devices of the same job). Find the occupancy distribution of the system and construct an "equivalent" system with $m+1$ queues in tandem that has the same occupancy distribution.

3.62 Consider a closed version of the queueing system of Problem 3.61, shown in Fig. 3.41. There are M jobs in the system at all times. A job uses the CPU and with probability p_i, $i = 1, \ldots, m$, is routed to the i^{th} I/O device. The service time of a job at the CPU (or the i^{th} I/O device) is exponentially distributed with mean $1/\mu_0$ (or $1/\mu_i$, respectively). We assume that all job service times at all queues are independent (including the times of successive visits to the CPU and I/O devices of the same job). Find the arrival rate of jobs at the CPU and the occupancy distribution of the system.

3.63 *Bounds on the Throughput of a Closed Queueing Network.* Packets enter the network of transmission lines shown in Fig. 3.42 at point A and exit at point B. A packet is first transmitted on one of the lines $L_1, \ldots, L_K$, where it requires on the average a transmission time $\overline{X}$, and is then transmitted in line L_{K+1}, where it requires on the average a transmission time $\overline{Y}$. To effect flow control, a maximum of $N \geq K$ packets are admitted into the system.

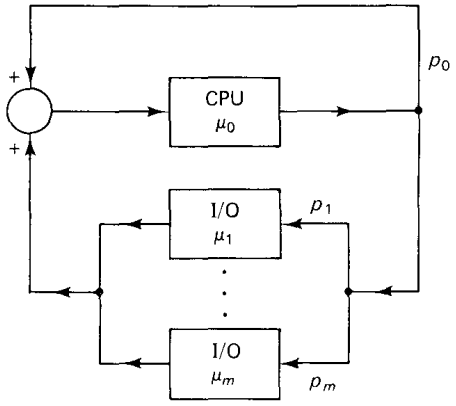


Figure 3.41 Closed queueing system for Problem 3.62.

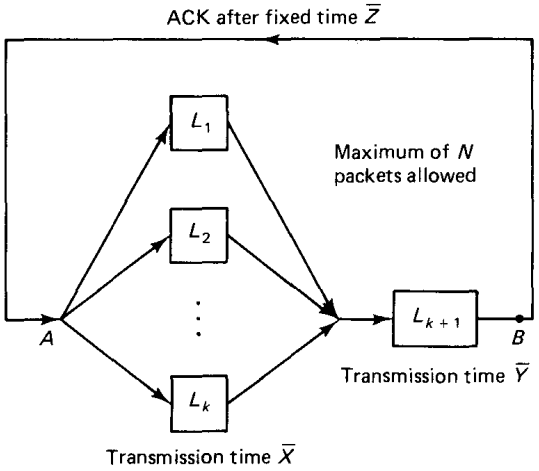


Figure 3.42 Closed queueing network for Problem 3.63.

Samsung Exhibit 1015, Page 273 of 573

Each time a packet exits the system at point B, an acknowledgment is sent back and reaches point A after a fixed time $\overline{Z}$. At that time, a new packet is allowed to enter the system. Use Little's Theorem to find upper and lower bounds for the system throughput under two circumstances:

(a) The method of routing a packet to one of the lines $L_1, \ldots, L_K$ is unspecified.

(b) The routing method is such that whenever one of the lines $L_1, \ldots, L_K$ is idle, there is no packet waiting at any of the other lines.

3.64 Consider the closed queueing network in Fig. 3.43. There are three customers who are doomed forever to cycle between queue 1 and queue 2. The service times at the queues are independent and exponentially distributed with mean μ_1 and μ_2. Assume that $\mu_2 < \mu_1$.

(a) The system can be represented by a four-state Markov chain. Find the transition rates of the chain.

(b) Find the steady-state probabilities of the states.

(c) Find the customer arrival rate at queue 1.

(d) Find the rate at which a customer cycles through the system.

(e) Show that the Markov chain is reversible. What does a departure from queue 1 in the forward process correspond to in the reversed process? Can the transitions of a single customer in the forward proces be associated with transitions of a single customer in the reverse process?

3.65 Consider the closed queueing network of Section 3.8.2 and assume that the service rate $\mu_j(m)$ at the j^{th} queue is independent of the number of customers m in the queue [$\mu_j(m) = \mu_j$ for all m]. Show that the utilization factor $U_j(M) = \lambda_j(M)/\mu_j$ of the j^{th} queue is given by

$$U_j(M) = \rho_j \frac{G(M-1)}{G(M)}$$

where $\rho_j = \overline{\lambda}_j/\mu_j$ (compare with Examples 3.21 and 3.22).

3.66 $M/M/1$ *System with Multiple Classes of Customers.* Consider an $M/M/1$-like system with first-come first-serve service and multiple classes of customers denoted $c = 1, 2, \ldots, C$. Let λ_i and μ_i be the arrival and service rate of class i.

(a) Model this system by a Markov chain and show that unless $\mu_1 = \mu_2 = \cdots = \mu_C$, its steady-state distribution does not have a product form. *Hint:* Consider a state $z = (c_1, c_2, \ldots, c_n)$ such that $\mu_{c_1} \neq \mu_{c_n}$. Write the global balance equations for state z.

(b) Suppose instead that the service discipline is last-come first-serve (as defined in Problem 3.49). Model the system by a Markov chain and show that the steady-state distribution has the product form

$$P(z) = P(c_1, c_2, \ldots, c_n) = \frac{\rho_{c_1}\rho_{c_2}\cdots\rho_{c_n}}{G}$$

where $\rho_c = \lambda_c/\mu_c$ and G is a normalizing constant.

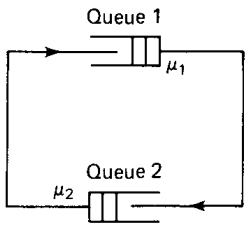


Figure 3.43 Closed queueing network for Problem 3.64.

APPENDIX A: REVIEW OF MARKOV CHAIN THEORY

The purpose of this appendix is to provide a brief summary of the results we need from discrete- and continuous-time Markov chain theory. We refer the reader to books on stochastic processes for detailed accounts.

3A.1 Discrete-Time Markov Chains

Consider a discrete-time stochastic process $\{X_n \mid n = 0, 1, 2, \ldots\}$ that takes values from the set of nonnegative integers, so the states that the process can be in are $i = 0, 1 \ldots$. The process is said to be a *Markov chain* if whenever it is in state i, there is a fixed probability P_{ij} that it will next be in state j regardless of the process history prior to arriving at i. That is, for all $n > 0$, $i_{n-1}, \ldots, i_0, i, j$,

$$\begin{aligned} P_{ij} &= P\{X_{n+1} = j \mid X_n = i,\ X_{n-1} = i_{n-1}, \ldots, X_0 = i_0\} \\ &= P\{X_{n+1} = j \mid X_n = i\} \end{aligned}$$

We refer to P_{ij} as the *transition probabilities*. They must satisfy

$$P_{ij} \geq 0\,, \sum_{j=0}^{\infty} P_{ij} = 1\,, \qquad i = 0, 1, \ldots$$

The corresponding transition probability matrix is denoted

$$P = \begin{bmatrix} P_{00} & P_{01} & P_{02} & \cdots \\ P_{10} & P_{11} & P_{12} & \cdots \\ \cdot & \cdot & \cdot & \cdot \\ P_{i0} & P_{i1} & P_{i2} & \cdots \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}$$

We will concentrate on the case where the number of states is infinite. There are analogous notions and results for the case where the number of states is finite.

Consider the n-step transition probabilities

$$P_{ij}^n = P\{X_{n+m} = j \mid X_m = i\}\,, \qquad n \geq 0, i, j \geq 0\,.$$

The *Chapman–Kolmogorov equations* provide a method for calculating P_{ij}^n. They are given by

$$P_{ij}^{n+m} = \sum_{k=0}^{\infty} P_{ik}^n P_{kj}^m\,, \qquad n, m \geq 0, i, j \geq 0$$

From these equations, we see that P_{ij}^n are the elements of the matrix P^n (the transition probability matrix P raised to the n^{th} power).

We say that two states i and j *communicate* if for some n and n', we have $P_{ij}^n > 0$ and $P_{ji}^{n'} > 0$. If all states communicate, we say that the Markov chain is *irreducible*.

We say that a state i of a Markov chain is *periodic* if there exists some integer $m \geq 1$ such that $P_{ii}^m > 0$ and some integer $d > 1$ such that $P_{ii}^n > 0$ only if n is a multiple of d. A Markov chain is said to be *aperiodic* if none of its states is periodic. A probability distribution $\{p_j \mid j \geq 0\}$ is said to be a *stationary distribution* for the Markov chain if

$$p_j = \sum_{i=0}^{\infty} p_i P_{ij} \qquad j = 0, 1, \ldots \tag{3A.1}$$

We will restrict attention to irreducible and aperiodic Markov chains, since this is the only type we will encounter. For such a chain, it can be shown that the limit

$$p_j = \lim_{n \to \infty} P\{X_n = j \mid X_0 = i\}, \qquad i = 0, 1, \ldots$$

exists and is independent of the starting state $X_0 = i$. Furthermore, we have (with probability 1)

$$p_j = \lim_{k \to \infty} \frac{\text{Number of visits to state } j \text{ up to time } k}{k}$$

which leads to the interpretation that p_j is the proportion of time or the frequency with which the process visits j, a time average interpretation. Note again that this frequency does not depend on the starting state. The following result will be of primary interest:

Theorem. In an irreducible, aperiodic Markov chain, there are two possibilities for the scalars $p_j = \lim_{n \to \infty} P\{X_n = j \mid X_0 = i\}$:

1. $p_j = 0$ for all $j \geq 0$, in which case the chain has no stationary distribution.
2. $p_j > 0$ for all $j \geq 0$, in which case $\{p_j \mid j \geq 0\}$ is the unique stationary distribution of the chain [*i.e.*, it is the only probability distribution satisfying Eq. (3A.1)].

A typical example of case 1 is a queueing system where the arrival rate exceeds the service rate, and the number of customers in the system increases to ∞, so the steady-state probability p_j of having any finite number of customers j in the system is zero. Note that case 1 never arises when the number of states is finite. In particular, for every irreducible and aperiodic Markov chain with states $j = 0, 1, \ldots, m$, there exists a unique probability distribution $\{p_j \mid j = 0, 1, \ldots, m\}$ satisfying $p_j = \sum_{i=0}^{m} p_i P_{ij}$ and $p_j > 0$ for all j.

In case 2, there arises the issue of characterizing the stationary distribution $\{p_j \mid j \geq 0\}$. For queueing systems, the following equations are often useful. Multiplying the equation $\sum_{i=0}^{\infty} P_{ji} = 1$ by p_j and using Eq. (3A.1), we have

$$p_j \sum_{i=0}^{\infty} P_{ji} = \sum_{i=0}^{\infty} p_i P_{ij}, \qquad j = 0, 1, \ldots \tag{3A.2}$$

These equations are known as the *global balance equations*. Note that $p_i P_{ij}$ may be viewed as the long-term frequency of transitions from i to j. Thus the global balance equations state that at equilibrium, the frequency of transitions out of j [left side of Eq. (3A.2)] equals the frequency of transitions into j [right side of Eq. (3A.2)].

Samsung Exhibit 1015, Page 276 of 573

A typical approach for finding the stationary distribution of an irreducible, aperiodic Markov chain is to try to solve the global balance equations. If a distribution satisfying these equations is found, then by the preceding theorem, it must be the stationary distribution.

The global balance equations can be generalized to apply to an entire set of states. Consider a subset of states S. By adding Eq. (3A.2) over all $j \in S$, we obtain

$$\sum_{j \in S} p_j \sum_{i \notin S} P_{ji} = \sum_{i \notin S} p_i \sum_{j \in S} P_{ij} \tag{3A.3}$$

An intuitive explanation of these equations is based on the fact that when the Markov chain is irreducible, the state (with probability 1) will return to the set S infinitely many times. Therefore, for each transition out of S there must be (with probability 1) a reverse transition into S at some later time. As a result, the frequency of transitions out of S equals the frequency of transitions into S. This is precisely the meaning of the global balance equations (3A.3).

3A.2 Detailed Balance Equations

As an application of the global balance equations, consider a Markov chain typical of queueing systems and, more generally, *birth–death* systems where two successive states can only differ by unity, that is, $P_{i,j} = 0$ if $|i - j| > 1$, cf. Fig. 3A.1. We assume that $P_{i,i+1} > 0$ and $P_{i+1,i} > 0$ for all i. This is a necessary and sufficient condition for the chain to be irreducible. Consider the sets of states

$$S = \{0, 1, \ldots, n\}$$

Application of Eq. (3A.3) yields

$$p_n P_{n,n+1} = p_{n+1} P_{n+1,n}\,, \qquad n = 0, 1, \ldots \tag{3A.4}$$

(*i.e.*, in steady-state), the frequency of transitions from n to $n + 1$ equals the frequency of transitions from $n + 1$ to n. These equations can be very useful in computing the stationary distribution $\{p_j \mid j \geq 0\}$ (see Sections 3.3 and 3.4).

Equation (3A.4) is a special case of the equations

$$p_j P_{ji} = p_i P_{ij}\,, \qquad i, j \geq 0 \tag{3A.5}$$

known as the *detailed balance equations*. These equations imply the global balance equations (3A.2) but need not hold in any given Markov chain. However, in many important special cases, they do hold and greatly simplify the calculation of the stationary

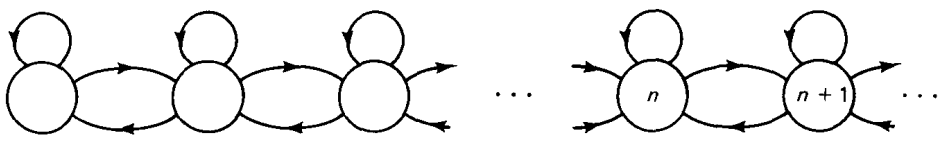


Figure 3A.1 Transition probability diagram for a birth–death process.

distribution. A common approach is to hypothesize the validity of the detailed balance equations and to try to solve them for the steady-state probabilities p_j, $j \geq 0$. There are two possibilities; either the system (3A.5) together with $\sum_j p_j = 1$ is inconsistent or else a distribution $\{p_j \mid j \geq 0\}$ satisfying Eq. (3A.5) will be found. In the latter case, this distribution will also satisfy the global balance equations (3A.2), so by the theorem given earlier, it is the unique stationary distribution.

3A.3 Partial Balance Equations

Some Markov chains have the property that their stationary distribution $\{p_j \mid j \geq 0\}$ satisfies a set of equations which is intermediate between the global and the detailed balance equations. For every node j, consider a partition $S_j^1, \ldots, S_j^k$ of the complementary set of nodes $\{i \mid i \geq 0, i \neq j\}$ and the equations

$$p_j \sum_{i \in S_j^m} P_{ji} = \sum_{i \in S_j^m} p_i P_{ij}\,, \qquad m = 1, 2, \ldots, k \tag{3A.6}$$

Equations of the form above are known as a set of *partial balance equations*. If a distribution $\{p_j \mid j \geq 0\}$ solves a set of partial balance equations, it will also solve the global balance equations, so it will be the unique stationary distribution of the chain. A technique that often proves useful is to guess the right set of partial balance equations satisfied by the stationary distribution and then proceed to solve them.

3A.4 Continuous-Time Markov Chains

A continuous-time Markov chain is a process $\{X(t) \mid t \geq 0\}$ taking values from the set of states $i = 0, 1, \ldots$ that has the property that each time it enters state i:

1. The time it spends in state i is exponentially distributed with parameter ν_i. We may view ν_i as the rate (in transitions/sec) at which the process makes a transition when at state i.
2. When the process leaves state i, it will enter state j with probability P_{ij}, where $\sum_j P_{ij} = 1$.

We may view

$$q_{ij} = \nu_i P_{ij}$$

as the rate (in transitions/sec) at which the process makes a transition to j when at state i. Consequently, we call q_{ij} the *transition rate* from i to j.

We will be interested in chains for which the discrete-time Markov chain with transition probabilities P_{ij} (called the *embedded chain*) is irreducible. We also require a technical condition, namely that the number of transitions in any finite length of time is finite with probability 1; chains with this property are called *regular*. (Nonregular chains almost never arise in queueing systems of interest. For an example, see [Ros83], p. 142.)

Under the preceding conditions, it can be shown that the limit

$$p_j = \lim_{t \to \infty} P\{X(t) = j \mid X(0) = i\} \tag{3A.7}$$

exists and is independent of the initial state i. We refer to p_j as the steady-state occupancy probability of state j. It can be shown that if $T_j(t)$ is the time spent in state j up to time t, then, regardless of the initial state, we have with probability 1,

$$p_j = \lim_{t \to \infty} \frac{T_j(t)}{t} \tag{3A.8}$$

that is, p_j can be viewed as the long-term proportion of time the process spends in state j. It can be shown also that either the occupancy probabilities are all zero or else they are all positive and they sum to unity. Queueing systems where the arrival rate is larger than the service rate provide examples where all occupancy probabilities are zero.

The *global balance equations* for a continuous-time Markov chain take the form

$$p_j \sum_{i=0}^{\infty} q_{ji} = \sum_{i=0}^{\infty} p_i q_{ij}, \qquad j = 0, 1, \ldots \tag{3A.9}$$

It can be shown that if a probability distribution $\{p_j \mid j \geq 0\}$ satisfies these equations, then each p_j is the steady-state occupancy probability of state j.

To interpret the global balance equations, we note that since p_i is the proportion of time the process spends in state i, it follows that $p_i q_{ij}$ is the frequency of transitions from i to j (average number of transitions from i to j per unit time). It is seen therefore that the global balance equations (3A.9) express the natural fact that the frequency of transitions out of state j (the left-side term $p_j \sum_{i=1}^{\infty} q_{ji}$) is equal to the frequency of transitions into state j (the right-side term $\sum_{i=0}^{\infty} p_i q_{ij}$).

The continuous-time analog of the detailed balance equations for discrete-time chains is

$$p_j q_{ji} = p_i q_{ij}, \qquad i, j = 0, 1, \ldots$$

These equations hold in birth–death systems where $q_{ij} = 0$ for $|i - j| > 1$, but need not hold in other types of Markov chains. They express the fact that the frequencies of transitions from i to j and from j to i are equal. One can also write a set of partial balance equations and attempt to solve them for the distribution $\{p_j \mid j \geq 0\}$. If a solution can be found, it provides the stationary distribution of the continuous chain.

To understand the relationship between the global balance equations (3A.9) for continuous-time chains and the global balance equations (3A.2) for discrete-time chains, consider any $\delta > 0$, and the discrete-time Markov chain $\{X_n \mid n \geq 0\}$, where

$$X_n = X(n\delta), \qquad n = 0, 1, \ldots$$

The stationary distribution of $\{X_n\}$ is clearly $\{p_j \mid j \geq 0\}$, the occupancy distribution of the continuous chain [cf. Eq. (3A.7)]. The transition probabilities of $\{X_n \mid n \geq 0\}$ can be derived by using the properties of the exponential distribution and a derivation which is very similar to the one used in Section 3.3.1 for the Markov chain of the $M/M/1$ queue. We obtain

$$\overline{P}_{ij} = \delta q_{ij} + o(\delta), \qquad i \neq j$$

$$\overline{P}_{jj} = 1 - \delta \sum_{\substack{i=0 \\ i \neq j}}^{\infty} q_{ji} + o(\delta)$$

Using these expressions and Eq. (3A.1), which is equivalent to the global balance equations for the discrete chain, we obtain

$$p_j = \sum_{i=0}^{\infty} p_i \overline{P}_{ij} = p_j\Big(1 - \delta \sum_{\substack{i=0 \\ i \neq j}}^{\infty} q_{ji} + o(\delta)\Big) + \sum_{\substack{i=0 \\ i \neq j}} p_i\big(\delta q_{ij} + o(\delta)\big)$$

and dividing by δ and letting $\delta \to 0$, we obtain the global balance equations (3A.9) for the continuous chain.

3A.5 Drift and Stability

Suppose that we are given an irreducible, aperiodic, discrete-time Markov chain. In many situations one is particularly interested in whether the chain has a stationary distribution $\{p_j\}$. In this case, $p_j > 0$ for all j and the chain is "stable" in the sense that all states are visited infinitely often with probability 1. The notion of *drift*, defined as

$$D_i = E\{X_{n+1} - X_n \mid X_n = i\} = \sum_{k=-i}^{\infty} k P_{i(i+k)}, \qquad i = 0, 1, \ldots$$

is particularly useful in this respect. Roughly speaking, the sign of D_i indicates whether, starting at i, the state tends to increase ($D_i > 0$) or decrease ($D_i < 0$). Intuitively, the chain will be stable if the drift is negative for all large enough states. This is established in the following lemma:

Stability Lemma [Pak69]. Suppose that $D_i < \infty$ for all i, and that for some scalar $\delta > 0$ and integer $\bar{i} \geq 0$ we have

$$D_i \leq -\delta, \qquad \text{for all } i > \bar{i}$$

Then the Markov chain has a stationary distribution.

Proof: Let $\beta = \max_{i \leq \bar{i}} D_i$. We have for each state i

$$\begin{aligned} E\{X_n \mid X_0 = i\} - i &= \sum_{k=1}^{n} E\{X_k - X_{k-1} \mid X_0 = i\} \\ &= \sum_{k=1}^{n} \sum_{j=0}^{\infty} E\{X_k - X_{k-1} \mid X_{k-1} = j\} P\{X_{k-1} = j \mid X_0 = i\} \\ &\leq \sum_{k=1}^{n} \left[\beta \sum_{j=0}^{\bar{i}} P\{X_{k-1} = j \mid X_0 = i\} \right. \\ &\quad \left. -\delta \left(1 - \sum_{j=0}^{\bar{i}} P\{X_{k-1} = j \mid X_0 = i\} \right) \right] \end{aligned}$$

$$= (\beta + \delta) \sum_{k=1}^{n} \sum_{j=0}^{\bar{i}} P\{X_{k-1} = j \mid X_0 = i\} - n\delta$$

from which we obtain

$$0 \leq E\{X_n \mid X_0 = i\} \leq n \left[(\beta + \delta) \sum_{j=0}^{\bar{i}} \left(n^{-1} \sum_{k=1}^{n} P\{X_{k-1} = j \mid X_0 = i\} \right) - \delta \right] + i$$

Dividing by n and taking the limit as $n \to \infty$, we obtain

$$0 \leq (\beta + \delta) \sum_{j=0}^{\bar{i}} p_j - \delta$$

which implies that $p_j > 0$ for some $j \in \{0, \ldots, \bar{i}\}$. Since the chain is assumed irreducible and aperiodic, it follows from the theorem of Section 3A.1 that there exists a stationary distribution. **Q.E.D.**

We also give without proof a converse to the preceding lemma:

Instability Lemma [Kap79]. Suppose that there exist integers $\bar{i} \geq 0$ and k such that

$$D_i > 0, \qquad \text{for all } i > \bar{i}$$

and

$$P_{ij} = 0, \qquad \text{for all } i \text{ and } j \text{ such that } 0 \leq j \leq i - k$$

Then the Markov chain does not have a stationary distribution; that is, $p_j = 0$ for all j.

APPENDIX B: SUMMARY OF RESULTS

Notation

p_n = Steady-state probability of having n customers in the system

λ = Arrival rate (inverse of average interarrival time)

μ = Service rate (inverse of average service time)

N = Average number of customers in the system

N_Q = Average number of customers waiting in queue

T = Average customer time in the system

W = Average customer waiting time in queue (does not include service time)

$\overline{X}$ = Average service time

$\overline{X^2}$ = Second moment of service time

Little's Theorem

$$N = \lambda T$$

$$N_Q = \lambda W$$

Poisson distribution with parameter *m*

$$p_n = \frac{e^{-m} m^n}{n!}, \quad n = 0, 1, \ldots$$

$$\text{Mean} = \text{Variance} = m$$

Exponential distribution with parameter λ

$$P\{\tau \leq s\} = 1 - e^{-\lambda s}, \qquad s \geq 0$$

$$\text{Density: } p(\tau) = \lambda e^{-\lambda \tau}$$

$$\text{Mean} = \frac{1}{\lambda}$$

$$\text{Variance} = \frac{1}{\lambda^2}$$

Summary of *M*/*M*/1 system results

1. Utilization factor (proportion of time the server is busy)
$$\rho = \frac{\lambda}{\mu}$$
2. Probability of n customers in the system
$$p_n = \rho^n (1 - \rho), \qquad n = 0, 1, \ldots$$
3. Average number of customers in the system
$$N = \frac{\rho}{1 - \rho}$$
4. Average customer time in the system
$$T = \frac{\rho}{\lambda(1 - \rho)} = \frac{1}{\mu - \lambda}$$
5. Average number of customers in queue
$$N_Q = \frac{\rho^2}{1 - \rho}$$
6. Average waiting time in queue of a customer
$$W = \frac{\rho}{\mu - \lambda}$$

Summary of *M*/*M*/*m* system results

1. Ratio of arrival rate to maximal system service rate
$$\rho = \frac{\lambda}{m\mu}$$

2. Probability of n customers in the system

$$p_0 = \left[\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!(1-\rho)}\right]^{-1}, \qquad n = 0$$

$$p_n = \begin{cases} p_0 \dfrac{(m\rho)^n}{n!}, & n \le m \\ p_0 \dfrac{m^m \rho^n}{m!}, & n > m \end{cases}$$

3. Probability that an arriving customer has to wait in queue (m customers or more in the system)

$$P_Q = \frac{p_0(m\rho)^m}{m!(1-\rho)} \qquad \text{(Erlang C Formula)}$$

4. Average waiting time in queue of a customer

$$W = \frac{\rho P_Q}{\lambda(1-\rho)}$$

5. Average number of customers in queue

$$N_Q = \frac{\rho P_Q}{1-\rho}$$

6. Average customer time in the system

$$T = \frac{1}{\mu} + W$$

7. Average number of customers in the system

$$N = m\rho + \frac{\rho P_Q}{1-\rho}$$

Summary of *M*/*M*/*m*/*m* system results

1. Probability of m customers in the system

$$p_0 = \left[\sum_{n=0}^{m} \left(\frac{\lambda}{\mu}\right)^n \frac{1}{n!}\right]^{-1}$$

$$p_n = p_0 \left(\frac{\lambda}{\mu}\right)^n \frac{1}{n!}, \qquad n = 1, 2, \ldots, m$$

2. Probability that an arriving customer is lost

$$p_m = \frac{(\lambda/\mu)^m / m!}{\sum_{n=0}^{m} (\lambda/\mu)^n / n!} \qquad \text{(Erlang B Formula)}$$

Summary of *M*/*G*/1 system results

1. Utilization factor

$$\rho = \frac{\lambda}{\mu}$$

2. Mean residual service time

$$R = \frac{\lambda \overline{X^2}}{2}$$

3. Pollaczek–Khinchin formula

$$W = \frac{R}{1-\rho} = \frac{\lambda \overline{X^2}}{2(1-\rho)}$$

$$T = \frac{1}{\mu} + W$$

$$N_Q = \frac{\lambda^2 \overline{X^2}}{2(1-\rho)}$$

$$N = \rho + \frac{\lambda^2 \overline{X^2}}{2(1-\rho)}$$

4. Pollaczek–Khinchin formula for $M/G/1$ queue with vacations

$$W = \frac{\lambda \overline{X^2}}{2(1-\rho)} + \frac{\overline{V^2}}{2\overline{V}}$$

$$T = \frac{1}{\mu} + W$$

where $\overline{V}$ and $\overline{V^2}$ are the first two moments of the vacation interval.

Summary of reservation/polling results

1. Average waiting time (m-user system, unlimited service)

$$W = \frac{\lambda \overline{X^2}}{2(1-\rho)} + \frac{(m-\rho)\overline{V}}{2(1-\rho)} + \frac{\sigma_V^2}{2\overline{V}} \qquad \text{(exhaustive)}$$

$$W = \frac{\lambda \overline{X^2}}{2(1-\rho)} + \frac{(m+\rho)\overline{V}}{2(1-\rho)} + \frac{\sigma_V^2}{2\overline{V}} \qquad \text{(partially gated)}$$

$$W = \frac{\lambda \overline{X^2}}{2(1-\rho)} + \frac{(m+2-\rho)\overline{V}}{2(1-\rho)} + \frac{\sigma_V^2}{2\overline{V}} \qquad \text{(gated)}$$

where $\rho = \lambda/\mu$, and $\overline{V}$ and σ_V^2 are the mean and variance of the reservation intervals, respectively, averaged over all users

$$\overline{V} = \frac{1}{m} \sum_{\ell=0}^{m-1} \overline{V_\ell}$$

$$\sigma_V^2 = \frac{1}{m} \sum_{\ell=0}^{m-1} \left(\overline{V_\ell^2} - \overline{V_\ell}^2 \right)$$

2. Average waiting time (m-user system, limited service)

$$W = \frac{\lambda\overline{X^2}}{2(1-\rho-\lambda\overline{V})} + \frac{(m+\rho)\overline{V}}{2(1-\rho-\lambda\overline{V})} + \frac{\sigma_V^2(1-\rho)}{2\overline{V}(1-\rho-\lambda\overline{V})} \quad \text{(partially gated)}$$

$$W = \frac{\lambda\overline{X^2}}{2(1-\rho-\lambda\overline{V})} + \frac{(m+2-\rho-2\lambda\overline{V})\overline{V}}{2(1-\rho-\lambda\overline{V})} + \frac{\sigma_V^2(1-\rho)}{2\overline{V}(1-\rho-\lambda\overline{V})} \quad \text{(gated)}$$

3. Average time in the system

$$T = \frac{1}{\mu} + W$$

Summary of priority queueing results

1. *Nonpreemptive priority.* Average waiting time in queue for class k customers

$$W_k = \frac{\sum_{i=1}^{n} \lambda_i \overline{X_i^2}}{2(1-\rho_1-\cdots-\rho_{k-1})(1-\rho_1-\cdots-\rho_k)}$$

2. *Nonpreemptive priority.* Average time in the system for class k customers

$$T_k = \frac{1}{\mu_k} + W_k$$

3. *Preemptive resume priority.* Average time in the system for class k customers

$$T_k = \frac{(1/\mu_k)(1-\rho_1-\cdots-\rho_k) + R_k}{(1-\rho_1-\cdots-\rho_{k-1})(1-\rho_1-\cdots-\rho_k)}$$

where

$$R_k = \frac{\sum_{i=1}^{k} \lambda_i \overline{X_i^2}}{2}$$

Heavy traffic approximation for the $G/G/1$ system

Average waiting time in queue satisfies

$$W \leq \frac{\lambda(\sigma_a^2 + \sigma_b^2)}{2(1-\rho)}$$

where

σ_a^2 = Variance of the interarrival times

σ_b^2 = Variance of the service times

λ = Average interarrival time

ρ = Utilization factor λ/μ, where $1/\mu$ is the average service time

Second Edition

Data Networks

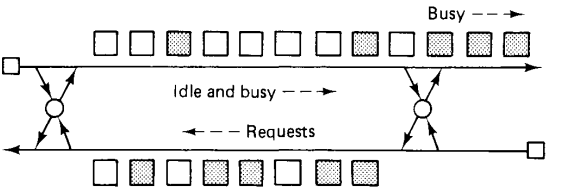
DIMITRI BERTSEKAS
Massachusetts Institute of Technology

ROBERT GALLAGER
Massachusetts Institute of Technology



PRENTICE HALL, Englewood Cliffs, New Jersey 07632

4



Multiaccess Communication

4.1 INTRODUCTION

The subnetworks considered thus far have consisted of nodes joined by point-to-point communication links. Each such link might consist physically of a pair of twisted wires, a coaxial cable, an optical fiber, a microwave radio link, and so on. The implicit assumption about point-to-point links, however, is that the received signal on each link depends only on the transmitted signal and noise on that link.

There are many widely used communication media, such as satellite systems, radio broadcast, multidrop telephone lines, and multitap bus systems, for which the received signal at one node depends on the transmitted signal at two or more other nodes. Typically, such a received signal is the sum of attenuated transmitted signals from a set of other nodes, corrupted by distortion, delay, and noise. Such media, called *multiaccess media*, form the basis for local area networks (LANs), metropolitan area networks (MANs), satellite networks, and radio networks.

The layering discussed in Chapters 1 and 2 is not quite appropriate for multiaccess media. One needs an additional sublayer, often called the *medium access control* (MAC) sublayer, beween the data link control (DLC) layer and the modem or physical layer. The

Samsung Exhibit 1015, Page 287 of 573

purpose of this extra sublayer is to allocate the multiaccess medium among the various nodes. As we study this allocation issue, we shall see that the separation of functions between layers is not as clean as it is with point-to-point links. For example, feedback about transmission errors is part of the ARQ function of the DLC layer, but is often also central to the problem of allocation and thus flow control. Similarly, much of the function of routing is automatically implemented by the broadcast nature of multiaccess channels.

Conceptually, we can view multiaccess communication in queueing terms. Each node has a queue of packets to be transmitted and the multiaccess channel is a common server. Ideally, the server should view all the waiting packets as one combined queue to be served by the appropriate queueing discipline. Unfortunately, the server does not know which nodes contain packets; similarly, nodes are unaware of packets at other nodes. Thus, the interesting part of the problem is that knowledge about the state of the queue is distributed.

There are two extremes among the many strategies that have been developed for this generic problem. One is the "free-for-all" approach in which nodes normally send new packets immediately, hoping for no interference from other nodes. The interesting question here is when and how packets are retransmitted when collisions (*i.e.*, interference) occur. The other extreme is the "perfectly scheduled" approach in which there is some order (round robin, for example) in which nodes receive reserved intervals for channel use. The interesting questions here are: (1) what determines the scheduling order (it could be dynamic), (2) how long can a reserved interval last, and (3) how are nodes informed of their turns?

Sections 4.2 and 4.3 explore the free-for-all approach in a simple idealized environment that allows us to focus on strategies for retransmitting collided packets. Successively more sophisticated algorithms are developed that reduce delay, increase available throughput, and maintain stable operation. In later sections, these algorithms are adapted to take advantage of special channel characteristics so as to reduce delay and increase throughput even further. The more casual reader can omit Sections 4.2.3 and 4.3.

Section 4.4 explores carrier sense multiple access (CSMA). Here the free-for-all approach is modified; a packet transmission is not allowed to start if the channel is sensed to be busy. We shall find that this set of strategies is a relatively straightforward extension of the ideas in Sections 4.2 and 4.3. The value of these strategies is critically dependent on the ratio of propagation delay to packet transmission time, a parameter called $\mathcal{J}$. If $\mathcal{J} \ll 1$, CSMA can decrease delay and increase throughput significantly over the techniques of Sections 4.2 and 4.3. The casual reader can omit Sections 4.4.2 and 4.4.4.

Section 4.5 deals with scheduling, or reserving, the channel in response to the dynamic requirements of the individual nodes. We start with satellite channels in Section 4.5.1; here the interesting feature is dealing with $\mathcal{J} \gg 1$. Next, Sections 4.5.2 to 4.5.4 treat the major approaches to LANs and MANs. These approaches can be viewed as reservation systems and differ in whether the reservations are scheduled in a free-for-all manner or in a round-robin manner. LANs are usually designed for the assumption that $\mathcal{J}$ is small, and Section 4.5.5 explores systems with higher speed or greater geographical coverage for which $\mathcal{J}$ is large.

Samsung Exhibit 1015, Page 288 of 573

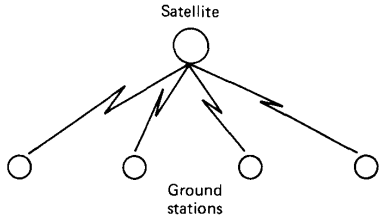
Finally, Section 4.6 explores packet radio. Here the interesting issue is that each node interferes only with a limited subset of other nodes; thus, multiple nodes can transmit simultaneously without interference.

Before going into any of the topics above in detail, we briefly discuss some of the most widely used multiaccess media.

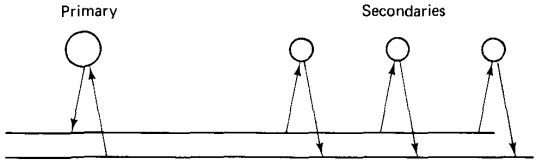
4.1.1 Satellite Channels

In a typical geosynchronous communication satellite system, many ground stations can transmit to a common satellite receiver, with the received messages being relayed to the ground stations (see Fig. 4.1). Such satellites often have separate antenna beams for different geographical areas, allowing independent reception and relaying between areas. Also, FDM (or TDM) can be used, permitting different earth stations within the region covered by a single antenna beam to be independently received.

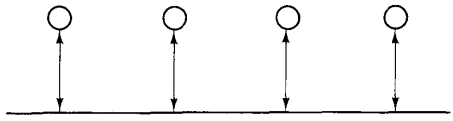
It is thus possible to use a satellite channel as a collection of virtual point-to-point links, with two virtual links being separated either by antenna beam or multiplexing. The potential difficulty with this approach is the same as the difficulty with using FDM or



(a) Satellite multiaccess channel



(b) Multidrop telephone line



(c) Multitap bus

Figure 4.1 Common multiaccess channels.

TDM for multiple sessions on a single point-to-point link, namely increased delay and underutilization of the medium.

In Section 4.5.1 we shall find that delay can be reduced and utilization increased by sharing the medium on a demand basis. This is more difficult than demand sharing (*i.e.*, statistical multiplexing) on a point-to-point link because the earth stations are not aware of the instantaneous traffic requirements of other earth stations. If several stations transmit at the same time (and in the same frequency band), their signals are garbled and incorrectly received.

4.1.2 Multidrop Telephone Lines

Another example of a multiaccess channel is a multidrop telephone line. Such lines connect one primary node with a number of secondary nodes; the signal transmitted by the primary node goes over one pair of wires and is received by all the secondary nodes. Similarly, there is a return pair of wires which carries the sum of the transmitted signals from all the secondary nodes to the primary node. Conceptually, this is like a satellite channel. The secondary nodes, or earth stations, share the path to the primary node, or satellite, whereas the primary node, or satellite, broadcasts to all the secondary nodes, or earth stations. Most communication on a multidrop phone line is intended to go from primary to secondary, or vice versa, whereas most communication on a satellite channel is relayed by the satellite from one earth station to another. Conceptually, this difference is not very important, since the major problem is that of sharing the channel from the secondary nodes to the primary, and it makes little difference whether the messages are removed at the primary node or broadcast back to all the secondary nodes.

The traditional mode of operation for multidrop telephone lines is for the primary node to poll (*i.e.*, request information from) each secondary node in some order. Each secondary node responds to its poll either by sending data back to the primary station or by indicating that it has no data to send. This strategy avoids interference between the secondary nodes, since nodes are polled one at a time, but there is a certain amount of inefficiency involved, both in the time to send a poll and the time to wait for a response from a node with no data.

4.1.3 Multitapped Bus

A third example of a multiaccess channel is a bus with multiple taps. In this case, each node can receive the signal sent by each other node, but again, if multiple nodes transmit at the same time, the received signal is garbled. We shall discuss this example later in the context of Ethernet, but for now, we observe that conceptually this channel is very similar to a satellite channel. Each node can communicate with each other node, but if nodes transmit simultaneously, the received signal cannot be correctly detected. The fact that nodes can hear each other directly here, as opposed to hearing each other via relay from the satellite, has some important practical consequences, but we will ignore this for now.

Samsung Exhibit 1015, Page 290 of 573

4.1.4 Packet Radio Networks

A fourth example of multiaccess communication is that of a packet radio network. Here each node is in reception range of some subset of other nodes. Thus, if only one node in the subset is transmitting, the given node can receive the transmission, whereas if multiple nodes are transmitting simultaneously in the same band, the reception will be garbled. Similarly, what one node transmits will be heard by a subset of the other nodes. In general, because of different noise conditions at the nodes, the subset of nodes from which a given node can receive is different from the subset to which the given node can transmit. The fact that each receiver hears a subset of transmitters rather than all transmitters makes packet radio far more complex than the other examples discussed. In the next four sections, we study multiaccess channels in which a receiver can hear all transmitters, and then in Section 4.6, we discuss packet radio networks in more detail.

4.2 SLOTTED MULTIACCESS AND THE ALOHA SYSTEM

Satellite channels, multidrop telephone lines, and multitap bus systems all share the feature of a set of nodes sharing a communication channel. If two or more nodes transmit simultaneously, the reception is garbled, and if none transmit, the channel is unused. The problem is somehow to coordinate the use of the channel so that exactly one node is transmitting for an appreciable fraction of the time. We start by looking at a highly idealized model. We shall see later that multiaccess channels can often be used in practice with much higher utilization than is possible in this idealized model, but we shall also see that these practical extensions can be understood more clearly in terms of our idealization.

4.2.1 Idealized Slotted Multiaccess Model

The idealized model to be developed allows us to focus on the problem of dealing with the contention that occurs when multiple nodes attempt to use the channel simultaneously. Conceptually, we view the system as in Fig. 4.1(a), with m transmitting nodes and one receiver.

It will be observed that aside from some questions of propagation delay, this same model can be applied with m nodes that can all hear each other (*i.e.*, the situation with a multitap bus). We first list the assumptions of the model and then discuss their implications.

1. *Slotted system.* Assume that all transmitted packets have the same length and that each packet requires one time unit (called a slot) for transmission. All transmitters are synchronized so that the reception of each packet starts at an integer time and ends before the next integer time.
2. *Poisson arrivals.* Assume that packets arrive for transmission at each of the m transmitting nodes according to independent Poisson processes. Let λ be the overall arrival rate to the system, and let λ/m be the arrival rate at each transmitting node.

Samsung Exhibit 1015, Page 291 of 573

3. *Collision or perfect reception.* Assume that if two or more nodes send a packet in a given time slot, then there is a *collision* and the receiver obtains no information about the contents or source of the transmitted packets. If just one node sends a packet in a given slot, the packet is correctly received.
4. *0,1,e Immediate feedback.* Assume that at the end of each slot, each node obtains feedback from the receiver specifying whether 0 packets, 1 packet, or more than one packet (e for error) were transmitted in that slot.
5. *Retransmission of collisions.* Assume that each packet involved in a collision must be retransmitted in some later slot, with further such retransmissions until the packet is successfully received. A node with a packet that must be retransmitted is said to be *backlogged.*

6a. *No buffering.* If one packet at a node is currently waiting for transmission or colliding with another packet during transmission, new arrivals at that node are discarded and never transmitted. An alternative to this assumption is the following.

6b. *Infinite set of nodes* ($m = \infty$). The system has an infinite set of nodes and each newly arriving packet arrives at a new node.

Discussion of assumptions. The slotted system assumption (1) has two effects. The first is to turn the system into a discrete-time system, thus simplifying analysis. The second is to preclude, for the moment, the possibility of carrier sensing or early collision detection. Carrier sensing is treated in Section 4.4 and early collision detection is treated in Section 4.5. Both allow much more efficient use of the multiaccess channel, but can be understood more clearly as an extension of the present model. Synchronizing the transmitters for slotted arrival at the receiver is not entirely trivial, but may be accomplished with relatively stable clocks, a small amount of feedback from the receiver, and some guard time between the end of a packet transmission and the beginning of the next slot.

The assumption of Poisson arrivals (2) is unrealistic for the case of multipacket messages. We discuss this issue in Section 4.5 in terms of nodes making reservations for use of the channel.

The assumption of collision or perfect reception (3) ignores the possibility of errors due to noise and also ignores the possibility of "capture" techniques, by which a receiver can sometimes capture one transmission in the presence of multiple transmissions.

The assumption of immediate feedback (4) is quite unrealistic, particularly in the case of satellite channels. It is made to simplify the analysis, and we shall see later that delayed feedback complicates multiaccess algorithms but causes no fundamental problems. We also discuss the effects of more limited types of feedback later.

The assumption that colliding packets must be retransmitted (5) is certainly reasonable in providing reliable communication. In light of this assumption, the no-buffering assumption (6a) appears rather peculiar, since new arrivals to backlogged nodes are thrown away with impunity, but packets once transmitted must be retransmitted until successful. In practice, one generally provides some buffering along with some form of flow control to ensure that not too many packets back up at a node. Our interest in this section, however, is in multiaccess channels with a large number of nodes, a relatively

Samsung Exhibit 1015, Page 292 of 573

small arrival rate λ, and small required delay (*i.e.*, the conditions under which TDM does not suffice). Under these conditions, the fraction of backlogged nodes is typically small, and new arrivals at backlogged nodes are almost negligible. Thus, the delay for a system without buffering should be relatively close to that with buffering. Also, the delay for the unbuffered system provides a lower bound to the delay for a wide variety of systems with buffering and flow control.

The infinite-node assumption (6b) alternatively provides us with an upper bound to the delay that can be achieved with a finite number of nodes. In particular, given any multiaccess algorithm (*i.e.*, any rule that each node employs for selecting packet transmission times), each of a finite set of nodes can regard itself as a set of virtual nodes, one for each arriving packet. With the application of the given algorithm independently for each such packet, the situation is equivalent to that with an infinite set of nodes (*i.e.*, assumption 6b). In this approach, a node with several backlogged packets will sometimes send multiple packets in one slot, causing a sure collision. Thus, it appears that by avoiding such sure collisions and knowing the number of buffered packets at a node, a system with a finite number of nodes and buffering could achieve smaller delays than with $m = \infty$; in any case, however, the $m = \infty$ delay can always be achieved.

If the performance using assumption 6a is similar to that using 6b, then we are assured that we have a good approximation to the performance of a system with arbitrary assumptions about buffering. From a theoretical standpoint, assumption 6b captures the essence of multiaccess communication much better than 6a. We use 6a initially, however, because it is less abstract and it provides some important insights about the relationship between TDM and other algorithms.

4.2.2 Slotted Aloha

The Aloha network [Abr70] was developed around 1970 to provide radio communication between the central computer and various data terminals at the campuses of the University of Hawaii. This multiaccess approach will be described in Section 4.2.4, but first we discuss an improvement called slotted Aloha [Rob72]. The basic idea of this algorithm is that each unbacklogged node simply transmits a newly arriving packet in the first slot after the packet arrival, thus risking occasional collisions but achieving very small delay if collisions are rare. This approach should be contrasted with TDM, in which, with m nodes, an arriving packet would have to wait for an average of $m/2$ slots for its turn to transmit. Thus, slotted Aloha transmits packets almost immediately with occasional collisions, whereas TDM avoids collisions at the expense of large delays.

When a collision occurs in slotted Aloha, each node sending one of the colliding packets discovers the collision at the end of the slot and becomes backlogged. If each backlogged node were simply to retransmit in the next slot after being involved in a collision, then another collision would surely occur. Instead, such nodes wait for some random number of slots before retransmitting.

To gain some initial intuitive understanding of slotted Aloha, we start with an instructive but flawed analysis. With the infinite-node assumption (*i.e.*, 6b), the number

Samsung Exhibit 1015, Page 293 of 573

of new arrivals transmitted in a slot is a Poisson random variable with parameter λ. If the retransmissions from backlogged nodes are sufficiently randomized, it is plausible to approximate the total number of retransmissions and new transmissions in a given slot as a Poisson random variable with some parameter $G > \lambda$. With this approximation, the probability of a successful transmission in a slot is Ge^{-G}. Finally, in equilibrium, the arrival rate, λ, to the system should be the same as the departure rate, Ge^{-G}. This relationship is illustrated in Fig. 4.2.

We see that the maximum possible departure rate (according to the argument above) occurs at $G = 1$ and is $1/e \approx 0.368$. We also note, somewhat suspiciously, that for any arrival rate less than $1/e$, there are two values of G for which the arrival rate equals the departure rate. The problem with this elementary approach is that it provides no insight into the dynamics of the system. As the number of backlogged packets changes, the parameter G will change; this leads to a feedback effect, generating further changes in the number of backlogged packets. To understand these dynamics, we will have to analyze the system somewhat more carefully. The simple picture below, however, correctly identifies the maximum throughput rate of slotted Aloha as $1/e$ and also shows that G, the mean number of attempted transmissions per slot, should be on the order of 1 to achieve a throughput close to $1/e$. If $G < 1$, too many idle slots are generated, and if $G > 1$, too many collisions are generated.

To construct a more precise model, assume that each backlogged node retransmits with some fixed probability q_r in each successive slot until a successful transmission occurs. In other words, the number of slots from a collision until a given node involved in the collision retransmits is a geometric random variable having value $i \geq 1$ with probability $q_r(1 - q_r)^{i-1}$. The original version of slotted Aloha employed a uniform distribution for retransmission, but this is more difficult to analyze and has no identifiable

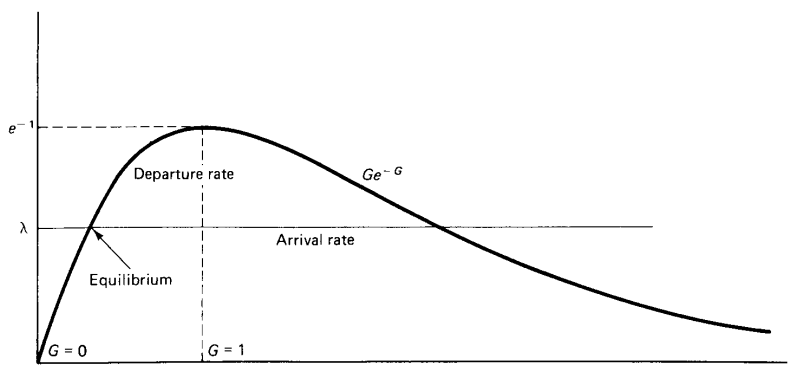


Figure 4.2 Departure rate as a function of attempted transmission rate G for slotted Aloha. Ignoring the dynamic behavior of G, departures (successful transmissions) occur at a rate Ge^{-G}, and arrivals occur at a rate λ, leading to a hypothesized equilibrium point as shown.

Samsung Exhibit 1015, Page 294 of 573

advantages over the geometric distribution. We will use the no-buffering assumption (6a) and switch later to the infinite node assumption (6b).

The behavior of slotted Aloha can now be described as a discrete-time Markov chain. Let n be the number of backlogged nodes at the beginning of a given slot. Each of these nodes will transmit a packet in the given slot, independently of each other, with probability q_r. Each of the $m-n$ other nodes will transmit a packet in the given slot if one (or more) such packets arrived during the previous slot. Since such arrivals are Poisson distributed with mean λ/m, the probability of no arrivals is $e^{-\lambda/m}$; thus, the probability that an unbacklogged node transmits a packet in the given slot is $q_a = 1 - e^{-\lambda/m}$. Let $Q_a(i, n)$ be the probability that i unbacklogged nodes transmit packets in a given slot, and let $Q_r(i, n)$ be the probability that i backlogged nodes transmit,

$$Q_a(i, n) = \binom{m-n}{i} (1 - q_a)^{m-n-i} q_a^i \tag{4.1}$$

$$Q_r(i, n) = \binom{n}{i} (1 - q_r)^{n-i} q_r^i \tag{4.2}$$

Note that from one slot to the next, the state (*i.e.*, the number of backlogged packets) increases by the number of new arrivals transmitted by unbacklogged nodes, less one if a packet is transmitted successfully. A packet is transmitted successfully, however, only if one new arrival and no backlogged packet, or no new arrival and one backlogged packet is transmitted. Thus, the state transition probability of going from state n to $n + i$ is given by

$$P_{n,n+i} = \begin{cases} Q_a(i, n), & 2 \le i \le (m-n) \\ Q_a(1, n)[1 - Q_r(0, n)], & i = 1 \\ Q_a(1, n)Q_r(0, n) + Q_a(0, n)[1 - Q_r(1, n)], & i = 0 \\ Q_a(0, n)Q_r(1, n), & i = -1 \end{cases} \tag{4.3}$$

Figure 4.3 illustrates this Markov chain. Note that the state can decrease by at most 1 in a single transition, and this makes it easy to calculate the steady-state probabilities

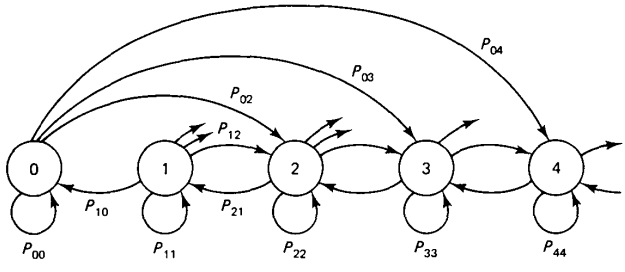


Figure 4.3 Markov chain for slotted Aloha. The state (*i.e.*, backlog) can decrease by at most one per transition, but can increase by an arbitrary amount.

Samsung Exhibit 1015, Page 295 of 573

iteratively, finding p_n for each successively larger n in terms of p_0 and finally, finding p_0 as a normalizing constant (see Problem 4.1). From this, the expected number of backlogged nodes can be found, and from Little's theorem, the average delay can be calculated.

Unfortunately, this system has some very strange properties for a large number of nodes, and the steady-state results above are very misleading. To get some intuitive understanding of this, note that we would like to choose the retransmission probability q_r to be moderately large, so as to avoid large delays after collisions. If the arrival rate is small and not many packets are involved in collisions, this works well and retransmissions are normally successful. On the other hand, if the system is afflicted with a run of bad luck and the number of backlogged packets n gets large enough to satisfy $q_r n >> 1$, then collisions occur in almost all successive slots and the system remains heavily backlogged for a long time.

To understand this phenomenon quantitatively, define the *drift* in state n (D_n) as the expected change in backlog over one slot time, starting in state n. Thus, D_n is the expected number of new arrivals accepted into the system [*i.e.*, $(m-n)q_a$] less the expected number of successful transmissions in the slot; the expected number of successful transmissions is just the probability of a successful transmission, defined as P_{succ}. Thus,

$$D_n = (m-n)q_a - P_{succ} \tag{4.4}$$

where

$$P_{succ} = Q_a(1,n)Q_r(0,n) + Q_a(0,n)Q_r(1,n) \tag{4.5}$$

Define the attempt rate $G(n)$ as the expected number of attempted transmissions in a slot when the system is in state n, that is,

$$G(n) = (m-n)q_a + nq_r$$

If q_a and q_r are small, P_{succ} is closely approximated as the following function of the attempt rate:

$$P_{succ} \approx G(n)e^{-G(n)} \tag{4.6}$$

This approximation can be derived directly from Eq. (4.5), using the approximation $(1-x)^y \approx e^{-xy}$ for small x in the expressions for Q_a and Q_r. Similarly, the probability of an idle slot is approximately $e^{-G(n)}$. Thus, the number of packets in a slot is well approximated as a Poisson random variable (as in the earlier intuitive analysis), but the parameter $G(n)$ varies with the state. Figure 4.4 illustrates Eqs. (4.4) and (4.6) for the case $q_r > q_a$ (this is the only case of interest, as discussed later). The drift is the difference between the curve and the straight line. Since the drift is the expected change in state from one slot to the next, the system, although perhaps fluctuating, tends to move in the direction of the drift and consequently tends to cluster around the two stable points with rare excursions between the two.

There are two important conclusions from this figure. First, the departure rate (*i.e.*, P_{succ}) is at most $1/e$ for large m. Second, the departure rate is almost zero for long periods whenever the system jumps to the undesirable stable point. Consider the effect of

Samsung Exhibit 1015, Page 296 of 573

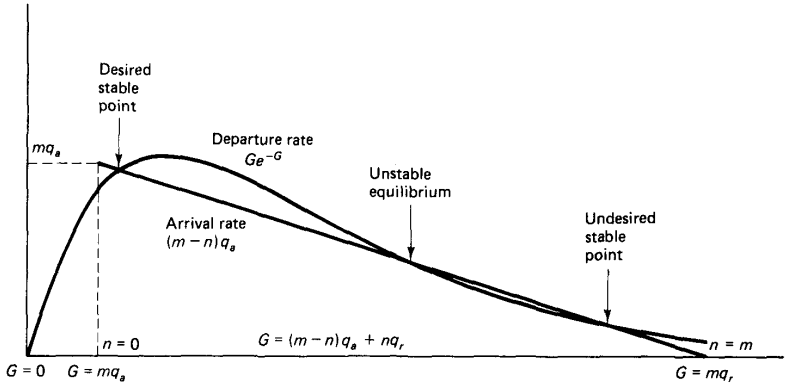


Figure 4.4 Instability of slotted Aloha. The horizontal axis corresponds to both the state n and the attempt rate G, which are related by the linear equation $G = (m-n)q_a + nq_r$ with $q_r > q_a$. For n to the left of the unstable point, D is negative and n drifts toward the desired stable point. For n to the right of the unstable point, D is positive and n drifts toward the undesired stable point.

changing q_r. As q_r is increased, the delay in retransmitting a collided packet decreases, but also the linear relationship between n and the attempt rate $G(n) = (m-n)q_a + nq_r$ changes [*i.e.*, $G(n)$ increases with n faster when q_r is increased]. If the horizontal scale for n is held fixed in Fig. 4.4, this change in attempt rate corresponds to a contraction of the horizontal $G(n)$ scale, and thus to a horizontal contraction of the curve Ge^{-G}. This means that the number of backlogged packets required to exceed the unstable equilibrium point decreases. Alternatively, if q_r is decreased, retransmission delay increases, but it becomes more difficult to exceed the unstable equilibrium point. If q_r is decreased enough (while still larger than q_a), the curve Ge^{-G} will expand enough in Fig. 4.4 that only one stable state will remain. At this stable point, and similarly when $q_r = q_a$, the backlog is an appreciable fraction of m, and this means both that an appreciable number of arriving packets are discarded and that the delay is excessively large.

The question of what values of q_r and arrival rate lead to stable behavior, particularly when q_r and arrival rate vary from node to node and infinite buffering is provided at each node, has received considerable theoretical attention (*e.g.*, [Tsy85]). These theoretical studies generally assume that nodes are considered backlogged immediately on arrival. Problem 4.8, however, illustrates that if q_r is small enough for stable operation, then the delay is considerably greater than that with TDM; thus these approaches are not of great practical importance.

If we replace the no-buffering assumption with the infinite-node assumption, the attempt rate $G(n)$ becomes $\lambda + nq_r$ and the straight line representing arrivals in Fig. 4.4 becomes horizontal. In this case, the undesirable stable point disappears, and once the state of the system passes the unstable equilibrium, it tends to increase without bound. In this case, the corresponding infinite-state Markov chain has no steady-state distribution

Samsung Exhibit 1015, Page 297 of 573

and the expected backlog increases without bound as the system continues to run. (See Section 3A.5 for further discussion of stability of such systems.)

From a practical standpoint, if the arrival rate λ is very much smaller than $1/e$, and if q_r is moderate, then the system could be expected to remain in the desired stable state for very long periods. In the unlikely event of a move to the undesired stable point, the system could be restarted with backlogged packets lost. Rather than continuing to analyze this rather flawed system, however, we look at modifications of slotted Aloha that cure this stability issue.

4.2.3 Stabilized Slotted Aloha

One simple approach to achieving stability should be almost obvious now. P_{succ} is approximately $G(n)e^{-G(n)}$, which is maximized at $G(n) = 1$. Thus, it is desirable to change q_r dynamically to maintain the attempt rate $G(n)$ at 1. The difficulty here is that n is unknown to the nodes and can only be estimated from the feedback. There are many strategies for estimating n or the appropriate value of q_r (*e.g.*, [HaL82] and [Riv85]). All of them, in essence, increase q_r when an idle slot occurs and decrease q_r when a collision occurs.

Stability and maximum throughput. The notion of stability must be clarified somewhat before proceeding. Slotted Aloha was called unstable in the last subsection on the basis of the representation in Fig. 4.4. Given the no-buffering assumption, however, the system has a well-defined steady-state behavior for all arrival rates. With the infinite-node assumption, on the other hand, there is no steady-state distribution and the expected delay grows without bound as the system continues to run. With the no-buffering assumption, the system discards large numbers of arriving packets and has a very large but finite delay, whereas with the infinite-node assumption, no arrivals are discarded but the delay becomes infinite.

In the following, we shall use the infinite-node assumption (6b), and define a multiaccess system as stable for a given arrival rate if the expected delay per packet (either as a time average or an ensemble average in the limit of increasing time) is finite. Ordinary slotted Aloha is unstable, in this sense, for any arrival rate greater than zero. Note that if a system is stable, then for a sufficiently large but finite number of nodes m, the system (regarding each arrival as corresponding to a new virtual node) has a smaller expected delay than TDM, since the delay of TDM, for a fixed overall arrival rate, increases linearly with m.

The maximum stable throughput of a multiaccess system is defined as the least upper bound of arrival rates for which the system is stable. For example, the maximum stable throughput of ordinary slotted Aloha is zero. Our purpose with these definitions is to study multiaccess algorithms that do not require knowledge of the number of nodes m and that maintain small delay (for given λ) independent of m. Some discussion will be given later to modifications that make explicit use of the number of nodes.

Returning to the stabilization of slotted Aloha, note that when the estimate of backlog is perfect, and $G(n)$ is maintained at the optimal value of 1, then (according to

Samsung Exhibit 1015, Page 298 of 573

the Poisson approximation) idles occur with probability $1/e \approx 0.368$, successes occur with probability $1/e$, and collisions occur with probability $1 - 2/e \approx 0.264$. Thus, the rule for changing q_r should allow fewer collisions than idles. The maximum stable throughput of such a system is at most $1/e$. To see this, note that when the backlog is large, the Poisson approximation becomes more accurate, the success rate is then limited to $1/e$, and thus the drift is positive for $\lambda > 1/e$. It is important to observe that this argument depends on all backlogged nodes using the same retransmission probability. We shall see in Section 4.3 that if nodes use both their own history of retransmissions and the feedback history in their decisions about transmitting, maximum stable throughputs considerably higher than $1/e$ are possible.

Pseudo-Bayesian algorithm. Rivest's pseudo-Bayesian algorithm [Riv85] is a particularly simple and effective way to stabilize Aloha. This algorithm is essentially the same as an earlier, independently derived algorithm by Mikhailov [Mik79], but Rivest's Bayesian interpretation simplifies understanding. The algorithm differs from slotted Aloha in that new arrivals are regarded as backlogged immediately on arrival. Rather than being transmitted with certainty in the next slot, they are transmitted with probability q_r in the same way as packets involved in previous collisions. Thus, if there are n backlogged packets (including new arrivals) at the beginning of a slot, the attempt rate is $G(n) = nq_r$, and the probability of a successful transmission is $nq_r(1 - q_r)^{n-1}$. For unstabilized Aloha, this modification would not make much sense, since q_r has to be relatively small and new arrivals would be unnecessarily delayed. For stabilized Aloha, however, q_r can be as large as 1 when the estimated backlog is negligible, so that new arrivals are held up only when the system is already estimated to be congested. In Problem 4.6 it is shown that this modification increases the probability of success if the backlog estimate is accurate.

The algorithm operates by maintaining an estimate $\hat{n}$ of the backlog n at the beginning of each slot. Each backlogged packet is then transmitted (independently) with probability $q_r(\hat{n}) = \min\{1, 1/\hat{n}\}$. The minimum operation limits q_r to at most 1, and subject to this limitation, tries to achieve an attempt rate $G = nq_r$ of 1. For each k, the estimated backlog at the beginning of slot $k + 1$ is updated from the estimated backlog and feedback for slot k according to the rule

$$\hat{n}_{k+1} = \begin{cases} \max\{\lambda, \hat{n}_k + \lambda - 1\}, & \text{for idle or success} \\ \hat{n}_k + \lambda + (e-2)^{-1}, & \text{for collision} \end{cases} \tag{4.7}$$

The addition of λ to the previous backlog accounts for new arrivals, and the max operation ensures that the estimate is never less than the contribution from new arrivals. On successful transmissions, 1 is subtracted from the previous backlog to account for the successful departure. Finally, subtracting 1 from the previous backlog on idle slots and adding $(e-2)^{-1}$ on collisions has the effect of decreasing $\hat{n}$ when too many idles occur and increasing $\hat{n}$ when too many collisions occur. For large backlogs, if $\hat{n} = n$, each of the n backlogged packets is independently transmitted in a slot with probability $q_r = 1/n$. Thus $G(n)$ is 1, and, by the Poisson approximation, idles occur with probability $1/e$ and

Samsung Exhibit 1015, Page 299 of 573

collisions with probability $(e-2)/e$, so that decreasing $\hat{n}$ by 1 on idles and increasing $\hat{n}$ by $(e-2)^{-1}$ on collisions maintains the balance between n and $\hat{n}$ on the average.

It is shown in Problem 4.9 that if the a priori probability distribution on n_k is Poisson with parameter $\hat{n}_k \geq 1$, then given an idle or successful transmission in slot k, the probability distribution of n_{k+1} is Poisson with parameter $\hat{n}_k + \lambda - 1$. Given a collision, the resulting distribution of n_{k+1} is not quite Poisson but is reasonably approximated as Poisson with parameter $\hat{n}_{k+1}$. For this reason, the algorithm is called pseudo-Bayesian.

Figure 4.5 provides some heuristic understanding of why the algorithm is stable for all $\lambda < 1/e$. The state of the system is characterized by n and $\hat{n}$, and the figure shows the expected drift in these variables from one slot to the next. If n and $\hat{n}$ are large and equal, the expected drift of each is $\lambda - 1/e$, which is negative. On the other hand, if the absolute value of $n - \hat{n}$ is large, the expected drift of n is positive, but the expected reduction in $|n - \hat{n}|$ is considerably larger. Thus, if the system starts at some arbitrary point in the $(n, \hat{n})$ plane, n might increase on the average for a number of slots, but eventually n and $\hat{n}$ will come closer together and then n will decrease on the average.

In applications, the arrival rate λ is typically unknown and slowly varying. Thus, the algorithm must either estimate λ from the time-average rate of successful transmissions or set its value within the algorithm to some fixed value. It has been shown by Mikhailov (see [Kel85]) and Tsitsiklis [Tsi87] that if the fixed value $1/e$ is used within the algorithm, stability is achieved for all actual $\lambda < 1/e$. Nothing has been proven about the behavior of the algorithm when a dynamic estimate of λ is used within the algorithm.

Approximate delay analysis. An exact analysis of expected delay for this algorithm, even with known λ, appears to be very difficult, but it is instructive to analyze an approximate model. Assume that λ is known and that the probability of successful transmission P_{succ} is $1/e$ whenever the backlog n is 2 or more, and that $P_{succ} = 1$ for

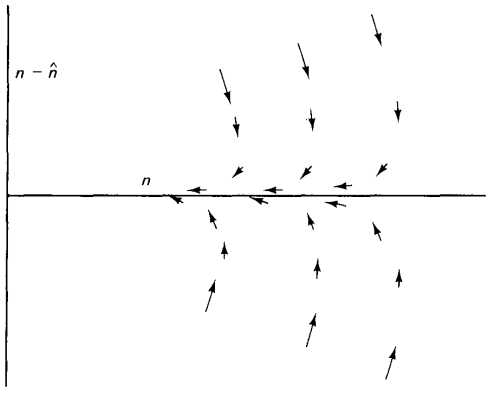


Figure 4.5 Drift of n and $n - \hat{n}$ for the pseudo-Bayesian stabilization algorithm. When the absolute value of $n - \hat{n}$ is large, it approaches 0 faster than n increases.

Samsung Exhibit 1015, Page 300 of 573

$n = 1$. This is a reasonable model for very small λ, since very few collisions occur and q_r is typically 1. It is also reasonable for large $\lambda < 1/e$, since typically n is large and $\hat{n} \approx n$.

Let W_i be the delay from the arrival of the i^{th} packet until the beginning of the i^{th} successful transmission. Note that if the system were first-come first-serve, W_i would be the queueing time of the i^{th} arrival. By the same argument as in Problem 3.32, however, the average of W_i over all i is equal to the expected queueing delay W. Let n_i be the number of backlogged packets at the instant before i's arrival; n_i does not include any packet currently being successfully transmitted, but does include current unsuccessful transmissions. W_i can be expressed as

$$W_i = R_i + \sum_{j=1}^{n_i} t_j + y_i \tag{4.8}$$

R_i is the residual time to the beginning of the next slot, and t_1 (for $n_i > 0$) is the subsequent interval until the next successful transmission is completed. Similarly, t_j, $1 < j \leq n_i$, is the interval from the end of the $(j-1)^{\text{th}}$ subsequent success to the end of the j^{th} subsequent success. After those n_i successes, y_i is the remaining interval until the beginning of the next successful transmission (*i.e.*, the i^{th} transmission overall).

Observe that for each interval t_j, the backlog is at least two, counting the new i^{th} arrival and the n_i packets already backlogged. Thus, each slot is successful with probability $1/e$, and the expected value of each t_j is e. Next, observe that Little's formula relates the expected values of W_i and n_i (*i.e.*, W_i is the wait not counting the successful transmission, and n_i is the number in the system not counting successful transmissions in process). Finally, the expected value of R_i is $1/2$ (counting time in slots). Thus, taking the expected value of Eq. (4.8) and averaging over i, we get

$$W = 1/2 + \lambda e W + E\{y\} \tag{4.9}$$

Now consider the system at the first slot boundary at which both the $(i-1)^{\text{st}}$ departure has occurred and the i^{th} arrival has occurred. If the backlog is 1 at that point (*i.e.*, only the i^{th} packet is in the system), then y_i is 0. Alternatively, if $n > 1$, then $E\{y_i\} = e - 1$. Let p_n be the steady-state probability that the backlog is n at a slot boundary. Then, since a packet is always successfully transmitted if the state is 1 at the beginning of the slot, we see that p_1 is the fraction of slots in which the state is 1 and a packet is successfully transmitted. Since λ is the total fraction of slots with successful transmissions, p_1/λ is the fraction of packets transmitted from state 1 and $1 - p_1/\lambda$ is the fraction transmitted from higher-numbered states. It follows that

$$E\{y\} = \frac{(e-1)(\lambda - p_1)}{\lambda} \tag{4.10}$$

Finally, we must determine p_1. From the argument above, we see that $\lambda = p_1 + (1 - p_0 - p_1)/e$. Also, state 0 can be entered at a slot boundary only if no arrivals occurred in the previous slot and the previous state was 0 or 1. Thus, $p_0 = (p_0 + p_1)e^{-\lambda}$. Solving for p_1 gives

$$p_1 = \frac{(1 - \lambda e)(e^{\lambda} - 1)}{1 - (e-1)(e^{\lambda} - 1)} \tag{4.11}$$

Samsung Exhibit 1015, Page 301 of 573

Combining Eqs. (4.9) to (4.11) yields

$$W = \frac{e - 1/2}{1 - \lambda e} - \frac{(e^{\lambda} - 1)(e - 1)}{\lambda[1 - (e - 1)(e^{\lambda} - 1)]} \tag{4.12}$$

This value of W is quite close to delay values for the pseudo-Bayesian algorithm obtained through simulation. Figure 4.6 plots $E\{W\}$ for this approximate model and compares it with the queueing delay for TDM with 8 and 16 nodes. It is quite surprising that the delay is so small even for arrival rates relatively close to $1/e$. It appears that the assumption of immediate feedback is unnecessary for stabilization strategies of this type to be stable; the argument is that feedback delay will make the estimate $\hat{n}$ less accurate, but $n/\hat{n}$ will still stay close to 1 for large n.

Binary exponential backoff. In the packet radio networks to be discussed in Section 4.6, and in some other multiaccess situations, the assumption of 0,1,e feedback on all slots is unrealistic. In some systems, a node receives feedback only about whether or not its own packets were successfully transmitted; it receives no feedback about slots in which it does not transmit. Such limited feedback is sufficient for slotted Aloha but is insufficient for the backlog estimation of the pseudo-Bayesian strategy. Binary exponential backoff [MeB76] is a stabilization strategy used in Ethernet that employs only this more limited form of feedback.

This strategy is very simple; if a packet has been transmitted unsuccessfully i times, then the probability of transmission in successive slots is set at $q_r = 2^{-i}$ (or is uniformly distributed over the next 2^i slots after the i^{th} failure). When a packet initially arrives in the system, it is transmitted immediately in the next slot.

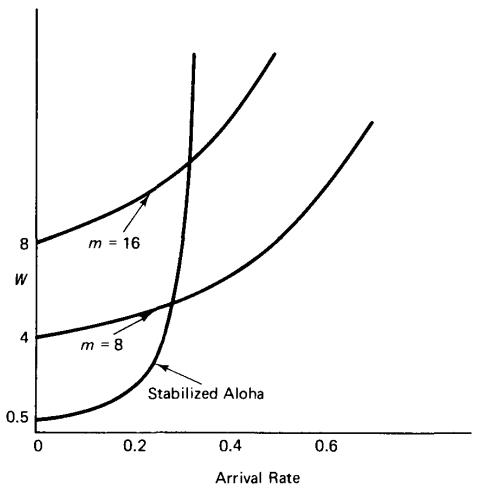


Figure 4.6 Comparison of expected waiting time W in slots, from arrival until beginning of successful transmission, for stabilized Aloha and for TDM with $m = 8$ and $m = 16$. For small arrival rates, the delay of stabilized Aloha is little more than waiting for the next slot, whereas as the arrival rate approaches $1/e$, the delay becomes unbounded.

Samsung Exhibit 1015, Page 302 of 573

Some rationale for this strategy can be seen by noting that when a packet first arrives (with this limited feedback), the node knows nothing of the backlog, so the immediate first transmission is reasonable. With successive collisions, any reasonable estimate of backlog would increase, motivating the decrease in the local q_r. To make matters worse, however, as q_r is reduced, the node gets less feedback per slot about the backlog, and thus, to play safe, it is reasonable to increase the backlog estimate by larger and larger amounts on each successive collision.

Unfortunately, in the limit as the number of nodes approaches infinity, this strategy is unstable for every arrival rate λ greater than 0 [Ald86]. It is unknown whether or not any strategy can achieve stability with this type of limited feedback. Problem 4.10, however, develops another strategy that can be used with this limited feedback and with a finite number of nodes with unlimited buffering to achieve finite expected delay for any λ less than 1. Unfortunately, the price of this high throughput is inordinately large delay.

4.2.4 Unslotted Aloha

Unslotted, or pure, Aloha [Abr70] was the precursor to slotted Aloha. In this strategy, each node, upon receiving a new packet, transmits it immediately rather than waiting for a slot boundary. Slots play no role in pure Aloha, so we temporarily omit the slotted system assumption. If a packet is involved in a collision, it is retransmitted after a random delay. Assume that if the transmission times for two packets overlap at all, the CRCs on those packets will fail and retransmission will be required. We assume that the receiver rebroadcasts the composite received signal (or that all nodes receive the composite signal), so that each node, after a given propagation delay, can determine whether or not its transmitted packets were correctly received. Thus, we have the same type of limited feedback discussed in the last subsection. Other types of feedback could be considered and Problem 4.11 develops some of the peculiar effects of other feedback assumptions.

Figure 4.7 shows that if one packet starts transmission at time t, and all packets have unit length, then any other transmission starting between $t-1$ and $t+1$ will cause a collision. For simplicity, assume an infinite number of nodes (*i.e.*, assumption 6b) and let n be the number of backlogged nodes at a given time. For our present purposes, a node is considered backlogged from the time it has determined that its previously transmitted packet was involved in a collision until the time that it attempts retransmission. Assume that the period until attempted retransmission τ is an exponentially distributed random variable with probability density $xe^{-x\tau}$, where x is an arbitrary parameter interpreted as a node's retransmission attempt rate. Thus, with an overall Poisson arrival rate of λ to the system, the initiation times of attempted transmissions is a time-varying Poisson process of rate $G(n) = \lambda + nx$ in which n is the backlog at a given time.

Consider the sequence of successive transmission attempts on the channel. For some given i, let τ_i be the duration of the interval between the initiations of the i^{th} and $(i+1)^{\text{th}}$ transmission attempt. The i^{th} attempt will be successful if both τ_i and τ_{i-1} exceed 1 (assuming unit length packets). Given the backlog in each intertransmission

Samsung Exhibit 1015, Page 303 of 573

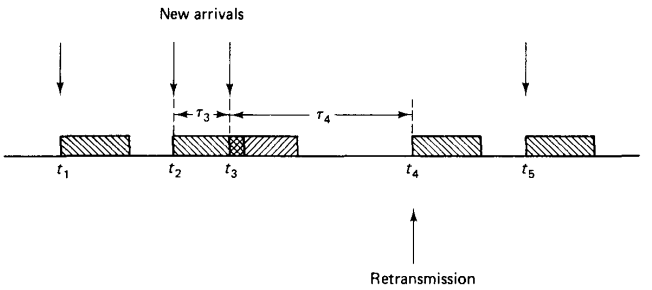


Figure 4.7 Unslotted Aloha. New arrivals are transmitted immediately and unsuccessful transmissions are repeated after a random delay. Packet transmission time is one unit, and two transmissions collide if their interdeparture interval is less than one unit.

interval, these intervals are independent. Thus, assuming a backlog of n for each interval, we have

$$P_{succ} = e^{-2G(n)} \tag{4.13}$$

Since attempted transmissions occur at rate $G(n)$, the throughput (*i.e.*, the expected number of successful transmissions per unit time) as a function of n is

$$\text{throughput}\,(n) = G(n)e^{-2G(n)} \tag{4.14}$$

Figure 4.8 illustrates this result. The situation is very similar to that of slotted Aloha, except that the throughput has a maximum of $1/(2e)$, achieved when $G(n) = 1/2$. The analysis above is approximate in the sense that Eq. (4.13) assumes that the backlog is the same in the intervals surrounding a given transmission attempt, whereas according to our definition of backlog, the backlog decreases by one whenever a backlogged packet initiates transmission and increases by one whenever a collided packet is detected. For small x (*i.e.*, large mean time before attempted retransmission), this effect is relatively small.

It can be seen from Fig. 4.8 that pure Aloha has the same type of stability problem as slotted Aloha. For the limited feedback assumed here, stability is quite difficult to achieve or analyze (as is the case with slotted Aloha). Very little is known about stabilization for pure Aloha, but if λ is very small and the mean retransmission time very large, the system can be expected to run for long periods without major backlog buildup.

One of the major advantages of pure Aloha is that it can be used with variable-length packets, whereas with slotted Aloha, long packets must be broken up to fit into slots and short packets must be padded out to fill up slots. This compensates for some of the inherent throughput loss of pure Aloha and gives it an advantage in simplicity. Some analysis, with simplifying assumptions, of the effect of variable-length packets appears in [Fer75] and [San80].

Samsung Exhibit 1015, Page 304 of 573

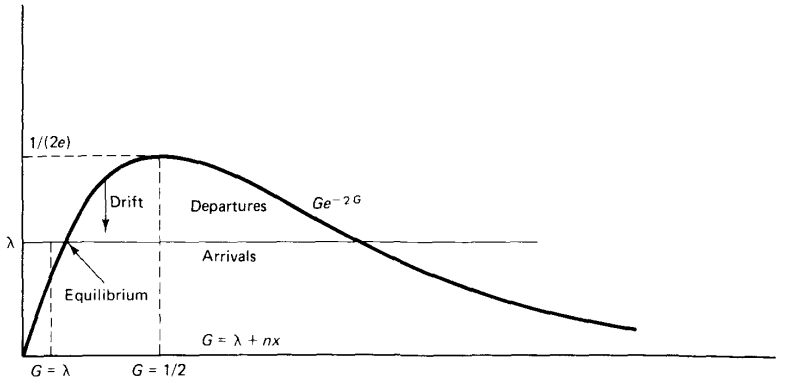


Figure 4.8 Pure Aloha as a function of the attempted transmission rate G. Successful departures leave the system at a rate Ge^{-2G}, and arrivals occur at a rate λ, leading to a hypothesized equilibrium at the point shown.

4.3 SPLITTING ALGORITHMS

We have seen that slotted Aloha requires some care for stabilization and is also essentially limited to throughputs of $1/e$. We now want to look at more sophisticated collision resolution techniques that both maintain stability without any complex estimation procedures and also increase the achievable throughput. To get an intuitive idea of how this can be done, note that with relatively small attempt rates, when a collision occurs, it is most likely between only two packets. Thus, if new arrivals could be inhibited from transmission until the collision was resolved, each of the colliding packets could be independently retransmitted in the next slot with probability $1/2$. This would lead to a successful transmission for one of the packets with probability $1/2$, and the other could then be transmitted in the following slot. Alternatively, with probability $1/2$, another collision or an idle slot occurs. In this case, each of the two packets would again be independently transmitted in the next slot with probability $1/2$, and so forth until a successful transmission occurred, which would be followed by the transmission of the remaining packet.

With the strategy above, the two packets require two slots with probability $1/2$, three slots with probability $1/4$, and i slots with probability $2^{-(i-1)}$. The expected number of slots for sending these two packets can thus be calculated to be three, yielding a throughput of $2/3$ for the period during which the collision is being resolved.

There are various ways in which the nodes involved in a collision could choose whether or not to transmit in successive slots. Each node could simply flip an unbiased coin for each choice. Alternatively (in a way to be described precisely later) each node could use the arrival time of its collided packet. Finally, assuming a finite set of nodes, each with a unique identifier represented by a string of bits, a node could use the successive bits of its identity to make the successive choices. This last alternative has the

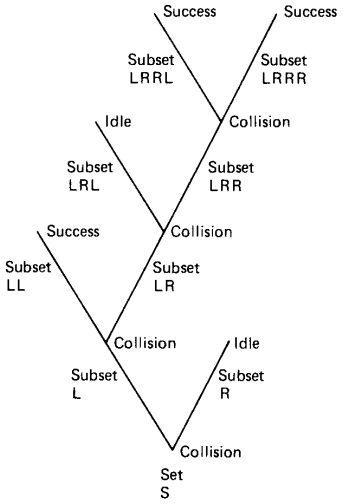
Samsung Exhibit 1015, Page 305 of 573

advantage of limiting the number of slots required to resolve a collision, since each pair of nodes must differ in at least one bit of their identifiers. All of these alternatives have the common property that the set of colliding nodes is split into subsets, one of which transmits in the next slot. If the collision is not resolved (*e.g.*, if each colliding node is in the same subset), then a further splitting into subsets takes place. We call algorithms of this type *splitting algorithms*. In the subsequent development of these algorithms, we assume a slotted channel, Poisson arrivals, collisions or perfect reception, $(0, 1, e)$ immediate feedback, retransmission of collisions, and an infinite set of nodes (*i.e.*, the assumptions 1 to 6b of Section 4.2.1).

4.3.1 Tree Algorithms

The first splitting algorithms were algorithms with a tree structure ([Cap77], [TsM78], and [Hay76]). When a collision occurs, say in the k^{th} slot, all nodes not involved in the collision go into a waiting mode, and all those involved in the collision split into two subsets (*e.g.*, by each flipping a coin). The first subset transmits in slot $k + 1$, and if that slot is idle or successful, the second subset transmits in slot $k + 2$ (see Fig. 4.9). Alternatively, if another collision occurs in slot $k + 1$, the first of the two subsets splits again, and the second subset waits for the resolution of that collision.

The rooted binary tree structure in Fig. 4.9 represents a particular pattern of idles, successes, and collisions resulting from such a sequence of splittings. S represents the



Slot	Xmit Set	Waiting Sets	Feedback
1	S	–	*e*
2	L	R	*e*
3	LL	LR, R	1
4	LR	R	*e*
5	LRL	LRR, R	0
6	LRR	R	*e*
7	LRRL	LRRR, R	1
8	LRRR	R	1
9	R	–	0

Figure 4.9 Tree algorithm. After a collision, all new arrivals wait and all nodes involved in the collision divide into subsets. Each successive collision in a subset causes that subset to again split into smaller subsets while other nodes wait.

Samsung Exhibit 1015, Page 306 of 573

set of packets in the original collision, and L (left) and R (right) represent the two subsets that S splits into. Similarly, LL and LR represent the two subsets that L splits into after L generates a collision. Each vertex in the tree corresponds to a subset (perhaps empty) of backlogged packets. Vertices whose subsets contain two or more packets have two upward branches corresponding to the splitting of the subset into two new subsets; vertices corresponding to subsets with 0 or 1 packet are leaf vertices of the tree.

The set of packets corresponding to the root vertex S is transmitted first, and after the transmission of the subset corresponding to any nonleaf vertex, the subset corresponding to the vertex on the left branch, and all of its descendant subsets, are transmitted before the subsets of the right branch. Given the immediate feedback we have assumed, it should be clear that each node, in principle, can construct this tree as the 0, 1, e feedback occurs; each node can keep track of its own subset in that tree, and thus each node can determine when to transmit its own backlogged packet.

The transmission order above corresponds to that of a stack. When a collision occurs, the subset involved is split, and each resulting subset is pushed on the stack (*i.e.*, each stack element is a subset of nodes); then the subset at the head of the stack (*i.e.*, the most recent subset pushed on the stack) is removed from the stack and transmitted. The list, from left to right, of waiting subsets in Fig. 4.9 corresponds to the stack elements starting at the head for the given slot. Note that a node with a backlogged packet can keep track of when to transmit by a counter determining the position of the packet's current subset on the stack. When the packet is involved in a collision, the counter is set to 0 or 1, corresponding to which subset the packet is placed in. When the counter is 0, the packet is transmitted, and if the counter is nonzero, it is incremented by 1 for each collision and decremented by 1 for each success or idle.

One problem with this tree algorithm is what to do with the new packet arrivals that come in while a collision is being resolved. A collision resolution period (CRP) is defined to be completed when a success or idle occurs and there are no remaining elements on the stack (*i.e.*, at the end of slot 9 in Fig. 4.9). At this time, a new CRP starts using the packets that arrived during the previous CRP. In the unlikely event that a great many slots are required in the previous CRP, there will be many new waiting arrivals, and these will collide and continue to collide until the subsets get small enough in this new CRP. The solution to this problem is as follows: At the end of a CRP, the set of nodes with new arrivals is immediately split into j subsets, where j is chosen so that the expected number of packets per subset is slightly greater than 1 (slightly greater because of the temporary high throughput available after a collision). These new subsets are then placed on the stack and the new CRP starts.

The tree algorithm is now essentially completely defined. Each node with a packet involved in the current CRP keeps track of its position in the stack as described above. All the nodes keep track of the number of elements on the stack and the number of slots since the end of the previous CRP. On the completion of that CRP, each node determines the expected number of waiting new arrivals, determines the new number j of subsets, and those nodes with waiting new arrivals randomly choose one of those j subsets and set their counter for the corresponding stack position.

Samsung Exhibit 1015, Page 307 of 573

The maximum throughput available with this algorithm, optimized over the choice of j as a function of expected number of waiting packets, is 0.43 packets per slot [Cap77]; we omit any analysis since we next show some simple ways of improving this throughput.

Improvements to the tree algorithm. First consider the situation in Fig. 4.10. Here, in slots 4 and 5, a collision is followed by an idle slot; this means that all the packets involved in the collision were assigned to the second subset. The tree algorithm would simply transmit this second subset, generating a guaranteed collision. An improvement results by omitting the transmission of this second subset, splitting it into two subsets, and transmitting the first subset. Similarly, if an idle again occurs, the second subset is again split before transmission, and so forth.

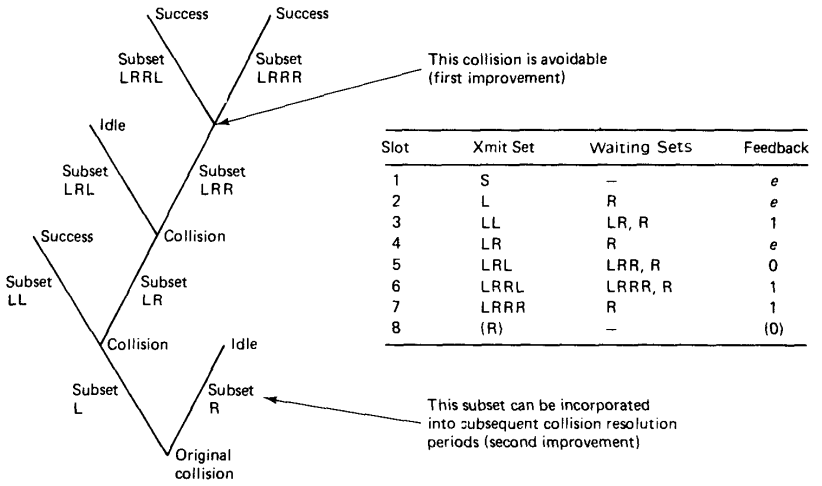
This improvement can be visualized in terms of a stack and implemented by manipulating counters just like the original tree algorithm. Each node must now keep track of an additional binary state variable that takes the value 1 if, for some $i \geq 1$, the last i slots contained a collision followed by $i - 1$ idles; otherwise, the state variable takes the value 0. If the feedback for the current slot is 0 and the state variable has the value 1, then the state variable maintains the value 1 and the subset on the top of the stack is split into two subsets that are pushed onto the stack in place of the original head element.

The maximum throughput with this improvement is 0.46 packets per slot [Mas80]. In practice, this improvement has a slight problem in that if an idle slot is incorrectly perceived by the receiver as a collision, the algorithm continues splitting indefinitely, never making further successful transmissions. Thus, in practice, after some number h of idle slots followed by splits, the algorithm should be modified simply to transmit the next subset on the stack without first splitting it; if the feedback is very reliable, h can be moderately large, whereas otherwise h should be small.

The next improvement in the tree algorithm not only improves throughput but also greatly simplifies the analysis. Consider what happens when a collision is followed by another collision in the tree algorithm (see slots 1 and 2 of Fig. 4.10). Let x be the number of packets in the first collision, and let x_L and x_R be the number of packets in the resultant subsets; thus, $x = x_L + x_R$. Assume that, a priori, before knowing that there is a collision, x is a Poisson random variable. Visualize splitting these x packets, by coin flip, say, into the two subsets with x_L and x_R packets, respectively, before knowing about the collision. Then a priori x_L and x_R are independent Poisson random variables each with half the mean value of x. Given the two collisions, then, x_L and x_R are independent Poisson random variables conditional on $x_L + x_R \geq 2$ and $x_L \geq 2$. The second condition implies the first, so the first can be omitted; this means that x_R, conditional on the feedback, is still Poisson with its original unconditional distribution. Problem 4.17 demonstrates this result in a more computational and less abstract way. Thus, rather than devoting a slot to this second subset, which has an undesirably small expected number of packets, it is better to regard the second subset as just another part of the waiting new arrivals that have never been involved in a collision.

When the idea above is incorporated into an actual algorithm, the first-come first-serve (FCFS) splitting algorithm, which is the subject of the next subsection, results. Before discussing this, we describe some variants of the tree algorithm.

Samsung Exhibit 1015, Page 308 of 573



Slot	Xmit Set	Waiting Sets	Feedback
1	S	–	*e*
2	L	R	*e*
3	LL	LR, R	1
4	LR	R	*e*
5	LRL	LRR, R	0
6	LRRL	LRRR, R	1
7	LRRR	R	1
8	(R)	–	(0)

Figure 4.10 Improvements in the tree algorithm. Subset LRR can be split without first being transmitted since the feedback implies that it contains two or more packets. Also, subset R is better combined with new arrivals since the number of packets in it is Poisson with an undesirably low rate.

Variants of the tree algorithm. The tree algorithm as described above requires all nodes to monitor the channel feedback and to keep track of when each collision resolution period ends. This is a disadvantage if the receivers at nodes are turned off when there are no packets to send. One way to avoid this disadvantage while maintaining the other features of the tree algorithm is to have new arrivals simply join the subset of nodes at the head of the stack. Thus, only currently backlogged nodes need to monitor the channel feedback. Algorithms of this type are called *unblocked stack algorithms*, indicating that new arrivals are not blocked until the end of the current collision resolution period. In contrast, the tree algorithm is often called a blocked stack algorithm.

With the tree algorithm, new arrivals are split into a variable number of subsets at the beginning of each collision resolution period, and then subsets are split into two subsets after each collision. With an unblocked stack algorithm, on the other hand, new arrivals are constantly being added to the subset at the head of the stack, and thus, collisions involve a somewhat larger number of packets on the average. Because of the relatively large likelihood of three or more packets in a collision, higher maximum throughputs can be achieved by splitting collision sets into three subsets rather than two. The maximum throughput thus available for unblocked stack algorithms is 0.40 [MaF85].

4.3.2 First-Come First-Serve Splitting Algorithms

In the second improvement to the tree algorithm described above, the second subset involved in a collision is treated as if it had never been transmitted if the first subset

Samsung Exhibit 1015, Page 309 of 573

contains two or more packets. Recall also that a set of colliding packets can be split into two subsets in a variety of ways (*e.g.*, by coin flipping, by node identities, or by arrival time). For our present purposes, the simplest choice is splitting by arrival time. By using this approach, each subset will consist of all packets that arrived in some given interval, and when a collision occurs, that interval will be split into two smaller intervals. By always transmitting the earlier arriving interval first, the algorithm will always transmit successful packets in the order of their arrival, leading to the name first-come first-serve (FCFS).

At each integer time k, the algorithm specifies the packets to be transmitted in slot k (*i.e.*, from k to $k+1$) to be the set of all packets that arrived in some earlier interval, say from $T(k)$ to $T(k)+\alpha(k)$. This interval is called the *allocation interval* for slot k (see Fig. 4.11). We can visualize the packets arriving after $T(k)+\alpha(k)$ as being in a

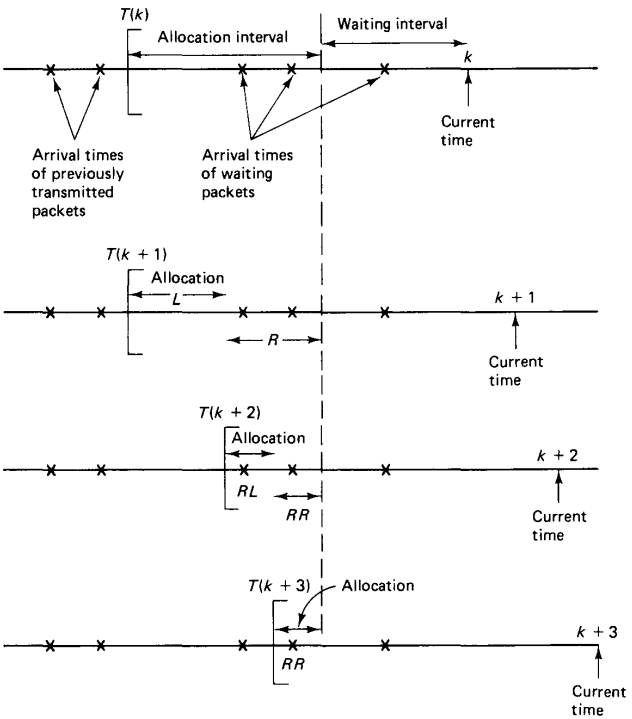


Figure 4.11 FCFS splitting algorithm. Packets are transmitted in order of arrival. On collisions, the allocation interval generating a collision is split into two subintervals, with the leftmost (earlier arrivals) transmitting first.

Samsung Exhibit 1015, Page 310 of 573