DATA NETWORKS

2nd Edition

by

Dimitri P. Bertsekas and Robert G. Gallager

Massachusetts Institute of Technology

WWW site for book information and orders

http://www.athenasc.com

Athena Scientific, Belmont, Massachusetts

Athena Scientific
Post Office Box 805
Nashua, NH 03060
U.S.A.

Email: info@athenasc.com
WWW: http://www.athenasc.com

Originally published in 1992 by Prentice Hall, Inc., Englewood Cliffs, NJ

Publisher's Cataloging-in-Publication Data

ISBN-10: 1-886529-22-1, ISBN-13: 978-1-886529-22-9

To Joanna and Marie

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

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

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

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.

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-

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

ABOUT THE AUTHORS

Dimitri Bertsekas

DIMITRI P. BERTSEKAS received a B.S. in Mechanical and Electrical Engineering from the National Technical University of Athens, Greece in 1965, and a Ph.D. in system science from the Massachusetts Institute of Technology in 1971. He has held faculty positions with the Engineering-Economic Systems Department, Stanford University, and the Electrical Engineering Department of the University of Illinois, Urbana. In 1979 he joined the faculty of the Massachusetts Institute of Technology where he is currently Professor of Electrical Engineering and Computer Science. He consults regularly with private industry, and has held editorial positions in several journals. He has been elected Fellow of the IEEE.

Professor Bertsekas has done research in control of stochastic systems, and in linear, nonlinear, and dynamic programming. He has written numerous papers in each of these areas. His current interests are primarily in data networks, distributed computation, and large-scale optimization. He is the author of *Dynamic Programming and Stochastic Control,* Academic Press, 1976, *Constrained Optimization and Lagrange Multiplier Methods,* Academic Press, 1982, *Dynamic Programming: Deterministic and Stochastic Models,* Prentice Hall, 1987, *Linear Network Optimization: Algorithms and Codes,* MIT Press, 1991, and the co-author of *Stochastic Optimal Control: The Discrete-Time Case,* Academic Press, 1978, and *Parallel and Distributed Computation: Numerical Methods,* Prentice Hall, 1989.

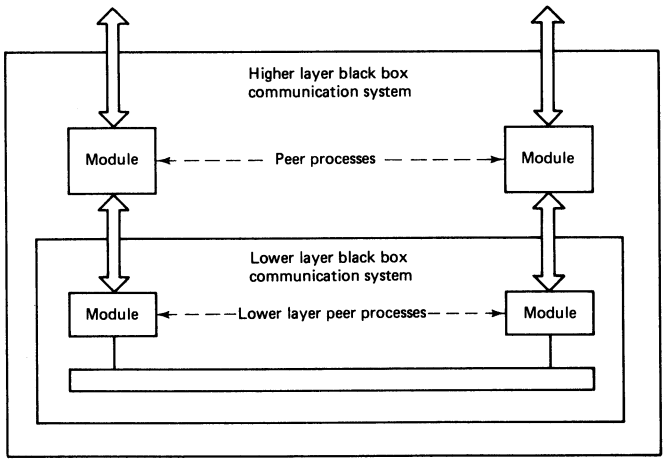
Robert Gallager

ROBERT G. GALLAGER received the S.B. degree in electrical engineering from the University of Pennsylvania, Philadelphia, PA, in 1953, and the S.M. and Sc.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, MA, in 1957 and 1960, respectively.

Following several years as a member of the Technical Staff at Bell Telephone Laboratories and service in the U.S. Army Signal Corps, he has been with the Massachusetts Institute of Technology since 1956. He is currently the Fujitsu Professor of Electrical Engineering and Computer Science, Co-Director of the Laboratory for Information and Decision Systems, and Co-Chairman of the Department Area I (Control, Communication, and Operations Research). He is a consultant to Codex Corporation. He is the author of the textbook, *Information Theory and Reliable Communication* (New York: Wiley, 1968). His major research interests are data communication networks, information theory, and communication theory.

Dr. Gallager was awarded the IEEE Baker Prize Paper Award in 1966 for the paper "A simple derivation of the coding theorem and some applications." He has been a member of the Board of Governors of the IEEE Information Theory Society from 1965 to 1970 and from 1979 to 1986, and was President of the Society in 1971. He is a Fellow of the IEEE and a member of the National Academy of Engineering. He was awarded the 1990 IEEE Medal of Honor for his contributions to communication coding techniques.

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

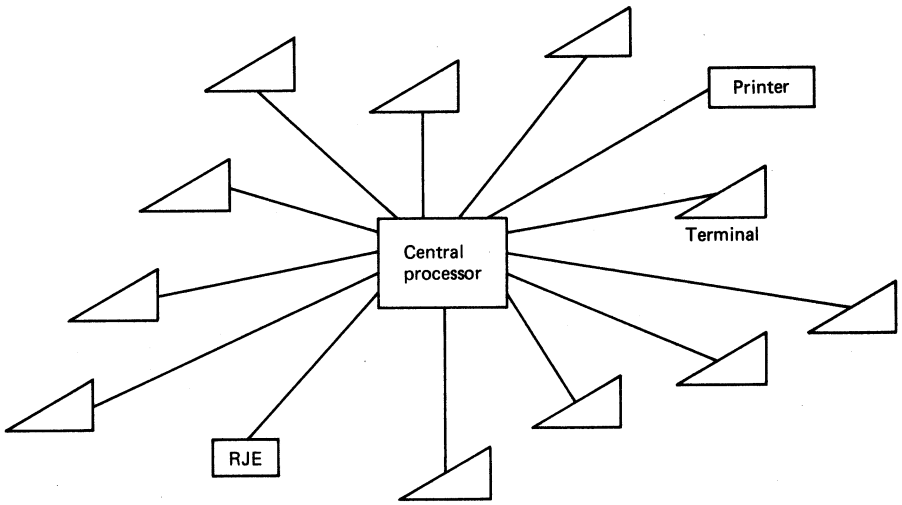


Figure 1.1 Network with one central processor and a separate communication link to each device.

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

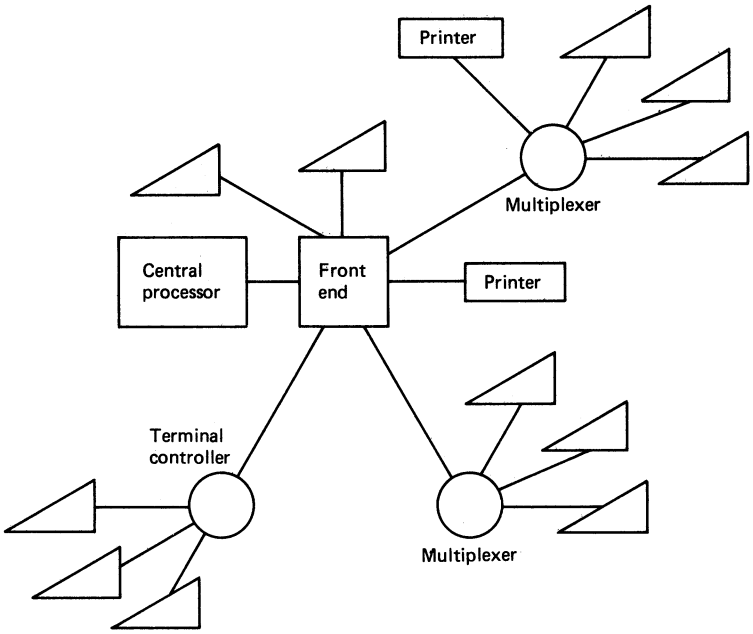


Figure 1.2 Network with one central processor but with shared communication links to devices.

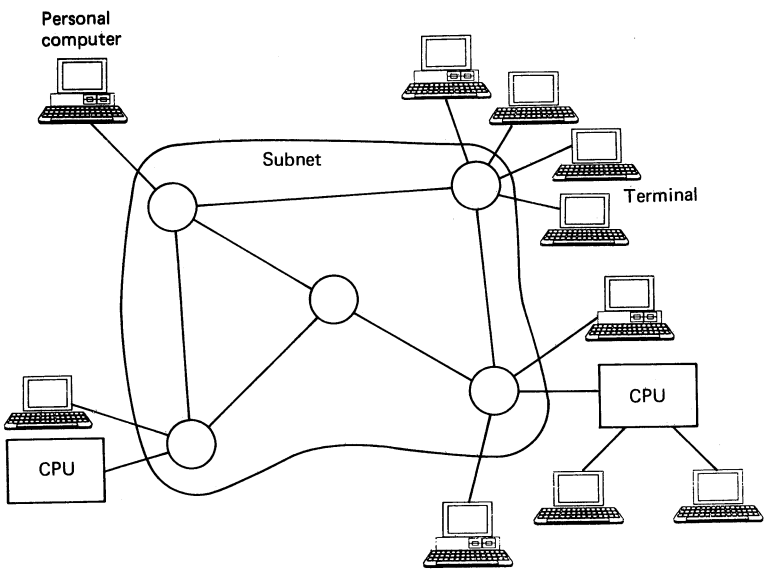


Figure 1.3 General network with a subnet of communication links and nodes. External devices are connected to the subnet via links to the subnet nodes.

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.

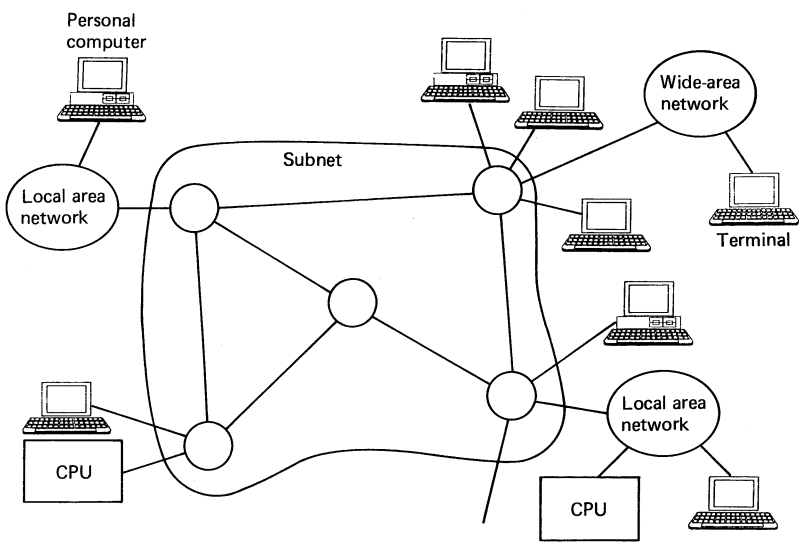


Figure 1.4 Network of interconnected networks. Individual wide area networks (WANs) and local networks (LANs) are connected via bridges and gateways.

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,

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

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

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

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,

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.

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-

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

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,

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

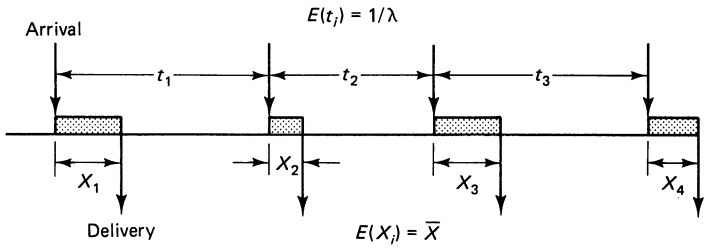


Figure 1.5 Link utilization. The expected transmission time of a message is $\overline{X}$. The expected interarrival period is $1/\lambda$. Thus, the link is used at most $\lambda\overline{X}$ of the time.

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

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

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.

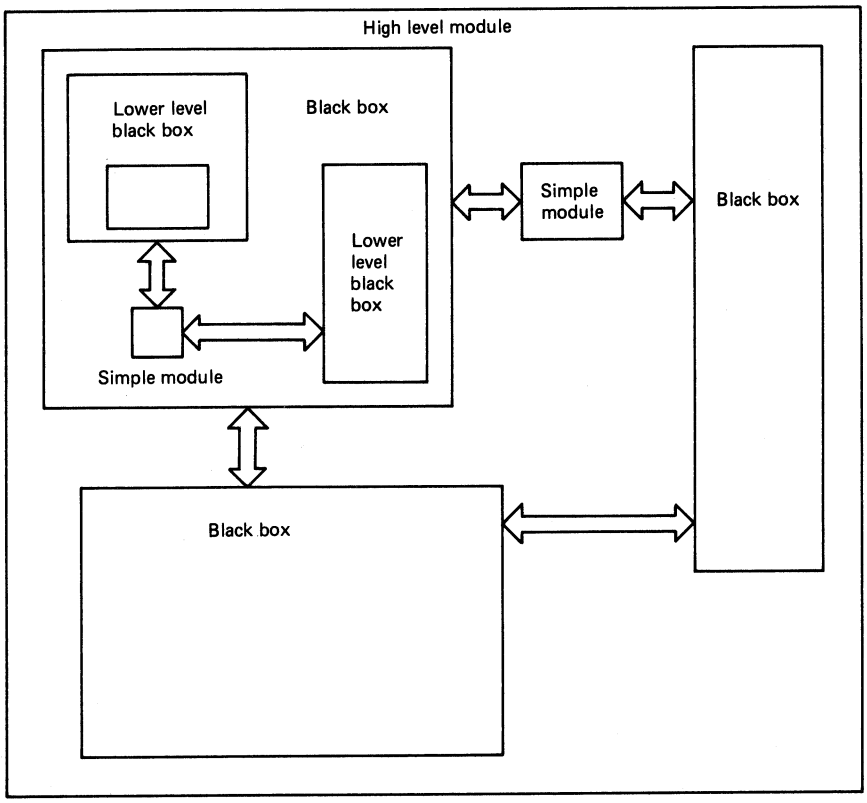


Figure 1.6 Hierarchy of nested black boxes. Each black box (except that at the lowest level) contains black boxes at a lower level, plus perhaps other modules.

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

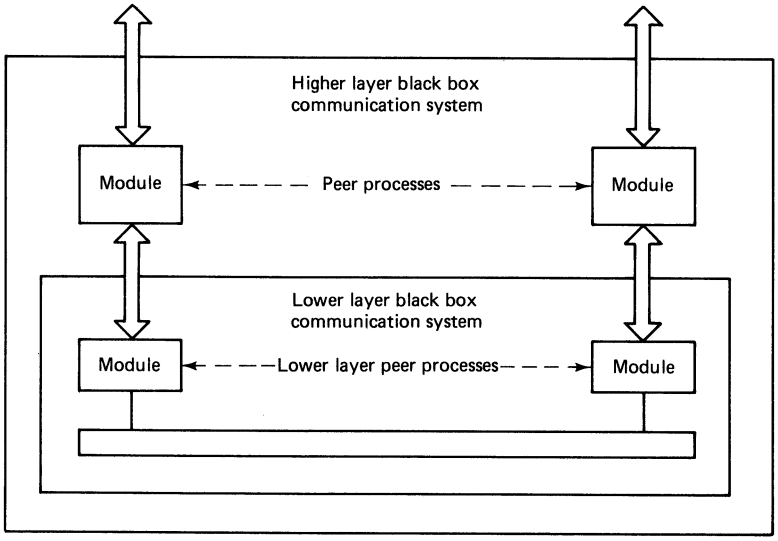


Figure 1.7 Peer processes within a black box communication system. The peer processes communicate through a lower-layer black box communication system that itself contains lower-layer peer processes.

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

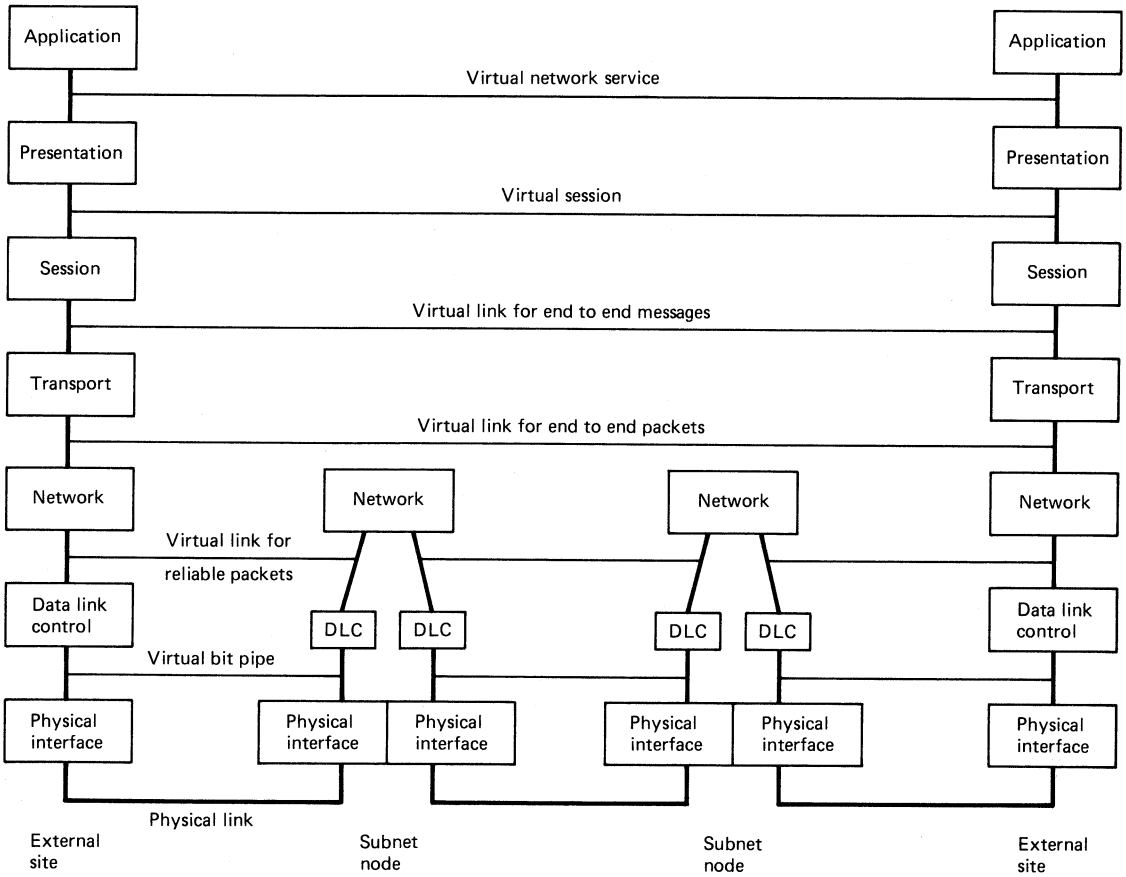


Figure 1.8 Seven-layer OSI network architecture. Each layer presents a virtual communication link with given properties to the next-higher layer.

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

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

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

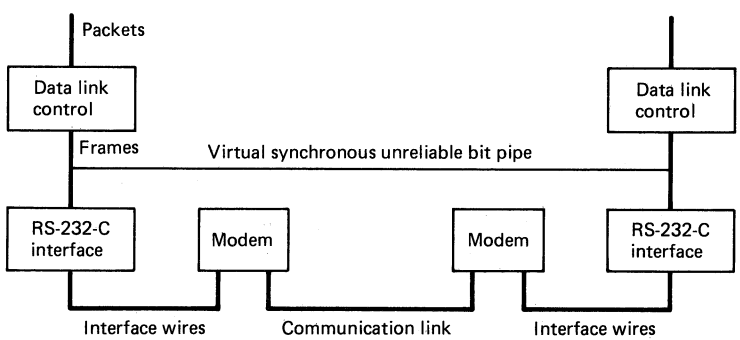


Figure 1.9 Layering with the interface between the DLC and the modem viewed as an interface over a physical medium consisting of a set of wires.

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

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.

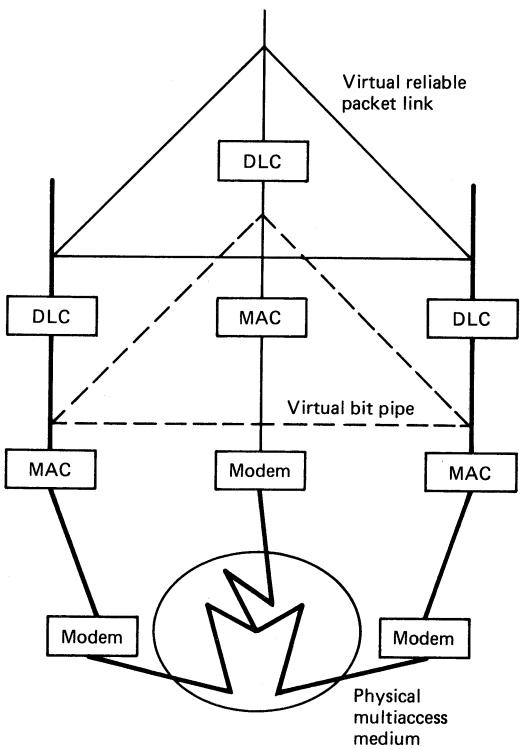


Figure 1.10 Layering for a multiaccess channel. The physical medium is accessed by all three users, each of whom hears the transmitted signals of the others. The DLC sublayer sees virtual point-to-point bit pipes below it. The MAC sublayer sees a multiaccess bit pipe, and the modems access the actual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

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

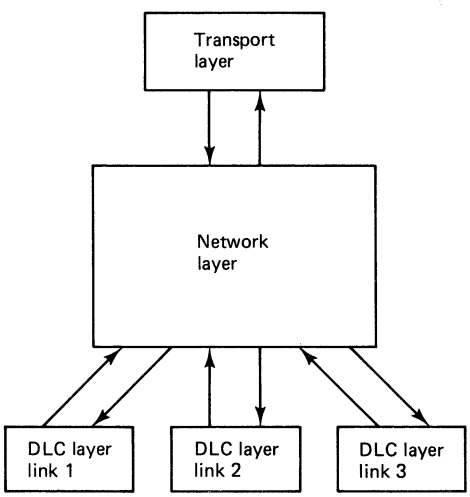


Figure 1.11 The network layer at a node or site can receive packets from a DLC layer for each incoming link and (in the case of a site) from the transport layer. It can send these packets out to the same set of modules.

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.

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

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

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

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

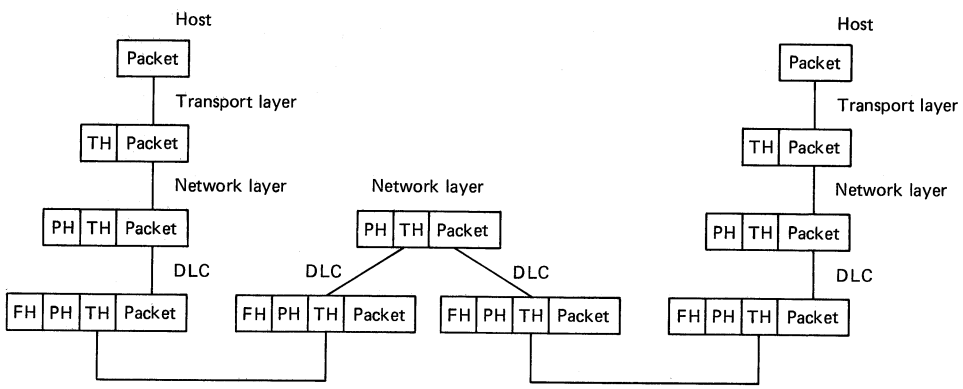


Figure 1.12 Illustration of various headers on a frame. Note that each layer looks only at its own header.

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

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.

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

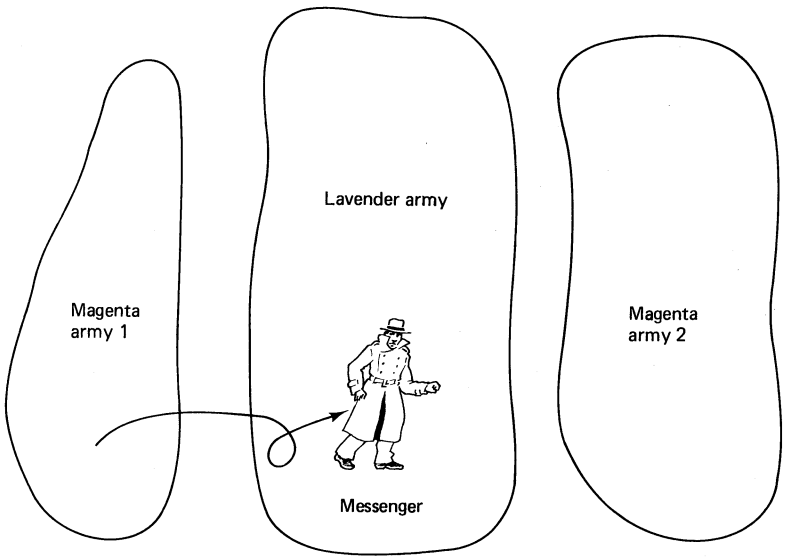


Figure 1.13 A messenger carries a message through enemy lines from magenta army 1 to magenta army 2. If the messenger is caught, the message is undelivered. Magenta army 1 is unaware of capture and magenta army 2 is unaware of existence of message.

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.

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?

References

[AaM81] AASHTIANI, H. Z., and MAGNANTI, T. L. 1981. Equilibria on a Congested Transportation Network, *SIAM J. Algebraic Discrete Methods*, 2:213–226.

[Abr70] ABRAMSON, N. 1970. The Aloha System—Another Alternative for Computer Communications, *Proc. Fall Joint Comput. Conf., AFIPS Conf.*, p. 37.

[Ahu79] AHUJA V. 1979. Routing and Flow Control in Systems Network Architecture, *IBM Systems J.*, 18:298–314.

[AkK77] AKINC, U., and KHUMAWALA, B. 1977. An Efficient Branch and Bound Algorithm for the Capacitated Warehouse Location Problem, *Management Sci.*, 23:585–594.

[Ald86] ALDOUS, D. 1986. *Ultimate Instability of Exponential Back-off Protocol for Acknowledgment-Based Transmission Control of Random Access Communication Channels.* Berkeley, CA: University of California, Dept. of Statistics.

[AlM76] ALCOUFFE, A., and MURATET, G. 1976. Optimum Location of Plants, *Management Sci.*, 23:267–274.

[Alt86] ALTES, T. 1986. *Minimum Delay Packet Length* (Report LIDS-TH-1602). Cambridge, MA: MIT Laboratory for Information and Decision Systems.

[AnP86] ANAGNOSTOU, M., and PROTONOTARIOS, E. 1986. Performance Analysis of the Selective Repeat ARQ Protocol, *IEEE Trans. Comm.*, COM-34:127–135.

[Ari84] ARIKAN, E. 1984. Some Complexity Results about Packet Radio Networks, *IEEE Trans. Inform. Theory*, IT-30:681–685.

[Ash90] ASH, G. R. 1990. Design and Control of Networks with Dynamic Nonhierarchical Routing, *IEEE Communications Magazine*, 28:34–40.

[Atk80] ATKINS, J. D. 1980. Path Control: The Transport Network of SNA., *IEEE Trans. Comm.*, COM-28:527–538.

[BaA81] BAKRY, S. H., and ACKROYD, M. H. 1981. Teletraffic Analysis for Single Cell Mobile Radio Telephone Systems, *IEEE Trans. Comm.*, COM-29:298–304.

[BaA82] BAKRY, S. H., and ACKROYD, M. H. 1982. Teletraffic Analysis for Multicell Mobile Radio Telephone Systems, *IEEE Trans. Comm.*, COM-30:1905–1909.

[BaC89] BALLERT, R., and CHING, Y.-C. 1989. SONET: Now It's the Standard Optical Network, *IEEE Comm. Mag.*, 29:8–15.

[Bac88] BACKES, F. 1988. Transparent Bridges for Interconnection of IEEE 802 LANs, *IEEE Network*, 2:5–9.

[Bal79] BALL, M. O. 1979. Computing Network Reliability, *Oper. Res.*, 27:823-838.

[BaP83] BALL, M. O., and PROVAN, J. S. 1983. Calculating Bounds on Reachability and Connectedness in Stochastic Networks, *Networks*, 13:253–278.

[Bar64] BARAN, P. 1964. On Distributed Communication Networks, *IEEE Trans. on Communications Systems*, CS-12:1–9.

[BaS83] BARATZ, L., and SEGALL, A. 1983. *Reliable Link Initialization Procedures* (Report RC10032). Yorktown Heights, NY: IBM Thomas J. Watson Research Center. Also *IEEE Trans. Comm.*, COM-36, 1988, pp. 144–152.

[BeG82] BERTSEKAS, D. P., and GAFNI, E. 1982. Projection Methods for Variational Inequalities with Application to the Traffic Assignment Problem, in D. C. Sorensen and R. J.-B. Wets (Eds.), *Mathematical Programming Studies*, Vol. 17. Amsterdam: North-Holland, pp. 139–159.

[BeG83] BERTSEKAS, D. P., and GAFNI, E. M. 1983. Projected Newton Methods and Optimization of Multicommodity Flows, *IEEE Trans. Automat. Control*, AC-28:1090–1096.

[Ben86] BENJAMIN, R. 1986. Analysis of Connection Survivability in Complex Strategic Communications Networks, *IEEE J. Select. Areas Comm.*, SAC-4:243–253.

[Ber76] BERTSEKAS, D. P. 1976. On the Goldstein–Levitin–Polyak Gradient Projection Method, *IEEE Trans. Automat. Control*, AC-21:174–184.

[Ber79a] BERTSEKAS, D. P. 1979. Algorithms for Nonlinear Multicommodity Network Flow Problems, in A. Bensoussan and J. L. Lions (Eds.), *International Symposium on Systems Optimization and Analysis*. New York: Springer-Verlag, pp. 210–224.

[Ber79b] BERTSEKAS, D. P. 1979. Dynamic Models of Shortest Path Routing Algorithms for Communication Networks with Multiple Destinations, *Proc. 1979 IEEE Conf. Dec. Control*, Ft. Lauderdale, FL, pp. 127–133.

[Ber80] BERTSEKAS, D. P. 1980. A Class of Optimal Routing Algorithms for Communication Networks, *Proc. 5th Internat. Conf. Comput. Comm.*, Atlanta, GA, pp. 71–76.

[Ber82a] BERTSEKAS, D. P. 1982. Distributed Dynamic Programming, *IEEE Trans. Automat. Control*, AC-27:610–616.

[Ber82b] BERTSEKAS, D. P. 1982. Dynamic Behavior of Shortest Path Routing Algorithms for Communication Networks, *IEEE Trans. Automat. Control*, AC-27:60–74.

[Ber82c] BERTSEKAS, D. P. 1982. Projected Newton Methods for Optimization Problems with Simple Constraints, *SIAM J. Control Optim.*, 20:221–246.

[Ber82d] BERTSEKAS, D. P. 1982. *Constrained Optimization and Lagrange Multiplier Methods*. New York: Academic Press.

[Ber83] BERTSEKAS, D. P. 1983. Distributed Asynchronous Computation of Fixed Points, *Math. Programming*, 27:107–120.

[Ber85] BERTSEKAS, D. P. 1985. A Unified Framework for Primal–Dual Methods in Minimum Cost Network Flow Problems, *Math. Programming*, 32:125–145.

[Ber87] BERTSEKAS, D. P. 1987. *Dynamic Programming: Deterministic and Stochastic Models*. Englewood Cliffs, NJ: Prentice Hall.

[Ber91] BERTSEKAS, D. P. 1991. *Linear Network Optimization: Algorithms and Codes*, MIT Press, Cambridge, MA.

[BeT89] BERTSEKAS, D. P., and TSITSIKLIS, J. N. 1989. *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, Englewood Cliffs, N.J.

[BGG84] BERTSEKAS, D. P., GAFNI, E. M., and GALLAGER, R. G. 1984. Second Derivative Algorithms for Minimum Delay Distributed Routing in Networks, *IEEE Trans. Comm.*, COM-32:911–919.

[BGS80] BIALLY, T., GOLD, B., and SENEFF, S. 1980. A Technique for Adaptive Voice Flow Control in Integrated Packet Networks, *IEEE Trans. Comm.*, COM-28:325–333.

[BGT84] BERTSEKAS, D. P., GENDRON, R., and TSAI, W. K. 1984. *Implementation of an Optimal Multicommodity Network Flow Algorithm Based on Gradient Projection and a Path Flow Formulation* (Report LIDS-P-1364). Cambridge, MA: MIT Laboratory for Information and Decision Systems.

[BGV79] BERTSEKAS, D. P., GAFNI, E. M., and VASTOLA, K. S. 1979. Validation of Algorithms for Optimal Routing of Flow in Networks, *Proc. 1978 IEEE Conf. Dec. Control*, San Diego, CA.

[Bin75] BINDER, R. 1975. A Dynamic Packet Switching System for Satellite Broadcast Channels, *Proc. ICC*, pp. 41.1–41.5.

[Bla83] BLAHUT, R. E. 1983. *Theory and Practice of Error Control Codes.* Reading, MA: Addison-Wesley.

[BLL84] BURMAN, D. Y., LEHOCZKY, J. P., and LIM, Y. 1984. Insensitivity of Blocking Probabilities in a Circuit-Switching Network, *Advances in Applied Probability*, 21:850–859.

[BoF77] BOORSTYN, R. R., and FRANK, H. 1977. Large-Scale Network Topological Optimization, *IEEE Trans. Comm.*, COM-25:29–47.

[BoM86] BOXMA, O. J., and MEISTER, B. 1986. Waiting-Time Approximations for Cyclic-Service Systems with Switch-Over Times, *Perform. Eval. Rev.*, 14:254–262.

[BoS82] BOCHMANN, G. V., and SUNSHINE, C. A. 1982. A Survey of Formal Methods, in P. Green (Ed.), *Computer Network Architecture.* New York: Plenum.

[BrB80] BRUELL, S. C., and BALBO, G., 1980. *Computational Algorithms for Closed Queueing Networks.* New York: Elsevier/North-Holland.

[BrC91a] BRASSIL, J. T., and CRUZ, R. L. 1991. Nonuniform Traffic in the Manhattan Street Network, *Proceedings of ICC '91*, 3:1647–1651.

[BrC91b] BRASSIL, J. T., and CRUZ, R. L. 1991. Bounds on Maximum Delay in Networks with Deflection Routing, *Proceedings of the 29th Allerton Conference on Communications, Control, and Computing*, Monticello, IL.

[Bux81] BUX, W. 1981. Local Area Networks: A Performance Comparison, *IEEE Trans. Comm.*, COM-29:1465–1473.

[CaC68] CANNON, M. D., and CULLUM, C. D. 1968. A Tight Upper Bound on the Rate of Convergence of the Frank–Wolfe Algorithm, *SIAM J. Control Optim.*, 6:509–516.

[CaH75] CARLEIAL, A. B., and HELLMAN, M. E. 1975. Bistable Behavior of Slotted Aloha-Type Systems, *IEEE Trans. Comm.*, COM-23:401–410.

[Cap77] CAPETANAKIS, J. I. 1977. *The Multiple Access Broadcast Channel: Protocol and Capacity Considerations*, Ph.D. dissertation, MIT, Dept. of Electrical Engineering and Computer Science, Cambridge, MA. Also 1979, *IEEE Trans. Inform. Theory*, IT-25:505–515.

[Car80] CARLSON, D. E. 1980. Bit-Oriented Data Link Control Procedures, *IEEE Trans. Comm.*, COM-28:455–467. (Also in [Gre82].)

[CFL79] CHLAMTAC, I., FRANTA W., and LEVIN, K. D. 1979. BRAM: The Broadcast Recognizing Access Method, *IEEE Trans. Comm.*, COM-27:1183–1190.

[CFN77] CORNUEJOLS, G., FISHER, M. L., and NEMHAUSER, G. L. 1977. Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms, *Management Sci.*, 23:789–810.

[CGK90] CIDON, I., GOPAL, I. S., KAPLAN, M., and KUTTEN, S. 1990. Distributed Control for PARIS, *Proc. of the 9th Annual ACM Symposium on Principles of Computing*, Quebec, Can.

[CiG88] CIDON, I., and GOPAL, I. S. 1988. PARIS: An Approach to Integrated High-Speed Private Networks, *International Journal of Digital and Analog Cabled Systems*, 1:77–85.

[ClC81] CLARK, G. C., and CAIN, J. B. 1981. *Error Correction Coding for Digital Communication*. New York: Plenum.

[CoG89] CONWAY, A. E., and GEORGANAS, N. D. 1989. *Queueing Networks—Exact Computational Algorithms*, MIT Press, Cambridge, MA.

[Com88] COMER, D. 1988. *Internetworking with TCP/IP, Principles, Protocols, and Architectures*, Prentice-Hall, Englewood Cliffs.

[Coo70] COOPER, R. B. 1970. Queues Served in Cyclic Order: Waiting Times, *Bell Systems Tech. J.*, 49:399–413.

[Coo81] COOPER, R. B. 1981, *Introduction to Queueing Theory* (2nd ed.). New York: Elsevier/ North-Holland.

[CPR78] CLARK, D. D., POGRAN, K. T., and REED, D. P. 1978. An Introduction to Local Area Networks. *Proc. IEEE*, pp. 1497–1517.

[Cru91a] CRUZ, R. L. 1991. A Calculus for Network Delay, Part I: Network Elements in Isolation, *IEEE Trans. on Inform. Theory*, IT-37:114–131.

[Cru91b] CRUZ, R. L. 1991. A Calculus for Network Delay, Part II: Network Analysis, *IEEE Trans. on Inform. Theory*, IT-37:132–141.

[CRW73] CROWTHER, W., RETTBURG, R., WALDEN, D., ORNSTEIN, S., and HEART, F. 1973. A System for Broadcast Communication: Reservation Aloha, *Proc. 6th Hawaii Internat. Conf. Syst. Sci.*, pp. 371–374.

[Daf71] DAFERMOS, S. C. 1971. An Extended Traffic Assignment Model with Applications to Two-Way Traffic, *Trans. Sci.*, 5:366–389.

[Daf80] DAFERMOS, S. C. 1980. Traffic Equilibrium and Variational Inequalities, *Trans. Sci.*, 14:42–54.

[Dan63] DANTZIG, G. B. 1963. *Linear Programming and Extensions*. Princeton, NJ: Princeton University Press.

[DeK81] DEMBO, R. S., and KLINCEWICZ, J. G. 1981. A Scaled Reduced Gradient Algorithm for Network Flow Problems with Convex Separable Costs, *Math. Programming Stud.*, 15:125–147.

[DGK79] DIAL, R., GLOVER, F., KARNEY, D., and KLINGMAN, D. 1979. A Computational Analysis of Alternative Algorithms and Labeling Techniques for Finding Shortest Path Trees, *Networks*, 9:215–248.

[DiK85] DISNEY, R. L., and KONIG, D. 1985. Queueing Networks: A Survey of Their Random Processes, *SIAM Rev.*, 27:335–403.

[DiS80] DIJKSTRA, E. W., and SHOLTEN, C. S. 1980. Termination Detection for Diffusing Computations, *Inf. Proc. Lett.*, 11:1–4.

[Dun79] DUNN, J. C. 1979. Rates of Convergence of Conditional Gradient Algorithms near Singular and Nonsingular Extremals, *SIAM J. Control Optim.*, 17:187–211.

[Eis79] EISENBERG, M. 1979. Two Queues with Alternating Service, *SIAM J. Appl. Math.*, 20:287–303.

[ELL90a] ECKBERG, A. E., LUAN, D. T., and LUCANTONI, D. M. 1990. Bandwidth Management: A Congestion Control Strategy for Broadband Packet Networks—Characterizing the Throughput-Burstiness Filter, *Computer Networks and ISDN Systems*, 20:415–423.

[ELL90b] ECKBERG, A. E., LUAN, D. T., and LUCANTONI, D. M. 1990. An Approach to Controlling Congestion in ATM Networks, *International Journal of Digital and Analog Communication Systems*, 3:199–209.

[ELS90] ESCOBAR, J., LAUER, G., and STEENSTRUP, M. 1990. Performance Analysis of Rate-Based Congestion Control Algorithm for Receiver-Directed Packet-Radio Networks, *Proc. of MILCOM '90*.

[EpB81] EPHREMIDES, A., and BAKER, D. J. 1981. The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm, *IEEE Trans. Comm.*, COM-29:1694–1701.

[Eph86] EPHREMIDES, A. 1986. The Routing Problem in Computer Networks, in I. F. Blake and H. V. Poor (Eds.), *Communication and Networks*. New York: Springer-Verlag, pp. 299–324.

[EpV89] EPHREMIDES, A., and VERDU, S. 1989. Control and Optimization Methods in Communication Network Problems, *IEEE Trans. Automat. Control*, 34:930–942.

[EsW66] ESSAU, L. R., and WILLIAMS, K. C. 1966. On Teleprocessing System Design, *IBM Systems J.*, 5:142–147.

[Eve75] EVEN, S. 1975. An Algorithm for Determining Whether the Connectivity of a Graph Is at Least k, *SIAM J. Comput.*, 4:393–396.

[EVW80] EPHREMIDES, A., VARAIYA, P., and WALRAND, J. 1980. A Simple Dynamic Routing Problem, *IEEE Trans. Automat. Control*, AC-25:690–693.

[FaL72] FARBER, D. J., and LARSON, K. C. 1972. *The System Architecture of the Distributed Computer System—The Communications System.*, paper presented at the Symposium on Computer Networks, Polytechnic Institute of Brooklyn, New York.

[FaN69] FARMER, W. D., and NEWHALL, E. E. 1969. An Experimental Distributed Switching System to Handle Bursty Computer Traffic, *Proc. ACM Symp. Problems Optim. Data Comm. Syst.*, pp. 1–33.

[FeA85] FERGUSON, M. J., and AMINETZAH, Y. J. 1985. Exact Results for Nonsymmetric Token Ring Systems, *IEEE Trans. Comm.*, COM-33:223–231.

[FeA89] FETTEROLF, P. C., and ANANDALINGAM, G. 1989. Optimizing Interconnection of Local Area Networks: An Approach Using Simulated Annealing, *Univ. of Pennsylvania Report*; to appear in Annals of Operations Research.

[Fer75] FERGUSON, M. J. 1975. A Study of Unslotted Aloha with Arbitrary Message Lengths, *Proc. 4th Data Comm. Symp.*, Quebec, Canada, pp. 5.20–5.25.

[FGK73] FRATTA, L., GERLA, M., and KLEINROCK, L. 1973. The Flow Deviation Method: An Approach to Store-and-Forward Communication Network Design, *Networks*, 3:97–133.

[Fin79] FINN, S. G. 1979. Resynch Procedures and a Fail-Safe Network Protocol, *IEEE Trans. Comm.*, COM-27:840–845.

[FlN74] FLORIAN, M., and NGUYEN, S. 1974. A Method for Computing Network Equilibrium with Elastic Demand, *Transport. Sci.*, 8:321–332.

[FiT84] FINE, M., and TOBAGI, F. A. 1984. Demand Assignment Multiple Access Schemes in Broadcast Bus Local Area Networks, *IEEE Trans. Comput.*, C-33:1130–1159.

[FKL86] FUJIWARA, T., KASAMI, T., and LIN, S. 1986. Error Detecting Capabilities of the Shortened Hamming Codes Adopted for Error Detection in IEEE Standard 802.3, *Int. Symp. I.T.*, Ann Arbor, Mich.

[FoF62] FORD, L. R., Jr., and FULKERSON, D. R. 1962. *Flows in Networks*. Princeton, NJ: Princeton University Press.

[FoS78] FOSCHINI, G. J., and SALZ, J. 1978. A Basic Dynamic Routing Problem and Diffusion, *IEEE Trans. Comm.*, COM-26:320–327.

[FrW56] FRANK, M., and WOLFE, P. 1956. An Algorithm for Quadratic Programming, *Naval Res. Logist. Quart.*, 3:149–154.

[FuC85] FUHRMANN, S. W., and COOPER, R. B. 1985. Stochastic Decompositions in the $M/G/1$ Queue with Generalized Vacations, *Oper. Res.*, 33:1117–1129.

[GaB81] GAFNI, E. M., and BERTSEKAS, D. P. 1981. Distributed Routing Algorithms for Networks with Frequently Changing Topology, *IEEE Trans. Comm.*, COM-29:11–18.

[GaB83] GAFNI, E. M., and BERTSEKAS, D. P. 1983. *Asymptotic Optimality of Shortest Path Routing* (Report LIDS-P-1307). Cambridge, MA: MIT Laboratory for Information and Decision Systems. Also 1987, *IEEE Trans. Inform. Theory*, IT-33:83–90.

[GaB84a] GAFNI, E. M., and BERTSEKAS, D. P. 1984. Two-Metric Projection Methods for Constrained Optimization, *SIAM J. Control Optim.*, 22:936–964.

[GaB84b] GAFNI, E. M., and BERTSEKAS, D. P. 1984. Dynamic Control of Session Input Rates in Communication Networks, *IEEE Trans. Automat. Control*, AC-29:1009–1016.

[Gaf79] GAFNI, E. M. 1979. *Convergence of a Routing Algorithm*, M.S. thesis, University of Illinois, Dept. of Electrical Engineering, Urbana, IL.

[Gaf82] GAFNI, E. M. 1982. *The Integration of Routing and Flow Control for Voice and Data in Integrated Packet Networks*, Ph.D. thesis, MIT, Dept. of Electrical Engineering and Computer Science, Cambridge, MA.

[GaG80] GALLAGER, R. G., and GOLESTAANI, S. J. 1980. Flow Control and Routing Algorithms for Data Networks, *Proc. 5th Internat. Conf. Comput. Comm.*, pp. 779–784.

[GaH83] GAVISH, B., and HANTLER, S. 1983. An Algorithm for Optimal Route Selection in SNA Networks, *IEEE Trans. Comm.*, COM-31:1154–1161.

[Gal68] GALLAGER, R. G. 1968. *Information Theory and Reliable Communications*. New York: Wiley.

[Gal77] GALLAGER, R. G. 1977. A Minimum Delay Routing Algorithm Using Distributed Computation, *IEEE Trans. Comm.*, COM-23:73–85.

[Gal78] GALLAGER, R. G. 1978. Conflict Resolution in Random Access Broadcast Networks, *Proc. AFOSR Workshop Comm. Theory Appl.*, Provincetown, MA, pp. 74–76.

[Gal81] GALLAGER, R. G. 1981. Applications of Information Theory for Data Communication Networks, in J. Skwirzynski (Ed.), *New Concepts in Multi-user Communication* (Series E, No. 43). Alphen aan den Rijn, The Netherlands: NATO Advanced Study Institutes (Sijthoff en Noordhoff).

[GaP88] GALLO, G. S., and PALLOTINO, S. 1988. Shortest Path Algorithms, *Annals of Operations Research*, 7:3–79.

[Gar87] GARCIA-LUNA-ACEVES, J. J. 1987. A New Minimum-Hop Routing Algorithm, *IEEE INFOCOM '87 Proceedings*, pp. 170–180.

[Gav85] GAVISH, B. 1985. Augmented Lagrangean-Based Algorithms for Centralized Network Design, *IEEE Trans. Comm.*, COM-33:1247–1257.

[GeK77] GERLA, M., and KLEINROCK, L. 1977. On the Topological Design of Distributed Computer Networks, *IEEE Trans. Comm.*, COM-25:48–60.

[GeK80] GERLA, M., and KLEINROCK, L. 1980. Flow Control: A Comparative Survey, *IEEE Trans. Comm.*, COM-28:553–574.

[GeK88] GERLA, M., and KLEINROCK, L. 1988. Congestion Control in Interconnected LANs, *IEEE Network*, 2:72–76.

[GeP85] GEORGIADIS, L., and PAPANTONI-KAZAKOS, P. 1985. *A 0.487 Throughput Limited Sensing Algorithm*. Storrs, CT: University of Connecticut.

[GeP87] GELEMBE, E., and PUJOLLE, G. 1988. *Introduction to Queueing Networks*. J. Wiley, N.Y.

[GeY82] GEORGE, F. D., and YOUNG, G. E. 1982. SNA Flow Control: Architecture and Implementation, *IBM Syst. J.*, 21:179–210.

[GGM85] GOODMAN, J., GREENBERG, A. G., MADRAS, N., and MARCH, P. 1985. On the Stability of Ethernet, *Proc. 17th Annual ACM Symp. Theory Comput.*, Providence, RI, pp. 379–387.

[GHS83] GALLAGER, R. G., HUMBLET, P. A., and SPIRA, P. M. 1983. A Distributed Algorithm for Minimum-Weight Spanning Trees, *ACM Trans. Programming Language Syst.*, 5:66–77.

[Gol64] GOLDSTEIN, A. A., 1964. Convex Programming in Hilbert Space, *Bull. Am. Math. Soc.*, 70:709–710.

[Gol80] GOLESTAANI, S. J. 1980. *A Unified Theory of Flow Control and Routing on Data Communication Networks*, Ph.D. thesis, MIT, Dept. of Electrical Engineering and Computer Science, Cambridge, MA.

[Gol90a] GOLESTAANI, S. J. 1990. Congestion-Free of Real-Time Traffic in Packet Networks, *Proc. of IEEE INFOCOM '90.* San Francisco, Cal., pp. 527–536.

[Gol90b] GOLESTAANI, S. J. 1990. A Framing Strategy for Congestion Management, *Proc. of SIGCOM '90.*

[Gop85] GOPAL, I. S., 1985. Prevention of Store-and-Forward Deadlock in Computer Networks, *IEEE Trans. Comm.*, COM-33:1258–1264.

[Gra72] GRAY, J. P. 1972. Line Control Procedures, *Proc. IEEE*, pp. 1301–1312.

[GrB83] GROVER, G. A., and BHARATH-KUMAR, K. 1983. Windows in the Sky—Flow Control in SNA Networks with Satellite Links, *IBM Systems J.*, 22:451–463.

[Gre82] GREEN, P. E. 1982. *Computer Network Architectures and Protocols.* New York: Plenum.

[Gre84] GREEN, P. E. 1984. Computer Communications: Milestones and Prophecies, *IEEE Trans. Comm.*, COM-32:49–63.

[Gre86] GREEN, P. E. 1986. Protocol Conversion, *IEEE Trans. Comm.*, COM-34:257–268.

[GrH85] GROSS, D., and HARRIS, C. M. 1985. *Fundamentals of Queueing Theory* (2nd ed.). New York: Wiley.

[GrH89] GREENBERG, A. G., and HAJEK, B. 1989. Deflection Routing in Hypercube Networks, *IEEE Trans. Comm.*, to appear.

[Gun81] GUNTHER, K. D. 1981. Prevention of Deadlocks in Packet-Switched Data Transport Systems, *IEEE Trans. Comm.*, COM-29:512–524.

[HaC90] HAJEK, B., and CRUZ, R. L. 1990. On the Average Delay for Routing Subject to Independent Deflections, *IEEE Trans. of Inform. Theory*, to appear.

[HaG86] HAHNE, E. L., and GALLAGER, R. G. 1986. *Round-Robin Scheduling for Fair Flow Control in Data Communication Networks* (Report LIDS-P-1537). Cambridge, MA: MIT Laboratory for Information and Decisions Systems.

[Hah86] HAHNE, E. 1986. Round-Robin Scheduling for Fair Flow Control in Data Communication Networks Ph.D. thesis, MIT, Dept. of Electrical Engineering and Computer Science, Cambridge, MA.

[Haj82] HAJEK, B. 1982. Birth-and-Death Processes with Phases and General Boundaries, *J. Appl. Problems*, 19:488–499.

[Haj88] HAJEK, B. 1988. Cooling Schedules for Optimal Annealing, *Math of Operations Research*, 13:311–329.

[Haj91] HAJEK, B. 1991. Bounds on Evacuation Time for Deflection Routing, *Distributed Computing*, 5:1–5.

[HaL82] HAJEK, B., and VAN LOON, T. 1982. Decentralized Dynamic Control of a Multiaccess Broadcast Channel, *IEEE Trans. Automat. Control*, AC-27:559–569.

[HaO84] HAJEK, B., and OGIER, R. G. 1984. Optimal Dynamic Routing in Communication Networks with Continuous Traffic, *Networks*, 14:457–487.

[Has72] HASHIDA, O. 1972. Analysis of Multiqueue, *Rev. Electron. Comm. Lab.*, 20:189–199.

[Hay76] HAYES, J. F. 1976. *An Adaptive Technique for Local Distribution* (Bell Telephone Laboratory Technical Memo TM-76-3116-1.) (Also 1978, *IEEE Trans. Comm.*, COM-26:1178–1186.)

[Hay81] HAYDEN, H. 1981. *Voice Flow Control in Integrated Packet Networks* (Report LIDS-TH-1152). Cambridge, MA: MIT Laboratory for Information and Decision Systems.

[Hay84] HAYES, J. F. 1984. *Modeling and Analysis of Computer Communications Networks*. New York: Plenum.

[HCM90] HAHNE, E. L., CHOUDHURY, A., and MAXEMCHUK, N. 1990. Improving the Fairness of Distributed-Queue-Dual-Bus Networks, *Infocom*, San Francisco, Ca., pp. 175–184.

[HeS82] HEYMAN, D. P., and SOBEL, M. J. 1982. *Stochastic Models in Operations Research*, Vol. 1. New York: McGraw-Hill.

[HlG81] HLUCHYJ, M. G., and GALLAGER, R. G. 1981. Multiaccess of a Slotted Channel by Finitely Many Users, *Proc. Nat. Telecomm. Conf.*, New Orleans, LA. (Also LIDS Report P-1131, MIT, Cambridge, MA, August 1981.)

[HSS86] HUMBLET, P. A., SOLOWAY, S. R., and STEINKA, B. 1986. *Algorithms for Data Communication Networks—Part 2*. Codex Corp.

[HuB85] HUANG, J.-C., and BERGER, T. 1985. Delay Analysis of the Modified 0.487 Contention Resolution Algorithm, *IEEE Trans. Inform. Theory*, IT-31:264–273.

[HuM80] HUMBLET, P. A., and MOSELY, J. 1980. Efficient Accessing of a Multiaccess Channel, *Proc. IEEE 19th Conf. Dec. Control*, Albuquerque, NM. (Also LIDS Report P-1040, MIT, Cambridge, MA, Sept. 1980.)

[Hum78] HUMBLET, P. A. 1978. *Source Coding for Communication Concentrators* (Report ESL-R-798). Cambridge, MA: MIT.

[Hum83] HUMBLET, P. A. 1983. A Distributed Algorithm for Minimum Weight Directed Spanning Trees, *IEEE Trans. Comm.*, COM-31:756–762.

[Hum86] HUMBLET, P. A. 1986. On the Throughput of Channel Access Algorithms with Limited Sensing, *IEEE Trans. Comm.*, COM-34:345–347.

[Hum91] HUMBLET, P. A. 1991. Another Adaptive Distributed Dijkstra Shortest Path Algorithm, *IEEE Trans. on Comm.*, COM-39:995–1003; also *MIT Laboratory for Information and Decision Systems Report LIDS-P-1775*, May 1988.

[HuS86] HUMBLET, P. A., and SOLOWAY, S. R. 1986. *Algorithms for Data Communication Networks —Part I*. Codex Corp.

[HuS88] HUMBLET, P. A. and S. SOLOWAY. 1988/1989. Topology Broadcast Algorithms, *Computer Networks and ISDN Systems*, 16:179–186; also *Codex Corporation Research Report;* revised May 1987 as Report LIDS-P-1692.

[IbC89] IBE, O. C., and CHENG, X. 1989. Approximate Analysis of Asymmetric Single-Service Token-passing Systems, *IEEE Trans. on Comm.*, COM-36:572–577.

[Ibe81] IBE, O. C. 1981. *Flow Control and Routing in an Integrated Voice and Data Communication Network*, Ph.D. thesis, MIT, Dept. of Electrical Engineering and Computer Science, Cambridge, MA.

[IEE86] *IEEE Journal on Selected Areas in Communications, Special Issue on Network Performance Evaluation*, SAC-4(6), Sept. 1986.

[IEE88] *IEEE Network*, Vol. 2, No. 1, 1988.

[IEE89] *IEEE Journal on Selected Areas in Communications*, Special Issue on Telecommunications Network Design and Planning, Vol. 7, No. 8, 1989.

[**Jac57**] JACKSON, J. R. 1957. Networks of Waiting Lines, *Oper. Res.*, 5:518–521.

[**Jaf81**] JAFFE, J. M. 1981. A Decentralized "Optimal" Multiple-User Flow Control Algorithm, *IEEE Trans. Comm.*, COM-29:954–962.

[**Jai68**] JAISWAL, N. K. 1968. *Priority Queues*. New York: Academic Press.

[**JaM82**] JAFFE, J. M., and MOSS, F. M. A. 1982. Responsive Routing Algorithm for Computer Networks, *IEEE Trans. on Comm.*, COM-30:1758–1762.

[**JBH78**] JACOBS, I. M., BINDER, R., and HOVERSTEN, E. V. 1978. General Purpose Packet Satellite Networks, *Proc. IEEE*, pp. 1448–1468.

[**Kap79**] KAPLAN, M. 1979. A Sufficient Condition for Nonergodicity of a Markov Chain, *IEEE Trans. Inform. Theory*, IT-25:470–471.

[**KaT75**] KARLIN, S., and TAYLOR, H. M. 1975. *A First Course in Stochastic Processes*. New York: Academic Press.

[**Kau81**] KAUFMAN, J. S. 1981. Blocking in a Shared Resource Environment, *IEEE Trans. Comm.*, COM-29:1474–1481.

[**KeB83**] KERSHENBAUM, A., and BOORSTYN, R. R. 1983. Centralized Teleprocessing Network Design, *Networks*, 13:279–293.

[**KeC90**] KEY, P. B., and COPE, G. A. 1990. Distributed Dynamic Routing Schemes, *IEEE Communications Magazine*, 28:54–64.

[**KeH63**] KEUHN, A. A., and HAMBURGER, M. J. 1963. A Heuristic Program for Locating Warehouses, *Management Sci.*, 9:643–666.

[**Kei79**] KEILSON, J. 1979. *Markov-Chain Models—Rarity and Exponentiality*. New York: Springer-Verlag.

[**Kel79**] KELLY, F. P. 1979. *Reversibility and Stochastic Networks*. New York: Wiley.

[**Kel85**] KELLY, F. P. 1985. Stochastic Models of Computer Communication Systems, *J. Roy. Statist. Soc. Ser. B*, 47(1).

[**Kel86**] KELLY, F. P. 1986. Blocking Probabilities in Large Circuit-Switched Networks, *Adv. Apl. Prob.*, 18:473–505.

[**KGB78**] KAHN, R. E., GRONEMEYER, S. A., BURCHFIEL, J., and KUNZELMAN, R. C. 1978. Advances in Packet Radio Technology, *Proc. IEEE*, pp. 1468–1496.

[**KGV83**] KIRKPATRICK, S., GELATT, C. D., Jr., and VECCHI, M. P. 1983. Optimization by Simulated Annealing, *Science*, 220:671–680.

[**Khu72**] KHUMAWALA, B. M. 1972. An Efficient Branch and Bound Algorithm for the Warehouse Location Problem, *Management Sci.*, 18:B718–B731.

[**KhZ89**] KHANNA, A., and ZINKY, J. 1989. The Revised ARPANET Routing Metric, *Proc. of SIGCOM '89*.

[**Kin62**] KINGMAN, J. F. C. 1962. Some Inequalities for the Queue *GI/G/1*, *Biometrika*, 49:315–324.

[**Kle64**] KLEINROCK, L. 1964. *Communication Nets: Stochastic Message Flow and Delay*. New York: McGraw-Hill.

[**Kle69**] KLEITMAN, D. 1969. Methods for Investigating the Connectivity of Large Graphs, *IEEE Trans. Circuit Theory*, CT-16:232–233.

[**Kle75**] KLEINROCK, L. 1975. *Queueing Systems*, Vol. 1. New York: Wiley.

[**Kle76**] KLEINROCK, L. 1976. *Queueing Systems*, Vol. 2. New York: Wiley.

[**KlL75**] KLEINROCK, L., and LAM, S. 1975. Packet Switching in Multiaccess Broadcast Channel: Performance Evaluation, *IEEE Trans. Comm.*, COM-23:410–423.

[**KlO77**] KLEINROCK, L., and OPDERBECK, H. 1977. Throughput in the ARPANET—Protocols and Measurements, *IEEE Trans. Comm.*, COM-25:95–104.

[KlS80] KLEINROCK, L., and SCHOLL. 1980. Packet Switching in Radio Channels: New Conflict Free Multiple Access Schemes, *IEEE Trans. Comm.*, COM-28:1015–1029.

[KlT75] KLEINROCK, L., and TOBAGI, F. A. 1975. Packet Switching in Radio Channels: Part 1: CSMA Modes and Their Throughput-Delay Characteristics, *IEEE Trans. Comm.*, COM-23:1400–1416.

[Kol36] KOLMOGOROV, A. 1936. Zur Theorie der Markoffschen Ketten, *Mathematische Annalen*, 112:155–160.

[Kri90] KRISHNAN, K. R. 1990. Markov Decision Algorithms for Dynamic Routing, *IEEE Communications Magazine*, 28:66–69.

[Kue79] KUEHN, P. J. 1979. Multiqueue Systems with Nonexhaustive Cyclic Service, *Bell System Tech. J.*, 58:671–698.

[KuR82] KUMMERLE, K., and REISER, M. 1982. Local Area Networks—An Overview, *J. Telecomm. Networks*, 1(4).

[KuS89] KUMAR, P. R., and SEIDMAN, T. I. 1989. Distributed Instabilities and Stabilization Methods in Distributed Real-Time Scheduling of Manufacturing Systems, *IEEE Trans. Automat. Control*, AC-35:289–298.

[LaH84] LANPHONGPANICH, S., and HEARN, D. 1984. Simplicial Decomposition of the Asymmetric Traffic Assignment Problems, *Trans. Res.*, 18B:123–133.

[LaK75] LAM, S., and KLEINROCK, L. 1975. Packet Switching in a Multiaccess Broadcast Channel: Dynamic Control Procedures, *IEEE Trans. Comm.*, COM-23:891–904.

[LaL86] LAM, Y. F., and LI, V. O. K. 1986. An Improved Algorithm for Performance Analysis of Networks with Unreliable Components, *IEEE Trans. Comm.*, COM-34:496–497.

[Lam80] LAM, S. 1980. A Carrier Sense Multiple Access Protocol for Local Networks, *Comput. Networks*, 4:21–32.

[Las70] LASDON, L. S. 1970. *Optimization Theory for Large Systems.* New York: Macmillan.

[LeG89] LE GALL, F. 1989. About Loss Probabilities for General Routing Policies in Circuit-Switched Networks, *IEEE Transactions on Comm.*, COM-37:57–59.

[Lei80] LEINER, B. M. 1980. A Simple Model for Computation of Packet Radio Network Communication Performance, *IEEE Trans. Comm.*, COM-28:2020–2023.

[LeP65] LEVITIN, E. S., and POLJAK, B. T. 1965. Constrained Minimization Methods, *USSR Comput. Math. Phys.*, 6:1–50.

[LeY89] LEE, M.-J., and YEE, J. R. 1989. An Efficient Near-Optimal Algorithm for the Joint Traffic and Trunk Routing Problem in Self-Planning Networks, *Proc. IEEE INFOCOM '89*, pp. 127–135.

[LiF82] LIMB, J. O., and FLORES, C. 1982. Description of Fasnet, or Unidirectional Local Area Communications Network, *Bell System Tech. J.*

[Lim84] LIMB, J. O. 1984. Performance of Local Area Networks at High Speed, *IEEE Trans. Comm.*, COM-32:41–45.

[LiS84] LI, V. O. K., and SILVESTER, J. A. 1984. Performance Analysis of Networks with Unreliable Components, *IEEE Trans. Comm.*, COM-32:1105–1110.

[Lit61] LITTLE, J. 1961. A Proof of the Queueing Formula $L = \lambda W$, *Oper. Res. J.*, 18:172–174.

[LMF88] LYNCH, N., MANSOUR, Y., and FEKETE, A. 1988. The Data Link Layer: Two Impossibility Results, *Proceedings of the 7th Annual ACM Symposium on Principles of Distributed Computing.* Toronto, Can., pp. 149–170.

[Luc90] LUCKY, R. 1989. *Information, Man & Machine.* New York: St. Martins Press.

[Lue84] LUENBERGER, D. G. 1984. *Linear and Nonlinear Programming.* Reading MA: Addison-Wesley.

[MaF85] MATHYS, P., and FLAJOLET, P. 1985. Q-ary Collision Resolution Algorithms in Random-Access Systems with Free or Blocked Channel Access, *IEEE Trans. Inform. Theory*, IT-31:217–243.

[Mal86] MALIS, A. G. 1986. *PSN End-to-End Functional Specification* (RFC 979). BBN Communications Corp., Network Working Group.

[MaN85] MAXEMCHUK, N. F., and NETRAVALI, A. N. 1985. Voice and Data on a CATV Network, *IEEE J. Select. Areas Comm.*, SAC-3:300–311.

[MaP88] MAGGS, B. M., and PLOTKIN, S. A. 1988. Minimum-Cost Spanning Tree as a Path Finding Problem, *Inf. Proc. Lett.*, 26:291–293.

[Mas80] MASSEY, J. L. 1980. *Collision-Resolution Algorithms and Random Access Communications* (Report UCLA-ENG-8016). Los Angeles: University of California.

[Max87] MAXEMCHUK, N. F. 1987. Routing in the Manhattan Street Network, *IEEE Trans. on Communications*, COM-35:503–512.

[McS77] MCGREGOR, P. V., and SHEN, D. 1977. Network Design: An Algorithm for the Access Facility Location Problem, *IEEE Trans. Comm.*, COM-25:61–73.

[McW77] MCQUILLAN, J. M., and WALDEN, D. C. 1977. The ARPANET Design Decisions, *Networks*, 1.

[MeB76] METCALFE, R. M., and BOGGS, D. R. 1976. Ethernet: Distributed Packet Switching for Local Computer Networks, *Comm. ACM*, 395–404.

[Met73] METCALFE, R. 1973. *Steady State Analysis of a Slotted and Controlled Aloha System with Blocking*, paper presented at 6th Hawaii Conf. System Sci., Honolulu.

[Mik79] MIKHAILOV, V. A. 1979. *Methods of Random Multiple Access*, Candidate Engineering thesis, Moscow Institute of Physics and Technology, Moscow.

[Min89] MINZNER, S. E. 1989. Broadband ISDN and Asynchronous Transfer Mode (ATM), *IEEE Commun. Mag.*, 27:17–24.

[MiT81] MIKHAILOV, V. A., and TSYBAKOV, B. S. 1981. Upper Bound for the Capacity of a Random Multiple Access System, *Problemy Peredachi Inform.* (USSR), 17:90–95.

[MoH85] MOSELY, J., and HUMBLET, P. A. 1985. A Class of Efficient Contention Resolution Algorithms for Multiple Access Channels, *IEEE Trans. Comm.*, COM-33:145–151.

[MoS82] MOSS, F. H., and SEGALL, A. 1982. An Optimal Control Approach to Dynamic Routing in Networks, *IEEE Trans. Automat. Control*, AC-27:329–339.

[MoS86] MONMA, C. L., and SHENG, D. D. 1986. Backbone Network Design and Performance Analysis: A Methodology for Packet Switching Networks, *IEEE J. Select. Areas Comm.*, SAC-4:946–965.

[Mos84] MOSELY, J. 1984. *Asynchronous Distributed Flow Control Algorithms*, Ph.D. thesis, MIT, Dept. of Electrical Engineering and Computer Science, Cambridge, MA.

[MRR78] MCQUILLAN, J. M., RICHER, I., ROSEN, E. C., and BERTSEKAS, D. P. 1978. *ARPANET Routing Algorithm Improvements, Second Semiannual Report* (prepared for ARPA and DCA). : Bolt, Beranek, and Newman, Inc.

[MRR80] MCQUILLAN, J. M., RICHER, I., and ROSEN, E. C. 1980. The New Routing Algorithm for the ARPANET, *IEEE Trans. Comm.*, COM-28:711–719.

[NBH88] NEWMAN, R. M., BUDRIKIS, Z., and HULLETT, J. 1988. Standards for Metropolitan Area Networks, *IEEE Commun. Mag.*, 26:20–28.

[Neu81] NEUTS, M. F. 1981. *Matrix-Geometric Solutions in Stochastic Models—An Algorithmic Approach.* Baltimore, MD: The Johns Hopkins University Press.

[NgH87] NG, M. J. T., and HOANG, D. B. 1987. Joint Optimization of Capacity and Flow Assignment in a Packet Switched Communications Network, *IEEE Trans. Comm.*, COM-35:202–209.

[NiS74] NISNEVITCH, L., and STRASBOURGER, E. 1974. Decentralized Priority in Data Communication, *Proc. 2nd Annual Symp. Comput. Architecture.*

[NoT78] NOMURA, M., and TSUKAMOTO, K. 1978. Traffic Analysis of Polling Systems, *Trans. Inst. Electron. Comm. Eng.* (Japan), J61-B:600–607.

[Nyq28] NYQUIST, H. 1928. Certain Topics in Telegraph Transmission Theory, *Trans. AIEE*, 47:617–644.

[PaG91a] PAREKH, A. K., and GALLAGER, R. G. 1991. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case, Laboratory for Information and Decision Systems Report LIDS-P-2040, M.I.T., Cambridge, MA.

[PaG91b] PAREKH, A. K., and GALLAGER, R. G. 1991. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple Node Case, Laboratory for Information and Decision Systems Report LIDS-P-2074, M.I.T., Cambridge, MA.

[Pak69] PAKES, A. G. 1969. Some Conditions for Ergodicity and Recurrence of Markov Chains, *Oper. Res.*, 17:1059–1061.

[Pap74] PAPE, U. 1974. Implementation and Efficiency of Moore-Algorithms for the Shortest Route Problem, *Math. Programming*, 7:212–222.

[PaS82] PAPADIMITRIOU, C. H., and STEIGLITZ, K. 1982. *Combinatorial Optimization: Algorithms and Complexity.* Englewood Cliffs, NJ: Prentice Hall.

[Per83] PERLMAN, R. 1983. Fault-Tolerant Broadcast of Routing Information, *Comput. Networks*, 7:395–405. (Also *Proc IEEE Infocom '83*, San Diego.)

[Per88] PERLMAN, R. 1988. Network Layer Protocols with Byzantine Robustness, Ph.D. Thesis, MIT Dept. of EECS.

[Pip81] PIPPENGER, N. 1981. Bounds on the Performance of Protocols for a Multiple Access Broadcast Channel, *IEEE Trans. Inform. Theory*, IT-27:145–151.

[PiW82] PINEDO, M., and WOLFF, R. W. 1982. A Comparison between Tandem Queues with Dependent and Independent Service Times, *Oper. Res.*, 30:464–479.

[Pol71] POLAK, E. 1971. *Computational Methods in Optimization.* New York: Academic Press.

[PrB84] PROVAN, J. S., and BALL, M. O. 1984. Computing Network Reliability in Time Polynomial in the Number of Cuts, *Oper. Res.*, 32:516–526.

[Pro83] PROAKIS, J. G. 1983. *Digital Communications.* New York: McGraw-Hill.

[Qur85] QURESHI, S. 1985. Adaptive Equalization, *Proc. IEEE*, pp. 1349–1387.

[RaH76] RAUBOLD, E., and HAENLE, J. 1976. A Method of Deadlock-Free Resource Allocation and Flow Control in Packet Networks, *Proc. Internat. Conf. Comput. Comm.*, Toronto.

[RaW83] RAMAMOORTHY, C. V., and WAH, B. W. 1983. The Isomorphism of Simple File Allocation, *IEEE Trans. Comput.*, C-32:221–231.

[ReC90] REGNIER, J., and CAMERON, W. H. 1990. State-Dependent Dynamic Traffic Management for Telephone Networks, *IEEE Communications Magazine*, 28:42–53.

[RFS91] ROBINSON, J., FRIEDMAN, D., and STEENSTRUP, M. 1991. Congestion Control in BBN Packet-Switched Networks, *Computer Communication Reviews.*

[Rin76] RINDE, J. 1976. TYMNET I: An Alternative to Packet Switching, *Proc. 3rd Internat. Conf. Comput. Comm.*

[Riv85] RIVEST, R. L. 1985. *Network Control by Bayessian Broadcast* (Report MIT/LCS/TM-285). Cambridge, MA: MIT, Laboratory for Computer Science.

[Rob72] ROBERTS, L. G. 1972. *Aloha Packet System with and without Slots and Capture* (ASS Note 8). Stanford, CA: Stanford Research Institute, Advanced Research Projects Agency, Network Information Center.

[Roc70] ROCKAFELLAR, R. T. 1970. *Convex Analysis.* Princeton, NJ: Princeton University Press.

[Roc84] ROCKAFELLAR, R. T. 1984. *Network Flows and Monotropic Programming.* New York: Wiley.

[Ros80] ROSS, S. M. 1980. *Introduction to Probability Models.* New York: Academic Press.

[Ros81] ROSEN, E. C. 1981. Vulnerabilities of Network Control Protocols: An Example. *Comput. Comm. Rev.*

[Ros83] ROSS, S. M. 1983. *Stochastic Processes.* New York: Wiley.

[Ros86a] ROSBERG, Z. 1986. Deterministic Routing to Buffered Channels, *IEEE Trans. Comm.* COM-34:504–507.

[Ros86b] ROSS, F. E. 1986. FDDI—A Tutorial, *IEEE Commun. Mag.*, 24:10–17.

[RoT90] ROSS, K., and TSANG, D. 1990. Teletraffic Engineering for Product-Form Circuit-Switched Networks, *Adv. Appl. Prob.*, 22:657–675.

[Ryb80] RYBCZYNSKI, A. 1980. X.25 Interface and End-to-End Virtual Circuit Service Characteristics, *IEEE Trans. Comm.*, COM-28:500–509. (Also in [Gre82].)

[SaH87] SASAKI, G., and HAJEK, B. 1986. Optimal Dynamic Routing in Single Commodity Networks by Iterative Methods, *IEEE Trans. Comm.*, COM-35:1199–1206.

[San80] SANT, D. 1980. Throughput of Unslotted Aloha Channels with Arbitrary Packet Interarrival Time Distribution, *IEEE Trans. Comm.*, COM-28:1422–1425.

[SaO82] SARACHIK, P. E., and OZGUNER, U. 1982. On Decentralized Dynamic Routing for Congested Traffic Networks, *IEEE Trans. Automat. Control*, AC-27:1233–1238.

[Sar82] SARACHIK, P. E. 1982. An Effective Local Dynamic Strategy to Clear Congested Multidestination Networks, *IEEE Trans. Automat. Control*, AC-27:510–513.

[Sas91] SASAKI, G. 1991. Input Buffer Requirements for Round Robin Polling Systems with Nonzero Switchover Times, *Proceedings of the 29th Allerton Conference on Communication, Control, and Computing*, Monticello, IL.

[Sch87] SCHWARTZ, M. 1987. *Telecommunication Networks Protocols, Modeling, and Analysis.* Reading: Addison Wesley.

[ScS80] SCHWARTZ, M., and STERN, T. E. 1980. Routing Techniques Used in Computer Communication Networks, *IEEE Trans. Comm.*, COM-28:539–552.

[Seg81] SEGALL, A. 1981. Advances in Verifiable Fail-Safe Routing Procedures, *IEEE Trans. Comm.*, COM-29:491–497.

[Seg83] SEGALL, A. 1983. Distributed Network Protocols, *IEEE Trans. Inform. Theory*, IT-29:23–34.

[Sha48] SHANNON, C. E. 1948. A Mathematical Theory of Communication, *Bell Syst. Tech J.*, 27:379–423 (Part 1), 623–656 (Part 2). (Reprinted in book form by the University of Illinois Press, Urbana, IL, 1949.)

[Sie86] SIEBERT, W. M. 1986. *Circuits, Signals, and Systems.* Cambridge, MA: MIT Press; and New York: McGraw-Hill.

[SiS81] SIDI, M., and SEGALL, A. 1981. A Busy-Tone-Multiple-Access-Type Scheme for Packet-Radio Networks, in G. Payolk (Ed.), *Performance of Data Communication Systems and Time Applications.* New York: North-Holland, pp. 1–10.

[SoH86] SOLOWAY, S. R., and HUMBLET, P. A. 1986. *On Distributed Network Protocols for Changing Topologies.* Codex Corp.

[SpG89] SPINELLI, J. M., and GALLAGER, R. G. 1989. Event Driven Topology Broadcast without Sequence Numbers, *IEEE Trans. on Comm.*, 37: 468–474.

[Spi85] SPINELLI, J. 1985. *Broadcasting Topology and Routing Information in Computer Networks.* (Report LIDS-TH-1470). Cambridge, MA.: MIT.

[Spi89] SPINELLI, J. M. 1989. Reliable Data Communication in Faulty Computer Networks, *Ph.D. Thesis, MIT Dept. EECS*, June 1989. Also *MIT LIDS Report LIDS-TH-1882.*

[SpM81] SPROULE, D. E., and MELLOR, F. 1981. Routing, Flow and Congestion Control and the Datapac Network, *IEEE Trans. Comm.*, COM-29:386–391.

[Sta85] STALLINGS, W. 1985. *Data Computer Communications*. New York: Macmillan.

[StA85] STUCK, B. W., and ARTHURS, E. 1985. *A Computer Communications Network Performance Analysis Primer*. Englewood Cliffs, NJ: Prentice Hall.

[Sti72] STIDHAM, S., Jr. 1972. $L = \lambda W$: A Discounted Analogue and a New Proof, *Oper. Res.*, 20:1115–1125.

[Sti74] STIDHAM, S., Jr. 1974. A Last Word on $L = \lambda W$, *Oper. Res.*, 22:417–421.

[StK85] STASSINOPOULOS, G. I., and KONSTANTOPOULOS, P. 1985. Optimal Congestion Control in Single Destination Networks, *IEEE Trans. Comm.*, COM-33:792–800.

[Syz90] SYZMANSKI, T. 1990. An Analysis of Hot-Potato Routing in a Fiber Optic Packet Switched Hypercube, *IEEE INFOCOM '90 Proceedings*, 1:918–925.

[Taj77] TAJIBNAPIS, W. D. 1977. A Correctness Proof of a Topology Information Maintenance Protocol for a Distributed Computer Network, *Commun. ACM*, 20:477–485.

[TaK85] TAKAGI, H., and KLEINROCK, L. 1985. Throughput Delay Characteristics of Some Slotted-Aloha Packet Radio Networks, *IEEE Trans. Comm.*, COM-33:1200–1207.

[Tak86] TAKAGI, H., 1986. *Analysis of Polling Systems*. Cambridge, MA: MIT Press.

[Tan89] TANENBAUM, A. S. 1989. *Computer Networks*. (2nd ed.). Englewood Cliffs, NJ: Prentice-Hall.

[TBF83] TOBAGI, F., BORGONOVO, F., and FRATTA, L. 1983. Express-Net: A High Performance Integrated-Services Local Area Network, *IEEE J. Select Areas Comm.*, SAC-1.

[ThC86] THAKER, G. H., and CAIN, J. B. 1986. Interactions between Routing and Flow Control Algorithms, *IEEE Trans. Comm.*, COM-34:269–277.

[Tob74] TOBAGI, F. A. 1974. *Random Access Techniques for Data Transmission over Packet Switched Radio Networks*, Ph.D. thesis, UCLA, Computer Science Dept., Los Angeles.

[ToK75] TOBAGI, F. A., and KLEINROCK, L. 1975. Packet Switching in Radio Channels: Part II: The Hidden Terminal Problem in CSMA and Busy-Tone Solution, *IEEE Trans. Comm.*, COM-23:1417–1433.

[ToW79] TOWSLEY, D., and WOLF, J. K. 1979. On the Statistical Analysis of Queue Lengths and Waiting Times for Statistical Multiplexors with ARQ Retransmission Schemes, *IEEE Trans. Comm.*, COM-27:693–702.

[Tsa89] TSAI, W. K. 1989. Convergence of Gradient Projection Routing Methods in an Asynchronous Stochastic Quasi-Static Virtual Circuit Network, *IEEE Trans. Aut. Control*, 34:20–33.

[TsB86] TSITSIKLIS, J. N., and BERTSEKAS, D. P. 1986. Distributed Asynchronous Optimal Routing in Data Networks, *IEEE Trans. Automat. Control* AC-31:325–331.

[Tsi89] TSITSIKLIS, J. N. 1989. *Markov Chains with Rare Transitions and Simulated Annealing* Math. of Operations Research, Vol. 14, pp. 70–90.

[Tsi87] TSITSIKLIS, J. N. 1987. *Analysis of a Multiaccess Control Scheme, IEEE Trans. Automat. Control*, AC-12:1017–1020.

[TsM78] TSYBAKOV, B. S., and MIKHAILOV, V. A. 1978. Free Synchronous Packet Access in a Broadcast Channel with Feedback, *Problemy Peredachi Inform.* (USSR), 14(4):32–59.

[TsM80] TSYBAKOV, B. S., and MIKHAILOV, V. A. 1980. Random Multiple Access of Packets: Part and Try Algorithm, *Problemy Peredachi Inform.* (USSR), 16:65–79.

[TsR90] TSANG, D., and ROSS, K. 1990. Algorithms to Determine Exact Blocking Probabilities for Multirate Tree Networks, *IEEE Trans. on Comm.*, 38:1266–1271.

[TsS90] TSITSIKLIS, J. N., and STAMOULIS, G. D. 1990. On the Average Communication Complexity of Asynchronous Distributed Algorithms, *MIT LIDS Report LIDS-P-1986.*

[Tsy85] TSYBAKOV, B. S. 1985. Survey of USSR Contributions to Random Multiple-Access Communications, *IEEE Trans. Inform. Theory*, IT-31:143–165.

[Tur86] TURNER, J. 1986. New Directions in Communications (or Which Way to the Information Age), *IEEE Communications Magazine*, 24:8–15.

[Twe82] TWEEDIE, R. L. 1982. Operator-Geometric Stationary Distributions for Markov Chains with Application to Queueing Models, *Adv. Appl. Probab.*, 14:368–391.

[Tym81] TYMES, L. 1981. Routing and Flow Control in TYMNET, *IEEE Trans. Comm.*, COM-29:392–398.

[Var90] VARVARIGOS, E. A. 1990. Optimal Communication Algorithms for Multiprocessor Computers, M.S. Thesis, Dept. of Electrical Engineering and Computer Science, M.I.T., also Center for Intelligent Control Systems Report, CICS-TH-192.

[Vas79] VASTOLA, K. S. 1979. *A Numerical Study of Two Measures of Delay for Network Routing*, M.S. thesis, University of Illinois, Dept. of Electrical Engineering, Urbana, IL.

[VvP83] VVEDENSKAYA, N. D., and PINSKER, M. S. 1983. Non-optimality of the Part-and-Try Algorithm, in *Abstracts of the International Workshop of Convolutional Codes, Multiuser Communication*, Sochi, USSR, pp. 141–148.

[Wal83] WALRAND, J. 1983. Probabilistic Look at Networks of Quasi-reversible Queues, *IEEE Trans. Inform. Theory*, IT-29:825–831.

[Wal88] WALRAND, J. 1988. *An Introduction to Queueing Networks.* Prentice Hall, Englewood Cliffs, N.J.

[WaO90] WATANABE, Y., and ODA, T. 1990. Dynamic Routing Schemes for International Networks, *IEEE Communications Magazine*, 28:70–75.

[Wec80] WECKER, S. 1980. The Digital Network Architecture, *IEEE Trans. Comm.*, COM-28:510–526.

[Wel82] WELDON, E. J. 1982. An Improved Selective Repeat ARQ Strategy, *IEEE Trans. Comm.*, COM-30:480–486.

[Whi83a] WHITT, W. 1983. The Queueing Network Analyzer, *The Bell System Technical Journal*, 62:2779–2815.

[Whi83b] WHITT, W. 1983. Performance of the Queueing Network Analyzer, *The Bell System Technical Journal*, 62:2817–2843.

[WiE80] WIESELTHIER, J. E., and EPHREMIDES, A. 1980. A New Class of Protocols for Multiple Access in Satellite Networks, *IEEE Trans. Automat. Control*, AC-25:865–880.

[Wol82a] WOLFF, R. W. 1982. Poisson Arrivals See Time Averages, *Oper. Res.*, 30:223–231.

[Wol82b] WOLFF, R. W. 1982. Tandem Queues with Dependent Service Times in Light Traffic, *Oper. Res.*, 82:619–635.

[Wol89] WOLFF, R. W. 1989. *Stochàstic Modelling and the Theory of Queues.* Prentice Hall, Englewood Cliffs, N.J.

[Yum81] YUM, T. P. 1981. The Design and Analysis of a Semidynamic Deterministic Routing Rule, *IEEE Trans. Comm.*, COM-29:498–504.

[Zan69] ZANGWILL, W. 1969. *Nonlinear Programming: A Unified Approach.* Englewood Cliffs, N.J.: Prentice Hall.

[Zha89] ZHANG, L. 1989. A New Architecture for Packet Switching Network Protocols, Ph.D. Thesis, *Dept. of Electrical Engineering and Computer Science, MIT.* Cambridge, MA.

[ZVK89] ZINKY, J., VICHNIAC, G., and KHANNA, A. 1989. Performance of the Revised Routing Metric in the ARPANET and MILNET, *Proc. of MILCOM '89.*

Index

A

ABM. *See* Asynchronous balanced mode
ADCPP, 72, 97–103
AD. *See* Amplitude modulation
ANSI. *See* American National Standards Institute
AM. *See* Amplitude modulation
ARM. *See* Asynchronous response mode
ARPANET, 2, 4, 19–20
  ARQ, 84–86
  flow control, 515
  framing, 87
  routing, 374–76, 404–6, 412
ARQ. *See* Automatic repeat request
ASCII code, 21, 31, 58, 86
ATM. *See* Asynchronous transfer mode
Abort capability for frames, 89
Access rights, 31
Acks, 66, 500
  end-to-end, 115–16
Adaptive equalizers, 45, 49, 140
Addressing, 40, 111–14
  in TCP, 124–25
Age field, 422, 425
Aggregation of queues, 254
Airline reservation systems, 7, 9, 10
Allocate message, 500
Aloha, 275–89, 352
  slotted, 277–87
  stabilized, 282–86
  unslotted, 287–89
American National Standards Institute, 97
Amplitude modulation, 48
Application layer, 31–32
Arc, 387, 394
Arrival rate, 12, 152
Arrival theorem, 239, 256
Asynchronous balanced mode, 98–103
Asynchronous character pipes, 38
Asynchronous response mode, 98
Asynchronous transfer mode, 32, 40, 55, 97, 128–39, 141
  adaptation layer, 135–39
  call size, 132
  congestion, 138–39
  CRC on header, 133-34
  flow control, 138–39
  use of SONET, 134–35
  virtual channel and path ID, 133
Attenuation, 56
Automatic repeat request, 39, 64–80, 272
  ARPANET, 84–86
  delay analysis, 190
  go back n, 72–81, 190
  packet radio, 347–48
  ring networks, 323
  selective repeat, 81–84
  stop and wait, 66–71

B

BRAM, 333
BSC. *See* Binary synchronous communication
Backlog, 276, 279
Backlog estimation, 283, 286
Backpressure, 507
Bandwidth, 50–51
Bellman's equation, 399
Bellman-Ford algorithm, 396–400, 404–410, 424, 481, 492
Binary exponential backoff, 286–87, 352
Binary synchronous communication, 87
Birth-death processes, 184, 215, 261
Bisynch, 87
Bit pipes, 20–23, 38, 65, 86
  intermittent synchronous, 38
  synchronous, 38
Bit stuffing, 88–90, 145, 320
Bits, 10
Black box, 17
Blocking probability, 180, 185
Bottleneck link, 526
Branch exchange heuristic, 444, 448
Bridge learning, 384
Bridged local area networks, 382–87
Bridges, 4, 29, 380
Broadband ISDN, 5, 11–12, 32, 55, 128–32
Broadcasting, 369, 375–76, 433, 485
Buffer management, 496–98
Buffer overflow, 496–99
Burke's theorem, 218, 255
Burst detection, 60–61
Bursty sessions, 14–15
Bus systems, 271, 274, 331–33
  undirectional, 334–41
Busy tones, 350–51

C

CCITT, 97
CRC. *See* cyclic redundancy checks
CRC-16 polynomial, 64
CRC-32 IEEE polynomial, 64
CRC-CCITT polynomial, 64, 98
CRP. *See* Collision resolution period
CSMA. *See* Carrier sense multiple access
CSMA/CD. *See* Carrier sense multiple access/collision detection
Cable TV, 56, 341–42
Call blocking, 494
Call-request packet X.25, 119
Capacity assignment, 439–44
Capacity, 51, 150
Capture effects, slotted Aloha, 355
Carrier offset, 53
Carrier sense multiple access, 272, 304–12, 352
  collision avoidance, 333
  collision detection, slotted, 317–18
  collision detection, unslotted, 318–20
  FCFS splitting, 310–12
  nonpersistent, 305
  packet radio, 350–51
  persistent, 305
  P-Persistent, 305, 307
  pseudo-Bayesian stabilization, 307–9
  slotted Aloha, 305–9
  unslotted Aloha, 309–10
  variable delay, 358
Cellular radio system, 247
Character based framing, 86–88
Choke packet, 510
Circuit switching, 14–17, 180, 185, 379
Coaxial cable, 7, 56
Code conversion, 31
Codex networks, 141
  data link control, 518
  flow control, 518
  routing, 378, 417, 476–77
  session identification, 113—15
Coding. *See* Data compression, Encryption, Error detection, or Error correction
Collision resolution, 276–77
  packet radio, 347–49
Collision resolution period, 291, 295–300
Collision-free set, 345
Communication channels, 20, 38, 40–57
  analog, 40
  bandpass, 47–52
  digital, 40, 53–54
  voice-grade, 49–50, 52
Concentrator location problem, 448
Concentrators, 1
Congestion control, 27, 116–17
  in ATM, 138–39 *See also* Flow control
Connected graph, 387, 394
Connectionless service, 26
  in ATM, 137–39
Connectivity, 445
Constrained MST problem, 447
Convergence sublayer in ATM, 136–39
Convex set or function, 453, 486
Convolution, 43
Convolutional codes, 61
Correctness, stop-and-wait, 69–70
Cut-through routing, 373
Cycle, 387
Cyclic redundancy check, 61–64, 140
  on ATM header, 133–34
  end-to-end, 116

D

db. *See* Decibels
DC component, 47, 48
DCE. *See* Data communication equipment
DECNET, 2, 19–20, 91, 93, 404
DLC. *See* Data link control
DLE (data link escape), 87
DNA. *See* Digital network architecture
DQDB. *See* Distributed queue dual bus
DTE. *See* Data terminal equipment
Data bases, remote access, 7–8
Data communication equipment, 21, 118
Data compression, 10, 31
Data link control, 20, 23–24, 37–40, 57–110, 271
  correctness, 76–79, 140
  standards, 97–103, 141
Data terminal equipment, 21, 118
Datagram:
  networks, 115
  routing, 16
  service, 26
Datagram routing, 363
Deadlock, 497–99
Decibels, 51
Decomposition of gueues, 254
Deflection routing, 372
Delay, 13
  CSMA, 308–9
  CSMA/CD, 318, 360
  FCFS splitting algorithm, 300–301

Delay *cont.*
Fiber distributed data interface, 329–30
processing, 150
propagation, 150
queueing, 150
satellite systems, 314–15
slotted Aloha, 284–86
token buses and polling systems, 331–32
token ring, 324–25
transmission, 150
Demand sharing, 274
Descent direction, 456
Designated port, 383
Detailed balance equations, 182, 214, 216, 218, 261, 263
Digital network architecture, 404
Digraph, 394
Dijkstra's algorithm, 401–3, 481
Directed arc, 394
Directed cycle, 394
Directed graph, 394
Directed walk, 394
Directory assistance, 30
Disconnect command, 100
Disconnect protocols, 103–10
Distributed algorithms, 28, 32–34, 39, 139
Distributed queue dual bus, 335–39, 353
Distributed shortest path construction, 404–10
Distributed spanning tree construction, 392
Drift, 264
Drift analysis:
CSMA, 306
FCFS splitting algorithm, 300
slotted Aloha, 280
Dynamic programming, 395
Dynamic routing, 16–17, 436–38, 491
Dynamic window adjustment, 510

E

EXT (end of text), 87
Electronic mail, 8, 9
Embedded chain, 262
Encryption, 31
End-to-end CRC, 116
End-to-end acks, 115–16, 119
End-to-end windows, 506–8, 515–17
Entropy, 91, 140
Ergodic system, 156
Erlang B formula, 179, 167
Erlang C formula, 175, 267
Error correction, 52, 61, 140
Error detection, 39, 53, 57–64, 140
bursts, 60–61
random strings, 60–61
Error recovery, 23, 32, 115–16
in TCP, 125–27
transport vs. network layer, 117–18
Essau-Williams algorithm, 448
Ethernet, 4, 48, 52, 274, 286, 317–20
degradation with size and speed, 334
Exponential distribution, 164, 266
Expressnet, 339–41
External site, 19

F

FDDI. *See* Fiber distributed data interface
FDM. *See* Frequency division multiplexing
Facsimile, 8
Feasible direction, 456
Feedback, multiaccess:
delayed, 303–4
immediate, 276
Feedback shift register, 62
Fiber distributed data interface, 326–30, 334, 339, 353
delay, 329–30
throughput, 330
File transfer, 13
Filtering, 41–53
First derivative length, 452
First-come first-serve splitting, 293–304
CSMA, 310–12
Flags, 88–90, 320
Flooding, 369, 419–25, 432, 490
Flooding with sequence numbers, 420
Flow control, 25–29, 30, 116–17, 158
combined with optimal routing, 519–24
fairness, 498–99, 505, 507, 524–29
in ARPANET, 515
in ATM, 138–39
in Codex network, 518
in PARIS network, 518
in SNA, 517
in TCP, 127–28
in TYMNET, 517
in X.25, 519
input rate adjustment, 519–29
max-min, 524–29, 534
means, 494–95
objectives, 496–99
optimal, 519–24
satellite links, 506–8, 517
transport layer, 509
window, 500-510
Flow deviation method, 458–64
Flow models, 433–37
Floyd-Warshall algorithm, 403–4
Forwarding database, 383
Fourier transforms, 44, 45
Fragment, 389
Fragmentation, internet protocol, 122
Frames, 23, 52, 65
internet protocol, 122
maximum fixed length, 97
maximum length, 140–41
maximum variable length, 93–97
Framing, 39, 86–97
bit-oriented, 88–90
characted based, 86–88
length fields, 90–92
overhead, 87, 89–90, 90–92
with errors, 92–93
Frank-Wolfe method, 458–64, 471, 488
Free-for-all multiaccess, 272
Frequency division multiplexing, 52, 151, 170, 177, 194
in satellite channels, 273
Frequency response, 43–46
Front end, 1

G

G/G/1 queue. *See* Queueing System
Gateways, 4, 28–29, 120, 380
Generator polynomials, 62–64, 142
Global balance equations, 168, 260, 263
Go back n ARQ, 72–81, 113, 143–44, 501
correctness proof, 76–79
efficiency, 80–81, 144
ideal, 82
rules, 74–76
rules with mod m, 80
Gradient project method, 459, 467–77, 486–87, 489
Graph, 387
Graphics, 8, 13
Guaranteed data rate, 494

H

HDLC, 39, 72, 97–103
faulty initialization, 106
Header:
frame, 23, 39, 65
packet, 25
Hessian matrix, 465
Hierarchy of modules, 17
Homenets, 341–42
Horizontal and vertical parity checks, 58–59
Hot potato routing, 372
Hybrids, 56

I

IEEE 802 standards, 48, 320, 323–24, 332–33, 335
IMP. *See* Interface message processor
IP. *See* Internet protocol
ISDN. *See* Integrated services digital network
ISO. *See* International Standards Organization
Idle fill, 38, 90, 321
Image transmission systems, 9
Implicit tokens, 333
Impulse response, 42
Incident arc, 387
Infinite node assumption, 276
Initialization protocols, 147
ARQ, 103–10
balanced, 107–9
fault in HDLC, 106
with link failures, 103–9
master-slave, 104–7
with node failures, 109–10
in TCP, 126
use of stop-and-wait, 104
Integrated services digital network, 5, 11–12, 54–56
basic service, 55
primary service, 55
Interactive sessions, 13, 15
Interface message processor, 2
Internal signaling network, 55
International Consultative Committee on Telegraphy and Telephony. *See* CCITT
International Standards Organization, 19, 97
terminology, 30
Internet protocol, 40, 120–23, 141
addressing, 121–22
datagrams, 120
fragmentation, 122
time to live, 123
Internet sublayer, 28–29
Internetworking, 28–29
Intersymbol interference, 42, 45–47, 49
Isarithmic method, 508

J

Jackson's theorem:
closed networks, 234
heuristic explanation, 227
with limit on the number of customers, 256
multiple classes of customers, 231
open networks, 223
state-dependent rates, 230

K

Kleitman's algorithm, 445–47
Kruskal algorithm, 390, 448

L

LAN. *See* Local area network
LAPB, 72, 97–103, 118
Last-come first-serve splitting algorithm, 302–3
Layering, 17–32, 35
Leaky bucket, 245, 511–15, 535
Linear codes, 59
Linear systems, 41–46, 140
Little's theorem, 152–62, 250–51
  slotted Aloha, 280
Liveness:
  go back n, 77–78
  stop-and-wait, 69–70
Load sharing, 8
Local access network, 438
Local access network design, 448–51
Local area networks, 4, 7, 29, 56, 271–72, 317–42, 352–53
Local loop, telephone network, 54, 56

M

M/D/1 queue. *See* Queueing system
M/G/1 queue. *See* Queueing system
M/G/∞ queue. *See* Queueing system
M/M/1 approximation. *See* Networks of queues
M/M/1 queue. *See* Queueing system
M/M/1/m queue. *See* Queueing system
M/M/∞ queue. *See* Queueing system
M/M/m queue. *See* Queueing system
M/M/m/m queue. *See* Queueing system
MAC. *See* Medium access control
MAN. *See* Metropolitan area network
MSAP, 333
Manchester coding, 48
Markov chain:
  aperiodic, 260
  continuous-time, 262–63
  discrete-time, 259–62
  FCFS splitting algorithm, 297
  instability, 265
  irreducible, 260
  M/M/1 system, 166
  M/M/∞ system, 177
  M/M/m system, 174
  M/M/m/m system, 179
  multidimensional, 180-86
  queueing networks, 223
  reversible, 182
  slotted Aloha, 279, 353
  stability, 264
  time reversible, 215
Max-min fairness, 328
Max-min flow control, 524–29, 534
Mean residual service time, 187
Mean value analysis, 238–40
Media access control, 24, 27, 29, 271
Memoryless property, 165
Message switching, 16
Messages, 10
  arrival rate, 12
  delay, 13
  length, 12–13
  ordering, 13
Metering, 214, 436
Metropolitan area networks, 271–72, 326–30, 333–42
Micro-processors, 5
Microwave channels, 56
Min-hop path, 370
Minimum distance, 60
Minimum weight spanning tree, 389, 484
Modems, 21, 38, 40, 48–52
Modularity, 17
Modulation, 140
  amplitude, 48–49
  phase-shift keying, 50
  quadrature amplitude, 49–52
Module, 17
Modulo arithmetic, 58
Multiaccess communication, 24, 195, 271–362
  canonic reservation system, 312
Multidrop telephone lines, 271, 274, 331, 342
Multiplexers, 1
Multiplexing, in TCP, 124–25
Multicap bus systems, 271, 274

N

NRM. *See* Normal response mode
NRZ code, 42, 48
Nak, 66
Network layer, 25–29
  point to point protocols, 110–20
Networks of queues:
  acyclic, 221
  closed, 233
  Jackson's theorem, 212, 221, 434
  Kleinrock independence approximation, 212, 221, 434, 440
  M/M/1 approximation, 213, 221, 434, 440, 453, 476
  multiple classes of customers, 230
  state dependent service rates, 229
  tandem queues, 209, 220
  with multiple classes of customers, 222
Newton's method, 467
Node of a graph, 387
Node-by-node windows, 501–6, 517
Noise, 45, 51
Nonlinear distortion, 53
Normal response mode, 98–101
Nyquist criterion, 47

O

OD pair. *See* Origin-destination pair
OSI. *See* Open systems interconnection
On-off flow, 11
Open systems interconnection, 19–20, 35
Optical fibers, 6–7, 15, 56, 326, 333
Optimal routing:
  algorithms, 455–75
  dynamic, 436–38, 491
  formulation, 433–36
Origin-destination pair, 434

P

PARIS network, 392, 518
PERT networks, 395
PSK. *See* Phase-shift keying
Packet blocking, 495
Packet discarding, 495
Packet radio nets, 57, 273, 275, 286, 344–51
  collision resolution, 347–49
  TDM, 346–47
  transmission radii, 349–50, 353
Packet scheduling, 495
Packet switching, 131
  why used in ATM, 16
Packets, 10, 39
  ordering, 13, 86
Parity check codes, 59–64, 140
Parity checks, 58
Partial balance equations, 262
Path in a graph, 387
Peer process (modules), 18, 23, 25, 28, 39
Periodic routing updates, 422
Permit, 117, 500
Permit (window) in TCP, 127
Phase jitter, 53
Phase-shift keying, 50
Physical layer, 20–23, 37–38, 40–57
Piggybacked requests, 67
Pipelining, 94–96
Point to point protocols, 37
Poisson distributing, 164, 266
Poisson process, 11, 164–65, 172–73, 275
  combined, 244
  merged, 165
  properties, 243
  split, 165, 245
Poll final bit, 99
Pollaczek-Khinchin formula, 186
Polling, 274, 331–32, 342–44
  generalized, 342–44
  hub, 331
  queueing analysis, 195–203
Polynomial division, 62
Port, 382
  in TCP, 124
Positive definite matrix, 465
Positive semidefinite matrix, 465
Presentation layer, 31
Prim-Dijkstra algorithm, 390, 448, 484
Primitive polynomials, 64
Priorities, FDDI, 326–27
Priority queueing system. *See* Queueing system
Product form, 182
Product form solution, 220, 223–25
Promiscuous mode, 382
Protocols. *See* Automatic reqeat request, Distributed algorithms, HDLC, X.25, X.21
Pseudo-Bayesian algorithm, 283–86, 352
  CSMA slotted Aloha, 307–9
  CSMA unslotted Aloha, 359
Pure Aloha, 287–89

Q

QAM. *See* Quadrature amplitude modulation
Quasistatic assumption, 177
Queueing system:
  closed, 158, 233, 256–58
  discrete-time M/M/1, 246
  G/G/1, 206–9, 253, 269
  last-come first-serve, 232
  last-come first-serve M/G/1 system, 253
  M/G/1 system, 267
    with arbitrary order of service, 248
    with batch arrivals, 251
    with busy period overhead, 251
    with delay for new busy period, 252
    with vacations, 192–95, 252, 268
  M/G/m/m, 179
  M/G/∞, 252
  M/M/1, 162–73, 175, 216, 266
    shared service, 247
    with multiple classes of customers. 258
    with state-dependent rates, 246
  M/M/1/m, 247, 255

M/M/2 with heterogeneous servers, 247
M/M/∞, 177–78, 216, 247
M/M/m, 174–77, 216, 266
M/M/m/m, 175, 179, 216, 267
networks. *See* Networks of queues
nonpreemptive priority, 203–5, 249–50, 269
with phase-type distributions, 232
polling, 160, 195, 268
preemptive resume priority, 205–6, 249, 269
priority with multiple servers, 249
processor sharing, 232
reservation, 195–204, 268
time reversible, 214

R

RN. *See* Request number
RS-232-c interface, 21–22
Radio channels, 56, 271 *See also* Packet radio nets)
Randomization, 214, 222, 436
Rate control schemes, 510–15
Real time computation, 8
Real time control, 13
Reassembly, 29
Receive-not-ready:
supervisory frame, 99–100
X.25 packet, 119
Receive-ready:
supervisory frame, 99–100
X.25 packet, 119
Register insertion rings, 330–31
Regular chain, 262
Reject supervisory frame, 99–100
Reject X.25 packet, 119
Reliability, 9, 13, 29, 443–48
Repeaters, 53–54
Request number, 67
Reservation system:
canonic multiaccess system, 312
exhaustive, 195
gated, 195
limited service, 201, 249
multiaccess, 272, 312–44
multiuser, 198
partially gated, 195
satellites 313–17
single-user 196
Reset command, 103
Reversibility, 214–21
Ring networks, 320–31
Ringing, 48
Round-robin scheduling, 495
Router, 380
Routing, 25–29, 363–491
adaptive, 368, 410–17, 455
asynchronous, 375, 404–10, 425
broadcast, 487
centralized, 367, 418
distributed, 367
hierarchical, 379–82
in the ARPANET, 374
in Codex network, 476–77
in circuit switching networks, 379
in SNA, 378
in the TYMNET, 376
interaction with flow control, 365
interconnected network, 379–87
main issues, 365–68
nonhierarchical, 380–82
optimal, 372, 433, 451–75
oscillations, 371, 374, 410–17
shortest path, 374–76, 387–417
shortest path first, 375, 382
static, 368
Routing *cont.*
tables, 365, 375–76, 377, 382
with frequently changing topology, 491

S

SDLC, 72, 97–103
SN. *See* Sequence number
SNA. *See* System network architecture
SONET, 54, 141
use in ATM, 134–35
SPTA. *See* Shortest path topology algorithm
STX (start of text), 87
SYN (synchronous idle), 86
Sampling theorem, 46–47, 51
Satellite channels, 56–57, 271, 273–74, 313–17, 352–53
Saturated cut method, 81–84, 140–41, 444
Scheduling for multiaccess, 272 (*See also* Reservations)
Segmentation and reassembly, in ATM, 136–39
Selective repeat ARQ, 144
ideal, 82
Selective-reject supervisory frame, 99–100
Sequence number, 66
Service rate, 152
Session holding time, 12
Session identification, 111–14
Session initiation, 31
Session layer, 30–31
Sessions, 11–14
Shannon capacity theorem, 50–51, 53, 140
Shortest path problem, 394
Shortest path routing, 370
Shortest path spanning tree, 400
Shortest path topology algorithm, 425–33, 483
Signal constellations, 50–51
Signal power, 51
Simulated annealing, 443
Sliding window, 72 (*See also* Go back n ARQ)
Slotted Aloha:
capture, 355
delay, 354
stability, 353
Slotted frequency-division multiplexing, 194
Slotted multiaccess sytems, 275–88
Slotted ring, 330
Socket in TCP, 124
Solid state technology, 5
Source coding theorem, 91–92, 140
Source routing, 385–87
Spanning tree, 369, 388
Spanning tree routing, 383–85
Spanning tree topoligy design, 447–48
Splitting algorithms, 289–304
first-come first-serve, 293–304, 352
LCFS, 302–3
round robin, 304
with delayed feedback, 303–4
zero or positive feedback, 342–44
Stability:
CSMA, 307–12
datagram networks, 410–14
FCFS splitting algorithm, 300
shortest path routing algorithm, 410–417
slotted Aloha, 281–86, 352
unslotted Aloha, 288, 359
virtual circuit networks, 414–17
Stack algorithms, 291 (*See also* Tree algorithms)
unblocked stack, 293
Standardization, 19–20, 32
Star configuration for rings, 324
Stationary distribution, 260, 263
Statistical multiplexing, 113, 150, 170, 176, 214
Steepest descent method, 465
Stop-and-wait, 142–43
ARQ, 66–69
correctness proof, 69–70
in initialization protocols, 104
Store-and-forward switching, 14–17
Strongly connected graph, 394
Subgraph, 388
Sublinear convergence rate, 461
Subnet, 2–3, 10
Subnet design, 439–48
Supercomputers, 8
Supervisory frames, 99–100
System network architecture (SNA), 19–20
explicit route control, 378
flow control, 501, 517
path control layer, 378
routing, 378
SDLC, 72, 97–103
session level pacing, 517
transmission control layer, 378
transmission group control, 378
virtual route control, 378, 517
virtual route pacing scheme, 501, 517

T

TCP. *See* Transport control protocol
TCP/IP. *See* Internet protocol, Transport control protocol
TDM. *See* Time-division multiplexing
TP4 (ISO transport protocol), 128
T1 carrier, 53–54
T3 carrier, 54
TYMNET, 2, 4, 19–20, 141
flow control, 517
routing, 376–78
session identification, 112–13
Technology:
communication, 6–7
computer, 5–6
Three army problem, 33–34, 71
Throughput:
CSMA FCFS splitting, 311
CSMA slotted Aloha, 306
CSMA unslotted Aloha, 310
CSMA/CD:
slotted, 318
unslotted, 319
FCFS splitting algorithm, 300–301
slotted Aloha, 282–83
token buses and polling systems, 331–32
unslotted Aloha, 288
Time invariance, 41
Time sharing system, 160, 236
Time to live, internet protocol, 123
Time window flow control, 511
Time-division multiplexing, 52, 151, 170, 177, 194, 214, 273, 277
for packet radio, 346–47
Time-outs, 67–70, 73
Token buses, 331–33
degradation with size and speed, 334
implicit tokens, 333
Token rings, 4, 320–30
Topological design, 437–51, 464
Topology, 418

Trailer frame, 23, 39, 65
Transition rate, 262
Transmission delay. *See* Delay
Transparent mode, 87
Transport control protocol, 40, 124–28, 141
  addressing, 124–25
  error recovery, 117–18, 125–27
  flow control, 127–28
  initialization, 126
  multilexing, 124–25
Transport layer, 29–30, 123–28
  standards, 123–24
Transport layer flow control, 509
Tree, 388
Tree algorithms, 290–93, 352
Truncation of Markov chains, 182, 254
Twisted pair, 7, 56

U

Unary-binary encoding, 91, 114
Undetectable errors, 60, 64, 114–15
Unnumbered acks, 100
Unnumbered frames, 100
Unslotted Aloha, 287–89
  precise feedback, 356
  throughput, 357
Unstable equilibrium for Aloha, 281
Utilization factor, 157, 169

V

VLSI. *See* Very large scale integration
Vacation Queueing system. *See* Queueing system
Very large scale integration, 5
Video conferencing, 12
Virtual channel number, X.25, 118
Virtual channels, 111
Virtual circuit routing, 16, 26, 111, 116, 363, 476–77, 490
Voice:
  digitized, 11
  packetized, 13–14
Voice mail, 8
Voice network, 5, 8
Voice-grade. *See* Communication channels

W

WAN. *See* Wide area network
Walk in a graph, 387
Wide area network, 4, 6
Window (permit) in TCP, 127
Window flow control, 117, 158, 500–10

X

X.21 interface, 21–22, 118
X.25 standard, 40, 118–20, 500-10
  LAPB, 97–103

Solutions Manual -- Data Networks, 2/E by Dimitri Bertsekas and Robert Gallager

TABLE OF CONTENTS

CHAPTER 1 1

CHAPTER 2 2

CHAPTER 3 23

CHAPTER 4 91

CHAPTER 5 114

CHAPTER 6 148

ACKNOWLEDGMENTS

Several of our students have contributed to this solutions manual. We are particularly thankful to Rajesh Pankaj, Jane Simmons, John Spinelli, and Manos Varvarigos.

CHAPTER 1 SOLUTIONS

1.1

There are 250,000 pixels per square inch, and multiplying by the number of square inches and the number of bits per pixel gives 5.61×10^8 bits.

1.2

a) There are 16×10^9 bits going into the network per hour. Thus there are 48×10^9 bits per hour traveling through the network, or 13.33 million bits per second. This requires 209 links of 64 kbit/sec. each.

b) Since a telephone conversation requires two people, and 10% of the people are busy on the average, we have 50,000 simultaneous calls on the average, which requires 150,000 links on the average. Both the answer in a) and b) must be multiplied by some factor to provide enough links to avoid congestion (and to provide local access loops to each telephone), but the point of the problem is to illustrate how little data, both in absolute and comparative terms, is required for ordinary data transactions by people.

1.3

There are two possible interpretations of the problem. In the first, packets can be arbitrarily delayed or lost and can also get out of order in the network. In this interpretation, if a packet from A to B is sent at time τ and not received by some later time t, there is no way to tell whether that packet will ever arrive later. Thus if any data packet or protocol packet from A to B is lost, node B can never terminate with the assurance that it will never receive another packet.

In the second interpretation, packets can be arbitrarily delayed or lost, but cannot get out of order. Assume that each node is initially in a communication state, exchanging data packets. Then each node, perhaps at different times, goes into a state or set of states in which it sends protocol packets in an attempt to terminate. Assume that a node can enter the final termination state only on the receipt of one of these protocol packets (since timing information cannot help, since there is no side information, and since any data packet could be followed by another data packet). As in the three army problem, assume any particular ordering in which the two nodes receive protocol packets. The first node to receive a protocol packet cannot go to the final termination state since it has no assurance that any protocol packet will ever be received by the other node, and thus no assurance that the other node will ever terminate. The next protocol packet to be received then finds neither node in the final termination state. Thus again the receiving node cannot terminate without the possibility that the other node will receive no more protocol packets and thus never terminate. The same situation occurs on each received protocol packet, and thus it is impossible to guarantee that both nodes can eventually terminate. This is essentially the same argument as used for the three army problem.

CHAPTER 2 SOLUTIONS

2.1

Let x(t) be the output for the single pulse shown in Fig. 2.3(a) and let y(t) be the output for the sequence of pulses in Fig. 2.3(b). The input for 2.3(b) is the sum of six input pulses of the type in 2.3(a); the first such pulse is identical to that of 2.3(a), the second is delayed by T time units, the third is inverted and delayed by 2T time units, etc. From the time invariance property, the response to the second pulse above is x(t-T) (i.e. x(t) delayed by T); from the time invariance and linearity, the response to the third pulse is -x(t-2T). Using linearity to add the responses to the six pulses, the overall output is

$$y(t) = x(t) + x(t-T) - x(t-2T) + x(t-3T) - x(t-4T) - x(t-5T)$$

To put the result in more explicit form, note that

$$x(t) = \begin{cases} 0 & ; \quad t < 0 \\ 1 - e^{-2t/T} & ; \quad 0 \le t < T \\ (e^2 - 1)e^{-2t/T} & ; \quad t \ge T \end{cases}$$

Thus the response from the i^{th} pulse ($1 \le i \le 6$) is zero up to time (i-1)T. For $t < 0$, then, $y(t) = 0$; from $0 \le t < T$

$$y(t) = x(t) = 1 - e^{-2t/T} \quad ; \; 0 \le t < T$$

From $T \le t < 2T$,

$$y(t) = x(t) + x(t-T)$$

$$= (e2 - 1)e^{-2t/T} + [1 - e^{-2(t-T)/T}]$$

$$= 1 - e^{-2t/T}$$

Similarly, for $2T \le t < 3T$,

$$y(t) = x(t) + x(t-T) - x(t-2T)$$

$$= (e^2 - 1)e^{-2t/T} + (e^2 - 1)e^{-2(t-T)/T} - [1 - e^{-2(t-2T)/T}]$$

$$= -1 + (2e^4 - 1)e^{-2t/T} \quad ; \; 2T \le t < 3T$$

A similar analysis for each subsequent interval leads to

$$y(t) = 1 - (2e^6 - 2e^4 + 1)e^{-2t/T} \quad ; \quad 3T \le t < 4T$$

$$= -1 + (2e^8 - 2e^6 + 2e^4 - 1)e^{-2t/T} \quad ; \quad 4T \le t < 6T$$

$$= -(e^{12} - 2e^8 + 2e^6 - 2e^4 + 1)e^{-2t/T} \quad ; \quad t \ge 6T$$

The solution is continuous over t with slope discontinuities at 0, 2T, 3T, 4T, and 6T; the value of y(t) at these points is y(0) = 0; y(2T) = .982; y(3T) = -.732; y(4T) = .766; y(6T) = -.968. Another approach to the problem that gets the solution with less work is to use x(t) to first find the response to a unit step and then view y(t) as the response to a sum of displaced unit steps.

2.2

From the convolution equation, Eq. (2.1), the output r(t) is

$$r(t) = \int_{-\infty}^{\infty} s(\tau)h(t-\tau)d\tau = \int_{\tau=0}^{T} h(t-\tau)d\tau$$

Note that $h(t-\tau) = \alpha e^{-\alpha(t-\tau)}$ for $t-\tau \geq 0$, (i.e. for $\tau \leq t$), and $h(t-\tau) = 0$ for $\tau > t$. Thus for $t < 0$, $h(t-\tau) = 0$ throughout the integration interval above. For $0 \leq t < T$, we then have

$$r(t) = \int_{\tau=0}^{t} \alpha e^{-\alpha(t-\tau)}d\tau + \int_{\tau=t}^{T} 0\, d\tau = 1 - e^{-\alpha t} \quad ; 0 \leq t < T$$

For $t \geq T$, $h(t-\tau) = \alpha e^{-\alpha(t-\tau)}$ over the entire integration interval and

$$r(t) = \int_{\tau=0}^{T} \alpha e^{-\alpha(t-\tau)}d\tau = e^{-\alpha(t-T)} - e^{-\alpha t} \quad ; \; t \geq T$$

Thus the response increases towards 1 for $0 \leq t \leq T$ with the exponential decay factor α, and then, for $t \geq T$, decays toward 0.

2.3

From Eq. (2.1),

$$r(t) = \int_{-\infty}^{\infty} e^{j2\pi f\tau}h(t-\tau)d\tau$$

Using $\tau' = t-\tau$ as the variable of integration for any given t,

$$r(t) = \int_{-\infty}^{\infty} e^{j2\pi f(t-\tau')}h(\tau')d\tau'$$

$$= e^{j2\pi ft}\int_{-\infty}^{\infty} e^{-j2\pi f\tau'}h(\tau')d\tau'$$

$$= e^{j2\pi ft}H(f)$$

where H(f) is as given in Eq. (2.3).

2.4

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

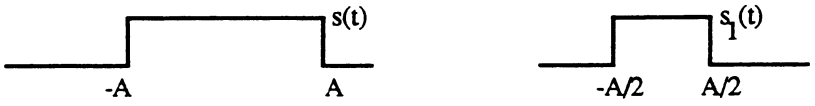
Since H(f) is 1 from $-f_0$ to f_0 and 0 elsewhere, we can integrate $\exp(j2\pi ft)$ from $-f_0$ to f_0, obtaining

$$h(t) = \frac{1}{j2\pi t}[\exp(j2\pi f_0 t) - \exp(-i2\pi f_0 t)] = \frac{\sin(2\pi f_0 t)}{\pi t}$$

Note that this impulse response is unrealizable in the sense that the response starts before the impulse (and, even worse, starts an infinite time before the impulse). None the less, such ideal filters are useful abstractions in practice.

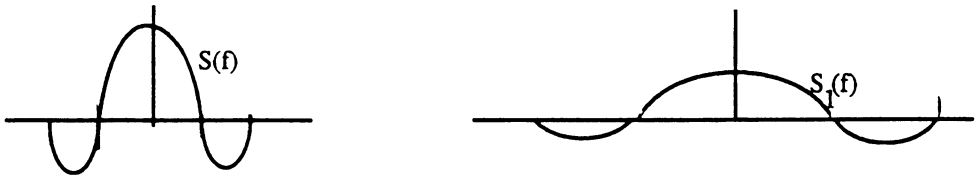
2.5

The function $s_1(t)$ is compressed by a factor of β on the time axis as shown below



$$S_1(f) = \int_{-\infty}^{\infty} s_1(t)e^{-j2\pi ft}dt = \int_{-\infty}^{\infty} s(\beta t)e^{-j2\pi ft}dt$$

$$= \int_{-\infty}^{\infty} s(\tau)e^{-j2\pi f\tau/\beta}\frac{d\tau}{\beta} = \frac{1}{\beta}S\left(\frac{f}{\beta}\right)$$



Thus $S_1(f)$ is attenuated by a factor of β in amplitude and expanded by a factor of β on the frequency scale; compressing a function in time expands it in frequency and vice versa.

2.6

a) We use the fact that $\cos(x) = [\exp(jx) + \exp(-jx)]/2$. Thus the Fourier transform of $s(t)\cos(2\pi f_0 t)$ is

$$\int_{-\infty}^{\infty} s(t)\frac{\exp(j2\pi f_0 t) + \exp(-j2\pi f_0 t)}{2}\exp(-j2\pi ft)\,dt$$

$$= \frac{s(t)}{2}\exp[-j2\pi(f-f_0)t]\,dt + \frac{s(t)}{2}\exp[-j2\pi(f+f_0)t]\,dt$$

$$= \frac{S(f-f_0)}{2} + \frac{S(f+f_0)}{2}$$

b) Here we use the identity $\cos^2(x) = [1+\cos(2x)]/2$. Thus the Fourier transform of $s(t)\cos^2(2pf_0t)$ is the Fourier transform of $s(t)/2$ plus the Fourier transform of $s(t)\cos[2p(2f_0)t]/2$. Using the result in part a, this is $S(f)/2 + S(f-2f_0)/4 + S(f+2f_0)/4$.

2.7

a) E{frame time on 9600 bps link} = 1000 bits / 9600 bps = 0.104 sec.

E{frame time on 50,000bps link} = 0.02 sec.

b) E{time for 10^6 frames on 9600 bps link} = $1.04 \cdot 10^5$ sec.

E{time for 10^6 frames on 50,000 bps link} = $2 \cdot 10^4$ sec.

Since the frame lengths are statistically independent, the variance of the total number of bits in 10^6 frames is 10^6 times the variance for one frame. Thus the standard deviation of the total number of bits in 10^6 frames is 10^3 times the standard deviation of the bits in one frame or $5 \cdot 10^5$ bits. The standard deviation of the transmission time is then

S.D.{time for 10^6 frames on 9600 bps link} = $5 \cdot 10^5 / 9600 = 52$ sec.

S.D.{time for 10^6 frames on 50,000 bps link} = $5 \cdot 10^5 / 50{,}000 = 10$ sec.

c) The point of all the above calculations is to see that, for a large number of frames, the expected time to transmit the frames is very much larger than the standard deviation of the transmission time; that is, the time per frame, averaged over a very long sequence of frames, is close to the expected frame time with high probability. One's intuition would then suggest that the number of frames per unit time, averaged over a very long time period, is close to the reciprocal of the expected frame time with high probability. This intuition is correct and follows either from renewal theory or from direct analysis. Thus the reciprocal of the expected frame time is the rate of frame transmissions in the usual sense of the word "rate".

2.8

Let x_{ij} be the bit in row i, column j. Then the ith horizontal parity check is

$$h_i = \Sigma_j x_{ij}$$

where the summation is summation modulo 2. Summing both sides of this equation (modulo 2) over the rows i, we have

$$\Sigma_i h_i = \Sigma_{i,j} x_{ij}$$

This shows that the modulo 2 sum of all the horizontal parity checks is the same as the modulo 2 sum of all the data bits. The corresponding argument on columns shows that the modulo 2 sum of the vertical parity checks is the same.

2.9

a) Any pattern of the form

--- 1 1 0 ---
--- 0 1 1 ---
--- 1 0 1 ---

will fail to be detected by horizontal and vertical parity checks. More formally, for any three rows i_1, i_2, and i_3, and any three columns j_1, j_2, and j_3, a pattern of six errors in positions $(i_1\ j_1)$, $(i_1\ j_2)$, $(i_2\ j_2)$, $(i_2\ j_3)$, $(i_3\ j_1)$, and $(i_3\ j_3)$ will fail to be detected.

b) The four errors must be confined to two rows, two errors in each, and to two columns, two errors in each; that is, geometrically, they must occur at the vertices of a rectangle within the array. Assuming that the data part of the array is J by K, then the array including the parity check bits is J+1 by K+1. There are (J+1)J/2 different possible pairs of rows (counting the row of vertical parity checks), and (K+1)K/2 possible pairs of columns (counting the column of horizontal checks). Thus there are (J+1)(K+1)JK/4 undetectable patterns of four errors.

2.10

Let $x = (x_1, x_2, \cdots x_N)$ and $x' = (x'_1, x'_2, \cdots x'_N)$ be any two distinct code words in a parity check code. Here N = K+L is the length of the code words (K data bits plus L check bits). Let $y = (y_1, \cdots y_N)$ be any given binary string of length N. Let D(x,y) be the distance between x and y (i.e. the number of positions i for which $x_i \neq y_i$). Similarly let D(x',y) and D(x,x') be the distances between x' and y and between x and x'. We now show that

$$D(x,x') \leq D(x,y) + D(x',y)$$

To see this, visualize changing D(x,y) bits in x to obtain y, and then changing D(x',y) bits in y to obtain x'. If no bit has been changed twice in going from x to y and then to x', then it was necessary to change D(x,y) + D(x',y) bits to change x to x' and the above inequality is satisfied with equality. If some bits have been changed twice (i.e. $x_i = x'_i \neq y_i$ for some i) then strict inequality holds above.

By definition of the minimum distance d of a code, $D(x,x') \geq d$. Thus, using the above inequality, if $D(x,y) < d/2$, then $D(x',y) > d/2$. Now suppose that code word x is sent and fewer than d/2 errors occur. Then the received string y satisfies $D(x,y) < d/2$ and for every other code word x', $D(x',y) > d/2$. Thus a decoder that maps y into the closest code word must select x, showing that no decoding error can be made if fewer than d/2 channel errors occur. Note that this argument applies to any binary code rather than just parity check codes.

2.11

The first code word given, 1001011 has only the first data bit equal to 1 and has the first, third, and fourth parity checks equal to 1. Thus those parity checks must check on the first data bit. Similarly, from the second code word, we see that the first, second, and fourth parity checks must check on the second bit. From the third code word, the first, second, and third parity check each check on the third data bit. Thus

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

The set of all code words is given by

0000000	0011110
1001011	1010101
0101101	0110011
1100110	1111000

The minimum distance of the code is 4, as can be seen by comparing all pairs of code words. An easier way to find the minimum distance of a parity check code is to observe that if x and x' are each code words, then x + x' (using modulo 2 componentwise addition) is also a code word. On the other hand, x + x' has a 1 in a position if and only if x and x' differ in that position. Thus the distance between x and x' is the number of ones in x + x'. It follows that the minimum distance of a parity check code is the minimum, over all non-zero code words, of the number of ones in each code word.

2.12

$$\begin{array}{r} D^3 \\ D^4+D^2+D+1 \overline{)\ D^7+D^5+D^4} \\ \underline{D^7+D^5+D^4+D^3} \\ D^3 \end{array} = \text{Remainder}$$

2.13

Let $z(D) = D^j + z_{j-1}D^{j-1} + \ldots + D^i$ and assume i<j. Multiplying G(D) times Z(D) then yields

$$g(D)z(D) = D^{L+j} + (z_{j-1} + g_{L-1})D^{L+j-1} + (z_{j-2} + g_{L-1}z_{j-1} + g_{L-2})D^{L+j-2} + \ldots + (g_1 + z_{i+1})D^{i+1} + D^i$$

Clearly the coefficient of D^{L+j} and the coefficient of D^i are each 1, yielding the desired two non-zero terms. The above case i<j arises whenever z(D) has more than one non-zero term. For the case in which z(D) has only one non-zero term, i.e. z(D) = Dj for some j, we have

$$g(D)z(D) = D^{L+j} + g_{L-1}D^{L+j-1} + \ldots + D^j$$

which again has at least two non-zero terms.

2.14

Suppose g(D) contains (1+D) as a factor; thus g(D) = (1+D)h(D) for some polynomial h(D). Substituting 1 for D and evaluating with modulo 2 arithmetic, we get g(1) = 0 because of the term (1+D) = (1+1) = 0. Let e(D) be the polynomial for some arbitrary undetectable error sequence. Then e(D) = g(D)z(D) for some z(D), and hence e(1) = g(1)z(1) = 0. Now $e(D) = \Sigma_i e_i D^i$, so $e(1) = \Sigma_i e_i$. Thus e(1) = 0 implies that an even number of elements e_i

are 1; i.e. that e(D) corresponds to an even number of errors. Thus all undetectable error sequences contain an even number of errors; any error sequence with an odd number of errors is detected.

2.15

a) Let D^{i+L}, divided by g(D), have the quotient $z^{(i)}(D)$ and remainder $c^{(i)}(D)$ so that

$$D^{i+L} = g(D)z^{(i)}(D) + c^{(i)}(D)$$

Multiplying by s_i and summing over i,

$$s(D)D^L = \Sigma_i\, s_i z^{(i)}(D) + \Sigma_i\, s_i c^{(i)}(D)$$

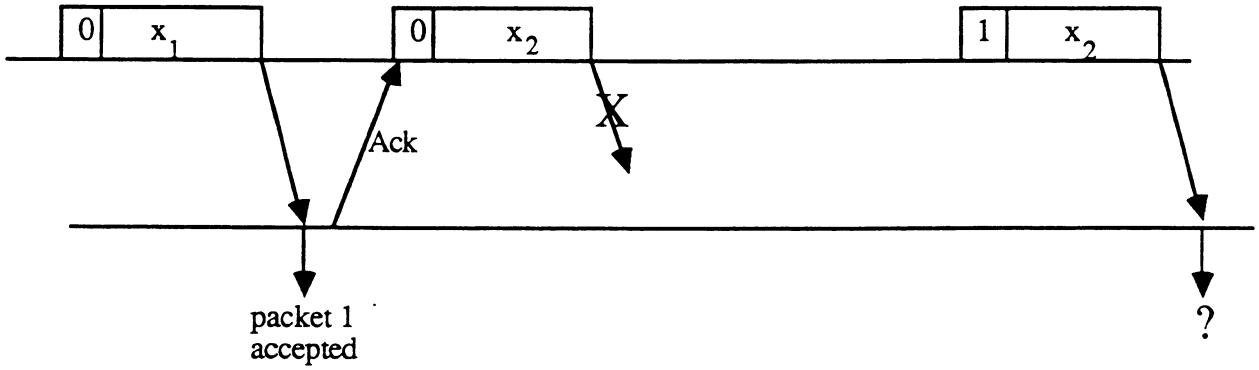
Since $\Sigma_i\, s_i c^{(i)}(D)$ has degree less than L, this must be the remainder (and $\Sigma_i\, s_i z^{(i)}(D)$ the quotient) on dividing $s(D)D^L$ by g(D). Thus $c(D) = \Sigma_i\, s_i c^{(i)}(D)$.

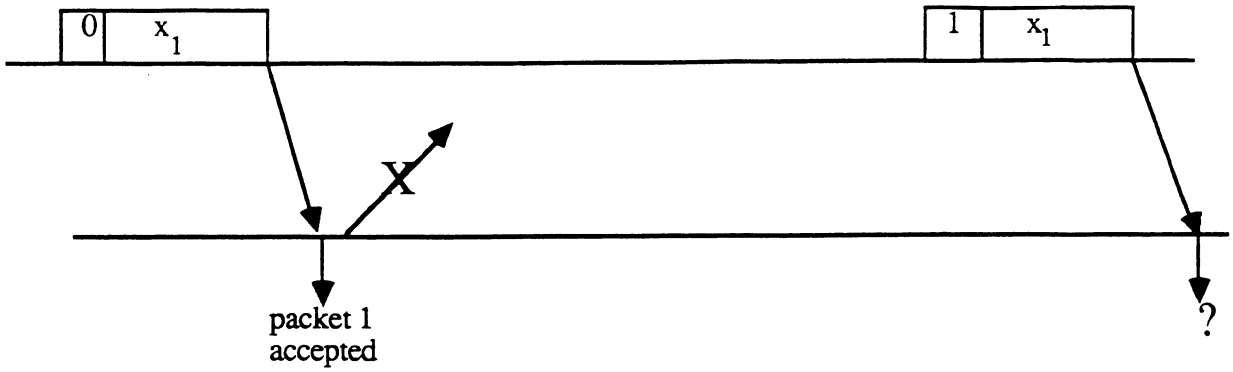
b) Two polynomials are equal if and only if their coefficients are equal, so the above polynomial equality implies

$$c_j = \Sigma_i\, s_i c_j^{(i)}$$

2.16

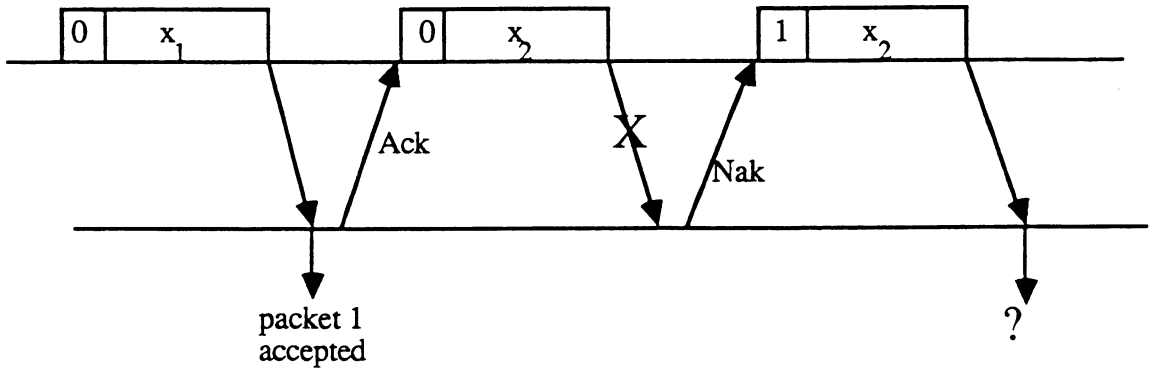
a) Consider the two scenarios below and note that these scenarios are indistinguishable to the receiver.

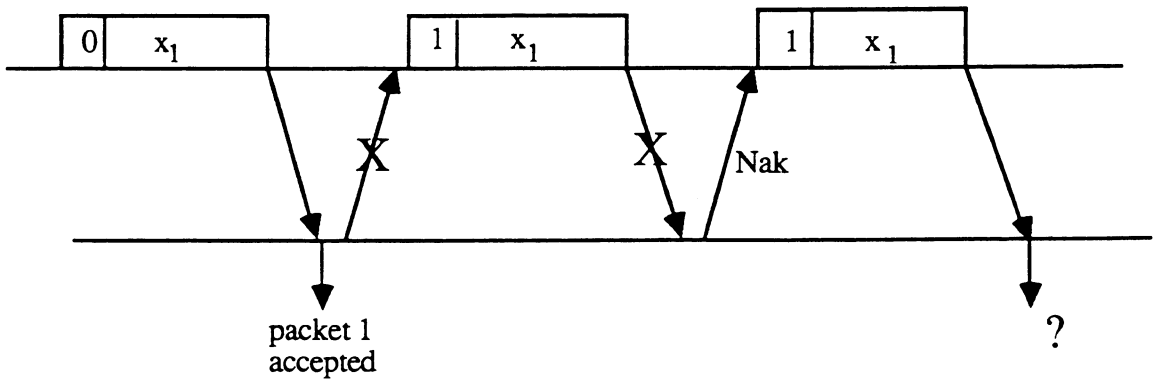




If the receiver releases the packet as x_2 in the questioned reception, then an error occurs on scenario 2. If the receiver returns an ack but doesn't release a packet (i.e. the appropriate action for scenario 2), then under scenario 1, the transmitter erroneously goes on to packet 3. Finally, if the receiver returns a nak, the problem is only postponed since the transmitter would then transmit $(2,x_2)$ in scenario 1 and $(2,x_1)$ in scenario 2. As explained on page 66, packets x_1 and x_2 might be identical bit strings, so the receiver can not resolve its ambiguity by the bit values of the packets.

b) The scenarios below demonstrate incorrect operation for the modified conditions.





2.17

a) $T = T_t + T_f + 2T_d$

b) $q = (1-p_t)(1-p_f)$

A packet is transmitted once with probability q, twice with probability (1-q)q, three times with probability $(1-q)^2q$, etc. Thus the expected number of transmissions of a given packet is

$$E\{\text{transmissions per packet}\} = \sum_{i=1}^{\infty} iq(1-q)^{i-1} = \frac{1}{q}$$

To verify the above summation, note that for any x, $0 \le x < 1$,

$$\sum_{i=1}^{\infty} ix^{i-1} = \sum_{i=1}^{\infty} \frac{dx^i}{dx} = \frac{d}{dx}\sum_{i=1}^{\infty} x^i = \frac{d}{dx}\left(\frac{x}{1-x}\right) = \frac{1}{(1-x)^2}$$

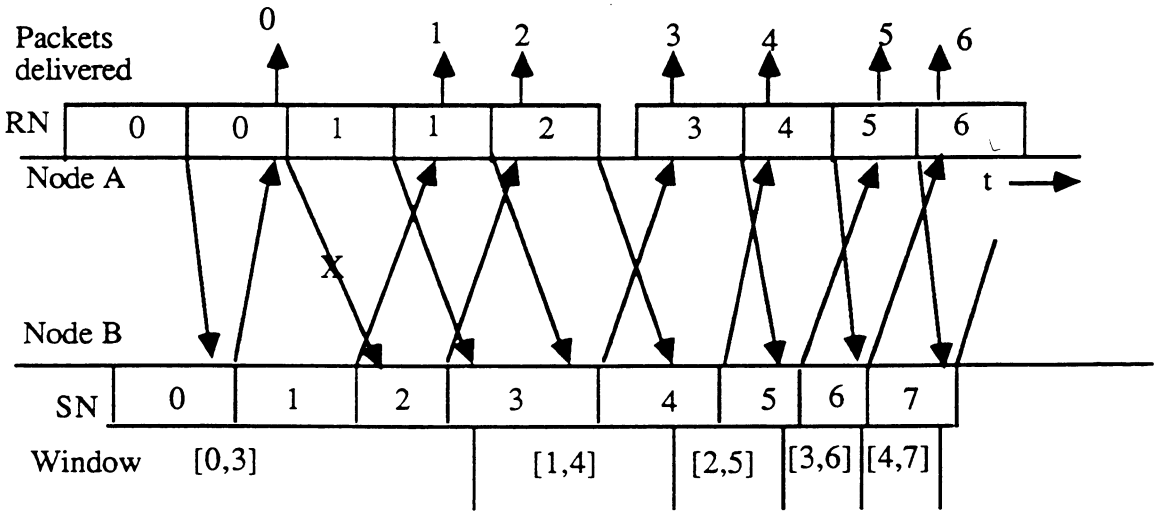
Using x for (1-q) above gives the desired result.

c) $E\{\text{time per packet}\} = (T_t + T_f + 2T_d)/q$

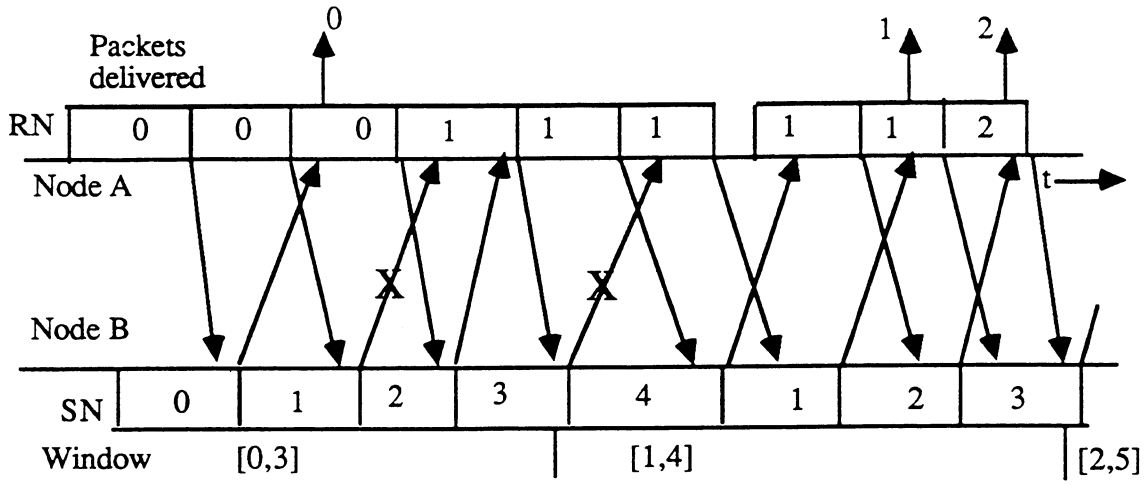
$= (1.3)/0.998 = 1.303$

Note that p_t and p_f have very little effect on E[time per packet] in stop and wait systems unless they are unusually large.

2.18

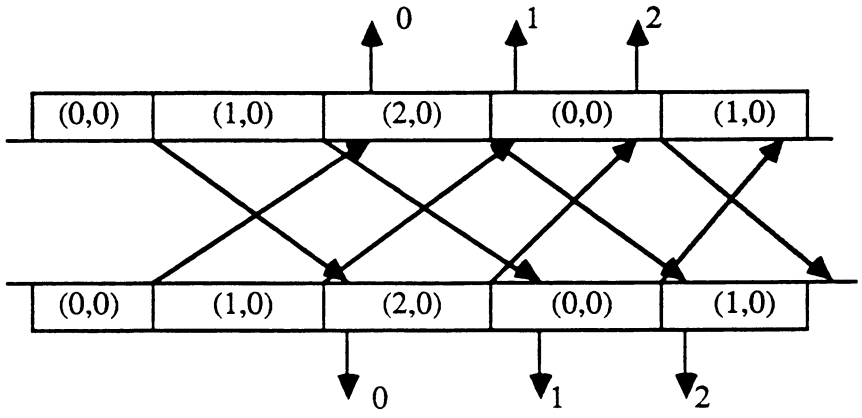


Assume that the transmitter always sends the next packet in order until reaching the end of the window.



2.19

The simplest such deadlock occurs if there is sufficient propagation delay in the system that each side can send n-1 frames (containing packets numbered 0 to n-2) before finishing receipt of the first frame from the other side. In this case, the n^{th} frame from each side will carry the packet numbered n-1 without acking any packets from the other side. Thus each side will go back to packet 0, but in the absence of errors, each side will be looking for packet n by time the repeat of packet 0 occurs. Each side will then cycle from 0 to n-1, and neither side will ever receive any acks. The diagram below illustrates this for n=3. The first number in each frame position is SN and the second is RN.



2.20

The simplest example is for node A to send packets 0 through n-1 in the first n frames. In case of delayed acknowledgements (i.e. no return packets in the interim), node A goes back and retransmits packet 0. If the other node has received all the packets, it is waiting for packet n, and if the modulus m equals n, this repeat of packet 0 is interpreted as packet n.

The right hand side of Eq. (2.24) is satisfied with equality if $SN = SN_{min}(t_1)+n-1$. This occurs if node A sends packets 0 through n-1 in the first n frames with no return packets from node B. The last such frame has $SN = n-1$, whereas SN_{min} at that time (say t_1) is 0.

Continuing this scenario, we find an example where the right hand side of Eq. (2.25) is satisfied with equality. If all the frames above are correctly received, then after the last frame, RN becomes equal to n. If another frame is sent from A (now call this time t_1) and if SN_{min} is still 0, then when it is received at B (say at t_2), we have $RN(t_2) = SN_{min}(t_1)+n$.

2.21

Let $RN(\tau)$ be the value of RN at node B at an arbitrary time τ; SN_{min} is the smallest packet number not yet acknowledged at A at time t (which is regarded as fixed) and $SN_{max}-1$ is the largest packet number sent from A at time t. Since $RN(\tau)$ is non decreasing in τ, it is sufficient to show that $RN(t+T_m+T_d) \leq SN_{max}$ and to show that $RN(t-T_m-T_d) \geq SN_{min}$.

For the first inequality, note that the packet numbered SN_{max} (by definition of SN_{max}) has not entered the DLC unit at node A by time t, and thus can not have started transmission by time t. Since there is a delay of at least T_m+T_d from the time a packet transmission starts until the completion of its reception, packet SN_{max} can not have been received by time $t+T_m+T_d$. Because of the correctness of the protocol, $RN(t+T_m+T_d)$ can be no greater than the number of a packet not yet received, i.e. SN_{max}.

For the second inequality, note that for the transmitter to have a given value of SN_{min} at time t, that value must have been transmitted earlier as the request number in a frame coming back from node B. The latest time that such a frame could have been formed is $t-T_m-T_d$, so by this time RN must have been at least SN_{min}.

2.22

a) If the transmitter never has to go back or wait in the absence of errors, then it can send a continuous stream of new packets in the absence of errors. In order for such a continuous stream to be sent, each packet must be acknowledged (i.e. SN_{min} must advance beyond the packet's number) before the next n-1 frames complete transmission. Thus these n-1 frame transmission times are in a race with the time, first, for the given packet to propagate over the channel and, second, for the acknowledgement to wait for the feedback frame in progress, then wait to be transmitted in the next feedback frame and propagated back to the original transmitter. In order for the feedback to always win the race, the minimum time for the n-1 frames to be transmitted must be greater than the maximum time for the feedback, i.e.,

$$(n-1)T_{min} > 2T_d + 2T_{max}$$

$$T_{max} < [(n-1)/2]T_{min} - T_d$$

b) If an isolated error occurs in the feedback direction, the feedback could be held up for one additional frame, leading to

$$(n-1)T_{min} > 2T_d + 3T_{max}$$

$$T_{max} < [(n-1)/3]T_{min} - (2/3)T_d$$

2.23

After a given packet is transmitted from node A, the second subsequent frame transmission termination from B carries the acknowledgement (recall that the frame transmission in progress from B when A finishes its transmission cannot carry the ack for that transmission; recall also that propagation and processing delays are negligible. Thus q is the probability of n-1 frame terminations from A before the second frame termination from B. This can be rephrased as the probability that out of the next n frame terminations from either node, either n-1 or n come from node A. Since successive frame terminations are equally likely and independently from A or B, this probability is

$$q = \sum_{i=n-1}^{n} \frac{n!}{i!(n-i)!} 2^{-n} = (n+1)2^{-n}$$

2.24

If an isolated error in the feedback direction occurs, then the ack for a given packet is held up by one frame in the feedback direction (i.e., the number RN in the feedback frame following the feedback frame in error reacknowledges the old packet as well as any new packet that might have been received in the interim). Thus q is now the probability of n-1 frame terminations from A before 3 frame terminations from B (one for the frame in progress, one for the frame in error, and one for the frame actually carrying the ack; see the solution to problem 2.23). This is the probability that n-1 or more of the next n+1 frame terminations come from A; since each termination is from A or B independently and with equal probability,

$$q = \sum_{i=n-1}^{n} \left(\frac{(n+1)!}{i!(n+1-i)!}\right) 2^{-n-1} = [n+2+(n+1)n/2]2^{-n-1}$$

2.25

As in the solution to problem 2.23, q is the probability of n-1 frame terminations coming from node A before two frame terminations come from node B. Frame terminations from A (and similarly from B) can be regarded as alternate points in a Poisson point process from A (or from B). There are two cases to consider. In the first, the initial frame is received from A after an even numbered point in the Poisson process at B, and in the second, the initial frame is received after an odd numbered point. In the first case, q is the probability that 2n-2 Poisson events from A occur before 4 Poisson events occur from B. This is the probability, in a combined Poisson point process of Poisson events from A and B, that 2n-2 or more Poisson events come from A out of the next 2n+1 events in the combined process. In the second case, q is the probability that 2n-2 Poisson events from A occur before 3 events occur from B. Since these cases are equally likely,

$$q = \frac{1}{2}\sum_{i=2n-2}^{2n+1} \left(\frac{(2n+1)!}{i!(2n+1-i)!}\right) 2^{-2n-1} + \frac{1}{2}\sum_{i=2n-2}^{2n} \left(\frac{(2n)!}{i!(2n-i)!}\right) 2^{-2n}$$

2.26

We view the system from the receiver and ask for the expected number of frames, γ, arriving at the receiver starting immediately after a frame containing a packet that is accepted

and running until the next frame containing a packet that is accepted. By the assumptions of the problem, if the packet in a frame is accepted, then the next frame must contain the next packet in order (if not, the transmitter must have gone back to some earlier packet, which is impossible since that earlier packet was accepted earlier and by assumption was acked in time to avoid the go back).

Since the next frame after a packet acceptance must contain the awaited packet, that packet is accepted with probability 1-p. With probability p, on the other hand, that next frame contains an error. In this case, some number of frames, say j, follow this next frame before the awaited packet is again contained in a frame. This new frame might again contain an error, but the expected number of frames until the awaited packet is accepted, starting with this new frame, is again γ. Thus, given an error in the frame after a packet acceptance, and given j further frames before the awaited packet is repeated, the expected number of frames from one acceptance to the next is $1+j+\gamma$.

Note that j is the number of frames that the transmitter sends, after the above frame in error, up to and including the frame in transmission when feedback arrives concerning the frame in error. Thus the expected value of j is β. Combining the events of error and no error on the next frame after a packet acceptance, we have

$$\gamma = (1-p) + p(1+\beta+\gamma) = 1 + p(\beta+\gamma)$$

Solving for γ and for $\nu = 1/\gamma$,

$$\gamma = (1+\beta p)/(1-p) \qquad \nu = (1-p)/(1+\beta p)$$

2.27

Note that the sending DLC could save only one packet if it waited for acknowledgements rather than continuing to transmit. Similarly the sending DLC could save an arbitrarily large number of packets by taking packets from the network layer at a rate faster than they can be transmitted. Thus what is desired is to show that at most $\beta+1$ packets need be stored without ever forcing the transmitter to wait. Thus we assume that a new packet is admitted from the network layer only when there are no previously transmitted packets that are known to have been unsuccessful on the last transmission (i.e. the system repeats nak'ed packets before accepting and transmitting new packets; the system accepts and transmits new packets while waiting for feedback information on old packets).

When the system is first initiated, one packet is admitted to the sending DLC from the network layer. We use this as the basis of an inductive argument on successive times at which a new frame is generated. By the inductive hypothesis, at most $\beta+1$ packets were stored at the end of the previous frame generation instant. At the time of generating the new frame, there are at most β outstanding frames (including the one just being completed) for which feedback has not been received. A new packet will be accepted from the network layer only if all packets stored are also in frames for which no feedback has yet been received. Thus if a new packet is accepted, the total number saved is increased to at most $\beta+1$, and if no new packet is accepted, the total number saved remains (by the inductive hypothesis) no more than $\beta+1$.

2.28

Under the given assumptions, the ARPANET ARQ works like ideal select repeat. That is, frames from the 8 channels can be sent in round robin order and the feedback for a channel is always available by time the channel is to be reused. Thus a packet is repeated if and only if the previous transmission was unsuccessful. Since all channels are constantly busy and only the frames in error lead to retransmission, the efficiency is 1-p.

2.29

a) When packet z is transmitted, the transmitter rule ensures that $z \leq SN_{min}+n-1$. At that time, $SN_{min} \leq RN$ since a packet cannot be acked before being received. Thus, at transmit time

$$z \leq RN + n - 1$$

Since RN is nondecreasing, this is also satisfied at receive time. To derive the lower bound on z, note that the transmitter rule specifies $z \geq SN_{min}$. Since $SN_{min}+n$ has never been sent before the current transmission of z, the first come first serve order on the link ensures that it is not received before z. Thus y_{top} at receive time is less than $SN_{min}+n$ at transmit time, so

$$z \geq SN_{min} > y_{top} - n$$

$$z \geq y_{top} - n + 1$$

b) We must ensure that m is large enough to always satisfy

$$z + m > y_{top} + k$$

We know from a) that $z > y_{top}-n$, and adding n+k to both sides of this equation, we know that $z+n+k > y_{top}+k$. Thus, choosing m = n+k (or, more generally, $m \geq n+k$) always satisfies the above equation. If m is chosen any smaller (say m = n+k-1), then when $z = y_{top}-n+1$ (which can happen after a goback), z+m will equal $y_{top}+k$, causing erroneous operation.

c) From b), $m \geq n+k > n$. From Eq. (2.47), $z \leq RN+n-1 < RN+m$; thus $z-m < RN$.

d) From b) and c), $m \geq n+k$ assures correct operation at the receiver. Since m > n, correct operation at the transmitter is assured as in goback n.

e) Initially $y_{top} = RN-1$, so for k=1, the receiver can initially accept only RN. On each accepted packet, RN and y_{top} are each incremented by 1, so at all times only RN can be accepted. Thus k=1 is ordinary goback n ARQ. For k=n, all received z must satisfy $z \leq y_{top}+k$, and we have ordinary selective repeat ARQ.

2.30

a) The sequence below shows the stuffed bits underlined for easy readability:

0 1 1 0 1 1 1 1 1 0 0 0 1 1 1 1 1 0 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 0 0 1 1 1 1 0 1 0

b) Here the flags are shown underlined and the removed (destuffed) bits as x's:

$\underline{01111110}11111x11\ 0011111x011111x111111x11$

$00\underline{01111110}1011111x$

2.31

The modified destuffing rule starts at the beginning of the string and destuffs bit by bit. A zero is removed from the string if the previous six bits in the already destuffed portion of the string have the value 01^5. For the given example, the destuffed string, with flags shown underlined and removed bits shown as x's, is as follows:

$$01101111x1111110111111x1\underline{01111110}$$

2.32

The hint shows that the data string $01^501x_1x_2...$ must have a zero stuffed after 01^5, thus appearing as $01^500x_1x_2....$ This stuffed pattern will be indistinguishable from the original string $01^500x_1x_2...$ unless stuffing is also used after 01^5 in the string $01^500x_1x_2....$ Thus stuffing must be used in this case. The general argument is then by induction. Assume that stuffing is necessary after 01^5 on all strings of the form $01^50kx_1x_2....$ Then such a stuffed sequence is $01^50k+1x_1x_2....$ It follows as before that stuffing is then necessary after 01^5 in the sequence $01^50k+1x_1x_2....$ Thus stuffing is always necessary after 01^5.

2.33

The stuffed string is shown below with the stuffed bits underlined and a flag added at the end.

$$1101101\underline{0}010\underline{0}10\underline{0}1110100\underline{0}101$$

The destuffing rule is to decode (destuff) the string bit by bit starting at the beginning. A given 0 bit is then deleted from the string if the preceding three decoded bits are 010. The flag is detected when a 1 is preceded by the three decoded bits 010 and the most recently decoded bit was not deleted. The above is a general rule for detecting any type of flag sequence, rather than just 0101; for this special case, it is sufficient to look for the substring 0101 in the received string; the reason for the simplification is that if an insertion occurs within the flag, it has to occur by simply a repetition of the first flag bit.

2.34

Let γ be $\log_2E\{K\} - j$. Since j is the integer part of $\log_2E\{K\}$, we see that γ must lie between 0 and 1. Expressing $A = E\{K\}2^{-j} + j + 1$ in terms of γ and $E\{K\}$, we get

$$A = 2^{\gamma} + \log_2E\{K\} - \gamma + 1$$

$$A - \log_2E\{K\} = 2^{\gamma} - \gamma + 1$$

This function of γ is easily seen to be convex (i.e., it has a positive second derivative). It has the value 2 at $\gamma = 0$ and at $\gamma = 1$ and is less than 2 for $0 < \gamma < 1$. This establishes that

$$A \leq \log_2 E\{K\} + 2$$

Finding the minimum of $2^{\gamma} - \gamma + 1$ by differentiation, the minimum occurs at

$$\gamma = -\log_2(\ln 2)$$

The value of $2^{\gamma} - \gamma + 1$ at this minimizing point is $[\ln 2]^{-1} + \log_2(\ln 2) + 1 = 1.914...$, so

$$A \geq \log_2 E\{K\} + (\ln 2)^{-1} + \log_2(\ln 2) + 1$$

2.35

Stuffed bits are always 0's and always follow the pattern 01^5. The initial 0 in this pattern could be a bit in the unstuffed data string, or could itself be a stuffed bit. As in the analysis of subsection 2.5.2, we ignore the case where this initial 0 is a stuffed bit since it is almost negligible compared with the other case (also a well designed flag detector would not allow a stuffed bit as the first bit of a flag). If a stuffed bit (preceded by 01^5 in the data) is converted by noise into a 1, then it is taken as a flag if the next bit is 0 and is taken as an abort if the next bit is 1. Thus an error in a stuffed bit causes a flag to appear with probability 1/2 and the expected number of falsely detected flags due to errors in stuffed bits is $K2^{-7}$. If one is less crude in the approximations, one sees that there are only K-6 places in the data stream where a stuffed bit could be inserted following 01^5 in the data; thus a more refined answer is that the expected number of falsely detected flags due to errors in stuffed bits is $(K-6)2^{-7}$.

There are eight patterns of eight bits such that an error in one of the eight bits would turn the pattern into a flag. Two of these patterns, 01^7 and 1^70, cannot appear in stuffed data. Another two of the patterns, 01^500 and 001^50, can appear in stuffed data but must contain a stuffed bit (i.e. the 0 following 1^5). The first of these cases corresponds to the case in which an error in a stuffed bit causes a flag to appear, and we have already analyzed this. The second corresponds to a data string 001^5. Thus the substrings of data for which a single error in a data bit can cause a flag to appear are listed below; the position in which the error must appear is shown underlined:

0 $\underline{0}$ 1 1 1 1 1
0 1 $\underline{0}$ 1 1 1 1 0
0 1 1 $\underline{0}$ 1 1 1 0
0 1 1 1 $\underline{0}$ 1 1 0
0 1 1 1 1 $\underline{0}$ 1 0

For any given bit position j in the K bit data string ($j \leq K-7$), the probability that one of these patterns starts on bit j is $2^{-7} + 4.2^{-8} = 3.2^{-7}$. Thus the probability of a false flag being detected because of an error on a data bit, starting on bit j of the data is $3p2^{-7}$. This is also the expected number of such flags, and summing over the bits of the data stream, the expected number is $(K-7)3p2^{-7}$. Approximating by replacing K-7 by K, and adding this to the expected number of false flags due to errors in stuffed bits, the overall probability of a false flag in a frame of length K is (1/32)Kp. If K-7 is not approximated by K, and if we recognize that the first pattern above can also appear starting at j=K-6, then the overall probability of a false flag is approximated more closely by (1/32)(K-6.5)p.

2.36

Let N be the number of overhead bits per packet, F the number of flag bits per packet, U the number of unary code bits per packet, and I the number of insertions per packet. Then

$$N = F + U + I; \qquad E\{N\} = E\{F\} + E\{U\} + E\{I\}$$

A flag will occur at the end of a packet if the next session has nothing to send; thus a flag (containing K bits) occurs at the end of a packet with probability p. It follows that

$$E\{F\} = pK$$

The number of unary bits following a packet is 0 if the next session has something to send and is $j \geq 1$ if the number of following sessions with nothing to send is j. Thus $P\{U=j\} = (1-p)p^j$ for $j \geq 1$.

$$E\{U\} = \sum_{j=1}^{\infty} j(1-p)p^j = \frac{p}{1-p}$$

Finally an insertion occurs if a packet starts with 01^{k-2}. Assuming independent equally likely binary digits in the packets (this is not particularly realistic for packet headers, but it is the only reasonable assumption without looking at the details of some particular protocol), the probability of an insertion at the beginning of a packet is $2^{-(K-1)}$. Thus

$$E\{I\} = 2^{-(K-1)}$$

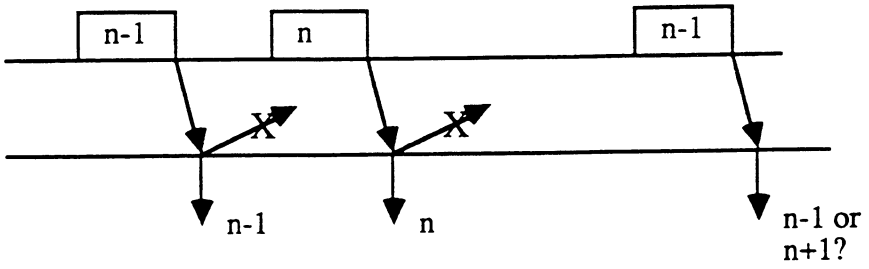
$$E\{N\} = pK + p/(1-p) + 2^{-(K-1)}$$

Note that it is not really necessary to do insertions at the beginning of the first packet following a flag; if one assumes that such insertions are not made, E{I} changes to $(1-p)2^{-(K-1)}$.

b) No problems occur using flags both for addressing as above and for DLC. The DLC regards the addressing flags as part of the data (which can be arbitrary anyway), and the stuffing due to the DLC flags is removed before the network layer sees it. This is one of the advantages of layering, that one doesn't have to worry about such interactions. Note however that the use of this particular flag for addressing will cause a slight increase in the number of insertions required at the DLC layer. When efficiency is important, one can not necessarily ignore the interactions between different layers.

2.37

a) Note that a given packet n can never be sent until after n-1 is acked; this is true even without the possibility of packets getting out of order on the line. To see this, consider the example below.



Note also that there is no possible reason to want to send a packet after it has been acked. Thus the only question here is whether it is possible, or sensible, to retransmit a given packet without waiting for an ack or a period 2T . The simplest rule for the transmitter (and probably the most practical unless T is very large) is for the transmitter to wait after sending each packet for either an ack or nak (i.e. RN equal to the sequence number just transmitted) or for a period of 2T, which guarantees that nothing remains on the link.

In order to leave the transmitter with more freedom than the above, we observe that there are three restrictions on when a given packet n can be transmitted. The first, that n-1 must be acknowledged, was mentioned above. The second is that no transmission of packet n-1 can be on the forward channel. The third is that no ack of packet n-2 can be on the return channel. The reason for the second restriction is that a transmission of packet n could arrive before that of n-1, causing n-1 to be mistaken for n+1. The reason for the third restriction is to avoid the ack for n-2 being mistaken for the ack for n. Letting t_1 be the time at which n-1 was last transmitted, we see that the second restriction above leads to the following rule. In order to transmit packet n at time t, one or more of the following conditions must be satisfied:

i) $t \geq t_1 + T$

ii) The number of acks of n-1 equals the number of transmissions of n-1 up to t

iii) The last ack of n-1 is over 2T seconds after the next to last transmission of n-1.

In addition, from restriction 2, one or more of the following conditions must also be satisfied, where t_2 is the time at which n-2 was last transmitted:

i) $t \geq t_2 + 2T$

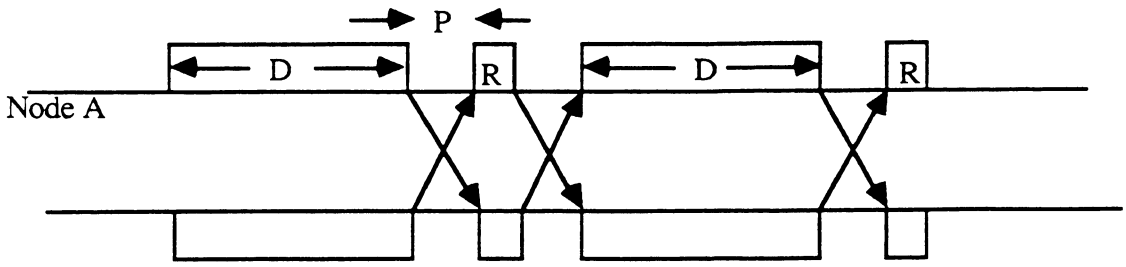
ii) The number of acks of n-2 equals the number of transmissions of n-2

iii) The last ack of n-2 is over 2T seconds after the next to last transmission of n-2.

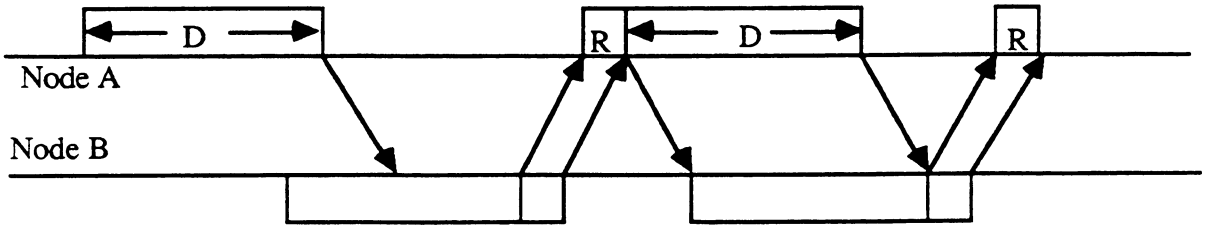
b) An algorithm must deal with the possibility of a frame that is lost (i.e., never arrives), and must successfully transmit packets after a frame is lost. If an algorithm succeeds in this case, then it must fail if a frame, regarded as lost, later arrives when a new packet of the same sequence number modulo 2 is expected.

2.38

For simplicity, look first at the case in which A and B both start at the same time.

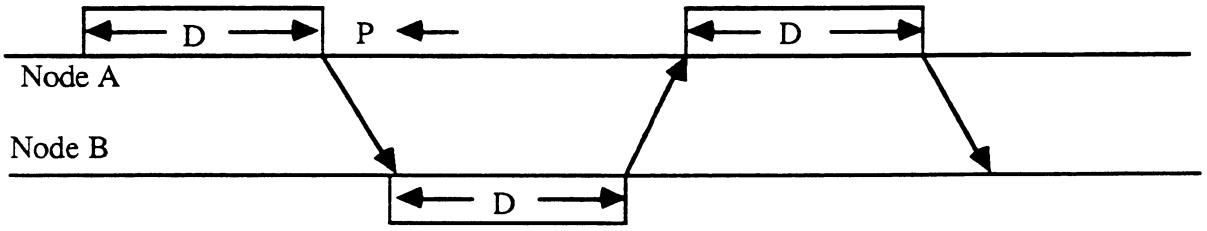


It can be seen that the above pattern is periodic with period D+R+2P, with one packet in each direction per period. Thus the rate is $(D+R+2P)^{-1}$.
Next, without loss of generality, suppose B starts its first transmission after A:



In the figure above, the pattern is periodic after the first frame in each direction. In general, if B completes its first transmission at time t and A completes its first transmission at $\tau \leq t$, then B starts its first ACK transmission at max(t, τ+P), since τ+P is the time at which B has completely received the first packet from A and t is the first time that the link is free from A to B. Node A starts to send its first ACK to B at t+P, which is thus received at B at t+R+2P. Similarly, node A receives the first ACK from B at max(t, τ+P)+R+P, at which time it starts to send its second packet.

b) The diagram below makes it clear that the two way transmission pattern is periodic with a period of 2D+2D, leading to rate $(2D+2P)^{-1}$.



2.39

a) $TC = (K+V)(j-1) + (K+V)\lceil M/K \rceil$

b) $E\{TC\} \approx (K+V)[j-1+(M/K)+1/2]$

Differentiating with respect to K and setting the result equal to 0, we get

$$K = \sqrt{\frac{MV}{j - 1/2}}$$

c) For j=1, it can be seen directly from Eq. (2.42) that TC is minimized by choosing K_{MAX} greater than the largest possible value of M (thus making all messages one packet long). The approximation in Eq. (2.43) is very poor in this case, but the solution $K_{MAX}=\infty$ in Eq. (2.44) is still valid, as seen above. For fixed length packets, the amount of fill required for very large K is prohibitive, so the approximation used in part b) above is reasonable and the resulting finite value for K is certainly reasonable.

2.40

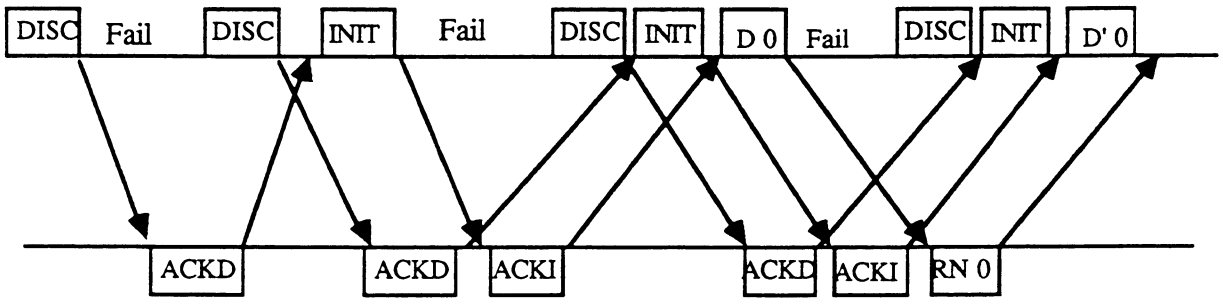
a) Using the properties of the A->B master slave protocol, B eventually receives the DISC message from A (perhaps after many attempts, using the assumption that each frame is correctly received with some probability bounded away from 0). Node B, if it has not already started to disconnect, will start to send DISC, which by the same argument is eventually received by A. Similarly B sends ACKD, which is eventually received by A (perhaps after many receptions of DISC at B and retransmissions of ACKD to A), and A regards the link as down after receiving both DISC and ACKD. Finally, when A receives DISC, it sends ACKD, which is eventually received by B, perhaps after many retransmissions of DISC from B and ACKD from A.

b) In the argument above, A regards the link as down upon receiving both DISC and ACKD, but there is no need for B to have received ACKD by this time. Thus A can start to re-initialize the link before B receives ACKD, and thus before B regards the link as down.

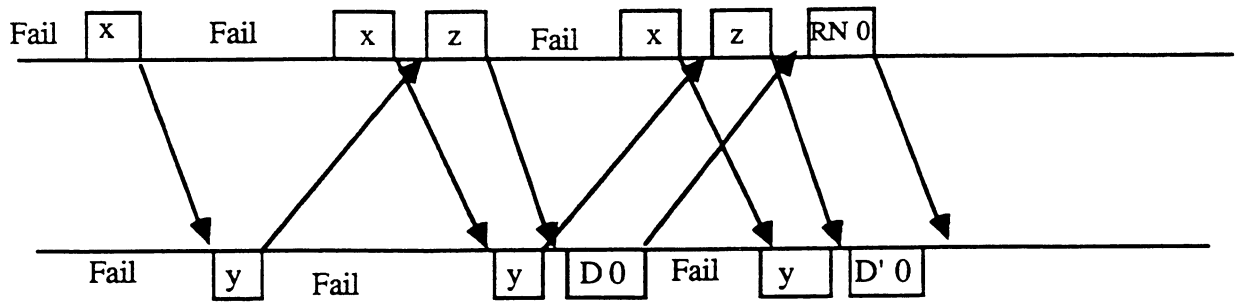
c) Note that the case being investigated here is symmetric (interchanging initialize and disconnect) to the example in part b, and thus the demonstration here shows that the example there causes no problems. Node B continues to send INIT (according to the B->A master/slave protocol) until receiving ACKI. Node A responds to each of these messages, but also sends a piggybacked ACKI when it attempts to disconnect by sending DISC. Thus node B must receive ACKI before or simultaneously with receiving DISC; in the simultaneous case, B regards the link as up instantaneously before starting to disconnect, and from this point, the scenario is the same as in part a). Note that the piggybacking is essential here, although alternate ways exist of co-ordinating the two master/slave protocols.

2.41

a)



b)



2.42

The protocol requires each node to respond to each INIT or DISC message with an ACKI or ACKD message. Thus if an additional INIT or DISC message were sent with each such ack, the protocol would continue to bounce messages back and forth forever, whereas there should be no need to continue to send INIT messages during up periods or DISC messages during down periods.

2.43

Consider integer numbering rather than numbering modulo m. Suppose packet number SN is sent at time t_1 and received at t_2. Let $SN_{min}(t_1)$ be the lower edge of the window at t_1 and $RN(t_2)$ be the lowest numbered packet not received by t_2. Because of the window, we have

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

Since $SN_{min}(t_1)$ is the greatest value of RN received up to t_1, and since RN(t) is increasing with t at node B,

$$SN_{min}(t_1) \leq RN(t_2)$$

Combining these equations, $SN \leq RN(t_2) + n - 1$. Conversely, at most M packets can be sent after packet number SN and before SN arrives at node B. Also, the packets on the link or already received at t_1 have numbers at most $SN_{min}(t_1)+n-1$. Thus the highest consecutive numbered packet received by time t_2 must be at most $SN_{min}(t_1)+n+M-1$, and $RN(t_2) \leq SN_{min}(t_1)+n+M$. Combining these relationships,

$$RN(t_2) - n - M \leq SN \leq RN(t_2) + n - 1$$

Thus the range of possible values of SN that could be received at t_2, including the end points, is 2n+M, and the modulus m must be that large to enable the sequence numbers to be properly interpreted at the receiver.

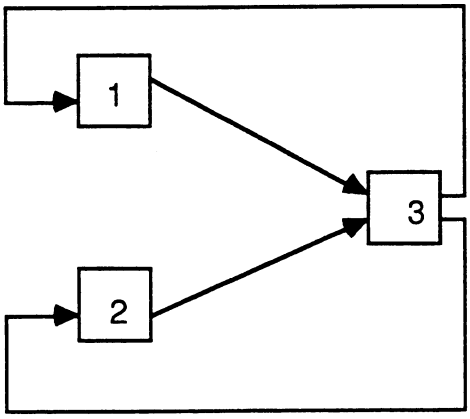
Next suppose a receive number $RN = RN(t_1)$ is sent at t_1 from B and is received at t_2 at node A. The largest possible value of $SN_{min}(t_2)$ occurs if node B receives packet RN at t_1^+ and has already received RN+1,...RN+n-1. Node B then sends RN+n as a receive number, which can be received by A by t_1^+. Node A then sends RN+n, ...RN+n+M-1 before t_2, and node B can send at most RN+n+M before t_2. Thus, $SN_{min}(t_2) \leq RN+n+M$. It follows that $SN_{max}(t_2) \leq RN+2n+M-1$. Thus, $m \geq 2n+M$ guarantees that RN, arriving at t_2, will not be falsely interpreted as a request for $SN_{max}(t_2)$.

CHAPTER 3 SOLUTIONS

3.1

A customer that carries out the order (eats in the restaurant) stays for 5 mins (25 mins). Therefore the average customer time in the system is T = 0.5*5 + 0.5*25 = 15. By Little's Theorem the average number in the system is N = λ*T = 5*15=75.

3.2



We represent the system as shown in the figure. The number of files in the entire system is exactly one at all times. The average number in node i is $\lambda_i R_i$ and the average number in node 3 is $\lambda_1 P_1 + \lambda_2 P_2$. Therefore the throughput pairs (λ_1, λ_2) must satisfy (in addition to nonnegativity) the constraint

$$\lambda_1(R_1 + P_1) + \lambda_2(R_2 + P_2) = 1.$$

If the system were slightly different and queueing were allowed at node 3, while nodes 1 and 2 could transmit at will, a different analysis would apply. The transmission bottleneck for the files of node 1 implies that

$$\lambda_1 \leq \frac{1}{R_1}$$

Similarly for node 2 we get that

$$\lambda_2 \leq \frac{1}{R_2}$$

Node 3 can work on only one file at a time. If we look at the file receiving service at node 3 as a system and let N be the average number receiving service at node 3, we conclude from Little's theorem that

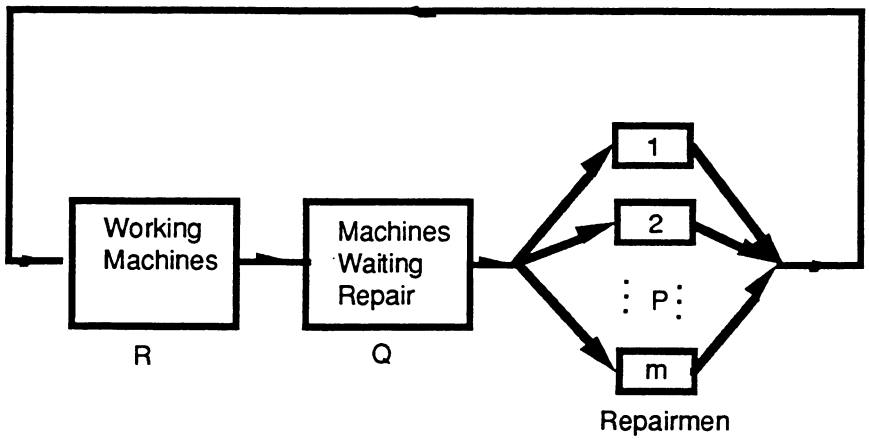
$$\lambda_1 P_1 + \lambda_2 P_2 = N$$

and $N \leq 1$

This implies that

$$\lambda_1 P_1 + \lambda_2 P_2 \leq 1$$

3.3



We represent the system as shown in the figure. In particular, once a machine breaks down, it goes into repair if a repairperson is available at the time, and otherwise waits in a queue for a repairperson to become free. Note that if m=1 this system is identical to the one of Example 3.7.

Let λ be the throughput of the system and let Q be the average time a broken down machine waits for a repairperson to become free. Applying Little's theorem to the entire system, we obtain

$$\lambda(R+Q+P) = N \qquad (1)$$

from which

$$\lambda(R+P) \leq N \qquad (2)$$

Since the number of machines waiting repair can be at most (N-m), the average waiting time λQ is at most the average time to repair (N-m) machines, which is (N-m)P. Thus, from Eq. (1) we obtain

$$\lambda(R+ (N - m)P + P) \geq N \qquad (3)$$

Applying Little's theorem to the repairpersons, we obtain

$$\lambda P \leq m \qquad (4)$$

The relations (2)-(4) give the following bounds for the throughput λ

$$\frac{N}{R + (N - m + 1)P} \leq \lambda \leq \min\left\{\frac{m}{P}, \frac{N}{R + P}\right\} \tag{5}$$

Note that these bounds generalize the ones obtained n Example 3.7 (see Eq. (3.9)).
By using the equation $T=N/\lambda$ for the average time between repairs, we obtain from Eq. (5)

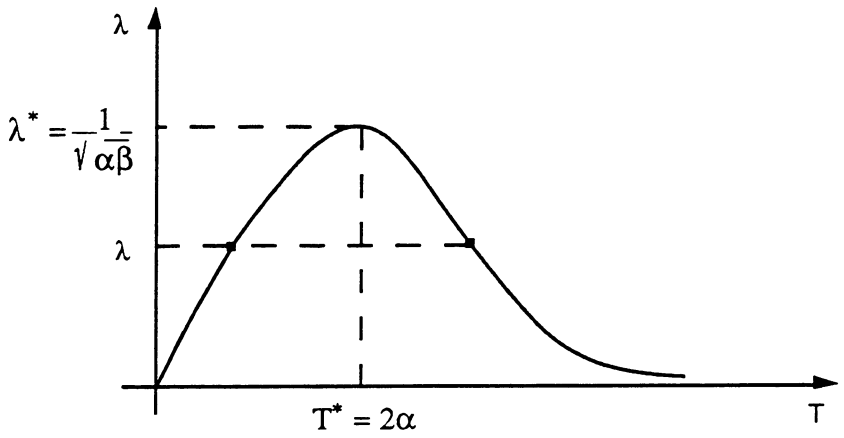
$$\min\{NP/m, R + P\} \leq T \leq R + (N - m + 1)P$$

3.4

If λ is the throughput of the system, Little's theorem gives $N = \lambda T$, so from the relation $T= \alpha + \beta N^2$ we obtain $T = \alpha + \beta\lambda^2 T^2$ or

$$\lambda = \sqrt{\frac{T - \alpha}{\beta T^2}} \tag{1}$$

This relation betweeen λ ands T is plotted below.



The maximum value of λ is attained for the value T^* for which the derivative of $(T - \alpha)/\beta T^2$ is zero (or $1/(\beta T^2) - 2(T - \alpha)/(\beta T^3) = 0$). This yields $T^* = 2\alpha$ and from Eq. (1), the corresponding maximal throughput value

$$\lambda^* = \frac{1}{\sqrt{\alpha\beta}} \tag{2}$$

(b) When $\lambda < \lambda^*$, there are two corresponding values of T: a low value corresponding to an uncongested system where N is relatively low, and a high value corresponding to a congested system where N is relatively high. This assumes that the system reaches a steady-state. However, it can be argued that when the system is congested a small increase in the number of cars in the system due to statistical fluctuations will cause an increase in the time in the system, which will tend to decrease the rate of departure of cars from the system. This will cause a further increase in the number in the system and a further increase in the time in the system, etc. In other words, when we are operating on the right side of

the curve of the figure, there is a tendency for *instability* in the system, whereby a steady-state is never reached: the system tends to drift towards a traffic jam where the car depature rate from the system tends towards zero and the time a car spends in the system tends towards infinity. Phenomena of this type are analyzed in the context of the Aloha multiaccess system in Chapter 4.

3.5

The expected time in question equals

$$\begin{aligned} E\{\text{Time}\} = (5 + E\{\text{stay of 2nd student}\})*P\{\text{1st stays less or equal to 5 minutes}\} \\ + (E\{\text{stay of 1st} \mid \text{stay of 1st} \geq 5\} + E\{\text{stay of 2nd}\})* \\ P\{\text{1st stays more than 5 minutes}\}. \end{aligned}$$

We have $E\{\text{stay of 2nd student}\} = 30$, and, using the memoryless property of the exponential distribution,

$$E\{\text{stay of 1st} \mid \text{stay of 1st} \geq 5\} = 5 + E\{\text{stay of 1st}\} = 35.$$

Also

$$\begin{aligned} P\{\text{1st student stays less or equal to 5 minutes}\} &= 1 - e^{-5/30} \\ P\{\text{1st student stays more than 5 minutes}\} &= e^{-5/30}. \end{aligned}$$

By substitution we obtain

$$E\{\text{Time}\} = (5 + 30)*(1 - e^{-5/30}) + (35 + 30)* e^{-5/30} = 35 + 30*e^{-5/30} = 60.394.$$

3.6

(a) The probability that the person will be the last to leave is 1/4 because the exponential distribution is memoryless, and all customers have identical service time distribution. In particular, at the instant the customer enters service, the remaining service time of each of the other three customers served has the same distribution as the service time of the customer.

(b) The average time in the bank is 1 (the average customer service time) plus the expected time for the first customer to finish service. The latter time is 1/4 since the departure process is statistically identical to that of a single server facility with 4 times larger service rate. More precisely we have

$$\begin{aligned} P\{\text{no customer departs in the next t mins}\} &= P\{\text{1st does not depart in next t mins}\} \\ &\quad * P\{\text{2nd does not depart in next t mins}\} \\ &\quad * P\{\text{3rd does not depart in next t mins}\} \\ &\quad * P\{\text{4th does not depart in next t mins}\} \\ &= (e^{-t})^4 = e^{-4t}. \end{aligned}$$

Therefore

$$P\{\text{the first departure occurs within the next t mins}\} = 1 - e^{-4t},$$

and the expected time to the next depature is 1/4. So the answer is 5/4 minutes.

(c) The answer will not change because the situation at the instant when the customer begins service will be the same under the conditions for (a) and the conditions for (c).

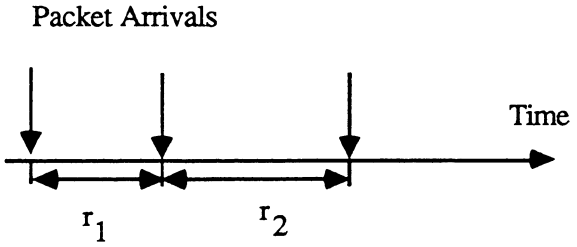
3.7

In the statistical multiplexing case the packets of at most one of the streams will wait upon arrival for a packet of the other stream to finish transmission. Let W be the waiting time , and note that $0 \leq W \leq T/2$. We have that one half of the packets have system time $T/2 + W$ and waiting time in queue W. Therefore

$$\text{Average System Time} = (1/2)T/2 + (1/2)(T/2+W) = (T+W)/2$$
$$\text{Average Waiting Time in Queue} = W/2$$
$$\text{Variance of Waiting Time} = (1/2)(W/2)^2+(1/2)(W/2)^2 = W^2/4.$$

So the average system time is between T/2 and 3T/4 and the variance of waiting time is between 0 and $T^2/16$.

3.8



Fix a packet. Let r_1 and r_2 be the interarrival times between a packet and its immediate predecessor, and successor respectively as shown in the figure above. Let X_1 and X_2 be the lengths of the predecessor packet, and of the packet itself respectively. We have:

$$P\{\text{No collision w/ predecessor or successor}\} = P\{r_1 > X_1, r_2 > X_2\} = P\{r_1 > X_1\}P\{r_2 > X_2\}.$$

$$P\{\text{No collision with any other packet}\} = P_1\, P\{r_2 > X_2\}$$

where

$$P_1 = P\{\text{No collision with all preceding packets}\}.$$

(a) For fixed packet lengths (= 20 msec)

$$P\{r_1 > X_1\} = P\{r_2 > X_2\} = e^{-\lambda*20} = e^{-0.01*20} = e^{-0.2}$$
$$P_1 = P\{r_1 \leq X_1\}.$$

Therefore the two probabilities of collision are both equal to $e^{-0.4} = 0.67$.

(b) For X exponentially distributed packet length with mean $1/\mu$ we have

$$P\{r_1 > X_1\} = P\{r_2 > X_2\} = \int_0^\infty P\{r_1 > X \mid X_1 = X\} p\{X_1 = X\} dX$$

$$= \int_0^\infty e^{-\lambda X} \mu e^{-\mu X} dX = \frac{\mu}{\lambda + \mu}$$

Substituting $\lambda = 0.01$ and $\mu = 0.05$ we obtain $P\{r_1 > X_1\} = P\{r_2 > X_2\} = 5/6$, and

$$P\{\text{No collision w/ predecessor or successor}\} = (5/6)^2 = 0.694.$$

Also P_1 is seen to be the steady-state probability of a customer finding an empty system in the M/M/∞ system with arrival and service rate λ and μ respectively. Therefore $P_1 = e^{-\lambda/\mu} = e^{-0.2}$. Therefore

$$P\{\text{No collision with any other packet}\} = e^{-0.2} 5/6 = 0.682.$$

3.9

(a) For each session the arrival rate is $\lambda = 150/60 = 2.5$ packets/sec. When the line is divided into 10 lines of capacity 5 Kbits/sec, the average packet transmission time is $1/\mu = 0.2$ secs. The corresponding utilization factor is $\rho = \lambda/\mu = 0.5$. We have for each session $N_Q = \rho^2/(1 - \rho) = 0.5$, $N = \rho/(1 - \rho) = 1$, and $T = N/\lambda = 0.4$ secs. For all sessions collectively N_Q and N must be multiplied by 10 to give $N_Q = 5$ and $N = 10$.

When statistical multiplexing is used, all sessions are merged into a single session with 10 times larger λ and μ; $\lambda = 25$, $1/\mu = 0.02$. We obtain $\rho = 0.5$, $N_Q = 0.5$, $N = 1$, and $T = 0.04$ secs. Therefore N_Q, N, and T have been reduced by a factor of 10 over the TDM case.

(b) For the sessions transmitting at 250 packets/min we have $\rho = (250/60)*0.2 = 0.833$ and we have $N_Q = (0.833)^2/(1 - 0.833) = 4.158$, $N = 5$, $T = N/\lambda = 5/(250/60) = 1.197$ secs. For the sessions transmitting at 50 packets/min we have $\rho = (50/60)*0.2 = 0.166$, $N_Q = 0.033$, $N = 0.199$, $T = 0.199/(50/60) = 0.239$.

The corresponding averages over all sessions are $N_Q = 5*(4.158 + 0.033) = 21$, $N = 5*(5 + 0.199) = 26$, $T = N/\lambda = N/(5*\lambda_1 + 5*\lambda_2) = 26/(5*(250/60)+5*(50/60)) = 1.038$ secs.

When statistical multiplexing is used the arrival rate of the combined session is $5*(250 + 50) = 1500$ packets/sec and the same values for N_Q, N, and T as in (a) are obtained.

3.10

(a) Let t_n be the time of the nth arrival, and $\tau_n = t_{n+1} - t_n$. We have for $s \geq 0$

$$P\{\tau_n > s\} = P\{A(t_n + s) - A(t_n) = 0\} = e^{-\lambda s}$$

(by the Poisson distribution of arrivals in an interval). So

$$P\{\tau_n \leq s\} = 1 - e^{-\lambda s}$$

which is (3.11).

To show that $\tau_1, \tau_2, \ldots$ are independent, note that (using the independence of the numbers of arrivals in disjoint intervals)

$$\begin{aligned} P\{\tau_2 > s \mid \tau_1 = \tau\} &= P\{0 \text{ arrivals in } (\tau, \tau+s] \mid \tau_1 = \tau\} \\ &= P\{0 \text{ arrivals in } (\tau, \tau+s]\} = e^{-\lambda s} = P\{\tau_2 > s\} \end{aligned}$$

Therefore τ_2 and τ_1 are independent.

To verify (3.12), we observe that

$$P\{A(t + \delta) - A(t) = 0\} = e^{-\lambda\delta}$$

so (3.12) will be shown if

$$\lim_{\delta \to 0} (e^{-\lambda\delta} - 1 + \lambda\delta)/\delta = 0$$

Indeed, using L'Hospital's rule we have

$$\lim_{\delta \to 0} (e^{-\lambda\delta} - 1 + \lambda\delta)/\delta = \lim_{\delta \to 0} (-\lambda e^{-\lambda\delta} + \lambda) = 0$$

To verify (3.13) we note that

$$P\{A(t + \delta) - A(t) = 1\} = \lambda\delta e^{-\lambda\delta}$$

so (3.13) will be shown if

$$\lim_{\delta \to 0} (\lambda\delta e^{-\lambda\delta} - \lambda\delta)/\delta = 0$$

This is equivalent to

$$\lim_{\delta \to 0} (\lambda e^{-\lambda\delta} - \lambda) = 0$$

which is clearly true.

To verify (3.14) we note that

$$P\{A(t + \delta) - A(t) \geq 2\} = 1 - P\{A(t + \delta) - A(t) = 0\} - P\{A(t + \delta) - A(t) = 1\}$$

$$= 1 - (1 - \lambda\delta + o(\delta)) - (\lambda\delta + o(\delta)) = o(\delta)$$

(b) Let N_1, N_2 be the number of arrivals in two disjoint intervals of lengths τ_1 and τ_2. Then

$$\begin{aligned} P\{N_1+N_2 = n\} &= \Sigma^n_{k=0} P\{N_1 = k, N_2 = n-k\} = \Sigma^n_{k=0} P\{N_1 = k\}P\{N_2 = n-k\} \\ &= \Sigma^n_{k=0} e^{-\lambda\tau 1}[(\lambda\tau_1)^k/k!]e^{-\lambda\tau 2}[(\lambda\tau_2)^{(n-k)}/(n-k)!] \\ &= e^{-\lambda(\tau 1 + \tau 2)}\Sigma^n_{k=0}[(\lambda\tau_1)^k(\lambda\tau_2)^{(n-k)}]/[k!(n-k)!] \\ &= e^{-\lambda(\tau 1 + \tau 2)}[(\lambda\tau_1 + \lambda\tau_2)^n/n!] \end{aligned}$$

(The identity

$$\Sigma^n_{k=0}[a^k b^{(n-k)}]/[k!(n-k)!] = (a + b)^n/n!$$

can be shown by induction.)

(c) The number of arrivals of the combined process in disjoint intervals is clearly independent, so we need to show that the number of arrivals in an interval is Poisson distributed, i.e.

$$\begin{aligned} &P\{A_1(t + \tau) + \ldots + A_k(t + \tau) - A_1(t) - \ldots - A_k(t) = n\} \\ &\qquad = e^{-(\lambda 1 + \ldots + \lambda k)\tau}[(\lambda_1 + \ldots + \lambda_k)\tau]^n/n! \end{aligned}$$

For simplicity let k=2; a similar proof applies for k > 2. Then

$$\begin{aligned} &P\{A_1(t + \tau) + A_2(t + \tau) - A_1(t) - A_2(t) = n\} \\ &\quad = \Sigma^n_{m=0} P\{A_1(t + \tau) - A_1(t) = m, A_2(t + \tau) - A_2(t) = n-m\} \\ &\quad = \Sigma^n_{m=0} P\{A_1(t + \tau) - A_1(t) = m\}P\{A_2(t + \tau) - A_2(t) = n-m\} \end{aligned}$$

and the calculation continues as in part (b). Also

$$\begin{aligned} &P\{1 \text{ arrival from } A_1 \text{ prior to } t \mid 1 \text{ occured}\} \\ &\quad = P\{1 \text{ arrival from } A_1, 0 \text{ from } A_2\}/P\{1 \text{ occured}\} \\ &\quad = (\lambda_1 t e^{-\lambda 1 t} e^{-\lambda 2 t})/(\lambda t e^{-\lambda t}) = \lambda_1/\lambda \end{aligned}$$

(d) Let t be the time of arrival. We have

$$\begin{aligned} &P\{t < s \mid 1 \text{ arrival occured}\} = P\{t < s, 1 \text{ arrival occured}\}/P\{1 \text{ arrival occured}\} \\ &= P\{1 \text{ arrival occured in } [t_1, s), 0 \text{ arrivals occured in } [s, t_2]\}/P\{1 \text{ arrival occured}\} \\ &= (\lambda(s - t_1)e^{-\lambda(s - t1)}e^{-\lambda(s - t2)})/(\lambda(t_2 - t_1)e^{-\lambda(t2 - t1)}) = (s - t_1)/(t_2 - t_1) \end{aligned}$$

This shows that the arrival time t is uniformly distributed in $[t_1, t_2]$.

(a) Let us call the two transmission lines 1 and 2, and let $N_1(t)$ and $N_2(t)$ denote the respective numbers of packet arrivals in the interval [0,t]. Let also $N(t) = N_1(t) + N_2(t)$. We calculate the joint probability $P\{N_1(t) = n, N_2(t) = m\}$. To do this we first condition on N(t) to obtain

$$P\{N_1(t) = n, N_2(t) = m\} = \Sigma_{k=0}^{\infty} P\{N_1(t) = n, N_2(t) = m \mid N(t) = k\}P\{N(t) = k\}.$$

Since

$$P\{N_1(t) = n, N_2(t) = m \mid N(t) = k\} = 0 \qquad \text{when} \quad k \neq n+m$$

we obtain

$$\begin{aligned} P\{N_1(t) = n, N_2(t) = m\} &= P\{N_1(t) = n, N_2(t) = m \mid N(t) = n + m\}P\{N(t) = n + m\} \\ &= P\{N_1(t) = n, N_2(t) = m \mid N(t) = n + m\}e^{-\lambda t}[(\lambda t)^{n+m}/(n + m)!] \end{aligned}$$

However, given that n+m arrivals occurred, since each arrival has probability p of being a line 1 arrival and probability 1-p of being a line 2 arrival, it follows that the probability that n of them will be line 1 and m of them will be line 2 arrivals is the binomial probability

$$\binom{n+m}{n} p^n (1-p)^m$$

Thus

$$P\{N_1(t) = n, N_2(t) = m\} = \binom{n+m}{n} p^n (1-p)^m e^{-\lambda t} \frac{(\lambda t)^{n+m}}{(n+m)!}$$

$$= e^{-\lambda tp} \frac{(\lambda tp)^n}{n!} \quad e^{-\lambda t(1-p)} \frac{(\lambda t(1-p))^m}{m!} \qquad (1)$$

Hence

$$P\{N_1(t) = n\} = \sum_{m=0}^{\infty} P\{N_1(t) = n, N_2(t) = m\}$$

$$= e^{-\lambda tp} \frac{(\lambda tp)^n}{(n)!} \sum_{m=0}^{\infty} e^{-\lambda t(1-p)} \frac{(\lambda t(1-p))^m}{m!}$$

$$= e^{-\lambda tp} \frac{(\lambda tp)^n}{(n)!}$$

That is, $\{N_1(t), t \geq 0\}$ is a Poisson process having rate λp. Similarly we argue that $\{N_2(t), t \geq 0\}$ is a Poisson process having rate $\lambda(1 - p)$. Finally from Eq. (1) it follows that the two processes are independent since the joint distribution factors into the marginal distributions.

(b) Let A, A_1, and A_2 be as in the hint. Let I be an interarrival interval of A_2 and consider the number of arrivals of A_1 that lie in I. The probability that this number is n is the probability of n successive arrivals of A_1 followed by an arrival of A_2, which is $\rho^n(1 - \rho)$. This is also the probability that a customer finds upon arrival n other customers waiting in an M/M/1 queue. The service time of each of these customers is exponentially distributed with parameter μ, just like the interarrival times of process A. Therefore the waiting time of the customer in the M/M/1 system has the same distribution as the interarrival time of process A_2. Since by part (a), the process A_2 is Poisson with rate $\mu - \lambda$, it follows that the waiting time of the customer in the M/M/1 system is exponentially distributed with parameter $\mu - \lambda$.

3.12

For any scalar s we have using also the independence of τ_1 and τ_2

$$P(\min\{\tau_1,\tau_2\} \geq s) = P(\tau_1 \geq s, \tau_2 \geq s) = P(\tau_1 \geq s)\, P(\tau_2 \geq s)$$

$$= e^{-\lambda_1 s} e^{-\lambda_2 s} = e^{-(\lambda_1 + \lambda_2)s}$$

Therefore the distribution of $\min\{\tau_1,\tau_2\}$ is exponential with mean $1/(\lambda_1 + \lambda_2)$.

By viewing τ_1 and τ_2 as the arrival times of the first arrivals from two independent Poisson processes fwith rates λ_1 and λ_2, we see that the equation $P(\tau_1 < \tau_2) = \lambda_1/(\lambda_1 + \lambda_2)$ follows from Problem 3.10(c).

Consider the M/M/1 queue and the amount of time spent in a state k>0 between transition into the state and transition out of the state. This time is $\min\{\tau_1,\tau_2\}$, where τ_1 is the time between entry to the state k and the next customer arrival and τ_2 is the time between entry to the state k and the next service completion. Because of the memoryless property of the exponential distribution, τ_1 and τ_2 are exponentially distributed with means $1/\lambda$ and $1/\mu$, respectively. It follows using the fact shown above that the time between entry and exit from stae k is exponentially distributed with mean $1/(\lambda+\mu)$. The probability that the transition will be from k to k+1 is $\lambda/(\lambda+\mu)$ and that the transition will be from k to k-1 is $\mu/(\lambda+\mu)$. For state 0 the amount of time spent is exponentially distributed with mean $1/\lambda$ and the probability of a transition to state 1 is 1. Because of this it can be seen that M/M/1 queue can be described as a continuous Markov chain with the given properties.

3.13

(a) Consider a Markov chain with state

n = Number of people waiting + number of empty taxi positions

Then the state goes from n to n+1 each time a person arrives and goes from n to n-1 (if $n \geq 1$) when a taxi arrives. Thus the system behaves like an M/M/1 queue with arrival rate 1 per min and departure rate 2 per min. Therefore the occupancy distribution is

$$p_n=(1-\rho)/\rho^n$$

where $\rho=1/2$. State n, for $0 \le n \le 4$ corresponds to 5, 4, 3, 2, 1 taxis waiting while $n > 4$ corresponds to no taxi waiting. Therefore

P{5 taxis waiting} = 1/2
P{4 taxis waiting} = 1/4
P{3 taxis waiting} = 1/8
P{2 taxis waiting} = 1/16
P{1 taxi waiting} = 1/32

and P{no taxi waiting} is obtained by subtracting the sum of the probabilities above from unity. This gives P{no taxi waiting} = 1/32.

(b) See the hint.

(c) This system corresponds to taxis arriving periodically instead of arriving according to a Poisson process. It is the slotted M/D/1 system analyzed in Section 6.3.

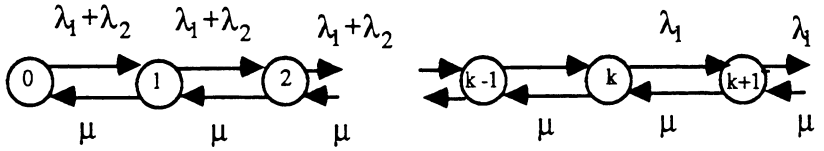
3.14

(a) The average message transmission time is $1/\mu = L/C$ so the service rate is $\mu = C/L$. When the number of packets in the system is larger than K, the arrival rate is λ_1. We must have

$$0 \le \lambda_1 < \mu$$
$$0 \le \lambda_2$$

in order for the arrival rate at node A to be less than the service rate for large state values. For these values, therefore, the average number of packets in the system will stay bounded.

(b) The corresponding Markov chain is as shown in the figure below. The steady-state probabilities satisfy



$$p_n = \rho^n p_0 \qquad \text{for } n \le k$$
$$p_n = \rho_1{}^{n-k}\rho^k p_0 \qquad \text{for } n > k$$

where $\rho = (\lambda_1 + \lambda_2)/\mu$, $\rho_1 = \lambda_1/\mu$. We have

$$\Sigma^{\infty}_{n=0}\, p_n = 1$$

or

$$p_0\Sigma^{k}_{n=0}\rho^n + \Sigma^{\infty}_{n=k+1}\rho_1{}^{n-k}\rho^k = 1$$

from which we obtain after some calculation

$$p_0 = [(1 - \rho)(1 - \rho_1)]/[1 - \rho_1 - \rho^k(\rho - \rho_1)] \qquad \text{for } \rho < 1$$

and

$$p_0 = (1 - \rho_1)/[1 + k(1 - \rho_1)] \qquad \text{for } \rho = 1$$

For packets of source 1 the average time in A is

$$T_1 = (1/\mu)(1 + N)$$

where

$$N = \Sigma^{\infty}_{n=0}\, np_n$$

is the average number in the system upon arrival. The average number in A from source 1 is

$$N_1 = \lambda_1 T_1$$

For packets of source 2 the average time in A is

$$T_2 = (1/\mu)(1 + N')$$

where

$$N' = \frac{\sum_{n=0}^{k-1} np_n}{\sum_{n=0}^{k-1} p_n}$$

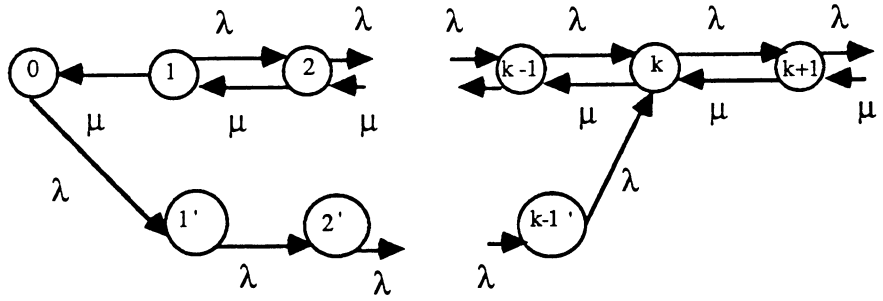
is the average number in the system found by an accepted packet of source 2. To find the average number in the system from source 2 we must find the arrival rate into node A of packets from source 2. This is

$$\lambda'_2 = \lambda_2 P\{\text{arriving packet from source 2 is accepted}\} = \lambda_2 \Sigma^{k-1}_{n=0}\, p_n$$

and the average number from source 2 in A is

$$N_2 = \lambda'_2 T_2$$

3.15



The transition diagram of the corresponding Markov chain is shown in the figure. We have introduced states 1', 2', . . ., (k-1)' corresponding to situations where there are customers in the system waiting for service to begin again after the system has emptied out. Using global balance equations between the set of states {1', 2', . . . ,i') and all other states, for i' = 1', . . . , (k-1)', we obtain $\lambda p_0 = \lambda p_{1'} = \lambda p_{2'} = \ldots = \lambda p_{(k-1)'}$, so

$$p_0 = p_{1'} = p_{2'} = \ldots = p_{(k-1)'}$$

Also by using global balance equations we have

$$\mu p_1 = \lambda p_0$$
$$\mu p_2 = \lambda(p_1 + p_{1'}) = \lambda(p_1 + p_0)$$
$$\ldots\ldots\ldots\ldots$$
$$\mu p_k = \lambda(p_{k-1} + p_{(k-1)'}) = \lambda(p_{k-1} + p_0)$$
$$\mu p_{i+1} = \lambda p_i \qquad i \geq k.$$

By denoting $\rho = \lambda/\mu$ we obtain

$$p_i = \rho(1 + \rho + \ldots + \rho^{i-1})p_0 \qquad 1 \leq i \leq k$$

$$p_i = \rho^{1+i-k}(1 + \rho + \ldots + \rho^{k-1})p_0 \qquad i > k.$$

Substituting these expressions in the equation $p_{1'} + \ldots + p_{(k-1)'} + p_0 + p_1 + \ldots = 1$ we obtain p_0

$$p_0\left(k + \sum_{i=1}^{k} \frac{\rho(1-\rho^i)}{1-\rho} + \frac{1-\rho^k}{1-\rho}\rho^2(1+\rho+\ldots)\right) = 1$$

$$p_0 = \left(k + \frac{\rho}{1-\rho}\sum_{i=1}^{k}(1-\rho^i) + \frac{\rho^2}{(1-\rho)^2}(1-\rho^k)\right)^{-1}$$

After some calculation this gives $p_0 = (1 - \rho)/k$ (An alternative way to verify this formula is to observe that the fraction of time the server is busy is equal to ρ by Little's theorem). Therefore, the fraction of time the server is idle is $(1 - \rho)$. When this is divided among the k

equiprobable states 0, 1', . . ., (k-1)' we obtain $p_0 = (1 - \rho)/k$. The average number in the system is

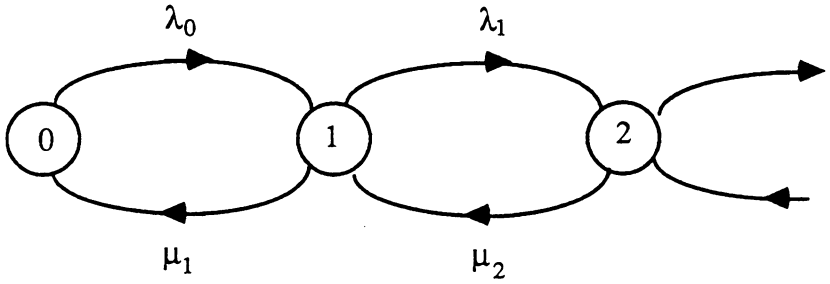
$$N = p_{1'} + 2p_{2'} + \ldots + (k-1)p_{(k-1)'} + \sum_{i=0}^{\infty} jp_i = p_0 \frac{k(k-1)}{2} + \sum_{i=0}^{\infty} ip_i$$

where the probabilities p_i are given in the equations above. After some calculation this yields

$$N = (k-1)/2 + \rho/(1 - \rho).$$

The average time in the system is (by Little's Theorem) $T = N/\lambda$.

3.16



The figure shows the Markov chain corresponding to the given system. The local balance equation for it can be written down as :

$$\rho_0 p_0 = p_1$$

$$\rho_1 p_1 = p_2$$

$$\ldots \qquad \ldots$$

$$\Rightarrow p_{n+1} = \rho_n p_n = \rho_{n-1}\rho_n p_{n-1} = \ldots = (\rho_0 \rho_1 \cdots \rho_n) p_0$$

but,

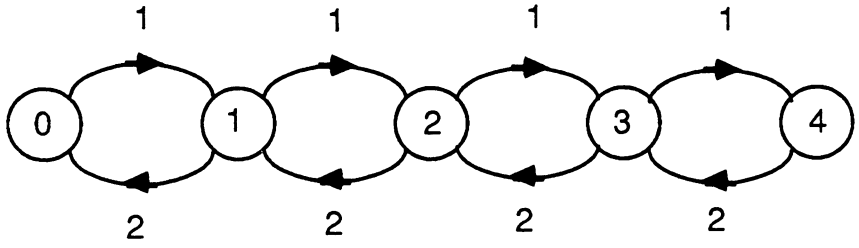
$$\sum_{i=0}^{\infty} p_i = p_0(1 + \rho_0 + \rho_0\rho_1 + \ldots.) = 1$$

$$\Rightarrow p_0 = \left[1 + \sum_{k=0}^{\infty} \left(\rho_0 \cdots \rho_k \right) \right]^{-1}$$

3.17

The discrete time version of the M/M/1 system can be characterized by the same Markov chain as the continuous time M/M/1 system and hence will have the same occupancy distribution.

3.18



$$p_1 = \frac{1}{2} p_0$$

$$p_n = \frac{1}{2} p_{n-1} \quad \text{for } 1 \le n \le 4$$

$$\sum_{i=0}^{4} p_i = 1$$

Solving the above equations we get,

$$p_n = \frac{2^{4-n}}{31} \quad \text{for } 0 \le n \le 4$$

$$N = \sum_{n=0}^{4} n\, p_n = \frac{26}{31}$$

P(a customer arrives but has to leave) = 1/31

Hence the arrival rate of passengers who join the queue =

$$(1-p_4)\,\lambda = \frac{30}{31} \text{ per minute } = \lambda_a \text{ (say)}$$

$$T = N/\lambda_a = \frac{26/31}{30/31} = \frac{13}{15} \text{ minutes}$$

3.19

We have here an M/M/m/m system where m is the number of circuits provided by the company. Therefore we must find the smallest m for which $p_m < 0.01$ where p_m is given by the Erlang B formula

$$p_m = \frac{(\lambda/\mu)^m/m!}{\sum_{n=0}^{m}(\lambda/\mu)^n/n!}.$$

We have $\lambda = 30$ and $\mu = 1/3$, so $\lambda/\mu = 30 \cdot 3 = 90$. By substitution in the equation above we can calculate the required value of m.

3.20

We view this as an M/M/m problem. We have

$\lambda=0.5$, $E(X) = 1/\mu = 3$, m=? so that $W<0.5$

We know that the utilization factor has to be less than 1 or m has to be greater than or equal to 2. By the M/M/n results we have

$$W = \frac{\frac{\lambda}{m\mu} P_Q}{\lambda\left(1-\frac{\lambda}{m\mu}\right)} = \frac{P_Q}{m\mu-\lambda}$$

$$\text{where } P_Q = \frac{p_0 \left(\frac{\lambda}{\mu}\right)^m}{m!\left(1-\frac{\lambda}{m\mu}\right)}$$

$$\text{and} \quad p_0 = \left[\sum_{n=0}^{m-1}\frac{(\lambda/\mu)^n}{n!} + \frac{(\lambda/\mu)^m}{m!(1-\lambda/\mu)}\right]^{-1}$$

As can be seen from the expressions above m should be at most 5 because at m=5 , W is less than 0.5 because P_Q is less than 1.

The following C program calculates the optimum m.

```
double P0(lambda,mu,m){
        mrho = lambda/mu;
        rho = mrho/m;
        for(n=0; n<m; n++)
                temp1 = pow(mrho,n)/ fact(n);
```

```
        temp2 = pow(mrho,m)/(fact(m)*(1-rho));
        return(1/( temp1 + temp2 )); /* this returns p0 */
}

int fact(n){
        if (n==0) return (1);
        else
                return(n* fact (n-1));
}

double W(lambda,mu,m){
        PQ = P0(lambda,mu,m) * pow(mrho,m) /
                                                (fact(m) * (1-rho));
return(PQ/(m *mu - lambda));
}       /* this returns W for a given m */

main() {
        lambda = 0.5; mu = 0.333; previous_W = 100.0;
        for(m=2; m<=5; m++)
                if ((temp = W(lambda,mu,m)) < Previous_W)
                        previous_W = temp;
                else
                        { print(m-1);
                        break;
                }
}
```

3.21

We have $p_n = \rho^n p_0$ where $\rho = \lambda/\mu$. Using the relation

$$\sum_{n=0}^{m} p_n = 1$$

we obtain

$$p_0 = \frac{1}{\sum_{n=0}^{m} \rho^n} = \frac{1 - \rho}{1 - \rho^{m+1}}$$

Thus

$$p_n = \frac{\rho^n(1 - \rho)}{1 - \rho^{m+1}}, \qquad 0 \le n \le m$$

3.22

(a) When all the courts are busy, the expected time between two departures is 40/5 = 8 minutes. If a pair sees k pairs waiting in the queue, there must be exactly k+1 departures from the system before they get a court. Since all the courts would be busy during this whole time, the average waing time required before k+1 departures is 8(k+1) minutes.

(b) Let X be the expected waiting time given that the courts are found busy. We have

$$\lambda = 1/10, \qquad \mu = 1/40, \qquad \rho = \lambda/(5\mu) = 0.8$$

and by the M/M/m results

$$W = \frac{\rho P_Q}{\lambda(1 - \rho)}$$

Since $W = XP_Q$, we obtain $X = W/P_Q = \rho/[\lambda(1 - \rho)] = 40$ min.

3.23

Let

$$p_m = P\{\text{the 1st m servers are busy}\}$$

as given by the Erlang B formula. Denote

r_m = Arrival rate to servers (m+1) and above
λ_m = Arrival rate to server m.

We have

$$r_m = p_m\lambda$$
$$\lambda_m = r_{m-1} - r_m = (p_{m-1} - p_m)\lambda.$$

The fraction of time server m is busy is

$$b_m = \lambda_m/\mu.$$

3.24

We will show that the system is described by a Markov chain that is identical to the M/lM/1 chain. For small δ we have

$$P\{k \text{ arrivals and } j \text{ departures}\} = 0(\delta) \qquad \text{if } k + j \geq 2$$

$$P\{0 \text{ arrivals and } 1 \text{ departure} \mid \text{starting state} = i \geq i\}$$
$$= P\{0 \text{ arrivals} \mid \text{starting state } i \geq 1\} \bullet P\{1 \text{ departure} \mid \text{starting state } i \geq 1\}$$

We have

$$P\{\ 0 \text{ arrivals} \mid \text{starting state } i \geq 1\} = P\{0 \text{ arrivals}\} = 1 - \lambda\delta + 0(\delta).$$

The probability $P\{1 \text{ departure} \mid \text{starting state } i > 1\}$ is obtained from the binomial distribution or sum of i Bernoulli trials, each with a probability of success equal to $(\mu/i)\,\delta + 0(\delta)$. We need the probability of one success, which is

$$\binom{i}{1}(1- (\mu/i)\,\delta + 0(\delta))^{i-1}\,((\mu/i)\,\delta + 0(\delta))$$

Therefore

$$P\{\ 0 \text{ arrivals and } 1 \text{ departure} \mid \text{starting state} = i \geq 1\}$$
$$= \binom{i}{1}(1- (\mu/i)\,\delta + 0(\delta))^{i-1}\,((\mu/i)\,\delta + 0(\delta)) \bullet (1-\lambda\delta + 0(\delta)) = \mu\delta + 0(\delta)$$

Similarly

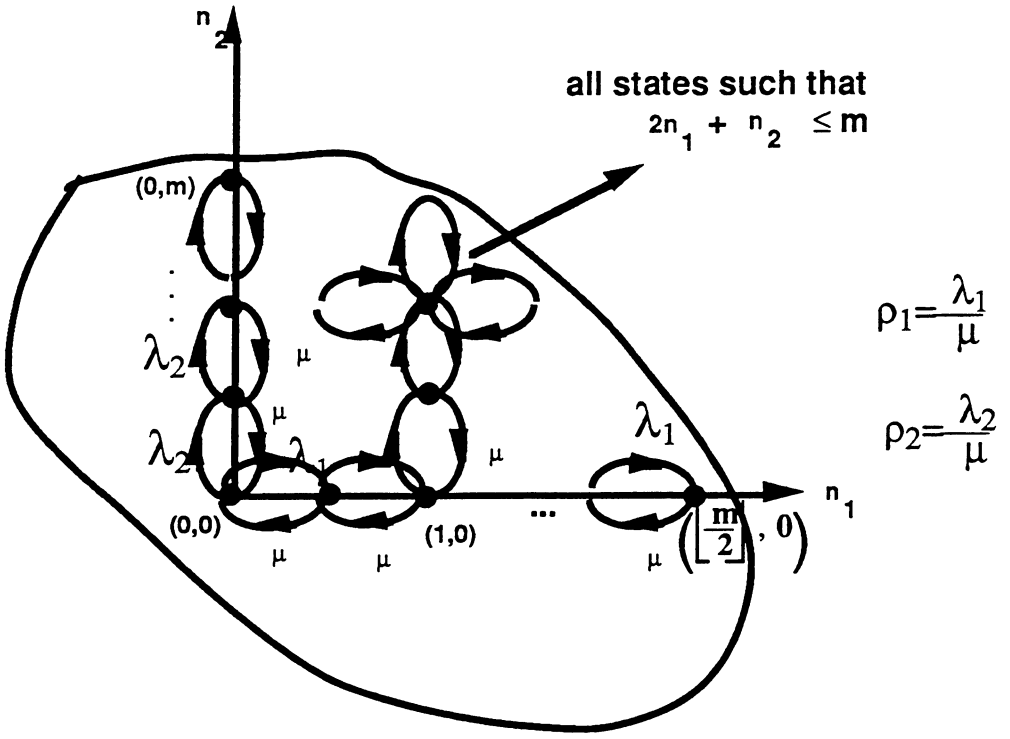
$$P\{1 \text{ arrival and } 0 \text{ departure} \mid \text{starting state} = i\}$$
$$= P\{1 \text{ arrival}\} \bullet P\{0 \text{ departure} \mid \text{starting state} = i\}$$

$$= (\lambda\delta + 0(\delta)) \bullet \left[\binom{i}{0}(1 - (\mu/i)\,\delta\ + 0(\delta))^{i}\right] = \lambda\delta + 0(\delta)$$

Thus the transition rates are the same as for the M/M/1 system.

3.25

Let n_1 be the number of radio-to-radio calls and n_2 be the number of radio-to-nonradio calls which have not finished yet. Then we have the following Markov chain:



The occupancy distribution $p(n_1,n_2)$ is of the form

$$p(n_1,n_2) = \rho_1^{n_1}(1-\rho_1)\rho_2^{n_2}(1-\rho_2)/G, \text{ for } 2n_1+n_2 \le m$$

and 0 otherwise (it is easy to check that this formula satisfies the global balance equations). To calculate G note that

$$\sum_{\{(n_1,n_2)|2n_1+n_2\le m\}} \sum p(n_1,n_2) = 1 \Rightarrow G = \sum_{\{(n_1,n_2)|2n_1+n_2\le m\}} \rho_1^{n_1}(1-\rho_1)\rho_2^{n_2}(1-\rho_2) =$$

$$\sum_{n_1=0}^{\lfloor \frac{m}{2} \rfloor} \sum_{n_2=0}^{m-2n_1} \rho_1^{n_1}(1-\rho_1)\rho_1^{n_2}(1-\rho_2) = \sum_{n_1=0}^{\lfloor \frac{m}{2} \rfloor} \rho_1^{n_1}(1-\rho_1)(1-\rho_2)\frac{1-\rho_2^{m-2n_1+1}}{1-\rho_2}$$

$$= (1-\rho_1)\sum_{n_1=0}^{\lfloor \frac{m}{2} \rfloor} \rho_1^{n_1} - \sum_{n_1=0}^{\lfloor \frac{m}{2} \rfloor}(1-\rho_1)\rho_2^{m+1}\left(\frac{\rho_1}{\rho_2^2}\right)^{n_1}$$

$$= 1-\rho_1^{\lfloor \frac{m}{2} \rfloor +1} - (1-\rho_1)\rho_2^{m+1} \frac{1-\left(\frac{\rho_1}{\rho_2^2}\right)^{\lfloor \frac{m}{2} \rfloor +1}}{1-\frac{\rho_1}{\rho_2^2}}$$

$$= 1-\rho_1^{\lfloor \frac{m}{2} \rfloor +1} - (1-\rho_1)\rho_2^{m+1-2\lfloor \frac{m}{2} \rfloor} \frac{\rho_2^{2\lfloor \frac{m}{2} \rfloor +2} - \rho_1^{\lfloor \frac{m}{2} \rfloor +1}}{\rho_2^2 - \rho_1}$$

$$\Rightarrow G = \begin{cases} 1-\rho_1^{\frac{m}{2}+1} - (1-\rho_1)\rho_2 \dfrac{\rho_2^{m+2} - \rho_1^{m/2+1}}{\rho_2^2 - \rho_1} & \text{if m even} \\ 1-\rho_1^{\frac{m+1}{2}} - (1-\rho_1)\rho_2^2 \dfrac{\rho_2^{m+1} - \rho_1^{\frac{m+1}{2}}}{\rho_2^2 - \rho_1} & \text{if m odd} \end{cases}$$

Let

p_1 = blocking probability of radio-to-radio calls

p_2 = blocking probability of radio-to-nonradio calls

Then

$$p_2 = \sum_{2n_1+n_2=m} p(n_1,n_2)$$

$$p_1 = \sum_{m-1 \le 2n_1+n_2 \le m} p(n_1,n_2) = p_2 + \sum_{2n_1+n_2=m-1} p(n_1,n_2).$$

But

$$p_2 = \sum_{n_1=0}^{\lfloor \frac{m}{2} \rfloor} p(n_1, m-2n_1) = \sum_{n_1=0}^{\lfloor \frac{m}{2} \rfloor} \rho_1^{n_1}(1-\rho_1)\rho_2^{m-2n_1}(1-\rho_2)/G =$$

$$= \frac{(1-\rho_1)(1-\rho_2)\rho_2^m}{G} \frac{1-\left(\rho_1/\rho_2^2\right)^{\lfloor \frac{m}{2} \rfloor +1}}{1-\frac{\rho_1}{\rho_2^2}}$$

and

$$p_1 = p_2 + \sum_{n_1=0}^{\lfloor \frac{m-1}{2} \rfloor} p(n_1, m-1-2n_1) = p_2 + \frac{(1-\rho_1)(1-\rho_2)\rho_2^{m-1}}{G} \cdot \frac{1-\left(\frac{\rho_1}{\rho_2^2}\right)^{\lfloor \frac{m+1}{2} \rfloor}}{1-\frac{\rho_1}{\rho_2^2}}$$

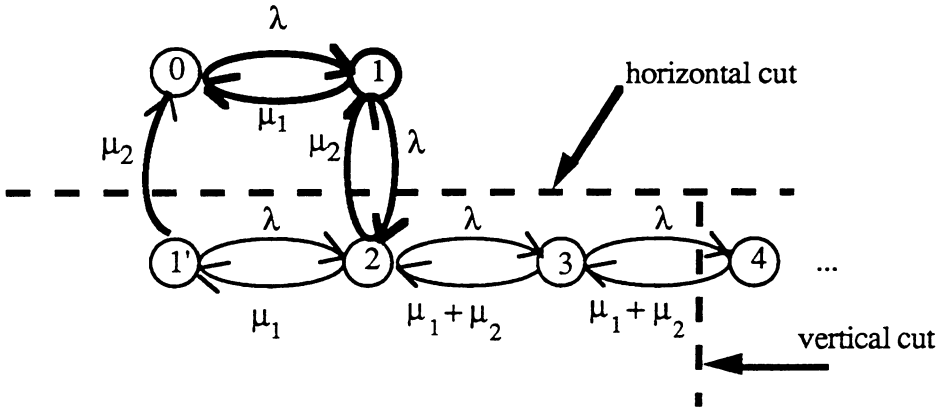
3.26

Define the state to be the number of operational machines. This gives a Markov chain, which is the same as in an M/M/1/m queue with arrival rate λ and service rate μ. The required probability is simply p_0 for such a queue.

3.27

Assume $\mu_1 > \mu_2$.

We have the following Markov chain:



Let state 1 represent 1 customer in the system being served by server 1
Let state 1' represent 1 customer in the system being served by server 2

i) Flow across a vertical cut

$$p_i = \frac{\lambda}{\mu_1 + \mu_2} p_{i-1} \qquad \text{for } i \geq 2$$

Therefore

$$\overline{p_i} = \left(\frac{\lambda}{\mu_1 + \mu_2}\right)^{i-2} p_2 \qquad \text{for } i \geq 2$$

ii) Flow in and out of state 1'

$$(\lambda + \mu_2)\, p_{1'} = p_2\, \mu_1$$

Therefore

$$p_{1'} = p_2\, \frac{\mu_1}{\lambda + \mu_2}$$

iii) Flow across horizontal cut

$$p_1\, \lambda = (p_{1'} + P_2)\, \mu_2$$

Therefore

$$p_1 = \frac{\mu_2}{\lambda}\left(p_2 + p_2\, \frac{\mu_1}{\lambda + \mu_2}\right) = p_2\, \frac{\mu_2}{\lambda}\left(1 + \frac{\mu_1}{\lambda + \mu_2}\right)$$

iv) Flow in and out of state 0

$$p_0\, \lambda = \pi_1\, \mu_1 + p_{1'}\, \mu_2$$

Therefore

$$p_0 = \frac{1}{\lambda}\, p_2 \left(\frac{\mu_1\, \mu_2}{\lambda}\left(1 + \frac{\mu_1}{\lambda + \mu_2}\right) + \frac{\mu_1\, \mu_2}{\lambda + \mu_2}\right)$$

We have

$$\sum_i p_i = 1$$

from which

$$p_2 = \left(\frac{1}{1 - \dfrac{\lambda}{\mu_1 + \mu_2}} + \frac{\left(1 + (\mu_2/\lambda)\right)\mu_1}{\lambda + \mu_2} + \frac{\left(1 + (\mu_1/\lambda)\right)\mu_2}{\lambda}\left(1 + \frac{\mu_1}{\lambda + \mu_2}\right)\right)^{-1}$$

3.28

We have

$$E\{f^2\} = E\left\{\left(\sum_{i=1}^{n}\gamma_i\right)^2\right\} = E\left\{E\left\{\left(\sum_{i=1}^{n}\gamma_i\right)^2 \mid n\right\}\right\} = E\{ns_\gamma^2 + n(n-1)\Gamma^2\}$$
$$= E\{n\}(s_\gamma^2 - \Gamma^2) + E\{n^2\}\Gamma^2$$

Since

$$E\{n\} = \lambda/\mu, \qquad E\{n\} = \sigma_n^2 + (\lambda/\mu) = \lambda/\mu + (\lambda/\mu)$$

we obtain

$$\sigma_f^2 = E\{f^2\} - F^2 = E\{f^2\} - (\lambda/\mu)^2\Gamma^2 = (\lambda/\mu)(s_\gamma^2 - \Gamma^2) + [(\lambda/\mu) + (\lambda/\mu)^2]\Gamma^2 - (\lambda/\mu)^2\Gamma^2$$
$$= (\lambda/\mu)s_\gamma^2$$

so finally

$$\sigma_f = (\lambda/\mu)^{1/2}s_\gamma$$

3.29

For each value of x, the average customer waiting time for each of the two types of customers (x items or less, vs more than x) is obtained by the P-K formula for an M/G/1 system with arrival rate and service time parameters depending on x. By computing the overall customer waiting time for each x in the feasible range [1,40], we can numerically compute the optimal value of x.

Here is a program to find x to minimize the average waiting time:

```
Lambda=1; Past_T= 1000000; T=0; x=-1;
while (x<=40) do
if (T> Past_T) do
        begin
                Past_T = T;
                x = x+1;
                lambda1 = lambda * x/40;
                E_service_time_1 = (1+x)/2;
                E_service_time_2 =  (41+x)/2;
                E_service_time_square1 = 0;
                E_service_time_square2 = 0;
                for i=1 to x do
                        E_service_time_square1 =
                        E_service_time_square1+(i*i);
```

```
                for i=x+1 to 40 do
                        E_service_time_square2 =
                                E_service_time_square2+(i*i);
                E_service_time_square1 =
                        E_service_time_square1/x;
                E_service_time_square2 =
                                E_service_time_square2/(40-x);
                T1 = E_service_time_1 +
                (lambda*E_service_time_square1/(2.0*(1-
                lambda1*E_service_time_1)));
                T2 = E_service_time_2 +
                        (lambda*E_service_time_square2/(2.0*(1-
                                        lambda2*E_service_time_2)));
                T = (T1*x + T2*(40-x))/40;
end;
        print(x);
```

3.30

From Little's Theorem (Example 1) we have that P{the system is busy} = $\lambda E\{X\}$. Therefore P{the system is empty} = 1 - $\lambda E\{X\}$.

The length of an idle period is the interarrival time between two typical customer arrivals. Therefore it has an exponential distribution with parameter λ, and its average length is $1/\lambda$.

Let B be the average length of a busy period and let I be the average length of an idle period. By expressing the proportion of time the system is busy as B/(I + B) and also as $\lambda E\{X\}$ we obtain

$$B = E\{X\}/(1 - \lambda E\{X\}).$$

From this the expression $1/(1 - \lambda E\{X\})$ for the average number of customers served in a busy period is evident.

3.31

The problem with the argument given is that more customers arrive while long-service customers are served, so the average service time of a customer found in service by another customer upon arrival is more than E{X}.

3.32

Following the hint we write for the ith packet

$$U_i = R_i + \sum_{j=1}^{N_i} X_{i-j}$$

where

U_i : Unfinished work at the time of arrival of the ith customer
R_i : Residual service time of the ith customer
N_i : Number found in queue by the ith customer
X_j : Service time of the jth customer

Hence

$$E\{U_i\} = E\{R_i\} + E\left\{\sum_{j=1}^{N_i} E\{X_{i-j} \mid N_i\}\right\}$$

Since X_{i-j} and N_i are independent

$$E\{U_i\} = E\{R_i\} + E\{X\}E\{N_i\}$$

and by taking limit as $i \to \infty$ we obtain $U = R + (1/\mu)N_Q = R + (\lambda/\mu)W = R + \rho W$, so

$$W = (U - R)/\rho.$$

Now the result follows by noting that both U and R are independent of the order of customer service (the unfinished work is independent of the order of customer service, and the steady state mean residual time is also independent of the customer service since the graphical argument of Fig. 3.16 does not depend on the order of customer service).

3.33

Consider the limited service gated system with zero packet length and reservation interval equal to a slot. We have

T_{TDM} = Waiting time in the gated system

For $E\{X^2\} = 0$, $E\{V\} = 1$, $\sigma_V^2 = 0$, $\rho = 0$ we have from the gated system formula (3.77)

Waiting time in the gated system $= (m + 2 - 2\lambda)/(2(1 - \lambda)) = m/(2(1 - \lambda)) + 1$

which is the formula for T_{TDM} given by Eq. (3.59) .

3.34

(a) The system utilization is ρ, so the fraction of time the system transmits data is ρ. Therefore the portion of time occupied by reservation intervals is $1 - \rho$.

(b) If

p: Fraction of time a reservation interval is followed by an empty data interval

and M(t) is the number of reservation intervals up to time t, then the number of packets transmitted up to time t is $\approx (1 - p)M(t)$. The time used for reservation intervals is $\approx M(t)E\{V\}$, and for data intervals $\approx (1 - p)M(t)E\{X\}$. Since the ratio of these times must be $(1 - \rho)/\rho$ we obtain

$$(1 - \rho)/\rho = (M(t)E\{V\})/((1 - p)M(t)E\{X\}) = E\{V\}/((1 - p)E\{X\})$$

or

$$1 - p = (\rho E\{V\})/((1 - \rho)E\{X\})$$

which using $\lambda = \rho/E\{X\}$, yields $p = (1 - \rho - \lambda E\{V\})/(1 - \rho)$

3.35

Consider a gated all-at-once version of the limited service reservation system. Here there are m users, each with independent Poisson arrival rate λ/μ. Each user has a separate queue, and is allowed to make a reservation for at most one packet in each reservation interval. This packet is then transmitted in the subsequent data interval. The difference with the limited service system of Section 3.5.2 is that here users share reservation and data intervals.

Consider the ith packet arrival into the system and suppose that the user associated with packet i is user j. We have as in Section 3.5.2

$$E\{W_i\} = E\{R_i\} + E\{N_i\}/\mu + (1 + E\{Q_i\} - E\{m_i\})E\{V\}$$

where W_i, R_i, N_i, μ, $E\{V\}$ are as in Section 3.5.2, Q_i is the number of packets in the queue of user j found by packet i upon arrival, and m_i is the number (0 or 1) of packets of user j that will start transmission between the time of arrival of packet i and the end of the frame in which packet i arrives. We have as in Section 3.5.2

$$R = \lim_{i\to\infty} E\{R_i\} + E\{N_i\}/\mu + (1 + E\{Q_i\} - E\{m_i\})E\{V\}$$

$$N = \lim_{i\to\infty} E\{N_i\} = \lambda W$$

$$Q = \lim_{i\to\infty} E\{Q_i\} = \lambda W/m$$

so there remains to calculate $\lim_{i\to\infty} E\{m_i\}$.

There are two possibilities regarding the time of arrival of packet i.

a) Packet i arrives during a reservation interval. This event, call it A, has steady state probability (1-ρ)

$$P\{A\} = 1-\rho.$$

Since the ratio of average data interval length to average reservation interval length is $\rho/(1-\rho)$ we see that the average steady state length of a data interval is $\rho E\{V\}/(1-\rho)$. Therefore the average steady state number of packets per user in a data interval is $\rho E\{V\}/((1-\rho)mE\{X\}) = \lambda E\{V\}/((1-\rho)m)$. This also equals the steady state value of $E\{m_i | A\}$ in view of the system symmetry with respect to users

$$\lim_{i\to\infty} E\{m_i | A\} = \frac{\lambda E\{V\}}{(1-\rho)m}.$$

b) Packet i arrives during a data interval. This event, call it B, has steady state probability ρ

$$P\{B\} = \rho.$$

Denote

$$\alpha = \lim_{i\to\infty} E\{m_i \mid B\},$$

$$\alpha_k = \lim_{i\to\infty} E\{m_i \mid B, \text{ the data interval of arrival of packet i contains k packets}\}.$$

Assuming $k > 0$ packets are contained in the data interval of arrival, there is equal probability $1/k$ of arrival during the transmission of any one of these packets. Therefore

$$\alpha_k = \sum_{n=1}^{k} \frac{1}{k}\frac{k-n}{m} = \frac{k(k-1)}{2km} = \frac{k-1}{m}.$$

Let P(k) be the unconditional steady-state probability that a nonempty data interval contains k packets, and $E\{k\}$ and $E\{k^2\}$ be the corresponding first two moments. Then we have using Bayes' rule

$\lim_{i\to\infty} P\{\text{The data interval of arrival of packet i contains k packets}\} = kP(k)/E\{k\}$.

Combining the preceding equations we have

$$\alpha = \sum_{k=1}^{m} \frac{kP(k)}{E\{k\}}\alpha_k = \sum_{k=1}^{m} \frac{P(k)k(k-1)}{2E\{k\}m} = \frac{E\{k^2\}}{2mE\{k\}} - \frac{1}{2m}.$$

We have already shown as part of the analysis of case a) above that

$$E\{k\} = \lambda E\{V\}/(1 - \rho)$$

so there remains to estimate $E\{k^2\}$. We have

$$E\{k^2\} = \sum_{k=1}^{m} k^2 P(k)$$

If we maximize the quantity above over the distribution P(k), k = 0,1,..., m subject to the constraints $\sum^m_{k=1} kP(k) = E\{k\}$, $\sum^m_{k=0} P(k) = 1$, $P(k) \geq 0$ (a simple linear programming problem) we find that the maximum is obtained for $P(m) = E\{k\}/m$, $P(0) = 1 - E\{k\}/m$, and $P(k) = 0$, k = 1,2,...,m-1. Therefore

$$E\{k^2\} \leq mE\{k\}.$$

Similarly if we minimize $E\{k^2\}$ subject to the same constraints we find that the minimum is obtained for $P(k'-1) = k' - E\{k\}$, $P(k) = 1 - (k' - E\{k\})$ and $P(k') = 0$ for $k \neq k' - 1, k'$ where k' is the integer for which $k' - 1 \leq E\{k\} < k'$. Therefore

$$E\{k^2\} \geq (k' - 1)^2(k' - E\{k\}) + (k')^2[1 - (k' - E\{k\})]$$

After some calculation this relation can also be written

$$E\{k^2\} \geq E\{k\} + (k' - 1)(2E\{k\} - k') \quad \text{for } E\{k\} \in (k' - 1, k'), \quad k' = 1, 2, \ldots, m$$

Note that the lower bound above is a piecewise linear function of $E\{k\}$, and equals $(E\{k\})^2$ at the breakpoints k' = 1,2,...,m. Summarizing the bounds we have

$$\frac{E\{k\} + (k' - 1)(2E\{k\} - k')}{2mE\{k\}} - \frac{1}{2m} \leq \alpha \leq \frac{1}{2} - \frac{1}{2m},$$

where k' is the positive integer for which

$$k' - 1 \leq E\{k\} < k' .$$

Note that as $E\{k\}$ approaches its maximum value m (i.e., the system is heavily loaded), the upper and lower bounds coincide. By combining the results for cases a) and b) above we have

$$\lim_{i\to\infty} E\{m_i\} = P\{A\} \lim_{i\to\infty} E\{m_i | A\} + P\{B\} \lim_{i\to\infty} E\{m_i | B\}$$

$$= (1-\rho)\frac{\lambda E\{V\}}{(1-\rho)m} + \rho\alpha$$

or finally

$$\lim_{i\to\infty} E\{m_i\} = \frac{\lambda E\{V\}}{m} + \rho\alpha$$

where α satisfies the upper and lower bounds given earlier. By taking limit as $i \to \infty$ in the equation

$$E\{W_i\} = E\{R_i\} + E\{N_i\}/\mu + (1 + E\{Q_i\} - E\{m_i\})E\{V\}$$

and using the expressions derived we finally obtain

$$W = \frac{\lambda E\{X^2\}}{2\left(1-\rho-\frac{\lambda E\{V\}}{m}\right)} + \frac{(1-\rho)E\{V^2\}}{2\left(1-\rho-\frac{\lambda E\{V\}}{m}\right)E\{V\}} + \frac{\left(1-\rho\alpha-\frac{\lambda E\{V\}}{m}\right)E\{V\}}{1-\rho-\frac{\lambda E\{V\}}{m}}$$

where α satisfies

$$\frac{E\{k\} + (k' - 1)(2E\{k\} - k')}{2mE\{k\}} - \frac{1}{2m} \le \alpha \le \frac{1}{2} - \frac{1}{2m},$$

$E\{k\}$ is the average number of packets per data interval

$$E\{k\} = \lambda E\{V\}/(1 - \rho)$$

and k' is the integer for which $k' - 1 \le E\{k\} < k'$. Note that the formula for the waiting time W becomes exact in the limit both as $\rho \to 0$ (light load), and as $\rho \to 1 - \lambda E\{V\}/m$ (heavy load) in which case $E\{k\} \to m$ and $\alpha \to 1/2 - 1/2m$. When $m = 1$ the formula for W is also exact and coincides with the one derived for the corresponding single user one-at-a-time limited service system.

3.36

For each session, the arrival rates, average transmission times and utilization factors for the short packets (class 1), and the long packets (class 2) are

$\lambda_1 = 0.25$ packets/sec,	$1/\mu_1 = 0.02$ secs,	$\rho_1 = 0.005$
$\lambda_2 = 2.25$ packets/sec,	$1/\mu_2 = 0.3$ secs,	$\rho_2 = 0.675$.

The corresponding second moments of transmission time are

$$E\{X_1^2\} = 0.0004 \qquad E\{X_2^2\} = 0.09.$$

The total arrival rate for each session is $\lambda = 2.5$ packets/sec. The overall 1st and 2nd moments of the transmission time, and overall utilization factors are given by

$$1/\mu = 0.1*(1/\mu_1) + 0.9*(1/\mu_2) = 0.272$$
$$E\{X^2\} = 0.1*E\{X_1^2\} + 0.9*E\{X_2^2\} = 0.081$$
$$\rho = \lambda/\mu = 2.5*0.272 = 0.68.$$

We obtain the average time in queue W via the P - K formula $W = (\lambda E\{X^2\})/(2*(1 - \rho)) = 0.3164$. The average time in the system is $T = 1/\mu + W = 0.588$. The average number in queue and in the system are $N_Q = \lambda W = 0.791$, and $N = \lambda T = 1.47$.

The quantities above correspond to each session in the case where the sessions are time - division multiplexed on the line. In the statistical multiplexing case W, T , N_Q and N are decreased by a factor of 10 (for each session).

In the nonpreemptive priority case we obtain using the corresponding formulas:

$$W_1 = (\lambda_1 E\{X_1^2\} + \lambda_2 E\{X_2^2\})/(2*(1 - \rho_1)) = 0.108$$
$$W_2 = (\lambda_1 E\{X_1^2\} + \lambda_2 E\{X_2^2\})/(2*(1 - \rho_1)*(1 - \rho_1 - \rho_2)) = 0.38$$
$$T_1 = 1/\mu_1 + W_1 = 0.128$$
$$T_2 = 1/\mu_2 + W_2 = 1.055$$
$$N_{Q1} = \lambda_1 * W_1 = 0.027 \qquad N_{Q2} = \lambda_2 * W_2 = 0.855$$
$$N_1 = \lambda_1 * T_1 = 0.032 \qquad N_2 = \lambda_2 * T_2 = 2.273.$$

3.37

(a)

$\lambda = 1/60$ per second
$E(X) = 16.5$ seconds
$E(X^2) = 346.5$ seconds
$T = E(X) + \lambda E(X^2)/2(1-\lambda E(X))$
$= 16.5 + (346.5/60)/2(1- 16.5/60) = 20.48$ seconds

(b) Non-Preemptive Priority

In the following calculation, subscript 1 will imply the quantities for the priority 1 customers and 2 for priority 2 customers. Unsubscripted quantities will refer to the overall system.

$$\lambda = \frac{1}{60}\ ,\ \lambda_1 = \frac{1}{300}\ ,\ \lambda_2 = \frac{1}{75}$$

$$E(X) = 16.5\ ,\ E(X_1) = 4.5\ ,\ E(X_2) = 19.5$$

$$E(X^2) = 346.5$$

$$R = \frac{1}{2}\lambda E(X^2) = 2.8875$$

$$\rho_1 = \lambda_1 E(X_1) = 0.015$$

$$\rho_2 = \lambda_2 E(X_2) = 0.26$$

$$W_1 = \frac{R}{1-\rho_1} = 2.931$$

$$W_2 = \frac{R}{1-\rho_2} = 4.043$$

$$T_1 = 7.4315,\ T_2 = 23.543$$

$$T = \frac{\lambda_1 T_1 + \lambda_2 T_2}{\lambda} = 20.217$$

(c) Preemptive Queueing

The arrival rates and service rates for the two priorities are the same for preemptive system as the non-preemptive system solved above.

$$E(X_1^2) = 22.5,\ E(X_2^2) = 427.5$$

$$R_1 = \frac{1}{2}\lambda_1 E(X_1^2) = 0.0075$$

$$R_2 = R_1 + \frac{1}{2}\lambda_2 E(X_2^2) = 2.8575$$

$$T_1 = \frac{E(X_1)(1-\rho_1) + R_1}{1-\rho_1}$$

$$T_2 = \frac{E(X_2)(1-\rho_1-\rho_2) + R_2}{(1-\rho_1)(1-\rho_1-\rho_2)}$$

$$T = (\lambda_1 T_1 + \lambda_2 T_2)/\lambda = 19.94$$

3.38

(a) The same derivation as in Section 3.5.3 applies for W_k, i.e.

$$W_k = R/(1 - \rho_1 - \ldots - \rho_{k-1})(1 - \rho_1 - \ldots - \rho_k)$$

where $\rho_i = \lambda_i/(m\mu)$, and R is the mean residual service time. Think of the system as being comprised of a serving section and a waiting section. The residual service time is just the time until any of the customers in the waiting section can enter the serving section. Thus, the residual service time of a customer is zero if the customer enters service immediately because there is a free server at the time of arrival, and is otherwise equal to the time between the customer's arrival, and the first subsequent service completion. Using the memoryless property of the exponential distribution it is seen that

$$R = P_Q E\{\text{Residual service time} \mid \text{queueing occurs}\} = P_Q/(m\mu).$$

(b) The waiting time of classes 1, . . ., k is not influenced by the presence of classes (k+1), . . . , n. All priority classes have the same service time distribution, thus, interchanging the order of service does not change the average waiting time. We have

$$W_{(k)} = \text{Average waiting time for the M/M/m system with rate } \lambda_1 + \ldots + \lambda_k.$$

By Little's theorem we have

$$\text{Average number in queue of class k} = \text{Average number in queue of classes 1 to k} - \text{Average number in queue of classes 1 to k-1}$$

and the desired result follows.

3.39

Let k be such that

$$\rho_1 + \ldots + \rho_{k-1} \leq 1 < \rho_1 + \ldots + \rho_k.$$

Then the queue of packets of priority k will grow infinitely, the arrival rate of each priority up to and including k-1 will be accomodated, the departure rate of priorities above k will be zero while the departure rate of priority k will be

$$\tilde{\lambda}_k = \frac{(1 - \rho_1 - \cdots - \rho_{k-1})}{\bar{X}_k}$$

In effect we have a priority system with k priorities and arrival rates

$$\tilde{\lambda}_i = \lambda_i \qquad \text{for } i < k$$

$$\tilde{\lambda}_k = \frac{(1 - \rho_1 - \cdots - \rho_{k-1})}{\bar{X}_k}$$

For priorities $i < k$ the arrival process is Poisson so the same calculation for the waiting time as before gives

$$W_i = \frac{\sum_{i=1}^{k} \tilde{\lambda}_i \bar{X}_i^2}{2(1 - \rho_1 - \cdots - \rho_{i-1})(1 - \rho_1 - \cdots - \rho_i)}, \qquad i < k$$

For priority k and above we have infinite average waiting time in queue.

3.40

(a) The algebraic verification using Eq. (3.79) listed below

$$W_k = R/(1 - \rho_1 - \ldots - \rho_{k-1})(1 - \rho_1 - \ldots - \rho_k)$$

is straightforward. In particular by induction we show that

$$\rho_1 W_1 + \cdots + \rho_k W_k = \frac{R(\rho_1 + \quad + \rho_k)}{1 - \rho_1 - \cdots - \rho_k}$$

The induction step is carried out by verifying the identity

$$\rho_1 W_1 + \cdots + \rho_k W_k + \rho_{k+1} W_{k+1} = \frac{R(\rho_1 + \cdots + \rho_k)}{1 - \rho_1 - \cdots - \rho_k} + \frac{\rho_{k+1} R}{(1 - \rho_1 - \cdots \rho_k)(1 - \rho_1 - \cdots - \rho_{k+1})}$$

The alternate argument suggested in the hint is straightforward.

(b) Cost

$$C = \sum_{k=1}^{n} c_k N_Q^k = \sum_{k=1}^{n} c_k \lambda_k W_k = \sum_{k=1}^{n} \left(\frac{c_k}{\bar{X}_k}\right) \rho_k W_k$$

We know that $W_1 \le W_2 \le \ldots\ldots \le W_n$. Now exchange the priority of two neighboring classes i and j=i+1 and compare C with the new cost

$$C' = \sum_{k=1}^{n} \left(\frac{c_k}{\bar{X}_k}\right) \rho_k W'_k$$

In C' all the terms except k = i and j will be the same as in C because the interchange does not affect the waiting time for other priority class customers. Therefore

$$C'-C = \frac{c_j}{\overline{X}_j}\rho_j W'_j + \frac{c_i}{\overline{X}_i}\rho_i W'_i - \frac{c_i}{\overline{X}_i}\rho_i W_i - \frac{c_j}{\overline{X}_j}\rho_j W_j.$$

We know from part (a) that

$$\sum_{k=1}^{n} \rho_k W_k = \text{constant}.$$

Since W_k is unchanged for all k except k = i and j (= i+1) we have

$$\rho_i W_i + \rho_j W_j = \rho_i W'_i + \rho_j W'_j.$$

Denote

$$B = \rho_i W'_i - \rho_i W_i = \rho_j W_j - \rho_j W'_j$$

Clearly we have $B \geq 0$ since the average waiting time of customer class i will be increased if class i is given lower priority. Now let us assume that

$$\frac{c_i}{\overline{X}_i} \leq \frac{c_j}{\overline{X}_j}$$

Then

$$C'-C = \frac{c_i}{\overline{X}_i}(\rho_i W'_i - \rho_i W_i) - \frac{c_j}{\overline{X}_j}(\rho_j W_j - \rho_j W'_j) = B\left(\frac{c_i}{\overline{X}_i} - \frac{c_j}{\overline{X}_j}\right)$$

Therefore, only if $\frac{c_i}{\overline{X}_i} < \frac{c_{i+1}}{\overline{X}_{i+1}}$ can we reduce the cost by exchanging the priority order of i and i+1. Thus, if (1,2,3,...,n) is an optimal order we must have

$$\frac{c_1}{\overline{X}_1} \geq \frac{c_2}{\overline{X}_2} \geq \frac{c_3}{\overline{X}_3} \geq \ldots \geq \frac{c_n}{\overline{X}_n}$$

3.41

Let D(t) and $T_i(t)$ be as in the solution of Problem 3.31. The inequality in the hint is evident from Figure 3.30, and therefore it will suffice to show that

$$\lim_{t\to\infty} \frac{1}{t} \sum_{i\in D(t)} T_i = \lim_{t\to\infty} \frac{1}{t} \sum_{i=1}^{\alpha(t)} T_i \qquad (1)$$

We first show that

$$T_k/t_k \to 0 \qquad \text{as } k \to \infty \qquad (2)$$

where t_k is the arrival time of the kth customer. We have by assumption

$$\lim_{k\to\infty} (k/t_k) = \lambda,$$

and by taking the limit as $k\to\infty$ in the relation

$$(k+1)/t_{k+1} - k/t_k = 1/t_{k+1} - ((t_{k+1} - t_k)/t_{k+1})(k/t_k)$$

we obtain

$$t_k/t_{k+1} \to 1 \qquad \text{as } k \to \infty \qquad (3)$$

We also have

$$\frac{\sum_{i=1}^{k} T_i}{t_k} = \frac{k}{t_k} \frac{\sum_{i=1}^{k} T_i}{k} \to \lambda T \qquad \text{as } k \to \infty \qquad (4)$$

so

$$\frac{\sum_{i=1}^{k+1} T_i}{t_{k+1}} - \frac{\sum_{i=1}^{k} T_i}{t_k} \to 0$$

or

$$\frac{T_{k+1}}{t_{k+1}} + \frac{\sum_{i=1}^{k} T_i}{t_k} \left(\frac{t_k}{t_{k+1}} - 1\right) \to 0$$

which proves (2).

Let $\varepsilon > 0$ be given. Then, from (2), there exists k such that $T_i < t_i \varepsilon$ for all $i > k$. Choose t large enough so that $\alpha(t) > k$. Then

$$\sum_{i=1}^{\beta(t)} M_i \le \int_0^t r(\tau)d\tau \le \sum_{i=1}^{\alpha(t)} M_i$$

or

$$\frac{\beta(t)}{t}\frac{\sum_{i=1}^{\beta(t)} M_i}{\beta(t)} \le \frac{1}{t}\int_0^t r(\tau)d\tau \le \frac{\alpha(t)}{t}\frac{\sum_{i=1}^{\alpha(t)} M_i}{\alpha(t)}$$

Under the assumptions

$$\lambda = \lim_{t\to\infty}\frac{\alpha(t)}{t} = \lim_{t\to\infty}\frac{\beta(t)}{t}$$

$$M = \lim_{k\to\infty}\frac{1}{k}\sum_{i=1}^{k} M_i$$

we have

$$R = \lambda M$$

where

$$R = \lim_{t\to\infty}\frac{1}{t}\int_0^t r(\tau)d\tau$$

is the time average rate at which the system earns.

(b) Take $r_i(t) = 1$ for all t for which customer i resides in the system, and $r_i(t) = 0$ for all other t.

(c) If X_i and W_i are the service and queueing times of the ith customer we have

$$M_i = X_i W_i + (X_i^2)/2$$

where the two terms on the right above correspond to payment while in queue and service respectively. Applying part (a) while taking expected values and taking the limit as $i \to \infty$, and using the fact that X_i and W_i are independent we obtain

$$U = \lambda(E\{X\}W + E\{(X_i^2)/2\})$$

where U, the rate at which the system earns, is the average unfinished work in the system. Since the arrival process is Poisson, U is also the average unfinished work seen by an arriving customer. Therefore U = W, and the P - K formula follows.

3.43

We have similar to Section 3.5

$$W = R + \rho W + \bar{W}_B \qquad (1)$$

where the mean residual service time is

$$R = \frac{\lambda \bar{n} \bar{X^2}}{2}$$

We derive the average waiting time of a customer for other customers that arrived in the same batch

$$\bar{W}_B = \sum_j r_j E\{W_B \mid \text{batch has size } j\}$$

where

P_j = Probability a batch has size j
r_j = Proportion of customers arriving in a batch of size j

We have

$$r_j = \frac{jP_j}{\sum_{n=1}^{\infty} nP_n} = \frac{jP_j}{\bar{n}}$$

Also since the customer is equally likely to be in any position within the batch

$$E\{W_B \mid \text{batch is of size } j\} = \sum_{k=1}^{j} (k \sim 1)\bar{X}\frac{1}{j} = \frac{j-1}{2}\bar{X}$$

Thus from (2)

$$E\{W_B\} = \sum_j \frac{jP_j(j-1)\bar{X}}{2\bar{n}} = \frac{\bar{X}(\bar{n^2} - \bar{n})}{2\bar{n}}$$

Substituting in (1) we obtain

$$W = \frac{R}{1-\rho} + \frac{\bar{W}_B}{1-\rho}$$

$$= \frac{\lambda \bar{n} \bar{X}^2}{2(1-\rho)} + \frac{\bar{x}(\overline{n^2} - \bar{n})}{2\bar{n}(1-\rho)}$$

3.44

(a) Let p_0 be the steady state probability that an arriving packet finds the system empty. Then, in effect, a packet has service time X with probability 1 - p_0, and service time X + Δ with probability p_0. The fraction of time the system is busy is obtained by applying Little's Theorem to the service portion of the system

$$\lambda[E\{X\}(1 - p_0) + (E\{X\} + E\{\Delta\})p_0] = \lambda(E\{X\} + E\{\Delta\}p_0)$$

This is also equal to P{system busy} $\doteq$ 1 - p_0, so by solving for p_0, we obtain

$$p_0 = (1 - \lambda E\{X\})/(1 + \lambda E\{\Delta\}) = (1 - \rho)/(1 + \lambda E\{\Delta\})$$

where $\rho = \lambda E\{X\}$.

(b) Let

E{I} = average length of an idle period
E{B} = average length of a busy period

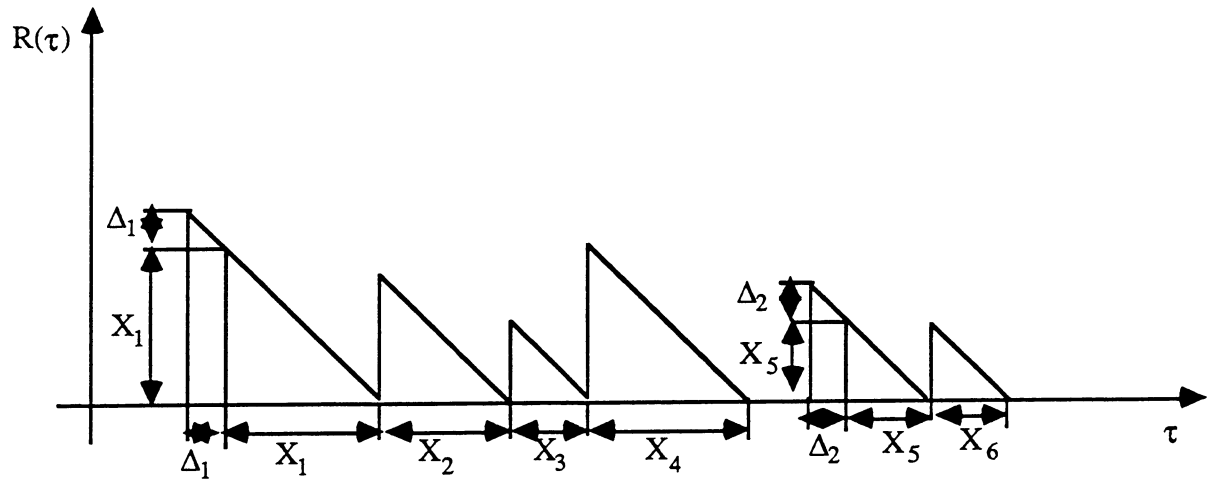
Since the arrival process is Poisson we have

E{I} = $1/\lambda$ = average time betwen last departure in a busy period and the next arrival in the system

$$p_0 = \frac{E\{I\}}{E\{I\} + E\{B\}} = \frac{1/\lambda}{1/\lambda + E\{B\}} = \frac{1 \sim \lambda E\{X\}}{1 + \lambda E\{\Delta\}}$$

$$E\{B\} = \frac{E\{X\} + E\{\Delta\}}{1 - E\{X\}\lambda} = \frac{E\{X\} + E\{\Delta\}}{1 - \rho}$$

(c) We calculate the mean residual time using a graphical argument



From the figure we have

$$\int_0^t R(\tau)d\tau = \sum_{i=1}^{M(t)} \frac{1}{2}X_i^2 + \sum_{i=1}^{N(t)} X_{j(i)}\Delta_i + \frac{1}{2}\Delta_i^2$$

where $X_{j(i)}$ is the service time of the first packet of the ith busy period, and

M(t) = # of arrivals up to t
N(t) = # of busy periods up to t

Taking the limit as $t\to\infty$, and using the fact

$$\lim_{t\to\infty}\frac{N(t)}{t} = \frac{1-p_0}{E\{B\}} = \frac{\lambda(1-\rho)}{1+\lambda E\{\Delta\}}$$

we obtain

$$R = \lim_{t\to\infty}\frac{\int_0^t R(\tau)d\tau}{t} = \lim_{t\to\infty}\left[\frac{M(t)}{t}\frac{\sum_{i=1}^{M(t)}\frac{1}{2}X_i^2}{M(t)} + \frac{N(t)}{t}\frac{\sum_{i=1}^{N(t)}X_{j(i)}\Delta_i + \frac{1}{2}\Delta_i^2}{N(t)}\right]$$

We have, as in Section 3.5, W = R +ρW or

$$W = R/(1 - \rho)$$

Substituting the expression for R obtained previously we obtain

$$W = \frac{\lambda E\{X^2\}}{2(1-\rho)} + \frac{\lambda}{2(1+\lambda E\{\Delta\})}[E\{(X+\Delta)^2\} - E\{X^2\}]$$

3.45

(a) It is easy to see that

Pr (system busy serving customers) = ρ

Pr (system idle) = $1-\rho$ = P(0 in system) + P(1 in idle system) + ... + P(k-1 in idle system)

It can be seen that

P(0 in system) = P(1 in idle system) = ... = P(k-1 in idle system) = $(1-\rho)/k$

implying that

$$P(\text{nonempty and waiting}) = \frac{(1-\rho)(k-1)}{k}$$

(b) A busy period in this problem is the length of time the system is nonempty. The end of each busy period is a renewal point for the system. Between two renewal points, the system is either waiting (with 0 or more customers) or serving.

Let $\overline{W}$ be the expected length of a waiting period.
Since arrivals are poisson, we have

$$\overline{W} = \frac{k}{\lambda} .$$

Let $\overline{S}$ be the expected length of a serving period.
Then the probability that the system is serving = $\rho = \frac{\overline{S}}{\overline{S} + \overline{W}}$
implying that

$$\frac{1}{\rho} - 1 = \frac{\overline{W}}{\overline{S}}$$

or

$$\overline{S} = \frac{\rho \overline{W}}{1-\rho} = \frac{\rho \frac{k}{\lambda}}{1-\rho}$$

Let $\bar{I}$ be the expected length of time the system is empty.

The expected length of a busy period $= \bar{S} + \overline{\bar{W}} - \bar{I}$

$$= \frac{\rho(k/\lambda)}{1-\rho} + \frac{k}{\lambda} - \frac{1}{\lambda}$$

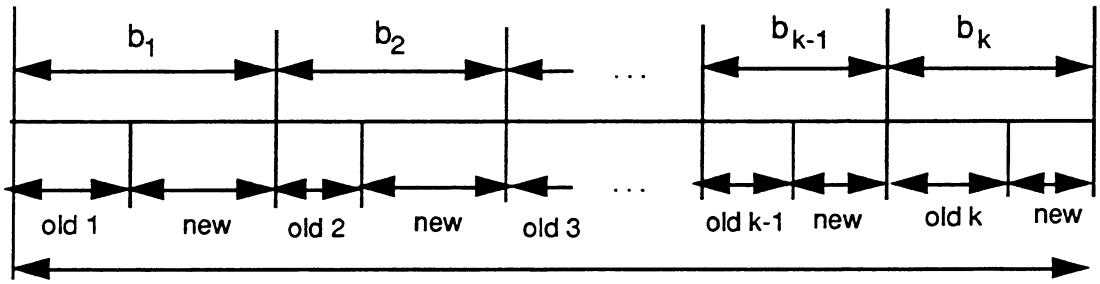
$$= \frac{\rho\, k + (1-\rho)\,(k-1)}{\lambda(1-\rho)} = \frac{k+\rho-1}{\lambda\,(1-\rho)}$$

$\frac{\rho\,(k/\lambda)}{1-\rho}$ is k times the average length of an M/G/1 busy period and $\frac{k-1}{\lambda}$

is the average time from the first arrival until the system starts serving.

(c) We will call the k packets that are in the system at the beginning of the busy period "old" packets. Those that come during the busy period (and are therefore served during this period) are called "new" packets.

We consider the following service discipline: the server transmits old packets only when it doesn't have new packets available at that moment (so it's not FCFS). Since this discipline doesn't depend on the length of the packets it will give the same average number of packets in the system. Thus a busy period looks as illustrated below:



Busy period

In a subperiod b_i of the busy period, the old packet i and the new packets combine to give the same distribution as a normal M/G/1 busy period except that there are an extra k-i old packets in the system. It is easy to see that the distribution of the length of $b_1, b_2, \ldots b_k$ is the same since each of them is exactly like an M/G/1 busy period.

$$\Rightarrow E(N \mid \text{serving}) = E(N \mid b_1)\, P(b_1 \mid \text{serving}) + \ldots + E(N \mid b_k)\, P(b_k \mid \text{serving})$$

$$P(b_i \mid \text{serving}) = \frac{1}{k}$$

$$E(N \mid b_i) = E(N_{M/G/1} \mid \text{busy}) + k\text{-}i$$

implying that

$$E(N \mid \text{serving}) = \frac{1}{k}\left(k\ E(N_{M/G/1} \mid \text{busy}) + \sum_{i=1}^{k}(k\text{-}i)\right)$$

$$= \frac{k\text{-}1}{2} + E(N_{M/G/1} \mid \text{busy})$$

We have

$$E(N_{M/G/1}) = E(N_{M/G/1} \mid \text{busy})\ \rho$$

from which

$$E(N_{M/G/1} \mid \text{busy}) = \frac{E(N_{M/G/1})}{\rho}$$

or

$$E(N \mid \text{serving}) = \frac{E(N_{M/G/1})}{\rho} + \frac{k\text{-}1}{2}$$

Also

$$E(N \mid \text{busy waiting}) = E(N \mid \text{waiting with 1})\ P(\text{waiting with 1} \mid \text{busy waiting}) + \ldots$$

$$+ E(N \mid \text{waiting with k-1})\ P(\text{waiting with k-1} \mid \text{busy waiting})$$

$$P(\text{waiting with i} \mid \text{busy waiting})$$

$$= P(\text{waiting with j} \mid \text{busy waiting}) = \frac{1}{k\text{-}1} \quad \text{for all } 0 < i, j < k$$

from which

$$E(N \mid \text{busy waiting}) = \frac{1}{k\text{-}1}\sum_{i=1}^{k-1} i \quad = \frac{k}{2}$$

(d) $E(N) = E(N \mid \text{busy waiting})\ P(\text{busy waiting}) + E(N \mid \text{busy serving})\ P(\text{busy serving})$

$$= \frac{k}{2}\ \frac{(k\text{-}1)\ (1\text{-}\rho)}{k} + \left(\frac{E(N_{M/G/1})}{\rho} + \frac{k\text{-}1}{2}\right)\rho$$

$$= E(N_{M/G/1}) + \frac{k\text{-}1}{2}$$

3.46

We have

$$W = R/(1 - \rho)$$

where

$$R = \lim_{t\to\infty}\left\{\frac{1}{t}\sum_{i=1}^{M(t)}\frac{1}{2}X_i^2 + \frac{1}{t}\sum_{i=1}^{L(t)}\frac{V_i^2}{2}\right\}$$

where L(t) is the number of vacations (or busy periods) up to time t. The average length of an idle period is

$$I = \int_0^\infty p(v)\left[\int_0^v v\lambda e^{-\lambda\tau}d\tau + \int_v^\infty \tau\lambda e^{-\lambda\tau}d\tau\right]dv$$

and it can be seen that the steady-state time average number of vacations per unit time

$$\lim_{t\to\infty}\frac{L(t)}{t} = \frac{1 - \rho}{I}$$

We have

$$\lim_{t\to\infty}\frac{1}{t}\sum_{i=1}^{L(t)}\frac{V_i^2}{2} = \lim_{t\to\infty}\frac{L(t)}{t}\frac{\sum_{i=1}^{L(t)}\frac{V_i^2}{2}}{L(t)} = \lim_{t\to\infty}\frac{L(t)}{t}\frac{\overline{V}^2}{2I} = \frac{\overline{V}^2(1 - \rho)}{2I}$$

Therefore

$$R = \frac{\lambda\overline{X}^2}{2} + \frac{\overline{V}^2(1 - \rho)}{2I}$$

and

$$W = \frac{\lambda\overline{X}^2}{2(1 - \rho)} + \frac{\overline{V}^2}{2I}$$

3.47

(a) Since arrival times and service times are independent, the probability that there was an arrival in a small interval δ at time $\tau - x$ and that this arrival is still being served at time τ is $\lambda\delta[1 - F_X(x)]$.

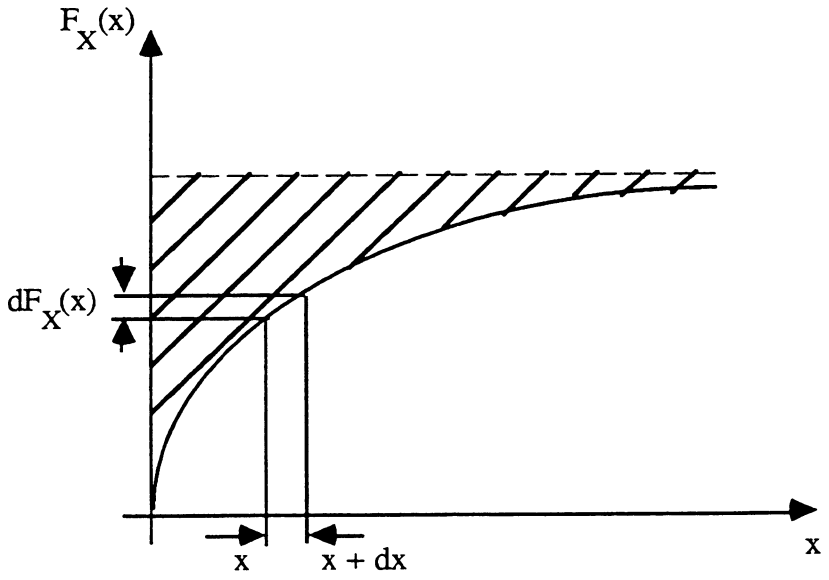
(b) We have

$$\overline{X} = \int_0^\infty x dF_X(x)$$

and by calculating the shaded area of the figure below in two different ways we obtain

$$\int_0^\infty x dF_X(x) = \int_0^\infty [1 - F_X(x)]dx$$

This proves the desired expression.



(c) Let $p_n(x)$ be the steady state probability that the number of arrivals that occurred prior to time $\tau - x$ and are still present at time τ is exactly n.

For $n \geq 1$ we have

$$p_n(x - \delta) = \{1 - \lambda[1 - F_X(x)]\delta\}p_n(x) + \lambda[1 - F_X(x)]\delta p_{n-1}(x)$$

and for n =0 we have

$$p_0(x - \delta) = \{1 - \lambda[1 - F_X(x)]\delta\}p_0(x).$$

Thus $p_n(x)$, n = 0, 1, 2, . . . are the solution of the differential equations

$$dp_n/dx = a(x)p_n(x) - a(x)p_{n-1}(x) \qquad \text{for } n \geq 1$$

$$dp_0/dx = a(x)p_0(x) \qquad \text{for } n = 0$$

where

$$a(x) = \lambda[1 - F_X(x)].$$

Using the known conditions

$$p_n(\infty) = 0 \qquad \text{for } n \geq 1$$
$$p_0(\infty) = 1$$

it can be verified by induction starting with $n = 0$ that the solution is

$$p_n(x) = [e^{-\int_x^\infty a(y)dy}] \frac{[\int_x^\infty a(y)dy]^n}{n!}, \qquad x \geq 0, \quad n = 0, 1, 2, \ldots$$

Since

$$\int_0^\infty a(y)dy = \lambda \int_0^\infty [1 - F_X(x)]dy = \lambda E\{X\}$$

we obtain

$$p_n(0) = e^{-\lambda E\{X\}} \frac{[\lambda E\{X\}]^n}{n!}, \qquad n = 0, 1, 2, \ldots$$

Thus the number of arrivals that are still in the system have a steady state Poisson distribution with mean $\lambda E\{X\}$.

3.48

(a) Denote

$$f(x) = E_r[(\max\{0, r-x\})^2]$$

and

$$g(x) = (E_r[\max\{0, r-x\}])^2,$$

where $E_r[\cdot]$ denotes expected value with respect to r (x is considered constant). We will prove that $f(x)/g(x)$ is montonically nondecreasing for x nonnegative and thus attain its minimum value for $x=0$. We have

$$\frac{\partial f(x)}{\partial x} = E_r\left[\frac{\partial}{\partial x}(\max\{0,r-x\})^2\right] = 2E_r\left[\max\{0,r-x\}\cdot \frac{\partial}{\partial x}(\max\{0,r-x\})\right],$$

where

$$\frac{\partial(\max\{0,r-x\})}{\partial x} = -\,u(r-x)$$

where $u(\cdot)$ is the step function. Thus

$$\frac{\partial f(x)}{\partial x} = -2E_r\,[(\max\{0,r-x\})\cdot u(r-x)] = -2E_r[\max\{0,r-x\}]$$

Assume for simplicity that r has a probability density function (the solution is similar in the more general case). Then

$$\frac{\partial g(x)}{\partial x} = 2E_r[\max\{0,r-x\}]\,E_r\left[\frac{\partial}{\partial x}\max\{0,r-x\}\right] = -2E_r[\max\{0,r-x\}]\cdot\int_{r>x}p(r)dr$$

Thus

$$\frac{\partial f(x)}{\partial x}\,g(x) - f(x)\frac{\partial g(x)}{\partial x} = 2E_r[\max\{0,r-x\}]E_r\left[(\max\{0,r-x\})^2\right]\int_{r>x}p(r)dr$$

$$-\,2E_r[\max\{0,r-x\}]E_r([\max\{0,r-x\}])^2$$

For $\frac{f(x)}{g(x)}$ monotonically nondecreasing we must have $\frac{\partial f(x)}{\partial x}\,g(x) - f(x)\,\frac{\partial g(x)}{\partial x} \geq 0$ or equivalently

$$E_r[(\max\{0,r-x\})^2]\int_{r>x}p(r)dr - (E_r[\max\{0,r-x\}])^2$$

$$= \int_{r>x}(r-x)^2p(r)dr\int_{r>x}p(r)dr - \left(\int_{r>x}(r-x)p(r)dr\right)^2 \geq 0$$

which is true by Schwartz's inequality. Thus the ratio

$$\frac{f(x)}{g(x)} = \frac{E_r[(\max\{0,r-x\})^2]}{(E_r[\max\{0,r-x\}])^2}$$

is monotonically nondecreasing and attains its minimum value at x=0. On the other hand, we have

$$\frac{f(0)}{g(0)} = \frac{E(r^2)}{[E(r)]^2},$$

since $r \geq 0$, and the result follows.

(b) We know (cf. Eq. (3.93)) that

$$I_k = -\min\{0, W_k + X_k - \tau_k\} = \max\{0, \tau_k - W_k - X_k\}$$

$$= \max\{0, \tau_k - S_k\},$$

where S_k is the time in the system. Since the relation in part (a) holds for any nonnegative scalar x, we can take expected values with respect to x as well, and use the fact that for any function of x, g(x), $E(g(x)^2) \geq E^2(g(x))$, and find that

$$E_{r,x}[(\max\{0,r-x\})^2] \geq \frac{\overline{r^2}}{(\bar{r})^2} (E_{r,x}[\max\{0,r-x\}])^2, \tag{1}$$

where x is considered to be a random variable this time. By letting $r = \tau_k$, $x = S_k$, and $k \to \infty$, we find from (1) that

$$\overline{I^2} \geq \frac{\overline{\tau^2}}{(\bar{\tau})^2} (\bar{I})^2$$

or

$$\frac{\overline{I^2} - (\bar{I})^2}{(\bar{I})^2} \geq \frac{\overline{\tau^2} - (\bar{\tau})^2}{(\bar{\tau})^2}$$

or

$$\sigma_I^2 \geq \frac{(\bar{I})^2}{(\bar{\tau})^2} \sigma_a^2 \quad \text{(since } \sigma_a^2 \text{ is defined as the variance of interarrival times)}$$

Since $\bar{I} = \frac{1-\rho}{\lambda}$ and $\bar{\tau} = \frac{1}{\lambda}$ we get

$$\sigma_I^2 \geq (1-\rho)^2 \sigma_a^2$$

By using Eq. (3.97), we then obtain

$$W \leq \frac{\lambda(\sigma_a^2 + \sigma_b^2)}{2(1-\rho)} - \frac{\lambda(1-\rho)\sigma_a^2}{2}$$

3.49

(a) Since the arrivals are Poisson with rate λ, the mean time until the next arrival starting from any given time (such as the time when the system becomes empty) is $1/\lambda$. The time average fraction of busy time is $\lambda E[X]$. This can be seen by Little's theorem applied to the service facility (the time average number of customers in the server is just the time average fraction of busy time), or it can be seen by letting $\sum_{i=1}^{n} X_i$ represent the time the server is busy with the first n customers, dividing by the arrival time of the n^{th} customer, and going to the limit.

Let E[B] be the mean duration of a busy period and $E[I] = 1/\lambda$ be the mean duration of an idle period. The time average fraction of busy time must be $E[B]/(E[B]+E[I])$. Thus

$$\lambda E[X] = E[B]/(E[B]+1/\lambda)\,; \qquad E[B] = \frac{E[X]}{1 - \lambda E[X]}$$

This is the same as for the FCFS M/G/1 system (Problem 3.30).

(b) If a second customer arrives while the first customer in a busy period is being served, that customer (and all subsequent customers that arrive while the second customer is in the system) are served before the first customer resumes service. The same thing happens for any subsequent customer that arrives while the first customer is actually in service. Thus when the first customer leaves, the system is empty. One can view the queue here as a stack, and the first customer is at the bottom of the stack. It follows that E[B] is the expected system time given a customer arriving to an empty system.

The customers already in the system when a given customer arrives receive no service until the given customer departs. Thus the system time of the given customer depends only on its own service time and the new customers that arrive while the given customer is in the system. Because of the memoryless property of the Poisson arrivals and the independence of service times, the system time of the given customer is independent of the number of customers (and their remaining service times) in the system when the given customer arrives. Since the expected system time of a given customer is independent of the number of customers it sees upon arrival in the system, the expected time is equal to the expected system time when the given customer sees an empty system; this is E[B] as shown above.

(c) Given that a customer requires 2 units of service time, look first at the expected system time until 1 unit of service is completed. This is the same as the expected system time of a customer requiring one unit of service (i.e., it is one unit of time plus the service time of all customers who arrive during that unit and during the service of other such customers). When one unit of service is completed for the given customer, the given customer is in service with one unit of service still required, which is the same as if a new customer arrived requiring one unit of service. Thus the given customer requiring 2 units of service has an expected system time of 2C. Extending the argument to a customer requiring n units of service, the expected system time is nC. Doing the argument backwards for a customer requiring 1/n of service, the expected system time is C/n. We thus conclude that $E[\text{system time} \mid X=x] = Cx$.

(d) We have

$$E[B] = \int_0^\infty Cx\, dF(x) = CE[X]; \quad C = \frac{1}{1 - \lambda E[X]}$$

3.50

(a) Since $\{p_j\}$ is the stationary distribution, we have for all $j \in S$

$$p_j\left(\sum_{i\in\bar{S}} q_{ji} + \sum_{i\in\bar{S}} q_{ji}\right) = \sum_{i\in\bar{S}} p_i q_{ij} + \sum_{i\in\bar{S}} p_i q_{ij}$$

Using the given relation, we obtain for all $j \in \bar{S}$

$$p_j \sum_{i\in\bar{S}} q_{ji} = \sum_{i\in\bar{S}} p_i q_{ij}$$

Dividing by $\sum_{i\in\bar{S}} p_i$, it follows that

$$\bar{p}_j \sum_{i\in\bar{S}} q_{ji} = \sum_{i\in\bar{S}} \bar{p}_i q_{ij}$$

for all $j \in \bar{S}$, showing that $\{\bar{p}_j\}$ is the stationary distribution of the truncated chain.

(b) If the original chain is time reversible, we have $p_j q_{ji} = p_i q_{ij}$ for all i and j, so the condition of part (a) holds. Therefore, we have $\bar{p}_j q_{ji} = \bar{p}_i q_{ij}$ for all states i and j of the truncated chain.

(c) The finite capacity system is a truncation of the two independent M/M/1 queues system, which is time reversible. Therefore, by part (b), the truncated chain is also time reversible. The formula for the steady state probabilities is a special case of Eq. (3.39) of Section 3.4.

3.51

(a) Since the detailed balance equations hold, we have

$$p_j q_{ji} = p_i q_{ij}$$

Thus for $i, j \in S_k$, we have

$$\frac{p_j}{u_k} q_{ji} = \frac{p_i}{u_k} q_{ij} \Leftrightarrow \pi_j q_{ji} = \pi_i q_{ij}$$

and it follows that the π_i, $i \in S_k$ satisfy the detailed balance equations. Also

$$\sum_{i \in S_k} \pi_i = \frac{\sum_{i \in S_k} p_i}{u_k} = \frac{u_k}{u_k} = 1$$

Therefore, $\{\pi_i \mid i \in S_k\}$ as defined above, is the stationary distribution of the Markov chain with state space S_k.

(b) Obviously

$$\sum_{k=1}^{K} u_k = \sum_{k=1}^{K} \sum_{j \in S_k} p_j = 1. \tag{1}$$

Also we have

$$u_k \tilde{q}_{km} = \sum_{\substack{j \in S_k \\ i \in S_m}} \pi_j q_{ji} u_k,$$

which in view of the fact $\pi_j u_k = p_j$ for $j \in S_k$, implies that

$$u_k \tilde{q}_{km} = \sum_{\substack{j \in S_k \\ i \in S_m}} p_j q_{ji} \tag{2}$$

and

$$u_m \tilde{q}_{mk} = \sum_{\substack{j \in S_m \\ i \in S_k}} \pi_j q_{ji} u_m = \sum_{\substack{j \in S_m \\ i \in S_k}} q_{ji} p_j = \sum_{\substack{i \in S_m \\ j \in S_k}} q_{ij} p_j.$$

(3)

Since the detailed balance equations $p_j q_{ji} = q_{ij} p_i$ hold, we have

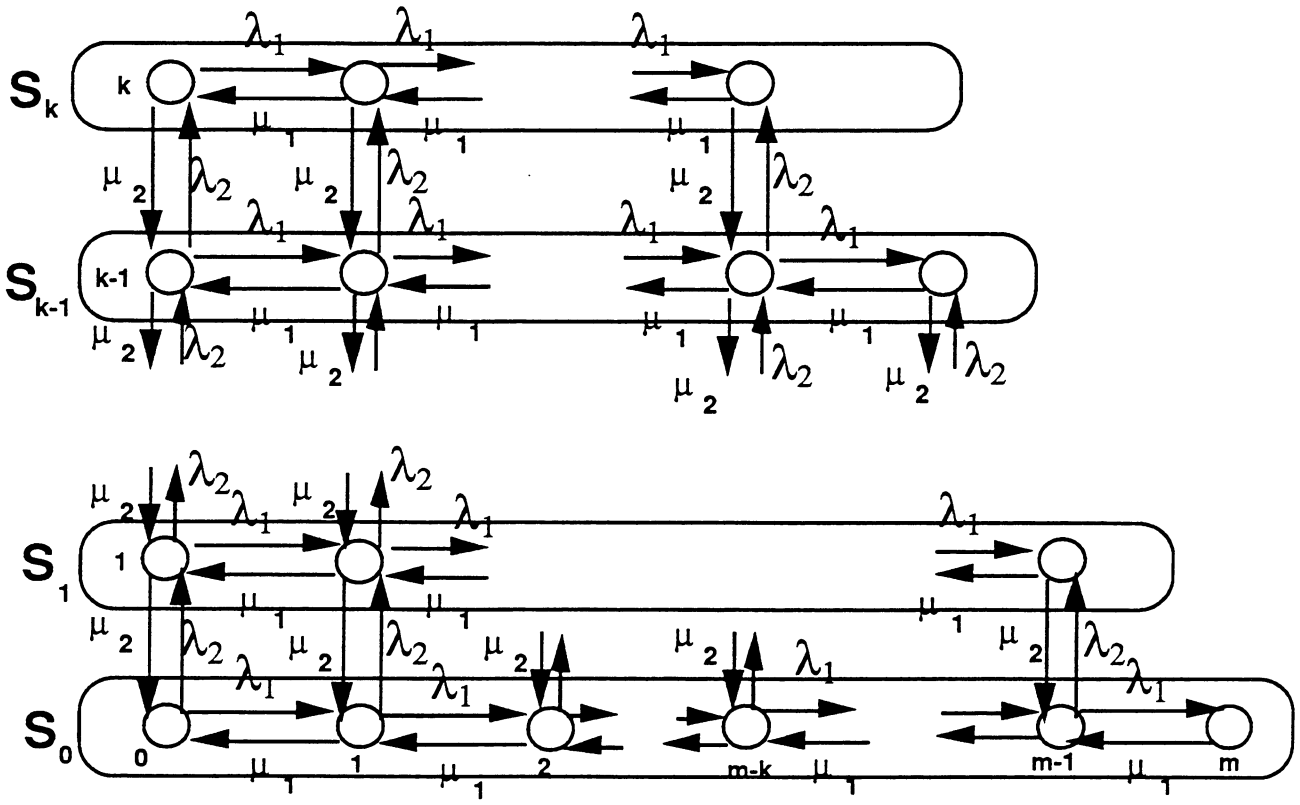
$$\sum_{\substack{j \in S_k \\ i \in S_m}} p_j q_{ji} = \sum_{\substack{i \in S_m \\ j \in S_k}} q_{ij} p_i \tag{4}$$

Equations (2)-(4) give

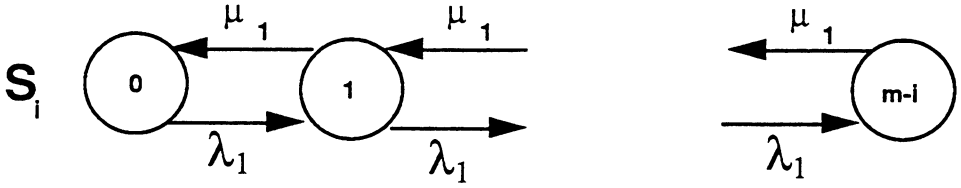
$$u_k \tilde{q}_{km} = u_m \tilde{q}_{mk}. \tag{5}$$

Equations (1) and (5) imply that $\{u_k \mid k=1,...,K\}$ is the stationary distribution of the Markov chain with states 1,...,k, and transition rates $\tilde{q}_{km}$.

(c) We will deal only with Example 3.13. (Example 3.12 is a special case with k=m).



For i =0,1,...,k we define S_i as the set of states (i,0), (i,1),..., (i,m-i), (see Figure 1). Then the truncated chain with state space S_i is



We denote by $\pi_j^{(i)}$ the stationary probability of state j of the truncated chain S_i and we let

$$\rho_1 = \frac{\lambda_1}{\mu_1}.$$

Then

$$\pi_{j+1}^{(i)} = \rho_1 \pi_j^{(i)}$$

Thus

$$\pi_0^{(i)} \sum_{j=0}^{m-i} \rho_1^j = 1.$$

or

$$\pi_0^{(i)} = \frac{1-p_1}{1-p_1^{m-i+1}}$$

Therefore,

$$\pi_j^{(i)} = \frac{1-\rho_1}{1-\rho_1^{m-i+1}} \rho_1^j \qquad i=0,1,2,\ldots,k, \quad j=0,1,2,\ldots,m-i$$

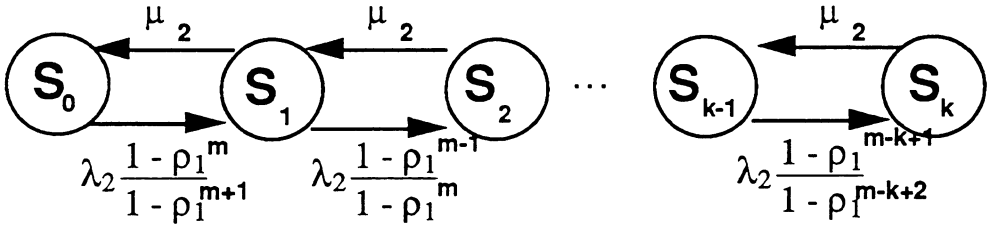
The transition probabilities of the aggregate chain are

$$\tilde{q}_{l,\,l+1} = \sum_{j\in S_l, i\in S_{l+1}} \pi_j^{(l)} q_{ji} = \sum_{j=0}^{m-l-1} \pi_j^{(l)} \lambda_2 = \lambda_2 (1-\pi_{m-l}^{(l)})$$

$$= \lambda_2 \left(1 - \frac{1-\rho_1}{1-\rho_1^{m-l+1}} \rho_1^{m-l}\right) = \lambda_2 \frac{1-\rho_1^{m-l}}{1-\rho_1^{m-l+1}}$$

$$\tilde{q}_{l+1,\,l} = \sum_{\substack{j\in S_{l+1} \\ i\in S_l}} \pi_j^{(l+1)} q_{ji} = \mu_2 \sum_{j\in S_{l+1}} \pi_j^{(l+1)} = \mu_2.$$

The aggregate chain is given by the following figure.



Thus we have

$$u_{l+1} = \rho_2 \cdot \frac{1-\rho_1^{m-l}}{1-\rho_1^{m-l+1}} u_l$$

from which

$$u_l = \rho_2^{l_2} \prod_{j=0}^{l-1} \left(\frac{1-\rho_1^{m-j}}{1-\rho_1^{m-j+1}} \right) = \rho_2 \frac{1-\rho_1^{m-l+1}}{1-\rho_1^{m+1}} u_0$$

Furthermore, we have

$$\sum_{l=0}^{k} u_l = 1,$$

or

$$u_0 \sum_{l=0}^{k} \rho_2^l \frac{1-\rho_1^{m-l+1}}{1-\rho_1^{m+1}} = 1$$

from which

$$u_0 = \frac{1-\rho_1^{m+1}}{\frac{1-\rho_2^{k+1}}{1-\rho_2} - \rho_1^{m+1} \cdot \frac{1-\left(\frac{\rho_2}{\rho_1}\right)^{k+1}}{1-\frac{\rho_2}{\rho_1}}}$$

and

$$u_l = u_0 \rho_2^l \frac{1-\rho_1^{m-l+1}}{1-\rho_1^{m+1}}$$

Thus

$$p(n_1 n_2) = \pi_{n_1}^{(n_2)} u_{n_2} = \frac{u_0 \rho_2^{n_2} (1-\rho_1) \rho_1^{n_1}}{1-\rho_1^{m+1}}$$

from which we obtain the product form

$$p(n_1 n_2) = \left(\frac{1-\rho_2^{k+1}}{1-\rho_2} - \rho_1^{m+1} \frac{1-\left(\frac{\rho_2}{\rho_1}\right)^{k+1}}{1-\frac{\rho_2}{\rho_1}} \right)^{-1} (1-\rho_1) \rho_1^{n_1} \rho_2^{n_2}$$

(d) We are given that the detailed balance equations hold for the truncated chains. Thus for $i, j \in S_k$, we have $p_i q_{ij} = p_j q_{ji}$. Furthermore,

$$\sum_{i \in S_k} \pi_i = \sum_{i \in S_k} \frac{\rho_i}{u_k} = \frac{u_k}{u_k} = 1$$

Thus $\{\pi_j \mid j \in S_k\}$ is the distribution for the truncated chain S_k and the result of part (a) holds.

To prove the result of part (b), we have to prove that the global balance equations hold for the aggregate chain, i.e.,

$$\sum_{m=1}^{k} \tilde{q}_{km} u_k = \sum_{m=1}^{k} u_m \tilde{q}_{mk},$$

or equivalently

$$\sum_{m=1}^{k} \sum_{j \in S_k} \sum_{i \in S_m} \pi_j u_k q_{ji} = \sum_{m=1}^{k} \sum_{j \in S_k} \sum_{i \in S_m} q_{ij} \pi_i u_m$$

For $j \in S_k$, we have $\pi_j u_k = p_j$, and for $i \in S_m$, we have $\pi_i u_m = p_i$, so we must show

$$\sum_{m=1}^{k} \sum_{j \in S_k} \sum_{i \in S_m} p_j q_{ji} = \sum_{m=1}^{k} \sum_{j \in S_k} \sum_{i \in S_m} q_{ij} p_i$$

or

$$\sum_{j \in S_k} \sum_{\text{all } i} p_j q_{ji} = \sum_{j \in S_k} \sum_{\text{all } i} p_i q_{ij} \tag{6}$$

Since $\{p_i\}$ is the distribution of the original chain, the global balance equations

$$\sum_{\text{all } i} p_j q_{ji} = \sum_{\text{all } i} p_i q_{ij}$$

By summing over all $j \in S_k$, we see that Eq. (6) holds. Since

$$\sum_{k=1} u_k = 1$$

and we just proved that the u_k's satisfy the global balance equations for the aggregate chain, $\{u_k \mid k = 1,...,K\}$ is the distribution of the aggregate chain. This proves the result of part (b).

3.52

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. In particular, $t_2 - t_1$ is independent of the (reversed system) arrival process to the left of t_2. In forward system terms, this means that $t_2 - t_1$ is independent of the departure process to the left of t_2.

3.53

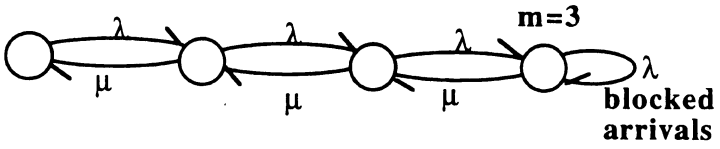
(a) If customers are served in the order they arrive then given that a customer departs at time t from queue 1, the arrival time of that customer at queue 1 (and therefore the time spent at queue 1), is independent of the departure process from queue 1 prior to t. Since the departures from queue 1 are arrivals at queue 2, we conclude that the time spent by a customer at queue 1 is independent of the arrival times at queue 2 prior to the customer's arrival at queue2. These arrival times, together with the corresponding independent (by Kleinrock's approximation) service times determine the time the customer spends at queue 2 and the departure process from queue 2 before the customer's departure from queue 2.

(b) Suppose packets with mean service time $1/\mu$ arrive at the tandem queues with some rate λ which is very small ($\lambda << \mu$). Then, the apriori probability that a packet will wait in queue 2 is very small.

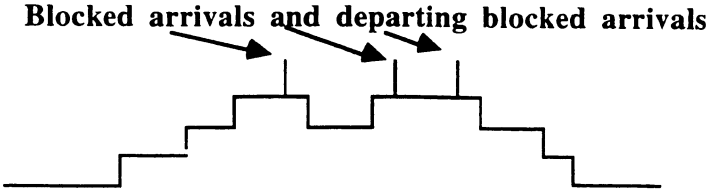
Assume now that we know that the waiting time of a packet in queue 1 was nonzero. This information changes the aposteriori probability that the packet will wait in queue 2 to at least 1/2 (because its service time in queue 1 will be less than the service time at queue 2 of the packet in front of it with probability 1/2). Thus the knowledge that the waiting time in queue 1 is nonzero, provides a lot of information about the waiting time in queue 2.

3.54

The Markov chain for an M/M/1/m system is



Since this is a birth/death process, the chain is reversible. If we include the arrivals that are blocked from the system, then the arrival process is Poisson (by definition of M/M/1/m). If we include the blocked arrivals also as departures, then the departure process is also Poisson (by reversibility).



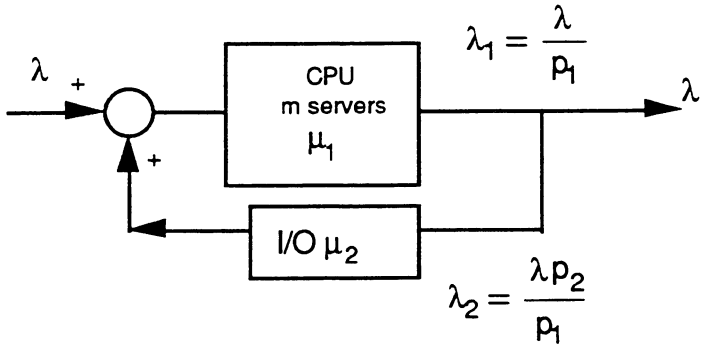
If we omit the blocked arrivals from consideration, the admitted arrival process has rate $\lambda(1-\rho_m)$, but this process is definitely not Poisson. One could, if one were truly masochistic, calculate things like the interarrival density, but the only sensible way to characterize the process is to characterize it jointly with the state process, in which case it is simply the process of arrivals during intervals when the state is less than m. The departure process, omitting the departures of blocked arrivals, is the same as the process of admitted arrivals, as can be seen by reversibility.

3.55

Let

$$\rho_1 = \frac{\lambda_1}{\mu_1}$$

$$\rho_2 = \frac{\lambda_2}{\mu_2}$$



Using Jackson's Theorem and Eqs. (3.34)-(3.35) we find that

$$P(n_1,n_2) = \begin{cases} p_0 \dfrac{(m\rho_1)^{n_1}}{n_1!} \cdot \rho_2^{n_2}(1-\rho_2), & n_1 \le m \\ p_0 \dfrac{m^m \rho_1^{n_1}}{m!}\, \rho_2^{n_2}(1-\rho_2), & n_1 > m \end{cases}$$

where

$$p_0 = \left[\sum_{n=0}^{m-1} \frac{(m\rho_1)^n}{n!} + \frac{(m\rho_1)^m}{m!(1-\rho_1)} \right]^{-1}$$

3.56

(a) We have

$$P(X_n=i) = (1-\rho)\rho^i; \quad i \ge 0; \quad \rho = \lambda/\mu$$

$$P(X_n=i, D_n=j) = P(D_n=j \mid X_n=i)\, P(X_n=i) = \begin{cases} \mu\Delta(1-\rho)\rho^i; & i \ge 1, j=1 \\ 0; & i=0, j=1 \\ (1-\mu\Delta)(1-\rho)\rho^i; & i \ge 1, j=0 \\ 1-\rho; & i=0, j=0 \end{cases}$$

(b) $P(D_n=1) = \sum_{i=1} \mu\Delta((1-\rho)\rho^i = \mu\Delta\rho = \lambda\Delta$

(c) $$P(X_n=i \mid D_n=1) = \frac{P(X_n=i, D_n=1)}{P(D_n=1)} = \begin{cases} \{\mu\Delta(1-\rho)\rho^i\}/\lambda\Delta = (1-\rho)\rho^{i-1}; & i \ge 1 \\ 0; & i=0 \end{cases}$$

(d) $P(X_{n+1}=i \mid D_n=1) = P(X_n=i+1 \mid D_n=1) = (1-\rho)\rho^i \;;\; i \geq 0$

In the first equality above, we use the fact that, given a departure between $n\Delta$ and $(n+1)\Delta$, the state at $(n+1)\Delta$ is one less than the state at $n\Delta$; in the second equality, we use part d). Since $P(X_{n+1}=i) = (1-\rho)\rho^i$, we see that X_{n+1} is statistically independent of the event $D_n=1$. It is thus also independent of the complementary event $D_n=0$, and thus is independent of the random variable D_n.

$$\begin{aligned} (e)\; P(X_{n+1}=i, D_{n+1}=j \mid D_n) &= P(D_{n+1}=j \mid X_{n+1}=i, D_n)P(X_{n+1}=i \mid D_n) \\ &= P(D_{n+1}=j \mid X_{n+1}=i)P(X_{n+1}=i) \end{aligned}$$

The first part of the above equality follows because X_{n+1} is the state of the Markov process at time $(n+1)\Delta$, so that, conditional on that state, D_{n+1} is independent of everything in the past. The second part of the equality follows from the independence established in e). This establishes that X_{n+1}, D_{n+1} are independent of D_n; thus their joint distribution is given by b).

(f) We assume the inductive result for k-1 and prove it for k; note that part f establishes the result for k=1. Using the hint,

$$\begin{aligned} P(X_{n+k}=i \mid D_{n+k-1}=1, D_{n+k-2},\ldots, D_n) &= P(X_{n+k-1}=i+1 \mid D_{n+k-1}=1, D_{n+k-2},\ldots, D_n) \\ &= \frac{P(X_{n+k-1}=i+1,\, D_{n+k-1}=1 \mid D_{n+k-2},\ldots, D_n)}{P(D_{n+k-1}=1 \mid D_{n+k-2},\ldots, D_n)} \\ &= \frac{P(X_{n+k-1}=i+1, D_{n+k-1}=1)}{P(D_{n+k-1}=1)} \\ &= P(X_{n+k-1}=i+1 \mid D_{n+k-1}=1) = P(X_{n+k}=i \mid D_{n+k-1}=1) \end{aligned}$$

The third equality above used the inductive hypothesis for the independence of the pair (X_{n+k-1}, D_{n+k-1}) from $D_{n+k-2},\ldots D_n$ in the numerator and the corresponding independence of D_{n+k-1} in the denominator. From part e), with n+k-1 replacing n, $P(X_{n+k}=i \mid D_{n+k-1}) = P(X_{n+k}=i)$, so

$$P(X_{n+k}=i \mid D_{n+k-1}=1, D_{n+k-2},\ldots, D_n) = P(X_{n+k})$$

Using the argument in e), this shows that conditional on $D_{n+k-2},\ldots, D_n$, the variable X_{n+k} is independent of the event $D_{n+k-1}=1$ and thus also independent of $D_{n+k-1}=0$. Thus X_{n+k} is independent of $D_{n+k-1},\ldots, D_n$. Finally,

$$\begin{aligned} P(X_{n+k}=i, D_{n+k}=j \mid D_{n+k-1},\ldots, D_n) &= P(D_{n+k}=j \mid X_{n+k}=i)P(X_{n+k}=i \mid D_{n+k-1},\ldots, D_n) \\ &= P(D_{n+k}=j \mid X_{n+k}=i)P(X_{n+k}=i) \end{aligned}$$

which shows the desired independence for k.

(g) This shows that the departure process is Bernoulli and that the state is independent of past departures; i.e., we have proved the first two parts of Burke's theorem without using

reversibility. What is curious here is that the state independence is critical in establishing the Bernoulli property.

3.57

The session numbers and their rates are shown below:

Session	Session number p	Session rate x_p
ACE	1	100/60 = 5/3
ADE	2	200/60 = 10/3
BCEF	3	500/60 = 25/3
BDEF	4	600/60 = 30/3

The link numbers and the total link rates calculated as the sum of the rates of the sessions crossing the links are shown below:

Link	Total link rate
AC	$x_1 = 5/3$
CE	$x_1 + x_3 = 30/3$
AD	$x_2 = 10/3$
BD	$x_4 = 10$
DE	$x_2 + x_4 = 40/3$
BC	$x_3 = 25/3$
EF	$x_3 + x_4 = 55/3$

For each link (i,j) the service rate is

$$\mu_{ij} = 50000/1000 = 50 \text{ packets/sec},$$

and the propagation delay is $D_{ij} = 2 \times 10^{-3}$ secs. The total arrival rate to the system is

$$\gamma = \Sigma_i x_i = 5/3 + 10/3 + 25/3 + 30/3 = 70/3$$

The average number on each link (i, j) (based on the Kleinrock approximation formula) is:

$$N_{ij} = \frac{\lambda_{ij}}{\mu_{ij} - \lambda_{ij}} + \lambda_{ij} D_{ij}$$

From this we obtain:

Link	Average Number of Packets on the Link
AC	(5/3)/(150/3 - 5/3) + (5/3)(2/1000) = 5/145 + 1/300
CE	1/4 + 1/50
AD	1/14 + 1/150
BD	1/4 + 1/50

DE	4/11 + 2/75
BC	1/5 + 1/60
EF	11/19 + 11/300

The average total number in the system is $N = \Sigma_{(i,j)} N_{ij} \equiv 1.84$ packet. The average delay over all sessions is $T = N/\gamma = 1.84 \times (3/70) = 0.0789$ secs. The average delay of the packets of an individual session are obtained from the formula

$$T_p = \sum_{(i,j) \text{ on } p} \left[\frac{\lambda_{ij}}{\mu_{ij}(\mu_{ij} - \lambda_{ij})} + \frac{1}{\mu_{ij}} + D_{ij} \right]$$

For the given sessions we obtain applying this formula

Session p	**Average Delay T_p**
1	0.050
2	0.053
3	0.087
4	0.090

3.58

We convert the system into a closed network with M customers as indicated in the hint. The (k+1)st queue corresponds to the "outside world". It is easy to see that the queues of the open systems are equivalent to the first k queues of the closed system. For example, when there is at least one customer in the (k+1)st queue (equivalently, there are less than M customers in the open system) the arrival rate at queue i is

$$\sum_{m=1}^{k} r_m \frac{r_i}{\sum_{j=1}^{k} r_j} = r_i .$$

Furthermore, when the (k+1)st queue is empty no external arrivals can occur at any queue i, i = 1,2,...,k. If we denote with $p(n_1,...,n_k)$ the steady state distribution for the open system, we get

$$p(n_1,n_2,...,n_k) = \begin{cases} 0 & \text{if } \sum_{i=1}^{k} n_i > M \\ \dfrac{\rho_1^{n_1} \rho_2^{n_2} \cdots \rho_k^{n_k} \rho_{k+1}^{\left(M - \sum_{i=1}^{k} n_i\right)}}{G(M)} & \text{otherwise} \end{cases}$$

where

$$\rho_i = \frac{r_i}{\mu},\ i = 1,2,\ldots,k,$$

$$\rho_{k+1} = \frac{\sum_{i=1}^{k} r_i(1-\sum_{j=1}^{k} p_{ij})}{\sum_{i=1}^{k} r_i}$$

and G(M) is the normalizing factor.

3.59

If we insert a very fast M/M/1 queue ($\mu \to \infty$) between a pair of queues, then the probability distribution for the packets in the rest of the queues is not affected. If we condition on a single customer being in the fast queue, since this customer will remain in this queue for $1/\mu$ ($\to 0$) time on the average, it is equivalent to conditioning on a customer moving from one queue to the other in the original system.

If $P(n_1,\ldots,n_k)$ is the stationary distribution of the original system of k queues and $P'(n_1,\ldots,n_k,n_{k+1})$ is the corresponding probability distribution after the insertion of the fast queue k+1, then

$$P(n_1,\ldots,n_k \mid \text{arrival}) = P'(n_1,\ldots,n_k,\ n_{k+1} = 1 \mid n_{k+1} = 1),$$

which by independence of $n_1, \ldots, n_k, n_{k+1}$, is equal to $P(n_1,\ldots,n_k)$.

3.60

Let U_j = utility function of j^{th} queue.

We have to prove that

$$\lim_{M\to\infty}(U_j(M)) = \lim_{M\to\infty}\frac{\lambda_j(M)}{\mu_j} = 1$$

But from problem 3.65 we have

$$U_j(M) = \rho_j \frac{G(M-1)}{G(M)}.$$

Thus it is enough to prove that

$$\lim_{M\to\infty}\frac{G(M)}{G(M-1)} = \rho_j$$

where $\rho_j = \max\{\rho_1,..., \rho_k\}$. We have

$$G(M) = \sum_{n_1+\ldots+n_k=M} \rho_1^{n_1}\ldots\rho_j^{n_j}\ldots\rho_k^{n_k} = \sum_{\substack{n_1+\ldots+n_k=M \\ n_j=0}} \rho_1^{n_1}\ldots\rho_j^{n_j}\ldots\rho_k^{n_k} + \sum_{\substack{n_1+\ldots+n_k=M \\ n_j\neq 0}} \rho_1^{n_1}\ldots\rho_j^{n_j}\ldots\rho_k^{n_k}$$

$$= A(M) + B(M) \qquad (1)$$

Since $\rho_j = \max\{\rho_1,...,\rho_k\}$ we have that

$$\lim_{M\to\infty}\frac{A(M)}{B(M)} = 0.$$

Thus, Eq. (1) implies that

$$\lim_{M\to\infty}\frac{\sum_{\substack{n_1+\ldots+n_k=M \\ n_j\neq 0}} \rho_1^{n_1}\ldots\rho_j^{n_j}\ldots\rho_k^{u_k}}{G(M)} = 1$$

or, denoting $n'_j = n_j - 1$,

$$\lim_{M\to\infty}\frac{\rho_j \sum_{n_1+\ldots+n'_j+\ldots+n_k=M-1} \rho_1^{n_1}\ldots\rho_j^{n'_j}\ldots\rho_k^{n_k}}{G(M)} = 1$$

or

$$\lim_{M\to\infty}\frac{\rho_j\, G(M-1)}{G(M)} = 1$$

3.61

We have $\sum_{i=0}^{m} p_i = 1$.

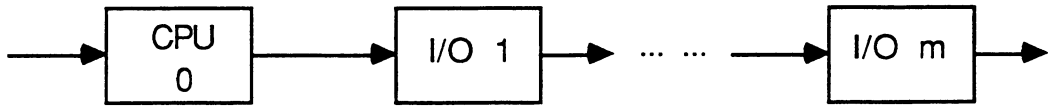
The arrival rate at the CPU is λ/p_0 and the arrival rate at the i^{th} I/O port is $\lambda p_i/p_0$. By Jackson's Theorem, we have

$$P(n_0, n_1, \ldots, n_m) = \prod_{i=0}^{m} \rho_i^{n_i} (1 - \rho_i)$$

where $\rho = \dfrac{\lambda}{\mu_0 p_0}$

and $\rho_i = \dfrac{\lambda p_i}{\mu_i p_0}$ for $i > 0$

The equivalent tandem system is as follows:



The arrival rate is λ. The service rate for queue 0 is $\mu_0 p_0$ and for queue i ($i > 0$) is $\mu_i p_0 / p_i$.

3.62

Let λ_0 be the arrival rate at the CPU and let λ_i be the arrival rate at I/O unit i. We have

$$\lambda_i = p_i \lambda_0, \quad i = 1,\ldots,m.$$

Let

$$\bar{\lambda}_0 = 1, \quad \bar{\lambda}_i = p_i, \quad i = 1,\ldots,m,$$

and

$$\rho_i = \frac{\bar{\lambda}_i}{\mu_i}, \quad i = 0,1,\ldots,m,$$

By Jackson's Theorem, the occupancy distribution is

$$P(n_0,n_1,\ldots,n_m) = \frac{\rho_0^{n_0}\rho_1^{n_1}\ldots\rho_m^{n_m}}{G(M)},$$

where G(M) is the normalization constant corresponding to M customers,

$$G(M) = \sum_{n_0+n_1+\ldots+n_m=M} \rho_0^{n_0}\rho_1^{n_1}\ldots\rho_m^{n_m}$$

Let

$$U_0 = \frac{\lambda_0}{\mu_0}$$

be the utilization factor of the CPU. We have

$$U_0 = P(n_0 \geq 1) = \sum_{\substack{n_0+n_1+\ldots+n_m=M \\ n_0 \geq 1}} P(n_0, n_1, \ldots, n_m)$$

$$= \sum_{n'_0+n_1+\ldots+n_m=M-1} \frac{\rho_0^{n'_0+1} \rho_1^{n_1} \ldots \rho_m^{n_m}}{G(M)}$$

$$= \rho_0 \frac{G(M-1)}{G(M)} = \frac{1}{\mu_0} \frac{G(M-1)}{G(M)},$$

where we used the change of variables $n'_0 = n_0 - 1$. Thus the arrival rate at the CPU is

$$\lambda_0 = \frac{G(M-1)}{G(M)}$$

and the arrival rate at the I/O unit i is

$$\lambda_i = \frac{p_i G(M-1)}{G(M)}, \quad i=1,\ldots,m.$$

3.63

(a) We have $\lambda = N/T$ and

$$T = T_1 + T_2 + T_3$$

where

T_1 = Average time at first transmission line
T_2 = Average time at second transmission line
$T_3 = \bar{Z}$

We have

$$\bar{X} \leq T_1 \leq N\bar{X} \qquad (1)$$

$$\bar{Y} \leq T_2 \leq N\bar{Y}$$

so

$$\frac{N}{N(\bar{X}+\bar{Y})+\bar{Z}} \le \lambda \le \frac{N}{\bar{X}+\bar{Y}+\bar{Z}} .$$

Also

$$\lambda \le \frac{K}{\bar{X}}, \qquad \lambda \le \frac{1}{\bar{Y}}$$

so finally

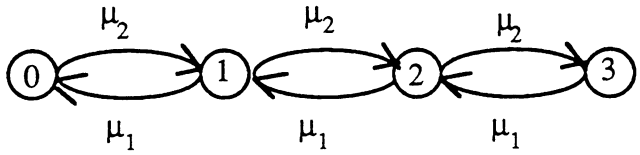
$$\frac{N}{N(\bar{X}+\bar{Y})+\bar{Z}} \le \lambda \le \min\left\{\frac{K}{\bar{X}}, \frac{1}{\bar{Y}}, \frac{N}{\bar{X}+\bar{Y}+\bar{Z}}\right\}$$

(b) The same line of argument applies except that in place of (1) we have

$$\bar{X} \le T_1 \le (N - K + 1)\bar{X}$$

3.64

(a) The state is determined by the number of customers at node 1 (one could use node 2 just as easily). When there are customers at node 1 (which is the case for states 1, 2, and 3), the departure rate from node 1 is μ_1; each such departure causes the state to decrease as shown below. When there are customers in node 2 (which is the case for states 0, 1, and 2), the departure rate from node 2 is μ_2; each such departure causes the state to increase.



(b) Letting p_i be the steady state probability of state i, we have $p_i = p_{i-1}\,\rho$, where $\rho = \mu_2/\mu_1$. Thus $p_i = p_0\rho^i$. Solving for p_0,

$$p_0 = [1 + \rho + \rho^2 + \rho^3]^{-1}, \quad p_i = p_0\,\rho^i; \quad i=1,2,3.$$

(c) Customers leave node 1 at rate μ_1 for all states other than state 0. Thus the time average rate r at which customers leave node 1 is $\mu_1(1-P_0)$, which is

$$r = \frac{\rho+\rho^2+\rho^3}{1+\rho+\rho^2+\rho^3}\,\mu_1$$

(d) Since there are three customers in the system, each customer cycles at one third the rate at which departures occur from node 1. Thus a customer cycles at rate r/3.

(e) The Markov process is a birth-death process and thus reversible. What appears as a departure from node i in the forward process appears as an arrival to node i in the backward

process. If we order the customers 1, 2, and 3 in the order in which they depart a node, and note that this order never changes (because of the FCFS service at each node), then we see that in the backward process, the customers keep their identity, but the order is reversed with backward departures from node i in the order 3, 2, 1, 3, 2, 1, ...

3.65

Since $\mu_j(m) = \mu_j$ for all m, and the probability distribution for state $n = (n_1,...,n_k)$ is

$$P(n) = \frac{\rho_1^{n_1} \cdots {}_k^{n_k}}{G(M)},$$

where

$$\rho_j = \frac{\bar{\lambda}_j}{\mu_j}$$

The utilization factor $U_j(M)$ for queue j is

$$U_j(M) = \sum_{\substack{n_1+...+n_k=M \\ \text{s.t. } n_j \neq 0}} P(n) = \frac{\sum_{\substack{n_1+...+n_k=M \\ n_j \neq 0}} \rho_1^{n_1} ... \rho_k^{n_k}}{G(M)},$$

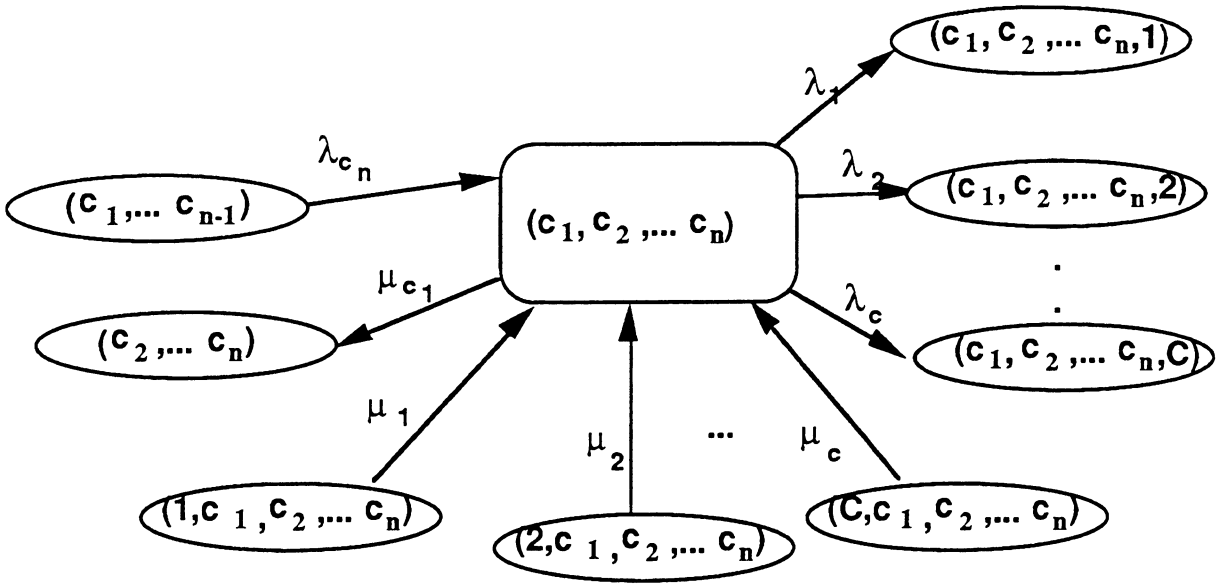
Denoting $n'_j = n_j - 1$, we get

$$U_j(M) = \frac{\sum_{n_1+...+n'_j+...+n_k=M-1} \rho_1^{n_1} ... \rho_j^{n'_1} ... \rho_k^{n_k} \cdot \rho_j}{G(M)} = \frac{\rho_j G(M-1)}{G(M)}.$$

3.66

Let c_i indicate the class of the i^{th} customer in queue.

We consider a state $(c_1,c_2,...,c_n)$ such that $\mu_{c_1} \neq \mu_{c_n}$.



If the steady-state distribution has a product form then the global balance equations for this state give

$$p(c_1)\cdots p(c_n)(\mu_{c_1} + \lambda_1 + \cdots + \lambda c) = p(c_1)\cdots p(c_{n-1})\lambda c_n + (\mu_1 p(1) + \mu_2 p(2) + \cdots + \mu_c p(c))\, p(c_1)\cdot\ldots\cdot p(c_n)$$

or

$$p(c_n)(\mu c_1 + \lambda_1 + \cdots + \lambda_c) = \lambda c_n + (\mu_1 p(1) + \mu_2 p(2) + \cdots + \mu_c p(c))\cdot p(c_n)$$

Denote

$$M = \mu_1 p(1) + \mu_2 p(2) + \cdots + \mu_c p(c)$$

$$\lambda = \lambda_1 + \cdots + \lambda_c.$$

Then

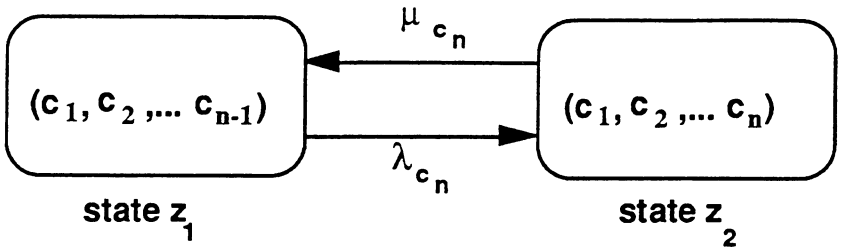
$$p(c_n)(\mu_{c1} + \lambda) = \lambda_{cn} + M\cdot p(c_n)$$

or

$$p(c_n) = \frac{\lambda_{c_n}}{\mu_{c_1} + \lambda - M}$$

This is a contradiction because even if $\lambda_{c_1} = 0$, $p(c_n)$ still depends on μ_{c_1}. The contradiction gives that $\mu_{c_1} = \mu$ = constant for every class. Thus we can model this system by a Markov chain only if $\mu_1 = \mu_2 = \cdots = \mu_c$.

(b) We will prove that the detailed balance equations hold. Based on the following figure



the detailed balance equations are

$$\lambda_{c_n} \cdot p(z_1) = \mu_{c_n} \cdot p(z_2)$$

or

$$\lambda_{c_n} \rho_{c_1} \ldots \rho_{c_{n-1}} = \mu_{c_n} \rho_{c_1} \ldots \rho_{c_n},$$

which obviously holds.

CHAPTER 4 SOLUTIONS

4.1

a) State n can only be reached from states 0 to n+1, so, using Eq. (3A.1),

$$p_n = \sum_{i=0}^{n+1} p_i P_{in} \quad ; \qquad 0 \le n < m$$

$$\sum_{n=0}^{m} p_n = 1$$

b) Solving for the final term, p_{n+1}, in the first sum above,

$$p_{n+1} = \frac{p_n(1-P_{nn}) - \sum_{i=0}^{n-1} p_i P_{in}}{P_{n+1,n}}$$

c) $p_1 = p_0(1-P_{00})/P_{10}$

$$p_2 = \frac{p_1(1-P_{11})-p_0P_{02}}{P_{21}} = p_0\left[\frac{(1-P_{00})(1-P_{11})}{P_{10}P_{21}} - \frac{P_{02}}{P_{21}}\right]$$

d) Combining the above equations with $p_0+p_1+p_2 = 1$, we get

$$p_0 = \frac{P_{10}P_{21}}{P_{10}P_{21}+(1-P_{00})P_{21}+(1-P_{00})(1-P_{11})-P_{02}P_{10}}$$

4.2

a) $P_{succ} = Q_a(1,n)Q_r(0,n) + Q_a(0,n)Q_r(1,n)$

Using Eqs(4.1) and (4.2),

$$P_{succ} = [(m-n)(1-q_a)^{m-n-1}q_a](1-q_r)^n + (1-q_a)^{m-n}[n(1-q_r)^{n-1}q_r]$$

$$= (1-q_a)^{m-n}(1-q_r)^n\left[(m-n)\frac{q_a}{1-q_a} + n\frac{q_r}{1-q_r}\right]$$

b) Approximate $q_a/(1-q_a)$ by q_a in the bracketed espression above. This is a good approximation for q_a small, whereas we cannot similarly approximate $(1-q_a)^{m-n}$ by 1 since m-n might be large. We also approximate $q_r/(1-q_r)$ by q_r. Using the approximation $(1-x)^y \approx e^{-xy}$, we then get

$$P_{succ} \approx \exp[-(m-n)q_a - nq_r]\ \{(m-n)q_a + nq_r\} = G(n)e^{-G(n)}$$

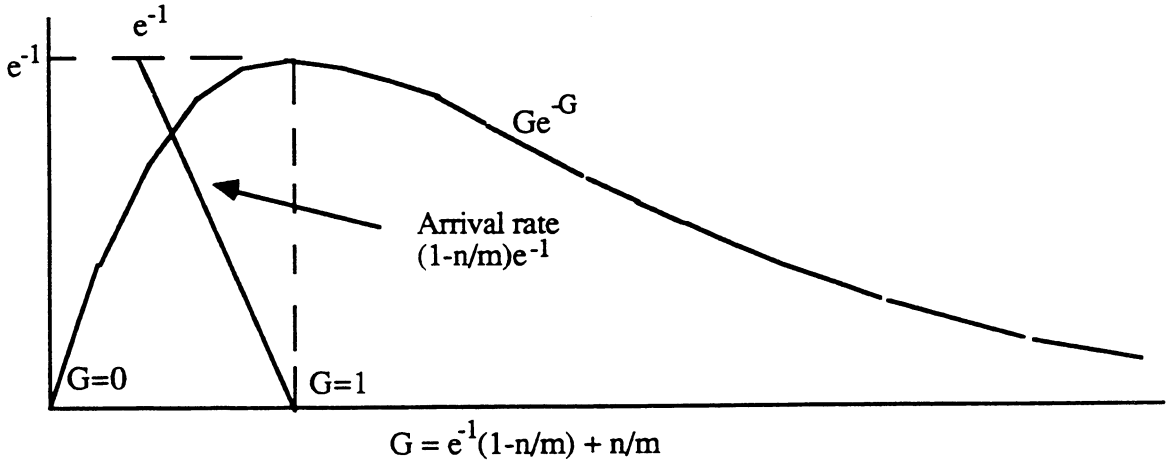
c) $(1-x)^y = \exp[y \ln(1-x)] = \exp[y\,(-x - x^2/2 - x^3/3 \ldots)]$

$= \exp(-yx)\exp[-x^2y/2 - x^3y/3 \ldots]$

which is equivalent to the desired relation. Note that for the approximation to be close (in terms of percentage error), x^2y must be close to 0.

4.3

a)



b) $P_{succ} = mq_r(1-q_r)^{m-1} \approx 1/e$

where we have used $mq_r = 1$ and the result of problem 4.2.

c) Note that a straight line from (mq_a, mq_a) to the point (1, 0) can only cross the curve Ge^{-G} once. For $mq_a \leq 1$ this crossing corresponds to a stable point by the argument in fig. 4.4. For $mq_a > 1$, we have $q_a > q_r$; when the departure rate exceeds the arrival rate, then n tends to decrease, which corresponds to motion toward the right on the figure. Thus the crossing corresponds to a stable point in this case also.

d) The stable point occurs at the intersection of the curve and straight line above, i.e. where

$$Ge^{-G} = (1-G)/(e-1)$$

Solving numerically, G = 0.4862.

e) Solving for n/m from G, we have n/m = (eG-1)/(e-1) = 0.1872. Thus at equilibrium a rather large fraction of the arriving packets (i.e., 0.1872) are not accepted by the system.

4.4

a) Let E{n} be the expected number of backlogged nodes, averaged over time. Since m-E{n} is the expected number of nodes that can accept packets, and q_a is the probability that each receives a packet in a slot, the expected number of accepted arrivals per slot is

$$E\{N_a\} = q_a(m-E\{n\})$$

b) Since a limited number (i.e., m) arrivals can be in the system at any time, the time average accepted arrival rate must equal the time average departure rate, which is the time average success rate, $E\{P_{succ}\}$. Thus

$$E\{P_{succ}\} = E\{N_a\} = q_a(m-E\{n\})$$

c) The expected number of packets in the system, $E\{N_{sys}\}$ immediately after the beginning of a slot is the expected backlog, $E\{n\}$, plus the expected number of arrivals accepted during the previous slot, $E\{N_a\}$. Thus,

$$E\{N_{sys}\} = E\{n\} + E\{N_a\} = E\{n\}(1-q_a) + q_a m$$

d) From Little's theorem, the expected delay T is $E\{N_{sys}\}$ divided by the accepted arrival rate $E\{N_a]\}$ Note that we are only counting the delay of the packets accepted into the system and note also that we are regarding accepted arrivals as arriving discretely at the slot boundaries.

$$T = E\{N_{sys}\}/E\{N_a\} = 1 + E\{n\}/[q_a(m-E\{n\})]$$

e) The above equations express the relevant quantities in terms of $E\{n\}$ and make clear that $E\{N_a\}$ and $E\{P_{succ}\}$ decrease and $E\{N_{sys}\}$ and T increase as n is decreased. Thus it makes no difference which of these quantities is optimized; improving one improves the others.

4.5

a) The probability that a packet is successful on the first slot is p, and given that it has not been successful before the ith slot, the probability that it is successful there is pq_r, i.e., the probability of retransmission times the probability of success. Thus the unconditioned probability of success on the second slot is $(1-p)pq_r$. Similarly, the probability of success on the third slot is $(1-p)(1-pq_r)pq_r$, and in general on the ith slot, i>2, is $(1-p)(1-pq_r)^{i-2}pq_r$. Thus, multiplying each term above by i and summing,

$$T = p + \sum_{i=2}^{\infty}(1-p)(1-pq_r)^{i-2}pq_r = 1 + \frac{1-p}{pq_r}$$

The solution to problem 2.17 b shows how to sum the above series.

b) The probability that a given packet transmission is successful is the probability that no other packets are transmitted in the same slot. If the given transmitted packet is a backlogged packet, then

$$p = (1-q_a)^{m-n}(1-q_r)^{n-1}$$

Approximating $(1-x)^k$ by e^{-kx} and approximating $(1-x)^{k-1}$ also by e^{-kx}, we get

$$p \approx e^{-G(n)} \; ; \quad G(n) = (m-n)q_a + nq_r$$

If the given transmitted packet is a new arrival, then p changes to $(1-q_a)^{m-n-1}(1-q_r)^n$, but the final result with the above approximation is the same. See the solution to problem 4.2 for a more complete discussion of these approximations.

c) Letting $G = G(n^*)$ and substituting e^{-G} for p in the solution to a),

$$T = 1 + (1-e^{-G})/(q_r e^{-G}) = 1 + (e^G-1)/q_r$$

Since $G = (m-n^*)q_a + n^*q_r$ and $Ge^{-G} = (m-n^*)q_a$, the ratio of these equations yields

$$e^G = 1 + \frac{n^*q_r}{(m-n^*)q_a}$$

$$T = 1 + \frac{n^*}{(m-n^*)q_a}$$

d) The two equations above relating G and n^* can be solved simultaneously (numerically) to yield $n^*/m = 0.124... \approx 1/8$. Using this in the equation for T above yields $1 + 0.472m$. Thus, $T \approx m/2$. We can compare this with TDM in two ways. First, if only one packet can be saved at a node, then a fraction $1-e^{-0.3} = 0.259...$ of the slots carry packets, so a fraction 0.136 of the arriving packets are discarded and the delay is roughly m/2 (slightly larger if the latest arrival for a node is discarded when a packet is already there, and slightly less if the later arrival is kept and the earlier thrown away). Alternatively, if no packets are thrown away, then the delay (from Eq. (3.58)) is m/1.4. Whichever way one looks at it, slotted Aloha does not look attractive from a delay standpoint if one achieves stability by choosing $q_r m = 1$.

4.6

a) Substituting Eqs. (4.1) and (4.2) into (4.5),

$$P_{succ} = (m-n)q_a(1-q_a)^{m-n-1}(1-q_r)^n + nq_r(1-q_a)^{m-n}(1-q_r)^{n-1}$$

Differentiating this with respect to q_r (for $n > 1$) and consolidating terms, we get

$$\frac{\partial P_{succ}}{\partial q_r} = n(1-q_a)^{m-n}(1-q_r)^{n-1}\left[\frac{1}{1-q_r} - \frac{q_a(m-n)}{1-q_a} - \frac{q_r n}{1-q_r}\right]$$

The quantity inside brackets is decreasing in q_r; it is positive for $q_r = 0$ and negative as q_r approaches 1. Thus there is a point at which this quantity is 0 and that point maximizes P_{succ}.

b) If we set q_r equal to q_a in the bracketed quantity above, it becomes $(1-q_a m)/(1-q_r)$. This is positive under the assumption that $q_a < 1/m$. Thus, since the quantity in brackets is decreasing in q_r, it is zero for $q_r > q_a$.

c)

$$\frac{dP_{succ}}{dq_a} = \frac{\partial P_{succ}}{\partial q_a} + \frac{dq_r(q_a)}{dq_a}\frac{\partial P_{succ}}{\partial q_r} = \frac{\partial P_{succ}}{\partial q_a}$$

The above relation follows because $\partial P_{succ}/\partial q_r = 0$ at $q_r(q_a)$. We then have

$$\frac{\partial P_{succ}}{\partial q_a} = n(1-q_a)^{m-n-1}(1-q_r)^n\left[\frac{1}{1-q_a} - \frac{q_a(m-n)}{1-q_a} - \frac{q_r n}{1-q_r}\right]$$

Note that the bracketed term here differs from the bracketed term in part a) only in the first term. Since the bracketed term in part a) is 0 at $q_r(q_a)$ and $q_r(q_a) > q_a$, it follows that the bracketed term here is negative. Thus the total derivative of P_{succ} with respect to q_a is negative.

d) If arrivals are immediately regarded as backlogged, then an unbacklogged node generates a transmission with probability $q_a q_r$. Thus the probability of success is modified by replacing q_a with $q_a q_r$. This reduces the value of q_a in P_{succ} and therefore, from part c), increases the value of P_{succ} at the optimum choice of q_r.

4.7

a) Note that one packet successfully leaves the system each slot in which one or more packets are transmitted. Thus if all waiting packets attempt transmission in every slot, a successful transmission occurs in every slot in which packets are waiting. Since the expected delay is independent of the order in which packets are successfully transmitted (since each packet requires one slot), we see that the expected delay is the same as that of a centralized slotted FCFS system. Now compare this policy with an arbitrary policy for transmitting waiting packets; assume any given sequence of packet arrival times. Each time the arbitrary policy fails to attempt a transmission in a slot with waiting packets, the FCFS system (if it has waiting packets) decreases the backlog by 1 while the other policy does not decrease the backlog. Thus the backlog for the arbitrary system is always greater than or equal to that of the FCFS system (a formal proof of this would follow by induction on successive slots). Thus, by Little's relation, the arbitrary system has an expected delay at least as great as the FCFS system.

b) This is just the slotted FDM system of section 3.5.1 with m=1 (i.e., a slotted M/D/1 queueing system). From Eq. (3.58), the queueing delay is $1/[2(1-\lambda)]$ slot times. The total delay, including service time, is then $1 + 1/[2(1-\lambda)]$.

c) The solution to b) can be rewritten as $1 + 1/2 + \lambda/[2(1-\lambda)]$ where the first term is the transmission time (i.e., 1 slot), the second term is the waiting time from an arrival to the beginning of a slot, and the third term is the delay due to collisions with other packets. If each subsequent attempt after an unsuccessful attempt is delayed by k slots, this last term is multiplied by k. Thus the new total delay is $3/2 + k\lambda/[(1-\lambda)]$.

4.8

a) Let X be the time in slots from the beginning of a backlogged slot until the completion of the first success at a given node. Let $q = q_r p$ and note that q is the probability that the node will be successful at any given slot given that it is still backlogged. Thus

$$P\{X=i\} = q(1-q)^{i-1} \; ; \; i \geq 1$$

$$E\{X\} = \sum_{i=1}^{\infty} iq(1-q)^{i-1} = \frac{1}{q}$$

The above summation uses the identity

$$\sum_{i=1}^{\infty} iz^{i-1} = \sum_{i=1}^{\infty} \frac{dz^i}{dz} = \frac{d\sum z^i}{dz} = \frac{d[z/(1-z)]}{dz} = \frac{1}{(1-z)^2}$$

Taking q = 1-z gives the desired result. A similar identity needed for the second moment is

$$\sum_{i=1}^{\infty} i^2 z^{i-1} = \sum_{i=1}^{\infty} \frac{d^2 z^{i+1}}{dz^2} - \sum_{i=1}^{\infty} \frac{dz^i}{dz} = \frac{d^2[z^2/(1-z)]}{dz^2} - \frac{1}{(1-z)^2} = \frac{1+z}{(1-z)^3}$$

Using this identity with q=1-z, we have

$$E\{X^2\} = \sum_{i=1}^{\infty} i^2 q(1-q)^{i-1} = \frac{2-q}{q^2} = \frac{2-pq_r}{(pq_r)^2}$$

b) For an individual node, we have an M/G/1 queue with vacations. The vacations are deterministic with a duration of 1 slot, and the service time has the first and second moments found in part a). Thus, using Eq. (3.55) for the queueing delay and adding an extra service time to get the system delay,

$$T = \frac{(\lambda/m)E\{X^2\}}{2(1-\rho)} + \frac{1}{2} + \frac{1}{q} = \frac{\lambda(2-\rho)}{2q^2(1-\rho)m} + \frac{1}{2} + \frac{1}{q}$$

Since the arrival rate is λ/m and the service rate is q, we have $\rho = \lambda/(mq)$. Substituting this into the above expression for T and simplifying,

$$T = \frac{1}{q_r p(1-\rho)} + \frac{1-2\rho}{2(1-\rho)}$$

c) For p=1 and $q_r=1/m$, we have $\rho = \lambda$, so that

$$T = \frac{m}{1-\lambda} + \frac{1-2\lambda}{2(1-\lambda)}$$

In the limit of large m, this is twice the delay of TDM as given in Eq. (3.59).

4.9

a) Let ν be the mean number of packets in the system. Given n packets in the system, with each packet independently transmitted in a slot with probability ν-1, the probability of an idle slot, $P\{I|n\}$ is $(1-\nu^{-1})^n$. The joint probability of an idle slot and n packets in the system is then

$$P\{n,I\} = P\{n\}P\{I|n\} = \frac{\exp(-\nu)\nu^n}{n!}(1-\nu^{-1})^n$$

$$P\{I\} = \sum_{n=0}^{\infty} P\{n,I\} = \sum_{n=0}^{\infty} \frac{\exp(-\nu)\,(\nu-1)^n}{n!} = \frac{1}{e}$$

b) Using the results above, we can find $P\{n \mid I\}$

$$P\{n \mid I\} = \frac{P\{n,I\}}{P\{I\}} = \frac{\exp(-\nu+1)\,(\nu-1)^n}{n!}$$

Thus, this probability is Poisson with mean $\nu-1$.

c) We can find the joint probability of success and n in the system similarly

$$P\{n,S\} = P\{n\}P\{S \mid n\} = \frac{\exp(-\nu)\,\nu^n}{n!}\, n(1-\nu^{-1})^{n-1}\nu^{-1} = \frac{\exp(-\nu)\,(\nu-1)^{n-1}}{(n-1)!}$$

$$P\{S\} = \sum_{n=0}^{\infty} \frac{\exp(-\nu)\,(\nu-1)^{n-1}}{(n-1)!} = \frac{1}{e}$$

d) From this, the probability that there were n packets in the system given a success is

$$P\{n \mid S\} = \frac{P\{n,S\}}{P\{S\}} = \frac{\exp(-\nu+1)\,(\nu-1)^{n-1}}{(n-1)!}$$

Note that n-1 is the number of remaining packets in the system with the successful packet removed, and it is seen from above that this remaining number is Poisson with mean $\nu-1$.

4.10

a) All nodes are initially in mode 2, so when the first success occurs, the successful node moves to mode 1. While that node is in mode 1, it transmits in every slot, preventing any other node from entering mode 1. When that node eventually transmits all its packets and moves back to mode 2, we return to the initial situation of all nodes in mode 2. Thus at most one node at a time can be in mode 1.

b) The probability of successful transmission, p_1, is the probability that no other node is transmitting. Thus $p_1 = (1-q_r)^{m-1}$. The first and second moment of the time between successful transmissions is the same computation as in problem 4.8a. We have

$$\overline{X} = \frac{1}{p_1} \qquad \overline{X^2} = \frac{2-p_1}{p_1^2}$$

c) The probability of some successful dummy transmission in a given slot when all nodes are in mode 2 is $p_2 = mq_r(1-q_r)^{m-1}$. The first two moments of the time to such a success is the same problem as above, with p_2 in place of p_1. Thus

$$\overline{v} = \frac{1}{p_2} \qquad \overline{v^2} = \frac{2-p_2}{p_2^2}$$

d) The system is the same as the exhaustive multiuser system of subsection 3.5.2 except for the random choice of a new node to be serviced at the end of each reservation interval. Thus for the ith packet arrival to the system as a whole, the expected queueing delay before the given packet first attempts transmission is

$$E\{W_i\} = E\{R_i\} + E\{N_i\}\overline{X} + E\{Y_i\}$$

where R_i is the residual time to completion of the current packet service or reservation interval and Y_i is the duration of all the whole reservation intervals during which packet i must wait before its node enters mode 1. Since the order of serving packets is independent of their service time, $E\{N_i\} = \lambda E\{W\}$ in the limit as i approaches infinity. Also, since the length of each reservation interval is independent of the number of whole reservation intervals that the packet must wait, $E\{Y_i\}$ is the expected number of whole reservation intervals times the expected length of each. Thus

$$W = \frac{R + E\{S\}\overline{v}}{1-\rho}, \quad \rho = \lambda\overline{X}$$

e) As in Eq. (3.64),

$$R = \frac{\lambda\overline{X^2}}{2} + \frac{(1-\rho)\overline{v^2}}{2\overline{v}}$$

Finally, the number of whole reservation intervals that the packet must wait is zero with probability 1/m, one with probability (1-1/m)/m, and in general i with probability $(1-1/m)^i/m$. Thus $E\{S\} = m-1$. Substituting these results and those of parts b) and c) into the above expression for W, we get the desired expression.

4.11

a) Since all nodes receive feedback at the same time, and the first node involved in a collision waits one time unit for transmissions currently in progress to cease, all retransmissions must be successful. We assume here that if feedback for one collision arrives while previous retransmissions are taking place (this can happen if τ is large), then the new retransmissions follow the old. Under heavy loading, we note that many packets will typically arrive and become backlogged during the retransmissions for the previous period. These will all be transmitted (and thus establish a reservation order for retransmission) in the first time unit of the next reservation interval. Feedback for other collisions during the reservation interval will normally arrive before the retransmissions for these backlog collisions are completed. If τ is large, there will be occasional successful transmissions during the reservation period, but only a small fraction (about $e^{-2\lambda}$) of the transmissions during the reservation period are successful and only a small fraction of time is occupied by reservation periods. Thus it is reasonable to approximate all arrivals during one retransmission period and the following reservation period as being retransmitted in the following retransmission period. This corresponds to the partially gated single user system of subsection 3.5.2 with deterministic service X=1 and deterministic reservation period $A=1+\tau$. From Eq. (3.73) with m=1, the queueing delay is then

$$W = \frac{\lambda}{2(1-\lambda)} + \frac{(1+\tau)(1+\lambda)}{2(1-\lambda)}$$

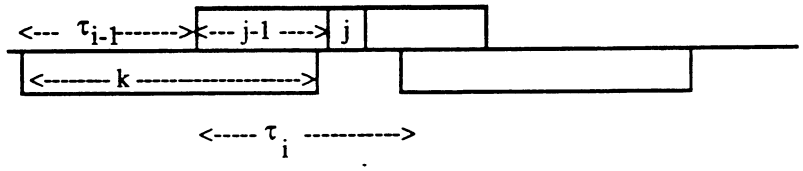
The system delay is T = W+1.

b) The above equation shows that W and T are finite for $\lambda < 1$

4.12

a) Let τ_i be the interval between the i^{th} and $i+1^{th}$ initiation of a k packet transmission group; thus τ_i is exponentially distributed with rate G and ...τ_{i-1}, τ_i , ... are independent. The j^{th} packet in the i^{th} transmission group will be successful if $\tau_i \geq j$ and if $\tau_{i-1} \geq k-j+1$ (see the diagram below). Thus

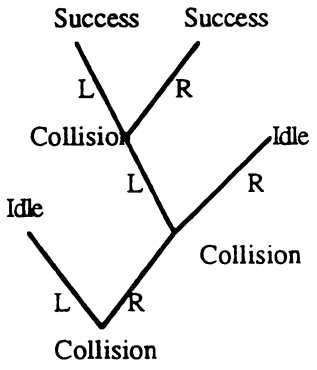
$$P_{succ} = e^{-Gj} e^{-G(k-j+1)} = e^{-G(k+1)}$$



b) Since the group attempt rate is G, the packet attempt rate is Gk. A fraction $e^{-G(k+1)}$ of the packets are successful, so the throughput is $Gk\, e^{-G(k+1)}$. This is maximized by G = 1/(k+1), leading to a maximum throughput of k/[e(k+1)]. This can be made as close to 1/e as desired by increasing k.

4.13

a) The tree and the corresponding operations for each slot are shown below



Slot	Transmit Set	Waiting Sets	Feedback
1	S	-	e
2	L	R	0
3	R	-	e
4	RL	RR	e
5	RLL	RLR,RR	1
6	RLR	RR	1
7	RR	-	0

b) The second collision (i.e., that on slot 3) would have been avoided by the first improvement to the tree algorithm.

c) e,0,e,1,1; the final set, RR, would have been incorporated into the next collision resolution period in the second improvement.

4.14

Note that collisions correspond to non-leaf nodes of the tree and idles or successes correspond to leaves. In the process of building a binary tree from the root, we start with one leaf (the root). In each successive step, one leaf node is converted to a non-leaf node and two new leaves are added, yielding a net gain of one leaf and one non-leaf. Thus in a binary rooted tree, the number of leaves exceeds the number of non-leaves by one. This means that the total number of nodes (which is the number of slots in a CRP) is one plus twice the number of collisions. In Figure 4.9, there are four collisions and nine slots, as predicted.

For the alternate approach, note that the stack depth increases by one for each collision and decreases by one for each success or idle. Viewing the stack as starting with the original set (one element) on the stack and terminating with an empty stack, we see that the number of decreases exceeds the increases by one, leading to the same answer as above.

4.15

a) The probability of i packets joining the left subset, given k packets in the original set, is given by the binomial distribution

$$\frac{k!\,2^{-k}}{i!(k-i)!}$$

b) Assuming $k \geq 2$, the CRP starts with an initial collision that takes one slot. Given that i packets go into the left subset, A_i is the expected number of additional slots required to transmit the left subset and A_{k-i} is the expected number on the right. Taking the expectation over the number i of packets in the left subset, we get the desired result,

$$A_k = 1 + \sum_{i=0}^{k} \frac{k!2^{-k}}{i!(k-i)!}(A_i + A_{k-i})$$

c) Note that

$$\sum_{i=0}^{k} \frac{k!2^{-k}}{i!(k-i)!} A_i = \sum_{i=0}^{k} \frac{k!2^{-k}}{i!(k-i)!} A_{k-i}$$

Thus

$$A_k = 1 + 2\sum_{i=0}^{k} \frac{k!2^{-k}}{i!(k-i)!} A_i = 1 + 2^{-k+1}A_k + 2\sum_{i=0}^{k-1} \frac{k!2^{-k}}{i!(k-i)!} A_i$$

Taking the A_k term to the left side of the equation, we have

$$c_{ik} = \frac{k!2^{-k+1}}{i!(k-i)!(1-2^{-k+1})};\ i<k; \quad c_{kk} = \frac{1}{1-2^{-k+1}}$$

Evaluating this numerically, $A_2 = 5$ and $A_3 = 23/3$.

4.16

a) Given an original set of $k \geq 2$ packets and given that $i \geq 1$ packets join the left subset, the expected number of slots required is $1 + B_i + B_{k-i}$ (i.e., one slot for the original collision, B_i slots on the average for the left subset and B_{k-i} for the right. Given that $i=0$ packets join the left subset, however, the expected number of slots required is $1 + 1 + (B_k - 1)$ (i.e., one slot for the original collision, one for the left subset, and, since the first collision is avoided in the resolution of the right subset, $B_k - 1$ for that final resolution. Thus, since i is binomially distributed,

$$B_k = 1 + \sum_{i=1}^{k} \frac{k!2^{-k}}{i!(k-i)!}(B_i + B_{k-i}) + 2^{-k}B_k$$

$$= 1 + \sum_{i=1}^{k-1} \frac{k!2^{-k}}{i!(k-i)!}(B_i + B_{k-i}) + 2^{-k+1}B_k + 2^{-k}B_0$$

b) Noting the symmetry between i and k-i above, noting that $B_0 = 1$, and taking the B_k terms to the left,

$$B_k(1-2^{-k+1}) = 1 + 2^{-k} + \sum_{i=1}^{k-1} \frac{k!2^{-k+1}}{i!(k-i)!} B_i$$

$$C'_{kk} = \frac{1+2^{-k}}{1-2^{-k+1}} \qquad C'_{ik} = \frac{k!2^{-k+1}}{i!(k-i)!(1-2^{-k+1})} \; ; i<k$$

4.17

a) The joint event $X_L = 0$ and $X_L + X_R \geq 2$ is equivalent to $X_L = 0$ and $X_R \geq 2$, so

$$P\{X_L=0 \mid X_L+X_R \geq 2\} = \frac{P\{X_L=0\}P\{X_R \geq 2\}}{P\{X_L+X_R \geq 2\}} = \frac{e^{-G}[1-(1+G)e^{-G}]}{1-(1+2G)e^{-2G}}$$

This uses the fact that X_L and X_R are independent and $X = X_L + X_R$ is Poisson with mean 2G. The other equalities use these same facts.

b) $$P\{X_L=1 \mid X_L+X_R \geq 2\} = \frac{P\{X_L=1\}P\{X_R \geq 1\}}{P\{X_L+X_R \geq 2\}} = \frac{Ge^{-G}(1-e^{-G})}{1-(1+2G)e^{-2G}}$$

c) $$P\{X_L \geq 2 \mid X_L+X_R \geq 2\} = \frac{P\{X_L \geq 2\}P\{X_R \geq 0\}}{P\{X_L+X_R \geq 2\}} = \frac{1-(1+G)e^{-G}}{1-(1+2G)e^{-2G}}$$

d) $$P\{X_R=1 \mid X_L=1,\ X_L+X_R\geq 2\} = \frac{P\{X_R=1\}P\{X_L=1\}}{P\{X_L=1\}P\{X_R\geq 1\}} = \frac{Ge^{-G}}{1-e^{-G}}$$

Note that this is just $P\{X_R=1 \mid X_R\geq 1\}$.

e) $$P\{X_R=i \mid X_L=0,\ X_L+X_R\geq 2\} = \frac{P\{X_R=i\}P\{X_L=0\}}{P\{X_L=0\}P\{X_R\geq 2\}} = \frac{G^i e^{-G}}{i!\,[1-(1+G)e^{-G}]}$$

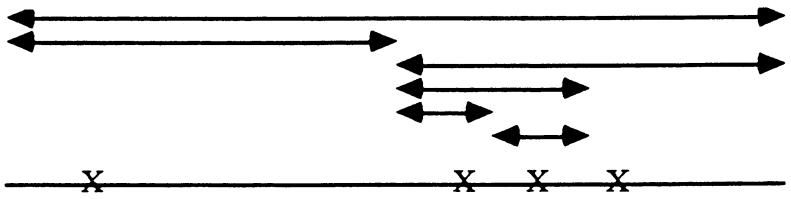
Note that this is $P\{X_R=i \mid X_R\geq 2\}$.

f) $$P\{X_R=i \mid X_L\geq 2,\ X_L+X_R\geq 2\} = \frac{P\{X_R=i\}P\{X_L\geq 2\}}{P\{X_L\geq 2\}} = \frac{G^i e^{-G}}{i!}$$

Note that this is $P\{X_R=i\}$.

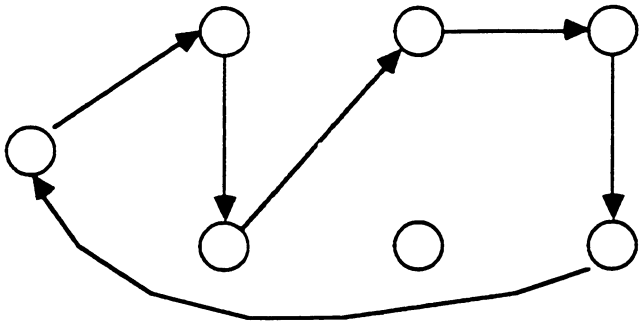
4.18

a) and b)



left boundary	interval size	right boundary	feedback at end	rule for next
T_k	α_0	$T_k+\alpha_0$	e	4.15
T_k	$\alpha_0/2$	$T_k+\alpha_0/2$	1	4.16
$T_k+\alpha_0/2$	$\alpha_0/2$	$T_k+\alpha_0$	e	4.15
$T_k+\alpha_0/2$	$\alpha_0/4$	$T_k+3\alpha_0/4$	e	4.15
$T_k+\alpha_0/2$	$\alpha_0/8$	$T_k+5\alpha_0/8$	1	4.16
$T_k+5\alpha_0/8$	$\alpha_0/8$	$T_k+3\alpha_0/4$	1	4.18

c) The diagram below shows the path through the Markov chain for the given sequence of events:



4.19

a) Consider first the expected number of successes on all but the final slot of a CRP and then the expected number on the final slot. Each non-final success occurs on a left interval, and, in terms of the Markov chain of Figure 4.13, corresponds to a transition from a left interval (top row of states) to a right interval (bottom row of states). Thus, for any given CRP, the number of non-final successes is the number of transitions from left to right states. Since the right states ($i \geq 1$) are entered only by these transitions, the number of non-final successes is the number of visits to right states (other than the state (R,0). Thus the expected number of non-final successes is the expected number of visits to right states, $i \geq 1$. Finally note that, except when the CRP consists only of a single idle slot (which occurs with probability $\exp(-G_0)$), the final slot of a CRP is a success. Thus

$$\bar{n} = 1 - \exp(-G_0) + \sum_{i=1}^{\infty} p(R,i)$$

b) Whenever a collision occurs in a left interval, the corresponding right interval is returned to the waiting interval. The number of packets successfully transmitted in a CRP is the number in the original allocation interval less the number returned by the mechanism above. Thus the expected number transmitted is the expected number in the original interval less the expected number returned; this is true despite statistical dependencies between the original number and the returned number. Given that an interval of size x is returned, the number of packets in the returned interval is Poisson with mean λx, independent of the past history of the CRP. Thus, given that a fraction f of the interval is returned (via one or more intervals), the expected number of returns is $\lambda\alpha_0 f$. Averaging over f (which is dependent on the number in the original interval), the expected number of returns is $\lambda\alpha_0 E\{f\}$. Thus

$$\bar{n} = \lambda\alpha_0[1 - E\{f\}]$$

4.20

Let n_k (or n, suppressing the subscript for slot time) be the backlog at slot k and assume n is Poisson with known mean ν. Each of the n packets is independently transmitted in slot k with probability $q_r(\nu)$, so the probability that the k^{th} slot is idle, given n, is $P\{I|n\} = [1-q_r(\nu)]^n$. Thus the joint probability of n backlogged packets and an idle slot is

$$P\{n,I\} = P\{n\}P\{I\,|n\} = \frac{\exp(-\nu)\nu^n}{n!}[1-q_r(\nu)]^n$$

$$P\{I\} = \sum_{n=0}^{\infty} P\{n,I\} = \sum_{n=0}^{\infty} \frac{\exp(-\nu)[\nu - \nu q_r(\nu)]^n}{n!} = \exp[-\nu q_r(\nu)]$$

$$P\{n\,|\,I\} = \frac{P\{n,I\}}{P\{I\}} = \frac{\exp[-\nu+\nu q_r(\nu)]\,[\nu-\nu q_r(\nu)]^n}{n!}$$

Thus, this probability is Poisson with mean $\nu-\nu q_r(\nu)$. Next consider a success

$$P\{n,S\} = P\{n\}P\{S\,|n\} = \frac{\exp(-\nu)\,\nu^n}{n!}\,n[1-q_r(\nu)]^{n-1}q_r(\nu)$$

$$= \frac{\exp(-\nu)\,[\nu-\nu q_r(\nu)]^{n-1}\nu q_r(\nu)}{(n-1)!}$$

$$P\{S\} = \sum_{n=1}^{\infty} P\{n,S\} = \nu q_r(\nu)\exp(-\nu)\sum_{n=1}^{\infty}\frac{[\nu-\nu q_r(\nu)]^{n-1}}{(n-1)!} = \nu q_r(\nu)\exp[-\nu q_r(\nu)]$$

$$P\{n\,|\,S\} = \frac{\exp[-\nu+\nu q_r(\nu)]\,[\nu-\nu q_r(\nu)]^{n-1}}{(n-1)!}$$

This says that the aposteriori distribution of n-1, given S, is Poisson with mean $\nu-\nu q_r(\nu)$.

4.21

a) Since X_1 and X_2 are non-negative random variables, $\max(X_1, X_2) \le X_1+X_2$ for all sample values. Taking expectations,

$$\overline{Y} \le 2\overline{X} = 2$$

Suppose X takes values β with probability $1-\varepsilon$ and $k\beta$ with probability ε. Since

$$\overline{X} = \beta(1-\varepsilon) + k\beta\varepsilon = 1, \text{ we have } \varepsilon = \frac{1-\beta}{\beta(k-1)}$$

Y takes on the value β with probability $(1-\varepsilon)2$ and the value $k\beta$ with probability $2\varepsilon-\varepsilon2$, so

$$\overline{Y} = \beta[1+(k-1)(2\varepsilon-\varepsilon^2)] = \beta + (1-\beta)(2-\varepsilon)$$

As k gets large, ε gets small and the final ε in the above expression is negligible. Thus, for small β, $E\{Y\} \approx 2$.

b) With a collision between two packets, the time until both transmissions are finished is the maximum of the two transmission times; the expected value of this is at most 2 from a), and the following idle slot adds a final β (A more refined analysis, using β as the minimum packet length, would show that $E\{Y\} \leq 2-\beta$, so the final β could be omitted).

c) The time between state transitions is β with probability $e^{-g(n)}$, $(1+\beta)$ with probability $g(n)e^{-g(n)}$, and at most $(2+\beta)$ with probability $[g^2(n)/2]e^{-g(n)}$ (ignoring collisions of more than two packets). Thus the expected time between transitions is at most

$$\beta e^{-g(n)} + (1+\beta)g(n)e^{-g(n)} + (1+\beta/2)g^2(n)e^{-g(n)}$$

d) The success probability in state n is $g(n)e^{-g(n)}$, so the expected number of departures per unit time is the ratio of this to expected time between transitions (this can be justified rigorously by renewal theory). Thus the expected number of departures per unit time is at least

$$\frac{g(n)e^{-g(n)}}{\beta e^{-g(n)} + (1+\beta)g(n)e^{-g(n)} + (1+\beta/2)g^2(n)e^{-g(n)}} = \frac{g(n)}{\beta+(1+\beta)g(n)+(1+\beta/2)g^2(n)}$$

e) Taking the derivative of this with respect to g(n), we find a maximum where $g^2(n) = \beta/(1+\beta/2)$. Thus for small β, g(n) is approximately the square root of β. Substituting this back into the expression in d), the maximum throughput (i.e., departures per unit time), is approximately $1-2\sqrt{\beta}$.

4.22

a) Let ν be the mean of the number n of backlogged packets. Then the unconditional probability density of the time to the first packet transmission attempt is

$$p(\tau) = \sum_{n=0}^{\infty} p(\tau|n)P\{n\} = \sum_{n=0}^{\infty} (\lambda+xn)\exp[-(\lambda+xn)\tau]\frac{e^{-\nu}\nu^n}{n!}$$

$$= \sum_{n=0}^{\infty} (\lambda+xn)\exp(-\lambda\tau-\nu)\frac{(\nu e^{-x\tau})^n}{n!}$$

Splitting this into two terms, the first multiplied by λ and the second by xn, we get

$$p(\tau) = (\lambda + x\nu e^{-\tau x})\exp[-\lambda\tau-\nu+\nu e^{-x\tau}]$$

b) The joint probability of a backlog n, a backlogged packet starting first (denoted by b), and a starting time τ (as a density) is given by

$$P\{n,b,\tau\} = \frac{xne^{-(\lambda+xn)\tau}e^{-\nu}\nu^n}{n!} = \frac{xe^{-\lambda\tau-\nu}(\nu e^{-x\tau})^n}{(n-1)}$$

Using the result in a),

$$P\{n,b|\tau\} = \frac{x\nu'}{\lambda+\nu'} \frac{\nu'^{n-1}\exp(-\nu')}{(n-1)!}$$

where $\nu' = \nu e^{-\tau x}$.

c) The joint probability of backlog n, a new arrival starting first (denoted by a), and starting time τ (as a density) is given by

$$P\{n,a,\tau\} = \frac{\lambda e^{-(\lambda+nx)\tau} e^{-\nu} \nu^n}{n!} = \frac{\lambda e^{-\lambda\tau-\nu}(e^{-x\tau}\nu)^n}{n!}$$

$$P\{n,a|\tau\} = \frac{\lambda}{\lambda+x\nu'} \frac{\nu'^n \exp(-\nu')}{n!}; \quad \nu' = \nu e^{-\tau x}$$

d) Let n' be n-1 if a backlogged packet starts and n' be n if a new arrival starts. Adding the result in b) to that in c), we have

$$P\{n'|\tau\} = \frac{\nu'^{n'}\exp(-\nu')}{n'!}$$

That is, n' is Poisson with mean $\nu' = \nu e^{-\tau x}$.

4.23

In both systems, the maximum throughput depends on what happens with large backlogs. For slotted Aloha, the expected number of transmissions in state n (for n large) is g(n), which is optimally chosen as $\sqrt{(2\beta)}$. For large backlogs, the state can be estimated relatively accurately, so that the probability of a successful slot following any given idle slot is approximately $\sqrt{(2\beta)}$. For the FCFS algorithm, the allocation interval at the beginning of a CRP is chosen to make $\sqrt{(2\beta)}$ the probability of an initial successful slot. Thus, the two systems perform in essentially the same way except when a collision is being resolved in the FCFS algorithm. On the first slot after a collision, the FCFS algorithm sends a fraction $\sqrt{\beta}$ of the colliding interval, which yields a success with probability approximately $2\sqrt{\beta}$. This is an improvement over slotted Aloha, and the improvement is still greater on the right interval of the CRP. Thus the number of idle slots required to transmit the next two packets after a collision for FCFS is smaller than for slotted Aloha. However, for small β, collisions occur in slotted Aloha on a fraction of packets roughly equal to g, which is $\sqrt{(2\beta)}$; the additional idle time required for each of these is proportional to $\sqrt{\beta}$. This means that the idle time per packet to resolve collisions in slotted Aloha is proportional to β, which is negligible to order $\sqrt{\beta}$.

4.24

a) Let E{t} be the expected time between initiations of successful packet transmissions, assuming a backlogged system with the number of transmissions after each idle slot having

a Poisson distribution with mean g. Using the same argument as in Eq. (4.43), but recognizing that the time occupied by a collision is now 2β, including the idle slot after the collision, we have

$$E\{t\} = [\beta+E\{t\}]e^{-g} + [1+\beta]ge^{-g} + [2\beta+E\{t\}][1-(1+g)e^{-g}]$$

$$E\{t\} = \frac{\beta e^{-g} + (1+\beta)ge^{-g} + 2\beta[1-(1+g)e^{-g}]}{ge^{-g}} = 1 + \frac{\beta(2e^{g}-1-g)}{g}$$

Minimizing numerically gives $E\{t\} = 1 + 3.31\beta$ at $g = 0.77$.

b) Using Little's relation on the time average of Eq. (4.42),

$$W = \frac{\overline{R} + \overline{y}}{1-\lambda E\{t\}} = \frac{\overline{R} + \overline{y}}{1-\lambda(1+3.31\beta)}$$

c) For small β, the contribution of the idle and collision intervals to the residual time R can be ignored since they are proportional to β^2. Thus,

$$\overline{R} \approx \frac{\lambda E\{(X+\beta)^2\}}{2} \approx \frac{\lambda\overline{X^2} + 2\lambda\beta}{2}$$

As in CSMA,

$$\overline{y} = E\{t\} - (1+\beta) = 2.31\beta$$

Substituting these results into the result in part b),

$$W \approx \frac{\lambda\overline{X^2} + \beta(4.62+2\lambda)}{2[1-\lambda(1+3.31\beta)]}$$

d) Clearly the Poisson approximation is very poor for a backlog of one, since collisions cannot occur. On the other hand, each interval t_i involves at least a backlog of two, so the effect of backlogs of one is seen only in the expected value of y, which is almost negligible.

4.25

We want to find the maximum time from when a given node starts to transmit in a collision until that node both stops transmitting and hears the channel become idle. Suppose a given node j starts to transmit at time t. By time t+β all other nodes must have ceased transmission. Thus, by time t+2β, node j ceases to hear these other transmissions. On the other hand, since by assumption a collision occurred, at least one other node started to transmit before t+β, so that node j must have ceased transmission also by t+2β. Note, however, that the definition of a collision is somewhat fuzzy. For example two nodes at one end of a bus could start transmitting almost simultaneously and then stop very quickly. A node at the other end of the bus could start transmitting almost b time units later and then

stop almost immediately because of the detected collision. A node at the first end of the bus, having heard the collision from the first two nodes cease, could start transmitting just before hearing the transmission from the second end of the bus, and another node at the second end could start transmitting after another β time units. Thus even though each node hears the channel become idle at most 2β time units after being involved in a collision, later collisions could be regarded as part of the same larger collision event.

4.26

a) The first transmission after a given idle detection will be successful if no other transmission starts within the next β time units. Since the process of initiations is Poisson with rate G, the probability of this is

$$P_{succ} = e^{-\beta G}$$

b) The mean time until the first initiation after an idle detection is 1/G (note that all nodes detect the channel as being idle at the same time). If this first initiation is successful, $1+\beta$ time units are required until the next idle detection; if the initiation is unsuccessful, 2β time units are required. Thus

$$E\{\text{time between idle detects}\} = G^{-1} + (1+\beta)e^{-\beta G} + 2\beta(1-e^{-\beta G})$$

c) The throughput T is the ratio of P_{succ} to the expected time between idle detects,

$$T = \frac{e^{-\beta G}}{G^{-1}+(1+\beta)e^{-\beta G}+2\beta(1-e^{-\beta G})} = \frac{1}{(G^{-1}+2\beta)e^{\beta G}+(1-\beta)}$$

d) We can maximize this by minimizing 1/T. Taking the derivative of 1/T with respect to G and setting it equal to 0, we find that the minimum of 1/T occurs at $\beta G = 1/2$. Substituting this into the expression for T, we get

$$T = \frac{1}{1+\beta(4\sqrt{e} - 1)} = \frac{1}{1+5.595\beta}$$

4.27

Each packet transmission is effectively extended by a round trip delay, mv. That is, if X is the time for a node to transmit a given packet plus token, then X is extended to X+mv if the token is not sent until the packet has returned to the sending node. The effective utilization factor then becomes $\rho = \lambda(E\{X\}+mv) = \lambda(1+mv)$. Using Eq. (3.76) with these modified values for ρ and X, we have

$$W = \frac{\lambda\overline{(X+mv)^2} + [m+\lambda(1+mv)]v}{2[1 - \lambda(1+mv+v)]}$$

Note that if the round trip delay is large relative to the packet transmission time, this causes a major increase in delay and a major decrease in maximum throughput.

4.28

Suppose a given node has a full transit buffer, and suppose the previous node on the ring has a never empty input buffer. Then that previous node will send a constant stream of packets (either from its transit buffer or input buffer) to the given node. If none of these packets are addressed to the given node, the hapless node must continue to transmit from its transit buffer, which remains full due to the constant input. In essence, in a fully loaded register insertion ring, the nodes that receive the most traffic are the ones permitted to transmit the most traffic. This problem could, of course, be overcome by a more complex protocol that prevents nodes from monopolizing the ring.

4.29

Given the placement of the first node at a given distance X_1 from the left end of the bus, the other node will be to the left of the first node with probability X_1 and to the right with probability $1-X_1$. Given that it is to the left, its expected distance from X_1 is $X_1/2$, and given that it is to the right, its expected distance is $(1-X_1)/2$. Thus the expected distance between X_1 and X_2, given X_1, is

$$E\{|X_2-X_1| \mid X_1\} = \frac{(X_1)^2}{2} + \frac{(1-X_1)^2}{2}$$

Averaging over X_1, we then have

$$E\{|X_2-X_1|\} = \int_0^1 \frac{(X_1)^2}{2} + \frac{(1-X_1)^2}{2}\, dX_1 = \frac{1}{3}$$

4.30

a) Node i is the lowest numbered node with a packet to send if nodes 0, 1, ..., i-1 have no packets (which occurs with probability $(1-q)^i$) and node i has a packet. Thus

$$P\{i\} = q(1-q)^i; \quad i \geq 0.$$

b) $k \geq 0$ is the number of successive sets of 2^j nodes that contain no packets. Thus k+1 sets of 2^j nodes each must be tested to find the first set with a packet, and j additional tests are necessary to find i within the final set of 2^j. Thus the required number of reservation slots is k+1+j.

c) For a given value of j, the expected number of reservation slots is E{k}+1+j. By the suggested approximation $k = i2^{-j}$, we have $E\{k\} = 2^{-j}E\{i\}$. Thus

$$E\{k\} = 2^{-j}\sum_{i=1}^{\infty} iq(1-q)^i = 2^{-j}q(1-q)\sum_{i=1}^{\infty} i(1-q)^{i-1} = 2^{-j}\frac{1-q}{q}$$

$$E\{\text{reservation slots}\} = 2^{-j}\frac{1-q}{q} + 1 + j$$

It is not much more difficult to find the exact value of $E\{k\}$. We note that k is the integer part of $i2^{-j}$, and thus $P\{k \geq n\} = P\{i \geq n2^j\}$. We then have

$$E\{k\} = \sum_{k=1}^{\infty} kP\{k\} = \sum_{n=1}^{\infty} P\{k \geq n\} = \sum_{n=1}^{\infty} P\{i \geq n2^j\}$$

$$= \sum_{n=1}^{\infty} \exp[n2^j \ln(1-q)] = \frac{\exp[n2^j \ln(1-q)]}{1 - \exp[n2^j \ln(1-q)]}$$

The exact value of $E\{k\}$ is less than the approximate value by a quantity somewhat less than 1/2.

d) Using the approximation for $E\{k\}$ again, we want to choose the integer $j \geq 0$ that minimizes $E\{\text{reservation slots}\}$. Observe that the expression for $E\{\text{reservation slots}\}$ above (temporarily regarding j as a real number) has a positive second derivative with respect to j. Thus the expression is minimized over integer $j \geq 0$ by choosing the smallest integer j for which the expression increases in going from j to j+1, i.e, for which

$$2^{-j}\frac{1-q}{q} + 1 + j < 2^{-j-1}\frac{1-q}{q} + 2 + j$$

This means that the minimum occurs at the smallest value of j for which $2^{j+1} > (1-q)/q$. Thus the minimizing j is the integer part of $\log_2[(1-q)/q]$.

We can find the minimizing j for $E\{\text{reservation slots}\}$ using the exact expression for $E\{k\}$ in the same way. Let $f(j) = \exp[2j\ln(1-q)]$ and note that $f(j+1) = f^2(j)$. Thus we want to find the smallest value of j for which

$$\frac{f(j)}{1-f(j)} < \frac{f^2(j)}{1-f^2(j)} + 1$$

This inequality is equivalent to $f(j) < 1 - f^2(j)$. The two sides of this inequality are equal for $f(j) = (\sqrt{5} - 1)/2$, so the minimizing j is the smallest integer j for which $f(j) < (\sqrt{5} - 1)/2$. Thus j is the smallest integer for which $2j\ln(1-q) < \ln[(\sqrt{5} - 1)/2]$.

4.31

a) A simultaneous transmission on links 1 and 3 causes a collision for the transmission on link 1; similarly, a collision on link 2 occurs if 2 and 3 are used simultaneously. Finally, simultaneous transmissions on links 1 and 2 cause a collision for both transmissions. Thus at most one link can be used successfully at a time and $f_1+f_2+f_3 \leq 1$. To view this in terms of Eq. (4.75), we let x_1, x_2, and x_3 be the collision free vectors (100), (010), and (001) respectively, and we let x_4 be the trivial CFV (000). Then, for $1 \leq i \leq 3$, f_i corresponds to a_i in Eq. (4.75) and the constraints $a_i \geq 0$ and $a_1+a_2+a_3+a_4=1$ is equivalent to $f_1+f_2+f_3 \leq 1$.

b) From Eq. (4.77), we have

$$p_1 = (1-q_3)(1-q_2)$$

$$p_2 = (1-q_3)(1-q_1)$$

$$p_3 = 1$$

Eq. (4.78) then gives us the fractional utilizations

$$f = f_1 = q_1(1-q_3)(1-q_2) \qquad \text{(i)}$$

$$f = f_2 = q_2(1-q_3)(1-q_1) \qquad \text{(ii)}$$

$$2f = f_3 = q_3 \qquad \text{(iii)}$$

Taking the ratio of (ii) and (i),

$$1 = \frac{q_2(1-q_1)}{q_1(1-q_2)} \; ; \qquad \text{thus } q_1 = q_2$$

c) Using $q_1 = q_2$ and $q_3 = 2f$ in (i) above, we have $f = q_1(1-2f)(1-q_1)$. Thus

$$\frac{f}{1-2f} = q_1(1-q_1) \le \frac{1}{4}$$

The inequality above follows by taking the maximum of $q_1(1-q_1)$ over q_1 between 0 and 1. It follows from this that $f \le 1/6$.

4.32

Let q_i be the attempt rate on link i and p_i be the probability of successful transmission on link i. A transmission on link 1 will be successful if no transmission is simultaneously taking place on link 3 (since link 5 is never used). Thus, assuming independent attempts, $p_1 = 1-q_3$. Similarly, a transmission on link 2 is successful if link 4 is not simultaneously carrying transmitting, so $p_2 = 1-q_4$. Transmissions on links 3 and 4 are always successful, so $p_3 = 1$ and $p_4 = 1$. Finally, a transmission on link 7 is successful if neither links 3 nor 4 are carrying transmissions simultaneously, so $p_3 = (1-q_3)(1-q_4)$. Eq. (4.78) states that the throughput on each link, f_i, is equal to $p_i q_i$. Combining these equations with the values for p_i found above and with the given throughputs, we have the equations

$$1/3 = f_1 = q_1(1-q_3)$$

$$1/3 = f_2 = q_2(1-q_4)$$

$$1/3 = f_3 = q_3$$

$$1/3 = f_4 = q_4$$

$$4/9 = f_7 = q_7(1-q_3)(1-q_4)$$

These throughputs will be feasible (under the assumptions given) if we can find values between 0 and 1 for the attempt rates q_i that satisfy these equations and if the node transmission rates are also between 0 and 1. The third and fourth equations show that $q_3 =$

$q_4 = 1/3$. Using these values in the first two equations, we find that $q_1 = q_2 = 1/2$. This indicates that the attempt rate from the left most node is 1, which is permissible. Finally, using $q_3 = q_4 = 1/3$ in the final equation gives us $q_7 = 1$. Thus we see that the given throughputs are feasible. The success probabilities are then easily found to be $p_1 = p_2 = 2/3$, $p_3 = p_4 = 1$, and $p_7 = 4/9$.

b) Since links 1 and 2 can not be used simultaneously, and links 3 and 4 always transmit on the slot following a successful transmission from 1 or 2 respectively, links 3 and 4 cannot be used simultaneously. Thus 2/3 of the slots are used to carry packets on either links 3 or 4. Thus at most 1/3 of the slots can carry packets successfully on link 7. What has happened here is that the assumption that nodes transmit independently of each other has been violated by the fact that links 3 and 4 are both forwarding traffic from the same node.

c) Since links 1 and 2 carry packets alternately, two successive attempts are never made on link 1, and thus (since link 3 always transmits only on the slot after receiving a packet on link 1) no packet on link 1 is ever subject to a collision. The same result holds for link 2, and thus $f_1 = f_2 = f_3 = f_4 = 1/2$. In this case, links 3 and 4 transmit on alternate slots, and all packets on link 7 suffer collisions.

4.33

The token arrival times, starting at t_1, are 4, 5, 6, 8, 9, 10, 12, 13, 14,.... . This can be expressed analytically as $t_i = 3+i+\lfloor(i-1)/3\rfloor$ for $i \geq 1$. The result as given in the bound is $t_i = i+2+\lfloor i/3\rfloor$ for $i \geq 1$. The corresponding token times, starting at t_1, are 4, 5, 7, 8, 9,11, 12, 13, ... The difference results from the bound taking t_0 as τ rather than as 0.

4.34

From Eq. (4.82), we see that the asymptotic round trip token time is $\tau m/(m+1) + T/(m+1)$. Node i receives α_i of that time for high priority traffic and $(\tau-T)/(m+1)$ for low priority traffic. Thus the fraction of traffic received by node i is

$$\frac{\alpha_i(m+1)}{\tau m+T} \text{ high priority;} \quad \frac{(\tau-T)m}{\tau m+T} \text{ low priority}$$

Fortunately this adds up to 1.

b) Since each node i requires only a fraction α_i/τ of the ring capacity for its high priority traffic, a fraction $(\tau-T)/\tau$ is left over for low priority. Since this is shared equally by each node, each node receives $(\tau-T)/(m\tau)$ for low priority traffic.

4.35

First assume A=1 and C1>0 where C1 is the value in counter 1.

> On the arrival of an idle slot in the downstream direction, decrement C1.
> On the arrival of a request bit in the upstream direction, increment C2 (the value of counter 2).

Next assume A=1 and C1=0.

On the arrival of an idle slot in the downstream direction,
1) Place the frame in the idle slot, setting the busy bit;
2) If there is a waiting frame in the supplementary queue, put it in the virtual queue, place C2 in counter 1 and set C2 to 0;
3) If there is no waiting frame, set A=0.

On the arrival of a request bit in the upstream direction, increment C2.

Next assume A=0.

On the arrival of an idle slot in the downstream direction, decrement C2.
On the arrival of a request bit in the upstream direction, increment C2.
On the arrival of a frame to be transmitted, put it in the virtual queue, place C2 in counter 1 and set C2 to 0.

Chapter 5 Solutions

5.1

The Prim-Dijkstra Algorithm Arbitrarily select node e as the initial fragment. Arcs are added in the following order: (d, e), (b, d), (b, c) {tie with (a, b) is broken arbitrarily}, (a, b), (a, f).

Kruskal's Algorithm Start with each node as a fragment. Arcs are added in the following order: (a,f), (b,d), (a,b) {tie with (b,c) is broken arbitrarily}, (b,c), (d,e).

The weight of the MST in both cases is 15.

5.2

The Bellman-Ford Algorithm By convention, $D_1^{(h)} = 0$, for all h. Initially $D_1^{(1)} = d_{1i}$, for all $i \neq 1$. For each successive $h \geq 1$ we compute $D_i^{(h+1)} = \min_j [D_j^{(h)} + d_{ji}]$, for all $i \neq 1$. The results are summarized in the following table.

i	D_i^1	D_i^2	D_i^3	D_i^4	D_i^5	Shortest path arcs†
1	0	0	0	0	0	
2	4	4	4	4	4	(1,2)
3	5	5	5	5	5	(1,3)
4	∞	7	7	7	7	(2,4)
5	∞	14	13	12	12	(6,5)
6	∞	14	10	10	10	(4,6)
7	∞	∞	16	12	12	(6,7)

†The arcs on the shortest path tree are computed *after* running the Bellman-Ford algorithm. For each $i \neq 1$ we include in the shortest path tree one arc (j,i) that minimizes Bellman's equation.

Dijkstra's Algorithm Refer to the algorithm description in the text. Initially: $D_1 = 0$; $D_i = d_{1i}$ for $i \neq 1$; $P = \{1\}$. The state after each iteration is

shown in the table below. P is not shown but can be inferred from i. Only the D_j's which are updated at each step are shown.

Iteration	i	D_1	D_2	D_3	D_4	D_5	D_6	D_7	Arc added
initial		0	4	5	∞	∞	∞	∞	
1	2			5	7	14	∞	∞	(1,2)
2	3				7	14	14	∞	(1,3)
3	4					13	10	∞	(2,4)
4	6					12		12	(4,6)
5	5							12	(6,5)
6	7								(6,7)

5.3

Let p_{ij} be the probability that link (i,j) fails during the lifetime of a virtual circuit. Let P_k be the probability that a path $k = (A, i, \ldots, j, B)$ remains intact. Since links fail independently we have:

$$P_k = (1 - p_{Ai}) \cdots (1 - p_{jB})$$

We want to find the path k for which P_k is maximized. Equivalently, we can find the path k for which $-\ln P_k$ is minimized.

$$-\ln P_k = -\ln(1 - p_{Ai}) - \cdots - \ln(1 - p_{jB})$$

Since the arc weights p_{ij} are small, $1 - p_{ij}$ is close to 1 and we may use the approximation $\ln z \approx z - 1$. This gives:

$$-\ln P_k \approx p_{Ai} + \cdots + p_{jB}$$

Therefore, the most reliable path from A to B is the shortest path using the weights given in the figure. Applying Dijkstra's algorithm gives the shortest path tree. We proceed as in problem 5.2.

Iteration	i	D_A	D_B	D_C	D_D	D_E	D_F	D_G	Arc added
initial		0	∞	0.01	∞	0.03	∞	∞	
1	C		∞		0.06	0.02	∞	∞	(A,C)
2	E		∞		0.04		0.06	∞	(C,E)
3	D		0.1				0.05	0.06	(E,D)
4	F		0.1					0.06	(D,F)
5	G		0.09						(D,G)
6	B								(G,B)

The most reliable path from A to B is (A,C,E,D,G,B). The probability that this path remains intact is

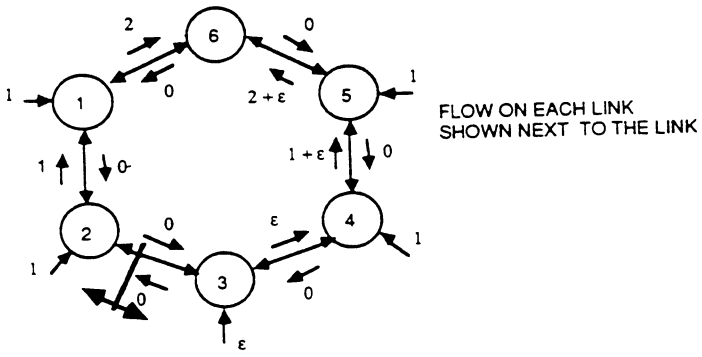
$$P_{ACEDGB} = (0.99)(0.99)(0.98)(0.98)(0.97) = 0.913$$

5.4

Let the weights for arcs AB, BC, and CA be 1, 2, and 2, respectively. Then an MST is $\{AB, BC\}$ whereas the shortest path tree rooted at C is $\{CA, CB\}$.

5.5

a) We consider the following network with an initial routing similar to example 1 in 5.2.5. A routing can be completely specified by indicating the link at which



the traffic changes from clockwise to counterclockwise. This link always carries zero traffic in both directions. For example, the routing in the above diagram would be called (2,3). With this as the initial routing, the subsequent routings would be: (4,5), (1,6), (5,6), (1,6)....

b) We proceed as in **a)** but add 1 to each length.

With an initial routing of (2,3), subsequent routings are: (3,4), (2,3).... Notice that the oscillations have been damped as compared to part **a)**, and a reasonable routing is always maintained.

With an initial routing of (1,2), subsequent routings are: (4,5), (1,2).... There are still undesirable oscillations, but the situation is not quite as bad as in **a)**.

With an initial routing of (1,6), subsequent routings are: (5,6), (1,6).... For this initial condition, the constant bias factor has had no effect on the oscillatory behavior.

By symmetry, the remaining three cases are equivalent to the above.

c) Notice that regardless of the choice of α, node 3 reverses its routing at each iteration. Therefore, the best that can be hoped for is oscillation between the two reasonable routings (2,3) and (3,4). In order to reduce oscillations with a

routing of (1,6), node 5 must continue to route counterclockwise. This requires that:

$$5\alpha > \alpha + 4 + \epsilon \quad \Rightarrow \quad \alpha > 1 + \epsilon/4$$

In order to reduce oscillations with a routing of (1,2), node 4 must continue to route counterclockwise. This requires that:

$$4\alpha + 1 > 2\alpha + 5 + 2\epsilon \quad \Rightarrow \quad \alpha > 2 + \epsilon$$

By symmetry, the remaining routings result is the same conditions. Therefore, for values of $\alpha > 2+\epsilon$ the routing of all nodes except node 3 eventually remains constant.

d) For this particular example, the averaging changes the link lengths, but has no effect on the routing decisions. The resulting routings are the same as in part **a)**.

5.6

(a) Let D_i be the shortest distance from node i to node 1 corresponding to lengths d_{ij}. We claim that

$$D_i \leq D_i^0, \qquad \forall\, i.$$

Given the definition of D_i^0, it will suffice to show that

$$D_i \leq \tilde{D}_i, \qquad \forall\, i \notin \cup_k N_k \cup \{1\}.$$

Indeed consider any node $i \notin \cup_k N_k \cup \{1\}$, and let $\tilde{P}_i$ be a shortest path from i to 1 corresponding to lengths $\tilde{d}_{ij}$. We have $\tilde{D}_m = \tilde{d}_{mn} + \tilde{D}_n$ for all arcs (m,n) of $\tilde{P}_i$, so by the definition of the sets N_k, we must have $d_{mn} \leq \tilde{d}_{mn}$ for all arcs (m,n) of $\tilde{P}_i$. Therefore, the length of P_i with respect to arc lengths d_{ij} is no more than its length with respect to arc lengths $\tilde{d}_{ij}$, implying that $D_i \leq \tilde{D}_i$. Thus we have $D_i \leq \tilde{D}_i$ for all $i \notin \cup_k N_k \cup \{1\}$ and $D_i \leq D_i^0$ for all i.

Now consider the Bellman-Ford method corresponding to arc lengths d_{ij} and starting from two different initial conditions. The first set of initial conditions is the standard $\hat{D}_i^0 = \infty$ for $i \neq 1$ and $\hat{D}_1^0 = 0$, and the corresponding iterates are denoted $\hat{D}_i^h$. The second set of initial conditions is D_i^0 as given in the problem statement and the corresponding iterates are denoted D_i^h. Since

$$D_i \leq D_i^0 \leq \hat{D}_i^0, \qquad \forall\, i,$$

we can show by using induction and the equations

$$\hat{D}_i^{h+1} = \min_j [d_{ij} + \hat{D}_j^h],$$

$$D_i^{h+1} = \min_j [d_{ij} + D_j^h],$$

$$D_i = \min_j [d_{ij} + D_j],$$

that

$$D_i \leq D_i^h \leq \hat{D}_i^h, \qquad \forall\, i,\, h.$$

Since $\hat{D}_i^h = D_i$ for $h \geq N-1$, it follows that $D_i^h = D_i$ for $h \geq N-1$, proving the desired result.

(b) As stated in the hint, when the length of a link (i,j) on the current shortest path tree increases, the head node i of the link should send an estimated distance $D_i = \infty$ to all nodes m such that (m,i) is a link. These nodes should send $D_m = \infty$ to their upstream neighbors if i is their best neighbor, that is, if link (m,i) lies on the shortest path tree, etc. Before any of the nodes k that sent $D_k = \infty$ to its upstream neighbors recalculates its estimated shortest distance, it should wait for a sufficient amount of time to receive from its downstream neighbors n any updated distances $D_n = \infty$ that may have resulted from the transmission of $D_i = \infty$.

5.7

Using the hint, we begin by showing that $h_i > h_{j_i}$ for all $i \neq 1$. Proof by contradiction. Suppose that there exists some $i \neq 1$ for which $h_i \leq h_{j_i}$. From the Bellman-Ford algorithm we have $D_i^{(h-1)} \geq D_i^{(h)}$. We define $h_1 = 0$ for completeness. Therefore, $D_{j_i}^{(h_i-1)} \geq D_{j_i}^{(h_{j_i})}$. However, if this held with equality it would contradict the definition of h_{j_i} as the largest h such that $D_{j_i}^{(h)} \neq D_{j_i}^{(h-1)}$. Therefore, $D_{j_i}^{(h_i-1)} > D_{j_i}^{(h_{j_i})}$. Using this strict inequality in the definition of j_i, $D_i^{(h_i)} = D_{j_i}^{(h_i-1)} + d_{j_i i}$, gives $D_i^{(h_i)} > D_{j_i}^{(h_{j_i})} + d_{j_i i}$. From the Bellman-Ford algorithm, we know that $D_i^{(h_{j_i}+1)} \leq D_{j_i}^{(h_{j_i})} + d_{j_i i}$. Using this in the previous expression gives $D_i^{(h_i)} > D_i^{(h_{j_i}+1)}$ which contradicts the definition of h_i as the largest h such that $D_i^{(h)} \neq D_i^{(h-1)}$. Therefore, the supposition that $h_i \leq h_{j_i}$ is incorrect. This proves the claim.

The subgraph mentioned in the problem contains $N-1$ arcs. To show that it is a spanning tree, we must show that it connects every node to node 1. To see this label each node i with h_i. Since the Bellman-Ford algorithm converges in at most $N-1$ iterations, we have $0 < h_i \leq N-1$ for all $i \neq 1$. Furthermore, $h_1 = 0$ and $h_i > h_j$, for all $i \neq 1$. Each node $i \neq 1$ is connected to a neighbor with a smaller label. We can trace a path in the subgraph from every node to node 1, therefore the subgraph must be a spanning tree.

Since the path lengths $D_i^{(h_i)}$ along the subgraph satisfy Bellman's equtation, the spanning tree is a shortest path spanning tree rooted at node 1.

5.8

Bellman's equation is

$$\begin{aligned} x_i &= \min_j \{x_j + d_{ji}\}, \quad i = 2, \ldots, N \\ x_1 &= 0 \end{aligned}$$

in the unknown vector $\vec{x}$. One solution is the set of shortest distances d_i from node 1 to each node i. Consider the subgraph G of all arcs (j, i) which are such that $D_i = D_j + dji$.

Claim: Every cycle of zero length not containing node 1 belongs to G.

Proof: If $(i_1, i_2), (i_2, i_3), \ldots, (i_k, i_1)$ is such a zero length cycle, we have

$$\begin{aligned} 0 &\leq D_{i_1} + d_{i_1 i_2} - D_{i_2} \\ 0 &\leq D_{i_2} + d_{i_2 i_3} - D_{i_3} \\ &\vdots \\ 0 &\leq D_{i_k} + d_{i_k i_1} - D_{i_1}. \end{aligned}$$

The sum of the right sides is 0, so the right side of each inequality is zero implying that the cycle belongs to G. This proves the claim.

Let C be the set of nodes that participate in a cycle of zero length not containing node 1. Let $\hat{C}$ be the set of nodes i that either belong to C or for which there is a node $j \in C$ and a directed path from j to i in the graph G. Note that $1 \notin \hat{C}$. For any $\delta \geq 0$ let

$$\begin{aligned} x_i &= D_i - \delta \quad \forall\, i \in \hat{C} \\ x_i &= D_i \qquad \forall\, i \notin \hat{C}. \end{aligned}$$

It is easily verified by substitution that the vector $\vec{x}$ defined above is a solution to Bellman's equation.

5.9

Define $S^1 = \{1\}$ and for $k = 1, 2, \ldots$ define

$$S_k = \left\{ \begin{array}{l} \text{all nodes } i \text{ such that either } i \in S_{k-1} \text{ or all} \\ \text{arcs } (j, i) \text{ have their head node } j \text{ in } S_{k-1} \end{array} \right\}$$

Claim: After some index k, S_k equals the entire set of nodes $\mathcal{N}$.

Proof: Let $\bar{S}_k = \mathcal{N} - S_k$ and suppose that S_k is non empty. Take any node $i \in \bar{S}_k$. Then by the connectivity assumption, i will have at least one incoming arc with its head node in S_k. Either all arcs (j, i) have their head node j in S_k, in which case $i \in S_{k+1}$ or else there is a node $j_1 \in \bar{S}_k$ for which (j_1, i) is an

arc. In the former case we see that S_{k+1} will be larger than S_k. In the latter case we repeat the process with i replaced by j_1. Eventually, we will obtain a node that belongs to S_{k+1} since otherwise a node in $\bar{S}_k$ would be reencountered thereby closing a directed cycle not containing node 1. This proves that S_{k+1} is larger than S_k. Therefore, S_k will be enlarged if it does not equal $\mathcal{N}$ and for k sufficiently large will equal N. This proves the claim.

Now if m_k is the number of nodes in S_k, renumber the nodes in $S_1 - S_0$ as $2, 3, \ldots, m_1$, then the nodes in $S_2 - S_1$ as $m_1 + 1, \ldots, m_2$ etc. With this numbering each arc (i, j) with $j \neq 1$ is such that $i \in S_{k_1}$ and $j \in S_{k_2} - S_{k_1}$ for some $k_1 < k_2$. The requirement of the problem is satisfied.

If the nodes are renumbered as indicated above, Bellman's equation can be written as

$$\begin{aligned} D_i &= \min_{j<i} \{D_j + d_{ji}\}, \quad i = 2, \ldots, N \\ D_1 &= 0 \end{aligned}$$

and can be solved by successive substitution, i.e., first solve for D_2, then for D_3, etc. This requires at most $O(N^2)$ operations.

5.10

(a) Refer to the algorithm statement in the text. Let B be the lower bound on the arc lengths. Then in step 1, each node $i \notin P$ with

$$D_i \leq \min_{j \notin P} \{D_j\} + B$$

can be added to P. To see why this is correct, recall that, at the start of each iteration, D_i for $i \notin P$ is the shortest distance from 1 to i for which all nodes on the path except i lie in P. The inductive argument which proved Dijkstra's algorithm required that each node added to P must have a shortest path for which all but the final node lie in P. This must be true for each node i which meets the above condition, since any path which included another node not in P would have a length of at least D_i.

(b) Assume that the shortest paths from node 1 to all other nodes have been found and have lengths D_j^*, $j \neq 1$. If link (i, k) increases in length, paths which do not traverse link k are not affected by this change. Therefore, we can initialize Dijkstra's algorithm as

$$P = \left\{ \begin{matrix} j \text{ such that a shortest path from 1 to } j \\ \text{does } \mathbf{not} \text{ traverse arc } (i, k) \end{matrix} \right\}$$

$$\begin{aligned} D_j &= D_j^* && \text{for all} j \in P \\ D_j &= \min_{l \in P} [D_l + d_{lj}] && \text{for all} j \notin P \end{aligned}$$

and continue with the ordinary algorithm.

5.11

(a) We have $D_1 = 0$ throughout the algorithm because initially $D_1 = 0$, and by the rules of the algorithm, D_1 cannot change.

We prove property (1) by induction on the iteration count. Indeed, initially (1) holds, since node 1 is the only node j with $D_j < \infty$. Suppose that (1) holds at the start of some iteration at which a node i is removed from V. If $i = 1$, which happens only at the first iteration, then at the end of the iteration we have $D_j = a_{j1}$ for all inward neighbors j of 1, and $D_j = \infty$ for all other $j \neq 1$, so D_j has the required property. If $j \neq 1$, then $D_j < \infty$ (which is true for all nodes of V by the rules of the algorithm), and (by the induction hypothesis) D_j is the length of some walk P_j starting at j, ending at 1, without going twice through 1. When D_i changes as a result of the iteration, D_i is set to $d_{ij} + D_j$, which is the length of the walk P_i consisting of P_j preceded by arc (i, j). Since $i \neq 1$, P_i does not go twice through 1. This completes the induction proof of property (1).

To prove property (2), note that for any j, each time j is removed from V, the condition $D_i \leq d_{ij} + D_j$ is satisfied for all $(i, j) \in \mathcal{A}$ by the rules of the algorithm. Up to the next entrance of j into V, D_j stays constant, while the labels D_i for all i with $(i, j) \in \mathcal{A}$ cannot increase, thereby preserving the condition $D_i \leq d_{ij} + D_j$.

(b) We first introduce the sets

$$I = \{i \mid D_i < \infty \text{ upon termination}\},$$

$$\overline{I} = \{i \mid d_i = \infty \text{ upon termination}\},$$

and we show that we have $D_j \in \overline{I}$ if and only if there is no walk to 1 from j. Indeed, if $i \in I$, then, since $i \notin V$ upon termination, it follows from condition (2) of part (a) that $j \in I$ for all $(j, i) \in \mathcal{A}$. Therefore, if $j \in \overline{I}$, there is no walk from node j to any node of I (and in particular, node 1). Conversely, if there is no walk from j to 1, it follows from condition (1) of part (a) that we cannot have $D_j < \infty$ upon termination, so $j \in \overline{I}$.

We show now that for all $j \in I$, we have $d_j = \min_{(j,i) \in \mathcal{A}} \{d_{ji} + D_i\}$ upon termination. Indeed, conditions (1) and (2) of part (a) imply that upon termination we have, for all $i \in I$,

$$D_j \leq d_{ji} + D_i, \qquad \forall\ j \text{ such that } (j, i) \in \mathcal{A}$$

while D_i is the length of some walk P_i from i to 1. Fix a node $m \in I$. By adding this condition over the arcs (j, i) of any walk P from m to 1, we see that the length of P is no less than D_m. Hence P_m is a shortest walk from m to 1. Furthermore, the equality $D_j = d_{ji} + D_i$ must hold for all arcs (j, i) on the shortest walks P_m, $m \in I$, implying that $D_j = \min_{(j,i) \in \mathcal{A}} \{d_{ji} + D_i\}$.

(c) If the algorithm never terminates, some D_j must decrease strictly an infinite number of times, generating a corresponding sequence of distinct walks P_j as

per condition (1) of part (b). Each of these walks can be decomposed into a path from j to 1 plus a collection of cycles. Since the number of paths from j to 1 is finite, and the length of the walk P_j is monotonically decreasing, it follows that P_j eventually must involve a cycle with negative length. By replicating this cycle a sufficiently large number of times, one can obtain walks from j to 1 with arbitrarily small length.

(d) Clear from the statement of Dijkstra's algorithm.

5.12

(a) We first note that the properties of part (a) of Problem 5.11. If upon termination we have $D_t = \infty$, then the extra test $d_{ij} + D_j + u_i < d_t$ for entering V is always passed, so the algorithm generates the same label sequences as the (many destinations) shortest path algorithm of Problem 5.11. Therefore, part(b) of Problem 5.11 applies and shows that there is no path from t to 1.

Let $\overline{D}_j$ be the final value of D_j obtained upon termination and suppose that $\overline{D}_t < \infty$. Assume, to arrive at a contradiction, that there is a path $P_t = (t, j_k, j_{k-1}, \ldots, j_2, j_1, t)$ that has length L_t with $L_t < \overline{D}_t$. For $m = 1, \ldots, k$, let L_{j_m} be the length of the path $P_m = (j_m, j_{m-1}, \ldots, j_2, j_1, t)$.

Let us focus on the node j_k following t on the path P_t. We claim that $L_{j_k} < \overline{D}_{j_k}$. Indeed, if this were not so, then j_k must have been removed at some iteration from V with D_{j_k} satisfying $D_{j_k} \leq L_{j_k}$. If D_t is the estimate of t at the start of that iteration, we would then have

$$d_{tj_k} + D_{j_k} \leq d_{tj_k} + L_{j_k} = L_t < \overline{D}_t \leq D_t,$$

implying that the shortest distance estimate of t would be reduced at that iteration from D_t to $d_{tj_k} + D_{j_k}$, which is less than the final estimate $\overline{D}_t$ – a contradiction.

Next we focus on the node j_{k-1} following j_k and t on the path P_t. We use a similar (though not identical) argument to show that $L_{j_{k-1}} < \overline{D}_{j_{k-1}}$. Indeed, if this were not so, then j_{k-1} must have been removed at some iteration from V with $D_{j_{k-1}}$ satisfying $D_{j_{k-1}} \leq L_{j_{k-1}}$. If D_{j_k} and D_t are the shortest distance estimates of j_k and t at the start of that iteration, we would then have

$$d_{j_k j_{k-1}} + D_{j_{k-1}} \leq d_{j_k j_{k-1}} + L_{j_{k-1}} = L_{j_k} < \overline{D}_{j_k} \leq D_{j_k},$$

and since $L_{j_k} + u_{j_k} \leq L_t < \overline{D}_t \leq D_t$, we would also have

$$d_{j_k j_{k-1}} + D_{j_{k-1}} < D_t - u_{j_k}.$$

From the above two equations, it follows that the shortest distance estimate of j_k would be reduced at that iteration from D_{j_k} to $d_{tj_k} + D_{j_k}$, which is less than the final label $\overline{D}_{j_k}$ – a contradiction.

Proceeding similarly, we obtain $L_{j_m} < \overline{D}_{j_m}$ for all $m = 1, \ldots, k$, and in particular $d_{j_1 1} = L_{j_1} < \overline{D}_{j_1}$. Since

$$d_{j_1 1} + u_{j_1} \leq L_t < \overline{D}_t,$$

and D_t is monotonically nonincreasing throughout the algorithm, we see that at the first iteration, j_1 will enter V with the label $a_{j_1 1}$, which cannot be less than the final estimate $\overline{D}_{j_1}$. This is a contradiction; the proof of part (b) is complete.

(b) The proof is identical to the proof of Problem 5.11(c).

5.13

Suppose that the sequence number field is finite with maximum equal to M. The exceptional circumstances referred to in the problem statement arise when the sequence number for updates of some node i becomes M within the memory of some other node, say j, due to a memory or communication error. Then the next time node i floods a new update into the network, it will receive M from node j through the feedback mechanism described at the end of section 5.3.2. The question now is how node i can convince node j to reduce its stored sequence number so that it can listen to a new update from node i.

The remedy is for node i, once it detects an error of the type described above, to issue a special "reset" packet which is flooded through the network. A node receiving a reset packet originated at node i sets its stored sequence number for node i to zero, and sends the reset packet to all its neighbors except the one from which the reset packet was received. In this way all nodes connected with node i will reset their sequence numbers to zero and the wraparound condition will be corrected.

There are two issues here: first how to avoid indefinite circulation of reset packets, and second how to guarantee that reset packets will not interfere with regular update packets or other reset packets from the same node. A clean way to ensure this (even if the links can reverse the order of reception of packets) is to add an age field to a reset packet which makes it "live" for exactly A seconds. The age limit A should be larger than the known upper bound for the time required for the reset packet to reach all nodes, so that the reset packet will live long enough to reset the sequence numbers of all nodes to zero. To avoid confusion node i should not issue any other update or reset packet for A seconds after issuing a reset packet. Finally, unlimited circulation of a reset packet, and confusion with other packets from node i, are avoided by requiring a node $j \neq i$ not to accept a reset packet or any update packet issued by node i if node j has received a reset packet from node i which is still "live." This is so because update packets from node i issued before the reset packet was issued cannot arrive at another node after the reset packet's age has expired under the assumption that an update packet reaches all nodes in time less than A. Note that this protocol could be used to operate flooding with a relatively small sequence number field. On the other hand, knowing an upper bound on the time required for an update packet to reach all nodes is a strong assumption, and one would like to minimize reliance on it.

5.14

A node is considered adjacent to the directed links for which it is the head node. Each node i decides upon a value associated with each of its adjacent links. We wish to find an algorithm which will reliably broadcast these values to each network node. To accomplish this we augment the SPTA as follows.

In addition to the main and port topology tables, let each node i keep similarly organized main and port information tables (I^i and I^i_j respectively) which contain entries for each directed link. The communication rules for these tables are the same as for the topology tables. When an entry for a non-adjacent link (m,n) changes in one of node i's port information tables, it updates the corresponding entry in its main table by setting

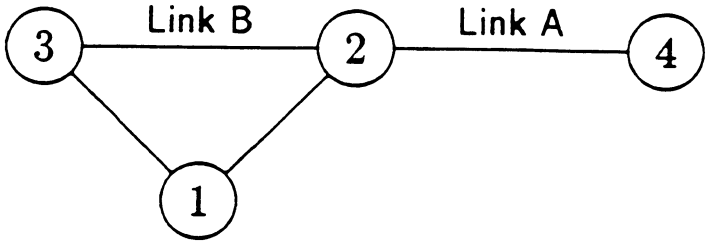
$$I^i(m,n) = I^i_{L(m)}(m,n),$$

where $L(m)$ is the label for node m which was assigned by the main topology table update algorithm. When the labels $L(m)$ are updated due to a topology change, each entry in I^i for a non adjacent link must be updated in this manner.

The value of $I^i(m,n)$ can be easily proven to be correct at each node i by using the fact that $L(m)$ at node i is the first hop on a shortest hop path from i to m. We first show that it is correct for nodes i which are 1 hop from m, then 2 hops etc.

5.15

(a) The algorithm can fail in the network shown below.



The table below shows a sequence of link status changes and message exchanges which lead the algorithm to fail. These events have been divided into numbered sections for reference purposes. The notation "$(i \rightarrow j, l\downarrow)$" is used to indicate that node i sends a message to node j indicating that link l is down. The entries in the topology column give the perceived status of link A at nodes 1 through 4 respectively. All links are initially up. We are only interested how the nodes determine the status of link A; messages concerning link B are not shown.

#	Description	Topology
1	Link A fails. $(2\rightarrow 1, A\downarrow)$ sent and received. $(2\rightarrow 3, A\downarrow)$ send and received.	dddd
2	Link B fails. $(1\rightarrow 3, A\downarrow)$ sent and received. $(3\rightarrow 1, A\downarrow)$ sent.	dddd
3	Link A is repaired. $(2\rightarrow 1, A\uparrow)$ sent and received.	uudu
4	$(3\rightarrow 1, A\downarrow)$ from #2 is received. $(1\rightarrow 3, A\uparrow)$ sent and received.	duuu
5	Link A fails. $(1\rightarrow 2, A\downarrow)$ sent and received. $(2\rightarrow 1, A\downarrow)$ sent and received.	ddud

After #5, the algorithm terminates with node 3 having an incorrect status for link A.

(b) We use the same scenario as in part (a) up to #4. The extra messages sent, due to the "including" rule, have no effect on the topology up to this point. The table below shows a scenario illustrating failure of the algorithm.

#	Description	Topology
4	$(1\rightarrow 3, A\uparrow)$ sent and received. $(1\rightarrow 2, A\uparrow)$ sent and received. $(3\rightarrow 1, A\downarrow)$ from #2 is received.	duuu
5	$(3\rightarrow 1, A\uparrow)$ sent and received. $(1\rightarrow 2, A\downarrow)$ sent and received. $(1\rightarrow 3, A\downarrow)$ sent and received.	uudu
6	$(2\rightarrow 1, A\uparrow)$ sent and received first. $(3\rightarrow 1, A\downarrow)$ sent and received second. $(1\rightarrow 3, A\uparrow)$ sent and received. $(1\rightarrow 2, A\uparrow)$ sent and received.	duuu
7	Same as #5.	uudu

Nodes 1 and 3 can oscillate indefinitely concerning their opinion of link A's status, and the algorithm never terminates. Although node 2 has the correct information, unfortunate message timing can stop it from helping the situation. This failure mode is at least as serious as that in part (a).

5.16

The ARPANET and the other algorithms which use sequence numbers are not affected. The sequence numbers can be used to sort out the correct message order. However, SPTA is clearly affected by a change in message order. For example, suppose that a link goes down and then up. If the adjacent nodes reverse the order of the two messages, the algorithm will fail to arrive at the correct topology.

5.17

(a) In the algorithm that follows, each node has two possible states, "connected" or "not connected". In addition, each node marks each of its neighbors with one of the following: "unknown", "not in tree", "incoming", or "outgoing". There are two kinds of messages used: "attach" and "ack". The following are the procedures executed at each node i.

Initially at each node i
state = "not connected"
mark(j) = "unknown" for each neighbor j

Start (Node 1 only)
state = "connected"
send "attach" to each neighbor

Receive "attach" from j
if state = "not connected"
  then state = "connected"
       mark(j) = "outgoing"
       if node i has neighbors other than j
         then send "attach" to each neighbor except j
         else send "ack" to j
              end
  else mark(j) = "not in tree"
       if mark(k) $\neq$ "unknown" for each neighbor k
         then send "ack" to the neighbor k such that mark(k) = "outgoing"†
              end

Receive "ack" from j
mark(j) = "incoming"
if mark(k) $\neq$ "unknown" for each neighbor k
  then send "ack" to the neighbor k such that mark(k) = "outgoing"†
       end

†Node 1 just terminates the algorithm; it has no "outgoing" neighbor

The above algorithm sends one "attach" and one "ack" message on each spanning tree link, and sends two "attach" messages (one in each direction) on each link not in the tree. Therefore, it sends a total of $2A$ messages.

(b) We use the spanning tree constructed in part (a) to simplify counting the nodes. Each node marks its "incoming" neighbors with either "heard from" or "not heard from". There are two messages used: "count nodes", and a messages containing a single number $j : 0 < j < N$. The following is the procedure for each node i.

Initialization
mark(j) = "not heard from" for all "incoming" neighbors
children = 0

Start (node 1 only)
send "count nodes" to all "incoming" neighbors

Receive "count nodes" from "outgoing" neighbor j
if there are any "incoming" neighbors
  then send "count nodes" on all incoming links
  else send "1" to j
       end

Receive n from "incoming" neighbor j
children = children + n

mark(j) = "heard from"
if mark(k) = "heard from" for all "incoming" neighbors k
  then send (children + 1) to the "outgoing" neighbor†
      end

†Node 1 has no outgoing neighbor. When it reaches this step, N = children + 1.

(c) The worst case for both algorithms is a linear network. Messages must propagate from node 1 to the end of the tree and then back again. This gives an upper bound of $2(N-1)T$ for both algorithms.

5.18

In the algorithm, for $j \notin P$, D_j is the minimum 1 hop distance from j to a member of P, and a_j is the member of P for which this minimum is obtained.

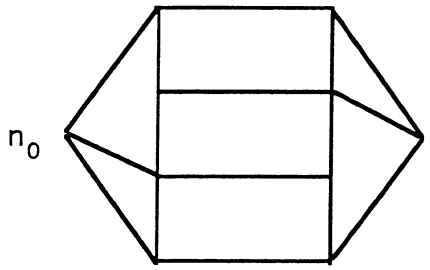
To show that this implements the Prim-Dijkstra algorithm, we must show that the graph defined by $G = (P,T)$ is a fragment of an MST, and that this fragment is enlarged at each iteration by the addition of a minimum weight outgoing arc. Then, by Proposition 1 in section 2.2, G will eventually be an MST.

Assume that P contains k nodes, that a_j is the closest member of P to j, and that $D_j = w_{ja_j}$ for $j \notin P$. Then step 1 chooses the minimum weight outgoing arc, and step 2 reestablishes the above assumptions about a_j and D_j for the new set P. The hypothesis is clearly true for $k = 1$ and by induction is true for all k.

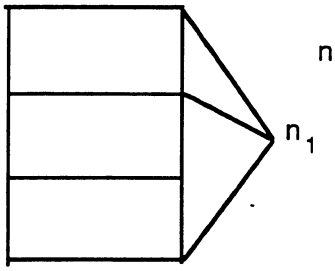
Each iteration of steps 1 and 2 requires a number of operations proportional to the number of nodes $i : i \notin P$. The algorithm terminates in $N-1$ iterations. Therefore, $O(N^2)$ operations are required.

5.19

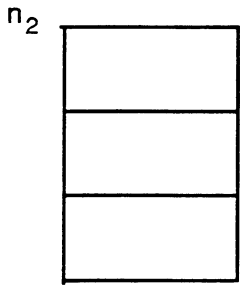
Choose n_1, n_1, n_2 as shown below



n_0 is 3-connected with every other node

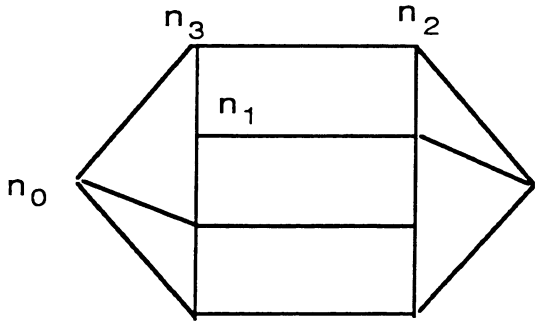


n_1 is 2-connected with every other node



n_2 is 1-connected with every other node

The network is not 4 - connected as shown below. (Removal of nodes n_0, n_1, and n_2 leaves node n_3 disconnected from the others.) The maximum k for which the network is k - connected is k = 3.



5.20

Modification of Kleitman's algorithm:

1st Step:

Choose a node n_0 and let k_0 be the maximum number k for which n_0 is k - connected to all other nodes. Set $k' = k_0$. If $k_0 = 1$ terminate, else delete n_0 and its adjacent arcs.

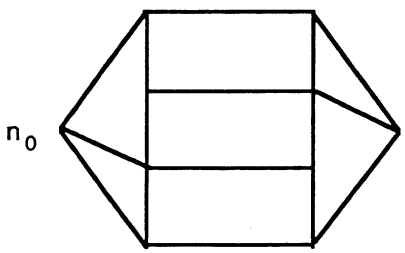
(m+1)st Step:

Choose a node n_m and let k_m be the maximum number k for which n_m is k - connected to all other nodes. Set $k' := \min\{k', k_m + m\}$. If $k_m \leq 1$ terminate, else delete n_m and its adjacent arcs and go to the (m + 2)nd step.

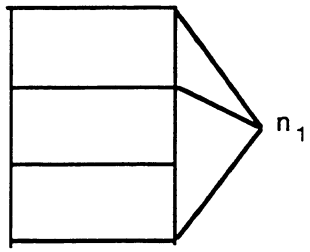
Claim: At termination the desired maximum number is k'.

Proof: The network cannot be k" - connected with k" > k' because the construction of k' is such that Kleitman's test of k" - connectedness for the sequence of nodes $n_0, n_1, \ldots,$ would fail. Also the algorithm will eventually terminate, and we will have $k_m \leq k' - m$ for every m. It follows that Kleitman's test of k' - connectivity is passed.

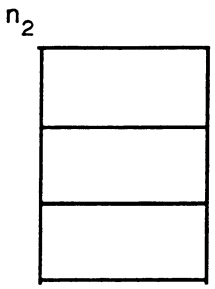
Application of the modified algorithm to the graph of Problem 5.19 works as follows:



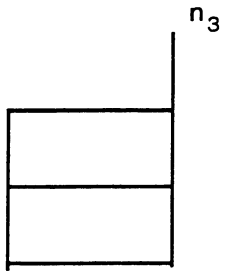
$k_0 = 3,\ k' = 3$



$k_1 = 2,\ k' = 3$



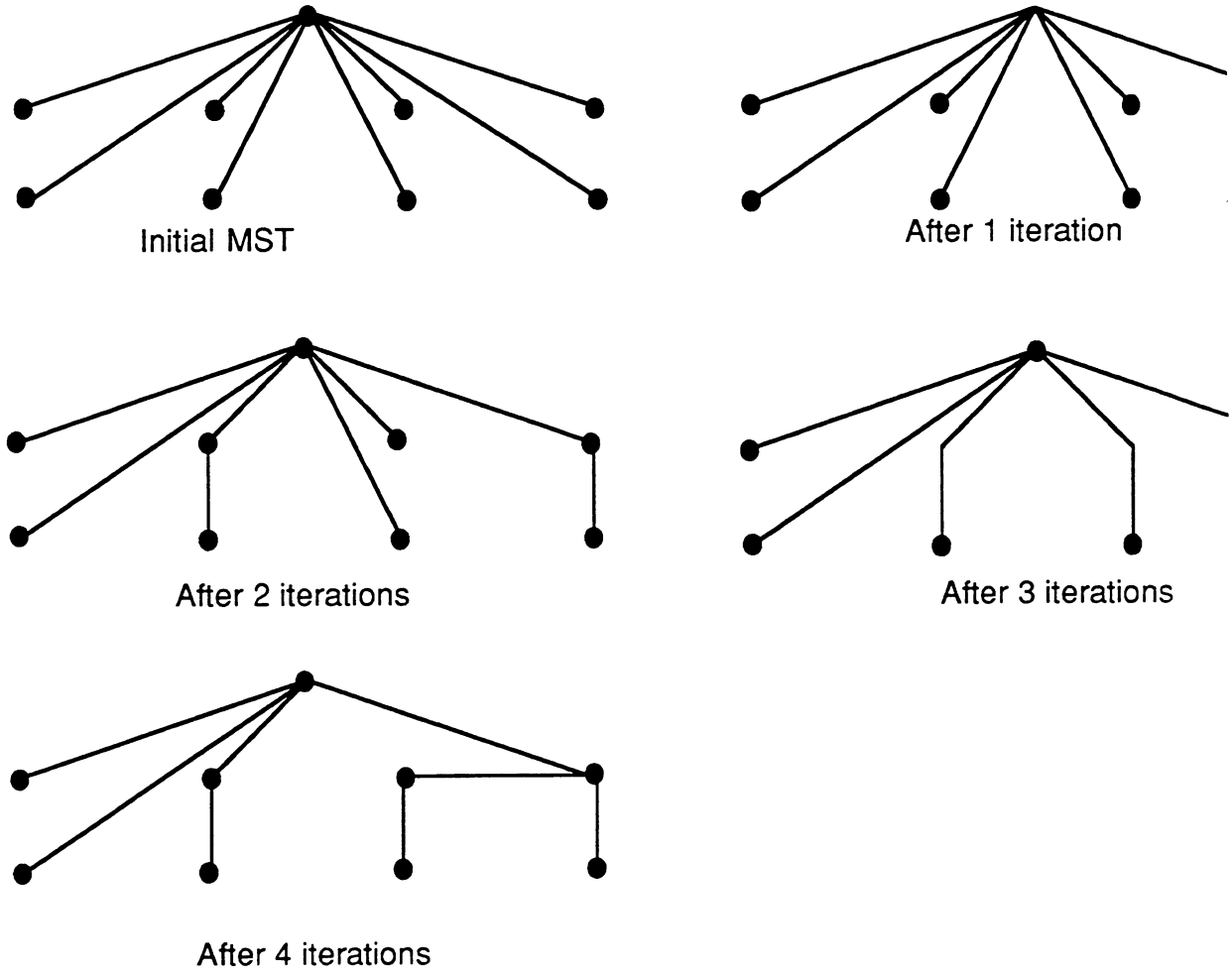
$k_2 = 2,\ k' = 3$



$k_3 = 1,\ k' = 3$

5.21

The sequence of generated spanning trees is shown below:



5.22

(a) Suppose every i to j walk contains an arc with weight greater or equal to a_{ij}. Consider an MST and the (unique) walk from i to j on the MST. If this walk does not consist of just arc (i,j), then replace an arc of this walk with weight greater or equal to a_{ij} with arc (i,j), thereby obtaining an MST.

Conversely suppose to obtain a contradiction, that (i,j) belongs to an MST and that there exists a walk W from i to j with all arcs having weight smaller than a_{ij}. We remove (i,j) from the MST obtaining two subtrees T_i and T_j containing i and j, respectively. The walk W must contain an arc that connects a node of T_i to a node of T_j. Adding that arc to the two subtrees T_i and T_j creates a spanning tree with smaller weight than that of the original, which is a contradiction.

(b) Walks from i to j that use nodes 1 through k+1 are of two types: 1) walks that use only

nodes 1 through k or 2) walks that go from i to k+1 using nodes 1 through k and then from k+1 to j using nodes 1 through k. The minimum critical weight of walks of type 1) is x_{ij}^k, while the critical weight over walks of type 2) is $\max\{x_{i(k+1)}^k, x_{(k+1)j}^k\}$. The characterization of x_{ij}^k given in the exercise follows.

5.23

(a) Let T^* be the given tree that spans the given subset of nodes and has minimal total weight W^*. Let T be a minimum weight spanning tree of I(G) and let W be its total weight. Finally, let R be a minimum weight tour of I(G) and let Y be its total weight.

By deleting any arc of R we obtain a spanning tree R' of I(G), which must have weight no more than the weight Y of R (since arc weights are nonnegative), and no less than the weight W of T [since T is a minimum weight spanning tree of I(G)]. Therefore

$$W \leq Y \qquad (1)$$

We will also show that

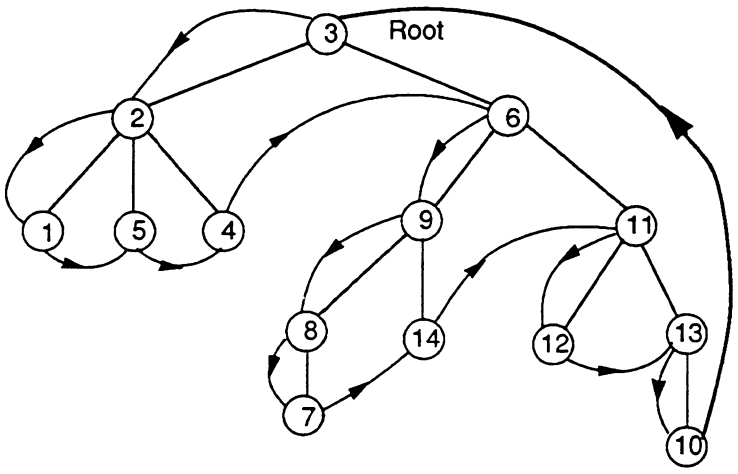
$$Y \leq 2W^* \qquad (2)$$

so that from (1) and (2), the desired result

$$W \leq 2W^*$$

follows.

By selecting an arbitrary node r of T^* as root we can view T^* as a tree rooted at r. Consider a depth-first traversal of T^* as illustrated in the following figure.

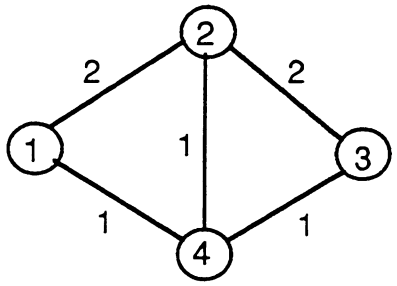


Traversal Order: 3, 2, 1, 5, 4, 6, 9, 8, 7, 14, 11, 12, 13, 10, 3

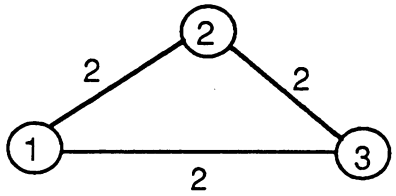
This traversal corresponds to a tour R' of I(G). Each arc (i,j) of the tour has length which

is less than or equal to the length of the unique path of T^* that connects i and j. The lengths of all these paths add up to $2W^*$, as can be seen from the figure. Therefore, the length Y' of the tour is no more than $2W^*$. Since Y is the weight of the minimum weight tour, we have $Y \leq Y'$, so it follows that $Y \leq 2W^*$.

(b) Consider the following grpah G with the weights shown next to the arcs, and let {1,2,3} be the subset of nodes that must be spanned by T^*.



The graph I(G) is shown below together with the corresponding arc weights.



We have $T^* = \{(1,4), (2,4), (3,4)\}$ with total weight $W^* = 3$. On the other hand T = {(1,2), (2,3)} and W = 4, so that

$$W^* < W < 2W^*$$

(c) Construct the minimum spanning tree T of I(G). For each arc (i,j) of T, consider a shortest path of G that starts at i and ends at j. Consider the subgraph G' of G that consists of the arcs of all the shortest paths corresponding to all the arcs of T. The sum of the weights of the arcs of G' is no more than the total weight W of T (it could be less because some arc of G' may be contained in more than one shortest path). Clearly G' is connected. Delete as many arcs as necessary to make it a tree. This tree, call it T', spans the required nodes and has total weight that is less or equal to W and therefore also less or equal to $2W^*$. Thus, the broadcasting algorithm that uses T' comes within a factor of 2 of being optimal.

5.24

See the hint.

5.25

If x_i is the flow carried by link i (i = 1, 2, 3), the corresponding path lengths are $C_i/(C_i-$

$x_i)^2$. At an optimum x_1 and x_3 must be equal since if, for example, $x_1 > x_3$ we will have that the length of x_1 is larger than the length of path 1 is larger than that of path 3 which contradicts the shortest path optimality condition.

Let x be the common value of x_1 and x_3 at the optimum. Then $x_2 = r - 2x$ and the solution of the problem follows the lines of the example of Section 5.5. We argue that, because $C_2 > C$, the only two possibilities are:

1) $x_2 = 0$ and $x = r/2$, which will occur for

$$C/(C - r/2)^2 \leq 1/C_2$$

2) $x_2 > 0$, and $x = (r - x_2)/2 > 0$ in which case x_2 and x are determined using the condition

$$C/(C - r/2 + x_2/2)^2 = C_2/(C_2 - x_2)^2.$$

5.26

(a) We have at x*

$$\partial D(x^*)/\partial x_1 = x_1^* = 2/3, \quad \partial D(x^*)/\partial x_2 = 2x_2^* = 2/3, \quad \partial D(x^*)/\partial x_3 = 1 + x_3^* = 1.$$

Therefore x* satisfies the shortest path condition and is optimal.

5.27

a) If the cost function is D(x) where x is the path flow vector, the first derivatives become

$$\frac{\partial D(x)}{\partial x_p} = \sum_{(i,j)} \frac{\partial D_{ij}(x)}{\partial x_p}.$$

A shortest path for a given OD pair is a path with minimal first derivative over all paths of the OD pair. The first derivatives of the reduced cost $D^r(x)$, i.e. the cost function obtained after the flows of the shortest paths are expressed in terms of the flows of the nonshortest paths in the cost function D(x), are given by

$$\frac{\partial D^r(x)}{\partial x_p} = \frac{\partial D(x)}{\partial x_p} - \frac{\partial D(x)}{\partial x_{p_w}}, \qquad \text{for all } p \in P_w$$

where p_w is the path from P_w that is shortest (has smallest $\partial D/\partial x_p$). The second derivatives are

$$\frac{\partial^2 D^r(x)}{(\partial x_p)^2} = \frac{\partial^2 D(x)}{(\partial x_p)^2} + \frac{\partial^2 D(x)}{(\partial x_{p_w})^2} - 2\frac{\partial^2 D(x)}{\partial x_p \partial x_{p_w}}.$$

The iteration for nonshortest paths becomes

$$x_p := \max\{0, x_p - [\frac{\partial^2 D^r(x)}{(\partial x_p)^2}]^{-1} \frac{\partial D^r(x)}{\partial x_p}\}.$$

The optimality condition is the same as in Section 5.5 (cf. eq. (5.59)).

b) In the special case where

$$D(x) = \sum_{(i,j)} D_{ij}(\tilde{F}_{ij}, F_{ij})$$

we have for a path p of priority class k

$$\frac{\partial D(x)}{\partial x_p} = \sum_{(i,j) \text{ on path } p} (p_k \frac{\partial D_{ij}}{\partial \tilde{F}_{ij}} + \frac{\partial D_{ij}}{\partial F_{ij}})$$

$$\frac{\partial^2 D(x)}{(\partial x_p)^2} = \sum_{(i,j) \text{ on path } p} (p_k^2 \frac{\partial^2 D_{ij}}{(\partial \tilde{F}_{ij})^2} + 2p_k \frac{\partial^2 D_{ij}}{\partial \tilde{F}_{ij} \partial F_{ij}} + \frac{\partial^2 D_{ij}}{(\partial F_{ij})^2})$$

and from here everything is the same as in part a).

5.28

The key in this problem is to calculate the 1st derivatives of the cost with respect to the portion x_t of R broadcast on any one spanning tree $t \in T$. We have:

$$\frac{\partial D}{\partial x_t} = \sum_{(i,j) \text{ on } t} D'_{ij},$$

as well as the constraint $\Sigma_{t \in T} x_t = R$. By the same argument used in Section 5.5, this leads to the following necessary and sufficient optimality condition (in addition to the ones of Section 5.5 for ordinary paths)

$x_t^* > 0 \quad \Rightarrow \quad$ t is shortest in that it has minimum $\Sigma_{(i,j)\in t} D_{ij}'$ over all trees in T.

(This means that at an optimum only the trees with minimum $\Sigma_{(i,j)\in t} D_{ij}'$ can carry a portion of R. Those that do must, of course, have equal $\Sigma_{(i,j)\in t} D_{ij}'$.) Given these facts, the gradient projection method generalizes easily. The iteration for flows on trees is the same as that for paths with the derivative $\Sigma_{(i,j)\in t} D_{ij}'$ for a tree t used in place of the 1st derivative length of a path flow. Similarly the case of multiple root nodes is a straightforward extension. The optimality conditions and gradient projection iterations for different roots are decoupled.

5.29

The length of a path is

$$\frac{\partial D}{\partial x_p} = \sum_{(i,j) \text{ on } p} (D_{ij}' + c_{ww'} D_{ji}').$$

The optimality condition is

$x_p^* > 0 \quad \Rightarrow$ p is shortest over all paths of the same OD pair with respect to length of link (i,j) $= D_{ij}' + c_{ww'} D_{ji}'$.

The extension of the gradient projection method is straightforward using the expression for path lengths given above.

5.30

For simplicity we drop the subscripts. Let $Q(F) = a + bF + 0.5\, cF^2$ be the quadratic function where a, b, c are its unknown coefficients. Denote $D(F) = F/(C - F)$. The derivatives are

$$D'(F) = C/(C - F)^2, \quad D''(F) = 2C/(C - F)^3$$

and

$$Q'(F) = b + cF, \qquad Q''(F) = c.$$

We determine a, b, c via the condition that D and its derivatives should equal Q and its corresponding derivatives at $F = \rho C$. This leads to the equations

$$c = 2/(1 - \rho)^3 C^2$$

$$b + \rho cC = 1/(1 - \rho)^2 C$$

$$a + \rho bC + 0.5\, \rho^2 cC^2/2 = \rho/(1 - \rho),$$

which can be easily solved to give the values of a, b, c. D(F) is then replaced by the function

$\tilde{D}(F)$

which equals D(F) for $F \leq \rho C$ and equals Q(F) otherwise.

The last assertion of the problem follows from the fact that the necessary condition for the F_{ij}^* to minimize

$$\sum_{(i,j)} D_{ij}(F_{ij})$$

(i.e. the shortest path condition of Section 5.5) is also a sufficient condition for minimizing

$$\sum_{(i,j)} \tilde{D}_{ij}(F_{ij})$$

when $F_{ij} \leq \rho_{ij} C_{ij}$ for all (i,j).

5.31

(a) For every x and y we have (by Taylor's theorem)

$$\int_0^1 \nabla f(x + ty)'y dt = f(x + y) - f(x)$$

so by applying this formula for $y = \alpha\Delta x$ we obtain

$$\begin{aligned} f(x + \alpha\Delta x) &= f(x) + \nabla f(x)'(\alpha\Delta x) + \int_0^1 [\nabla f(x + t\alpha\Delta x) - \nabla f(x)]'(\alpha\Delta x) dt \\ &\leq f(x) + \alpha\nabla f(x)'\Delta x + \alpha\int_0^1 |\nabla f(x + t\alpha\Delta x) - \nabla f(x)|\,|\Delta x|\, dt \\ &\leq f(x) + \alpha\nabla f(x)'\Delta x + \frac{\alpha^2 L}{2}|\Delta x|^2 \qquad (1) \end{aligned}$$

which is the desired relation.

(b) Minimizing both sides of (1) over $\alpha \in [0, 1]$ we obtain

$$\min_{\alpha\in[0,1]} f(x + \alpha\Delta x) \le f(x) + \min_{\alpha\in[0,1]} \{\alpha\nabla f(x)'\Delta x + \frac{\alpha^2 L}{2}|\Delta x|^2\} \qquad (2)$$

Since $\nabla f(x)'\Delta x < 0$ the minimum on the right is attained for some $\alpha' > 0$. The unconstrained minimum is attained at the scalar α^* for which the derivative $\nabla f(x)'\Delta x + \alpha^* L\,|\Delta x|^2$ is zero or

$$\alpha^* = -\frac{\nabla f(x)'\Delta x}{L\,|\Delta x|^2}.$$

If $\alpha^* \ge 1$ or $\nabla f(x)'\Delta x + L\,|\Delta x|^2 < 0$ the minimum on the right side of (2) is attained for $\alpha'=1$, and we obtain

$$\min_{\alpha\in[0,1]} f(x + \alpha\Delta x) \le f(x) + \nabla f(x)'\Delta x + \frac{L}{2}|\Delta x|^2 \le \nabla f(x) + \frac{\nabla f(x)'\Delta x}{2} \qquad (3)$$

where the last inequality follows from the relation $\nabla f(x)'\Delta x + L\,|\Delta x|^2 < 0$.

If $\alpha^* < 1$ then the minimum over [0, 1] is attained for $\alpha' = \alpha^*$ and substitution in (2) yields

$$\min_{\alpha\in[0,1]} f(x + \alpha\Delta x) \le f(x) - \frac{|\nabla f(x)'\Delta x|^2}{L\,|\Delta x|^2} + \frac{|\nabla f(x)'\Delta x|^2}{L^2\,|\Delta x|^4}\,\frac{L\,|\Delta x|^2}{2}$$

$$= f(x) - \frac{|\nabla f(x)'\Delta x|^2}{2L\,|\Delta x|^2} \le f(x) - \frac{|\nabla f(x)'\Delta x|^2}{2LR^2}.$$

c) If $\{x^k\}$ has a limit point x^*, then since $\{f(x^k)\}$ is monotonically decreasing, we must have $f(x^k)\to f(x^*)$ which implies $\delta^k\to 0$. Therefore $\nabla f(x^k)'\Delta x^k\to 0$, and the result follows as stated in the hint.

5.32

(a) Applying the necessary condition for the projection problem with $x = x^k$ we obtain

$$s\nabla f(x^k)'(\bar{x}^k - x^k) \le -|\bar{x}^k - x^k|^2 \qquad (1)$$

Using the conclusion of part b) of Problem 5.31 we obtain

$$\min_{\alpha\in[0,1]} f(x^k + s\Delta x^k) \le f(x^k) + \delta^k$$

where

$$\Delta x^k = \bar{x}^k - x^k$$

and where, [using also (1)],

$$\delta^k \le -\frac{|\Delta x^k|^2}{2s} \qquad \text{if } \nabla f(x^k)'\Delta x^k + L\,|\Delta x^k|^2 < 0$$

$$\delta^k \le -\frac{|\Delta x^k|^4}{2s^2LR^2} \qquad \text{otherwise}$$

Therefore if $\{x^k\}$ has a limit point x^* we must have $f(x^k)\to f(x^*)$ and $\delta^k\to 0$. Therefore .

$$\Delta x^k \to 0, \qquad \{\bar{x}^k\} \to x^* ,$$

and by taking limit in the condition

$$[x^k - s\nabla f(x^k) - \bar{x}^k]'\,(x - \bar{x}^k) \le 0$$

for all $x\in X$ we obtain $\nabla f(x^*)'(x - x^*) \ge 0$ for all $x\in X$.

(b) Using (1) and part a) of Problem 5.31 we obtain (for $\alpha=1$)

$$f(x^{k+1}) \le f(x^k) + \nabla f(x^k)'\Delta x^k + \frac{L}{2}|\Delta x^k|^2$$

$$\le f(x^k) - \frac{|\Delta x^k|^2}{s} + \frac{L}{2}|\Delta x^k|^2$$

$$= f(x^k) - (\frac{1}{s} - \frac{L}{2})\,|\Delta x^k|^2$$

If $s < 2/L$ then the cost is reduced by at least a positive constant multiple of $|\Delta x^k|^2$ at the kth iteration. The result follows similarly as for part a).

5.33

Consider the change of variables $y = T^{-1}x$ or $x = Ty$. Then the problem can be written in terms of the y variables as

$$\begin{aligned} &\min\ f(Ty) \\ &\text{subject to } Ty \geq 0 \end{aligned}$$

or equivalently, because T is diagonal with positive diagonal elements,

$$\begin{aligned} &\min\ h(y) \\ &\text{subject to } y \geq 0 \end{aligned}$$

The second iteration in the problem is the gradient projection iteration for the second problem above. We have

$$\frac{\partial h(y)}{\partial y_i} = \sqrt{b_i}\,\frac{\partial f(x)}{\partial x_i}$$

Substituting this expression in the second iteration and multiplying throughout by $\sqrt{b_i}$ we obtain the first iteration of the problem. So the diagonally scaled version of the gradient projection iteration is the same as the ordinary version applied to a problem involving the diagonally scaled variables y_i.

5.34

(a) Since an arrival comes every τ time units, the system starts empty, and each arrival stays for H time units, we see that just after time H-τ there will be a total of H/τ arrivals. The first departure occurs at time H and at the same time an arrival occurs, which maintains the total number $N_1(t) + N_2(t)$ in the system at the level H/τ. Similarly, this number is maintained for all t.

(b) We first calculate N_1^* and N_2^*. The optimality condition is that the partial cost derivatives with respect to N_1 and N_2 are equal, so we have

$$\gamma_1 N_1^* = \gamma_2 N_2^*.$$

By combining this equation with the constraint

$$N_1^* + N_1^* = \frac{H}{\tau}, \tag{1}$$

we obtain

$$N_1^* = \frac{\gamma_2}{\gamma_1 + \gamma_2}\frac{H}{\tau}, \quad N_2^* = \frac{\gamma_1}{\gamma_1 + \gamma_2}\frac{H}{\tau}.$$

Define for all t

$$N(t) = N_1(t) + N_2(t), \tag{2}$$

$$N_1^*(t) = \frac{\gamma_2}{\gamma_1+\gamma_2} N(t), \tag{3}$$

$$N_2^*(t) = \frac{\gamma_1}{\gamma_1+\gamma_2} N(t), \tag{4}$$

and note that for $t > H$ we have

$$N(t) = N_1^* + N_2^* = \frac{H}{\tau}, \quad N_1^*(t) = N_1^*, N_2^*(t) = N_2^*. \tag{5}$$

The relation

$$\gamma_1 N_1(t) \le \gamma_2 N_2(t)$$

is equivalent to

$$\gamma_1 N_1(t) \le \gamma_2 (N(t)-N_1(t))$$

or

$$N_1(t) \le \frac{\gamma_2}{\gamma_1+\gamma_2} N(t) = N_1^*(t),$$

where the last equation follows by using Eq. (3). Thus, we have

$$\gamma_1 N_1(t) \le \gamma_2 N_2(t) \Leftrightarrow N_1(t) \le N_1^*(t), \tag{6}$$

and similarly

$$\gamma_2 N_2(t) \le \gamma_1 N_1(t) \Leftrightarrow N_2(t) \le N_2^*(t). \tag{7}$$

We will now prove by induction that all $k = 0,1,...$, and all $t \in [kT,(k+1)T)$, we have

$$\frac{N_1(t)-N_1^*(kT)}{N_1^*} \le \frac{\gamma_1+\gamma_2}{\gamma_1} \frac{T}{H}, \tag{8}$$

$$\frac{N_2(t)-N_2^*(kT)}{N_2^*} \le \frac{\gamma_1+\gamma_2}{\gamma_1} \frac{T}{H}. \tag{9}$$

We claim that these relations are true for $k = 0$. Indeed we have

$$N_1^*(0) = N_1(0) = N_2(0) = 0,$$

so by the rules of the algorithm, all VCs arriving in the interval [0,T] will be assigned to link 1. Since the number of these VCs is at most T/τ, we have

$$N_1(t) \le N_1^*(0) + \frac{T}{\tau}. \tag{10}$$

By using Eq. (1), we see that

$$\frac{T}{\tau} = \frac{\gamma_1+\gamma_2}{\gamma_2}\frac{T}{H}N_1^*,$$

so Eq. (10) yields

$$\frac{N_1(t)-N_1^*(0)}{N_1^*} \le \frac{\gamma_1+\gamma_2}{\gamma_2}\frac{T}{H},$$

thus verifying Eq. (8) for k=0. Eq. (9) holds for k=0 since $N_2^*(t) = 0$.

We will now show that Eqs. (8) and (9) hold for the typical k, assuming they hold for all preceding k.

We assume without loss of generality that $\gamma_1 N_1(kT) \le \gamma_2 N_2(kT)$ or equivalently by Eq. (6),

$$N_1(kT) \le N_1^*(kT). \tag{11}$$

Then by the rules of the algorithm, all VCs arriving in the interval [kT, (k+ 1)T) will be assigned to link 1. Since the number of these VC's is at most T/t, we have

$$N_1(t) \le N_1^*(kT) + \frac{T}{\tau}, \quad \forall\, t \in [kT,(k+1)T). \tag{12}$$

By using Eq. (1), we see that

$$\frac{T}{\tau} = \frac{\gamma_1+\gamma_1}{\gamma_2}\frac{T}{H}N_1^*,$$

so Eq. (12) yields

$$\frac{N_1(t)-N_1^*(kT)}{N_1^*} \le \frac{\gamma_1+\gamma_2}{\gamma_2}\frac{T}{H}, \tag{13}$$

proving Eq. (8) for the typical k.

In view of Eq. (11), we also have

$$N_2(kT) \ge N_2^*(kT)$$

as well as

$$N_2(t) \le N_2(kT), \quad \forall\, t \in [kT,(k+1)T).$$

Therefore we have

$$N_2(T) - N_2^*(kT) \geq N_2(t) - N_2^*(kT), \quad \forall\, t \in [kT,(k+1)T)$$

and Eq. (9) holds in view of the induction hypothesis. Thus the induction proof of Eqs. (8) and (9) is complete.

From Eqs. (8) and (9), since $N_1^*(t) = N_1^*$, $N_2^*(t) = N_2^*$ for $t > H$, we have for all $t > H$

$$\frac{N_1(t) - N_1^*}{N_1^*} \leq \frac{\gamma_1 + \gamma_2}{\gamma_2} \frac{T}{H}, \tag{14}$$

$$\frac{N_2(t) - N_2^*}{N_2^*} \leq \frac{\gamma_1 + \gamma_2}{\gamma_1} \frac{T}{H}. \tag{15}$$

Since $N_1(t) - N_1^* = N_2 = N_2(t)$, from Eq. (14) we obtain

$$N_2^* - N_2(t) \leq \frac{\gamma_1+\gamma_2}{\gamma_2} \frac{T}{H} N_1^* = \frac{\gamma_1+\gamma_2}{\gamma_1} \frac{T}{H} N_2^*$$

or equivalently

$$\frac{N_2^* - N_2(t)}{N_2^*} \leq \frac{\gamma_1+\gamma_2}{\gamma_1} \frac{T}{H}.$$

Combining this relation with Eq. (15), we obtain

$$\frac{|N_2(t) - N_2^*|}{N_2^*} \leq \frac{\gamma_1+\gamma_2}{\gamma_1} \frac{T}{H},$$

and we similarly prove the remaining relation

$$\frac{|N_1(t) - N_1^*|}{N_1^*} \leq \frac{\gamma_1+\gamma_2}{\gamma_2} \frac{T}{H}.$$

5.35

To make the protocol workable it is essential to number sequentially the exploratory packets. (This is needed, for example, in order to avoid confusion between transmitter and receiver regarding two distinct VC setup requests. There are also other reasons for this as will be seen shortly.) There should be a separate sequence number for each origin - destination (OD) pair, and it will be assumed that the sequence number field is so large that wraparound never occurs in the absence of memory or transmission errors.

Indefinite circulation can be avoided if each node relays a received exploratory packet only to neghbor nodes not yet visited by the packet (i.e., the nodes that are not stamped on the

packet). This rule guarantees that the exploratory packet will travel on all possible routes from origin to destination that contain no loops. Thus the destination will receive one copy of the exploratory packet for each distinct route that was up (roughly) at the time the exploratory packet was being flooded through the network. This gives the greatest choice of routes to the receiver, but creates a problem with excessive number of packets being communicated.

To limit circulation of exploratory packets a number of schemes is possible provided each node stores the largest sequence number received for every OD pair. One possibility is to flood the exploratory packet to all neighbors (except the one from which the packet was received) only once - the first time the packet is received. Another possibility is to check the number of nodes the exploratory packet has visited and to relay the packet only if either it has a larger sequence number than the number of the packet latest flooded for the same OD pair, or if it has visited fewer nodes than the previous packet with the same sequence number. This last scheme guarantees that an exploratory packet will be received via a route with minimal number of nodes.

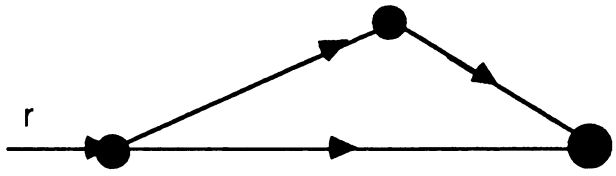
There is a problem if the receiver's response to a VC request never reaches the transmitter because of link or node failures along the chosen route. This can be handled by the transmitter using a time out, and transmitting a new exploratory packet for the same VC carrying, however, a **new** sequence number. Note that the transmitter should have ultimate responsibility for accepting or rejecting a VC connection, and the receiver is passive in this regard.

Finally if a node can crash, and not remember the sequence numbers last used, a scheme such as the one of Section 5.3.2 can be used.

5.36

(a) For small values of r_w the first derivative length of a path is almost equal to D'(0) times the number of links of the path. Therefore a path that does not have minimum number of links cannot be shortest and therefore cannot carry flow at the optimum.

(b) Consider the single OD pair network shown below:



Each link has cost function $D(F) = F + 0.5F^2$. Then, by applying the shortest path condition, it is seen that for $r \leq 1$ the optimal routing is to send all flow on the one-link path, but for $r > 1$ the optimal routing is to send $(1 + 2r)/3$ on the one-link path and $(r - 1)/3$ on the two-link path.

5.37

The origin of each OD pair w sends a message carrying the value of r_w along the shortest path for the current iteration. Each link (i,j) accumulates the shortest path link flow

$$\bar{\bar{F}}_{ij}$$

and sends it together with F_{ij}, D'_{ij} and D''_{ij} to all origins. All origins can then calculate the stepsize α^* of (5.69) and change the path flows x_p according to the iteration

$$x_p := x_p + \alpha^*(\bar{x}_p - x_p),$$

where

$$\bar{x}_p = r_w \text{ if p is the shortest path and } \bar{x}_p = 0 \text{ otherwise.}$$

5.38

(a) The average round trip delays on paths 1 and 2 corresponding to x are $T_1(x)$ and $T_2(x)$. These are used to estimate the average number of unacknowledged packets on paths 1 and 2 according to

$$\bar{N}_1 = \bar{x}_1 T_1(x), \qquad \bar{N}_2 = \bar{x}_2 T_2(x).$$

The actual average number on the two paths are

$$\tilde{N}_1 = \tilde{x}_1 T_1(\tilde{x}), \qquad \tilde{N}_2 = \tilde{x}_2 T_2(\tilde{x}).$$

Therefore if the routing method used equalizes the ratios

$$\frac{N_i}{\tilde{N}_i}$$

the relation of part (a) must hold.

(b) We assume with no loss of generality

$$x_1 + x_2 = \tilde{x}_1 + \tilde{x}_2 = r$$

$$\tilde{x}_1 < x_1 \qquad \tilde{x}_2 > x_2.$$

Therefore, by the hypothesis of part (b), we must have

$$\tilde{x}_1 < x_1 \quad \Rightarrow \quad T_1(x) > T_1(\tilde{x}) \quad \text{and} \quad T_2(x) < T_2(\tilde{x}).$$

From the relation of part (a) we have

$$\frac{\tilde{x}_1}{\bar{x}_1} = \frac{\tilde{x}_2}{\bar{x}_2}\frac{T_1(x)}{T_1(\tilde{x})}\frac{T_2(\tilde{x})}{T_2(x)} > \frac{\tilde{x}_2}{\bar{x}_2}$$

Therefore $\tilde{x}_1 > \bar{x}_1$ and, since $x_1 + x_2 = \bar{x}_1 + \bar{x}_2$, we must have $\tilde{x}_2 < \bar{x}_2$.

(c) The vectors x, $\tilde{x}$, and $\bar{x}$ lie on the same line of the 2-dimensional plane, and $\tilde{x}$ lies between x and $\bar{x}$. We now argue that a convex function that has a lower value at $\bar{x}$ than at x must also have a lower value at $\tilde{x}$ than at x

5.39

See the references cited.

5.40

(a) Let D_i^* be the correct shortest distance to 1 from each node i.

Claim 1: Eventually, $D_i = D_i^*$ for all nodes i.

Proof: Assume that $D_i = D_i^*$ for all nodes i that have a k hop shortest path to node 1. Consider a node j that has a k+1 hop shortest path to node 1 of the form (j,i,...,1). This node will receive $D_i^* + d_{ij}$ from i and therefore will have $D_j = D_j^*$. The assumption is clearly true for k = 0, and by induction it is true for all k.

Claim 2: A finite time after claim 1 is satisfied, node 1 receives an ACK from each of its neighbors.

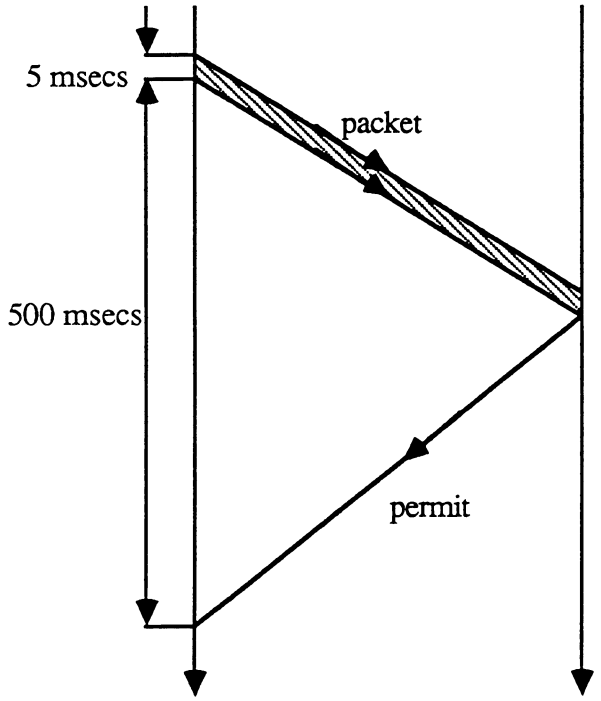
Proof: When claim 1 is satisfied, the set of arcs connecting each node $i \neq 1$ with its predecessor forms a directed shortest path spanning tree rooted at node 1. (For a proof of this, see the discussion in the text following Bellman's equation.) Consider a leaf node j on the tree. Each of its neighbors must send an ACK in response to its last distance measurement. Therefore, j sends an ACK to its predecessor. By induction on the depth of the tree, node 1 eventually receives an ACK from each neighbor for which it is the predecessor. Node 1 clearly receives ACK's from its other neighbors as well.

(b) The main advantage of this algorithm over the one in Section 5.2.4 is that node 1 is notified of termination, and hence knows when its estimates are correct. The main disadvantage is that this algorithm requires a specific initial condition at each node, making it difficult to restart when a distance measurement or link status changes.

CHAPTER 6 SOLUTIONS

6.1

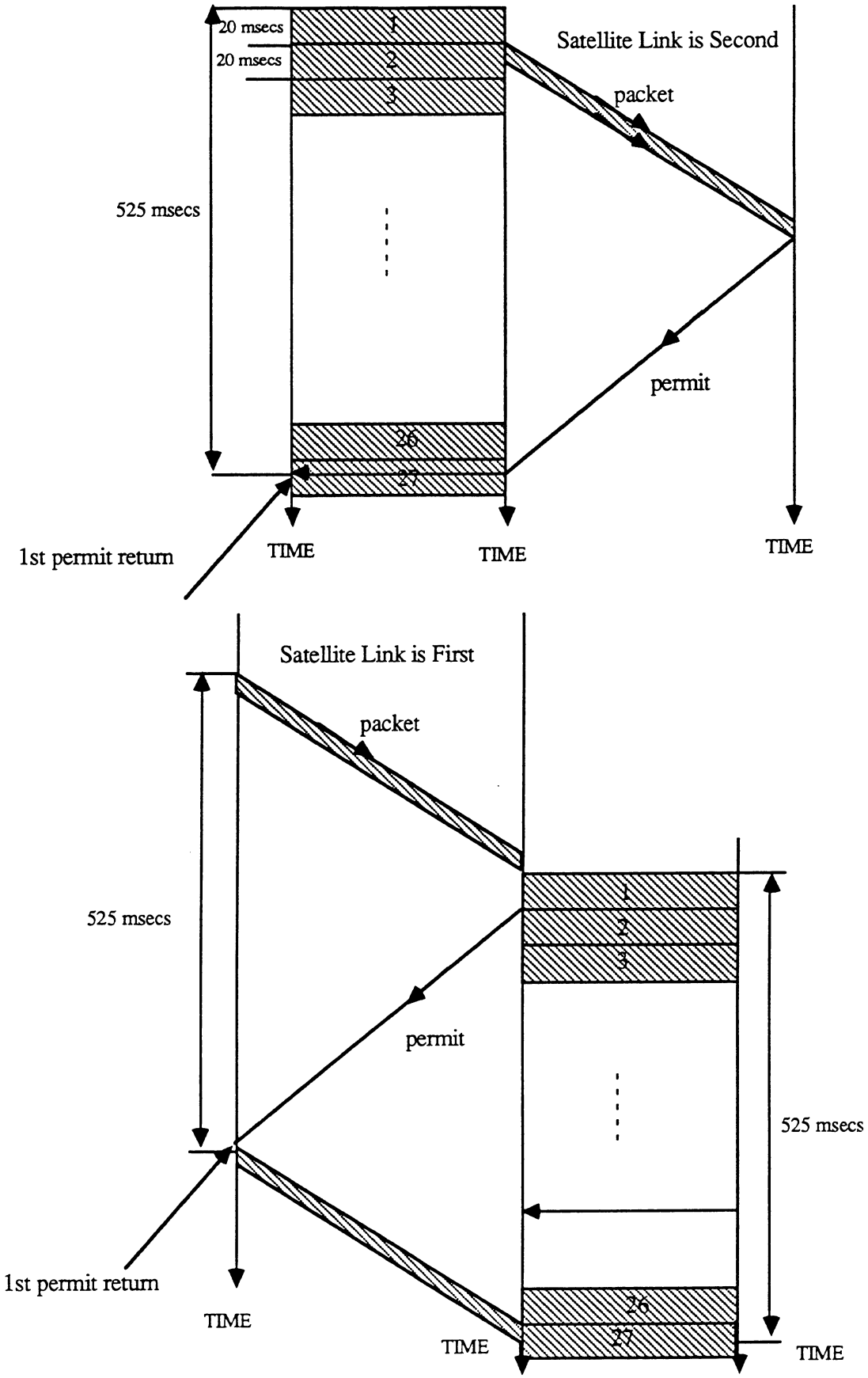
The window size required is at least 101 packets since the delay from the time a packet begins transmission to the time its permit returns is 505 msecs. If the transmission time of the permit or the time the permit is delayed at the receiver are nonzero then the window size should be accordingly larger. See the figure below.



Total window size needed = 101 packets
(assuming zero transmission time for the permit

6.2

The Maximum throughput is limited by the transmission rate of the terrestrial link (50 packets/sec). The timing diagrams below show that an end-to-end window size of $\lceil 525/20 \rceil$ = 27 packets is required to achieve full speed transmission in both cases where the terrestrial link comes before and after the satellite link.

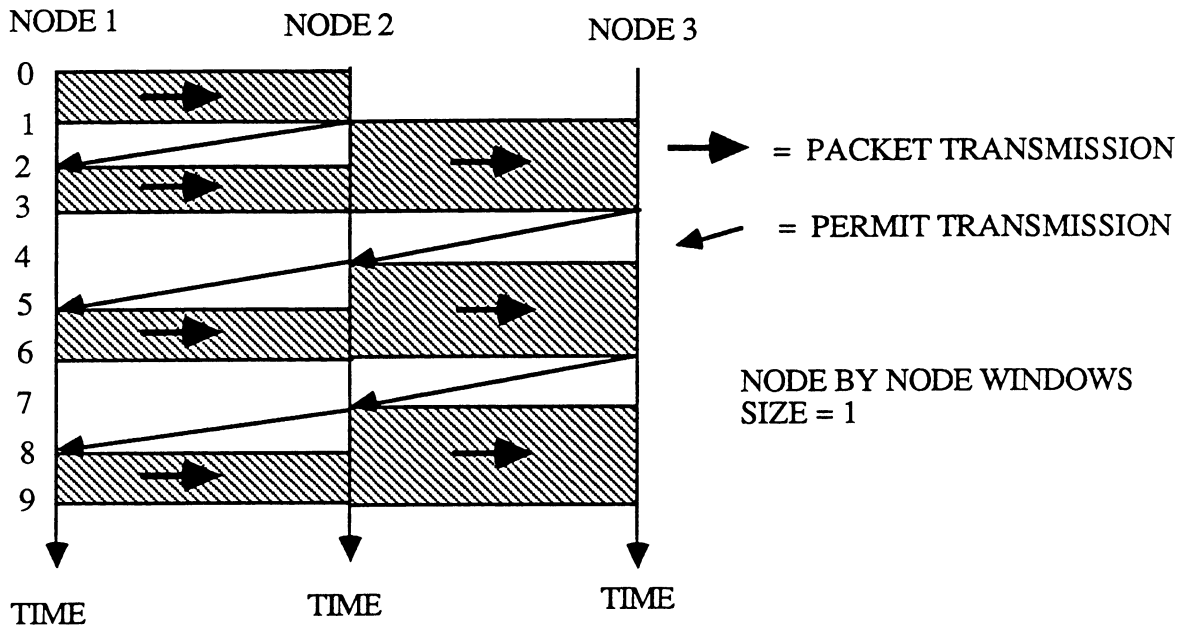

20 msecs
20 msecs
1
2
3
Satellite Link is Second
packet
525 msecs
permit
26
27
1st permit return
TIME
TIME
TIME
Satellite Link is First
packet
525 msecs
1
2
3
permit
525 msecs
1st permit return
TIME
TIME
26
27
TIME

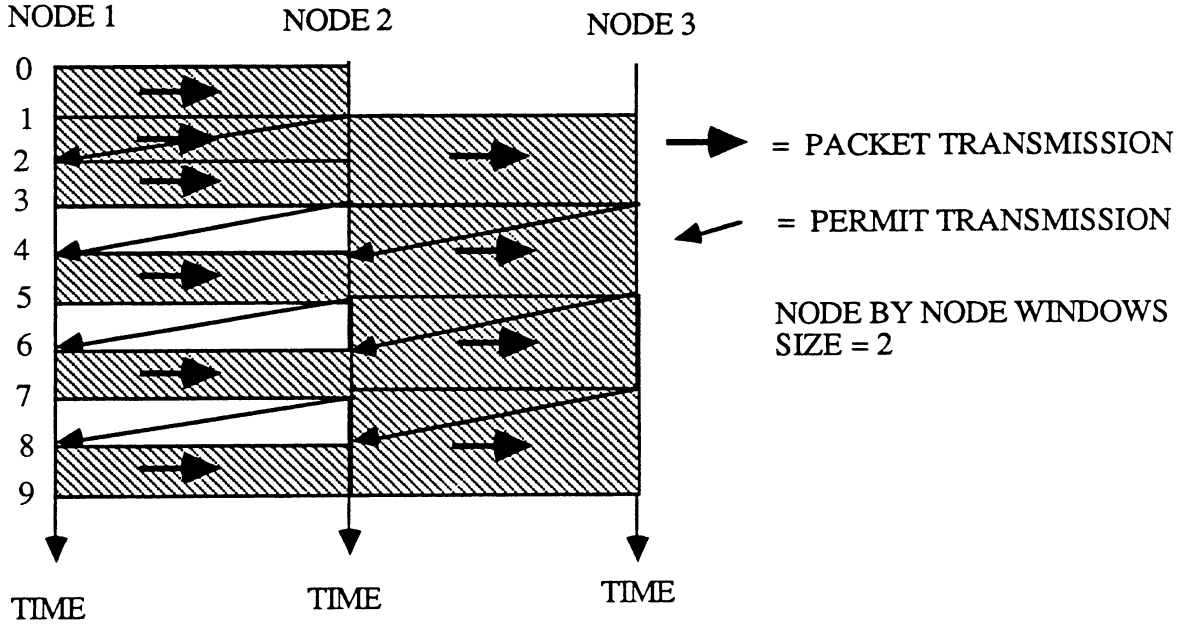
6.3

We need a window of at least 1 for the terrestrial link, and a window of at least 26 in the satellite link. If the permit transmission delays are nonzero, larger windows are required for nonstop transmission to be possible.

6.4

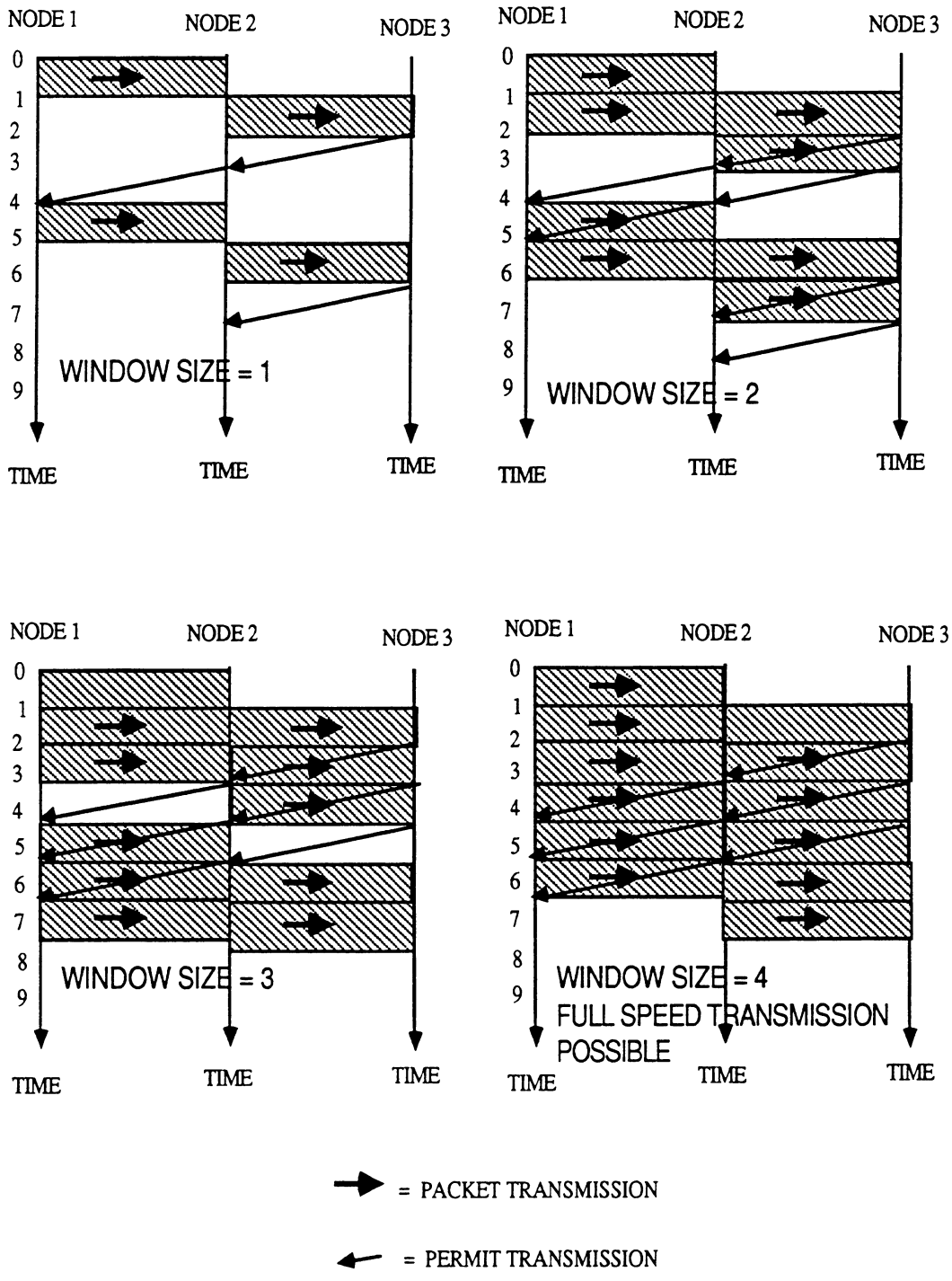
See the following timing diagrams:





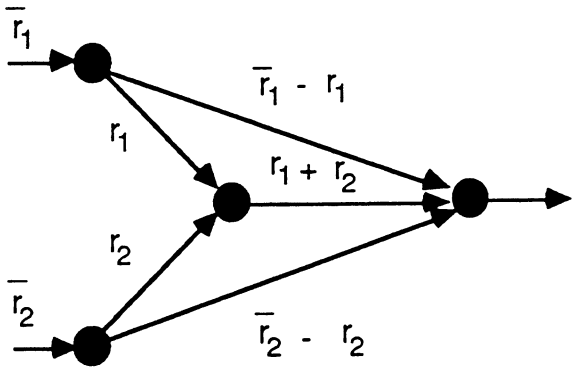
6.5

The following example shows that the alternative scheme requires a larger window, and therefore also more memory. Note also that if link (i, i+1) is a satellite link, the window of link (i-1, i) must also be large if a permit is sent by i after an ACK is received from (i+1). This is not so if a permit can be sent upon delivery of a packet to the DLC.



6.6

The augmented network including the overflow links for the two OD pairs is shown in the figure below:



Consider three cases:

a) $r_2 < \bar{r}_2$, $r_1 < \bar{r}_1$, b) $r_2 = \bar{r}_2$, $r_1 < \bar{r}_1$, c) $r_2 = \bar{r}_2$, $r_1 = \bar{r}_1$.

Case a): By symmetry here $r_1 = r_2$. The optimality condition is

$$2r_1 + 2(r_1 + r_2) = a/r_1^2$$

which implies $r_1 = r_2 = (a/6)^{1/3}$. This case is in effect for

$$(\alpha/6)^{1/3} < \bar{r}_2 = 1$$

which yields $a < 6$.

Case b): Here the optimality condition is

$$2r_1 + 2(r_1 + \bar{r}_2) = \frac{a}{r_1^2}$$

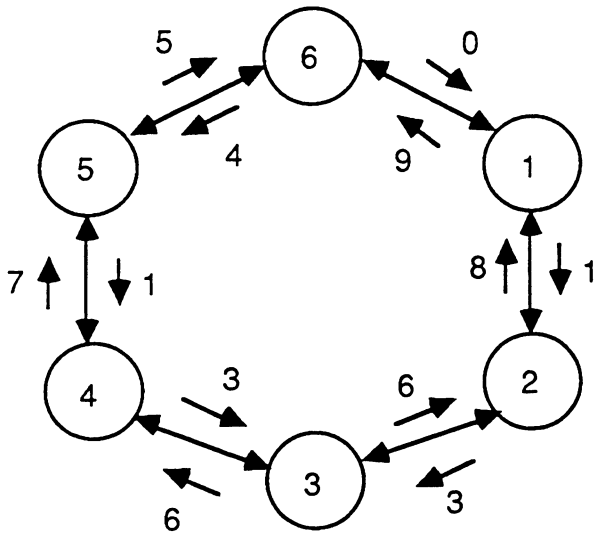
or equivalently $4r_1^3 + 2r_1^2 = a$ which can be solved for r_1. This case holds for

$$6 \le a < 4\bar{r}_1^3 + 2\bar{r}_1^2 = 4{,}200$$

Case c): This case where no flow control is exercised holds for $a \ge 4200$.

6.7

The number of sessions for each network link are shown in the figure below:



NUMBER OF SESSIONS ON EACH LINK SHOWN NEXT TO THE LINK

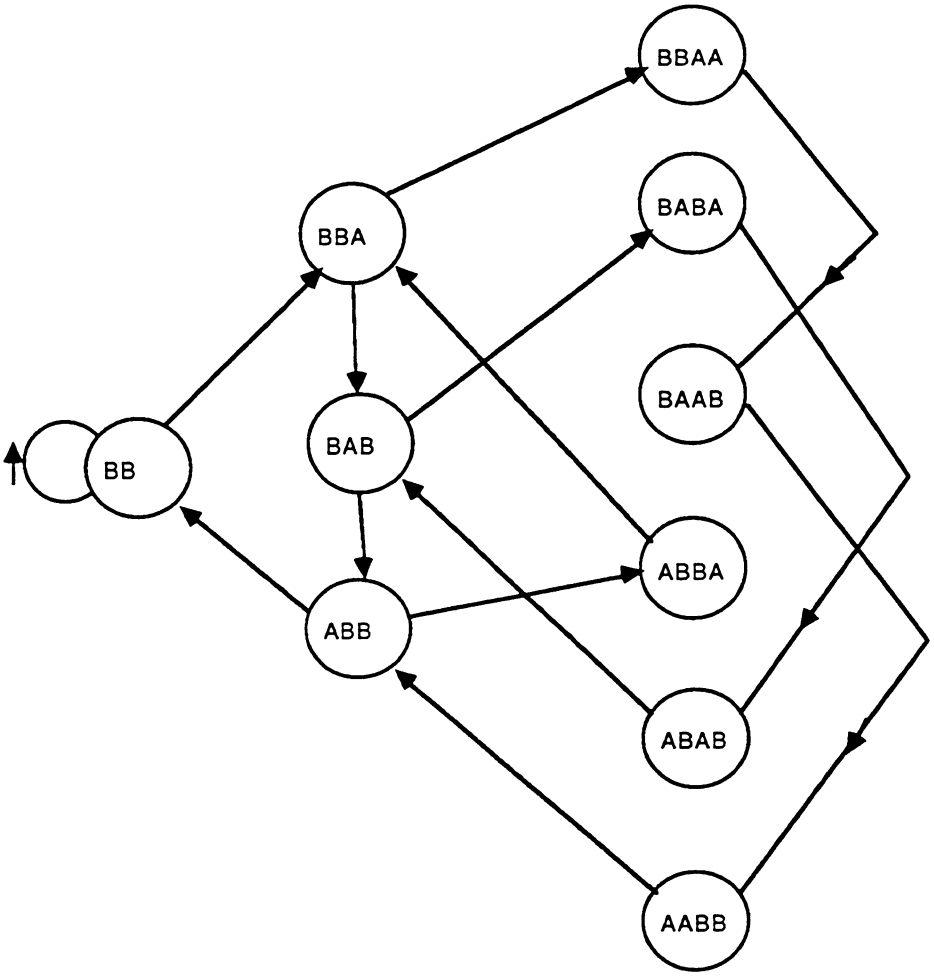
There are nine counterclockwise sessions and they all go through arc (1,6). At Step 1 this arc is saturated, and the rate of all counterclockwise sessions is fixed at 1/9. At Step 2 of the algorithm arc (4,5) (the one that carries the most clockwise seesions) is saturated and the rate of the sessions going through it is fixed at 1/7. There are only two sessions (5 to 6 and 3 to 4 clockwise) that don't go through this arc. At Step 3 the arc (3,4) becomes saturated and the rate of session 3 to 4 (clockwise) is fixed at 2/7. Finally at Step 4 of the algorithm arc (5,6) becomes saturated and the rate of session 5 to 6 gets fixed at 3/7. The solution is now complete since the rate of all sessions has been fixed.

6.8

For each session p add an artificial node p and a link from p to the entry node of the session having capacity b_p.

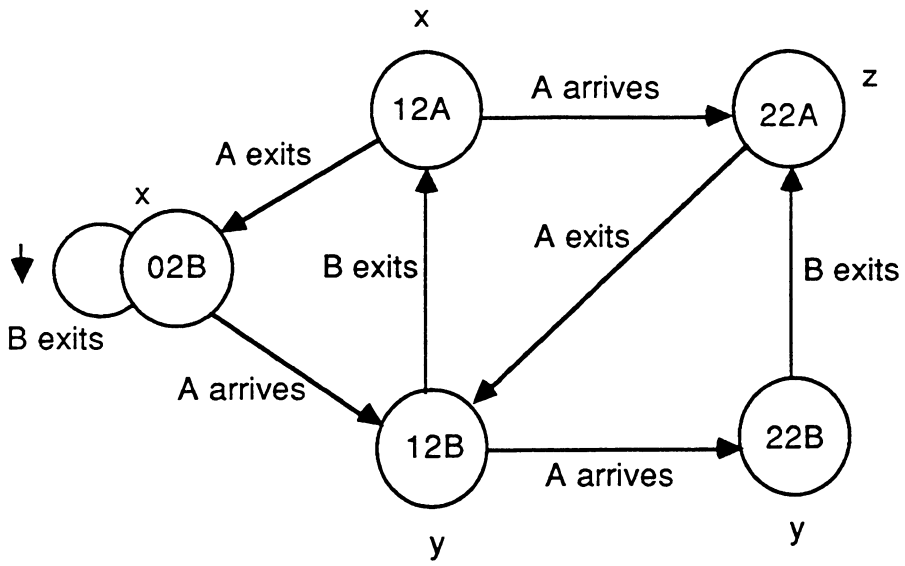
6.9

(a) The Markov chain is shown in the figure below. The state gives the session and the order of arrival (and also transmission) of the packets in link L. Note that because the permit delay of the session B packets is zero, there are always two packets from B in link L. All transition probabilities for a small interval δ are $\delta + o(\delta)$ corresponding to a unity transition rate. To see that the steady state occupancy probability is 1/10 for every state verify that these probabilities satisfy the global balance equations. This is seen from the fact that there are as many transition arcs coming into each state as there are going out. The total steady state throughput is one packet/sec which is the transmission rate of link L. The steady state throughput from session A is 0.4 because only 4 out of 10 states correspond to transmission of a packet from A, while the remaining states correspond to transmission of packets from B.



(b) The state transition diagram is given below. The state is the triple

(number in L from A, number in L from B, session currently transmitting)



All transitions have unity rate. Applying the global balance equations to states (02B), and (22B) we see that

steady state prob. of (02B) = steady state prob. of (12A) = x

steady state prob. of (12B) = steady state prob. of (22A) = y

Let

z = steady state prob. of (22A)

Applying the global balance equations to nodes (12B) and (22A) we obtain

$x + z = 2y, \qquad z = x + y$

Combining these equations with $2x + 2y + z = 1$ we obtain

$x = 1/9, \qquad y = 2/9, \qquad z = 1/3.$

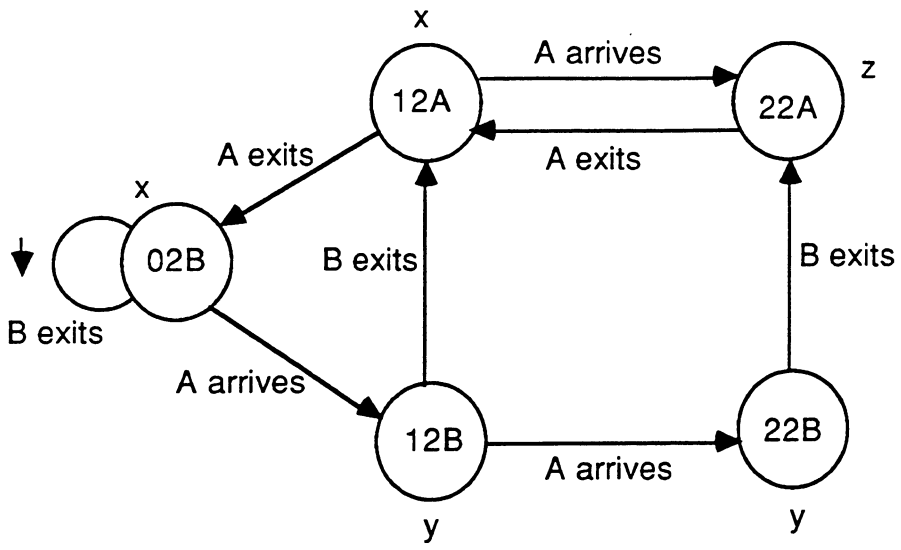
A transmission of a B packet is completed from states (02B), (12B), (22B) with rate 1, while a transmission of an A packet is completed from states (12A), and (22A) with rate 1. Therefore the ratio of throughputs for B and A is

$(x + 2y)/(x + z) = 5/4$

The throughput of the entire system is 1 since there is always a packet from B to transmit on link L. Therefore the throughput for A is 4/9 and the throughput for B is 5/9. Note that this throughput allocation (corresponding to the round robin discipline) is more fair than for the case of part a) (first - come first - serve discipline).

(c) The state transition diagram for the nonpreemptive priority case is given below. The state is again the triple

(number in L from A, number in L from B, session currently transmitting)



All transitions have unity rate. Similarly as in case b) we see that

steady state prob. of (02B) = steady state prob. of (12A) = x

steady state prob. of (12B) = steady state prob. of (22A) = y

Let

z = steady state prob. of (22A)

Applying the global balance equations to nodes (12B) and (22A) we obtain

$x = 2y, \qquad z = x + y$

Combining these equations with $2x + 2y + z = 1$ we obtain

$x = 2/9, \qquad y = 1/9, \qquad z = 1/3.$

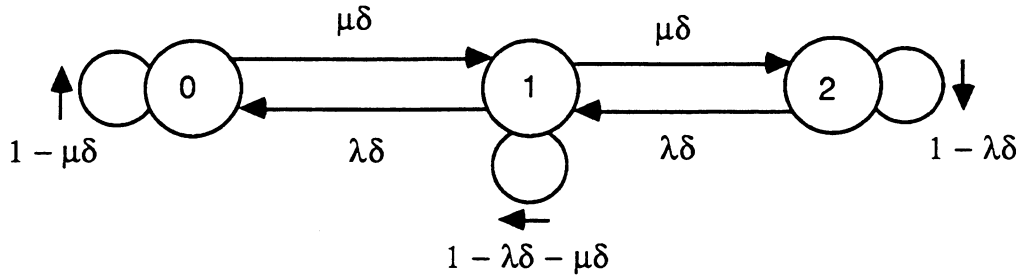
A transmission of a B packet is completed from states (02B), (12B), (22B) with rate 1, while a transmission of an A packet is completed from states (12A), and (22A) with rate 1. Therefore the ratio of throughputs for B and A is

$(x + 2y)/(x + z) = 4/5$

Again the throughput of the entire system is 1. Therefore the throughput for A is 5/9 and the throughput for B is 4/9. As expected, the throughput of A increases over the previous cases since A is now given priority.

6.10

(a) Let state i (i = 0, 1, 2) denote the number of permits available at the source. Then, for small δ the Markov chain for successive δ intervals is given below



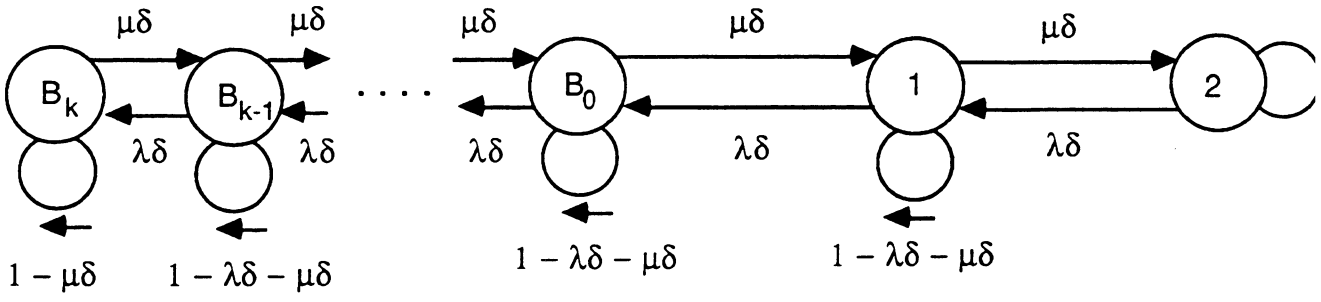
(b) The steady state probabilities are found from $P_0 + P_1 + P_2 = 1$, $P_0\mu\delta = P_1\lambda\delta$, and $P_1\mu\delta = P_2\lambda\delta$. The probability that an arriving packet will be discarded is

$$P_0 = \frac{1}{1 + \frac{\mu}{\lambda} + (\frac{\mu}{\lambda})^2}$$

(c) The probability of a discard increases as delay from source to node (and/or node to source) is taken into account. This is easy to see as the delay becomes very large, since then the permits are very slow to return to the source and almost all source packets are rejected. Analytically, this is not hard to establish for exponentially distributed delays.

(d) A buffer at the source simply adds states to the Markov chain above. Let B_i be the state where there are i packets in the buffer but there are no available permits. The corresponding Markov chain is shown in the figure below. It is seen from the figure that the probability of an arriving packet finding the buffer full is

$$P_k = \frac{1}{1 + \frac{\mu}{\lambda} + (\frac{\mu}{\lambda})^2 + \ldots + (\frac{\mu}{\lambda})^{\kappa+2}}$$



6.11

(a) Suppose that the protocol has worked correctly up to a time t (e.g. the initial time), and at that time the source knows that the destination is awaiting a packet n. Then, for a window size W, the next acknowledgement to be received by the source should distinguish between whether the destination is awaiting packet n, n+1, . . . , n+W-1 or n+W. If the modulus m exceeds W then these possibilities can be distinguished. If $m \leq W$ then the source can be confused. Thus the protocol works correctly for $W \leq m - 1$, and incorrectly for $W \geq m$.

(b) The window size W is no longer restricted by the modulus m. The source can determine the number of the packet being awaited by the destination by counting the number of modulo numbers contained in the acknowledgements.

(c) The destination does not need to know the window size W to perform its part of the protocol. Therefore, the source can change the window size without prior agreement from the destination. However, for the scheme in (a), the window size is still restricted by the modulus.

(d) If the destination delays acknowledging the last packet it has received until it receives the next packet, 1 unit of the window size W is held at the destination and from then on the number of packets sent by the source but not yet received by the destination can be W - 1 at most. Thus the window size can, effectively, be reduced.

6.12

(a) In the memory of node i at any one time there can be at most 2 packets destined for node j that have arrived at node i via a particular incoming link of i, or from a particular external site connected to node i. If the number of links coming into node i, and the number of sites connected to node i that can carry or send packets destined for node j is $n_{j,i}$, then in order to guarantee that every arriving packet for j can be placed in a buffer, node i must have $2n_{j,i}$ packet buffers for packets destined for j.

(b) This scheme constrains all packets with the same destination to use the same link outgoing from a node at every node in the network, thus using paths from a spanning tree. Therefore, not all routings are permissible. Also if the routing changes the buffering scheme ought to be adjusted. Furthermore the scheme is unfair to sessions that go to unpopular destinations. The scheme has the advantage that the required buffering at node i is determined by the degrees $n_{j,i}$ of the node rather than by the potentially large number of sessions that use the node.

6.13

Three rules are needed:

a) At the time a VC is established no permits should become available to the transmitting node of each link on the VC's path.
b) A permit that cannot be used by a VC because no packet is available to send should be returned to the receiver (perhaps after a time out).

c) A node should keep track of the number of outstanding permits it has sent to the transmitter nodes at the other end of its incoming links. The strategy should ensure that this number times the maximum packet length should not exceed the amount of free storage available at the node.

6.14

For full speed transmission the window size W of each session must satisfy $WX \geq d$, where X is the packet transmission time at the origin node of the session and d is the round trip delay. If the trnsmission capacity is increased by a factor K, X will be decreased by a factor K. If propagation delay is dominated by transmission delay, d will also be decreased by a factor K, so the window size of each session should not change. However, the total window size will increase by a factor K, since the total number of sessions is increased by a factor K.

If transmission delay is dominated by propagation delay, d will not change, so the window size of each session should increase by a factor K. The total window size will then increase by a factor K^2, since the total number of sessions is increased by a factor K.

6.15

The formulation as a Markov chain is similar to the case considered in Section 3.1. The states and their interpretation does not change. In particular, the states are 0,1,..., and for i = 0,1,...,W, state i corresponds to W-i permits available and no packets without permits waiting. The states i = W+1, W+2,..., correspond to i-W packets without permits waiting and no permits available. However, the state transitions occur at the times 0, W/r, 2W/r,..., just after a set of permits arrives. The probability of k packets arriving in W/r seconds are

$$a_k = \frac{e^{-\lambda W/r}(\lambda W/r)^k}{k!}.$$

The transition probabilities are as follows:

$$P_{io} = \begin{cases} \sum_{n=0}^{W-i} a_n & \text{if } i \leq W\text{-}1 \\ a_{i-W} & \text{otherwise} \end{cases}$$

$$P_{oi} = \begin{cases} \sum_{n=0}^{W-i} a_n & \text{if } i \leq W\text{-}1 \\ a_{i-W} & \text{otherwise} \end{cases}$$

and for $i \neq 0$ and $j \neq 0$,

$$P_{ji} = \begin{cases} a_{i-j+W} & \text{if } j \leq i+W \\ 0 & \text{otherwise} \end{cases}$$

6.16

We assume here that the desired OD pair input rates are known, and that each node knows all link cost functions, and its own input rate penalty function. The simplest distributed scheme that can be used is for the links to broadcast to all nodes the values of their total flows (using a flooding scheme for example). Then each node can execute the gradient projection iteration and change accordingly the values of its path flows and the flow on its own overflow link.

6.17

Suppose there are three nodes 1, 2, 3, and two links (1,2) and (2,3), with capacities 2 and 4, respectively. Consider three sessions with paths

$$p_1 = (1,2), \qquad p_2 = (2,3), \qquad p_3 = (1,2,3).$$

The max-min fair rate is

$$r_1 = 1, \qquad r_2 = 3, \qquad r_3 = 1.$$

If session 1 is eliminated, the new max-min fair rate is

$$r_2 = 2, \qquad r_3 = 2.$$

If the capacity of link (1,2) is increased to 4 units, the new max-min fair rate is

$$r_1 = 2, \qquad r_2 = 2, \qquad r_3 = 2.$$

Thus by reducing the load of the network either by eliminating sessions or by increasing some link capacities the **minimal** rate will not decrease, but not necessarily all the rates.

6.18

See the second reference cited.

6.19

(a) Suppose i is backlogged in some interval $[\tau,t]$. Then for each j,

$$\frac{T_i^{l_i}(\tau,t)}{T_j^{l_i}(\tau,t)} \geq \frac{r_i}{r_j}.$$

Summing over j

$$T_i^{l_i}(\tau,t) \geq \frac{r_i(t-\tau)}{\rho(l_i)} \qquad (*)$$

Thus a session i backlog of size q is always cleared in q/r_i time units. Now letting $Q_i(t)$ be the size of the backlog at time t, we obtain

$$Q_i(t) = A_i(\tau,t) - T_i^{l_i}(\tau,t) \leq W_i + \left(r_i - \frac{r_i}{\rho(l_i)}\right)(t-\tau) < W_i$$

where the last inequality follows from the assumption $\rho(l_i) < 1$. Finally, since session i backlogs are cleared at a rate greater than r_i, it follows that a bit is never backlogged for more than W_i/r_i time units.

(b) Since Eq. (6.11) holds at each link it follows that (*) must hold as well. Thus the guaranteed backlog clearing rate over the session i route is

$$g_i = \min_{l \in L} \frac{r_i}{\rho(l)} = \frac{r_i}{\rho_{max}^i},$$

where L is the set of links traversed by i. Now if session i is backlogged in at least one link during the interval $[\tau,t]$, then since we are ignoring propagation delay and the traffic is perfectly pipelined, it follows that

$$T_i^{l_i}(\tau,t) > \frac{r_i(t-\tau)}{\rho_{max}^i}.$$

The bounds on delay and backlog follow directly from this fact.

(c) This problem can be solved analogously to parts (a) and (b). The key observation is that when $g_i \geq r_i$, we have at link l_i (the first link traversed by session i)

$$T_i^{l_i}(\tau,t) \geq g_i(t-\tau)$$

during all intervals $[\tau,t]$ that session i is backlogged at link l_i. By reasoning similar to part (b) we see that this property holds at all links traversed by session i. Thus there are never more than W_i bits in the network, and no bit spends more than W_i/g_i time units in the network.